Subject Name: **Source Code Management**

Cluster: Beta

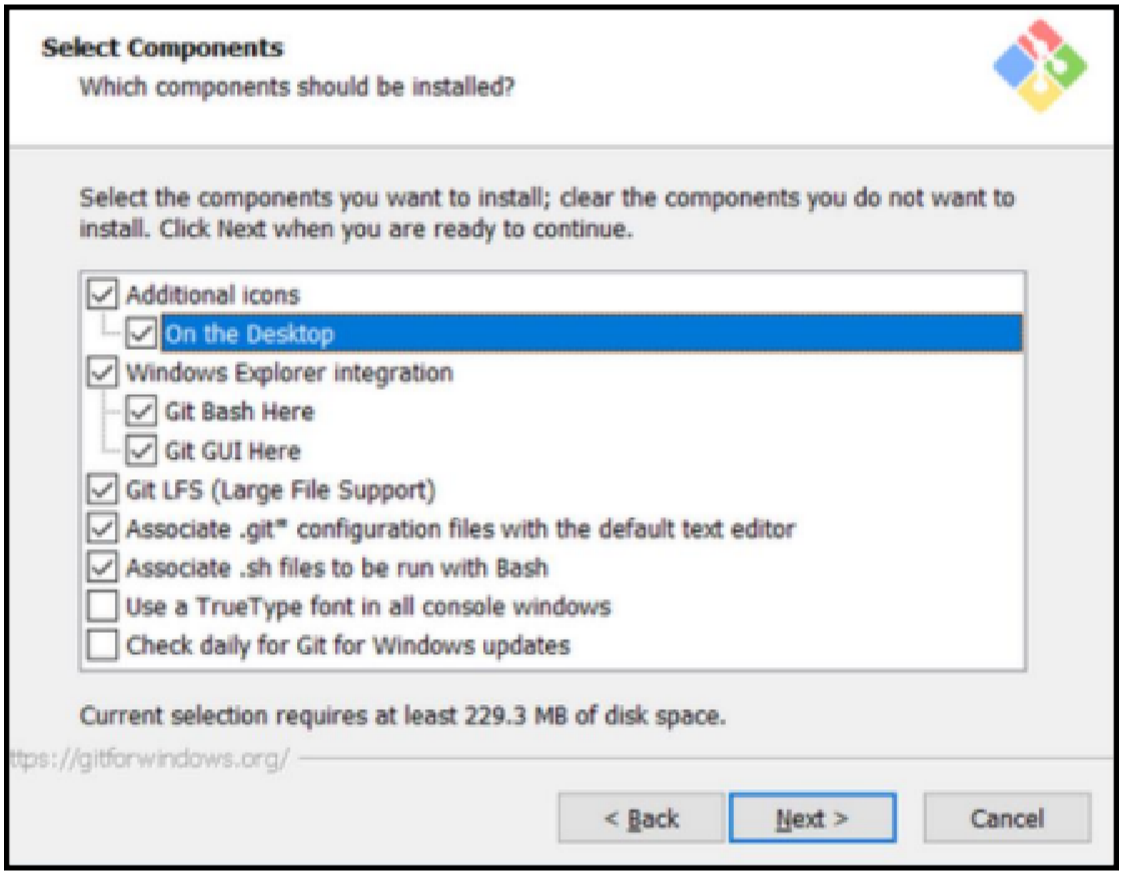
Submitted by – Devesh Gupta
Roll number – 2110990428
Group – G8-B
Department - CSE

Submitted to – Monit Sir

Aim: Setting up the git client.

Git Installation: Download the Git installation program (Windows, Mac, or Linux) from Git - Downloads (git-scm.com).

When running the installer, various screens appear (Windows screens shown). Generally, you can accept the default selections, except in the screens below where you do not want the default selections:In the Select Components screen, make sure Windows Explorer Integration is selected as shown:


Select Components
Which components should be installed?

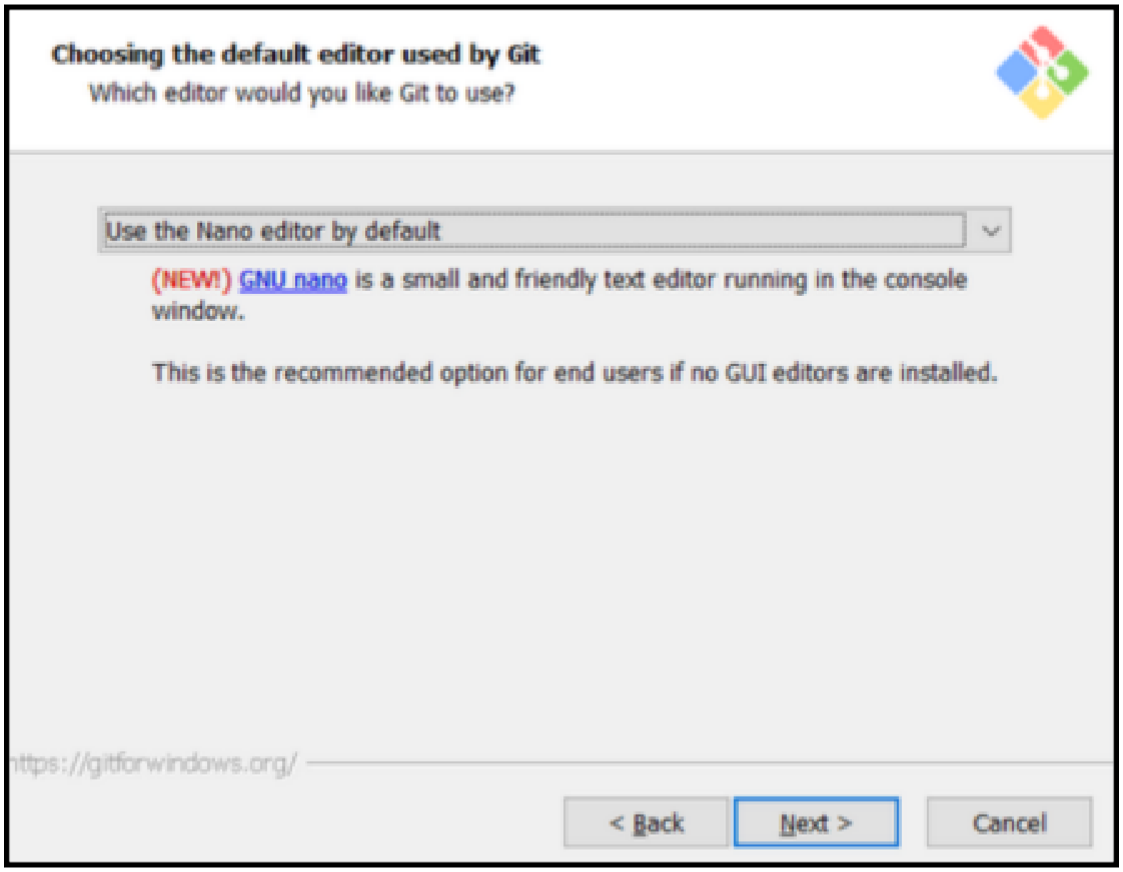
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [✓] Additional icons
  - [✓] On the Desktop
- [✓] Windows Explorer integration
  - [✓] Git Bash Here
  - [✓] Git GUI Here
- [✓] Git LFS (Large File Support)
- [✓] Associate .git* configuration files with the default text editor
- [✓] Associate .sh files to be run with Bash
- [ ] Use a TrueType font in all console windows
- [ ] Check daily for Git for Windows updates

Current selection requires at least 229.3 MB of disk space.

In the choosing the default editor is used by Git dialog, it is strongly recommended that you DO NOT select default VIM editor- it is challenging to learn how to use it, and there are better

modern editors available. Instead, choose Notepad++ or Nano – either of those is much easier to use. It is strongly recommended that you select Notepad++.



Choosing the default editor used by Git

Which editor would you like Git to use?

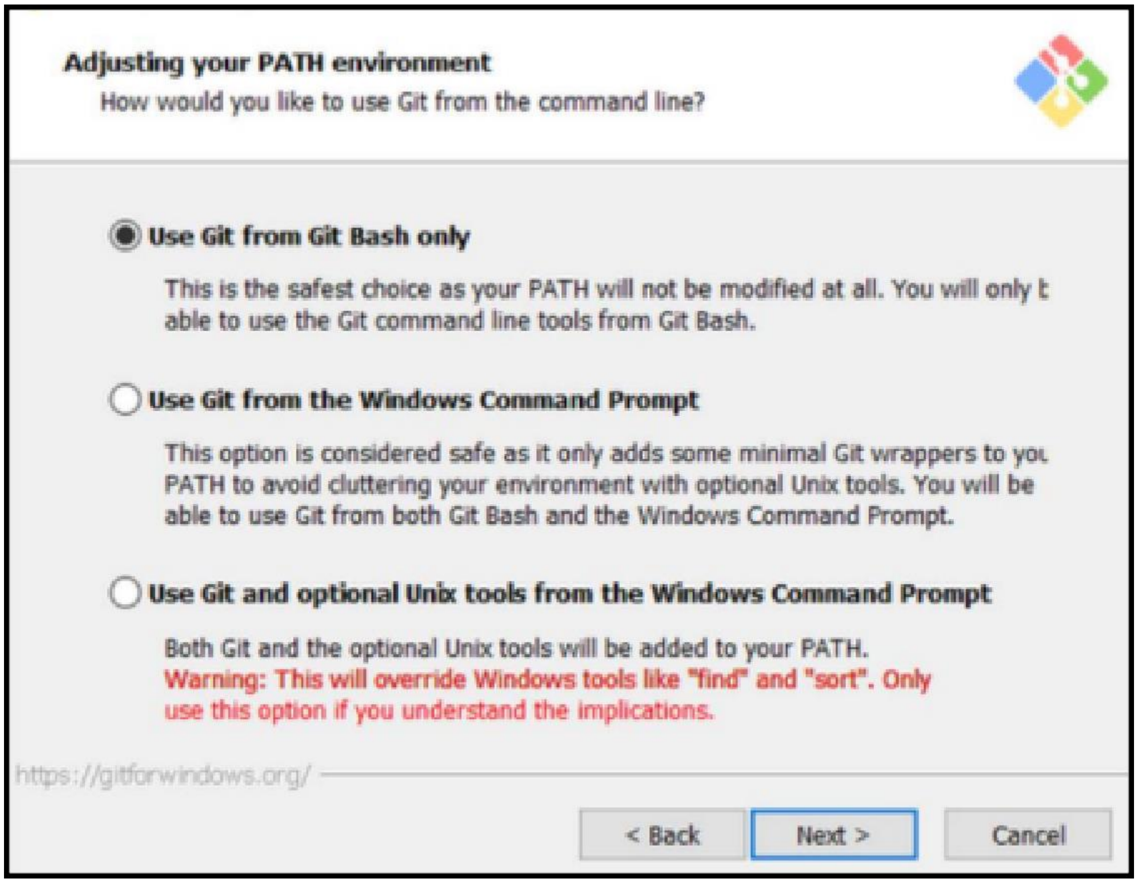
Use the Nano editor by default

(NEW!) GNU nano is a small and friendly text editor running in the console window.

This is the recommended option for end users if no GUI editors are installed.

In the Adjusting your PATH screen, all three options are acceptable:

1. Use Git from Git Bash only: no integration, and no extra command in your command path.
2. Use Git from the windows Command Prompt: add flexibility – you can simply run git from a windows command prompt, and is often the setting for people in industry – but this does add some extra commands.
3. Use Git and optional Unix tools from the Windows Command Prompt: this is also a robust choice and useful if you like to use Unix like commands like grep.



Adjusting your PATH environment

How would you like to use Git from the command line?

[✓ Use Git from Git Bash only]

This is the safest choice as your PATH will not be modified at all. You will only b
able to use the Git command line tools from Git Bash.

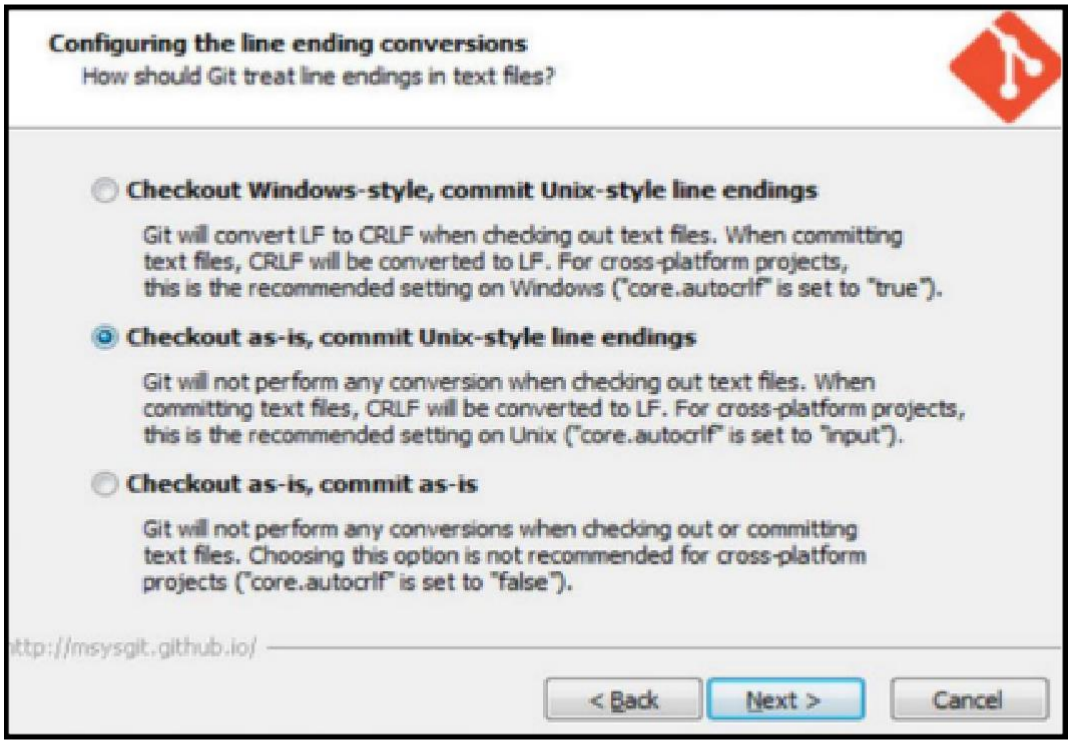
Use Git from the Windows Command Prompt

This option is considered safe as it only adds some minimal Git wrappers to you
PATH to avoid cluttering your environment with optional Unix tools. You will be
able to use Git from both Git Bash and the Windows Command Prompt.

Use Git and optional Unix tools from the Windows Command Prompt

Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only
use this option if you understand the implications.

In the Configuring the line ending screen, select the middle option (Checkout-as-is, commit Unix-style line endings) as shown. This helps migrate files towards the Unix-style (LF) terminators that most modern IDE's and editors support. The Windows convention (CR-LF line termination) is only important for Notepad.


Configuring the line ending conversions

How should Git treat line endings in text files?

( ) **Checkout Windows-style, commit Unix-style line endings**

Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").

(●) **Checkout as-is, commit Unix-style line endings**

Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").

( ) **Checkout as-is, commit as-is**

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").

Configuring Git to ignore certain files:

This part is extra important and required so that your repository does not get cluttered with garbage files.

By default, Git tracks all files in a project. Typically, this is not what you want; rather, you want Git to ignore certain files such as .bak files created by an editor or .class files created by the Java compiler. To have Git automatically ignore particular files, create a file named .gitignore (note that the filename begins with a dot) in the C:\users\name folder (where name is your MSOE login name).

NOTE: The .gitignore file must NOT have any file extension (e.g. .txt). Windows normally tries to place a file extension (.txt) on a file you create from File Explorer - and then it (by default) HIDES the file extension. To avoid this, create the file from within a useful editor (e.g.Notepad++ or UltraEdit) and save the file without a file extension).

Edit this file and add the lines below (just copy/paste them from this screen); these are patterns for files to be ignored (taken from examples provided at https://github.com/github/gitignore.)

```
#Lines (like this one) that begin with # are comments; all other lines are rules


# common build products to be ignored at MSOE
*.o
*.obj
*.class
*.exe


# common IDE-generated files and folders to ignore
workspace.xml
bin
/
out
/
.classpath
# uncomment following for courses in which Eclipse .project files are not checked
in # .project


#ignore automatically generated files created by some common applications,
operating systems
*.bak
*.log
*.ldb
~*
.DS_Store*
._*
Thumbs.d
b


# Any files you do not want to ignore must be specified starting with ! # For example,
if you didn't want to ignore .classpath, you'd uncomment the following rule: #
!.classpath
```

Note: You can always edit this file and add additional patterns for other types of files you might want to ignore. Note that you can also have a .gitignore files in any folder naming additional files to ignore. This is useful for project specific build products. Once Git is installed, there is some remaining custom configuration you must do. Follow the steps below:

a. From within File Explorer, right-click on any folder. A context menu appears containing the commands " Git Bash here" and "GitGUI here". These commands permit you to launch either Git client. For now, select

Git Bash here.

b. Enter the command (replacing name as appropriate) **git config -- global core.excludesfile c:/users/name/. gitignore**

This tells Git to use the .gitignore file you created in step 2

NOTE: TO avoid typing errors, copy and paste the commands shown here into the Git Bash window, using the arrow keys to edit the red text to match your information.

c. Enter the command **git config --global user.Email "name@msoe.edu"** This links your Git activity to your email address. Without this, your commits will often show up as "unknown login". Replace name with your own MSOE email name.

d Enter the command **git config --global user.name "Your Name"** Git usesthisto log your activity. Replace "Your Name" by your actual first and last name.

e. Enter the command **git config --global push.**default simple This ensures that all pushes go back to the branch from which they were pulled. Otherwise pushes will go to the master branch, forcing a merge.

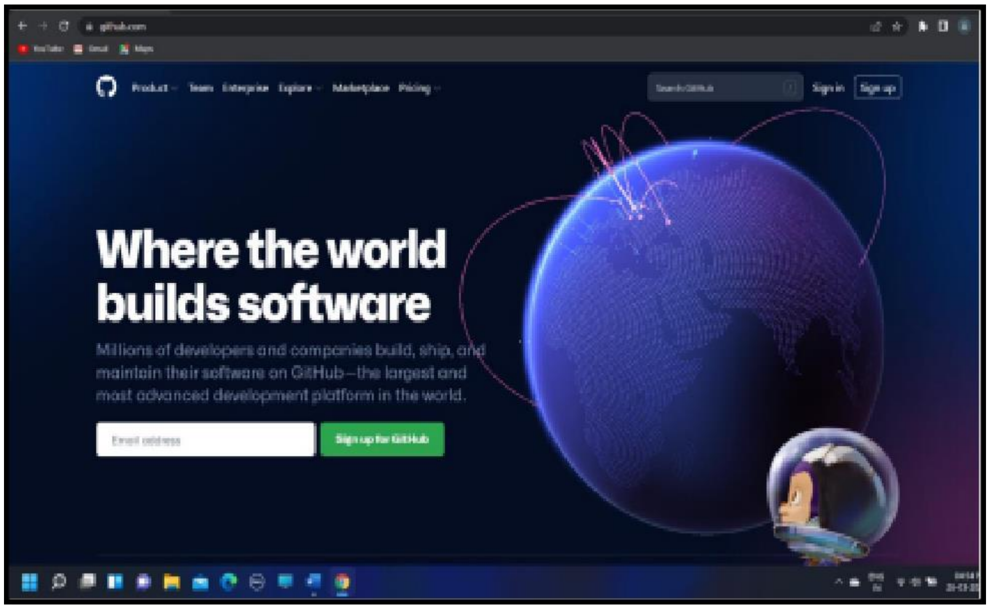
Experiment No. 02

Aim:

Setting up GitHub Account

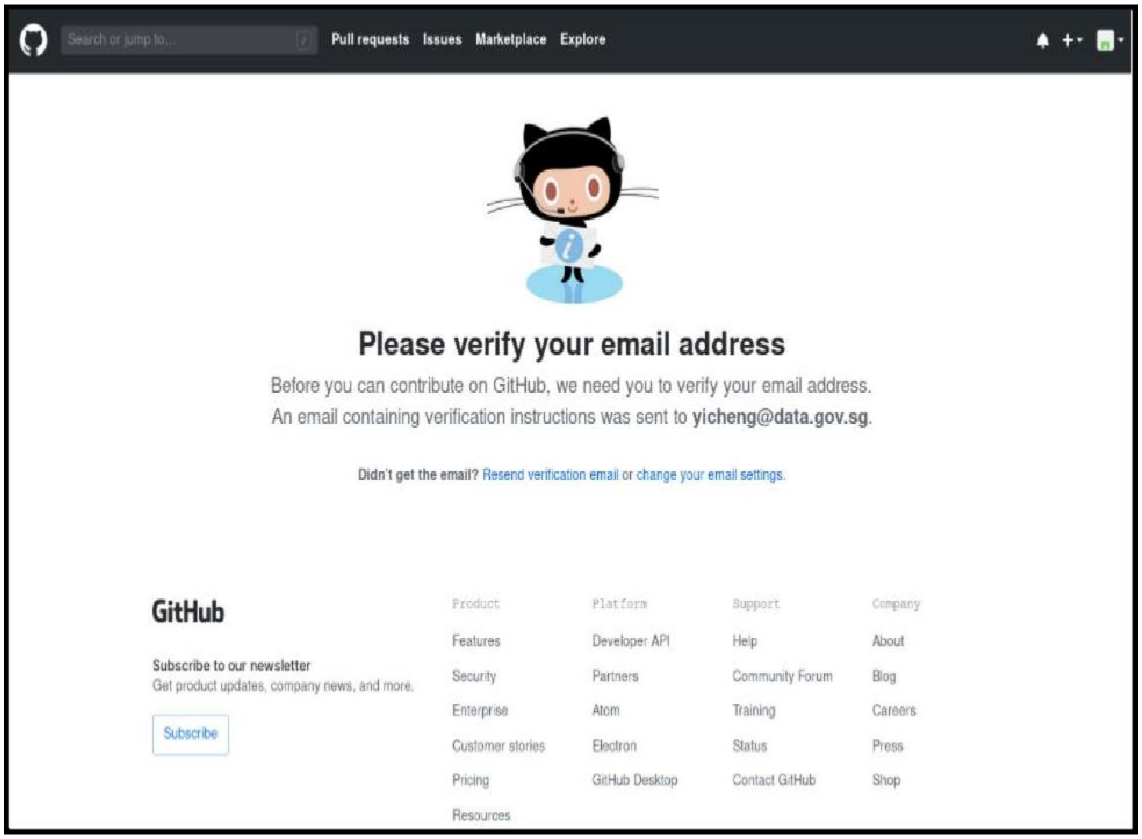
The first steps in starting with GitHub are to create an account, choose a product that fits your needs best, verify your email, set up two-factor authentication, and view your profile. There are several types of accounts on GitHub. Every person who uses GitHub has their own user account, which can be part of multiple organisations and teams. Your user account is your identity on GitHub.com and represents you as an individual.

1. Creating an account: To sign up for an account on GitHub.com, navigate to https://github.com/ and follow the prompts. To keep your GitHub account secure you should use a strong and unique password.


Where the world builds software
Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.
Sign up for GitHub

2. **Choosing your GitHub product:** You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want.

