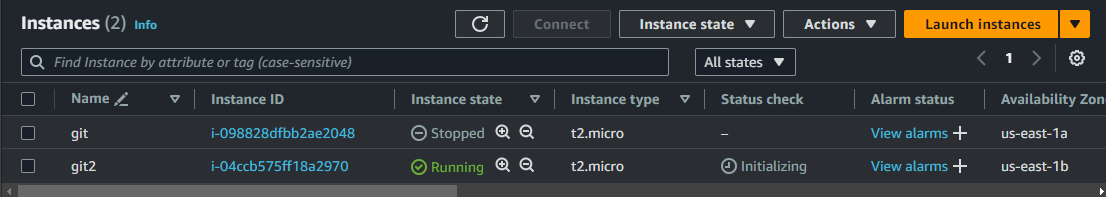
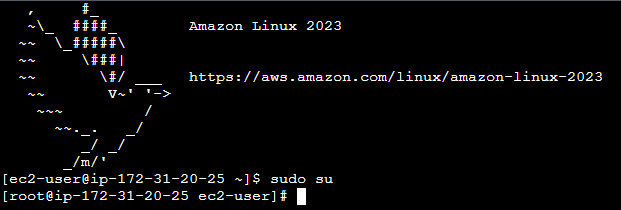
**Git – GitHub**

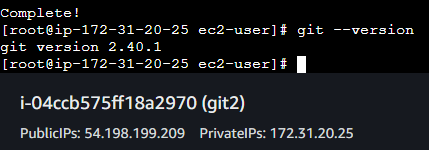
* Process for creating a repository on GitHub and pushing the local Repository to the Centralized server.
* First we need an instance.
* Launch Instance and give the inbound rule for the HTTP.
* We need http because we want to push the local repo to the GitHub.
* Instance Created.



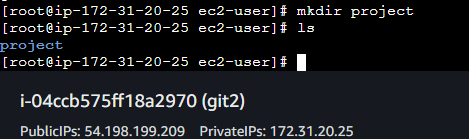
* Now connect the Instance.
* Login with the root user as ec2-user don’t have permissions.



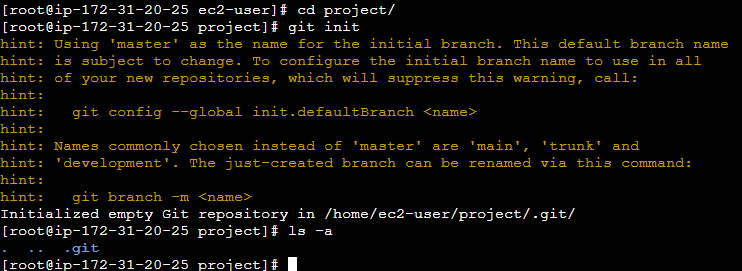
* Now we need to install git.
* Hit command “**yum install git -y**” to install git.
* Git is installed, hit command “**git --version**” to check the version of the git.



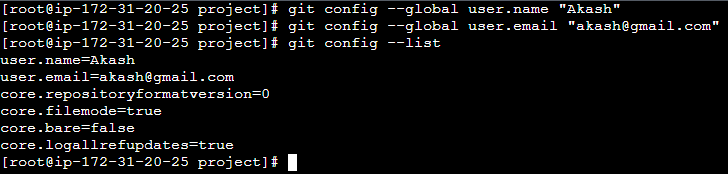
* Now we need a directory to make a local repository.
* Hit command “**mkdir directoryname**” to create a directory.



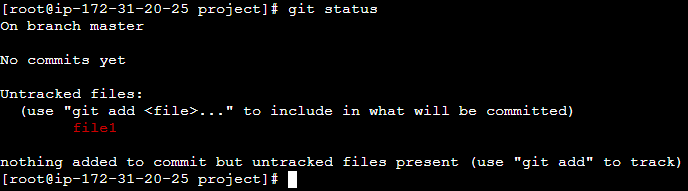
* Now we need to enter into the directory recently created.
* Now we need to initialize the directory to local repository.
* Hit command “**git init**”.
* Upon initializing the git a file named .git will be created.
* Hit command “**ls -a**” to view the .git file.



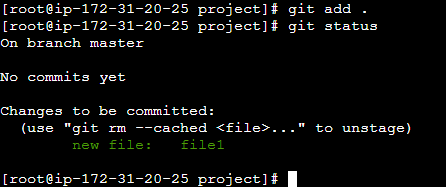
* We have to configure the git.
* Hit command “**git config –global user.name “Akash”.**
* “**git config –global user.email “akash@gmail.com”.**
* Now we have added username and email to the configuration.
* To view the username and email hit command “**git config –list**”.



