Git BOTTOM TOP

Initial consideration was

<https://github.com/torvalds/linux.git>

* But it took over an hour to clone the repository using

git clone

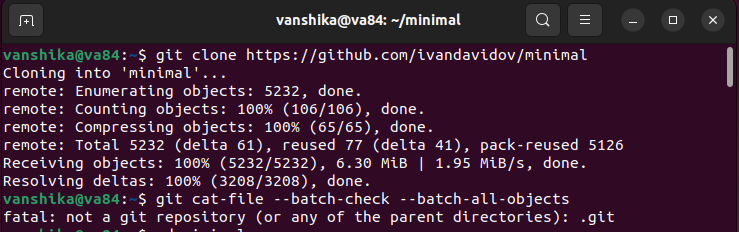
* Then

git cat-file –batch-check –batch-all-objects took over 2 hours

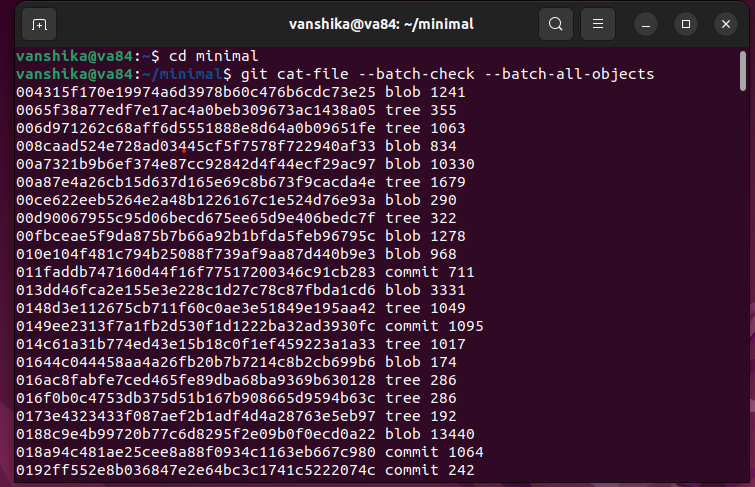
but couldn’t extract all the objects list.

