



**Git User Manual**

**Copyright © 2018 i-exceed technology solutions private limited. All rights reserved**

No part of this document may be reproduced, stored in a retrieval system, adopted or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, scanning or otherwise), translated into any language or computer language, without the prior written permission of i-exceed technology solutions private limited.

Due care has been taken to make this document as accurate as possible. However, i-exceed takes no responsibilities for the contents hereof and shall not be responsible for any loss or damage caused to the user by the direct or indirect use of this document. Furthermore, i-exceed reserves the right to modify or otherwise change in any manner the content hereof, without obligation to notify any person of such revision or changes.

**TABLE OF Contents**

Introduction ------------------------------------------------------ 4

Login & Request access to the project -------------------------------- 4

Git Configuration ------------------------------------------------------ 7

**Working With GIT**

Git clone –----------------------------------------------------- 8

Git Status ------------------------------------------------------ 9

Git Add/commit/push ------------------------------------------------------ 9

Check Logs/Diffs ------------------------------------------------------ 12

Rollback changes ------------------------------------------------------ 15

# Introduction

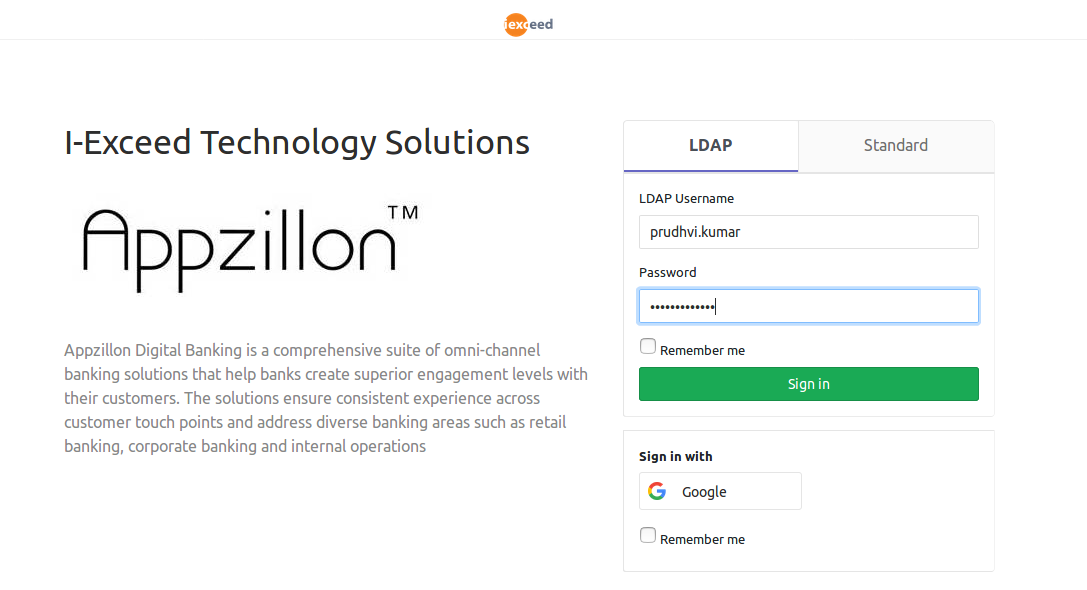
This document has been prepared to serve as git user manual for source code management using git. The following are the detailed instructions to use Git and GitLab.

* Login and Access request to the project.
* Git configuration
* Add / commit / push / pull files to/from Repository
* Check logs / diffs
* Rollback changes
* Conflicts