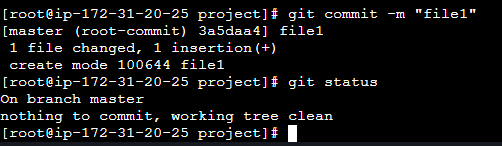
* Now we need a file, create a file and add something in it.



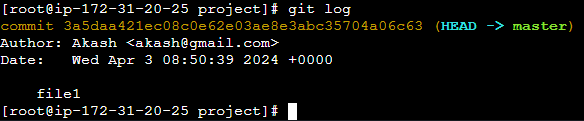
* Now the file is in workspace area we need to add it in the staging area.
* Hit command “**git add .**”



* Now the file is added in the staging area.
* We have to add the file in the local repository.
* Hit command “**git commit -m “file1**”



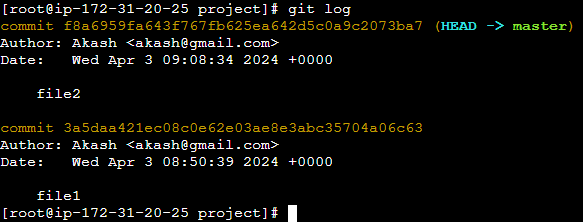
* Now the file is added to the local repository and there is nothing to commit.
* Git **“git log**” to check the author, email and commit id with data and time.



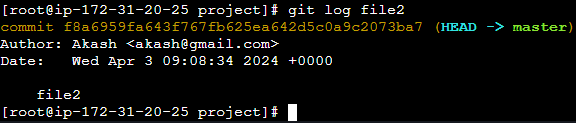
* We can also view the log in one line with command “**git log –oneline**”



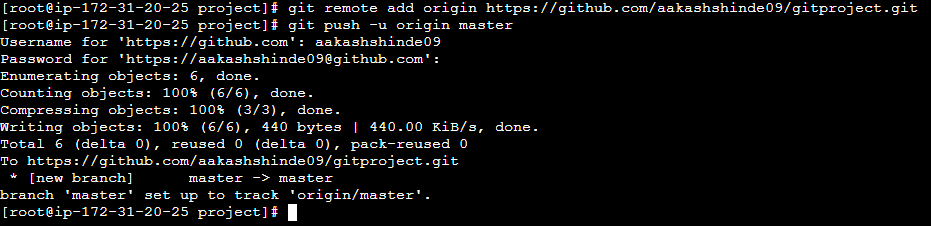
* Now create another file and add it to the local repository.
* Hit “git log”.
* Git log shows all the logs but if we want to find the log for a particular file then we can find it with the commit comment given.



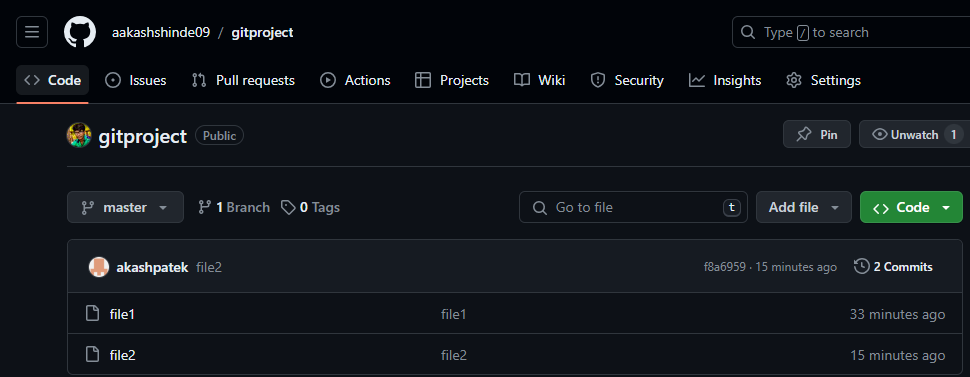
* We have to only see the log of file2.
* Hit command “**git log file2**”.



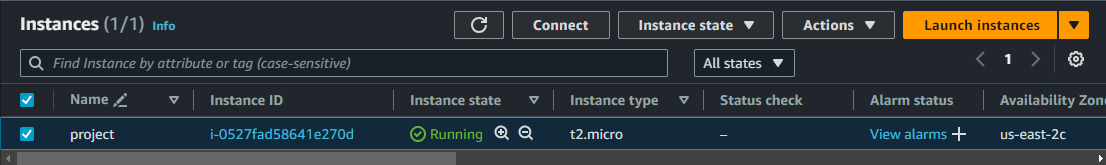
* Now we need to push it to the GitHub.
* Login to GitHub and create a public repository.
* Now we need to add the URL of the GitHub repo to the Git.
* Hit command “**git remote add origin** [**https://github.com/aakashshinde09/gitproject.git**](https://github.com/aakashshinde09/gitproject.git)”.
* Now we need to create a token for the authorization purpose as we don’t have password.
* Go to your GitHub account, Click on the profile Icon and click settings.
* Scroll down and Click Developer settings.
* Click on the personal access tokens and then **click Tokens(classic).**
* Click on Generate new token(classic).
* Enter the name for the token.
* Select the scope for the token.
* Scope is the permissions for the user using token.
* Click generate token.
* Now copy the token and save it somewhere safe as it is shown only once.
* Go back to the instance.
* Hit command “**git push -u origin master**”.
* Now it will ask for the username.
* Enter the username and for the password paste the token which you have copied.