3. **Verifying your email address:** To ensure you can use all the features in your GitHub plan, verify your email address after signing up for a new account.



Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.
An email containing verification instructions was sent to **yicheng@data.gov.sg**.

Didn't get the email? Resend verification email or change your email settings.

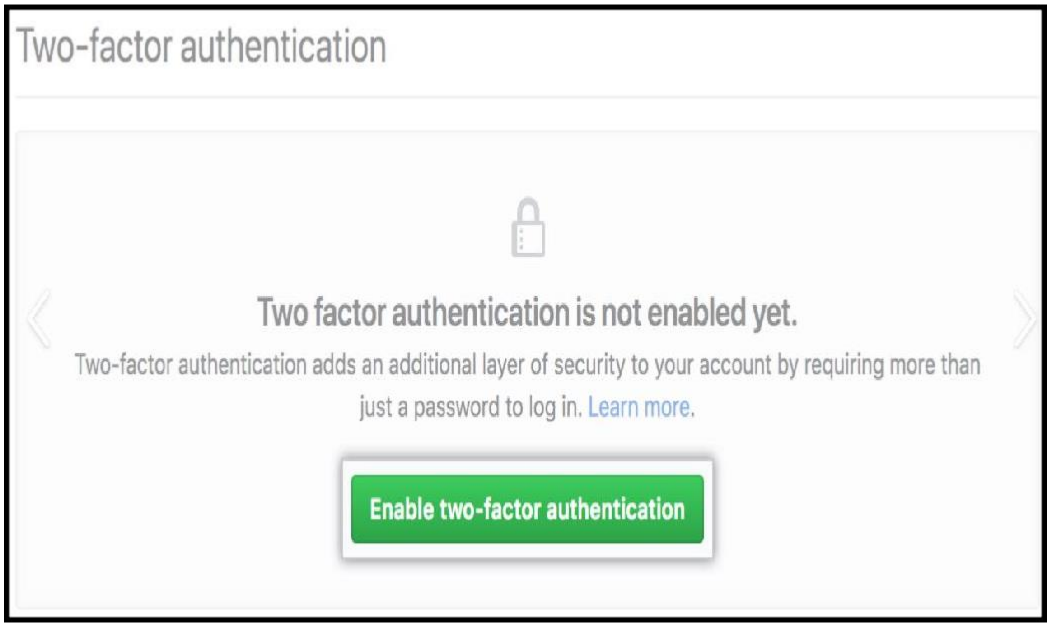
GitHub

Subscribe to our newsletter
Get product updates, company news, and more.

Subscribe

Product	Platform	Support	Company
Features	Developer API	Help	About
Security	Partners	Community Forum	Blog
Enterprise	Atom	Training	Careers
Customer stories	Electron	Status	Press
Pricing	GitHub Desktop	Contact GitHub	Shop
Resources			

4. **Configuring two-factor authentication:** Two-factor authentication, or 2FA, is an extra layer of security used when logging into websites or apps.

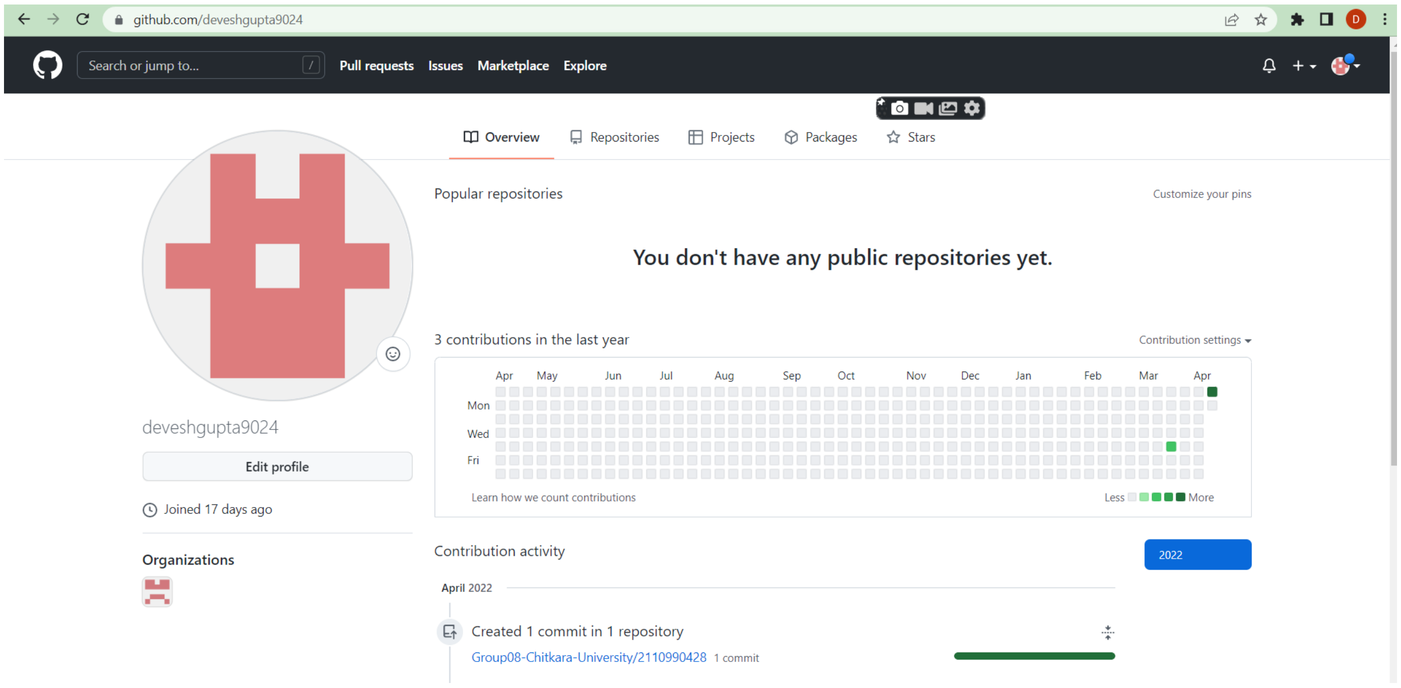


Two-factor authentication

Two factor authentication is not enabled yet.

Two-factor authentication adds an additional layer of security to your account by requiring more than just a password to log in. Learn more.

5. **Viewing your GitHub profile and contribution graph:** Your GitHub profile tells people the story of your work through the repositories and gists you've pinned, the organisation memberships you've chosen to publicize, the contributions you've made, and the projects you've created



deveshgupta9024

Joined 17 days ago

Organizations

Popular repositories

You don't have any public repositories yet.

3 contributions in the last year

Contribution activity

April 2022

Created 1 commit in 1 repository

Group08-Chitkara-University/2110990428 1 commit

CHITKARA UNIVERSITY **Experiment No.03**

Aim: Program to generate logs

Basic Git init :

Git init command creates a new Git repository. It can be used to convert an existing, undersigned project to a Git repository or initialize a new, empty repository. Most other Git commands are not available outside of an initialize repository, so this is usually the first command you'll run ina new project.

Basic Git status :

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show you any information regarding the committed project history.

Basic Git commit :

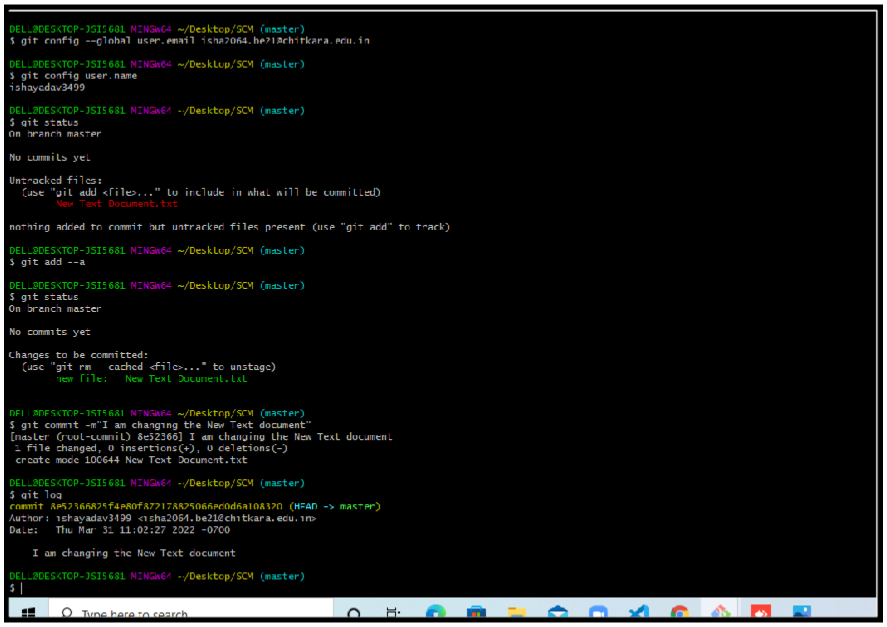
The git commit command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as "safe" versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, The git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit and git add are two of themost frequently used

Basic Git add command :

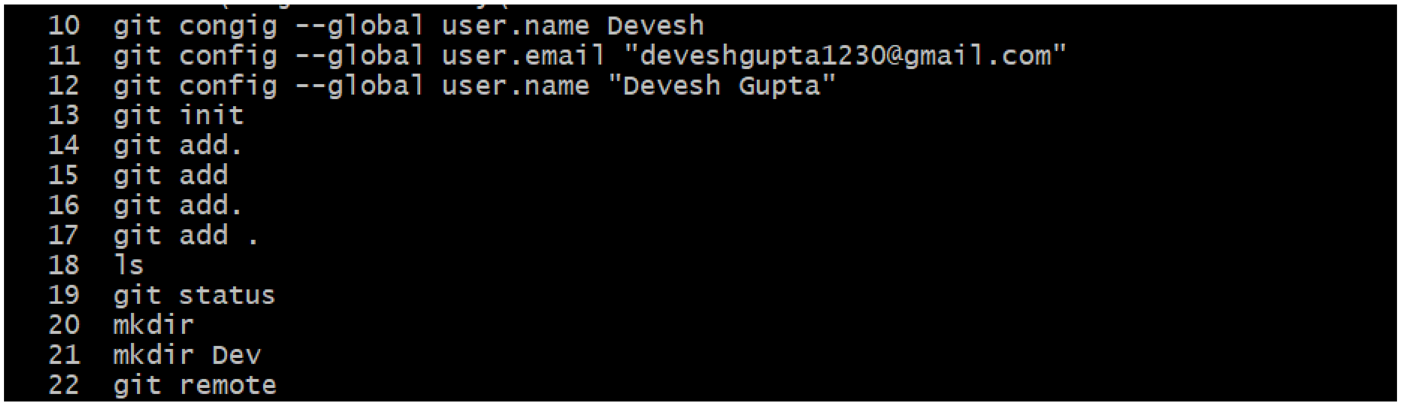
The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit

Basic Git log

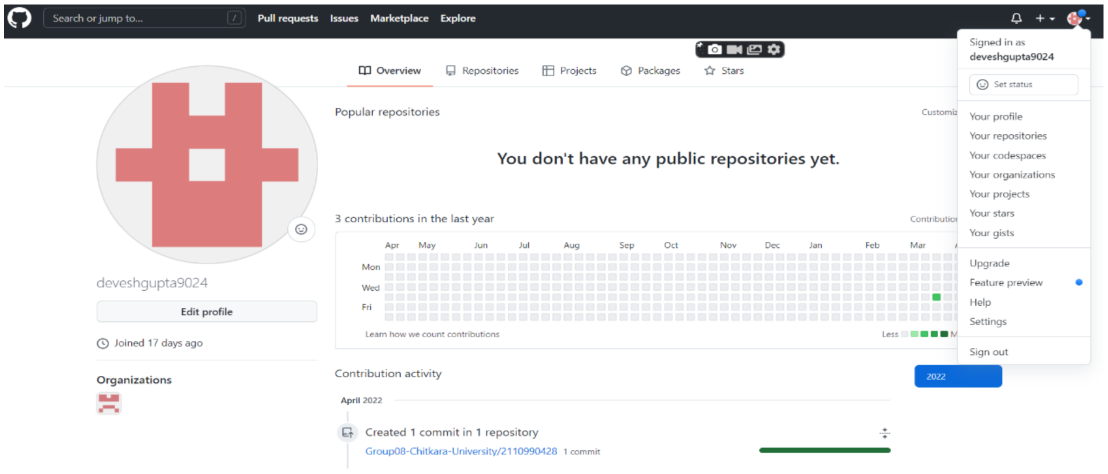
Git log command is one of the most usual commands of git. It is the most useful command for Git. Every time you need to check the history, you have to use the git log command. The basic git log command will display the most recent commits and the status of the head. It will use as



```
DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git config --global user.email isha2064.be21@chitkara.edu.in

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git config user.name
ishayadav3499

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master

No commits yet

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

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

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git add --a

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm   cached <file>..." to unstage)
        new file:   New Text Document.txt


DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git commit -m"I am changing the New Text document"
[master (root-commit) 8e52366] I am changing the New Text document
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 New Text Document.txt

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git log
commit 8e52366825f4e80f87217882506 6ed0d6a108320 (HEAD -> master)
Author: ishayadav3499 <isha2064.be21@chitkara.edu.in>
Date:   Thu Mar 31 11:02:27 2022 -0700

    I am changing the New Text document

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ |
```



```
10  git congig --global user.name Devesh
11  git config --global user.email "deveshgupta1230@gmail.com"
12  git config --global user.name "Devesh Gupta"
13  git init
14  git add.
15  git add
16  git add.
17  git add .
18  ls
19  git status
20  mkdir
21  mkdir Dev
22  git remote
```



Popular repositories

You don't have any public repositories yet.

3 contributions in the last year

Learn how we count contributions

deveshgupta9024

Joined 17 days ago

Organizations

Contribution activity

April 2022

Created 1 commit in 1 repository

Group08-Chitkara-University/2110990428 1 commit

CHITKARA UNIVERSITY **Experiment No.04**

Aim:

Create and visualize branches in Git

How to create branches:

The main branch in git is called master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the files which are created in branch are not shown in master branch. We also can merge both the parent(master) and child (other branches).

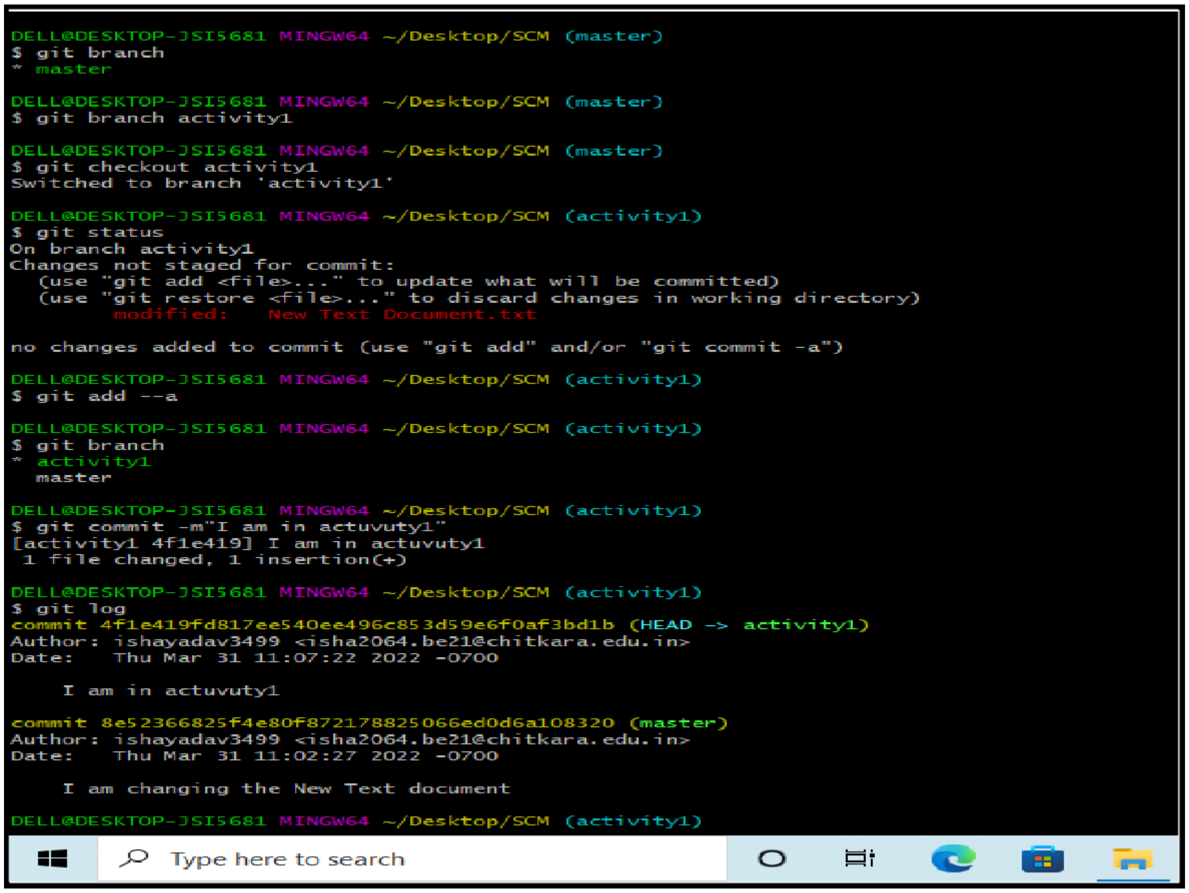
1. For creating a new branch: git branch “name of branch”
2. To check how many branches we have : git branch
3. To change the present working branch: git checkout “name of the branch”

Visualizing Branches:

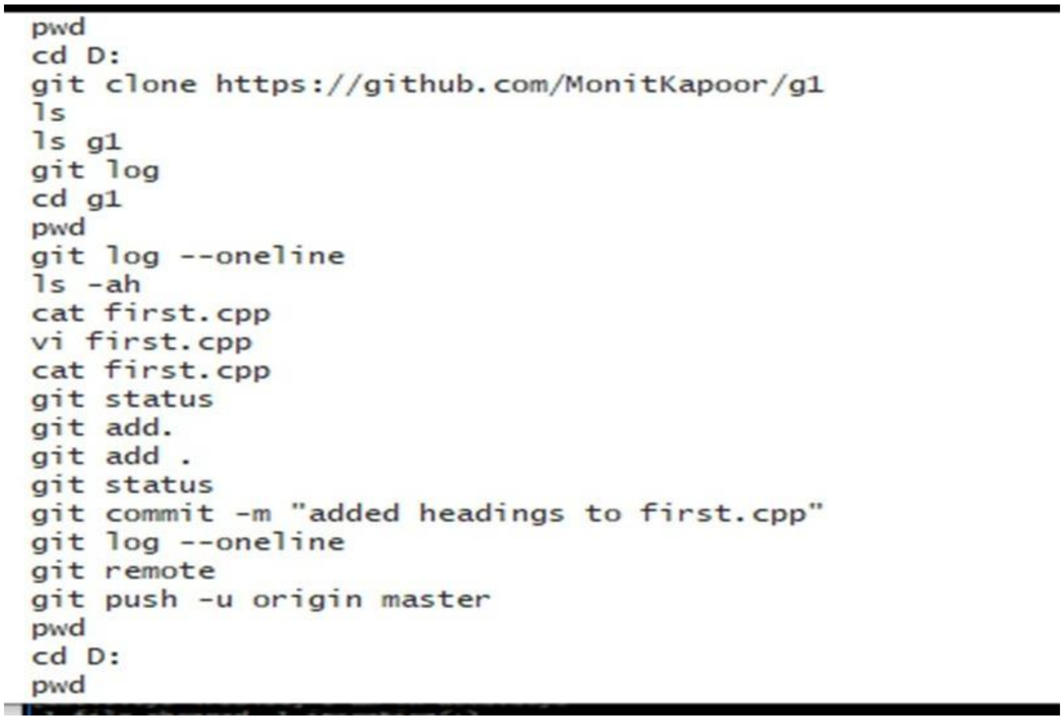
To visualize, we have to create a new file in the new branch “activity1” instead of the master branch. After this we have to do three step architecture i.e. working directory, staging area and git repository.

After this I have done the 3 Step architecture which is tracking the file, send it to stagging area and finally we can rollback to any previously saved version of this file.

After this we will change the branch from activity1 to master, but when we switch back to master branch the file we created i.e “hello” will not be there. Hence the new file will not be shown in the master branch. In this way we can create and change different branches. We can also merge the branches by using the git merge command. In this way we can create and change different branches. We can also merge the branches by using git merge command.



```
DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git branch
* master

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git branch activity1

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (master)
$ git checkout activity1
Switched to branch 'activity1'

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (activity1)
$ git status
On branch activity1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   New Text Document.txt

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

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (activity1)
$ git add --a

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (activity1)
$ git branch
* activity1
  master

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (activity1)
$ git commit -m"I am in actuvuty1"
[activity1 4f1e419] I am in actuvuty1
 1 file changed, 1 insertion(+)

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (activity1)
$ git log
commit 4f1e419fd817ee540ee496c853d59e6f0af3bd1b (HEAD -> activity1)
Author: ishayadav3499 <isha2064.be21@chitkara.edu.in>
Date:   Thu Mar 31 11:07:22 2022 -0700

    I am in actuvuty1

commit 8e52366825f4e80f872178825066ed0d6a108320 (master)
Author: ishayadav3499 <isha2064.be21@chitkara.edu.in>
Date:   Thu Mar 31 11:02:27 2022 -0700

    I am changing the New Text document

DELL@DESKTOP-JSI5681 MINGW64 ~/Desktop/SCM (activity1)
```

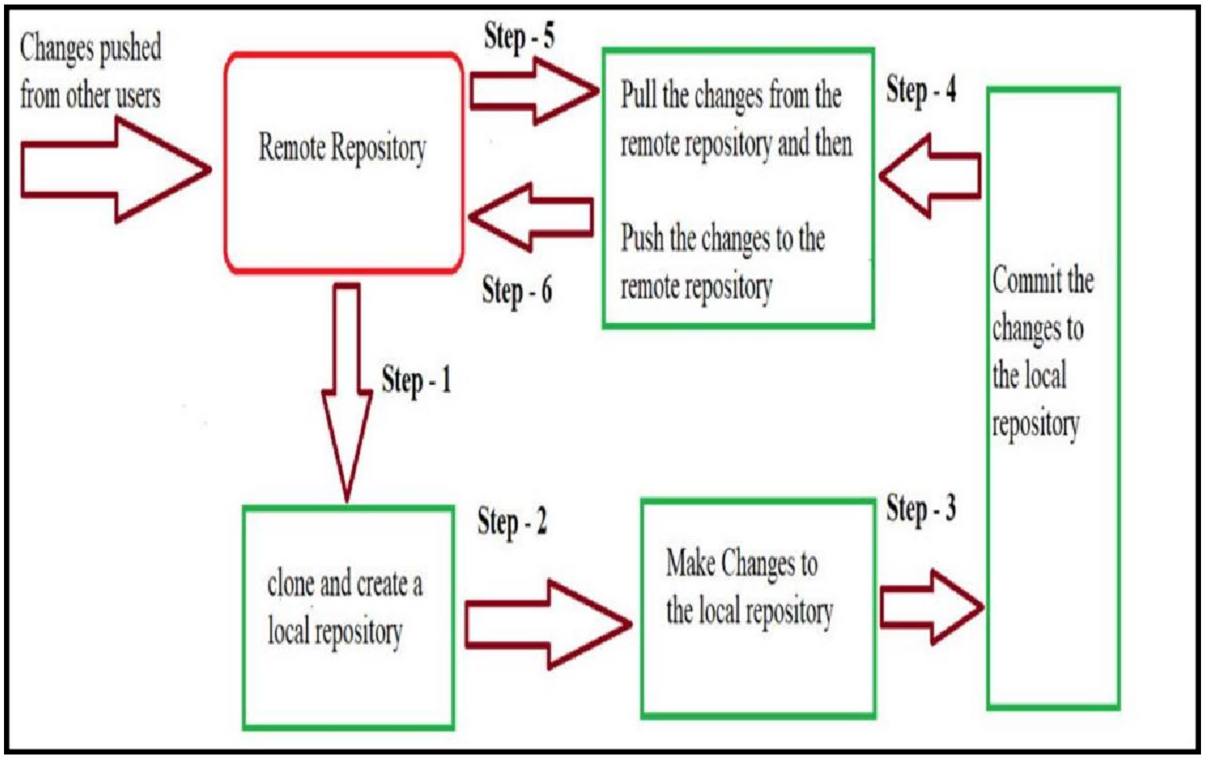


```
pwd
cd D:
git clone https://github.com/MonitKapoor/g1
ls
ls g1
git log
cd g1
pwd
git log --oneline
ls -ah
cat first.cpp
vi first.cpp
cat first.cpp
git status
git add.
git add .
git status
git commit -m "added headings to first.cpp"
git log --oneline
git remote
git push -u origin master
pwd
cd D:
pwd
```

CHITKARA UNIVERSITY **Experiment No.05**

Aim: Git lifecycle description

Git is used in our day-to-day work, we use Git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Lifecycle description that git has and understand more about its life cycle. Let us see some of the basic steps that we have to follow while working with Git-


Changes pushed from other users
Remote Repository
Step - 5
Pull the changes from the remote repository and then
Push the changes to the remote repository
Step - 6
Step - 4
Commit the changes to the local repository
Step - 1
clone and create a local repository
Step - 2
Make Changes to the local repository
Step - 3

• **Step 1**- We first clone any of the code residing in the remote repository to make our won local repository.

