Git

Git usage guidelines

singsys

ABSTRACT

This document provides detailed guidelines to implement Git code management system.

Document Revision History

Revision	Date	Comments	Author(s)
1.0	02-12-2014	Initial release	Shashwat Tripathi

Changes Description

No changes yet.

Table of Contents

1 Introduction

Git is a distributed version controlling system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.

Singsys will use Git for managing codes for all the projects. This document describes our new workflow and a usage guide in detail to help you out in getting started with Git.

2 Git server

We have setup our own Git server where all the codes for all projects will be kept. You are required to log into the system using your username and password. Your account on the Git server has been created by Git administrators. In case you have any doubt or need help using the system you can contact any of the Git administrator listed in the appendix.

You can access the system at

http://182.71.125.14

In this system, you can see all the activity for your projects. You can browse code, create new branches

3 Terminology

3.1 Project

All projects will have a corresponding project on the Git server. Project on the server can only be created by the Git administrator. However, you can also create your own project on the Git server for experimenting and learning purposes. We are allowing every developer to create upto 5 personal projects for now to experiment.

Multiple members are assigned with different privileges for one project.

3.2 Push and Pull

Uploading your code from the local system to the server is called ‘push’.

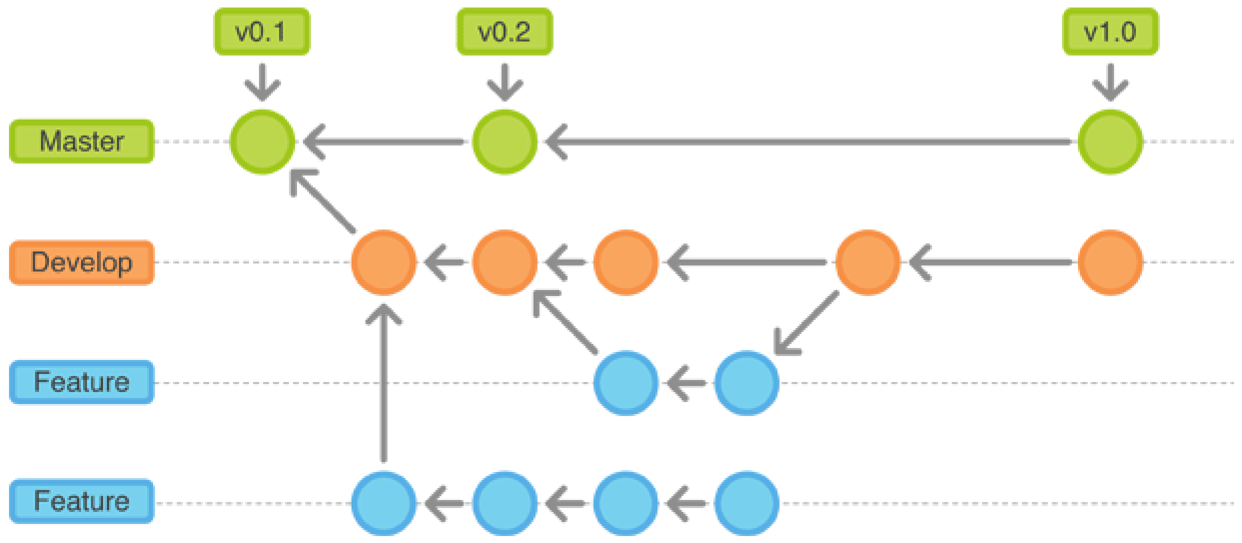
Downloading code from the server into your local system is called ‘pull’

3.3 Branch

Branches are the independent working copies of your project code. All projects will have one ‘master’ branch and one ‘develop’ branch. Master branch will contain all the working, in use, and properly tested code which is to be used on production environment. Development branch (‘develop’) will contain code from the active development. All the developers can upload (‘push’ in Git terms) their codes to the development branch.

Master branch is a **protected** branch which means developers won't be able to push and merge their work directly into the master branch.

Below is a pictorial representation of the branches in Git.



4 User privileges

We have following user privileges associated with a project.

Check the below URL for a detailed information
http://182.71.125.14/help/permissions/permissions

4.1 Master

User will these rights will usually approve or reject the requests submitted by the developers to merge their code into the master branch.