So, the repository being used is

<https://github.com/ivandavidov/minimal.git>

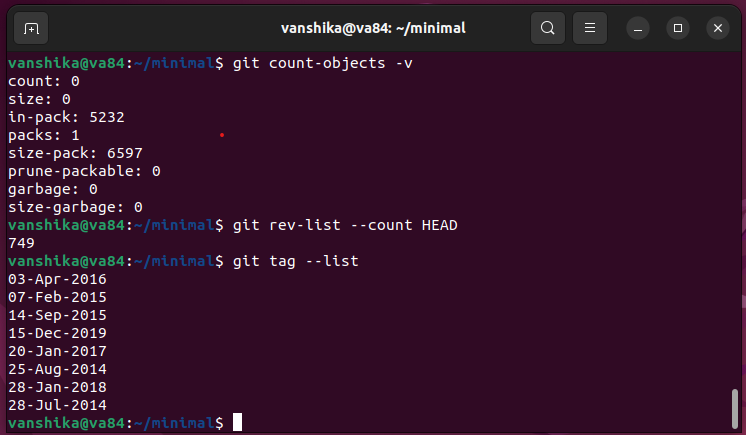


git cat-file –batch-check –batch-all-objects

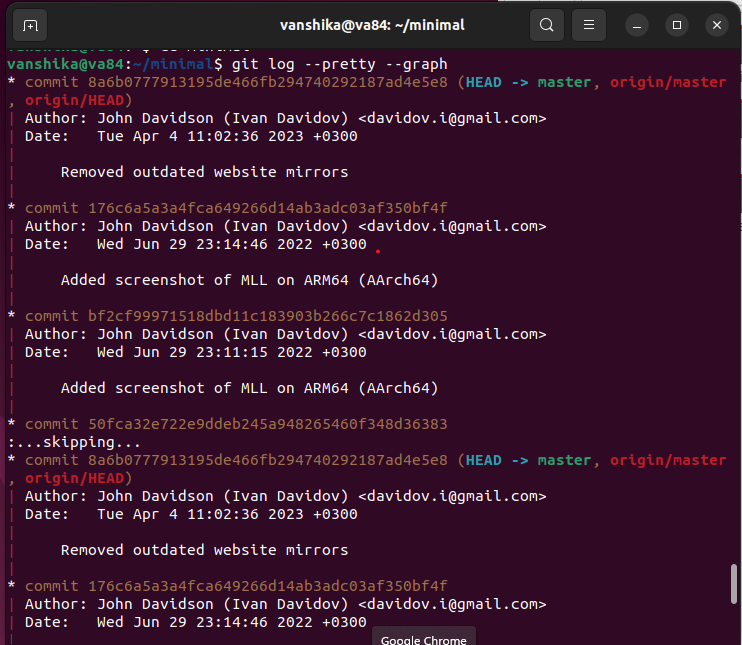


Got the entire list of objects, blobs, trees, commits, tags in the repository.

* git count-objects -v (gives the total number of objects (including blobs and trees))
* git rev-list –count HEAD (gives the number of commits)
* git tag –list (gives the number of tags)

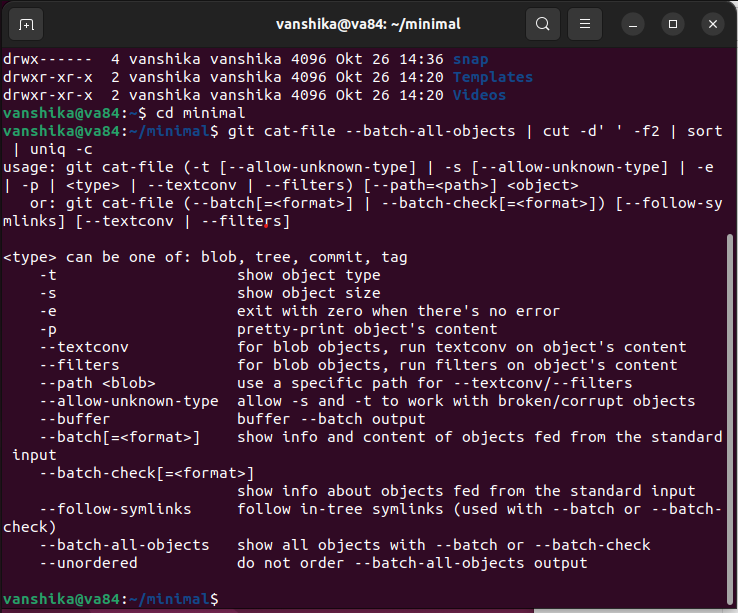


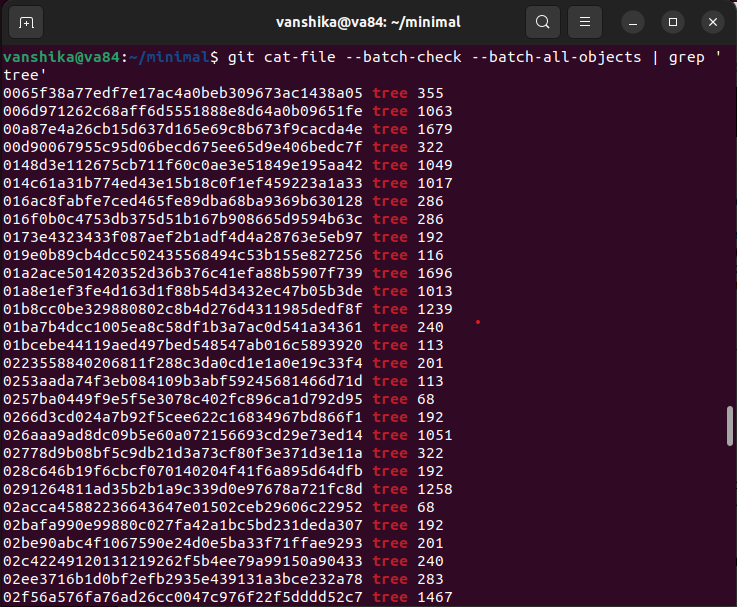
git log –pretty –graph (gives the commits and the hash)