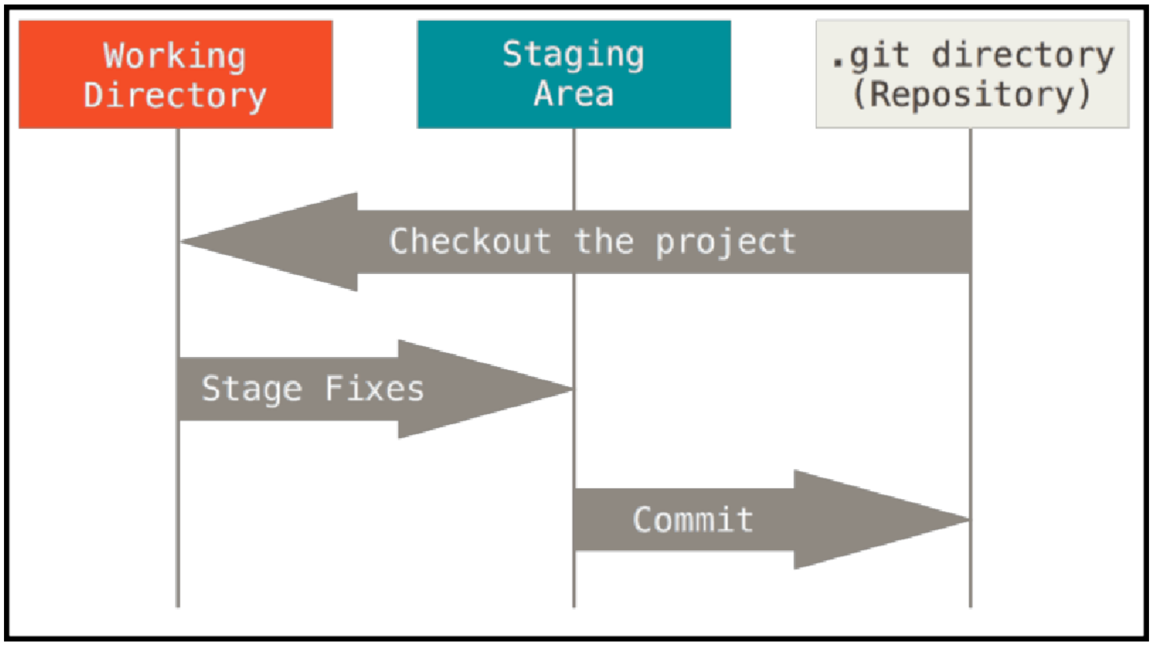
• **Step 2-** We edit the files that we have cloned in our local repository and make the necessary changes in it.

• **Step 3**- We commit our changes by first adding them to our staging area and committing them with a commit message.

• <u>Step 4 and Step 5-</u> We first check whether there are any of the changes done in the remote repository by some other users and we first pull that changes.

• **Step 6-** If there are no changes we push our changes to the remote repository and we are done with our work.

When a directory is made a git repository, there are mainly 3 states which make the essence of Git version Control System. The three states are



1. Working Directory

Whenever we want to initialize aur local project directory to make a Git repository, we use the git init command. After this command, git becomes aware of the files in the project although it does not track the files yet. The files are further tracked in the staging area.

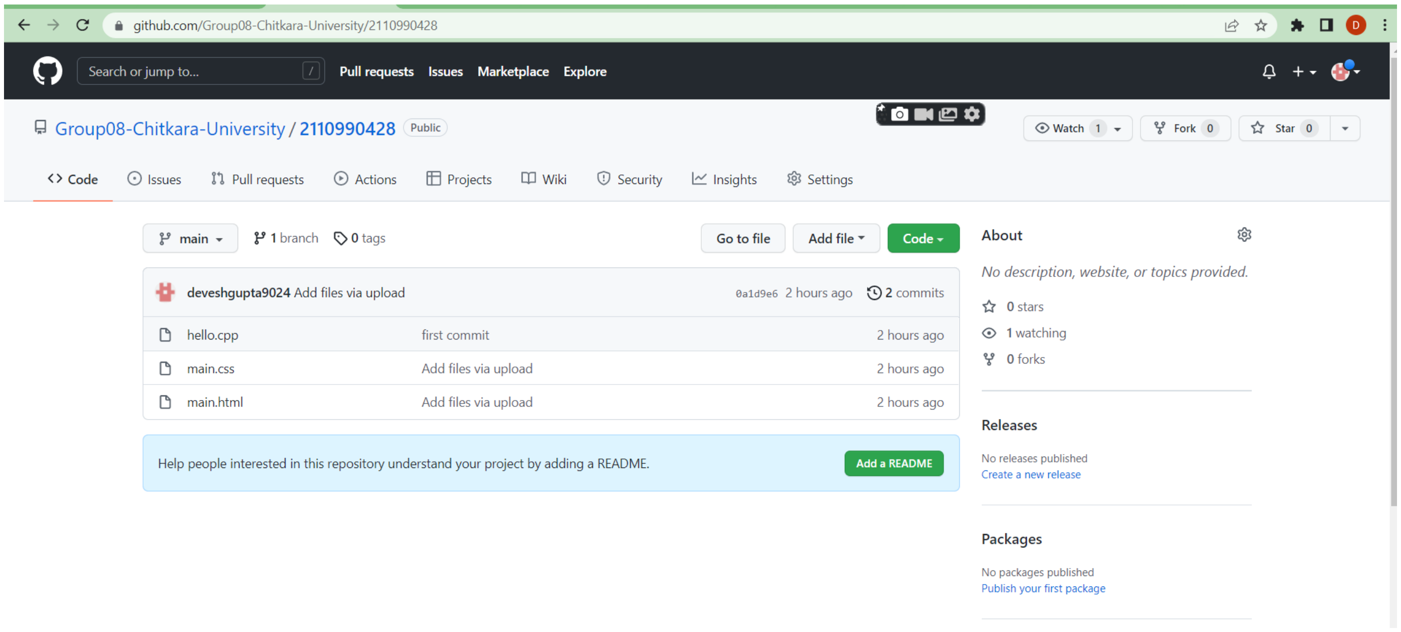
2. Staging Area

Now, to track files the different versions of our files we use the command git add. git add command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to get tracked, examples include node modules, temporary files, etc.

Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the .git folder inside the index file. git add<filename> git add.

3. Git Directory:

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit aur files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message. git commit -m <Message>


github.com/Group08-Chitkara-University/2110990428

Group08-Chitkara-University / 2110990428 Public

deveshgupta9024 Add files via upload 0a1d9e6 2 hours ago 2 commits

hello.cpp	first commit	2 hours ago
main.css	Add files via upload	2 hours ago
main.html	Add files via upload	2 hours ago

Help people interested in this repository understand your project by adding a README.

About

No description, website, or topics provided.

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Subject Name: **Source Code Management**

Subject Code: **CS181**

Cluster: Beta

Department: **CSE**

Submitted By:Devesh Gupta
2110990428
G08

Submitted To:
Monit kapoor

# List of Programs #

CHITKARA
UNIVERSITY
PUNJAB

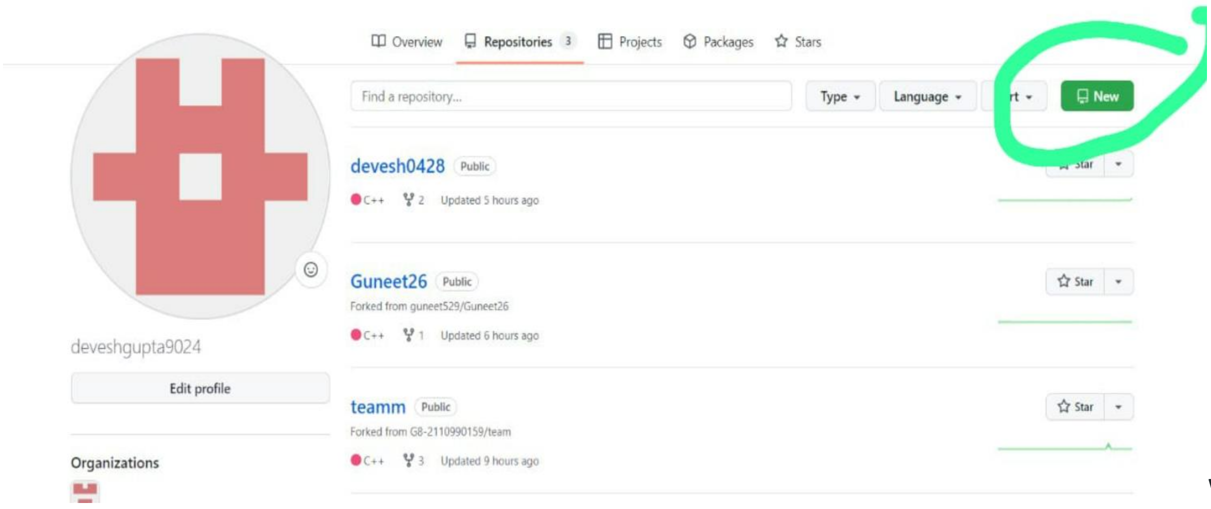
Experiment No. 06

Aim: Add collaborators on Github Repo

Theory:

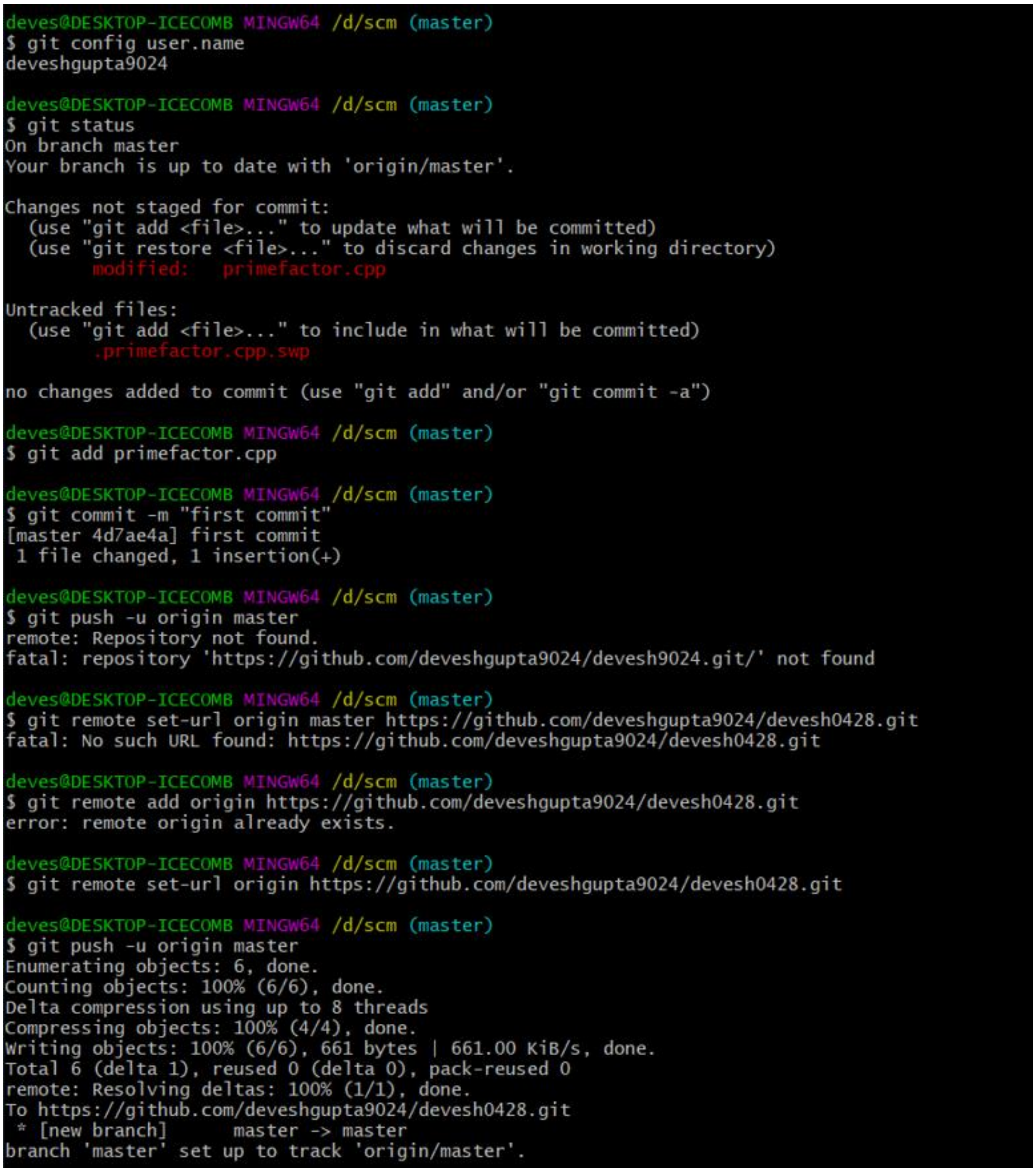
1. Create a New Repository.

   A repository is like a folder for your project. You can have any number of public and private repositories in your user account. Repositories can contain folders and files, images, videos, spreadsheets, and data sets, as well as the revision history for all files in the repository. For more information, see "About repositories."



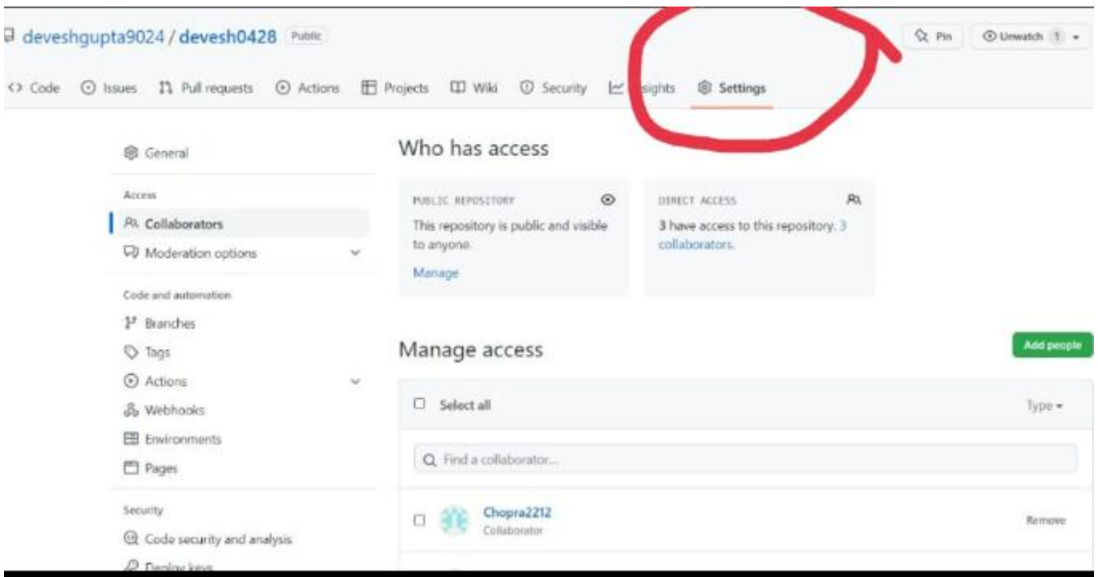
When you create a new repository, you should initialize the repository with a READMEfile to let people know about your project. For more information, see "Creating a new repository."

