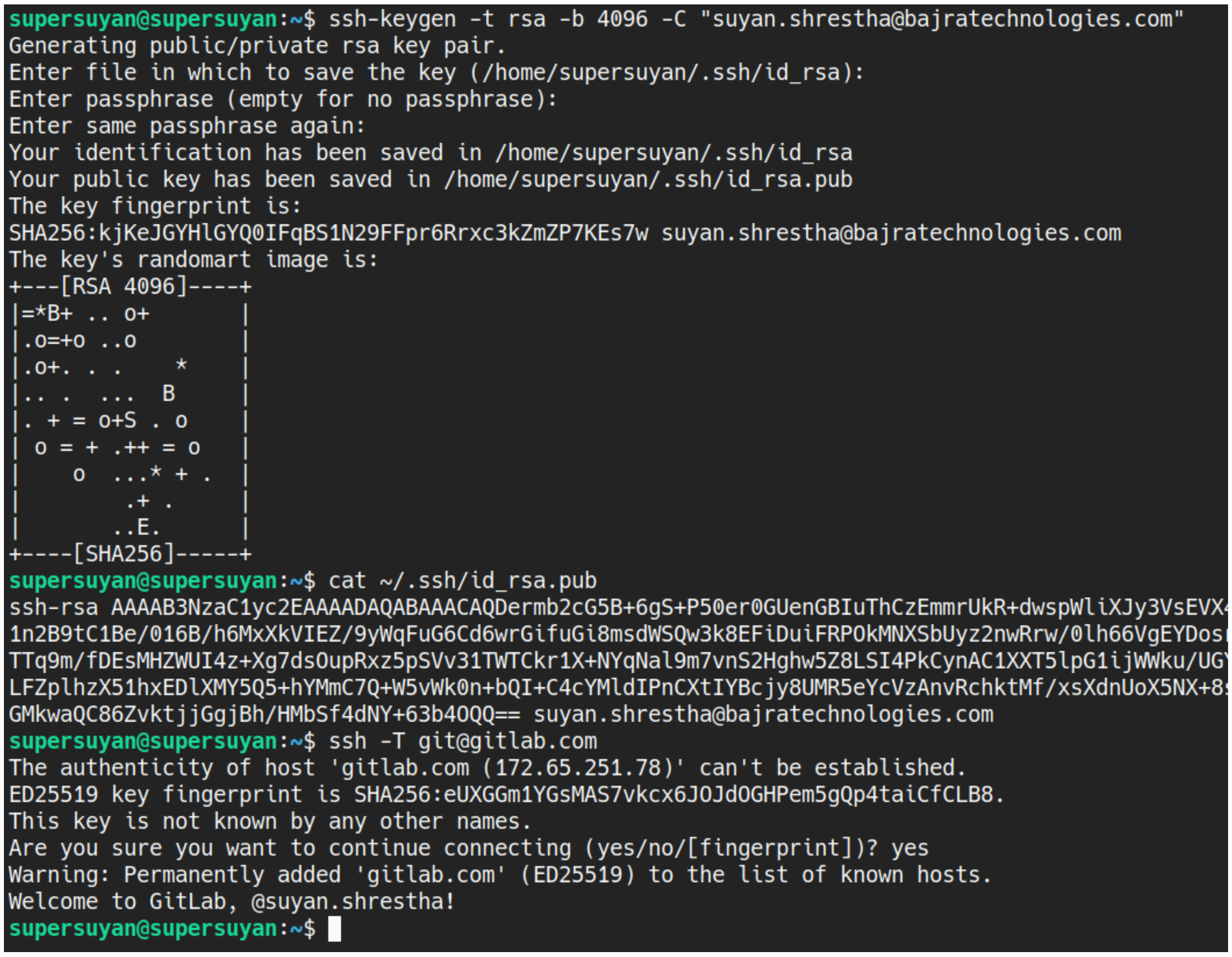
ii. “Welcome to GitLab, @suyan.shrestha!” will be the result if its correctly configured.