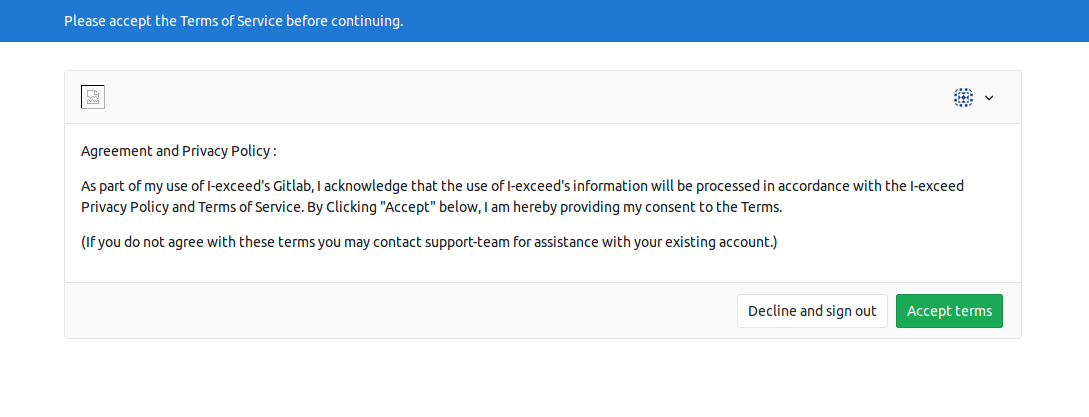
# Login & Request access to the project

Open a browser tab and navigate to the below specified URL.

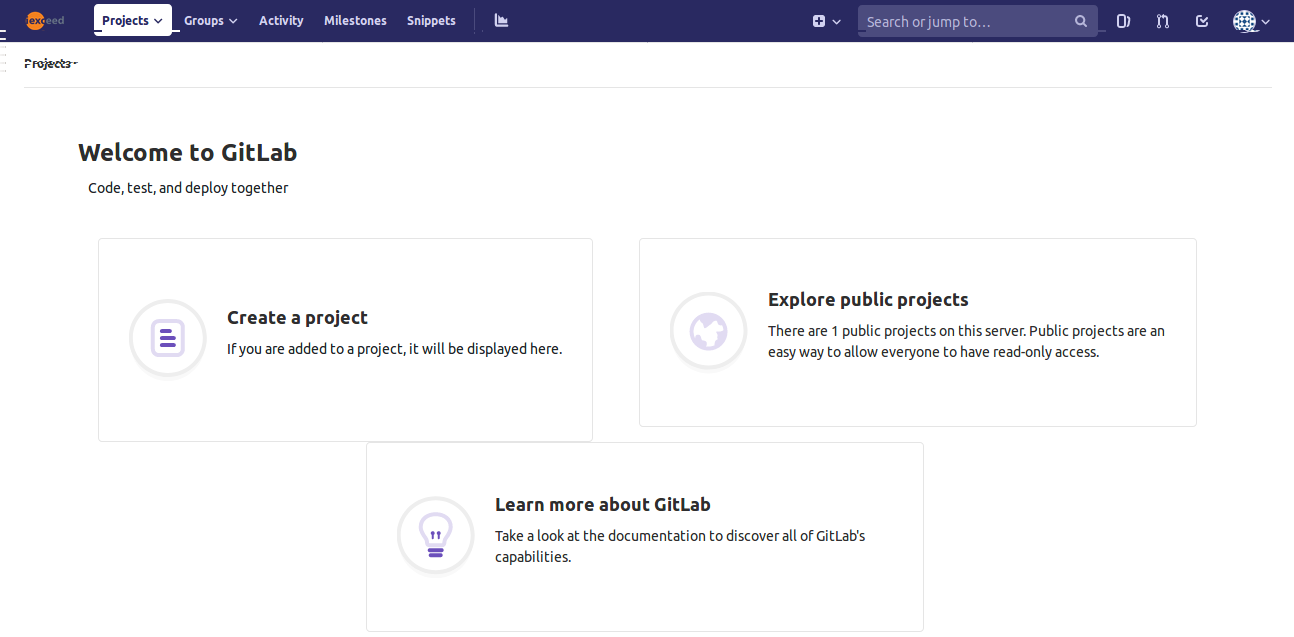
GitLab URL : **<http://192.168.1.6:81/>**



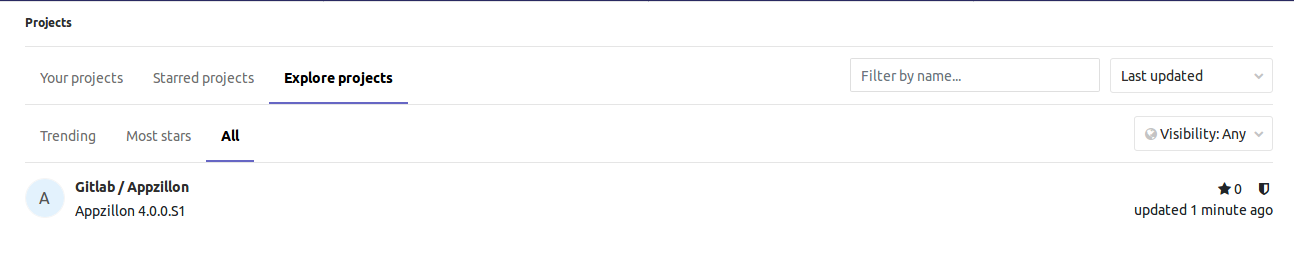
Login with your system login credentials [user.name and password] and sign-in to your GitLab account after accepting the Terms and conditions.



