NAME: T MANOJ

EMPID : LYAKE2KHS

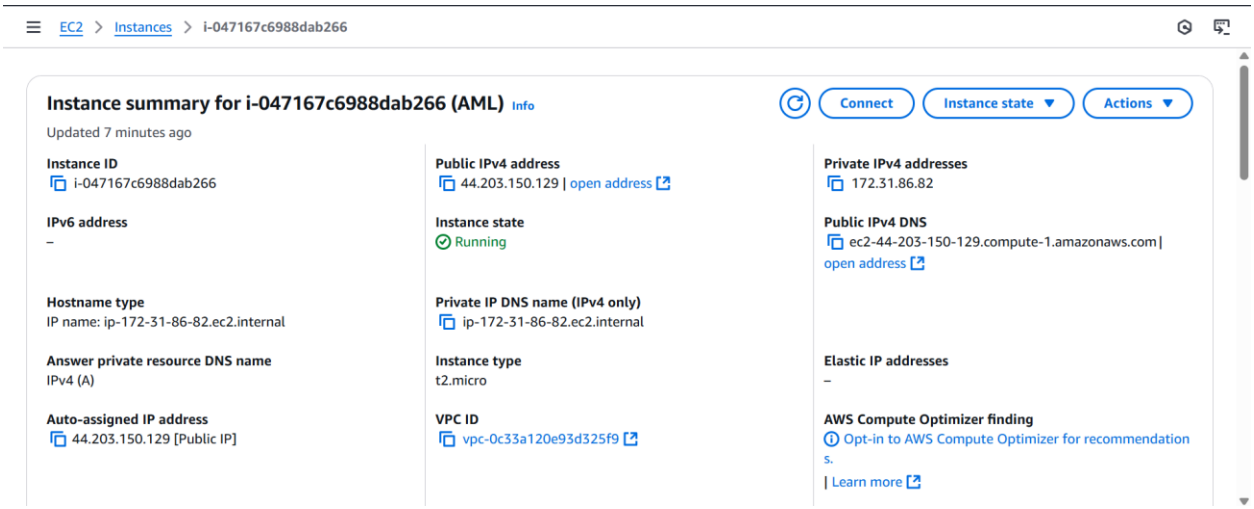
GIT COMMANDS [left-right Architecture]

1. Install Git on an EC2 Server

Step 1: Launch an EC2 Instance

1. Go to the **AWS Management Console** → Navigate to **EC2**
2. Click on **Launch Instance**
3. Select an **Amazon Linux,.**
4. Configure instance settings as needed
5. In the **User Data** section, enter the script to install Git:

```
#!/bin/bash
sudo yum update -y
sudo yum install git -y
```