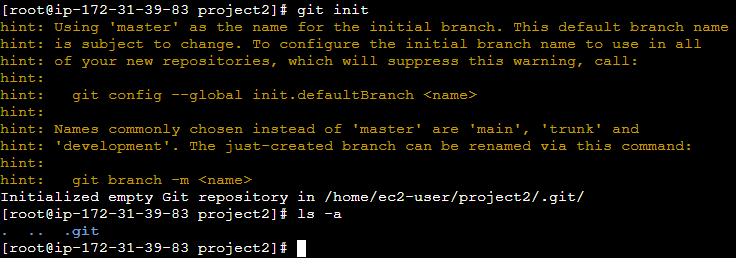
* Now check the GitHub the files should be visible there.



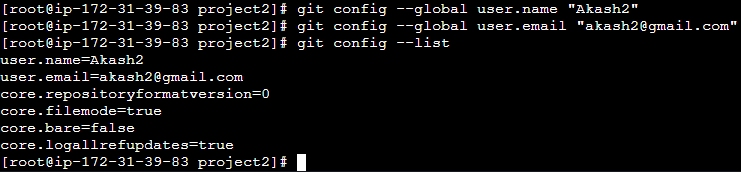
* Our push operation is successful.
* Now we need to pull the files in another instance.
* Create a new instance.



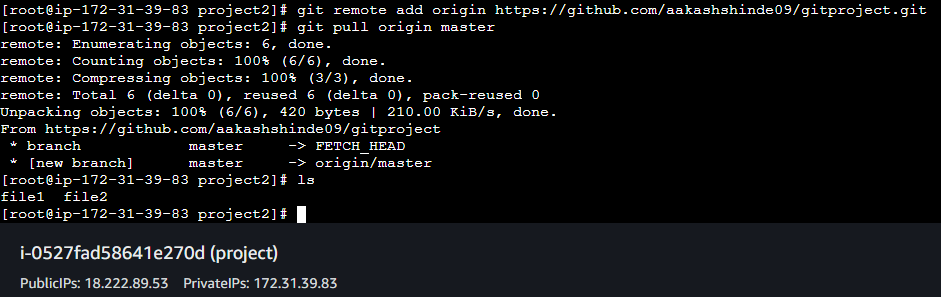
* Now connect the instance.
* Login with root user.
* Now install git.
* Create a new directory and initialize it.



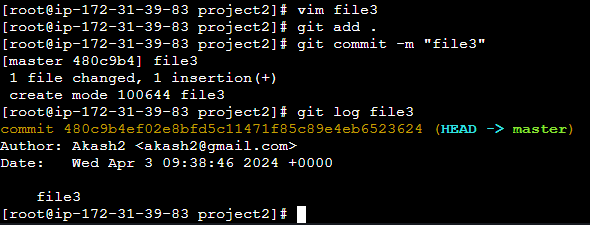
* Configure git, add username and email.



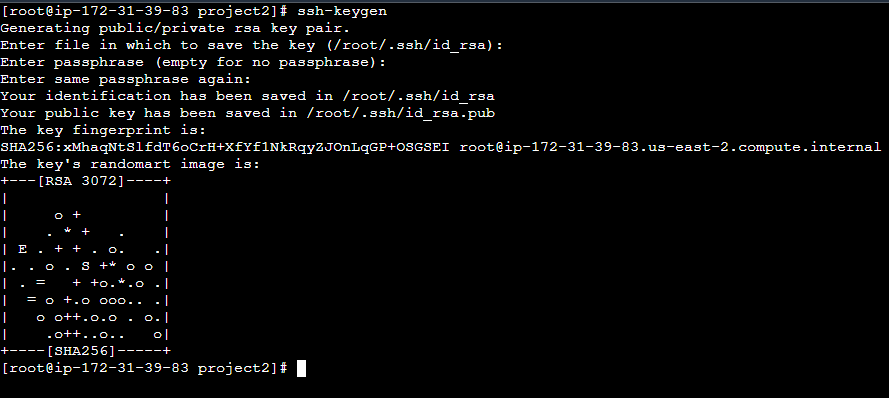
* Now add the origin of the GitHub repo.
* Hit command “**git pull origin master**” to pull the files from the GitHub repo.
* Hit “ls” to check the files.