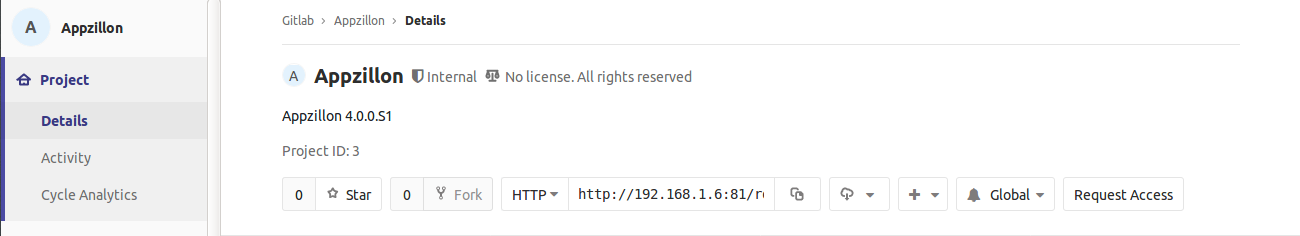
After Successful Login you will see the GitLab Dashboard as below.



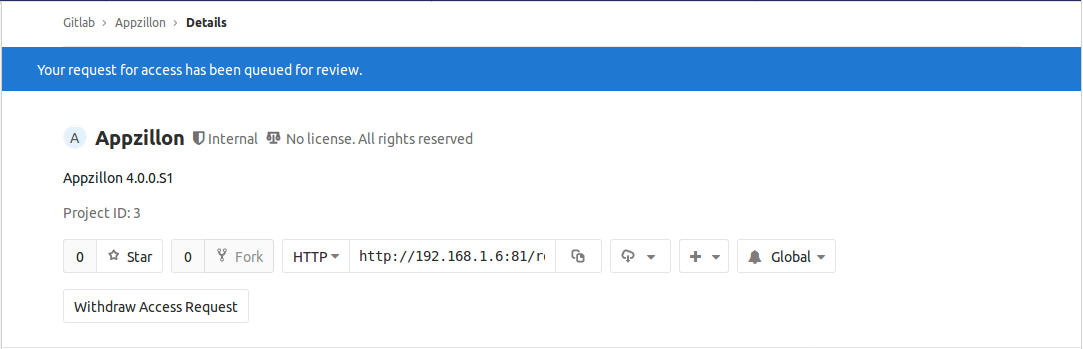
On the Dashboard navigate to “Explore public Projects” tab and go to “ALL” to find the respective project.



Next, click on the appropriate project and Request access to the project from “Request access” button on top right.

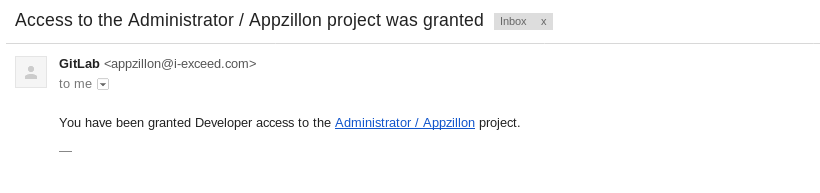


You will get an acknowledgment of the Request as below once you have sent successful request.

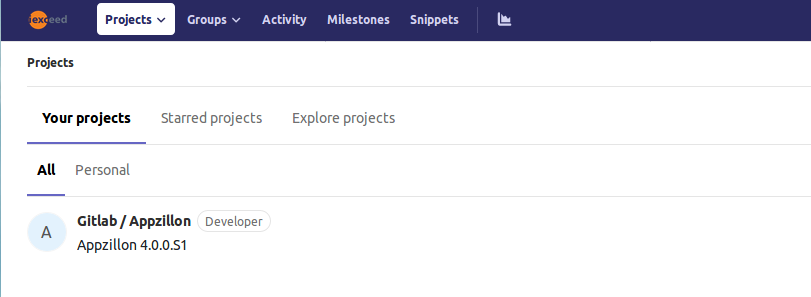


Now, send a mail saying that you have requested access for the particular GIT Repository.