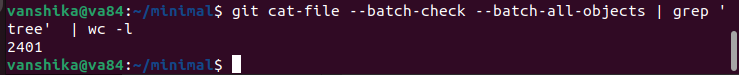
git cat-file –batch-check –batch-all-objects | cut -d’ ‘ -f2 | sort | uniq -c

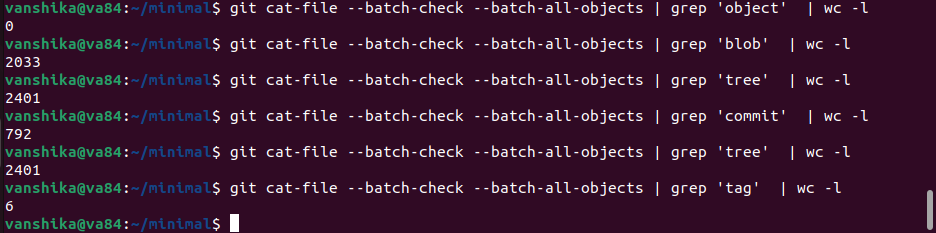
(This command retrieves information about all objects in the repository, extracts the object types, sorts them, and then counts the unique occurrences of each types)



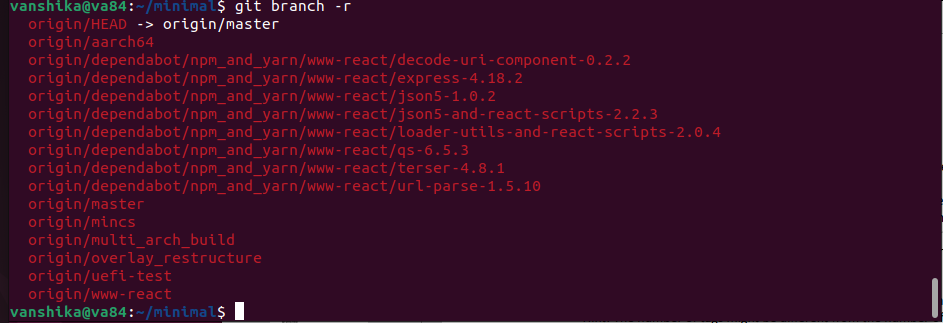


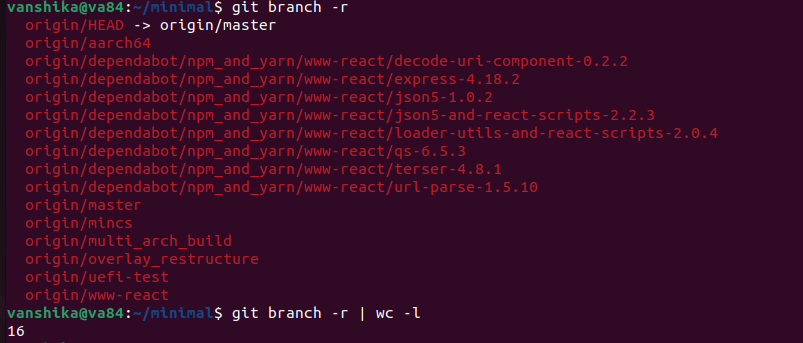
git cat-file –batch-check –batch-all-objects | grep ‘tree’ | wc -l