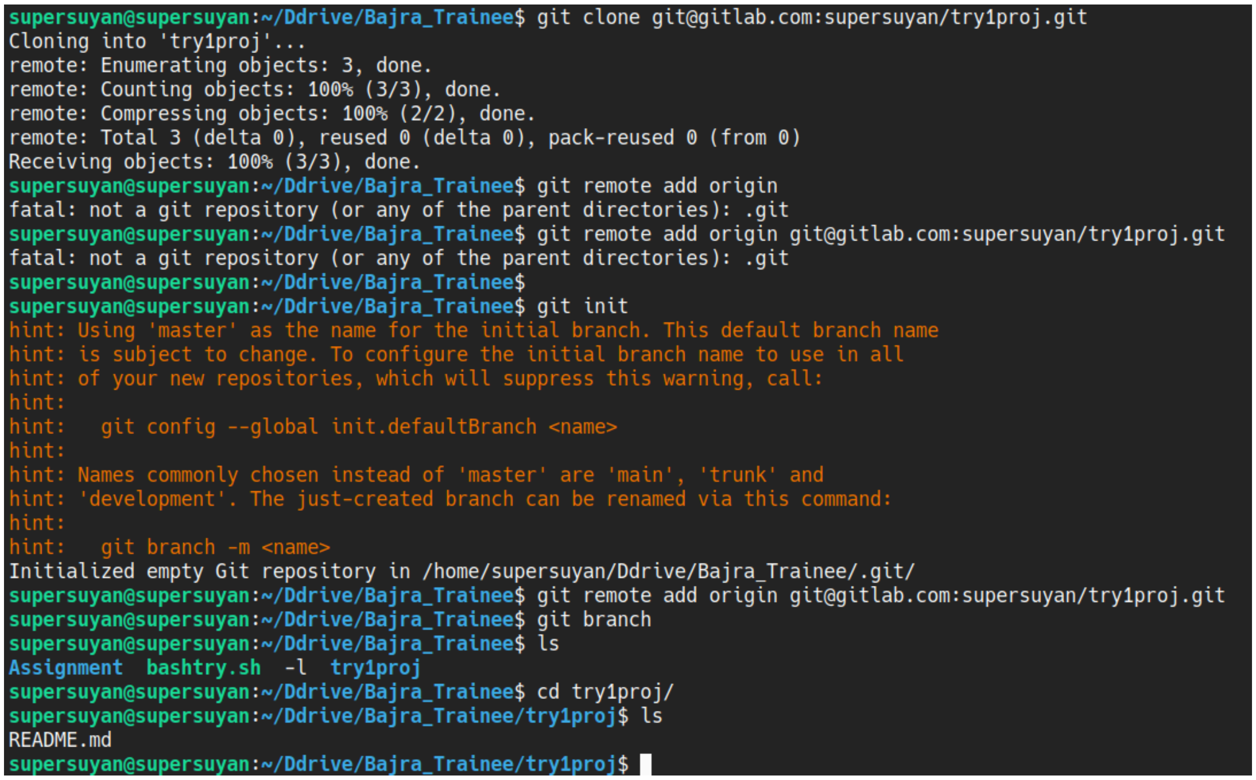


```
supersuyan@supersuyan:~$ ssh-keygen -t rsa -b 4096 -C "suyan.shrestha@bajratechnologies.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/supersuyan/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/supersuyan/.ssh/id_rsa
Your public key has been saved in /home/supersuyan/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:kjKeJGYHlGYQ0IFqBS1N29FFpr6Rrxc3kZmZP7KEs7w suyan.shrestha@bajratechnologies.com
The key's randomart image is:
+---[RSA 4096]----+
|=*B+ .. o+       |
|.o=+o ..o        |
|.o+. . .    *    |
|.. .   ...  B    |
|. + = o+S . o    |
| o = + .++ = o   |
|    o  ...* + .  |
|        .+ .     |
|       ..E.      |
+----[SHA256]-----+
supersuyan@supersuyan:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDermb2cG5B+6gS+P50er0GUenGBIuThCzEmmrUkR+dwspWliXJy3VsEVX
1n2B9tC1Be/016B/h6MxXkVIEZ/9yWqFuG6Cd6wrGifuGi8msdWSQw3k8EFiDuiFRPOkMNXSbUyz2nwRrw/0lh66VgEYDos
TTq9m/fDEsMHZWUI4z+Xg7dsOupRxz5pSVv31TWTCkr1X+NYqNal9m7vnS2Hghw5Z8LSI4PkCynAC1XXT5lpG1ijWWku/UG
LFZplhzX51hxEDlXMY5Q5+hYMmC7Q+W5vWk0n+bQI+C4cYMldIPnCXtIYBcjy8UMR5eYcVzAnvRchktMf/xsXdnUoX5NX+8
GMkwaQC86ZvktjjGgjBh/HMbSf4dNY+63b40QQ== suyan.shrestha@bajratechnologies.com
supersuyan@supersuyan:~$ ssh -T git@gitlab.com
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ED25519 key fingerprint is SHA256:eUXGGm1YGsMAS7vkcx6JOJdOGHPem5gQp4taiCfCLB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.com' (ED25519) to the list of known hosts.
Welcome to GitLab, @suyan.shrestha!
supersuyan@supersuyan:~$
```