4.2 Developer

User will these rights will have write access to the all the branches except protected branches. They can always create new branch, delete unprotected branches.

Developers will usually push their work to the development branch ('develop') and once they are done with their work they will need to create a merge request using the web interface to merge their work into the master branch.

Users with master rights on the project can review and approve the merge request.

Master users can also leave their feedback or comment on the merge request if something is wrong and need to be fixed. Developers can leave their comment and push the code after making required changes to the corresponding branch. All the activities after the merge request will be shown with merge request in web interface. Once everything is corrected, master users can approve the merge request which will merge the code from development branch to the master branch.

A developer can also create a merge request if s/he is facing any issue or need help in coding even before his/her work is completed. All users can discuss by leaving their comments on the merge request. When a master user approves the merge request, latest work from the development branch will be merged into the master branch.

4.3 Reporter and Guest

Our Git server also supports tracking of bugs. However, we will stick with our current BTS for tacking of bugs.

Reporter and Guests can create new issue and comment on existing issue. Reporter users can also download the code of the project while guest users cannot.

5 Using Git – step by step

5.1 Installation

Git is supported on all the systems including OSX, Windows and Linus distributions.

OSX and Linux users may already have the Git installed in their system. To check if you have Git installed or not on your system, just run 'git' command from the terminal. If you see a message saying 'command not found' then you need to install the Git. Else, check the version of the Git by using 'git --version' command. We will recommended using of latest version of Git. If you do have latest version of the Git then install it again to update.

Follow the instructions given the below URL for downloading and installing Git.

http://git-scm.com/downloads

5.2 Configuration

Git does not require anything to configure and in most cases you are good to go. But you may want to set your identity using the below commands

```
git config --global user.name 'your name'
git config --global user.email 'your email'
```

Use your real name in user name, not the username from your Git credentials. Use your official email address (…@singsys.com) in the email address.

5.3 Git command line

There is very good interactive tutorial developed by github. Please run through this tutorial available at below URL.

https://try.github.io

5.3.1 Initializing a git repository

After you have installed and configured Git on your system. You can add git version controlling to your existing or new project by using the git initialization command.

Follow below steps:

1. Create a directory for your new project say 'GettingGit'
2. Add three blank files to your project say 'f1.txt', 'f2.txt' and 'f3.txt'

Now open terminal and move inside your project directory 'GettingGit' by using 'cd' command(s).

Note: I am using Windows OS and have created folder inside the E drive.

Now run following command to initialize a local git repository.

```
git init
```

This will create a hidden folder with name '.git' where it manages your local git repository.

5.3.2 Git status command

Git status command is very useful. It tells you about your modified files, untracked files etc. The result of status command may be different according the current state of your repository. It also tells you the command you may want to run next.

If you run 'git status command now, you will see following result

```
$ git status
On branch master


Initial commit


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


        f1.txt
        f2.txt
        f3.txt


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

First line tells you the name of the branch you are current on. Git creates first branch itself with name master.

It also list the files those are not being tracked by the Git. You need to always tell git to track file.

5.3.3 Tracking file

You can add file to the git by using ‘git add’ command in the following way.

```
git add f1.txt
git add f2.txt
git add f3.txt
```

Or, you can add all files at once using the following command

```
git add --all
```

This command will add all new and modified files to the staging area.

What is staging area?
Stating area is a place where git keeps all your files ready to be committed in the next commit

What is a commit?
Whenever you run commit command, git creates a snapshot of your project. It is something like a copy of your project. You can always move back to the any commit.

Now, if you run git status command again, you will find all files in the ‘changes to be committed’ area. This are is called ‘staging area’.

5.3.4 Committing changes

Whenever you run commit command, git creates a snapshot of your project.

You are required to pass a message with each commit in following way.

