PROJECT MANAGEMENT WITH GIT

1.Setting Up and Basic Commands :

Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.

Step 1:

- mkdir Git -> Creates a folder/directory in the present working directory.
- cd Git ->Changes the directory to the git folder which was created.

Step 2:

- git init ->Initializes a empty git repository.
- We can see the .git folder created in the git folder, in some cases the file is hidden and to see that hidden file we need to click on view the hidden files.

Step 3:

- Vim file.txt->Creates a empty filr of txt extension in the current directory.
- Writing or adding content to the file.
- Git add ./ git add <file name> ->Stage the file with add . or add <file name>. add .will stage the whole files in the current directory and are ready to commit
- Commit the file with appropriate commit message.

Step 4:

- git log ->Displays all the history of commits with commit messages along with the author name and email.
- Every commits has a unique commit ID.

Step 5:

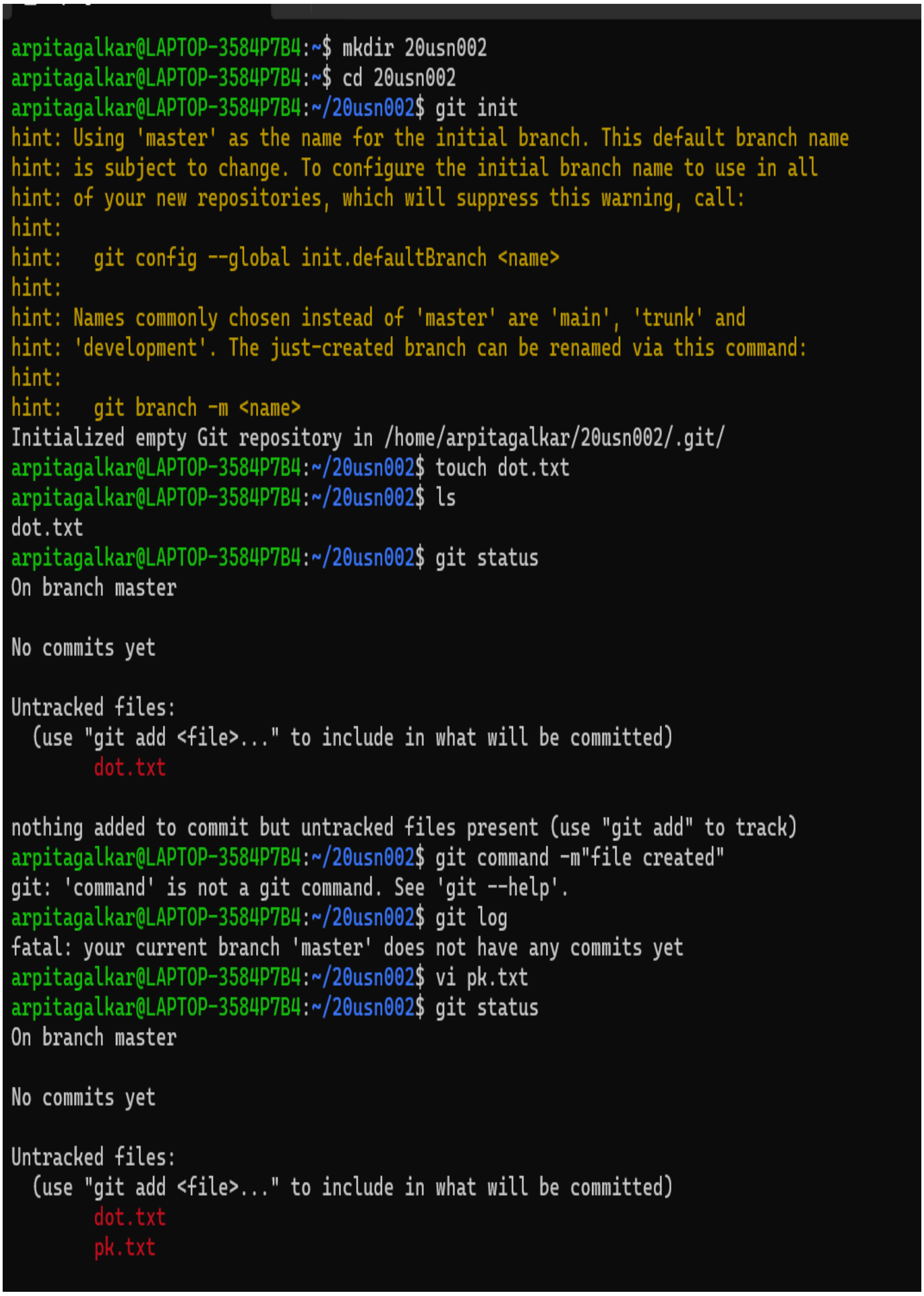
- If we want to add anything else to exitsting file,then we open it add or write the contents.
- Then again stage the file and commit it with appropriate commit message.