2. Now Copy the HTTP link of your repo and paste it it on your 'Git CLI', and merge the local repo in remote repo



```
deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git config user.name
deveshgupta9024

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ 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 restore <file>..." to discard changes in working directory)
        modified:   primefactor.cpp

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

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

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git add primefactor.cpp

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git commit -m "first commit"
[master 4d7ae4a] first commit
 1 file changed, 1 insertion(+)

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git push -u origin master
remote: Repository not found.
fatal: repository 'https://github.com/deveshgupta9024/devesh9024.git/' not found

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git remote set-url origin master https://github.com/deveshgupta9024/devesh0428.git
fatal: No such URL found: https://github.com/deveshgupta9024/devesh0428.git

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git remote add origin https://github.com/deveshgupta9024/devesh0428.git
error: remote origin already exists.

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git remote set-url origin https://github.com/deveshgupta9024/devesh0428.git

deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 661 bytes | 661.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/deveshgupta9024/devesh0428.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

3. Go to Collaborators in Repo Setting, Add the username or email of Collaborator you want to add in your Repo.


deveshgupta9024 / devesh0428 Public

Who has access

PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage

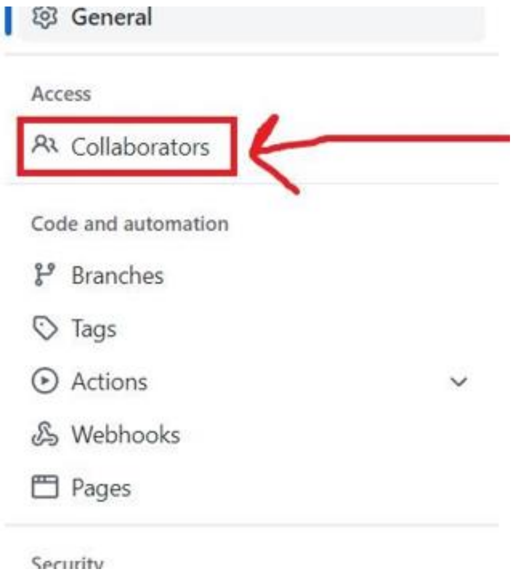
DIRECT ACCESS
3 have access to this repository. 3 collaborators.

Manage access

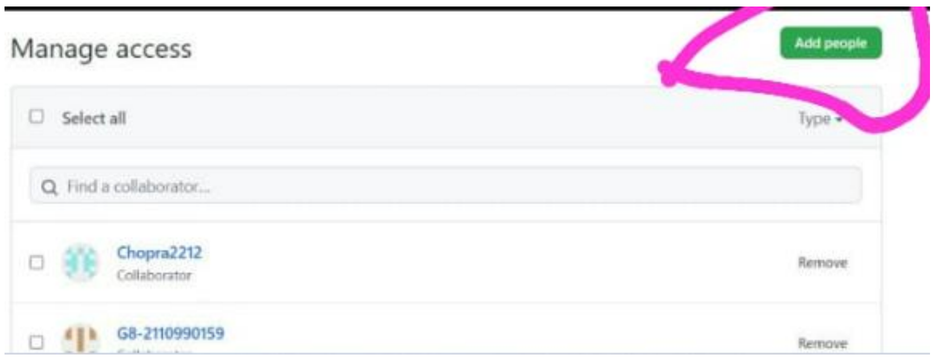
Chopra2212
Collaborator

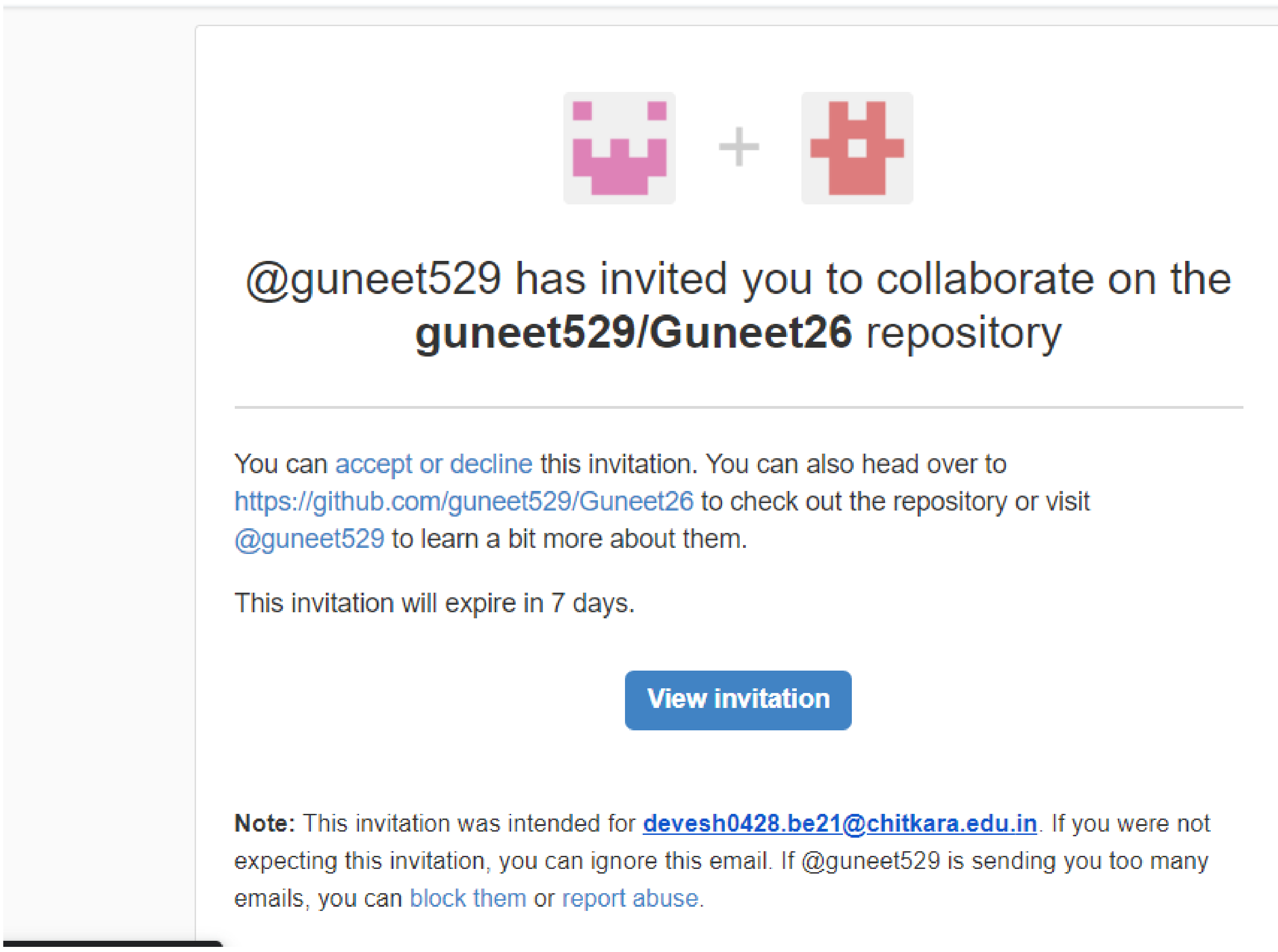
- [redacted] To add members to your repository, open your repository and select settings option in the navigation bar.

- Click on Collaborators option under the access tab.



- After clicking on collaborators GitHub asks you to enter your password to confirm the access to the repository.
- After entering the password you can manage access and add/remove team members to your project.
- To add members click on the add people option and search the id of your respective team member.





@guneet529 has invited you to collaborate on the **guneet529/Guneet26** repository

You can accept or decline this invitation. You can also head over to https://github.com/guneet529/Guneet26 to check out the repository or visit @guneet529 to learn a bit more about them.

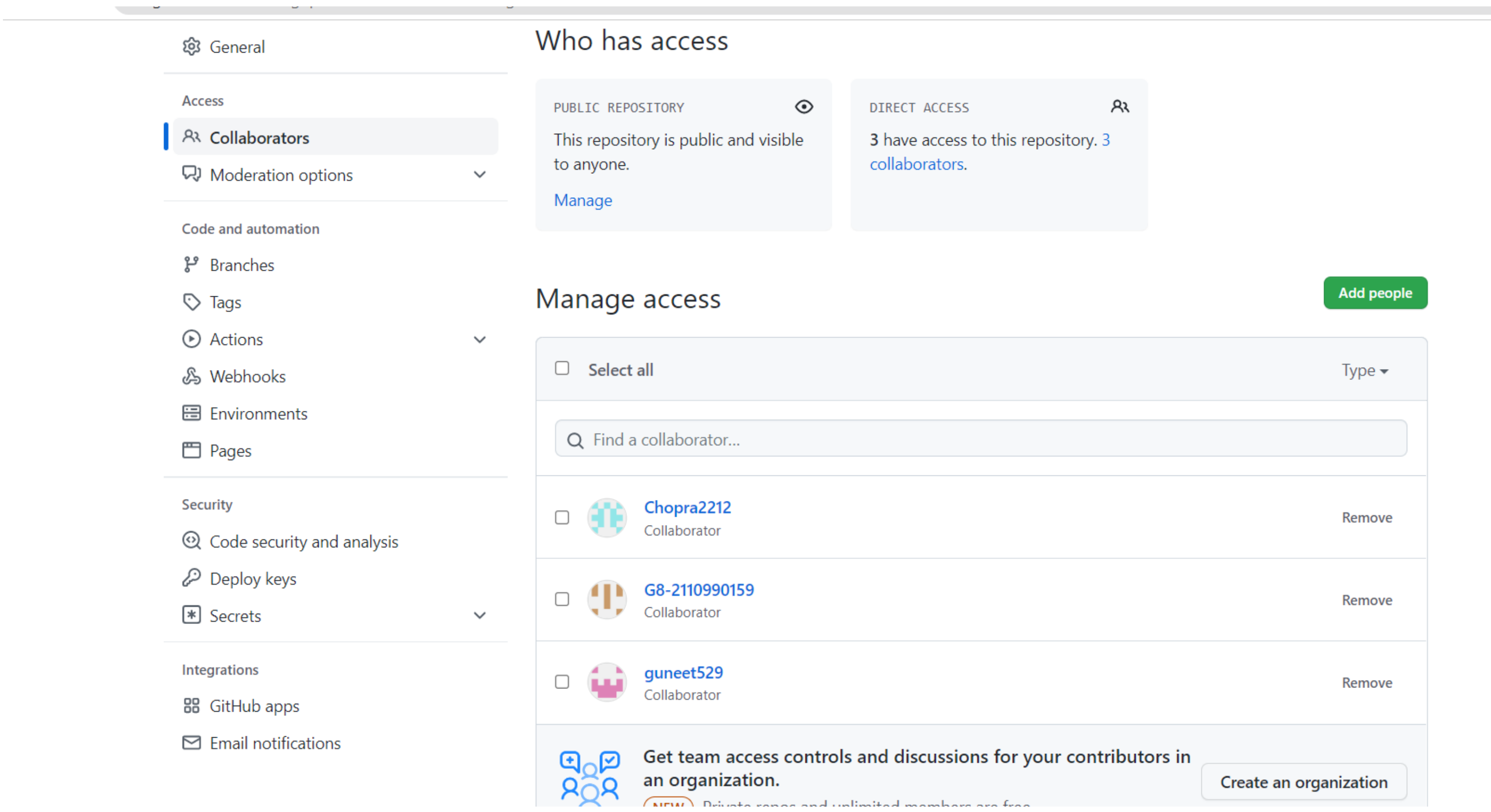
This invitation will expire in 7 days.

View invitation

Note: This invitation was intended for devesh0428.be21@chitkara.edu.in. If you were not expecting this invitation, you can ignore this email. If @guneet529 is sending you too many emails, you can block them or report abuse.

- Invitation Mail is sent to the Collaborator, the collaborator has to accept this Invitation.

- Now Collaborator has now access to the this Repo.


General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Pages

Security

Code security and analysis

Deploy keys

Secrets

Integrations

GitHub apps

Email notifications

Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

Manage

DIRECT ACCESS

3 have access to this repository. 3 collaborators.

Manage access

Add people

Select all

Type

Find a collaborator...

Chopra2212
Collaborator
Remove

G8-2110990159
Collaborator
Remove

guneet529
Collaborator
Remove

Get team access controls and discussions for your contributors in an organization.

Create an organization

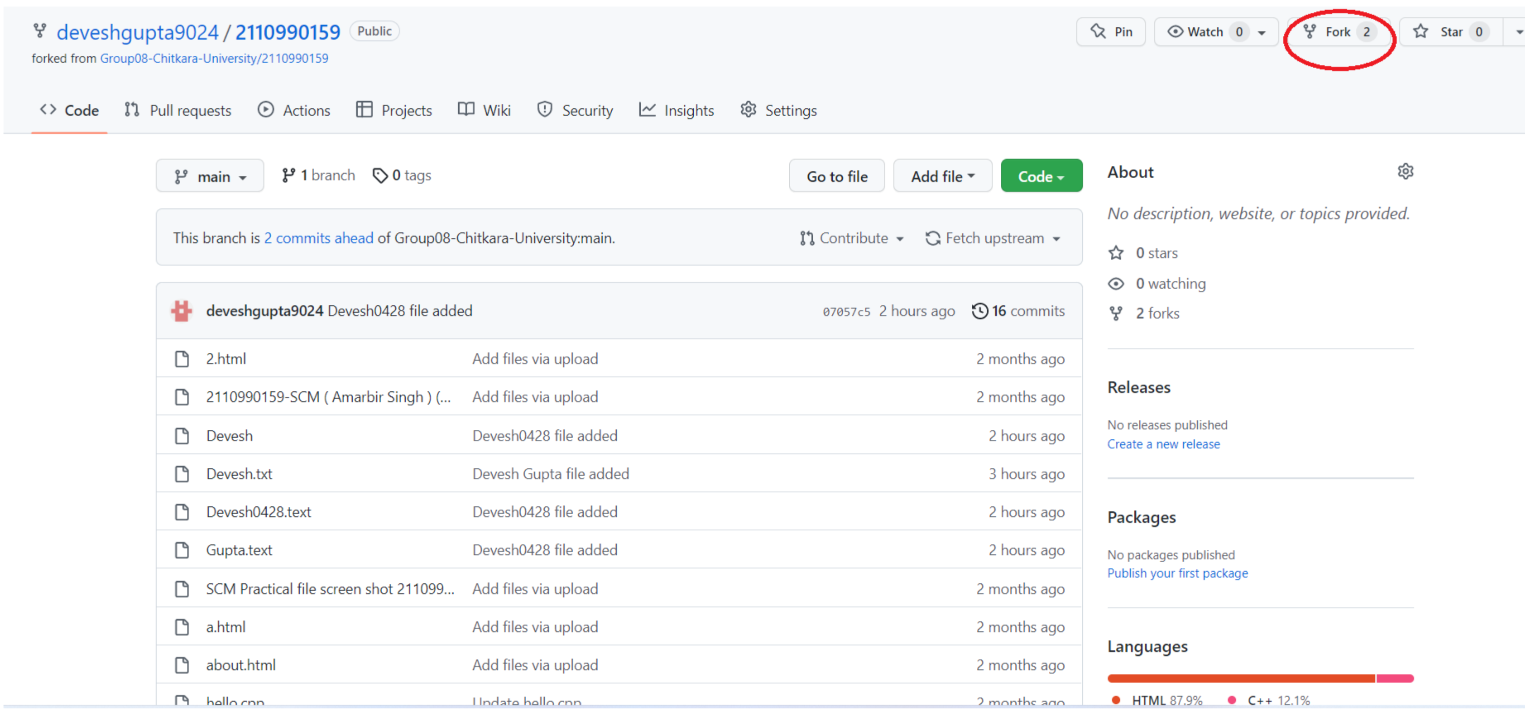
Experiment No. 07

Aim: Fork and Commit

Theory:

Fork :- A Fork is copy of a repository . Forking a repository allows you to freely experiment with changes without affecting the original Project.

1. To fork a repository first thing you need to do is, sign in to your GitHub account and then you came to the repository you want to fork, so here for demo purpose am using panda repository.



deveshgupta9024 / 2110990159 Public
forked from Group08-Chitkara-University/2110990159

This branch is 2 commits ahead of Group08-Chitkara-University:main.

deveshgupta9024 Devesh0428 file added 07057c5 2 hours ago 16 commits

2.html	Add files via upload	2 months ago
2110990159-SCM (Amarbir Singh) (...	Add files via upload	2 months ago
Devesh	Devesh0428 file added	2 hours ago
Devesh.txt	Devesh Gupta file added	3 hours ago
Devesh0428.text	Devesh0428 file added	2 hours ago
Gupta.text	Devesh0428 file added	2 hours ago
SCM Practical file screen shot 211099...	Add files via upload	2 months ago
a.html	Add files via upload	2 months ago
about.html	Add files via upload	2 months ago
hello.cpp	Update hello.cpp	2 months ago

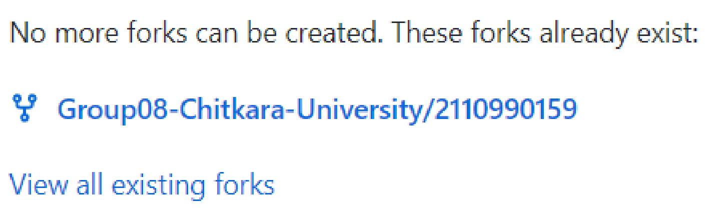
About
No description, website, or topics provided.
0 stars
0 watching
2 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

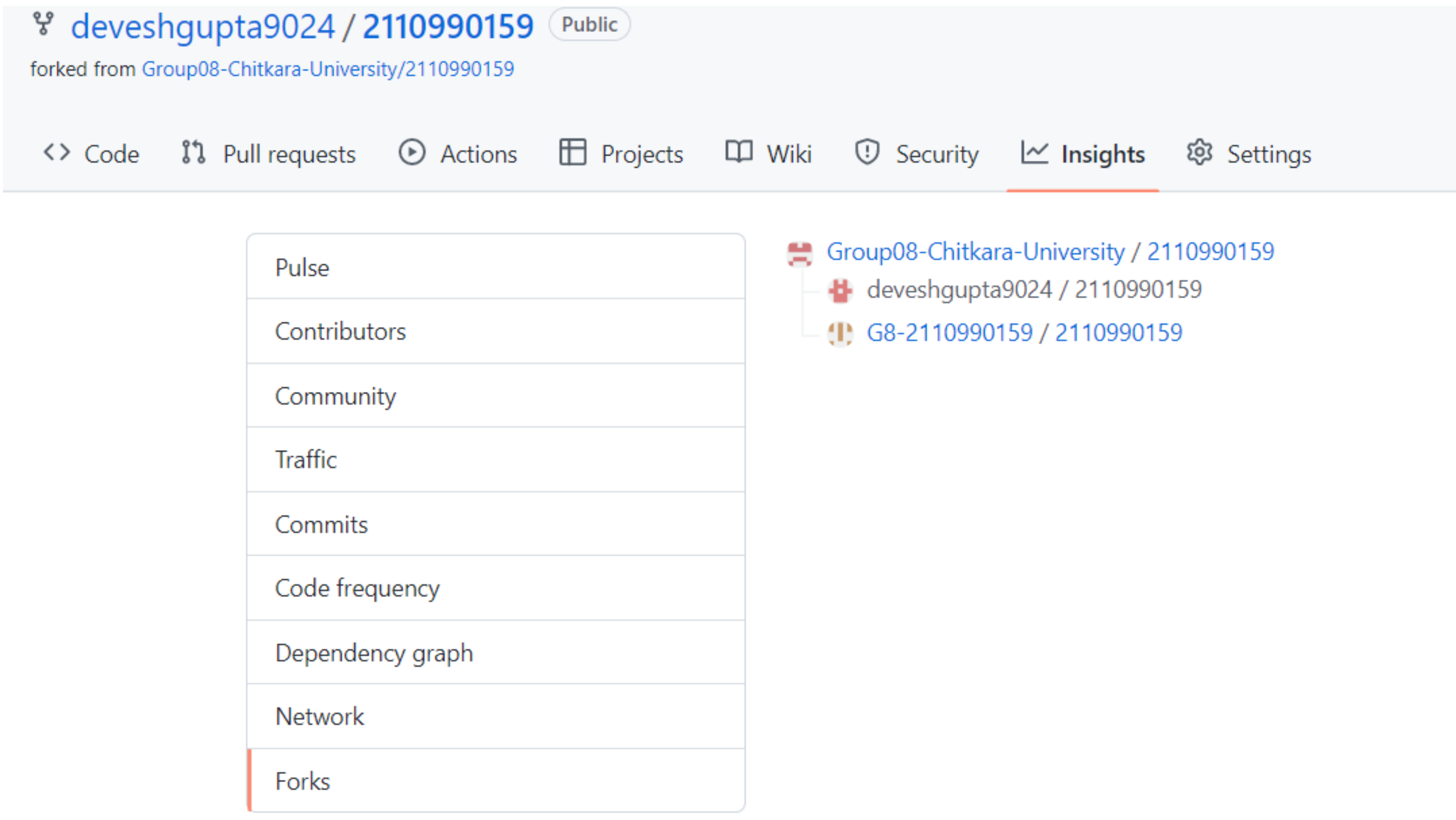
Languages
HTML 87.9% C++ 12.1%

2. Click on the **Fork** button on right upside corner. Then it will ask to create a new fork, add description if you want and then click on create fork.


No more forks can be created. These forks already exist:

Group08-Chitkara-University/2110990159

View all existing forks


deveshgupta9024 / 2110990159 Public

forked from Group08-Chitkara-University/2110990159

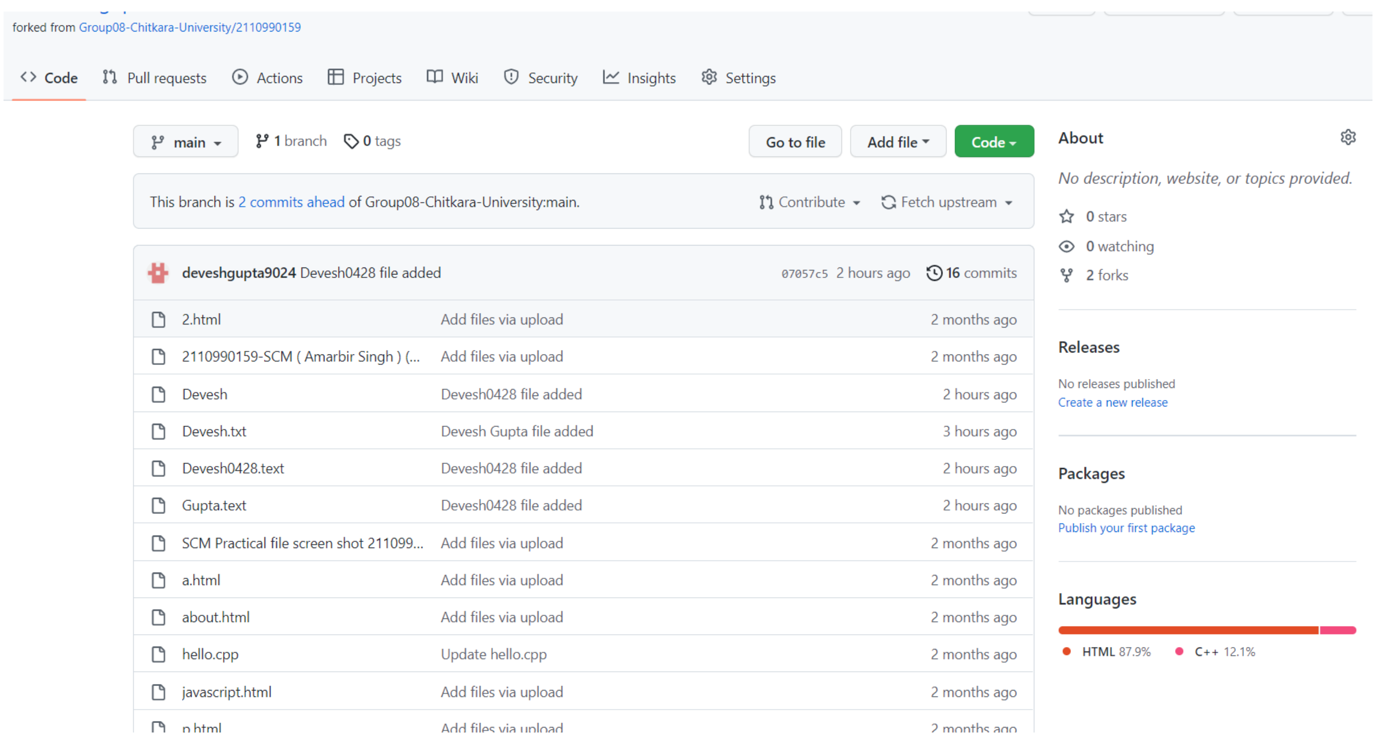
Group08-Chitkara-University / 2110990159

deveshgupta9024 / 2110990159

G8-2110990159 / 2110990159

Note – actually I forgot to take screen shot that time so , now I am posting after fork.

3. Now you will have a copy of the repo you have forked from other user. Now you can do any modification you want without making changes to main source code.


forked from Group08-Chitkara-University/2110990159

This branch is 2 commits ahead of Group08-Chitkara-University:main.

deveshgupta9024 Devesh0428 file added 07057c5 2 hours ago 16 commits

2.html	Add files via upload	2 months ago
2110990159-SCM (Amarbir Singh) (...	Add files via upload	2 months ago
Devesh	Devesh0428 file added	2 hours ago
Devesh.txt	Devesh Gupta file added	3 hours ago
Devesh0428.text	Devesh0428 file added	2 hours ago
Gupta.text	Devesh0428 file added	2 hours ago
SCM Practical file screen shot 211099...	Add files via upload	2 months ago
a.html	Add files via upload	2 months ago
about.html	Add files via upload	2 months ago
hello.cpp	Update hello.cpp	2 months ago
javascript.html	Add files via upload	2 months ago
n.html	Add files via upload	2 months ago

About

No description, website, or topics provided.

0 stars
0 watching
2 forks

Releases

No releases published
Create a new release

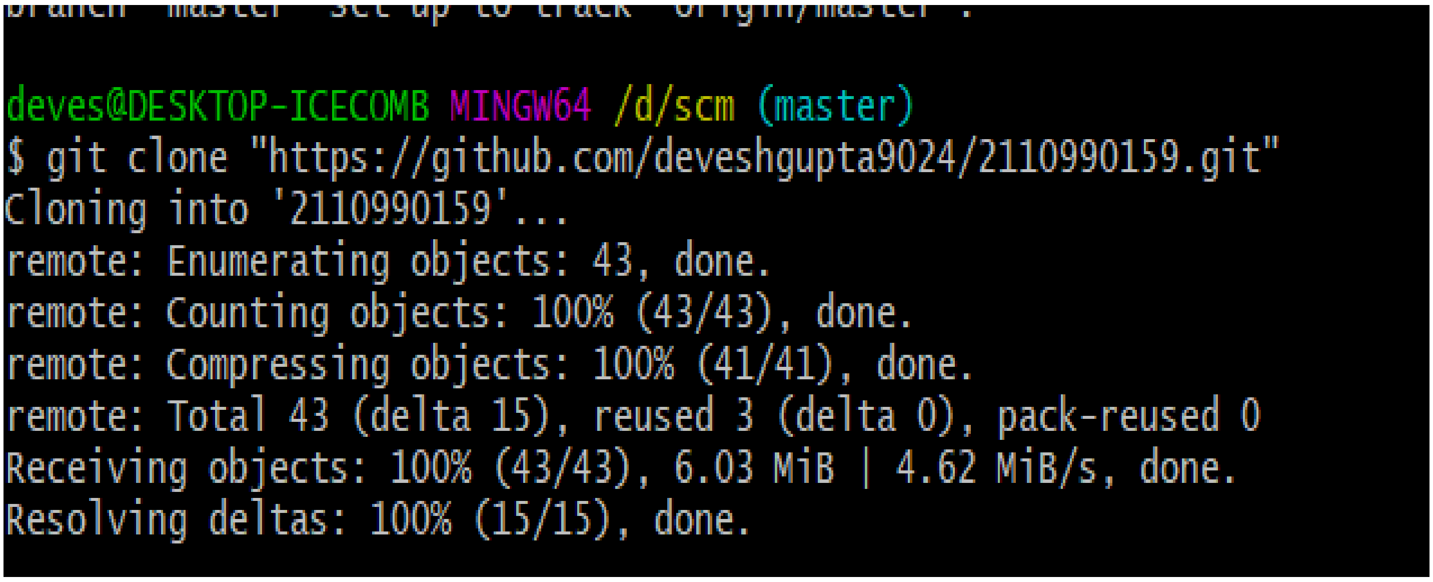
Packages

No packages published
Publish your first package

Languages

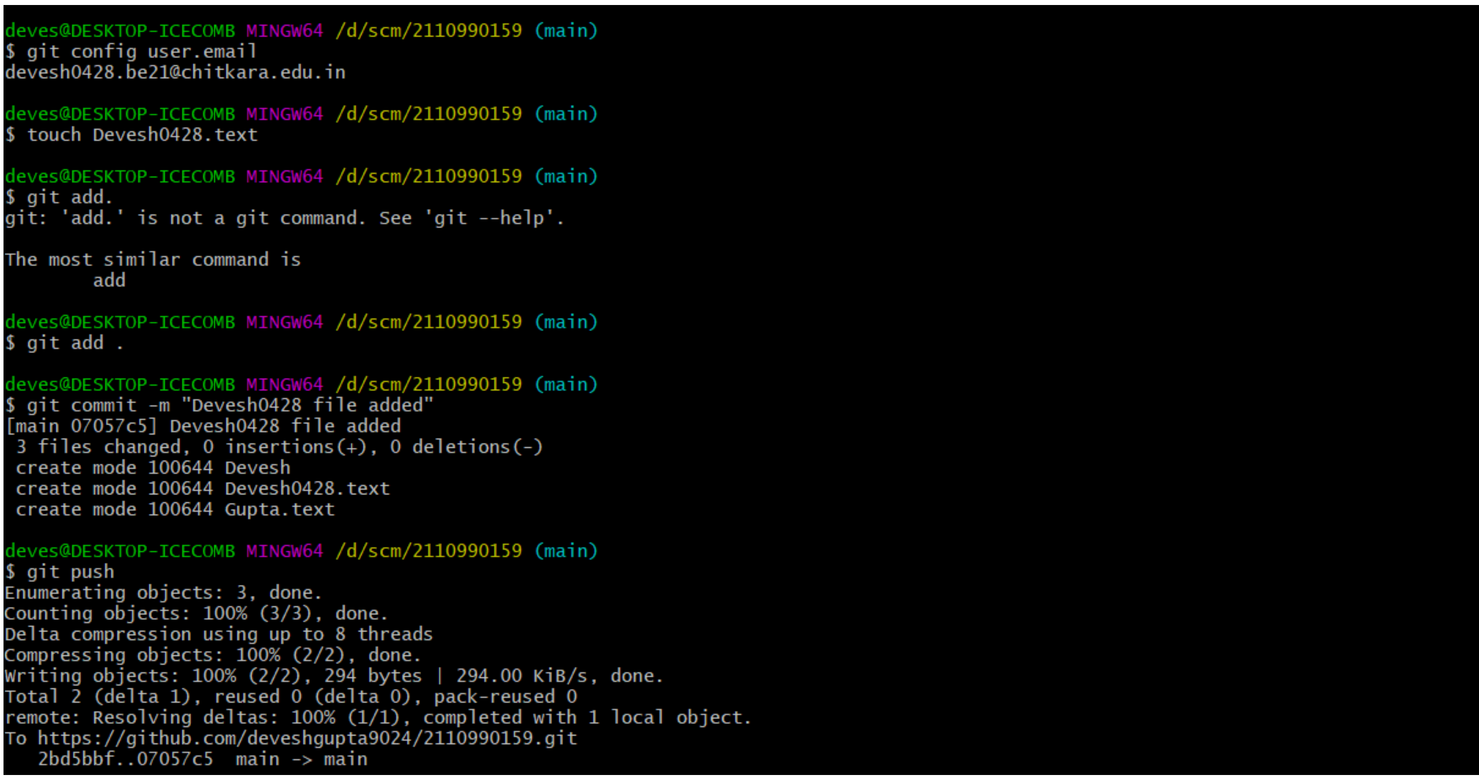
● HTML 87.9% ● C++ 12.1%

4. Now we will clone the forked repo to our pc .



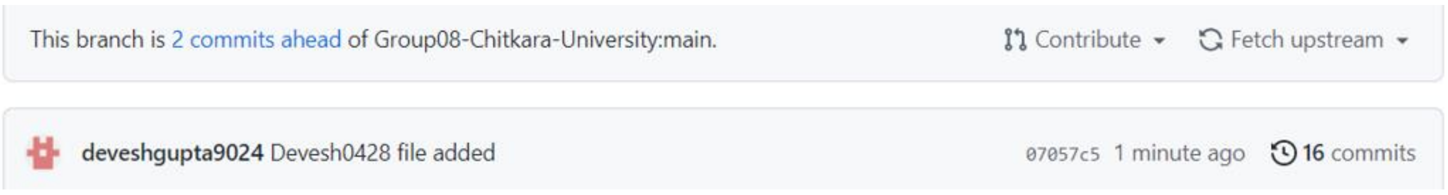
```
deves@DESKTOP-ICECOMB MINGW64 /d/scm (master)
$ git clone "https://github.com/deveshgupta9024/2110990159.git"
Cloning into '2110990159'...
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 43 (delta 15), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (43/43), 6.03 MiB | 4.62 MiB/s, done.
Resolving deltas: 100% (15/15), done.
```

5. Now Open the file make changes in it and commit it and push it to remote.