```
git commit –m ‘my first commit’
```

Running the above command will save the snapshot of your project.

Now if you run git status command again then it will show you below message

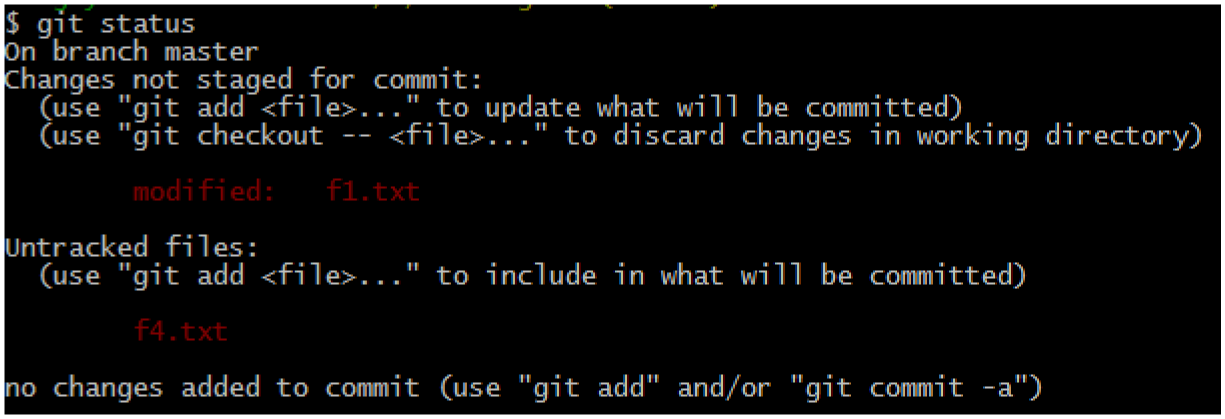
```
$ git status
On branch master
nothing to commit, working directory clean
```

5.3.5 Git add command

‘git add’ is multipurpose command. This can be used to start tracking new files, add your modified or deleted files to the staging area.

Now add a new file ‘f4.txt’ in your project directory and edit file ‘f1.txt’ by writing some text into the file. I have written a single line ‘line 1’ in ‘f1.txt’.

Now run git status command



```
$ git status
On branch 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:   f1.txt

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

        f4.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Git clearly tells you the whole story that file f1.txt has been modified and there is another new file which not being tracked by Git.

Remember, we need to tell git to start tracking our new files. This need to done only once. Like you see for file ‘f1.txt’ which is already being tracked by the Git so git tells us that this file is modified.

Now, we are adding both ‘new file’ and ‘modified file’ to the staging area using below command

```
git add f1.txt f2.txt
```

Note: you can multiple files to the staging area by separating file names by space. If your file name already contains white space in its name then you will need to give file names in single or double quotes.

You can also do it using following command without having to specify file names.

```
git add --all
```

Above command will stage new files, modified files and deleted files ready to be committed.

Now you can commit our work using commit command as below

```
git commit –m ‘learned staging changes’
```

Shorthand:

You can add and commit in one single command also by specifying ‘-a’ parameter in commit command like below

```
git commit –a –m ‘learned staging changes’
```

5.3.6 Adding remote

You have not yet uploaded your work on the remote server.

Now go the http://182.71.125.14/ and create new project there. Name it anything. I am naming it same as it is on my local ‘GettingGit’

Once your project is created on the server, it will show some useful commands on the screen to push your code to the server.

Since we already have our local repository on our system. We will add remote and push our work remote server using following command.

New remote can be added using following command:

```
git remote add origin 'http://182.71.125.14/shashwat/gettinggit.git'
```

In above command we have added a remote location in our git repository. It is possible to add multiple remote location. Here, 'origin' is just friendly name of our remote repository. We have used 'origin' as it is a recommended name for remote repository. You can use any name you want to use.

Now we will push our changes to the remote repository. Make sure you are connected to the internet.

```
git push -u origin master
```

This command is telling git to push your master branch from local to the remote repository. '-u' flag add tracking reference in your master branch with master branch of remote repository. So later on, you can look for changes on the server made by other users.

Note:

If you are a user with developer rights, you won't be able to push your work to the master branch anymore. Because master branch is protected by default on the server so developers will not be able to push their work directly to the master branch.

Solution is to create a new branch, do commit(s) in the new branch and then create a merge request.

When you do a push, all your local commits are updated to the remote repository. You can browser code and commits from the web browser.

Since you are the owner of this project, you can always write to the protected branch. But we will create a new branch from here to understand our new workflow.

Note:
You can still commit your new work in master branch in your local repository. But you won't be able to push your commits to remote repository.