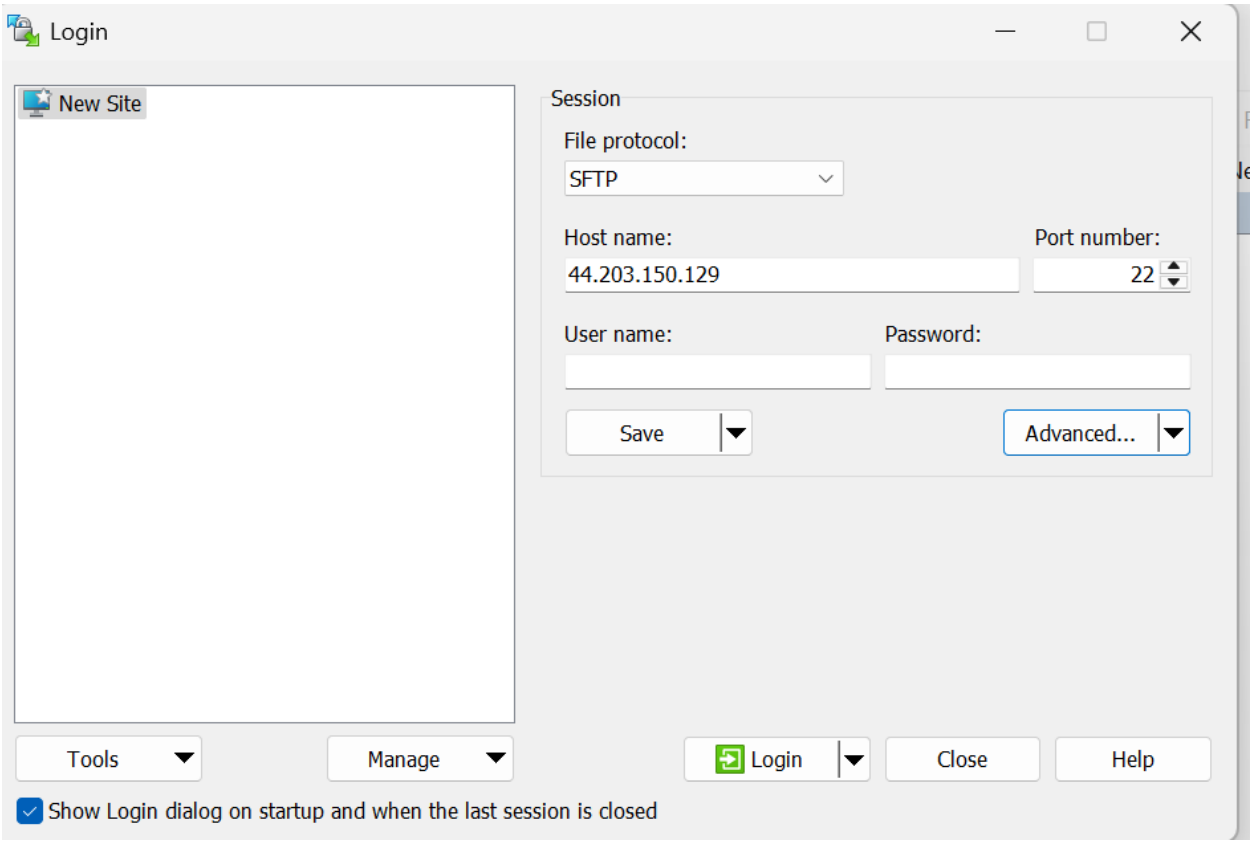
6. Launch the instance and **SSH into it using PuTTY**
7. Verify Git installation:

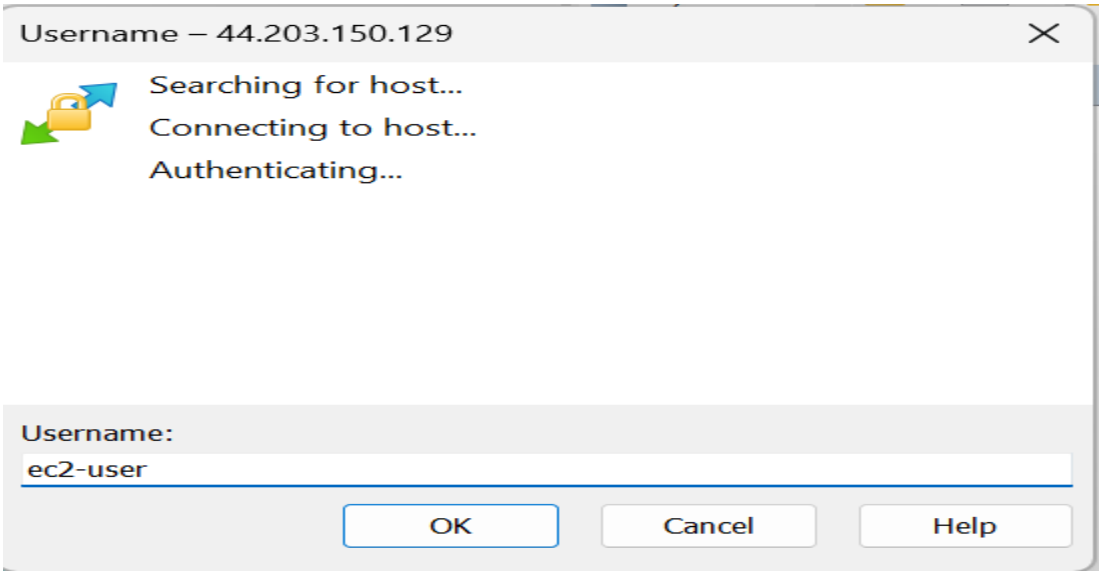

```
git --version
```

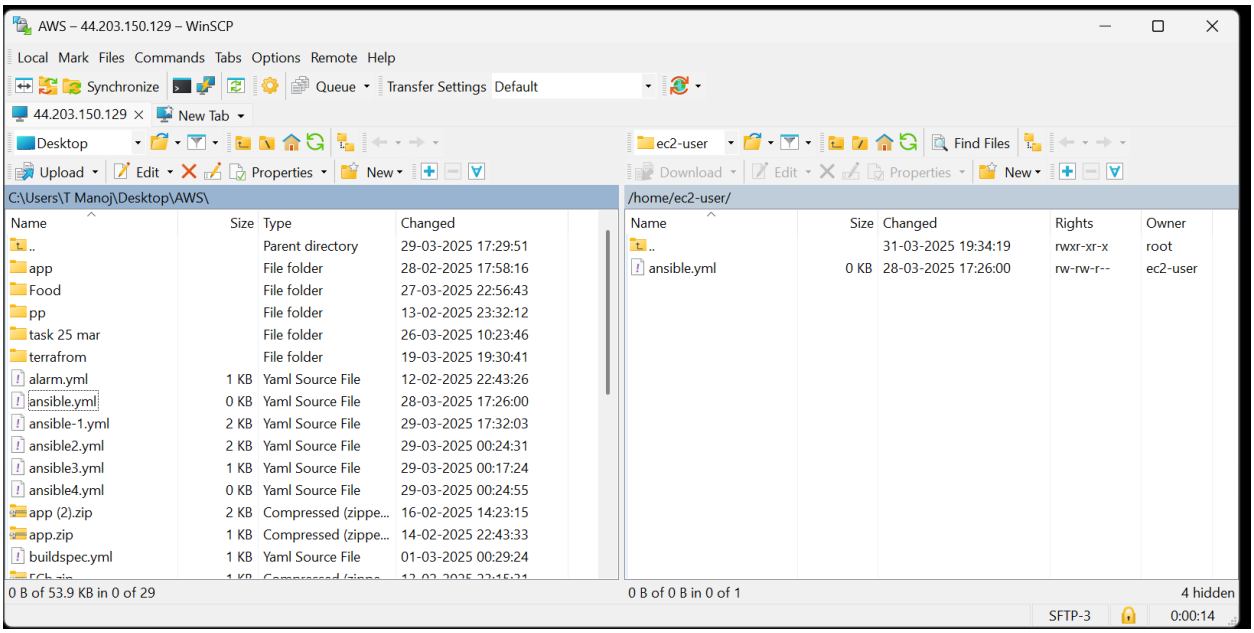
2. Transfer Code to EC2 Server

Step 2: Use WinSCP to Copy Code

1. Download and install **WinSCP**
2. Open **WinSCP**
3. Select **SFTP** as the file protocol
4. Enter **Public IP** of EC2 and SSH key details
5. Transfer files from your local system to the EC2 instance


Login
New Site
Session
File protocol:
SFTP
Host name:
44.203.150.129
Port number:
22
User name:
Password:
Save
Advanced...
Tools
Manage
Login
Close
Help
Show Login dialog on startup and when the last session is closed


Username – 44.203.150.129
Searching for host...
Connecting to host...
Authenticating...
Username:
ec2-user
OK
Cancel
Help



Step 3: Verify Files on EC2

1. SSH into EC2:

```
ls -l
```

Ensure the uploaded files are present.