```
deves@DESKTOP-ICECOMB MINGW64 /d/scm/2110990159 (main)
$ git config user.email
devesh0428.be21@chitkara.edu.in

deves@DESKTOP-ICECOMB MINGW64 /d/scm/2110990159 (main)
$ touch Devesh0428.text

deves@DESKTOP-ICECOMB MINGW64 /d/scm/2110990159 (main)
$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add

deves@DESKTOP-ICECOMB MINGW64 /d/scm/2110990159 (main)
$ git add .

deves@DESKTOP-ICECOMB MINGW64 /d/scm/2110990159 (main)
$ git commit -m "Devesh0428 file added"
[main 07057c5] Devesh0428 file added
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Devesh
 create mode 100644 Devesh0428.text
 create mode 100644 Gupta.text

deves@DESKTOP-ICECOMB MINGW64 /d/scm/2110990159 (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 294 bytes | 294.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/deveshgupta9024/2110990159.git
   2bd5bbf..07057c5  main -> main
```

6. Now you can see your commit in the github the commit is only reflected to your forked repository.


This branch is 2 commits ahead of Group08-Chitkara-University:main.

Contribute | Fetch upstream

deveshgupta9024 Devesh0428 file added 07057c5 1 minute ago **16** commits

Experiment No. 08

Aim: Merge and Resolve Conflicts

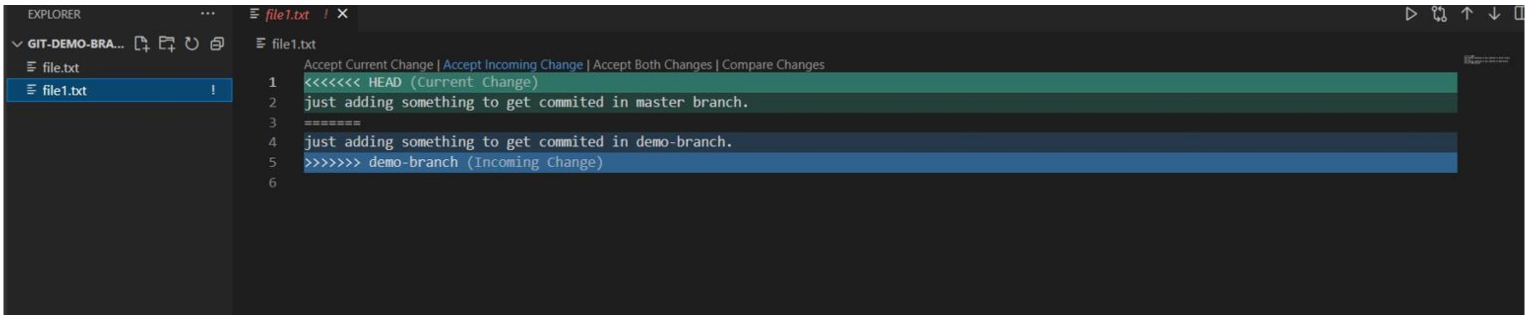
Theory:

1. Do changes in master branch and commit those change. And checkout to different branch and again do changes and commit it. Now checkout to master branch and merge that branch in master.

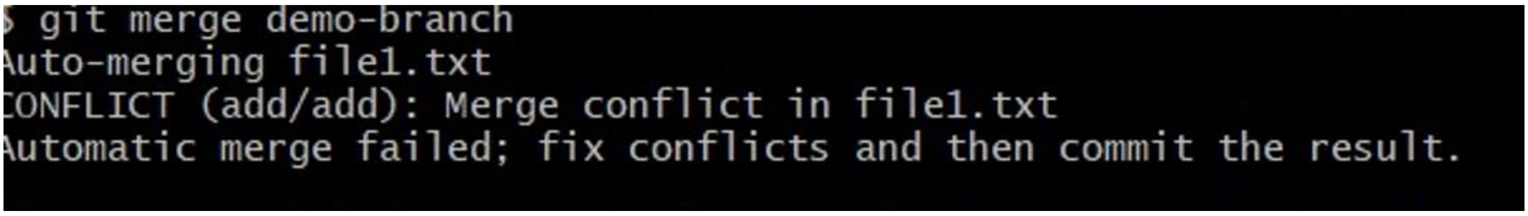
COMMIT IN MASTER BRANCH

COMMIT IN DEMO-BRANCH BRANCH

Now try to merge it will give Conflicts Error
Use Vs code to solve the conflict


EXPLORER
GIT-DEMO-BRA...
file.txt
file1.txt
file1.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

```
<<<<<<< HEAD (Current Change)
just adding something to get commited in master branch.
=======
just adding something to get commited in demo-branch.
>>>>>>> demo-branch (Incoming Change)
```



```
$ git merge demo-branch
Auto-merging file1.txt
CONFLICT (add/add): Merge conflict in file1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

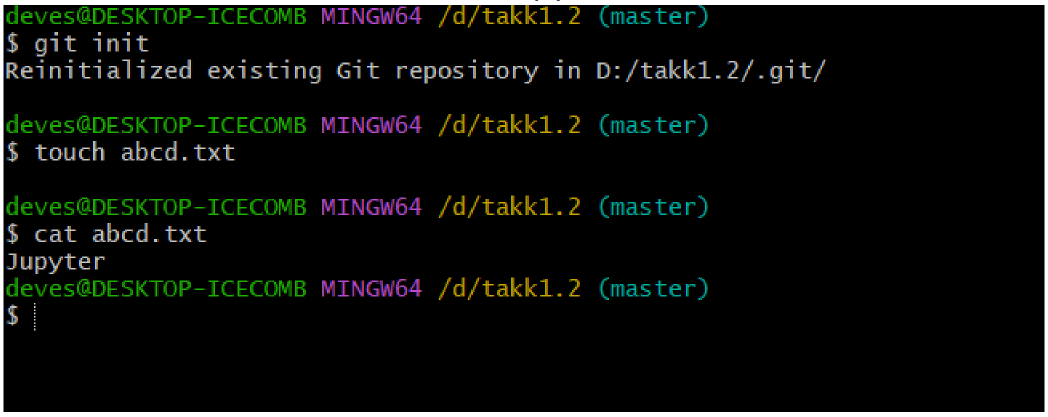
Experiment No. 09

Aim: Reset and Revert

Theory:

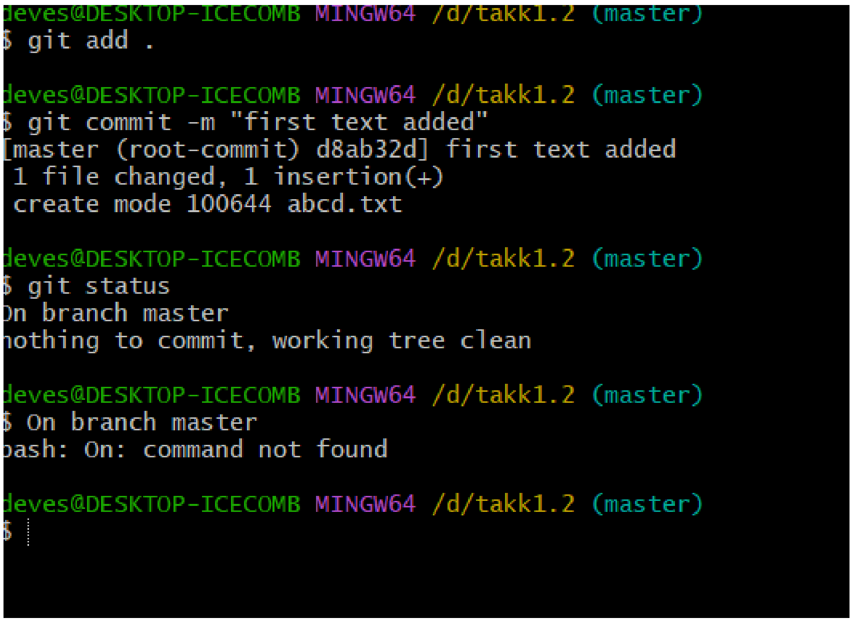
AIM: RESET and REVERT While Working with Git in certain situations we want to undo changes in the working area or index area, sometimes remove commits locally or remotely and we need to reverse those changes. There are 3 different ways in which we can undo the changes in our repository, these are git checkout, git reset, and git revert. Git checkout and git reset in fact can be used to manipulate commits or individual files. These commands can be confusing so it's important to find out the difference between them and to know which command should be used at a particular point of time.

Let us create a file abcd.txt and "Jupyter " written inside it.



```
deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git init
Reinitialized existing Git repository in D:/takk1.2/.git/

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ touch abcd.txt

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ cat abcd.txt
Jupyter
deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

We can see that we have done a single commit and the text document that has been committed with “Python” in it. These updates are currently in the working area and to see them we will see those using git status.

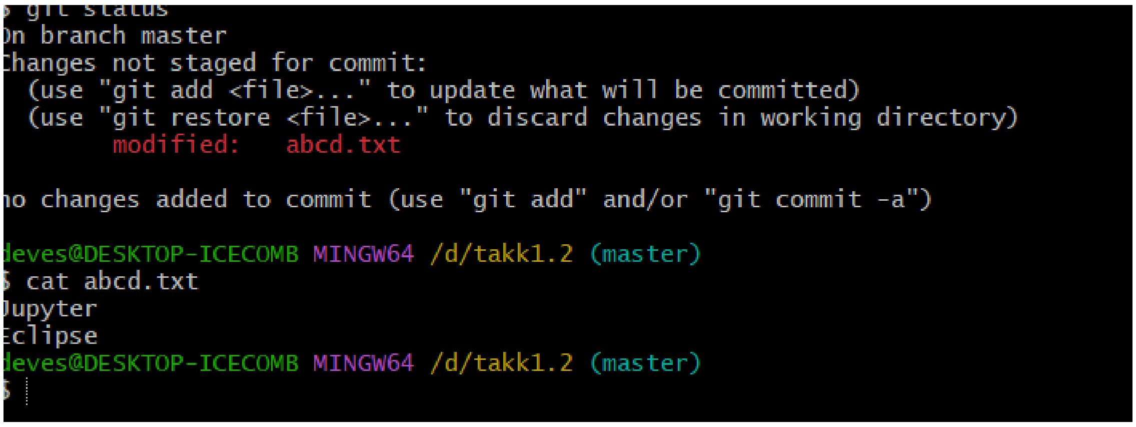


```
leves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git add .

leves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git commit -m "first text added"
[master (root-commit) d8ab32d] first text added
 1 file changed, 1 insertion(+)
 create mode 100644 abcd.txt

leves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
nothing to commit, working tree clean

leves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ On branch master
bash: On: command not found

leves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

Now we have a change “java” which is untracked in our working repository and we need to discard this change. So, the command that we should use here is

Git checkout

git checkout is used to discard the changes in the working repository. git checkout When we write git checkout command and see the status of our git repository and also the text document we can see that our changes are being discarded from the working directory and we are again back to the test document that we had before



```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   abcd.txt

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

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ cat abcd.txt
Jupyter
Eclipse
deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

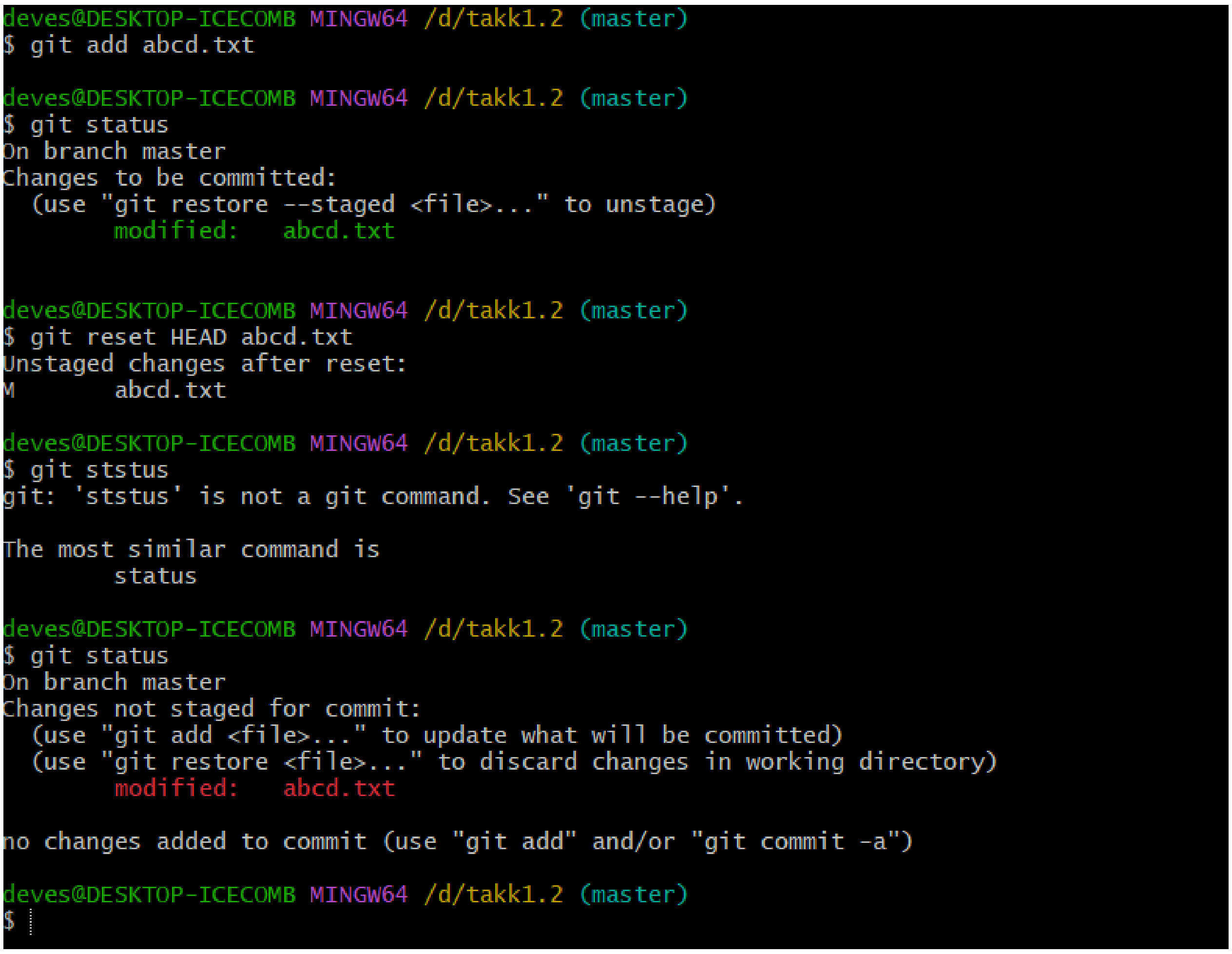
We want to unstage the file and the command that we would be using to unstage our file is –

Git reset

git reset is used when we want to unstage a file and bring our changes back to the working directory. git reset can also be used to remove commits from the local repository

git reset HEAD

Whenever we unstage a file, all the changes are kept in the working area

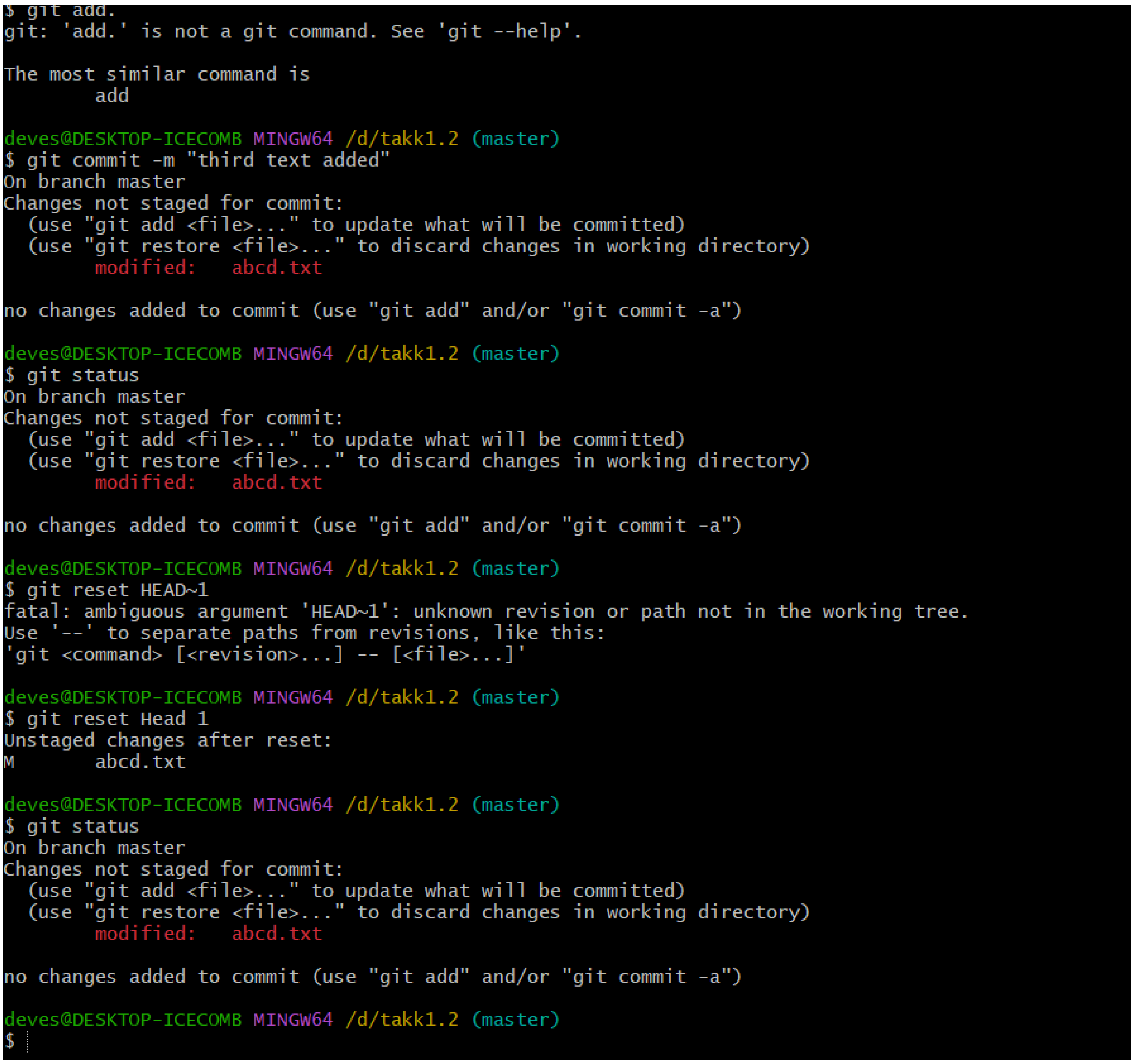


```
deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git add abcd.txt

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   abcd.txt


deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git reset HEAD abcd.txt
Unstaged changes after reset:
M       abcd.txt

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git ststus
git: 'ststus' is not a git command. See 'git --help'.

