

Assignment – 4

Version Control System setup and usage using GIT

Date : 23-02-2019

Aim

Version Control System setup and usage using GIT. Try the following features

1. Creating a repository
2. Checking out a repository
3. Adding content to the repository
4. Committing the data to a repository
5. Updating the local copy
6. Comparing different revisions
7. Revert
8. Conflicts and a conflict Resolution

GIT

Git is the most commonly used version control system today and is quickly becoming the standard for version control. Git is a distributed version control system, meaning your local copy of code is a complete version control repository. These fully-functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server. This paradigm differs from centralized version control where clients must synchronize code with a server before creating new versions of code.

Git-Hub

GitHub is a Web-based Git version control repository hosting service. It provides all of the distributed version control and source code management (SCM) functionalities of Git while topping it with a few of its own features. It is a heaven for the developers where they can store their projects and get connected with like-minded people.

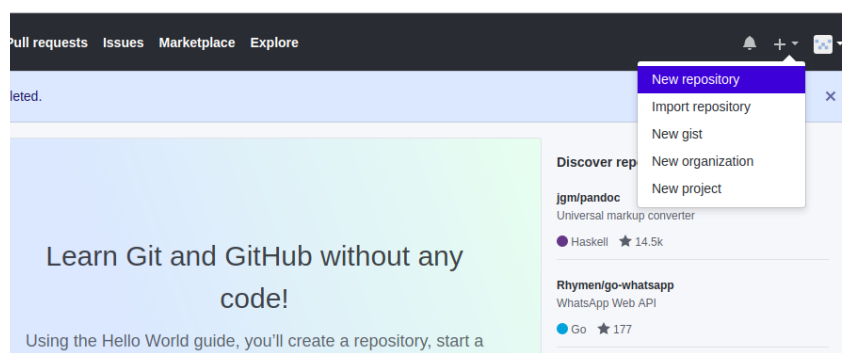
1. Creating a repository

By using GitHub - GitHub is a code hosting platform for version control and collaboration

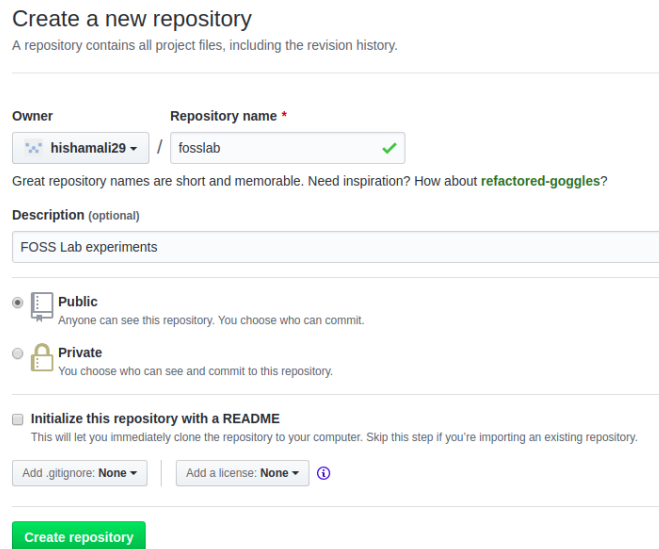
#Create a new repository on GitHub.

Steps

(a) In github profile, on the upper right corner click '+' and then select New repository.



(b) Write a name for repository, write a short description about repository and click create repository.



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history.' Below this, there are two input fields: 'Owner' with a dropdown menu showing 'hishamali29' and 'Repository name' with a text input 'fosslab' and a green checkmark. A hint text says 'Great repository names are short and memorable. Need inspiration? How about [refactored-goggles?](#)'. Below that is a 'Description (optional)' text area containing 'FOSS Lab experiments'. There are two radio button options: 'Public' (selected) with the description 'Anyone can see this repository. You choose who can commit.' and 'Private' with 'You choose who can see and commit to this repository.' There is also an unchecked checkbox 'Initialize this repository with a README' with a subtext 'This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.' At the bottom, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None' with an information icon. A green 'Create repository' button is at the very bottom.

2. Checking out a repository

Checkout is the act of switching between different versions of a target entity. 'git checkout' command lets us navigate between the branches created by git branch

#To create a new branch and change into a that branch
Command : git checkout -b <brach_name>

```
hishamali@ideapad-330:~/Public$ git checkout -b new_branch
Switched to a new branch 'new_branch'
hishamali@ideapad-330:~/Public$
hishamali@ideapad-330:~/Public$
hishamali@ideapad-330:~/Public$ git branch
  master
* new_branch
hishamali@ideapad-330:~/Public$
```

3. Adding content to the repository

To add content to remote repository which is located in GitHub, we need to first initialize our local repository with git and upload our files to remote repository

Steps

(a) Create a new directory in local computer with the same name of the remote repository
Command : mkdir <directory_name>

```
hishamali@ideapad-330:~/Desktop$ mkdir fosslab
hishamali@ideapad-330:~/Desktop$ ls
fosslab
hishamali@ideapad-330:~/Desktop$
```

(b) Initialize local directory with git

To do this go to created directory and initialize with git

Change the current working directory to your local project.