3. Initialize a Git Repository on EC2

Step 4: Create a Working Directory

1. Create a directory for your project:

```
mkdir gitproc
cd gitproc
```

2. Move your files into this directory:

```
ec2-user@ip-172-31-86-82:~/gitproc
[ec2-user@ip-172-31-86-82 ~]$ ll
total 0
-rw-rw-r--. 1 ec2-user ec2-user 0 Mar 28 11:56 ansible.yml
[ec2-user@ip-172-31-86-82 ~]$ cd /home/ec2-user
[ec2-user@ip-172-31-86-82 ~]$ ll
total 0
-rw-rw-r--. 1 ec2-user ec2-user 0 Mar 28 11:56 ansible.yml
[ec2-user@ip-172-31-86-82 ~]$ mkdir gitproc
[ec2-user@ip-172-31-86-82 ~]$ cd gitproc/
[ec2-user@ip-172-31-86-82 gitproc]$ scp /home/ec2-user/ansible.yml .
[ec2-user@ip-172-31-86-82 gitproc]$ l
-bash: l: command not found
[ec2-user@ip-172-31-86-82 gitproc]$ ll
total 0
-rw-r--r--. 1 ec2-user ec2-user 0 Mar 31 14:21 ansible.yml
[ec2-user@ip-172-31-86-82 gitproc]$
```

Step 5: Initialize Git

1. Run the following command to initialize a new Git repository:

```
git init
```

```
-rw-r--r--. 1 ec2-user ec2-user 0 Mar 31 14:21 ansible.yml
[ec2-user@ip-172-31-86-82 gitproc]$ 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/ec2-user/gitproc/.git/
[ec2-user@ip-172-31-86-82 gitproc]$
```

2. Add all files to the repository:

```
git add .
```

```
[ec2-user@ip-172-31-86-82 gitproc]$ git add .
```

3. Commit the changes:

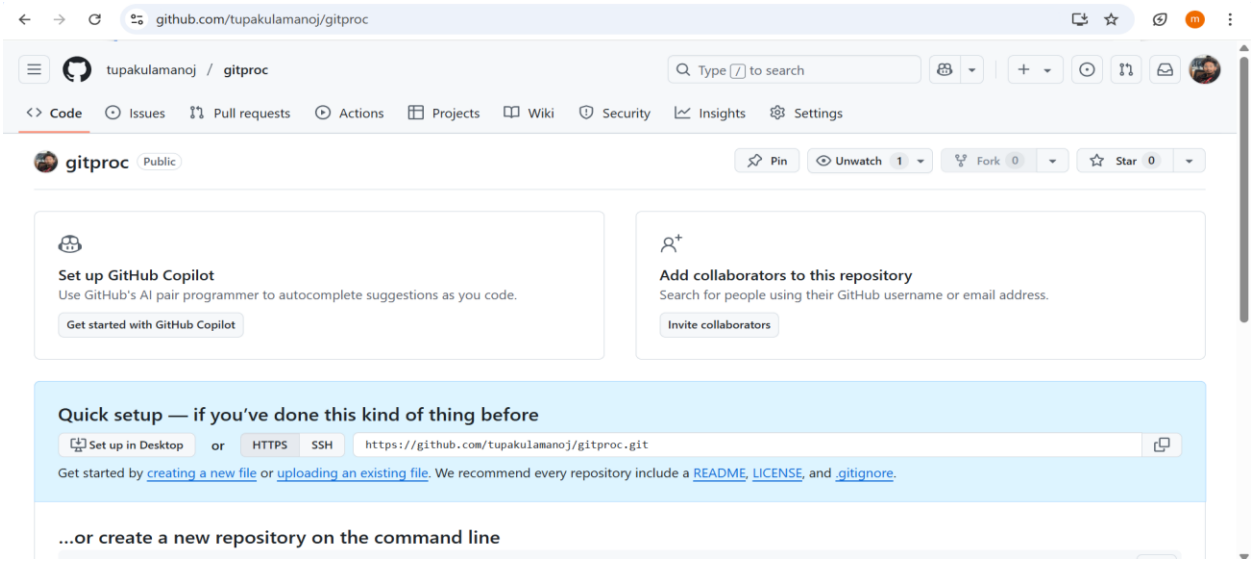
```
git commit -m "Initial commit"
```

```
[ec2-user@ip-172-31-86-82 gitproc]$ git commit -m "initial commit"
[master (root-commit) 4d3ce86] initial commit
 Committer: EC2 Default User <ec2-user@ip-172-31-86-82.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 ansible.yml
[ec2-user@ip-172-31-86-82 gitproc]$
```