5.3.7 Creating new branch

You can create a new branch by using following command

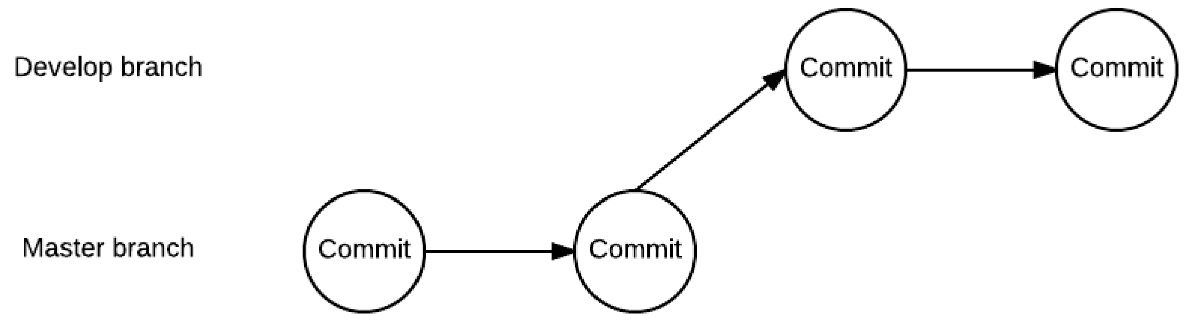
```
git branch develop
```

Here, you have created a new branch with name 'develop'. But you are still at master branch, if you do a commit now, your new work will be committed to the master branch, not in the develop branch.

Changing your branch is called 'checkout'. You can checkout new branch by following command.

```
git checkout develop
```

Following picture represents our current branching structure. Commits in the develop branch are just example commits.



You can also create and checkout new branch by a single command

```
git checkout -b develop
```

Above command will create new branch and move your pointer to the new branch at the same time.

Now make some changes and do one or two commits using following command.

```
git commit -a -m 'worked in develop branch'
```

I had explained above command already but again telling you that above command will stage all the changes and do a commit in the same time. It is a shorthand of below two commands

```
git add --all
git commit -m 'worked in develop branch'
```

And you should often run ‘git status’ command to check the status of your repository. It tells you about what files you have removed, added and modified. It also tells you about the commands you may want to run next.

Now I am assuming you have made a few commits in your develop branch. In next section I will tell you how to update your work from develop branch on the server.

5.3.8 Creating branch on server

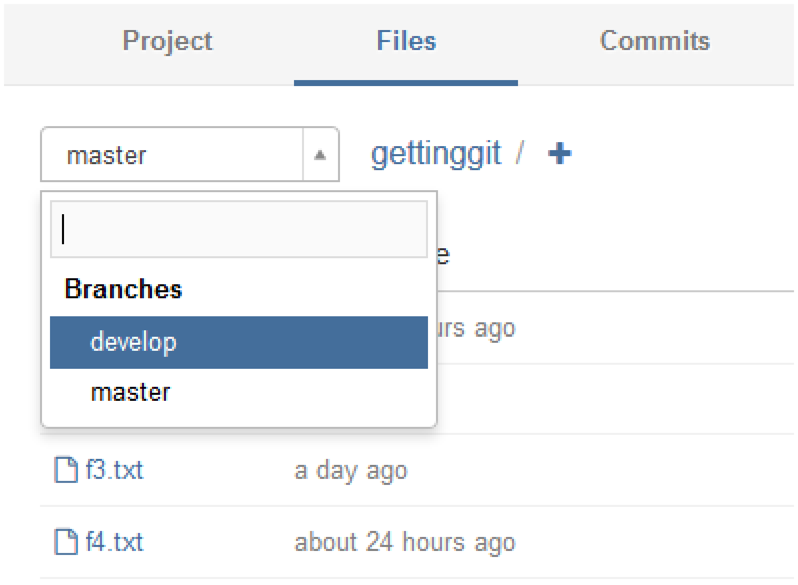
In previous section you have created a new branch locally but you have not yet uploaded your work on the server. Basically, what you do need is to create a corresponding branch on the remote repository as well and then upload your work onto the server.

You can do so with the following command(s)

```
git push -u origin develop
```

You need to be on 'develop' branch when running this command. If you are not run command 'git checkout develop'. In above command, 'origin' is the name of our remote repository and 'develop' is the branch name on the remote repository. You can give any name you want to give for your remote branch. But it is good to use same name as it is in your local repository. What '-u' flag does is, it stores the tracking information with the branch so next time you can only run 'git push' to push your changes to the remote branches. The behavior of 'git push' command is configurable and changed in different versions of Git.