Command : `cd <local_directory_name>`

```
hishamalip@ideapad-330:~/Desktop$ cd fosslab/  
hishamalip@ideapad-330:~/Desktop/fosslab$
```

#To initialize local directory as Git repository

Command : `git init` (This creates a new subdirectory named '.git' that contains necessary repository files)

```
hishamalip@ideapad-330:~/Desktop/fosslab$ git init  
Initialized empty Git repository in /home/hishamalip/Desktop/fosslab/.git/  
hishamalip@ideapad-330:~/Desktop/fosslab$ ls -a  
.  ..  .git  
hishamalip@ideapad-330:~/Desktop/fosslab$
```

(c) Add the files in your new local repository

First check if there is any untracked files present before adding files

Command : `git status`

```
hishamalip@ideapad-330:~/Public/fosslab$ git status  
On branch master  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
    pgm1.sh  
  
nothing added to commit but untracked files present (use "git add" to track)  
hishamalip@ideapad-330:~/Public/fosslab$
```

To add untracked files to local repository

Command : `git add .` (. is used to add all files)

```
hishamalip@ideapad-330:~/Public/fosslab$ git add .  
hishamalip@ideapad-330:~/Public/fosslab$ git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
  
    new file:   pgm1.sh  
  
hishamalip@ideapad-330:~/Public/fosslab$
```

4. Committing the data to a repository

Stores the current contents of the index in a new commit along with a log message from the user describing the changes.

#To commit files that are added in local repository

Command : git commit -m "commit message"

```
hishamali@ideapad-330:~/Public/fosslab$ git commit -m "initial commit"
[master (root-commit) a5f1dc1] initial commit
1 file changed, 2 insertions(+)
create mode 100644 pgm1.sh
hishamali@ideapad-330:~/Public/fosslab$
```

5. Updating the local copy

The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repository.

Steps

(a) To add a new remote, use the git remote add on the directory where repository is stored at

Command : git remote add origin <url address of remote repository>

```
hishamali@ideapad-330:~/Desktop/fosslab$ git remote add origin https://github.com/
hishamali29/fosslab.git
hishamali@ideapad-330:~/Desktop/fosslab$
```

(b) To add files in local repository to remote repository

Command : git push origin <branch_name> (Here branch is master)

```
hishamali@ideapad-330:~/Desktop/fosslab$ git push origin master
Username for 'https://github.com': hishamali29
Password for 'https://hishamali29@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 236 bytes | 236.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hishamali29/fosslab.git
 * [new branch]      master -> master
hishamali@ideapad-330:~/Desktop/fosslab$
```

6. Comparing different revisions

Comparing different version allow us to see changes we have made on same file. For comparing revisions we need to get the hash value

Steps

(a) Check the changes we have made

Command : git reflog (this will return list of files we have made changed with an id for corresponding changes)

```
hishamali@ideapad-330:~/Public/fossilab$ git reflog
85da957 (HEAD -> master) HEAD@{0}: commit: bad update
a5f1dc1 (origin/master) HEAD@{1}: commit (initial): initial commit
hishamali@ideapad-330:~/Public/fossilab$
```

(b) Comparing different versions

Command : git diff <first commit id> <second commit id>

```
hishamali@ideapad-330:~/Public/fossilab$ git diff a5f1dc1 85da957
diff --git a/pgm1.sh b/pgm1.sh
index a7749a1..1d92fb1 100644
--- a/pgm1.sh
+++ b/pgm1.sh
@@ -1,2 +1,3 @@
 echo "hello world"
+echo "bad update"
hishamali@ideapad-330:~/Public/fossilab$
```

7. Revert

Revert some existing commits. Revert command rollback changes we have committed. It create new commit from a specified commit by inverting it. Hence adds a new commit history to the project, but it doesn't modify the existing one.