Git basic commands

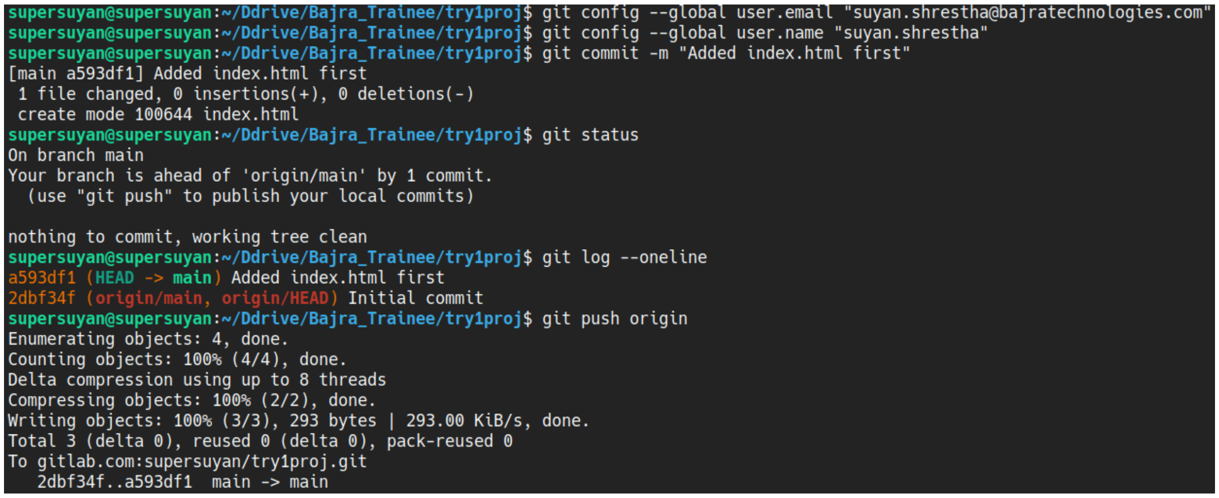
a. Git init : start a new repo
b. Git clone <repo> : obtain repo from existing url
c. git add <file> : adds file to staging area
d. git commit -m "<Message>" : records or snapshots the file permanently in the version history with a message <Message>.
e. Git status : shows the list of files that have been changed along with the files that are yet to be added or committed.
f. Git checkout branchname : to just switch to a branch
g. Git checkout -b branchname : to actually create a branch and then switch to that branch
h. git branch -d branchname : to delete the branch
i. Git reset filename : undo a git add filename