Step 6:

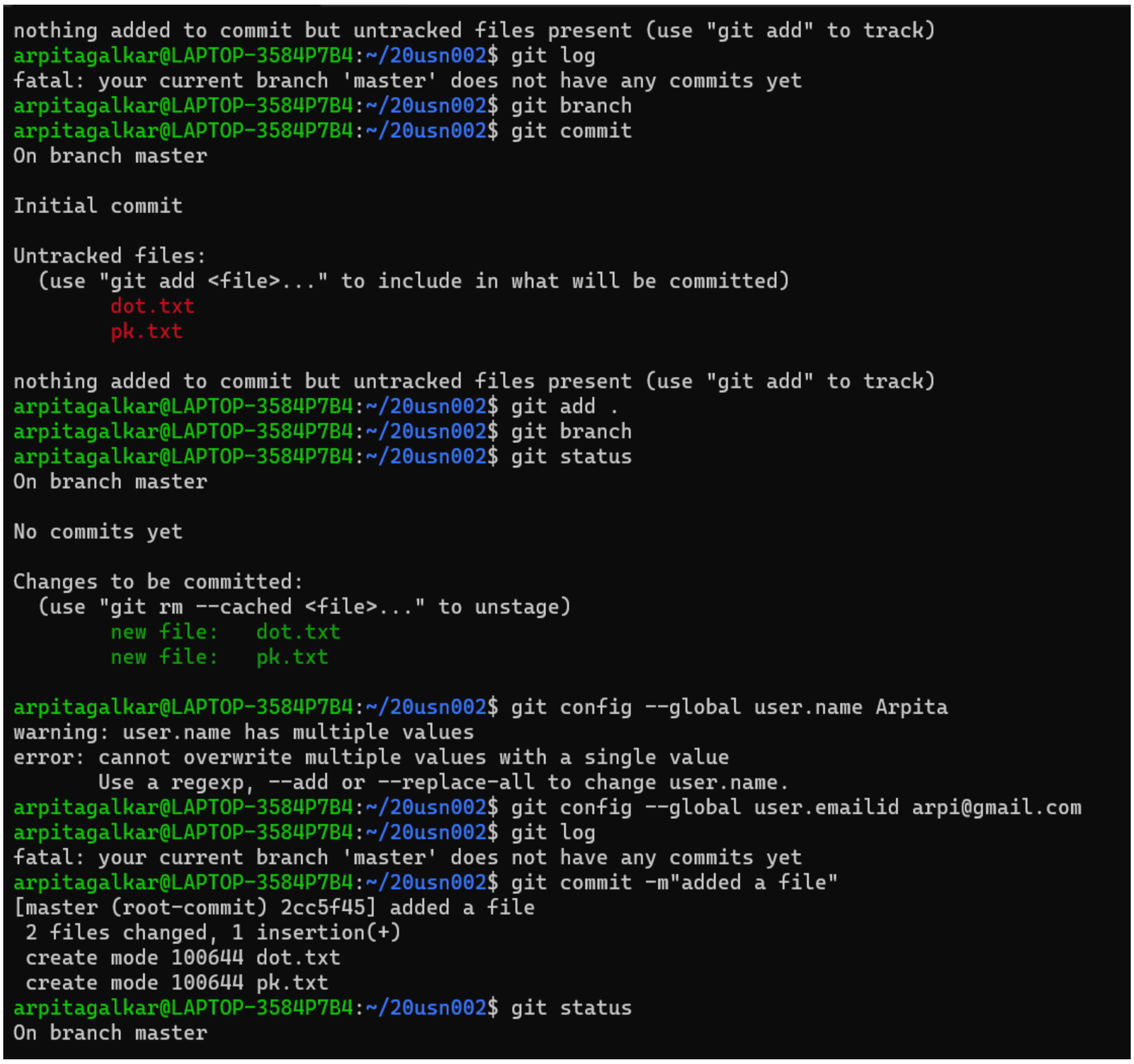
- git status ->It checks the status ,like on which branch we are and is there any changes made which are not committed.
- If no any other changes has been made after recent commit, it displays the working tree is clean.
-