Command : git revert <commit id we want to revert>

This will revert the changes specified by the last commit to the first commit

```
hishamali@ideapad-330:~/Public/fossilab$ git revert 85da957
```

Then we will get a prompt of revert that we are going to do. Save it if we want to revert.

```
Revert "bad update"

This reverts commit 85da9572c7e59229e1613ae89961d51a1343a879.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Changes to be committed:
#   modified:   pgm1.sh
#
```

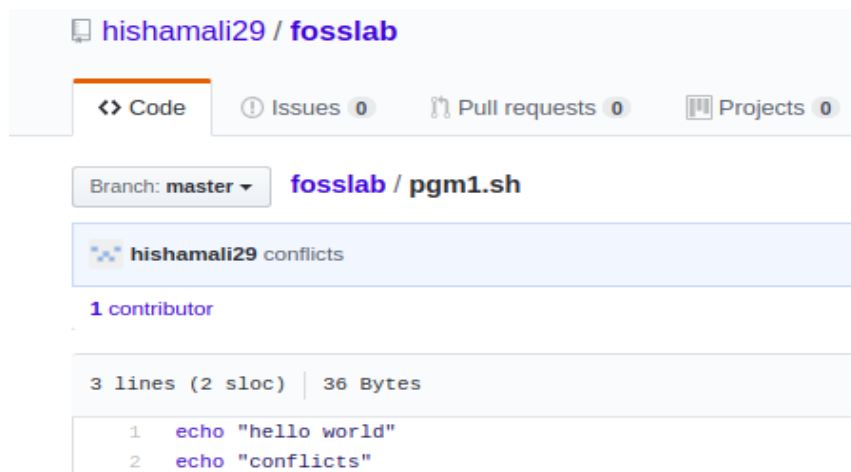
Output after revert

```
hishamali@ideapad-330:~/Public/fosslab$ git revert 85da957
[master 73dbe02] Revert "bad update"
1 file changed, 1 deletion(-)
hishamali@ideapad-330:~/Public/fosslab$ git reflog
73dbe02 (HEAD -> master) HEAD@{0}: revert: Revert "bad update"
85da957 HEAD@{1}: commit: bad update
a5f1dc1 (origin/master) HEAD@{2}: commit (initial): initial commit
hishamali@ideapad-330:~/Public/fosslab$
```

8. Conflicts and a conflict Resolution

If we changed file in remote repository directly we will use a command pull, to pull in the remote changes to local computer. But if we forgot to pull the changes there will be conflict if we tries to push our files after changing local file.

So i manually changes file in remote repository



Now im going to edit file in local repository and push it to remote repository.

Command : nano filename

```
echo "hello world"
echo "conflicts testing"
```

Now pushing chnged file to remote repository

Command : git push origin <brach name>

```
hishamali@ideapad-330:~/Public$ nano pgm1.sh
hishamali@ideapad-330:~/Public$ git push origin master
Username for 'https://github.com': hishamali29
Password for 'https://hishamali29@github.com':
To https://github.com/hishamali29/fosslab.git
 ! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/hishamali29/fosslab.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
hishamali@ideapad-330:~/Public$
```

If the content of a file in GitHub website and local computer doesn't match, there will be conflict. Here we ended up in a conflict. To fix the conflict pull the changes first.

Command : `git pull origin <branch name>`

```
hishamali@ideapad-330:~/Public$ git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/hishamali29/fossilab
* branch                master      -> FETCH_HEAD
   8ac164a..b95fc54      master      -> origin/master
Auto-merging pgm1.sh
CONFLICT (content): Merge conflict in pgm1.sh
Automatic merge failed; fix conflicts and then commit the result.
hishamali@ideapad-330:~/Public$
```

Then we will result in conflict error message like “merge conflict in <brach-name>

Now check the file

Command : `nano filename`

And we can see conflict markers which shows us where the conflict occurred. And there will be different version of the file separated by equal (=) sign. Head will be most recent commit and the other will be the changes made on remote repository.

```
GNU nano 2.9.8                                pgm1.sh
echo "hello world"
<<<<<<< HEAD
echo "conflicts testing"
=====
echo "conflicts"
>>>>>>> b95fc5428996566eda7941ad7126b07b653a5cdc
```

We can select which version to choose. Delete all other version and conflict marker. Then add the files and commit message and push it to required branch.

So im choosing changes made in website and deleting changes made from local repository and save file

```
GNU nano 2.9.8                                pgm1.sh
echo "hello world"
echo "conflicts"
```


Now go back to terminal and add, commit and push new file

Command : git add <filename>
 git commit -m "commit message"
 git push origin <branch name>

```
hishamali@ideapad-330:~/Public$ git add .
hishamali@ideapad-330:~/Public$ git commit -m "conflicts resolved"
[master 3578c38] conflicts resolved
hishamali@ideapad-330:~/Public$ git push origin master
Username for 'https://github.com': hishamali29
Password for 'https://hishamali29@github.com':
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (9/9), 800 bytes | 800.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0)
To https://github.com/hishamali29/fosslab.git
   b95fc54..3578c38  master -> master
hishamali@ideapad-330:~/Public$
```

And go to the pull request tab in remote repository and click “Merge Pull Request” and confirm the merge. It will be resulted with a message “Pull request successfully merged and closed”. This is how to fix conflicts.

Result

Studied about Version Control System setup and usage using GIT and its various uses and commands