Now changes are updated on the server. You will see two branches on the server 'master' and 'develop' as shown in below screen



You can switch the branches on remote and browser codes in different branches. You should see changes in 'develop' branch those are not yet merged in 'master' branch.

In next section you will learn how to submit a merge request.

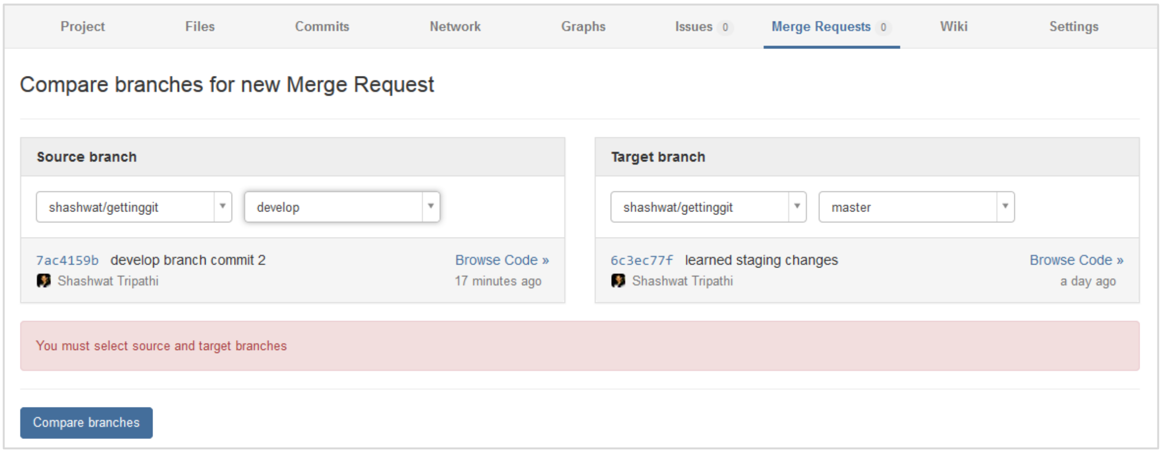
5.3.9 Creating a merge request

Once you have pushed your work on develop branch, you should create a merge request to update your work from 'develop' branch to the 'master' branch.

To create a merge request follow the below steps:

1. Login to the Git server by using your Git credentials.
2. Navigate to the project.
3. Click on 'Merge Requests' from project navigation bar.

4. Click on big green button 'New Merge Request'
5. You should see similar screen as shown below



There are two sections 'Source branch' and 'Target branch' in the this windows. You should select 'develop' branch as source branch and 'master' as target branch.

6. Click on 'Compare branches' button.
7. In next screen, there is a form where you will need to provide following information.
   a. Title (give any suitable title)
   b. Description (optional)
   c. Assign to (select person who is going to review and approve your merge request)
   d. Milestone (optional)
8. Click on submit merge request button to create merge request.

Once a merge request is created, everyone from project member or group member can leave their comment on the work you've done. You can do changes and push them if required.