Step 7:

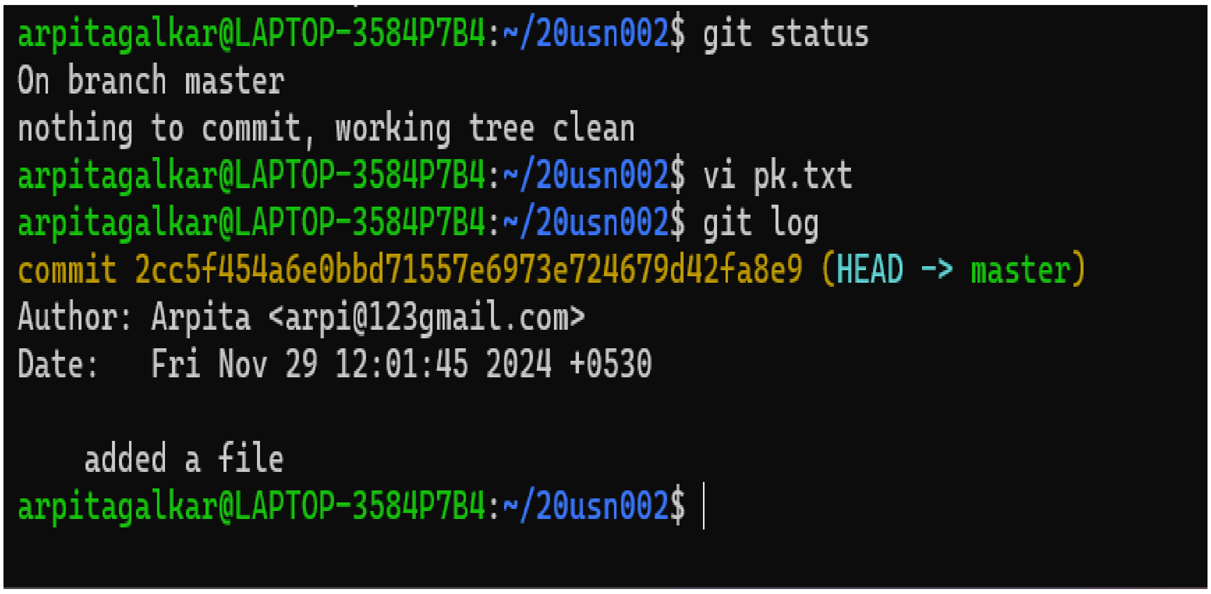
- git log->This will display the history of the commit.



```
arpitagalkar@LAPTOP-3584P7B4:~$ mkdir 20usn002
arpitagalkar@LAPTOP-3584P7B4:~$ cd 20usn002
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ 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/arpitagalkar/20usn002/.git/
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ touch dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ ls
dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git command -m"file created"
git: 'command' is not a git command. See 'git --help'.
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git log
fatal: your current branch 'master' does not have any commits yet
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ vi pk.txt
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dot.txt
        pk.txt
```



```
nothing added to commit but untracked files present (use "git add" to track)
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git log
fatal: your current branch 'master' does not have any commits yet
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git branch
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git commit
On branch master

Initial commit

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

nothing added to commit but untracked files present (use "git add" to track)
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git add .
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git branch
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   dot.txt
        new file:   pk.txt

arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git config --global user.name Arpita
warning: user.name has multiple values
error: cannot overwrite multiple values with a single value
       Use a regexp, --add or --replace-all to change user.name.
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git config --global user.emailid arpi@gmail.com
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git log
fatal: your current branch 'master' does not have any commits yet
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git commit -m"added a file"
[master (root-commit) 2cc5f45] added a file
 2 files changed, 1 insertion(+)
 create mode 100644 dot.txt
 create mode 100644 pk.txt
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git status
On branch master
```



```
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git status
On branch master
nothing to commit, working tree clean
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ vi pk.txt
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$ git log
commit 2cc5f454a6e0bbd71557e6973e724679d42fa8e9 (HEAD -> master)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 12:01:45 2024 +0530

    added a file
arpitagalkar@LAPTOP-3584P7B4:~/20usn002$
```

2.Creating and Managing Branches

- Creating a new branch named “feature-branch.” Switch to the “master “ branch. Merge the “feature -branch” into “master.

Step 1:

- git branch feature-branch ->This will create a new branch of the master branch in which the contents and files are copied from the master branch.
- git branch->This command will show all the branches which we have made and the current branch will be in green colour.