4. Push Code to GitHub

Step 6: Create a GitHub Repository

1. Go to GitHub and log in
2. Click on **New Repository**
3. Provide a **Repository Name**, select **Public/Private**, and click **Create**
4. Copy the repository **HTTPS/SSH URL**



Step 7: Add Remote Repository

1. Add the GitHub repository as the remote origin:

```
git remote add origin
```

2. Verify the remote URL:

```
git remote -v
```

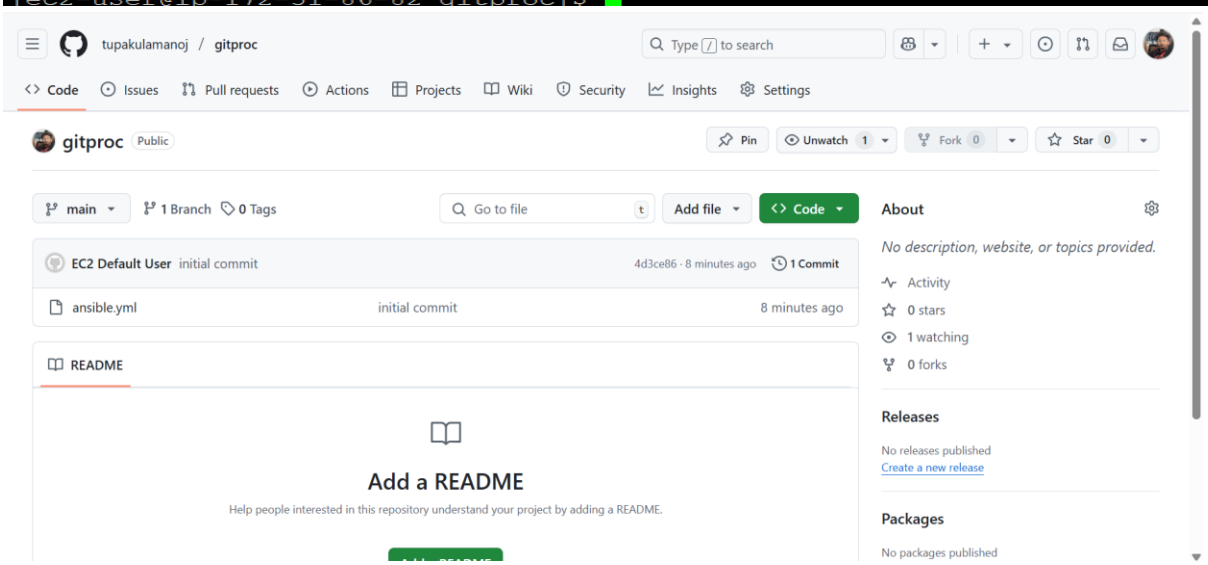
```
[ec2-user@ip-172-31-86-82 gitproc]$ git remote add origin https://github.com/tupakulamanoj/gitproc.git
[ec2-user@ip-172-31-86-82 gitproc]$ git remote -v
origin  https://github.com/tupakulamanoj/gitproc.git (fetch)
origin  https://github.com/tupakulamanoj/gitproc.git (push)
[ec2-user@ip-172-31-86-82 gitproc]$
```

Step 8: Push Code to GitHub

1. Push the code:

```
git push -u origin main
```

```
[ec2-user@ip-172-31-86-82 gitproc]$ git branch -m main
[ec2-user@ip-172-31-86-82 gitproc]$ git push -u origin main
Username for 'https://github.com': tupakulamanoj
Password for 'https://tupakulamanoj@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 232 bytes | 232.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/tupakulamanoj/gitproc.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
[ec2-user@ip-172-31-86-82 gitproc]$
```