The most similar command is
        status

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   abcd.txt

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

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

We are back to the working directory, where our changes are present but the file is now unstaged. Now there are also some commits that we don't want to get committed and we want to remove them from our local repository. To see how to remove the commit from our local repository let's stage and commit the changes that we just did and then remove that commit We have 2 commits now, with the latest being the Added “another programming lang” commit which we are going to remove. The command that we would be using now is

Git reset HEAD~1: Points to be noted –

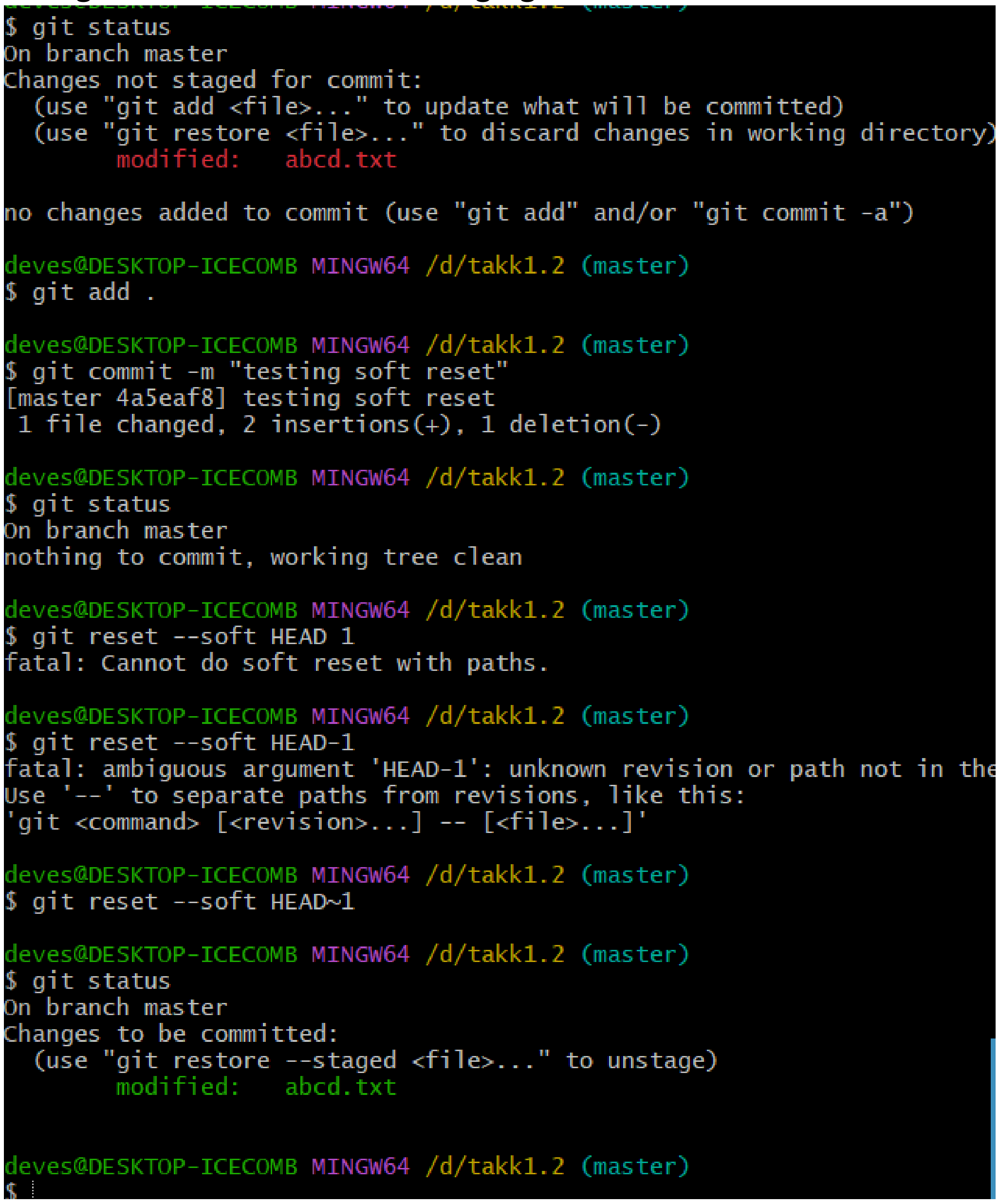
HEAD~1 here means that we are going to remove the topmost commit or the latest commit that we have done.
We cannot remove a specific commit with the help of git reset, for example: we cannot say that we want to remove the second commit or the third commit, we can only remove latest commit or latest 2 commits ... latest N commits. (HEAD~n) [n here means n recent commits that needs to be deleted].



```
$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git commit -m "third text added"
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   abcd.txt

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

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   abcd.txt

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

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git reset HEAD~1
fatal: ambiguous argument 'HEAD~1': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git reset Head 1
Unstaged changes after reset:
M       abcd.txt

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   abcd.txt

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

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

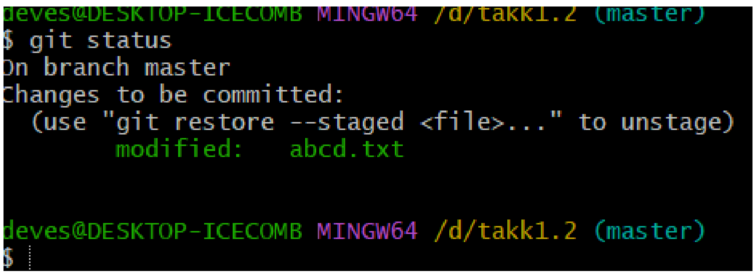
After using the above command we can see that our commit is being deleted and also our file is again unstaged and is back to the working directory. There are different ways in which git reset can actually keep your changes.

git reset --soft HEAD~1 - This command will remove the commit but would not unstage a file. Our changes still would be in the staging area.



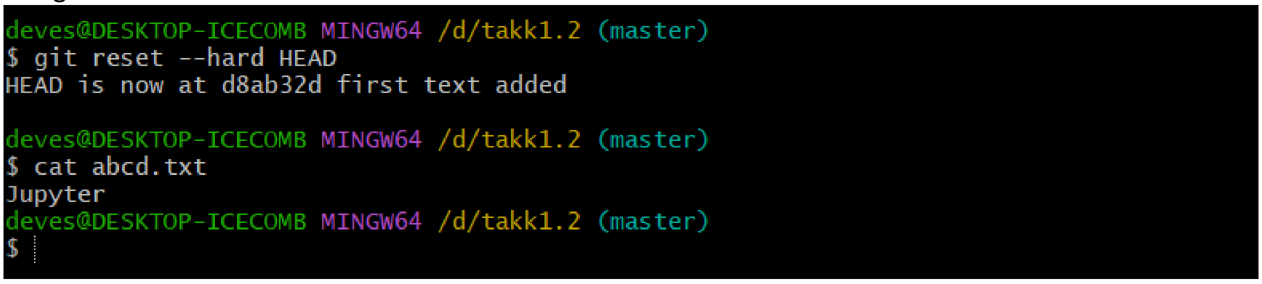
```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   abcd.txt

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

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git add .

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git commit -m "testing soft reset"
[master 4a5eaf8] testing soft reset
 1 file changed, 2 insertions(+), 1 deletion(-)

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
nothing to commit, working tree clean

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git reset --soft HEAD 1
fatal: Cannot do soft reset with paths.

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git reset --soft HEAD-1
fatal: ambiguous argument 'HEAD-1': unknown revision or path not in the
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git reset --soft HEAD~1

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   abcd.txt


deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

git reset --mixed HEAD~1 or git reset HEAD~1 - This is the default command that we have used in the above example which removes the commit as well as unstages the file and our changes are stored in the working directory



```
deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   abcd.txt


deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

<u>git reset --hard HEAD~1</u> - This command removes the commit as well as the changes from your working directory. This command can also be called destructive command as we would not be able to get back the changes so be careful while using this command.



```
deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ git reset --hard HEAD
HEAD is now at d8ab32d first text added

deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$ cat abcd.txt
Jupyter
deves@DESKTOP-ICECOMB MINGW64 /d/takk1.2 (master)
$
```

Points to keep in mind while using git reset command - If our commits are not published to remote repository, then we can use git reset.
Use git reset only for removing commits that are present in our local directory and not in remote directory.
We cannot remove a specific commit with the help of git reset, for eg: we cannot say that we want to remove the second commit or the third commit, we can only remove latest commit or latest 2 commits ... latest N commits. (HEAD~n) [n here means n recent commits that needs to be deleted]

We just discussed above that the git reset command cannot be used to delete commits from the remote repository, then how do we remove the unwanted commits from the remote repository The command that we use here is

Git revert

git revert is used to remove the commits from the remote repository. Since now our changes are in the working directory, let's add those changes to the staging area and commit them.
Now let's push our changes to the remote repository.
Now we want to delete the commit that we just added to the remote repository. We could have used the git reset command but that would have deleted the commit just from the local repository and not the remote repository. If we do this then we would get conflict that the remote commit is not present locally. So, we do not use git reset here. The best we can use here is git revert.

Things to remember – • Using git revert we can undo any commit, not like git reset where we could just remove “n” recent commits. Now let’s first understand what git revert does, git revert removes the commit that we have done but adds one more commit which tells us that the revert has been done. Let’s look at the example –
Note: If you see the lines as we got after the git revert command, just visit your default text editor for git and commit the message from there, or it could directly take to you your default editor. We want a message here because when using git revert, it does not delete the commit instead makes a new commit that contains the removed changes from the commit. We can see that the new commit is being added.
However since this commit is in local repository so we need to do git push so that our remote repository also notices that the change has been done. And as we can see we have a new commit in our remote repository and C++ which we added in our 2nd commit is being removed from the local as well as the remote repositor

Project SCM

Submitted by:-
Devesh Gupta and team
2110990428

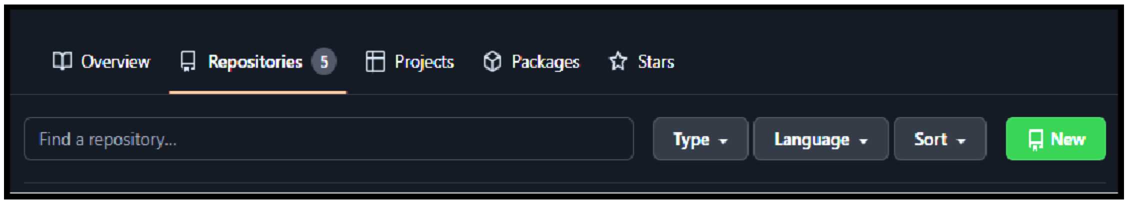
Submitted to:-
Dr.Monit Kapoor

Create a distributed Repository and add members in project team

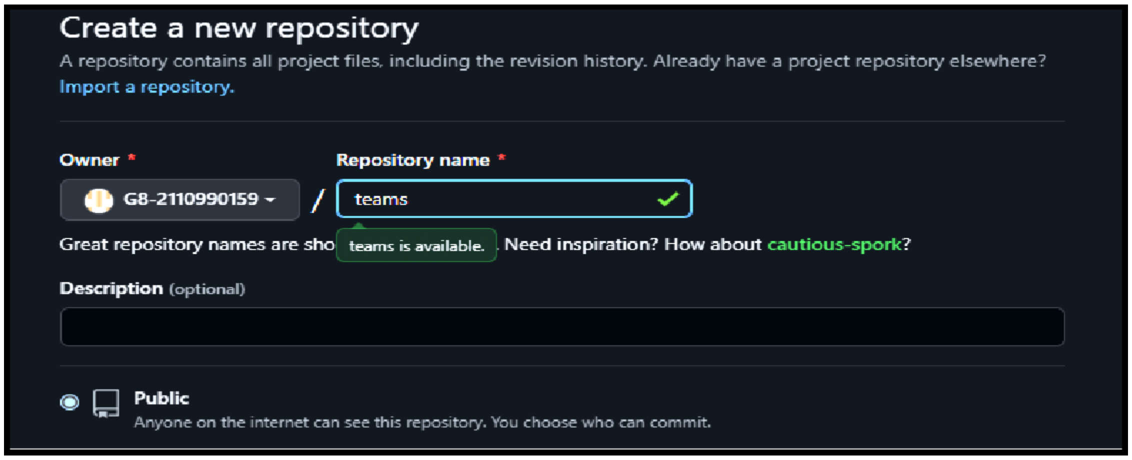
YOU HAVE TO MAKE A REPOSITORY IN GITHUB.

Step1) CREATING REPOSITORY IN GITHUB

The page looks like as: -



By clicking on new you are able to make a new repository.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

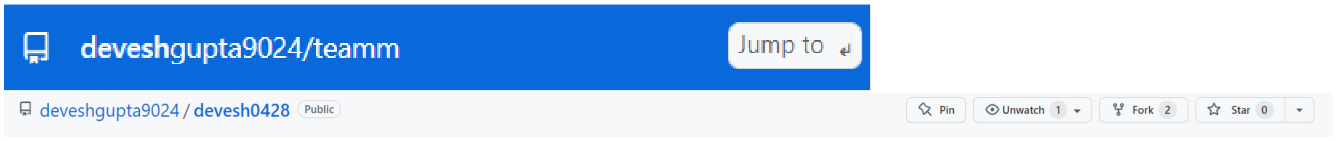
Owner * G8-2110990159 / Repository name * teams

Great repository names are sho teams is available. Need inspiration? How about cautious-spork?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Write the repository name and click on next.

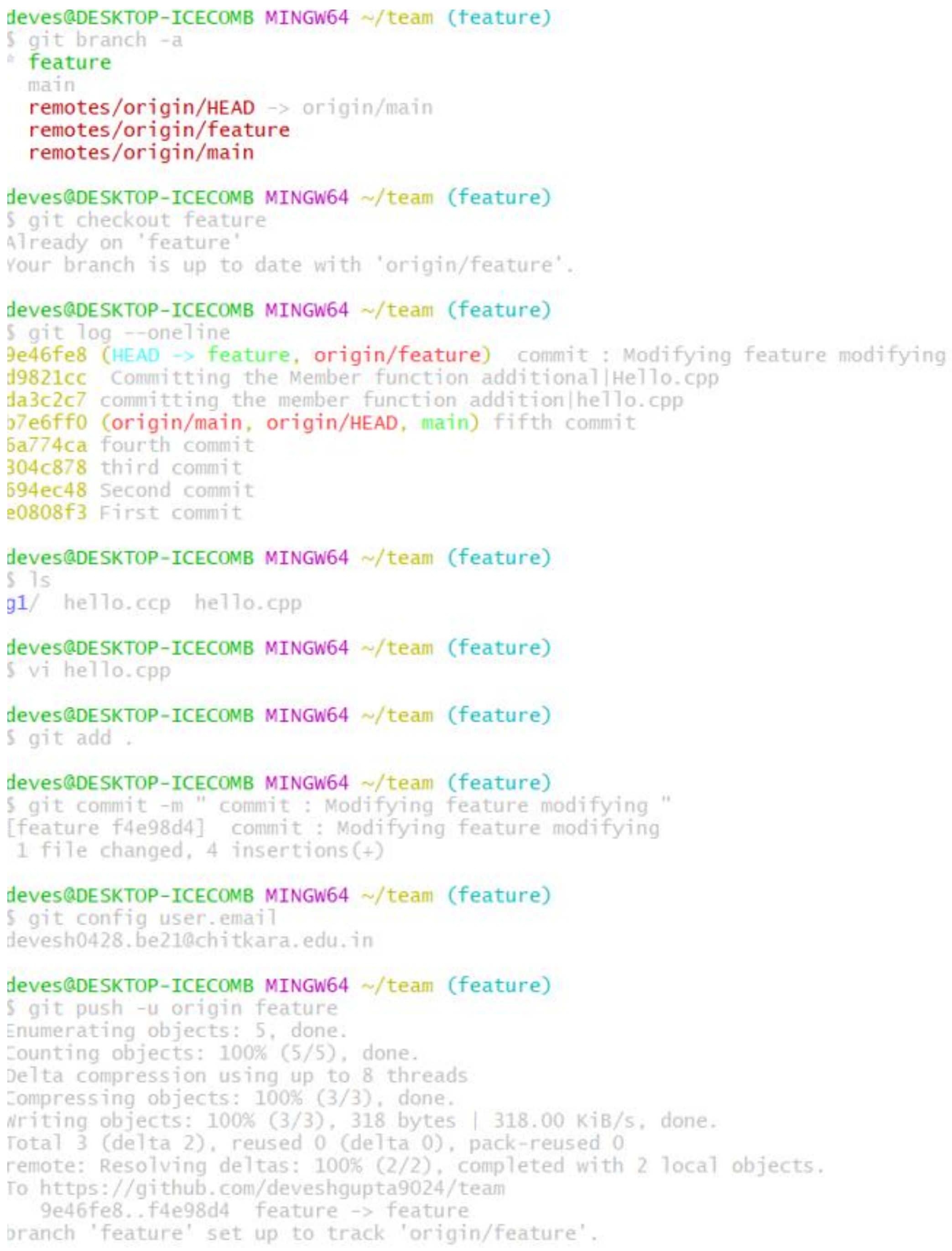


Your GITHUB Repository has been created.

<u>Step2) GIT ADDING REMOTE BRANCH</u>

Git stores a branch as a reference to a commit, and a branch represents the tip of a series of commits.

You Can do it by typing: -
1.git branch -M main



```
deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git branch -a
* feature
  main
  remotes/origin/HEAD -> origin/main
  remotes/origin/feature
  remotes/origin/main

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git checkout feature
Already on 'feature'
Your branch is up to date with 'origin/feature'.

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git log --oneline
9e46fe8 (HEAD -> feature, origin/feature)  commit : Modifying feature modifying
d9821cc  Committing the Member function additional|Hello.cpp
da3c2c7 committing the member function addition|hello.cpp
b7e6ff0 (origin/main, origin/HEAD, main) fifth commit
6a774ca fourth commit
304c878 third commit
694ec48 Second commit
e0808f3 First commit

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ ls
g1/  hello.ccp  hello.cpp

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ vi hello.cpp

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git add .

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git commit -m " commit : Modifying feature modifying "
[feature f4e98d4]  commit : Modifying feature modifying
 1 file changed, 4 insertions(+)

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git config user.email
devesh0428.be21@chitkara.edu.in

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git push -u origin feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 318 bytes | 318.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/deveshgupta9024/team
   9e46fe8..f4e98d4  feature -> feature
branch 'feature' set up to track 'origin/feature'.
```

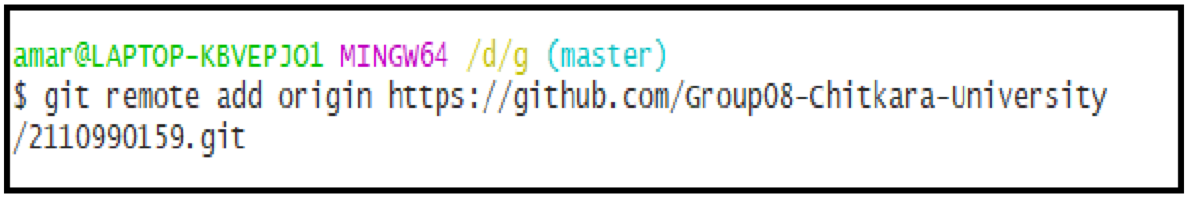
<u>Step3) GIT ADDING REMOTE ORIGIN</u>

Is a Git repository that's hosted on the Internet

You Can do it by typing: -

1.git remote add origin

The page looks like as: -



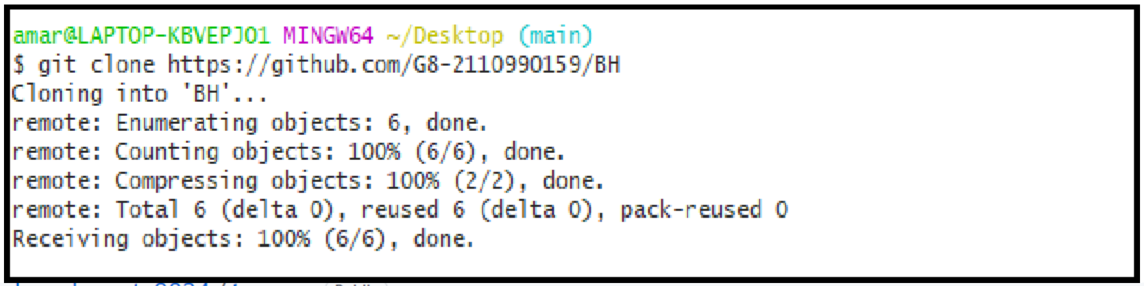
```
amar@LAPTOP-KBVEPJO1 MINGW64 /d/g (master)
$ git remote add origin https://github.com/Group08-Chitkara-University
/2110990159.git
```

Step4) GIT CLONE

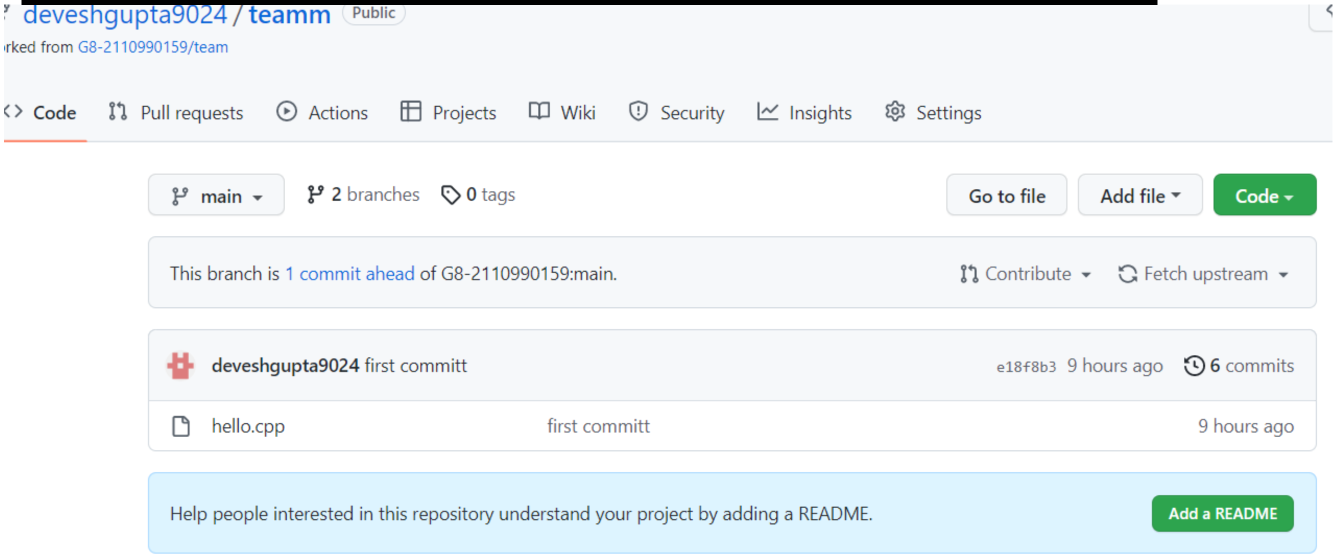
The git clone command is used to upload local repository content to a remote repository.

You Can do it by typing: -
1.git clone url

The page looks like as: -