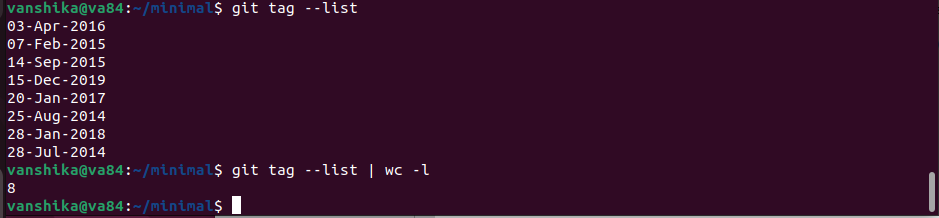


git branch -r (to check all the branches available in the repository)





git tag –list



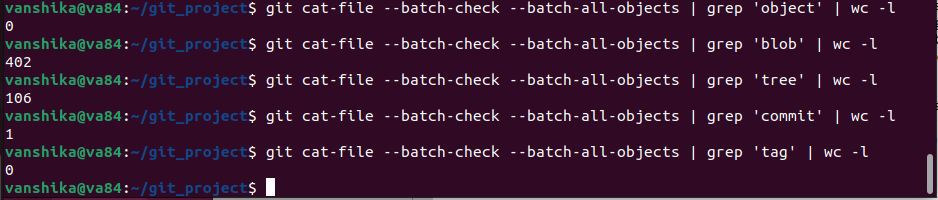
git clone –depth=1 URL FOLDER

* ‘git clone’: This is the command used to clone git repository.
* ‘—depth=1’: This option specifies that only the latest commit and its associated files should be cloned, creating a shallow clone with a history of depth 1.
* ‘URL’: This is the URL of the git repository to be cloned.
* ‘FOLDER’: This is the name of the local folder where you want to clone the repository.

This command will create a shallow clone of the repository in the ‘FOLDER’ with only the latest commits and its associated files.

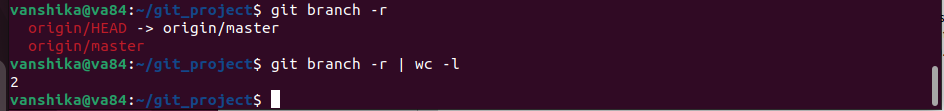
The number of objects, blobs, trees, commits and tags is lesser and different from the actual git clone.

In this clone this is only the recent commit that has been considered.

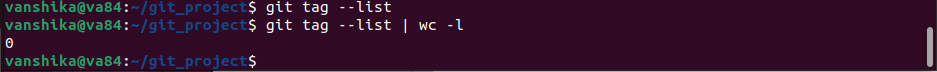


git branch -r

git branch -r | wc -l



git tag --list



Explain when specifying this option for a clone could be useful:

This option specifies that only the latest commit and its associated files should be cloned, creating a shallow clone with a history of depth 1.

GIT top bottom:

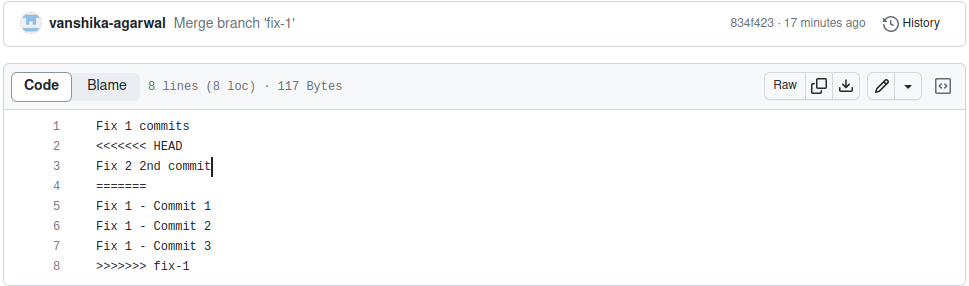
My github repository is: https://github.com/vanshika-agarwal/git\_project.git