5. Handling Merge Conflicts

Step 9: Simulating a Merge Conflict

1. **Create a new working directory**

```
mkdir gitproc1
cd gitproc1
```

```
Initialized empty Git repository in /home/ec2-user/gitproc1/.git/
[ec2-user@ip-172-31-86-82 gitproc1]$ git pull https://github.com/tupakulamanoj/g
itproc.git
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 212 bytes | 212.00 KiB/s, done.
From https://github.com/tupakulamanoj/gitproc
 * branch            HEAD       -> FETCH_HEAD
[ec2-user@ip-172-31-86-82 gitproc1]$
```

Create a new feature branch

```
git branch hello
```

```
ec2-user@ip-172-31-86-82:~/first
[ec2-user@ip-172-31-86-82 first]$ git branch
  febranch1
  hello
* main
[ec2-user@ip-172-31-86-82 first]$
```

2. Modify a file and commit the changes

```
git add file.txt
git commit -m "second commit"
```

```
ec2-user@ip-172-31-86-82:~/gitproc1
  GNU nano 8.3                    sam.txt                    Modified
this is sample message

^G Help      ^O Write Out  ^F Where Is   ^K Cut      ^T Execute   ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste    ^J Justify   ^/ Go To Line
```

```
[ec2-user@ip-172-31-86-82 gitproc1]$ git commit -m "second commit"
[master 0ba0256] second commit
 Committer: tupakulamanoj <ec2-user@ip-172-31-86-82.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 sam.txt
[ec2-user@ip-172-31-86-82 gitproc1]$
```

2. **Merge feature branch into main**

```
git merge hello
```

```
[ec2-user@ip-172-31-86-82 first]$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)
[ec2-user@ip-172-31-86-82 first]$ git merge hello
Auto-merging ind.txt
CONFLICT (content): Merge conflict in ind.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Step 10: Resolving Merge Conflict

1. Check the conflicting file:

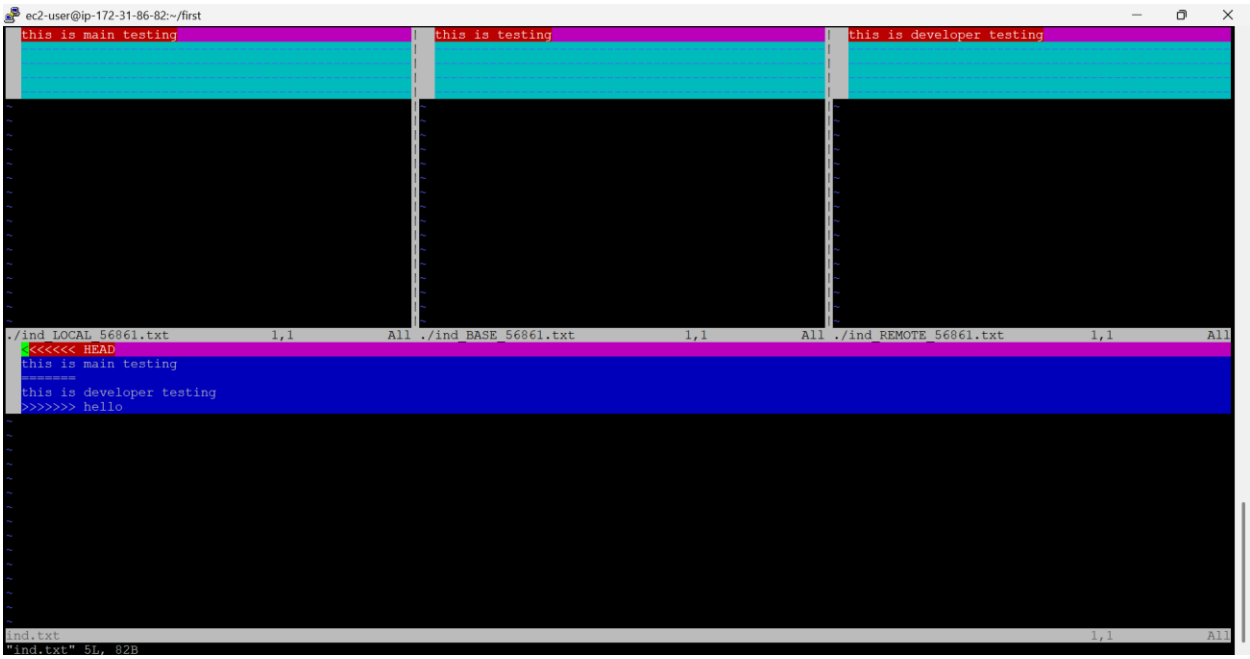