```
amar@LAPTOP-KBVEPJO1 MINGW64 ~/Desktop (main)
$ git clone https://github.com/G8-2110990159/BH
Cloning into 'BH'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```



deveshgupta9024 / teamm Public

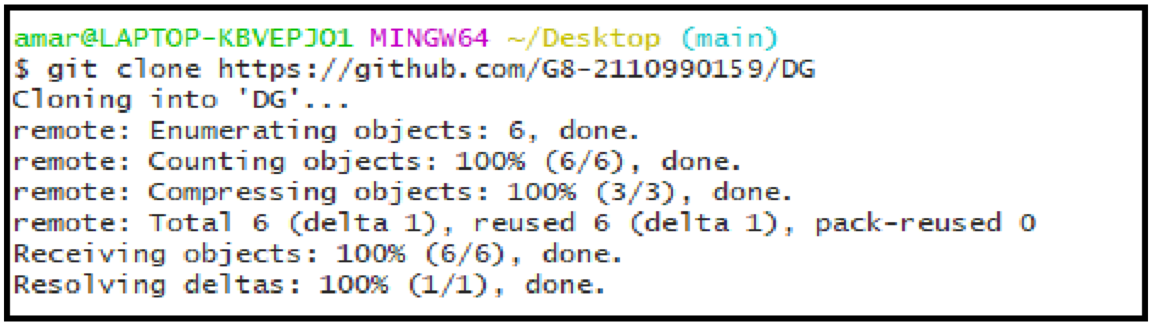
rked from G8-2110990159/team

This branch is 1 commit ahead of G8-2110990159:main.

deveshgupta9024 first committ e18f8b3 9 hours ago 6 commits

hello.cpp first committ 9 hours ago

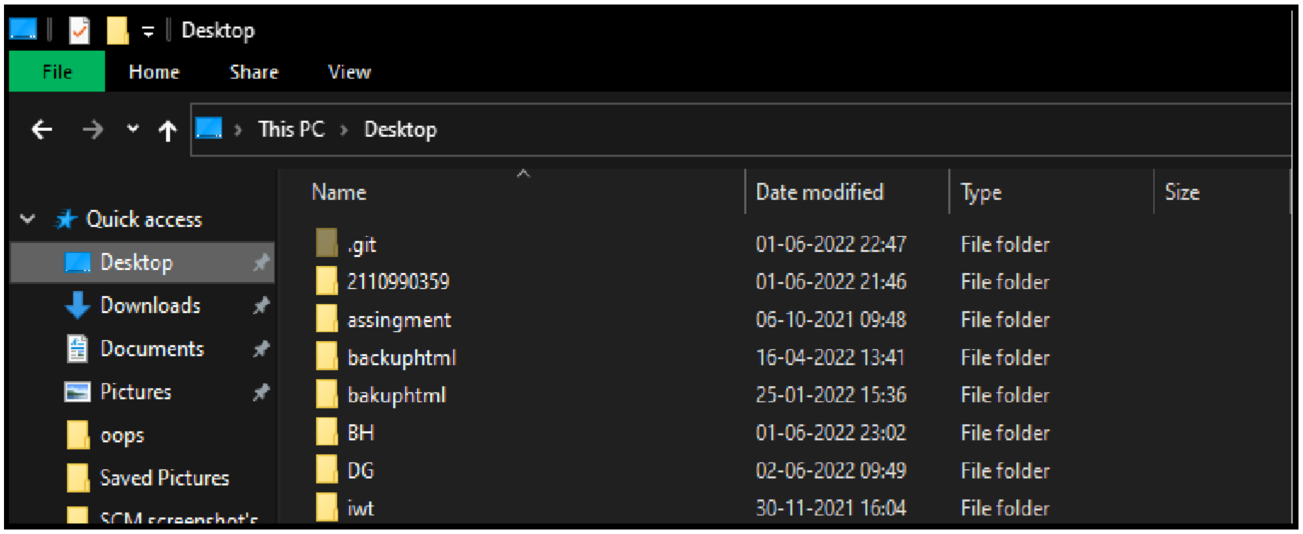
Help people interested in this repository understand your project by adding a README.



```
amar@LAPTOP-KBVEPJO1 MINGW64 ~/Desktop (main)
$ git clone https://github.com/G8-2110990159/DG
Cloning into 'DG'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 1), reused 6 (delta 1), pack-reused 0
Receiving objects: 100% (6/6), done.
Resolving deltas: 100% (1/1), done.
```

1.Document in your system

The page looks like as: -



Desktop

File Home Share View

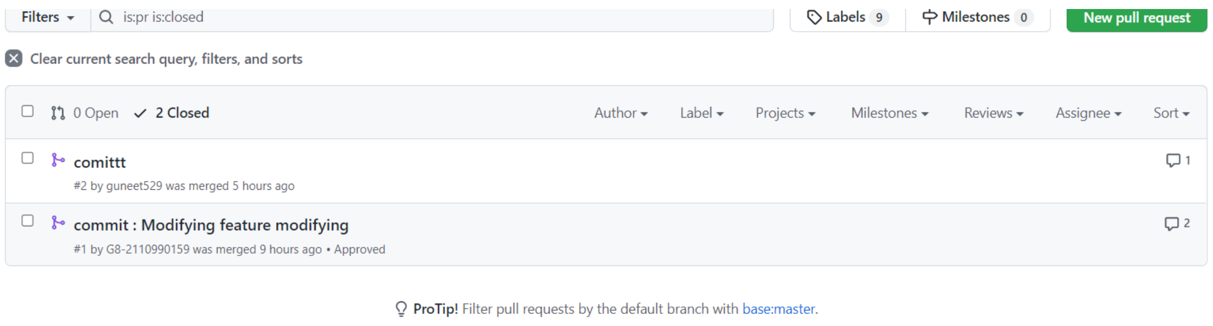
This PC › Desktop

Name	Date modified	Type	Size
.git	01-06-2022 22:47	File folder	
2110990359	01-06-2022 21:46	File folder	
assingment	06-10-2021 09:48	File folder	
backuphtml	16-04-2022 13:41	File folder	
bakuphtml	25-01-2022 15:36	File folder	
BH	01-06-2022 23:02	File folder	
DG	02-06-2022 09:49	File folder	
iwt	30-11-2021 16:04	File folder	

Quick access: Desktop, Downloads, Documents, Pictures, oops, Saved Pictures

2. Document in your GITHUB Repository

The page looks like as: -



Filters is:pr is:closed Labels 9 Milestones 0 New pull request

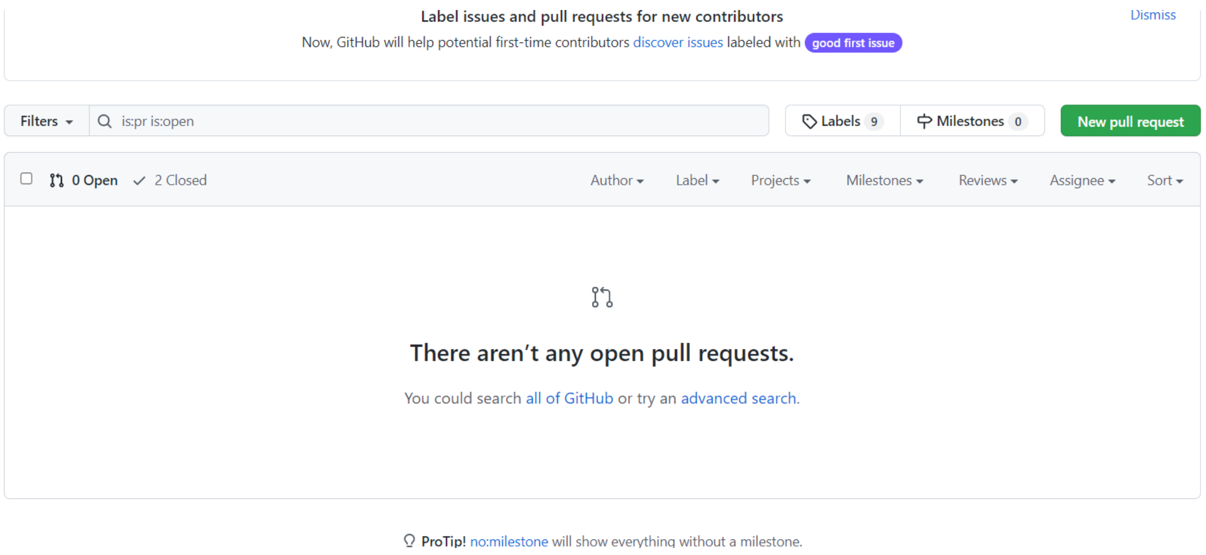
Clear current search query, filters, and sorts

0 Open ✓ 2 Closed Author Label Projects Milestones Reviews Assignee Sort

comittt
#2 by guneet529 was merged 5 hours ago 1

commit : Modifying feature modifying
#1 by G8-2110990159 was merged 9 hours ago • Approved 2

ProTip! Filter pull requests by the default branch with base:master.



Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

Dismiss

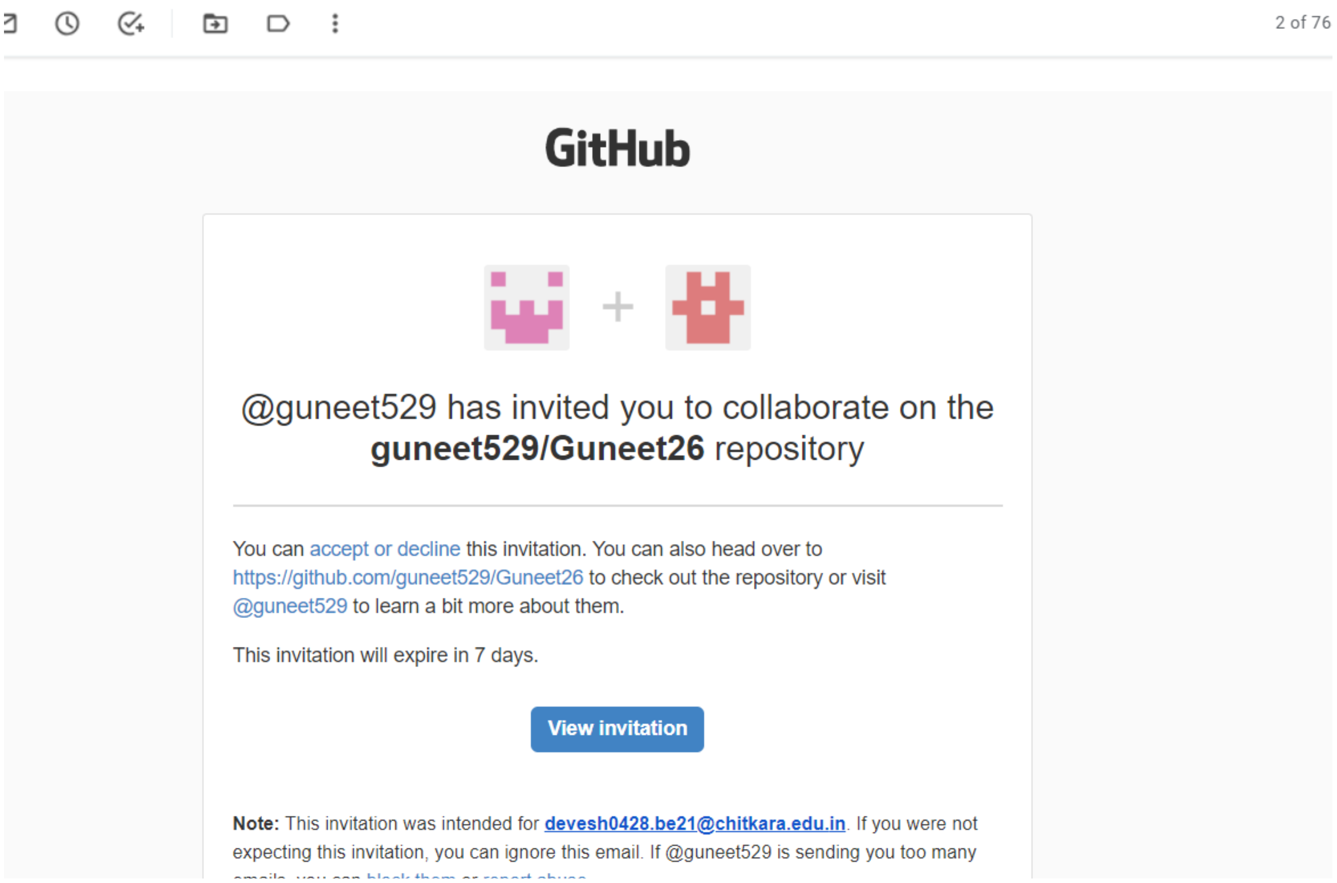
Filters | is:pr is:open | Labels 9 | Milestones 0 | New pull request

0 Open ✓ 2 Closed | Author | Label | Projects | Milestones | Reviews | Assignee | Sort

There aren't any open pull requests.

You could search all of GitHub or try an advanced search.

ProTip! no:milestone will show everything without a milestone.



2 of 76

GitHub

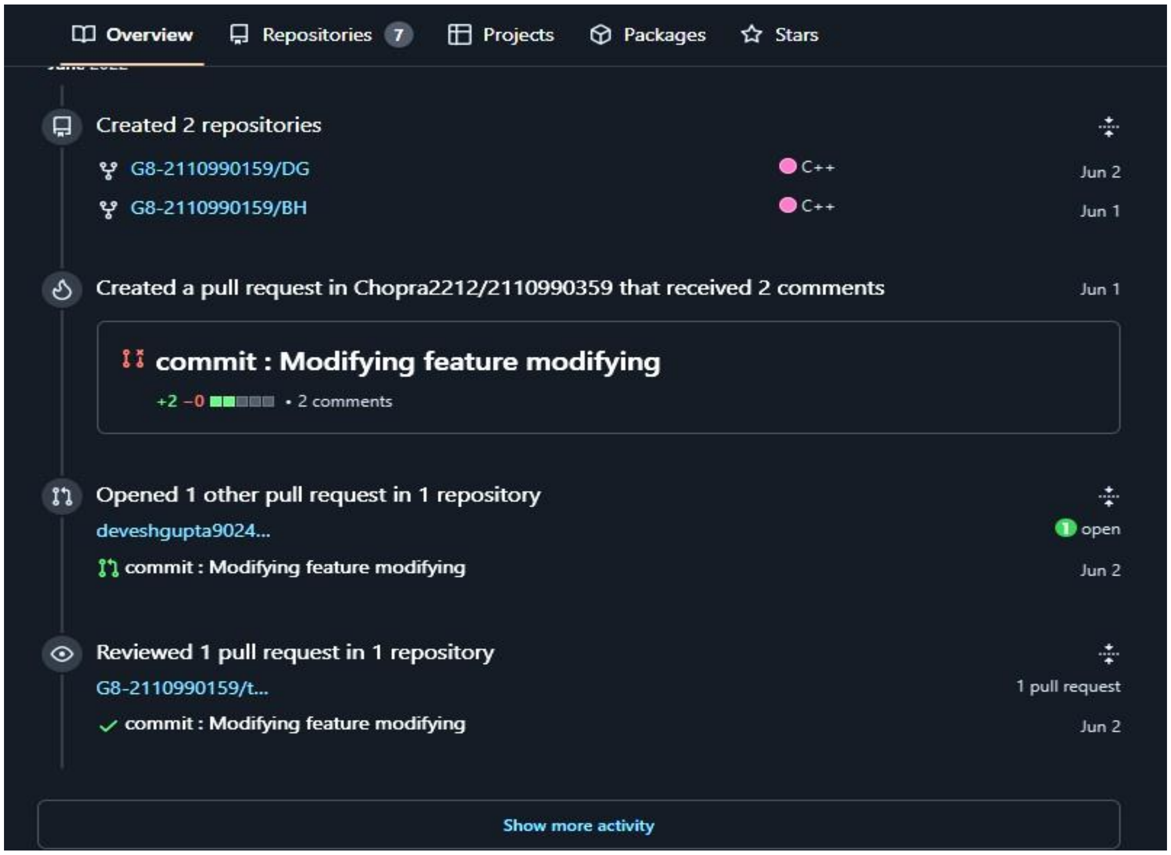
@guneet529 has invited you to collaborate on the **guneet529/Guneet26** repository

You can accept or decline this invitation. You can also head over to https://github.com/guneet529/Guneet26 to check out the repository or visit @guneet529 to learn a bit more about them.

This invitation will expire in 7 days.

View invitation

Note: This invitation was intended for devesh0428.be21@chitkara.edu.in. If you were not expecting this invitation, you can ignore this email. If @guneet529 is sending you too many



Created 2 repositories
G8-2110990159/DG C++ Jun 2
G8-2110990159/BH C++ Jun 1

Created a pull request in Chopra2212/2110990359 that received 2 comments Jun 1

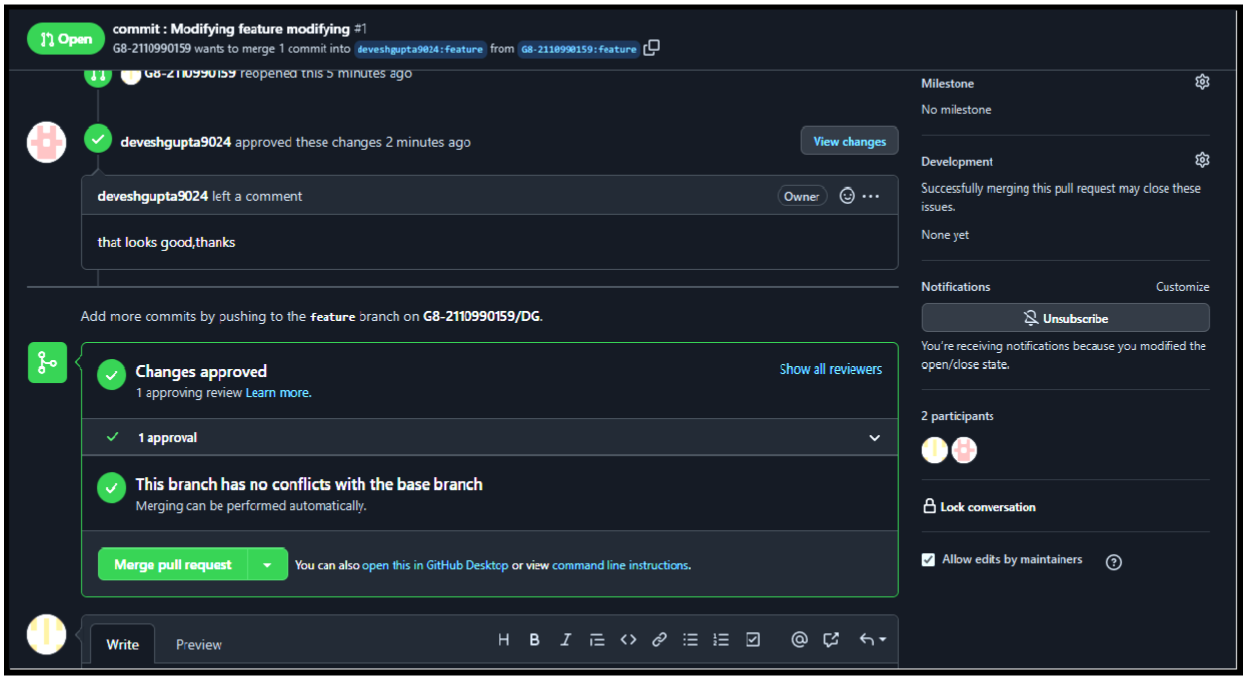
commit : Modifying feature modifying
+2 −0 • 2 comments

Opened 1 other pull request in 1 repository
deveshgupta9024... 1 open
commit : Modifying feature modifying Jun 2

Reviewed 1 pull request in 1 repository
G8-2110990159/t... 1 pull request
✓ commit : Modifying feature modifying Jun 2

Step of working with members :

1.


Open commit : Modifying feature modifying #1
G8-2110990159 wants to merge 1 commit into deveshgupta9024:feature from G8-2110990159:feature

G8-2110990159 reopened this 5 minutes ago

deveshgupta9024 approved these changes 2 minutes ago

deveshgupta9024 left a comment Owner

that looks good,thanks

Add more commits by pushing to the feature branch on G8-2110990159/DG.

Changes approved
1 approving review Learn more.

✓ 1 approval

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

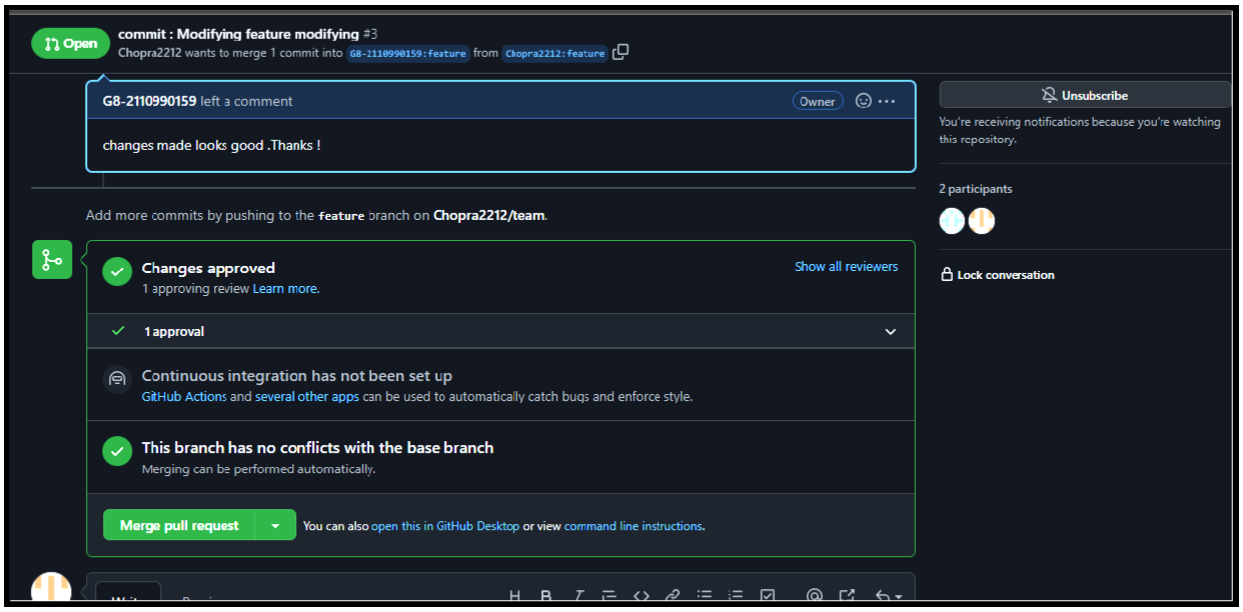
Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Notifications Customize
You're receiving notifications because you modified the open/close state.

2 participants

2.



commit : Modifying feature modifying #3

Chopra2212 wants to merge 1 commit into G8-2110990159:feature from Chopra2212:feature

G8-2110990159 left a comment

changes made looks good .Thanks !

Add more commits by pushing to the feature branch on Chopra2212/team.

Changes approved
1 approving review Learn more.