Once your request to the Repository is approved, you will be receiving a acknowledgment mail saying that “Access granted.”



After getting access to the requested repo you can see the Repository on your hone dashboard. [it contains all repo’s for which you have access]



# Git configuration

After Successful installation of Git, setup the initial configuration of git with user-name and mail-id for the Git

Open Terminal [Ubuntu]

* Set your **user-name** [same as your login-Id]
* Set your **mail -id** [example@i-exceed.com] with below commands.

$ **git config --global user.name “user.name”**

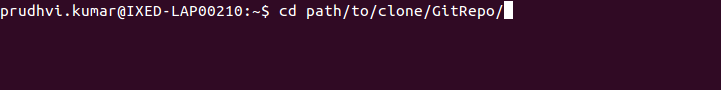
$ **git config --global user.email “email-id”**

# Working with GIT

## Navigate to the directory:

* Open the terminal and navigate to the path where you would like to save the sources.

Eg : **cd /home/i-exceed/user/Git/Appzillon**



## To clone the sources from Remote Repository:

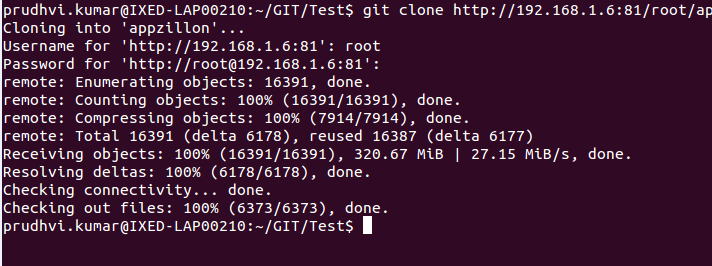
* To clone / checkout the sources from the Remote repository use the following command.

Command : $ **git clone < Repo URL >**

Eg : git clone http://192.168.1.6:81/root/appzillon.git

To clone the sources from the remote repository, you should be the member of that project with respective permissions.

After entering the git clone command you will be asked for user-name and password. Once it is validated, it will clone the sources to the respective path with the specified name.



## Modify the files:

* Navigate to the cloned sources, Modify the files you would like to and save them as-usual.
* Now we have to send the modified files from our local working copy to the remote repository.

## Check status:

* To check the status i.e; to list the modified/ created / deleted files use the following command.

Command : $ **git status**

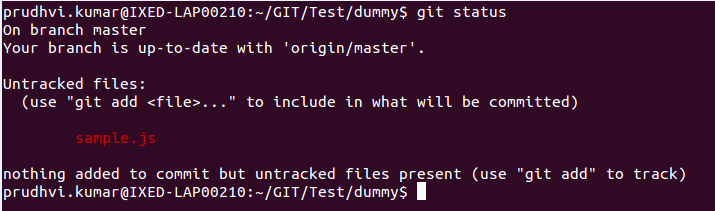
## Check the Remote repository URL:

* To make sure that your remote repository URL is correct.. use the following command.

Command : $ **git remote -V**

## Add / commit / push files to Remote repository:

* Now we have created and modified a file called ***sample.js*** and saved to the files to the local working directory.
* Now check the status of the local working copy with the git status command and it will list the files which you have added / modified / deleted to the local copy.

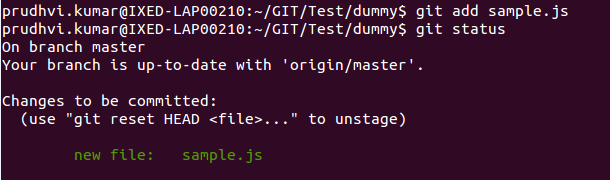


* Here we have a untracked file called sample.js in local copy which has to be pushed to remote repo.
* To add this file/files to the staging [ready to commit] use the git add command as following.