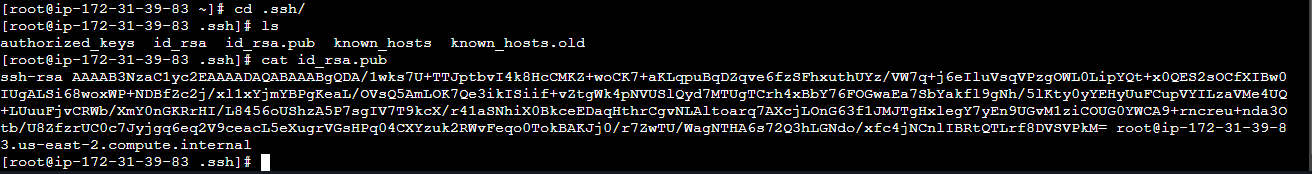
* Now edit the files pulled or create a new file to push.
* Create a file file3.
* Now add the file to the local repo follow the process below.



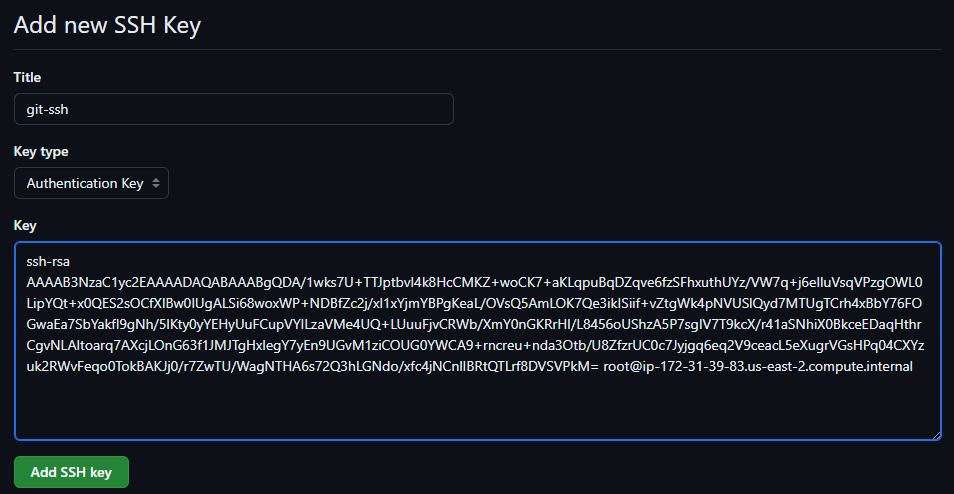
* Now we need to push the file to the GitHub.
* For that we can use the token or we can use SSH key.
* Now hit command “ssh -keygen” to generate the key.



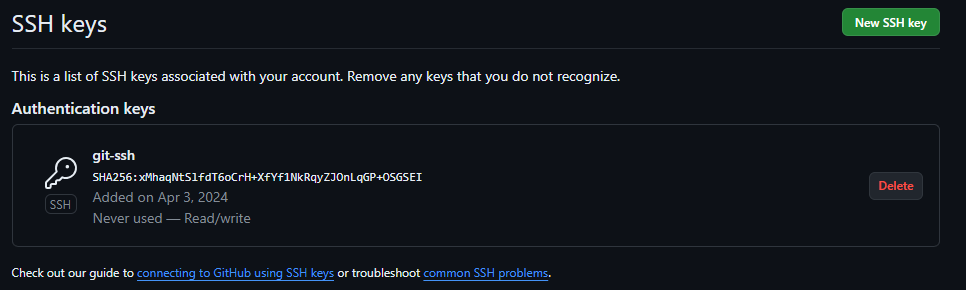
* Now go to the generated key location.
* And copy the public key.



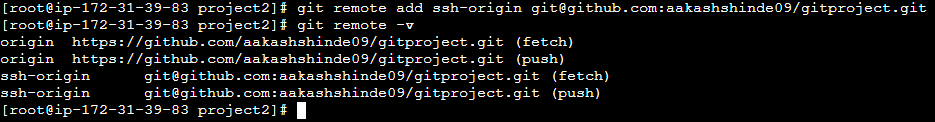
* Now come back to the GitHub account and click the profile icon.
* Click on settings.
* Click on SSH and GPG keys.
* Enter the name for the ssh key.
* Now paste the public key and click Add SSH key.



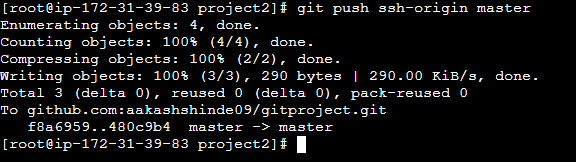
* SSH key added.