```
[ec2-user@ip-172-31-86-82 first]$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)
[ec2-user@ip-172-31-86-82 first]$ git merge hello
Auto-merging ind.txt
CONFLICT (content): Merge conflict in ind.txt
Automatic merge failed; fix conflicts and then commit the result.
```


2. **Edit the file to resolve the conflict**

```
[ec2-user@ip-172-31-86-82 first]$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)
[ec2-user@ip-172-31-86-82 first]$ git merge hello
Auto-merging ind.txt
CONFLICT (content): Merge conflict in ind.txt
Automatic merge failed; fix conflicts and then commit the result.
```

3. **Use Git Merge Tool (Optional)**

```
git mergetool
```



4. **Mark the conflict as resolved and commit the changes**

```
git add file.txt
git commit -m "last commit"
```

```
[ec2-user@ip-172-31-86-82 first]$ git commit -m "lastcommit"
[main c70d6f4] lastcommit
 Committer: tupakulamanoj <ec2-user@ip-172-31-86-82.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author
```

5. Push changes to GitHub

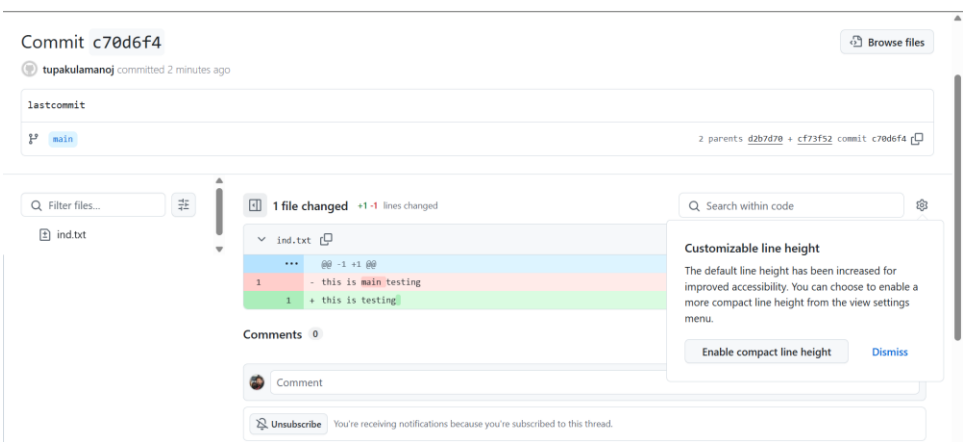
```
git push origin main
```

```
[ec2-user@ip-172-31-86-82 first]$ git push -u origin main
Username for 'https://github.com': tupakulamanoj
Password for 'https://tupakulamanoj@github.com':
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (21/21), 1.69 KiB | 863.00 KiB/s, done.
Total 21 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/tupakulamanoj/gitproc.git
   d6b62b3..c70d6f4  main -> main
branch 'main' set up to track 'origin/main'.
[ec2-user@ip-172-31-86-82 first]$
```

6. Verify Changes on GitHub

1. Go to your GitHub repository
2. Check the **commits** and **branches** to ensure everything is merged successfully