Step 2:

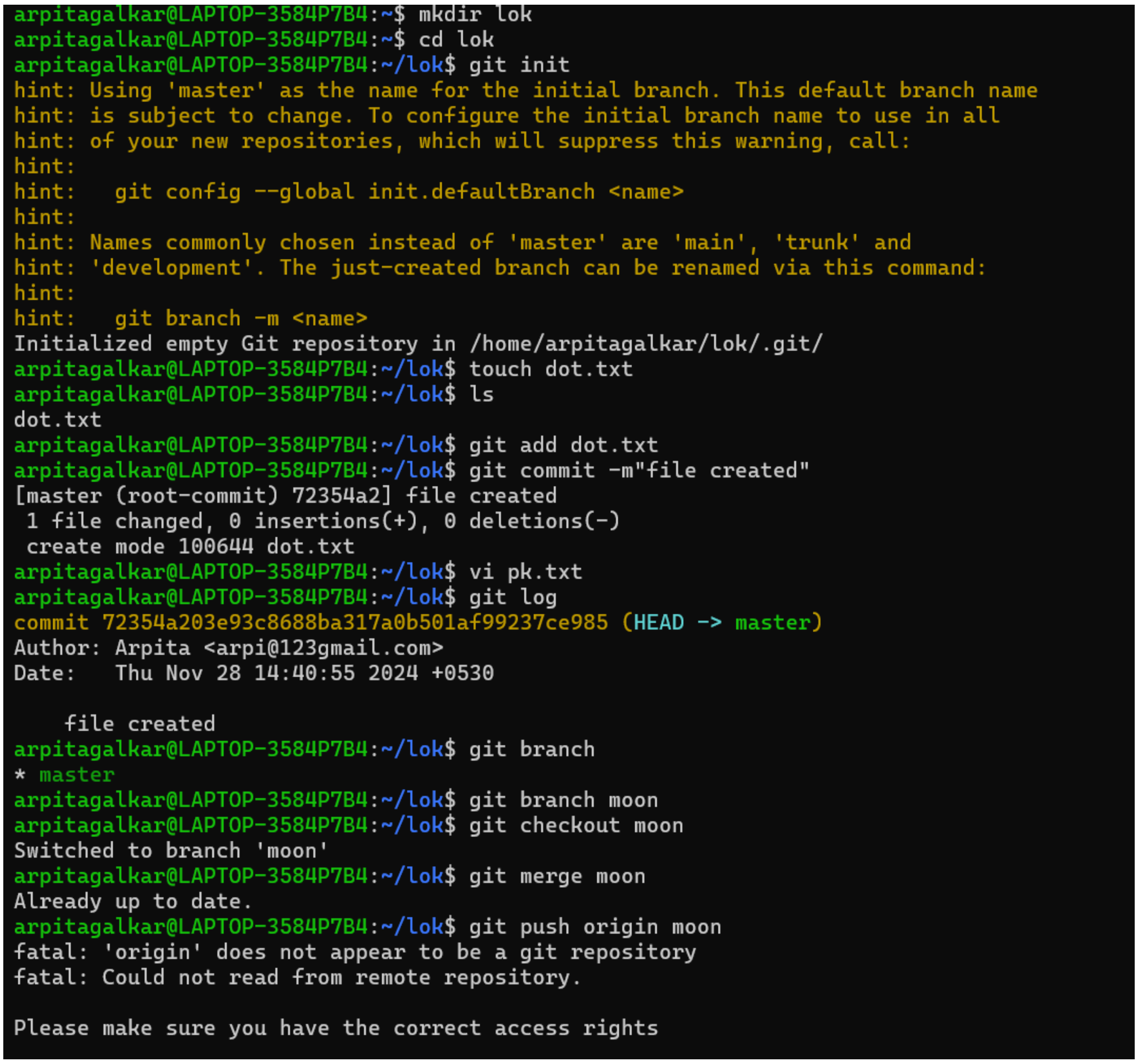
- git checkout <branch name>-> It is used to switch one branch to another branch, Here we are moving from branch master to the feature-branch.
- And also we can edited the file(file.txt).