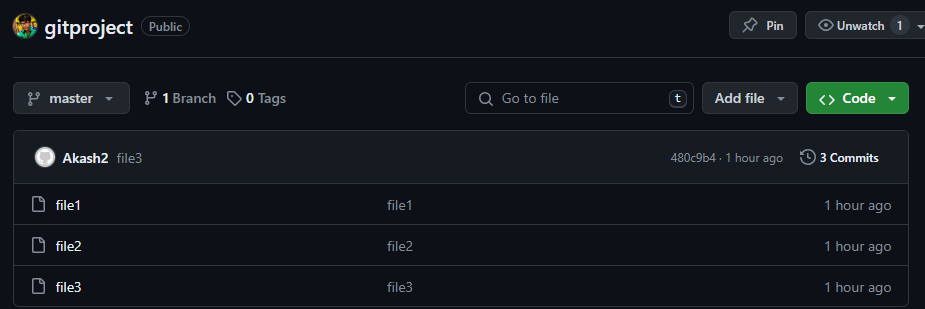
* Now copy the SSH URL.
* Come back to the git and go to the directory where Git is initialized.
* Hit command “git remote add ssh-origin SSH\_URL”.
* Hit command “git remote -v” to check the origins added.



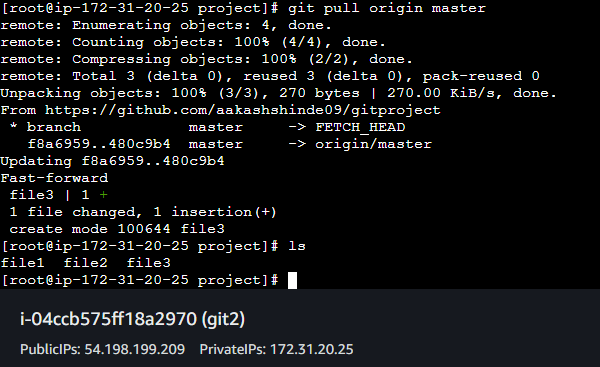
* Now hit command “git push ssh-origin master”



* Now check the GitHub.
* File3 added successfully.

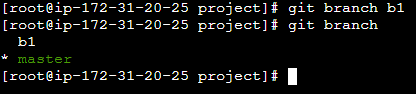


* Now we need to pull this on another instance.

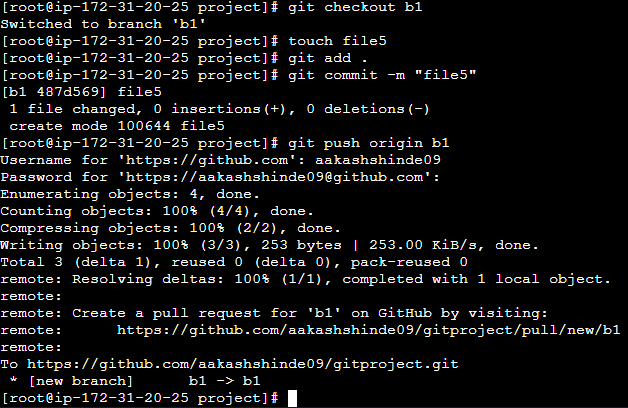


Git – Branch – Merge

* Creating a branch in git.
* Hit command “git branch branchname”.
* Hit command “git branch” to view all the branches and check the current branch.



* Creating file on branch b1.
* Hit command “git checkout b1” to jump on b1 branch.
* Upon creating a branch all the files will be copied to the branch.
* Create file, add the file to the staging area and then commit it.
* Push the file to the GitHub.
* The branch b1 will be created automatically on the GitHub.

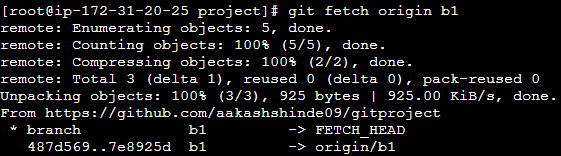


**git fetch:**

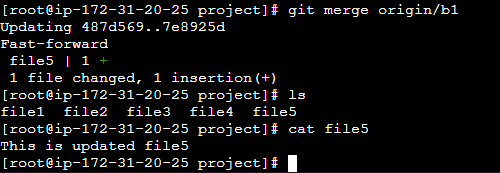
* git fetch retrieves the latest changes from the remote repository, including branches and tags, and stores them in your local repository's database. However, it doesn't integrate these changes into your working copy.
* It's useful for reviewing changes before merging or pulling them into your local branches. After fetching, you can inspect the changes using commands like git log origin/master to see the commits on the remote branch.
* Fetching doesn't change your local branches; it only updates the remote tracking branches such as origin/master.