Set up SSH connection by generating an SSH key using an

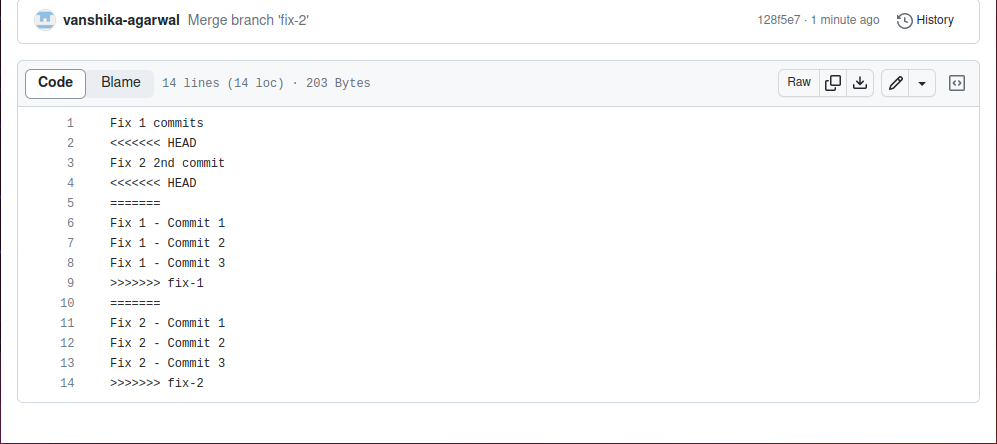
* cd ~/.ssh
* ssh -keygen -o -t rsa -C EMAIL\_ID
* cat id\_rsa.pub
* They key obtained from here is entered in the github SSH and GPG keys.

Working on the cloning the repository and working on the branches.

* Create a repository in github.
* Clone it using SSH: git clone URL
* To make changes in the main branch
* git checkout
* touch Check.txt
* git add Check.txt
* git commit -m “Adding another line”
* git push
* Creating a branch called fix 1
* git checkout -b fix-1
* echo “fix-1 commits” >> example.txt
* git add example.txt
* git commit -m “Fix 1 1st commit”
* git checkout main
* git merge fix-1
* git push
* Creating a branch called fix 2
* git checkout -b fix-2
* echo “Fix 2 conflicts” >> Check.txt
* git add Check.txt
* git commit -m “Fix 2 2nd commit”
* git checkout main
* git merge fix-2
* git push –all origin
* Trying to create merge conflict in fix 1
* git checkout fix-1
* echo "Fix 1 - Commit 1" >> example.txt
* git add example.txt
* git commit -m "Fix 1 - Commit 1"
* echo "Fix 1 - Commit 2" >> example.txt
* git add example.txt
* git commit -m "Fix 1 - Commit 2"
* echo "Fix 1 - Commit 3" >> example.txt
* git add example.txt
* git commit -m "Fix 1 - Commit 3"
* Trying to create merge conflict in fix 2
* git checkout fix-2
* echo "Fix 2 - Commit 1" >> example.txt
* git add example.txt
* git commit -m "Fix 1 - Commit 1"
* echo "Fix 2 - Commit 2" >> example.txt
* git add example.txt
* git commit -m "Fix 1 - Commit 2"
* echo "Fix 2 - Commit 3" >> example.txt
* git add example.txt
* git commit -m "Fix 1 - Commit 3"
* To merge the branches
* git merge fix-1 (which is displaying merge conflict)
* git add example.txt
* git merge –continue



* git merge fix-2 (which is displaying merge conflict)
* git add example.txt
* git merge --continue



git branch -a (Check all the branches and which branch is being worked on)

To add a tag with a message for each of the merge commits (version-1, version-2):

git tag -a -m “Merge fix-1 into main” version-1

git tag -a -m “Merge fix-2 into main with conflicts” version-2

Comments:

In Top-down it was expected to create merge conflict for fix-2 but I created a merge conflict for both fix-1 and fix-2 to understand the concept of merge.

As Fix 1 commit and Fix 2 2nd commit was entered in the example.txt file of main earlier and that was also present in the Fix-1 and Fix-2 1st commits so that remained in the head during the merge, the conflict was resolved and the remaining part of the text file of branches was added in the main file.