Step 3:

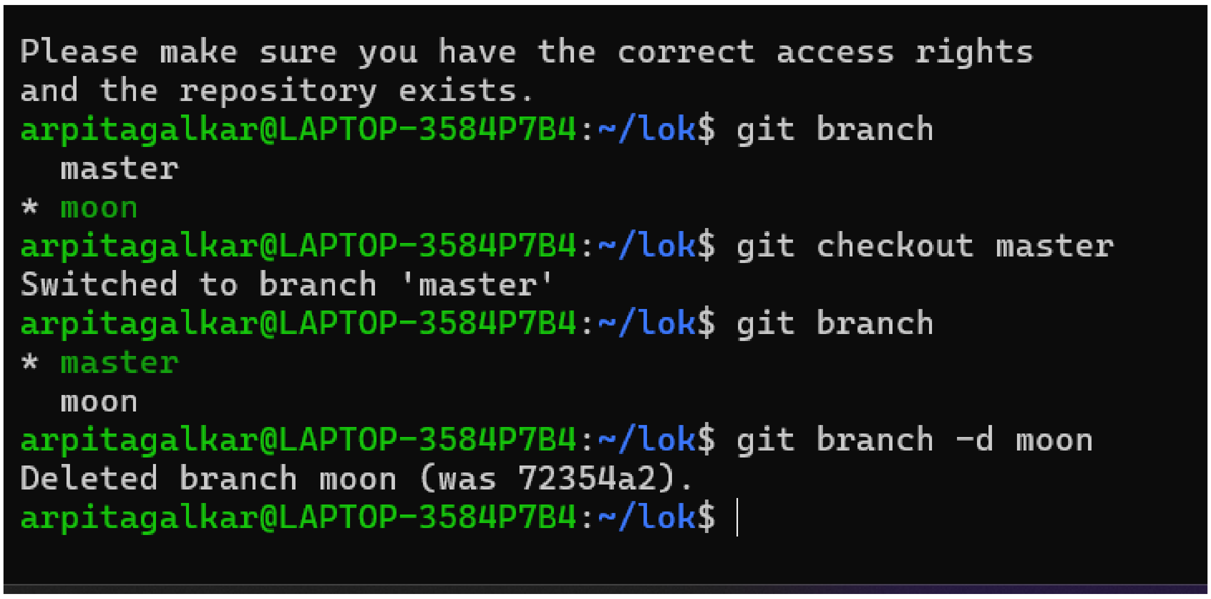
- Here in the feature-branch we staged the file and commit with appropriate message that “1st line of FB(v1)”.
- So here the file has been changed, but in the master branch it will be as it is until we merge the feature-branch with the master branch.

Step 4:

- For merging the file to the master branch we first need to move to the main /master branch using “git checkout master” command.
- Then we can merge the branch with “git merge feature-branch” command.
- So, now the files will be merged ,the changes or the edits in the feature-branch will be merged.



```
arpitagalkar@LAPTOP-3584P7B4:~$ mkdir lok
arpitagalkar@LAPTOP-3584P7B4:~$ cd lok
arpitagalkar@LAPTOP-3584P7B4:~/lok$ 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/arpitagalkar/lok/.git/
arpitagalkar@LAPTOP-3584P7B4:~/lok$ touch dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ ls
dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git add dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git commit -m"file created"
[master (root-commit) 72354a2] file created
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ vi pk.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git log
commit 72354a203e93c8688ba317a0b501af99237ce985 (HEAD -> master)
Author: Arpita <arpi@123gmail.com>
Date:   Thu Nov 28 14:40:55 2024 +0530

    file created
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git branch
* master
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git branch moon
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git checkout moon
Switched to branch 'moon'
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git merge moon
Already up to date.
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git push origin moon
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
```



```
Please make sure you have the correct access rights
and the repository exists.
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git branch
  master
* moon
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git checkout master
Switched to branch 'master'
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git branch
* master
  moon
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git branch -d moon
Deleted branch moon (was 72354a2).
arpitagalkar@LAPTOP-3584P7B4:~/lok$
```

3.Creating and Managing Branches

- ❖ Write the commands to stash your changes, switch branches , and then apply the stashed changes.

Step 1:

- ❖ Moving to the feature branch and have made the changes in the file(file .txt) using the commands git checkout feature branch and vi file.txt for the changing the branch and editing the file respectively.
- ❖ Here we have not staged and committed the changes in the file(file.txt).