**git pull:**

* git pull not only fetches the changes from the remote repository but also merges them into your current branch (or rebases your changes on top of the fetched changes, depending on your configuration).
* It's essentially a combination of git fetch followed by git merge or git rebase, depending on your configuration.
* git pull is useful when you want to update your local branch with the latest changes from the remote and immediately integrate them into your working copy.
* Using Git fetch.
* First we need to make some changes in the file on GitHub.
* After making changes hit command “git fetch origin b1”.

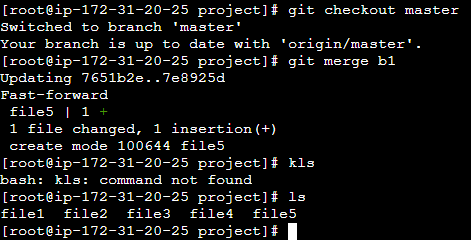


* Now the file is fetched but if we try to see the updated content it will not be visible as we have fetched the file.
* To see the content, we have to merge it.
* Hit command “git merge origin/b1”
* Then you can see the content of the file using cat command.



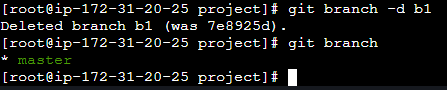
Merging the branch b1 in master branch.

* To merge the b1 branch to the master branch first we need to jump to the master branch.
* Hit command “git checkout master” to jump to the master branch.
* Hit command “git merge b1”.
* The b1 branch will be merged with the master branch.



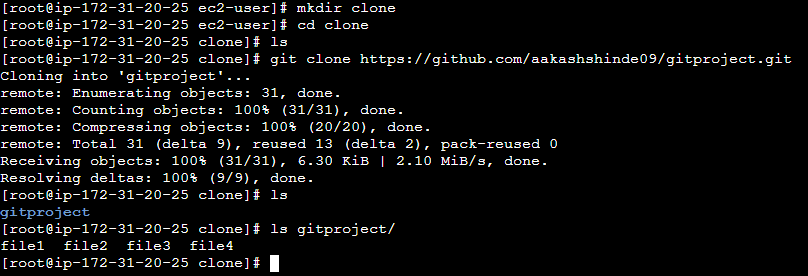
Deleting the branch

* To delete the branch hit command “git branch -d b1”.



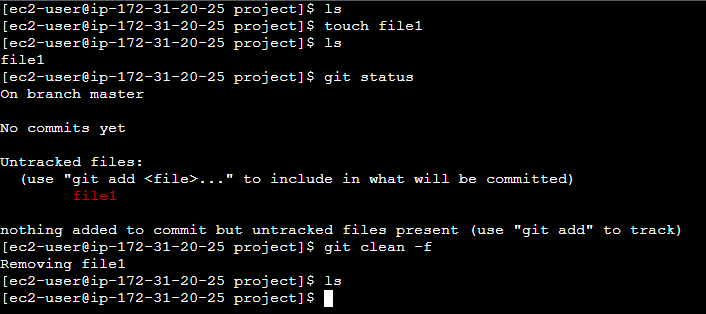
Git Clone :

git clone is a Git command used to create a copy of a remote repository onto your local machine. It copies all the files, branches, and commit history from the remote repository to your local machine. This is typically the first step you would take when starting to work with a project hosted in a Git repository.



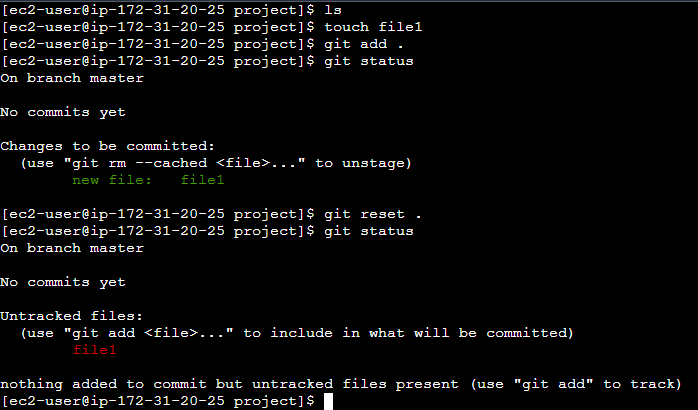
git clean:

* git clean is used to remove untracked files from the working directory. These are files that are not staged or committed to the Git repository.
* By default, git clean only removes untracked files. However, it can also be used with flags to remove directories and ignored files.