j. Git commit –amend : now, suppose most recent commit had some stuff left to be added. Now instead of creating a completely new commit for that addition, we cant just use –amend to edit the most recent commit.



```
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ git clone git@gitlab.com:supersuyan/try1proj.git
Cloning into 'try1proj'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ git remote add origin
fatal: not a git repository (or any of the parent directories): .git
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ git remote add origin git@gitlab.com:supersuyan/try1proj.git
fatal: not a git repository (or any of the parent directories): .git
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/supersuyan/Ddrive/Bajra_Trainee/.git/
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ git remote add origin git@gitlab.com:supersuyan/try1proj.git
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ git branch
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ ls
Assignment  bashtry.sh  -l  try1proj
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee$ cd try1proj/
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ ls
README.md
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ 
```

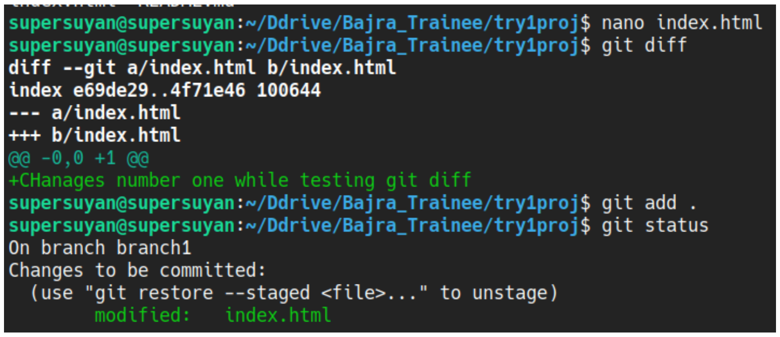
Doing git config for initial setup



```
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git config --global user.email "suyan.shrestha@bajratechnologies.com"
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git config --global user.name "suyan.shrestha"
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git commit -m "Added index.html first"
[main a593df1] Added index.html first
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git log --oneline
a593df1 (HEAD -> main) Added index.html first
2dbf34f (origin/main, origin/HEAD) Initial commit
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git push origin
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To gitlab.com:supersuyan/try1proj.git
   2dbf34f..a593df1  main -> main
```

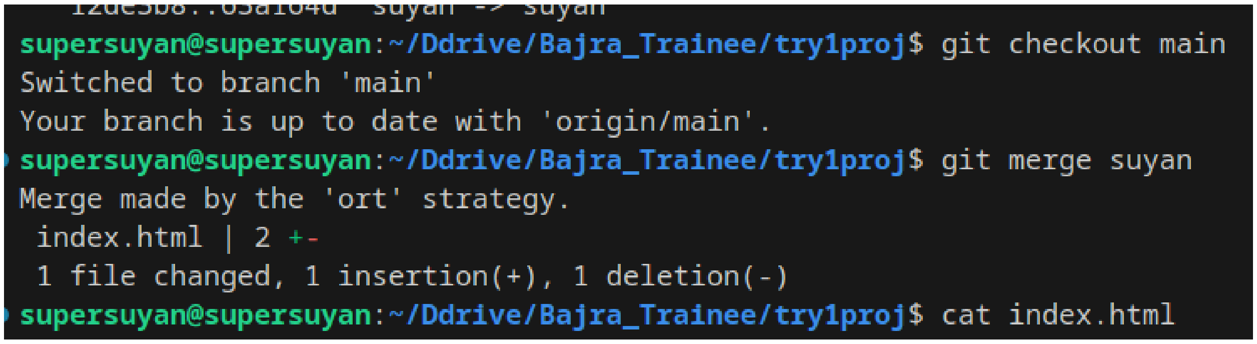
Testing git diff



```
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ nano index.html
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git diff
diff --git a/index.html b/index.html
index e69de29..4f71e46 100644
--- a/index.html
+++ b/index.html
@@ -0,0 +1 @@
+CHanages number one while testing git diff
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git add .
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git status
On branch branch1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html
```