Command : $ **git add <filename>**

Eg: git add sample.js

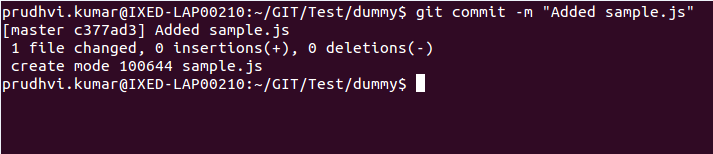
* Check the status again after adding the files to the staging to ensure all the files have been added or not with git status command.



* Now, the sample file has been moved to statging and ready to commit.
* Use the git commit command to commit the changes with a commit message.

Command : $ **git commit <filename> -m “commit message”**

Eg: git commit sample.js -m “Added sample.js”

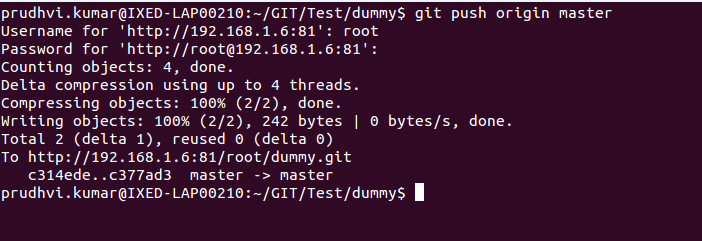


* Now you have committed the changes. Now you have to push the local changes to the remote repository to make the changes available for other developers to use them.
* Use the following command to push the commits to remote. Please enter the project credentials when prompted.

Command : $ **git push origin <branch name>**

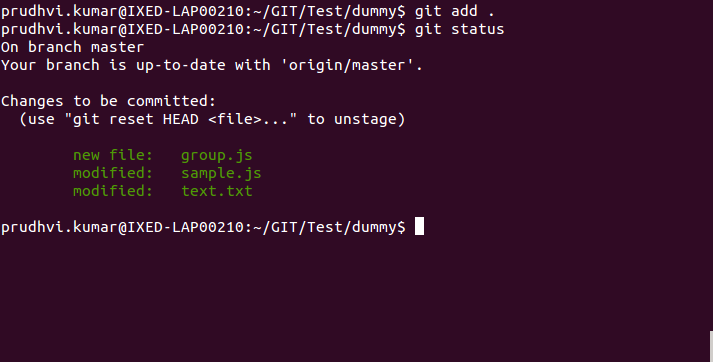
Eg: git push origin master

* By default git push command will take the master branch to push the changes. So, you have to be careful while pushing the changes to the remote repository.



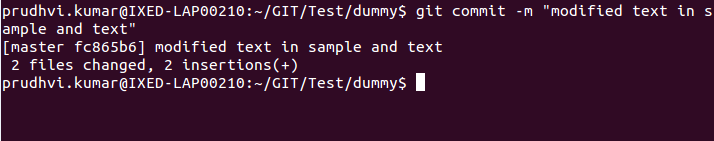
* Now, What if we have to push multiple files to the remote repository. So, add / modify / delete some files and check the status with git status command.
* To add multiple files at a single time instead of single files, use the following command.

Command : $ **git add .**



* To Commit multiple files use the following command.

Command : $ **git commit -m “Appropriate Commit message”**



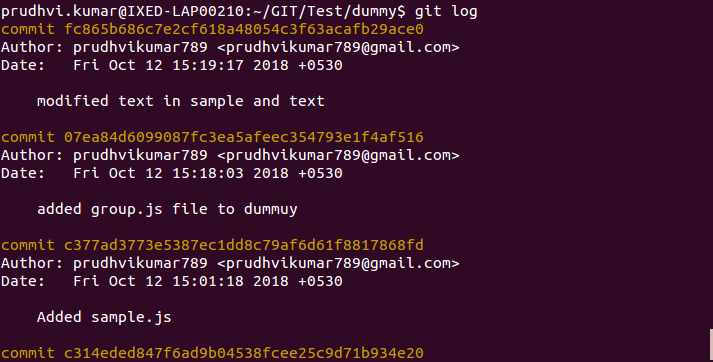
* This will commit the files with a single history log which would be easy to maintain . Now, the committed files are ready to send to remote Repository.
* To Push the multiple files after commit, use git push command.



## Check Logs / Diffs :

* To check the log history of a particular repository use the following command .

Command : $ **git log**



* To check the log history of a particular file, use the following command.