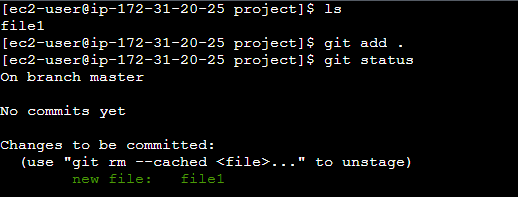
git reset:

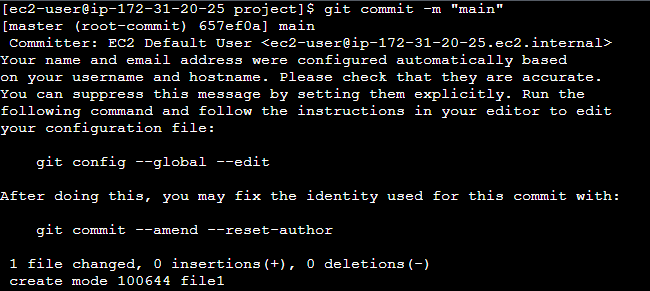
* git reset is a powerful command used to reset the current HEAD to a specified state. It can be used to undo changes in various ways:
* git reset --soft moves HEAD to a previous commit but keeps the changes in the staging area.
* git reset --mixed moves HEAD to a previous commit and unstages the changes, but keeps them in the working directory.
* git reset --hard moves HEAD to a previous commit and discards all changes, both staged and unstaged, back to that commit. This is a potentially dangerous operation as it can't be undone easily and permanently deletes unsaved changes.
* git reset can also be used to move branches to a different commit, effectively rewriting history. However, this should be used with caution, especially on shared branches, as it can cause confusion and conflicts for collaborators.

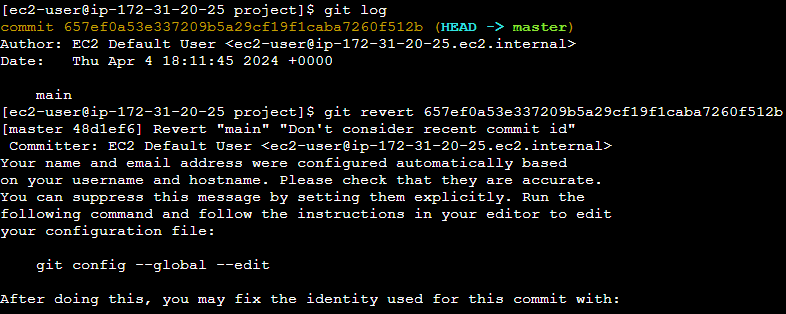


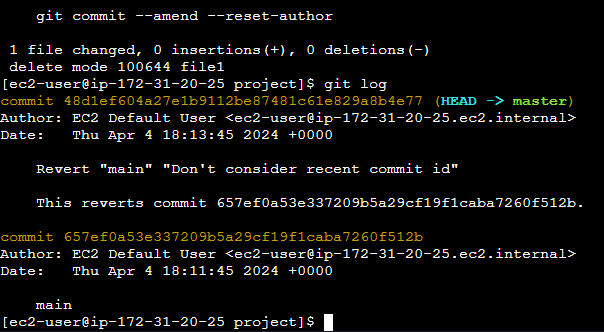
git revert:

* git revert is used to create a new commit that undoes the changes made by a previous commit. It doesn't delete history; instead, it creates a new commit that inversely applies the changes introduced by the specified commit.
* This is useful when you want to undo changes made by a specific commit without altering the repository's history. It's often preferred for shared repositories where rewriting history is discouraged.