Now, I wanted to make changes in suyan branch, then merge that suyan branch to main branch

a. Simply go to main branch, then use git merge suyan

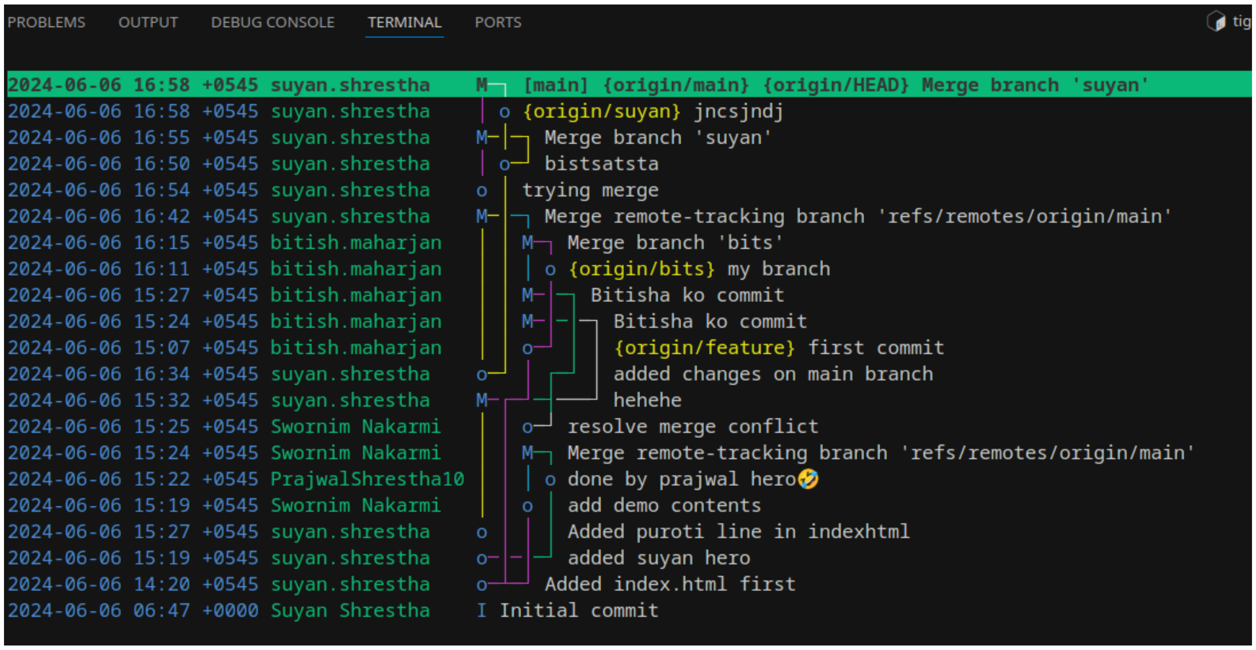


```
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git merge suyan
Merge made by the 'ort' strategy.
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ cat index.html
```

a. Git stash : dont commit changes right now, hold them temporarily. Just commit them after a while.
    i. Stash is better than commit

Tig

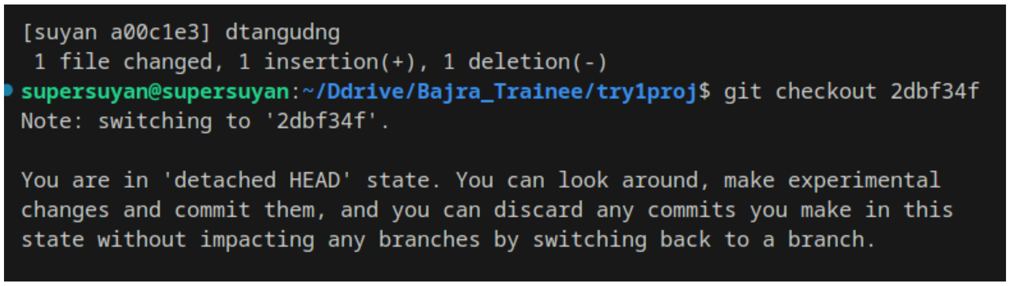
a. Shows tree structure of who committed what