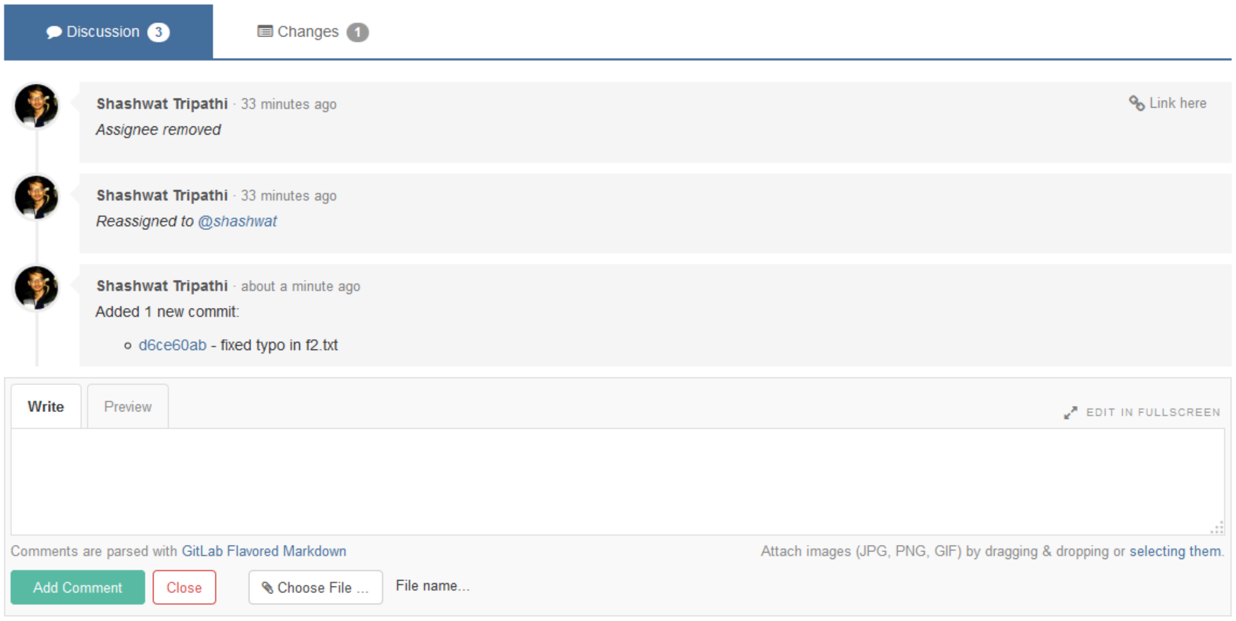
I am assuming that your project master left a comment asking you do fix a typo in your work.

Now here is how you will do it.

1. Open terminal
2. Navigate inside your project directory.
3. Use command 'git checkout develop' to switch to the develop branch if you are not already on it.
4. Edit the file using your preferred editor.
5. Do a commit and push using following commands

```
git commit -a -m 'fixed typo in f2.txt'
git push origin develop
```

This newly done commit will be shown in the discussion in the merge request page as shown in below figure.



Shashwat Tripathi · 33 minutes ago
Assignee removed

Shashwat Tripathi · 33 minutes ago
Reassigned to @shashwat

Shashwat Tripathi · about a minute ago
Added 1 new commit:
- d6ce60ab - fixed typo in f2.txt

Comments are parsed with GitLab Flavored Markdown
Attach images (JPG, PNG, GIF) by dragging & dropping or selecting them.

Now you've corrected the issue so the corresponding person will approve your request.

Here, I have assigned this merge request to myself. So I have approved my merge request.

Once your merge request is approved, you can pull changes from server to your local repository using pull command as below.

```
git pull
```

This command will fetch and merge the changes from the remote server into your local repository.

Now your both branches are same. You can carry on your work in 'develop' branch, make commits and push changes to the server. You will need to create merge request again whenever you want to merge your code into the master branch.

5.4 Ignoring files

You may not want always want to upload all of your files in your project directory. These files may be

- Builds (like ipa, apk, dll, exe etc)
- Data files (you should not store database or any user data related file into git repository)

This is done by adding a '.gitignore' file into your project. All your rules to ignore files are written inside this file. This file is not created by default. You will need to create it yourself. Google 'git ignoring files' to learn more about patters.

Some common '.gitignore' files can be found at below url for different type of projects. You can use these patterns into your project while adding your own patters in the same file.

https://github.com/github/gitignore

6 Appendix

6.1 Git commands

Please refer to the following resources to learn more about git commands.

1. http://ndpsoftware.com/git-cheatsheet.html
2. https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf

6.2 Reference books

You should refer to the following books to gain complete command on Git.

1. http://git-scm.com/book/en/v2
   This is freely available official book written by Scott Chacon (master of git). You can read it online or download in pdf.
2. http://singsys.org/ubr_uploads/Git-Version-Control-Cookbook.pdf
3. http://gitref.org/

6.3 Got any friend?

In mean time if you come up with any issue or need help, don't forget Google is always your best friend.

Also, if you are facing any difficulty in getting started with Git, got any issue or have any doubt, please ask me. I would love to help you out with your problem.

6.4 Administrators

Following users are assigned with admin rights. You should reach them if you have any account related problem or to create to new projects.

1. Shahwat Tripathi
2. Sanjay Kumar Shukla
3. Gulshan Verma