1 approval

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

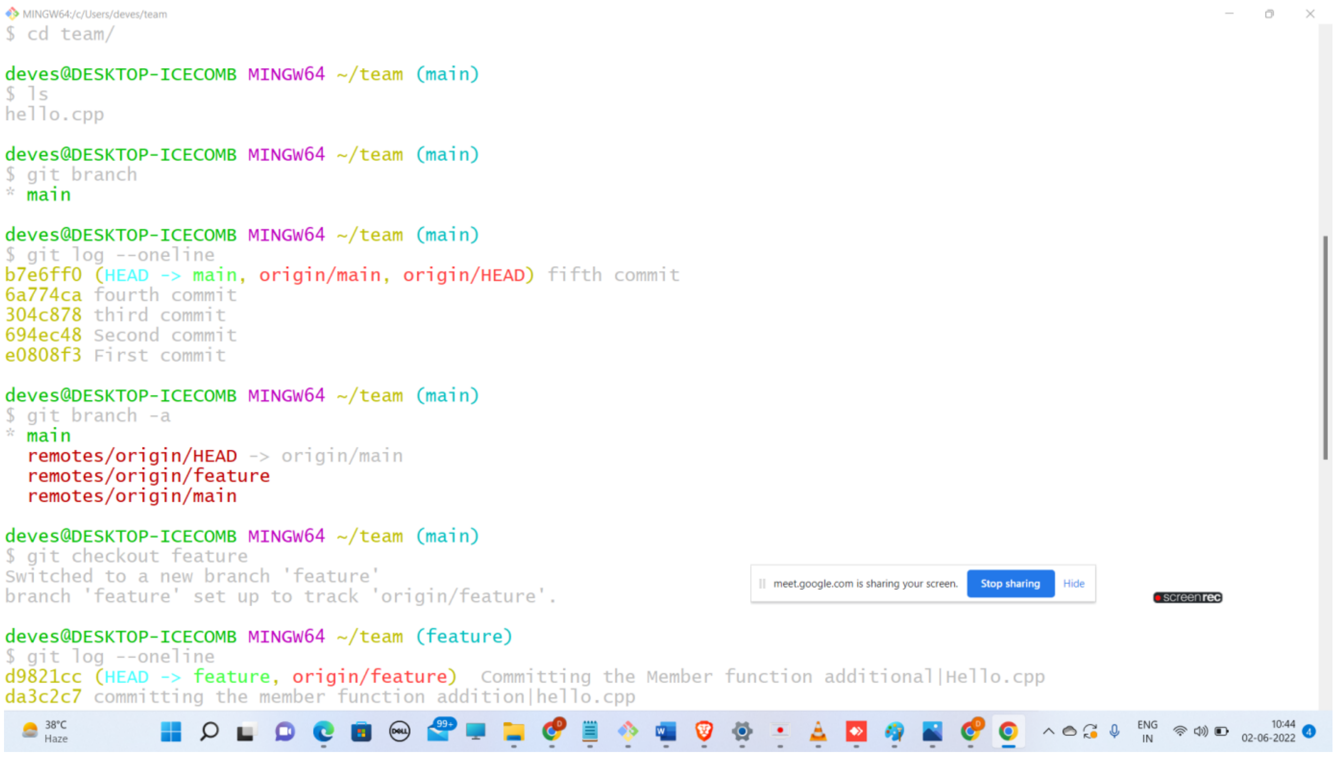
This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request — You can also open this in GitHub Desktop or view command line instructions.

You're receiving notifications because you're watching this repository.

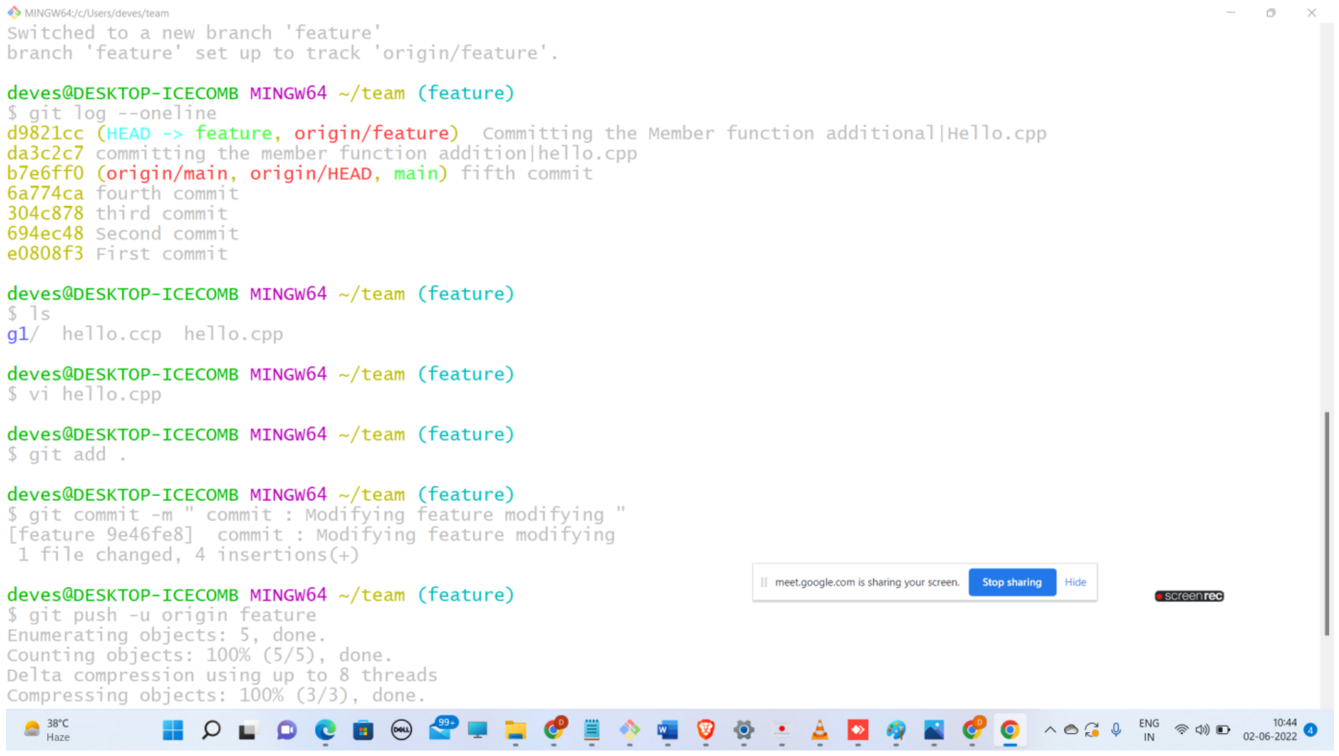
2 participants

<u>Showing steps on git Bash :</u>

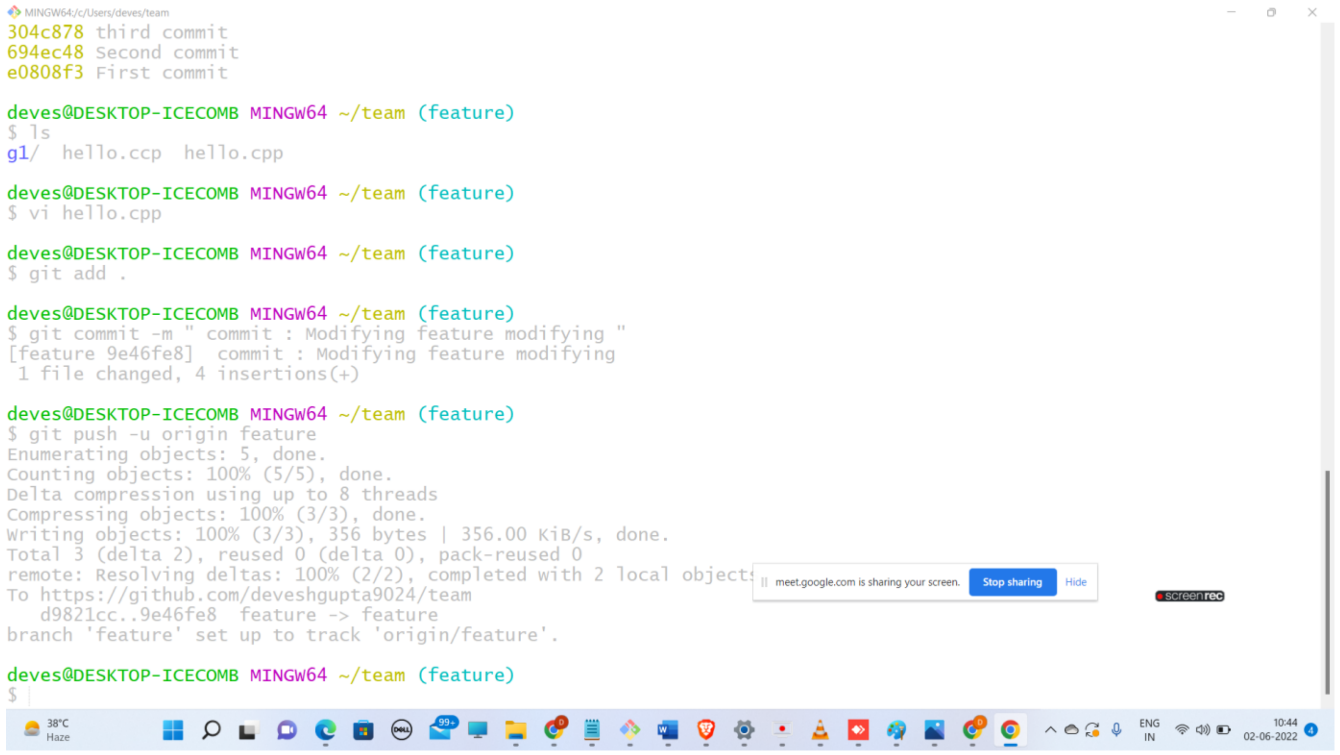


```
MINGW64:/c/Users/deves/team
$ cd team/

deves@DESKTOP-ICECOMB MINGW64 ~/team (main)
$ ls
hello.cpp

deves@DESKTOP-ICECOMB MINGW64 ~/team (main)
$ git branch
* main

deves@DESKTOP-ICECOMB MINGW64 ~/team (main)
$ git log --oneline
b7e6ff0 (HEAD -> main, origin/main, origin/HEAD) fifth commit
6a774ca fourth commit
304c878 third commit
694ec48 Second commit
e0808f3 First commit

deves@DESKTOP-ICECOMB MINGW64 ~/team (main)
$ git branch -a
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/feature
  remotes/origin/main

deves@DESKTOP-ICECOMB MINGW64 ~/team (main)
$ git checkout feature
Switched to a new branch 'feature'
branch 'feature' set up to track 'origin/feature'.

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git log --oneline
d9821cc (HEAD -> feature, origin/feature)  Committing the Member function additional|Hello.cpp
da3c2c7 committing the member function addition|hello.cpp
```

1.


MINGW64:/c/Users/deves/team

```
Switched to a new branch 'feature'
branch 'feature' set up to track 'origin/feature'.

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git log --oneline
d9821cc (HEAD -> feature, origin/feature)  Committing the Member function additional|Hello.cpp
da3c2c7 committing the member function addition|hello.cpp
b7e6ff0 (origin/main, origin/HEAD, main) fifth commit
6a774ca fourth commit
304c878 third commit
694ec48 Second commit
e0808f3 First commit

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ ls
g1/  hello.ccp  hello.cpp

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ vi hello.cpp

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git add .

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git commit -m " commit : Modifying feature modifying "
[feature 9e46fe8]  commit : Modifying feature modifying
 1 file changed, 4 insertions(+)

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git push -u origin feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
```


MINGW64:/c/Users/deves/team

```
304c878 third commit
694ec48 Second commit
e0808f3 First commit

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ ls
g1/  hello.ccp  hello.cpp

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ vi hello.cpp

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git add .

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git commit -m " commit : Modifying feature modifying "
[feature 9e46fe8]  commit : Modifying feature modifying
 1 file changed, 4 insertions(+)

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$ git push -u origin feature
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 356 bytes | 356.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local object
To https://github.com/deveshgupta9024/team
   d9821cc..9e46fe8  feature -> feature
branch 'feature' set up to track 'origin/feature'.

deves@DESKTOP-ICECOMB MINGW64 ~/team (feature)
$
```

You can view, comment on, or merge this pull request online at:

https://github.com/deveshgupta9024/devesh0428/pull/2

Commit Summary

- 9e096bd comittt

File Changes (1 file)

- **M** primefactor.cpp (2)

Patch Links:

- https://github.com/deveshgupta9024/devesh0428/pull/2.patch
- https://github.com/deveshgupta9024/devesh0428/pull/2.diff

—
Reply to this email directly, view it on GitHub, or unsubscribe.
You are receiving this because you are subscribed to this thread.

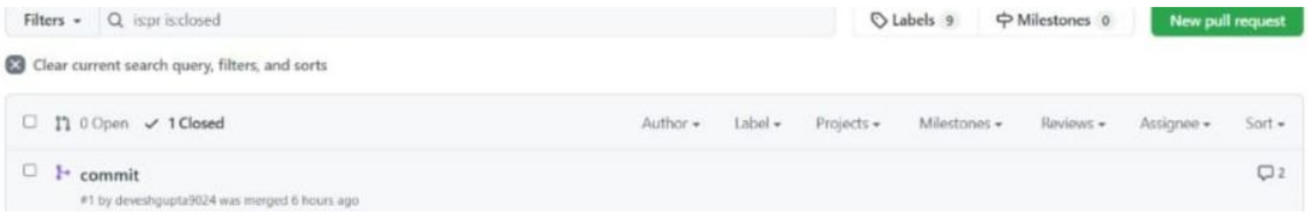
Re: [G8-2110990159/team] first committ (PR #4) External Inbox ×

G8-2110990159 <notifications@github.com> Unsubscribe
to G8-2110990159/team, me, Author

@G8-2110990159 approved this pull request.

that look good thanks

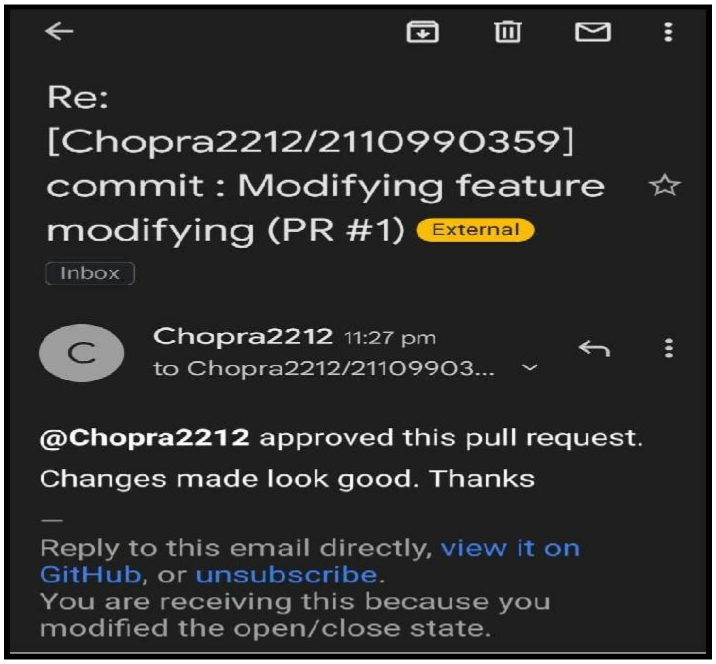
—
Reply to this email directly, view it on GitHub, or unsubscribe.
You are receiving this because you authored the thread.



commit
#1 by deveshgupta9024 was merged 6 hours ago
2

Approve mail recived :

1.



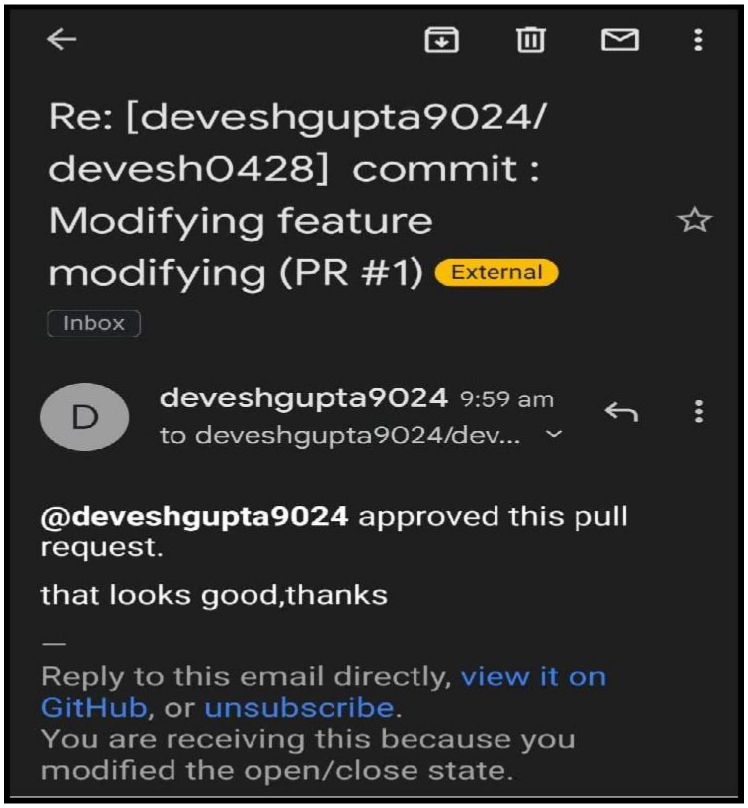
Re: [Chopra2212/2110990359] commit : Modifying feature modifying (PR #1) External

Chopra2212 11:27 pm
to Chopra2212/21109903...

@Chopra2212 approved this pull request.

Changes made look good. Thanks

—
Reply to this email directly, view it on GitHub, or unsubscribe.
You are receiving this because you modified the open/close state.



Re: [deveshgupta9024/ devesh0428] commit : Modifying feature modifying (PR #1) External

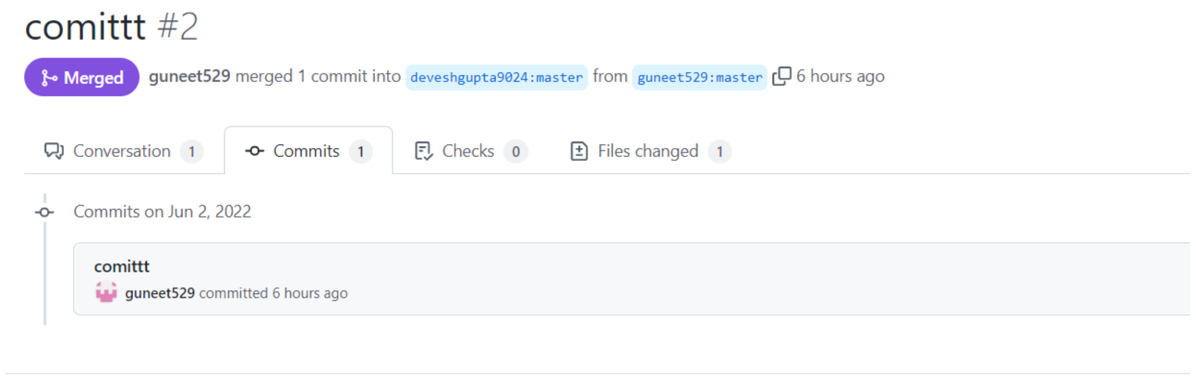
Inbox

deveshgupta9024 9:59 am
to deveshgupta9024/dev...

@deveshgupta9024 approved this pull request.

that looks good,thanks

—
Reply to this email directly, view it on GitHub, or unsubscribe.
You are receiving this because you modified the open/close state.

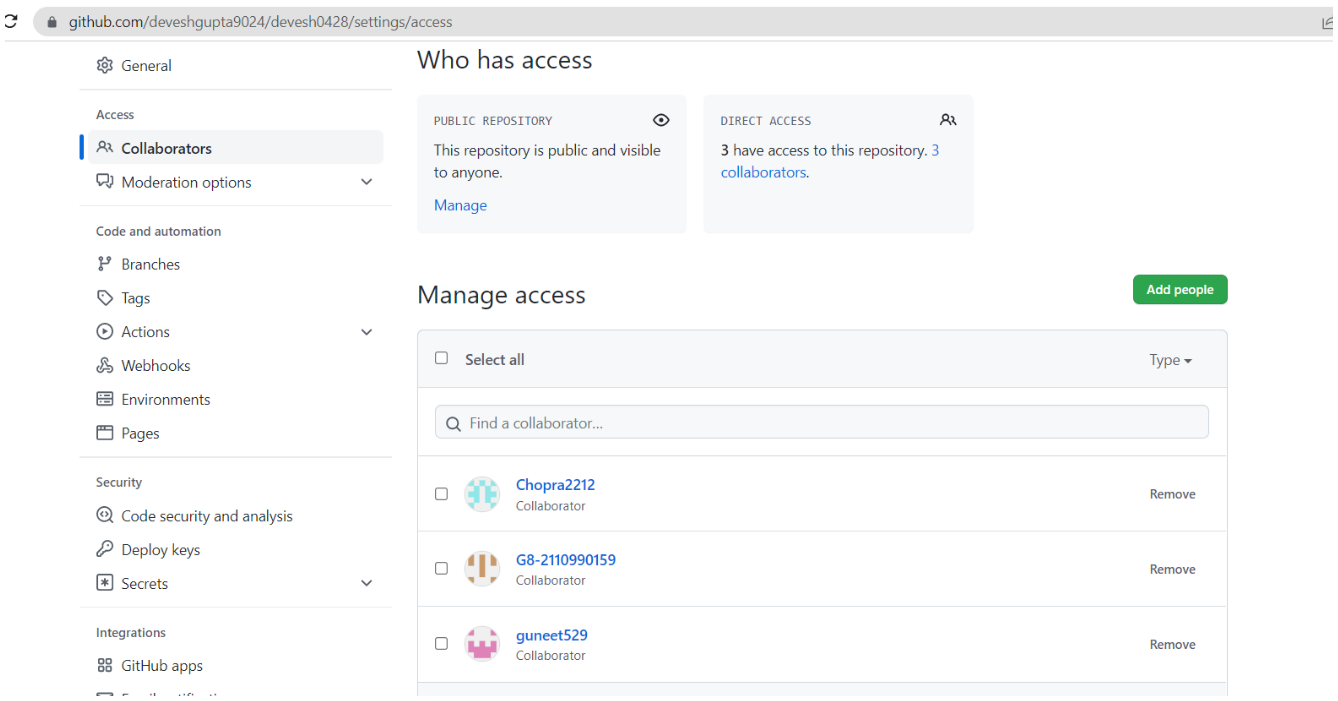


comittt #2

Merged guneet529 merged 1 commit into `deveshgupta9024:master` from `guneet529:master` 6 hours ago

Commits on Jun 2, 2022

comittt
guneet529 committed 6 hours ago


github.com/deveshgupta9024/devesh0428/settings/access

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Pages

Security

Code security and analysis

Deploy keys

Secrets

Integrations

GitHub apps

Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

Manage

DIRECT ACCESS

3 have access to this repository. 3 collaborators.

Manage access

Add people

Select all | Type

Find a collaborator…

Chopra2212	Collaborator	Remove
G8-2110990159	Collaborator	Remove
guneet529	Collaborator	Remove