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS tig

```
2024-06-06 16:58 +0545 suyan.shrestha      M─┐  [main] {origin/main} {origin/HEAD} Merge branch 'suyan'
2024-06-06 16:58 +0545 suyan.shrestha      │ o {origin/suyan} jncsjndj
2024-06-06 16:55 +0545 suyan.shrestha      M─│─┐ Merge branch 'suyan'
2024-06-06 16:50 +0545 suyan.shrestha      │ o─┘ bistsatsta
2024-06-06 16:54 +0545 suyan.shrestha      o │ trying merge
2024-06-06 16:42 +0545 suyan.shrestha      M─│─┐ Merge remote-tracking branch 'refs/remotes/origin/main'
2024-06-06 16:15 +0545 bitish.maharjan     │ │ M─┐ Merge branch 'bits'
2024-06-06 16:11 +0545 bitish.maharjan     │ │ │ o {origin/bits} my branch
2024-06-06 15:27 +0545 bitish.maharjan     │ │ M─│─┐ Bitisha ko commit
2024-06-06 15:24 +0545 bitish.maharjan     │ │ M─│─│─┐ Bitisha ko commit
2024-06-06 15:07 +0545 bitish.maharjan     │ │ o─┘ │ │ {origin/feature} first commit
2024-06-06 16:34 +0545 suyan.shrestha      o─┘ │ ┌─┘ │ added changes on main branch
2024-06-06 15:32 +0545 suyan.shrestha      M─┌─│─│───┘ hehehe
2024-06-06 15:25 +0545 Swornim Nakarmi     │ │ o─┘ resolve merge conflict
2024-06-06 15:24 +0545 Swornim Nakarmi     │ │ M─┐ Merge remote-tracking branch 'refs/remotes/origin/main'
2024-06-06 15:22 +0545 PrajwalShrestha10   │ │ │ o done by prajwal hero🤣
2024-06-06 15:19 +0545 Swornim Nakarmi     │ │ o │ add demo contents
2024-06-06 15:27 +0545 suyan.shrestha      o │ │ │ Added puroti line in indexhtml
2024-06-06 15:19 +0545 suyan.shrestha      o─│─│─┘ added suyan hero
2024-06-06 14:20 +0545 suyan.shrestha      o─┴─┘ Added index.html first
2024-06-06 06:47 +0000 Suyan Shrestha      I Initial commit
```