Step 2:

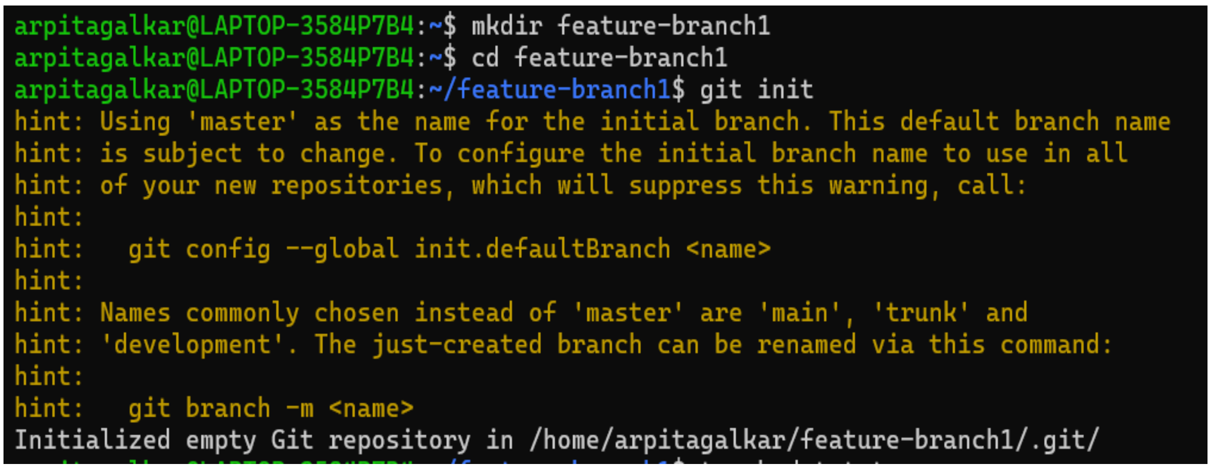
- ❖ In this step we have stashed the changes which have been made in the file.txt file in the feature branch .
- ❖ Here we have not added / staged the file and committed the changes .
- ❖ The changes will be saved in the branch without the committing the changes.
- ❖ The command used for stashing the changes is “git stash”.

Step 3:

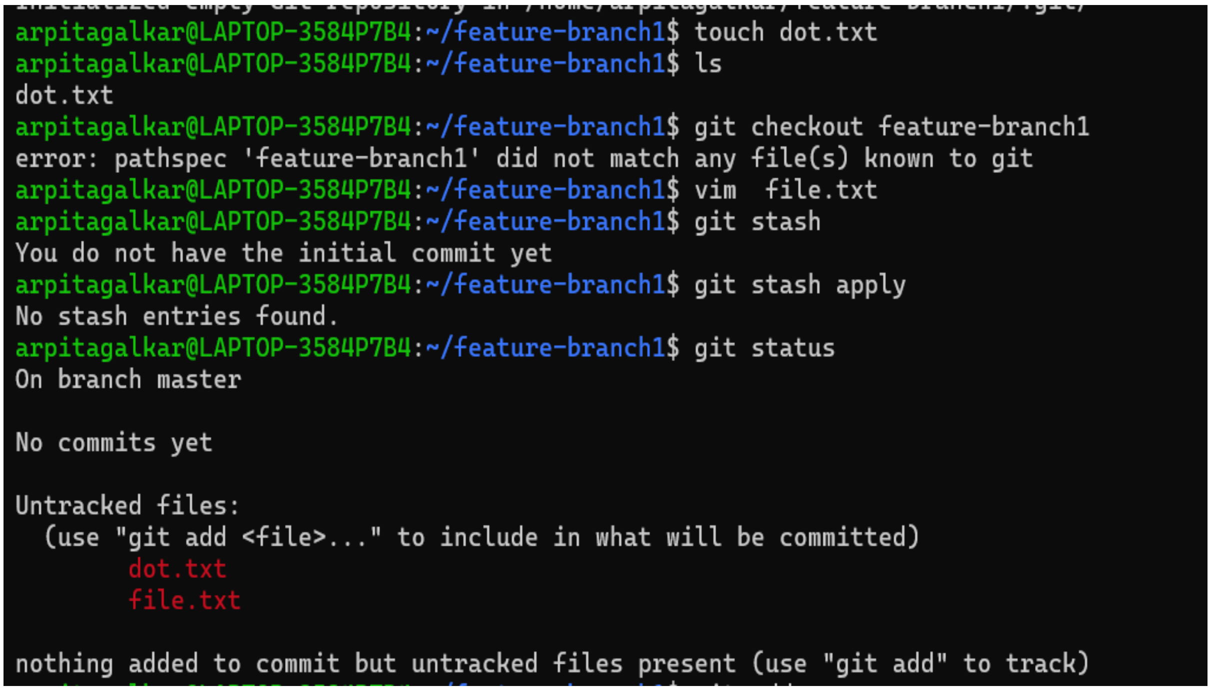
- ❖ Before applying the changes made in the feature branch, first we need move to the master branch.
- ❖ Then in the master branch we applied the stash using “git stash apply” where changes made in the feature branch.
- ❖ After applying the stash to the master .It will give us a message saying that the applied stash is not staged and committed in the master branch.