```
[ec2-user@ip-172-31-86-82 first]$ git log --oneline
c70d6f4 (HEAD -> main) lastcommit
cf73f52 (hello) developer commit
d2b7d70 test commit
15bcdb0 ind commit
5d226d7 (febranch1) first main commit
f09c0f8 dev commit
1ca1b69 first commit
d6b62b3 (origin/main) main commit
4d3ce86 initial commit
[ec2-user@ip-172-31-86-82 first]$ ll
total 16
-rw-r--r--. 1 ec2-user ec2-user  0 Apr  1 04:59 ansible.yml
-rw-r--r--. 1 ec2-user ec2-user 17 Apr  1 05:43 ind.txt
-rw-r--r--. 1 ec2-user ec2-user 82 Apr  1 05:39 ind.txt.orig
-rw-r--r--. 1 ec2-user ec2-user 20 Apr  1 05:00 index.html
-rw-r--r--. 1 ec2-user ec2-user 30 Apr  1 05:10 sample.txt
[ec2-user@ip-172-31-86-82 first]$ cat ind.txt
this is testing
[ec2-user@ip-172-31-86-82 first]$ cat ind.txt.orig
<<<<<<< HEAD
this is main testing
=======
this is developer testing
>>>>>>> hello
[ec2-user@ip-172-31-86-82 first]$ git push -u origin main
Username for 'https://github.com': tupakulamanoj
Password for 'https://tupakulamanoj@github.com':
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (21/21), 1.69 KiB | 863.00 KiB/s, done.
Total 21 (delta 7), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (7/7), done.
To https://github.com/tupakulamanoj/gitproc.git
   d6b62b3..c70d6f4  main -> main
branch 'main' set up to track 'origin/main'.
[ec2-user@ip-172-31-86-82 first]$
```



OTHER COMMANDS

1. Git Stash

Temporarily saves changes and restores a clean working directory.

```
git stash          # Save changes
git stash list     # View all stashed changes
git stash apply    # Reapply the most recent stash
git stash pop      # Apply and remove the stash from the list
git stash drop     # Delete the most recent stash
```

2. Git Fork

Creates a copy of a remote repository on your GitHub account.

- Go to the **GitHub repository** → Click **Fork**
- Clone the forked repository locally:

```
git clone <forked-repo-URL>
```

3. Git Cherry-Pick

Applies a specific commit from one branch to another.

```
git checkout main           # Switch to the main branch
git cherry-pick <commit-ID> # Apply a specific commit
```

4. Git Diff

Shows differences between files.

```
git diff                  # Show differences in the working directory
git diff --staged         # Show staged but uncommitted changes
git diff <branch1> <branch2> # Compare two branches
```

5. Git Restore

Discards changes in the working directory.

```
git restore <file>        # Restore a specific file
git restore --staged <file> # Unstage a file
```

6. Git Reset

Unstages or reverts changes in the working directory.

```
git reset HEAD <file>      # Unstage a file
git reset --hard <commit>  # Reset repository to a specific commit,
discarding changes
git reset --soft <commit>  # Reset but keep changes staged
```

7. Git Revert

Creates a new commit that undoes changes from a specific commit.

```
git revert <commit-ID> # Revert a commit
```

8. Git Clone

Copies a remote repository to your local machine.

```
git clone URL # Clone a repository
```

9. Git Reflog

Shows history of branch movements and resets.

```
git reflog          # Display commit history, resets, and movements
```

10. Git Squash

Combines multiple commits into one (done using interactive rebase).

```
git rebase -i HEAD~3  # Squash last 3 commits
```

- Change `pick` to `squash` for merging commits.

11. Git Tagging

Marks specific commits (e.g., version releases).

```
git tag v1.0              # Create a tag
git tag -a v1.0 -m "Version 1.0" # Annotated tag
git push origin v1.0      # Push tag to remote
```

12. Git Amend

Modifies the most recent commit.

```
git commit --amend -m "Updated commit message"
```

13. Git Blob

Represents file content without metadata.

```
git hash-object -w <file>  # Store file as a blob
git cat-file -p <blob-ID>  # Show blob content
```

14. Git Worktree

Creates additional working directories linked to the same repository.

```
git worktree add <path> <branch>  # Add a worktree
git worktree list                 # List all worktrees
git worktree remove <path>         # Remove a worktree
```

15. Git GC (Garbage Collection)

Cleans up unnecessary files and optimizes repository.

```
git gc                 # Run garbage collection
git gc --prune=now     # Remove unreferenced objects
```