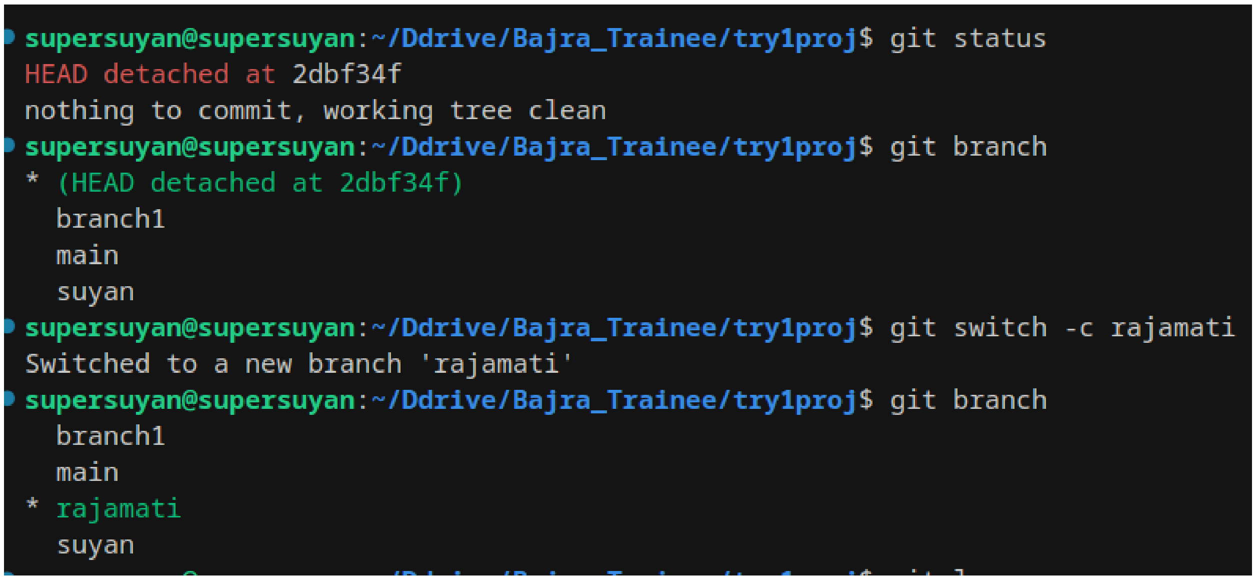
a. Git switch -c branchname : creates branch like checkout -b
b. Git checkout “commit_hash”



```
[suyan a00c1e3] dtangudng
 1 file changed, 1 insertion(+), 1 deletion(-)
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git checkout 2dbf34f
Note: switching to '2dbf34f'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.
```

c. If head is detached due to jumping to some commithash, then head is pointing to anonymous branch. WE will have to create a new branch in order to solve this problem.



```
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git status
HEAD detached at 2dbf34f
nothing to commit, working tree clean
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git branch
* (HEAD detached at 2dbf34f)
  branch1
  main
  suyan
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git switch -c rajamati
Switched to a new branch 'rajamati'
supersuyan@supersuyan:~/Ddrive/Bajra_Trainee/try1proj$ git branch
  branch1
  main
* rajamati
  suyan
```

d. Git tags are references that point to specific commits in the history of a Git repository. They are commonly used to mark specific points in the project’s history, often used to denote versions or releases (e.g., v1.0.0, v2.1.1).
e. git fetch : download objects and refs from another repository