Step 4:

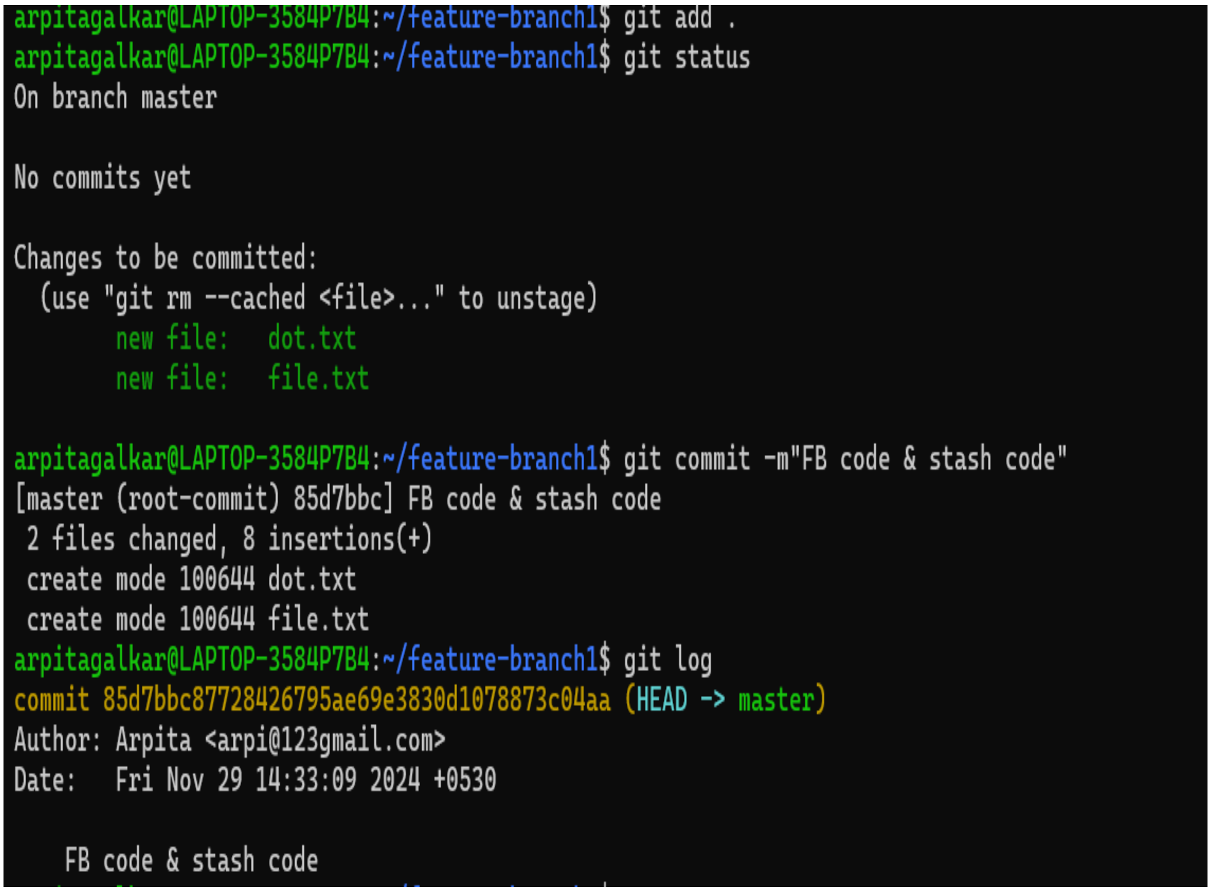
- ❖ After stashing we check status using “git status” ,there it shows that the file(file.txt) is modified but not staged yet.
- ❖ After staging ,if we check the status again it shows that changes to be committed yet.
- ❖ Committed with appropriate message that “FB code & stash code”.
- ❖ After committing the changes we can see the log of the repository.



```
arpitagalkar@LAPTOP-3584P7B4:~$ mkdir feature-branch1
arpitagalkar@LAPTOP-3584P7B4:~$ cd feature-branch1
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ 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/arpitagalkar/feature-branch1/.git/
```



```
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ touch dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ ls
dot.txt
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git checkout feature-branch1
error: pathspec 'feature-branch1' did not match any file(s) known to git
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ vim  file.txt
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git stash
You do not have the initial commit yet
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git stash apply
No stash entries found.
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git status
On branch master

No commits yet

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

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