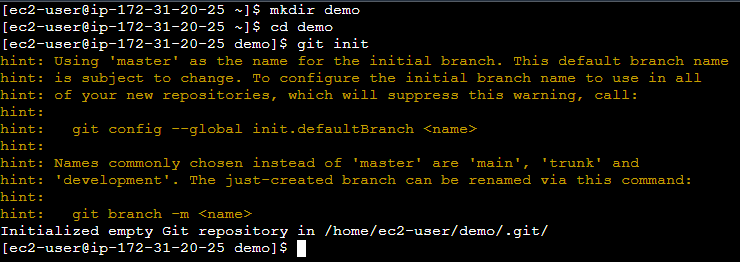




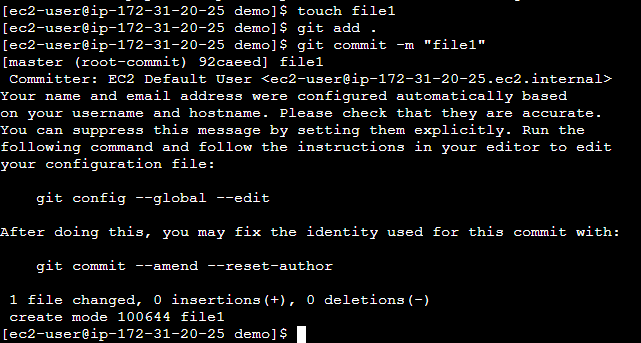


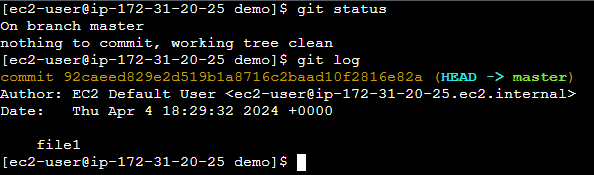
GitLab

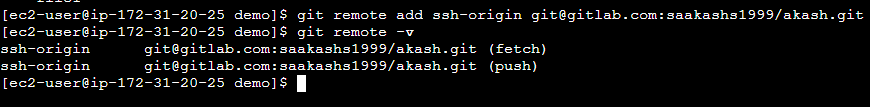
* Create a new project in Gitlab.
* Open the instance the create a directory.
* Initialize the directory to the local repo.



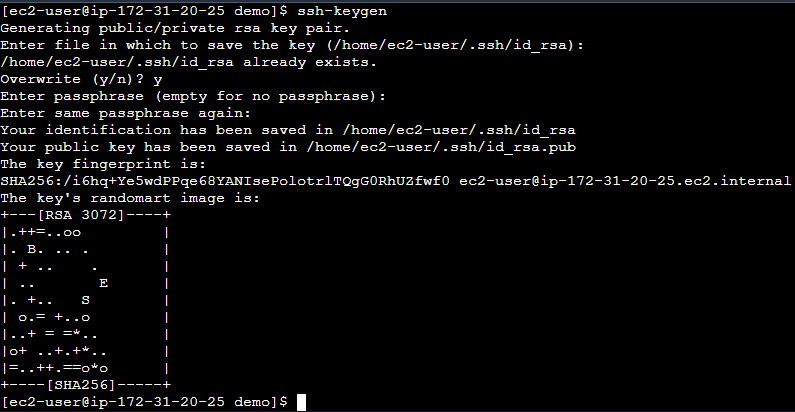
* Now create a file.
* Add the file to the staging area.
* Commit the file to add it in the local repository from the staging area.
* Now copy the ssh link.
* Add the origin URL.
* Hit command “git remote add ssh-origin URL”.
* Hit command “git remote -v” to check the remote URLs.



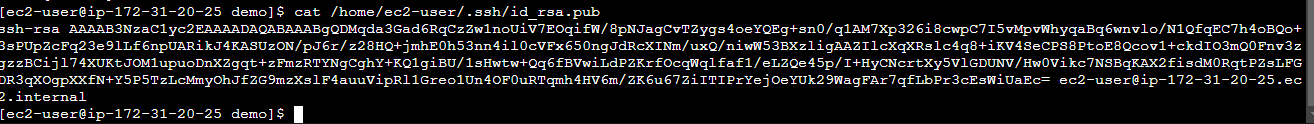




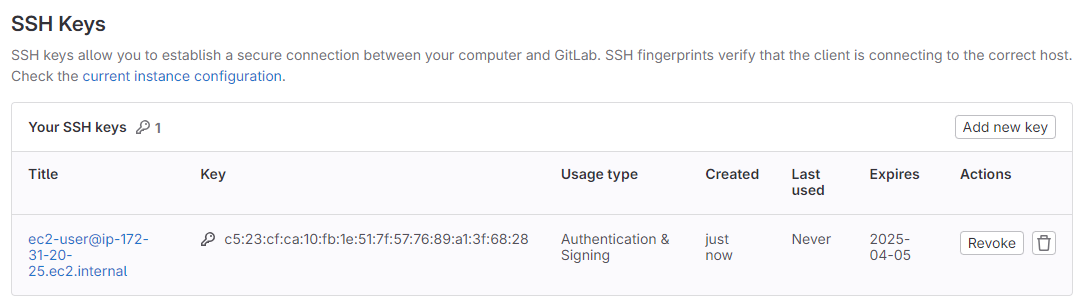
* Now we need to generate a ssh-key.
* Hit command “ssh-keygen”.



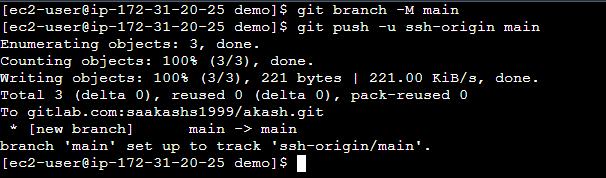
* Now copy the key from the location.



* Now come back to Gitlab.
* Click on the profile Icon.
* Click Edit profile.
* Click SSH keys.
* Click Add new.
* Now paste the key.
* Click Add key.
* SSH key will be created.



* Here in Gitlab the branch is Main.
* The branch in our Git is master.
* We need to change the branch from master to main.
* Hit command “git branch -M main” to convert the master branch to main.
* Now hit command “git push -u ssh-origin main”.



* Now check the Gitlab our file will be visible there.