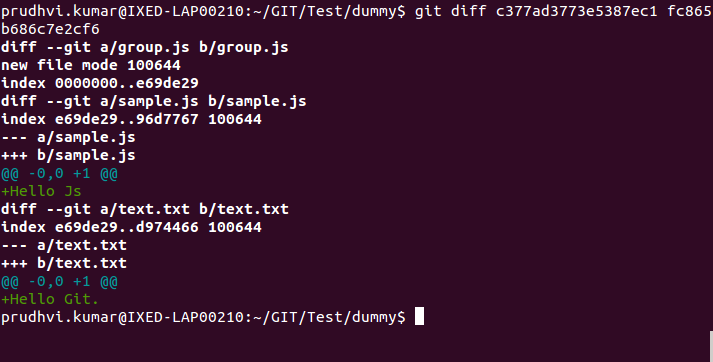
Command : $ **git log <filename>**

Eg: git log sample.js

* To check the Differences of two particular commits , use the following command.

Command : $ git diff <commit Hash-1> <commit Hash-2>

Eg : git diff 64gdbhew6g 78eywhs6cdd



* To check the Differences of two **files** , use the following command.

Command : $ **git diff <file-1> <file-2>**

Eg : git diff sample.js dummy.js

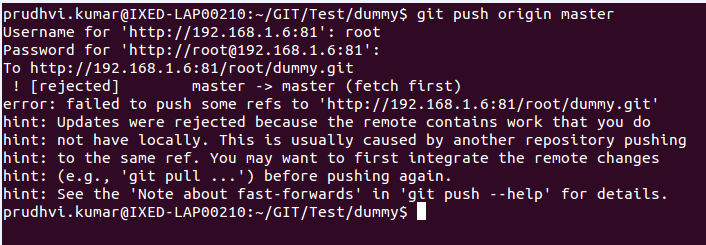
## Push Conflicts :

* Make sure your local working copy has all the changes from remote repository. If not pull the changes from the remote repository with following command.

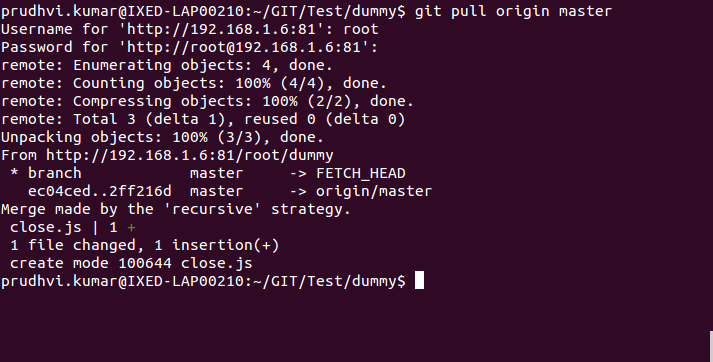
Command :$ **git pull origin <branch name>**

Eg: git pull origin master

* If you try to push the changes without being sync with remote repository, you will face the conflicts shows that you local working copy is not updated with remote repository.



* So, pull the changes first and then try to push the local changes to remote repository with the git push command.



* To avoid conflicts while pushing the changes to remote repository make sure you have the remote changes with you in the local working copy.

## Rollback to Previous change:

* To rollback the changes after adding the files to the staging use the following command so that the changes you made before adding them to staging will be rollback as untracked files.

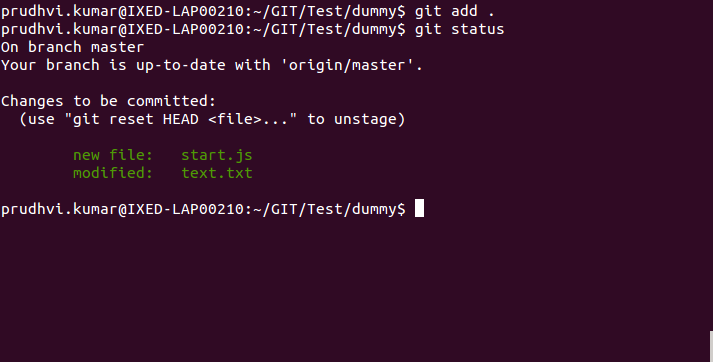
Command : $ **git reset --mixed**

* To rollback to a particular commit, use the following command.

Command : $ **git reset <commit hash-value>**

Eg : git reset 87wgbchbdh6sx

From :



To