Merge vs Rebase vs Squash

a. Git merge
    i. Pulls in latest changes from main to feature branch

Git best commit practices

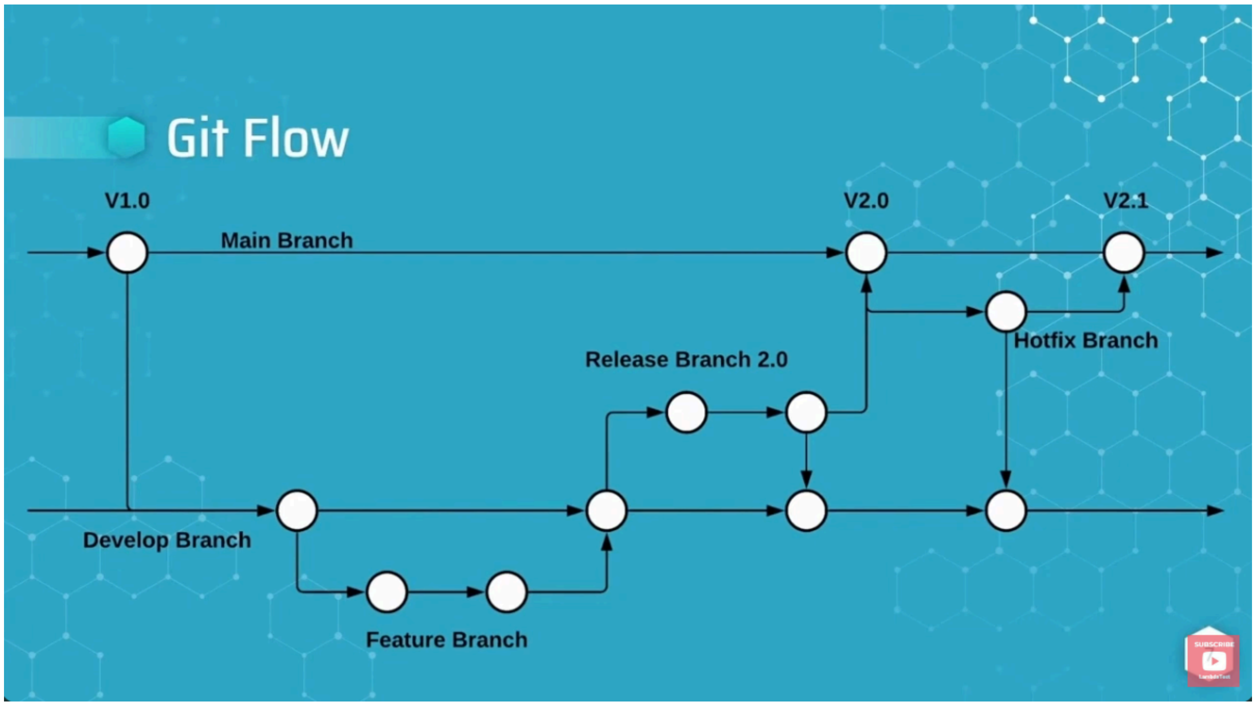
a. git shortlog : which groups commits by user, again showing just the subject line for concision:
b. Begin all subject lines with a capital letter.
    i. Accelerate to 88 miles per hour

        ii. Not “accelerate to 88 miles per hour”

c. Do not end the subject line with a period
    i. Open the pod bay doors
    ii. Not “Open the pod bay doors.”
d. Use the imperative mood in the subject line
    i. Update getting started documentation
    ii. Not “Updated getting started documentation”
e. Wrap the body at 72 characters
    i. Dont write longass characters.
    ii. Keep the commit message generally under 50 words
f. Make small, specific commits
g. Sometimes if we want to write something longer than 50 characters as commit message, then

    Git commit -m “title 50 char long” -m “body of message that is long”

Git flow



Types of branches

a. Main Branch
    i. contain production-ready code that can be released.
    ii. can be tagged at various commits in order to signify different versions or releases of the code, and other branches will be merged into the main branch after they have been sufficiently vetted and tested.
b. Develop branch

    i. maintained throughout the development process, and contains pre-production code with newly developed features that are in the process of being tested.
    ii. Feature branches just are created off the develop branch, and merged back to it after the features are reviews and completed.

c. Feature Branch
    i. When adding new features to your code, just make a new branch.
    ii. Such branches are made off the develop branch and are merged back to develop branch again after completion of feature.

d. Release branch
    i. If we have enough features to be released to production, just use a release branch
    ii. The release is feature complete, so we dont add more features to it.
    iii. THen release branch is merged to main branch, and that merge point is tagged with a version using git tag
    iv. Now main branch is also merged to develop branch, so that devs working on development mode get access to latest features.

e. Hotfix branch
    i. the hotfix branch is used to quickly address necessary changes in your main branch.
    ii. If there are bugs in main branch, then this helps to solve them up
    iii. After fix, hotfix branch must be merged back to main and develop branches.

Day4

Web Basics

HTML

Overview of HTML

a. standard markup language used to create web pages.
b. It describes the structure of a web page semantically and originally included cues for the appearance of the document.

Document Structure

- structured like a tree

a. . <!DOCTYPE html>: Declares the document type and version of HTML.
b. <html>: The root element that contains all other elements.
c. <head>: Contains meta-information about the document (e.g., title, character set).