```
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git add .
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   dot.txt
        new file:   file.txt

arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git commit -m"FB code & stash code"
[master (root-commit) 85d7bbc] FB code & stash code
 2 files changed, 8 insertions(+)
 create mode 100644 dot.txt
 create mode 100644 file.txt
arpitagalkar@LAPTOP-3584P7B4:~/feature-branch1$ git log
commit 85d7bbc87728426795ae69e3830d1078873c04aa (HEAD -> master)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 14:33:09 2024 +0530

    FB code & stash code
```

4.Collaboration And Remote Repositories:

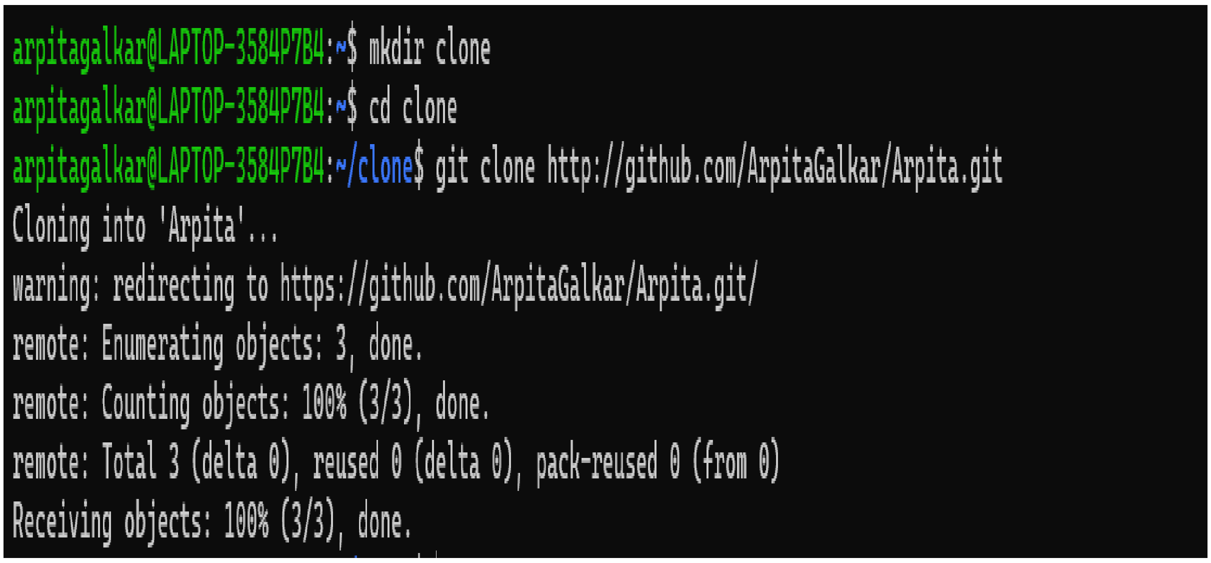
- Clone a remote Git repository to your local machine.

Step 1:

- To clone a remote git repository, first we need to open the github.com and open any one of the account on the github.com.
- After that we have choosen the repository which we want to clone into our local machine.
- After choosing the repository, in that repository we cliked on the green botton “code” , which open a drop down list of links in that ,we copied the “HTTPS” link from that ->http://github.com/ArpitaGalkar/Arpita.git

Step 2:

- In the second step we want to open our git bash and in some directory where you need to clone the remote repository we need to move to that location using “cd<path name>” command.
- Here we want to copy the repository to the folder clone in the c so we move to that location.
- git clone http://github.com/ArpitaGalkar/Arpita.git ->This command will copy the repository from remote to the local machine in the working directory.
- You can see above the “Arpita “ repository is successfully copied in the clone directory/folder.



```
arpitagalkar@LAPTOP-3584P7B4:~$ mkdir clone
arpitagalkar@LAPTOP-3584P7B4:~$ cd clone
arpitagalkar@LAPTOP-3584P7B4:~/clone$ git clone http://github.com/ArpitaGalkar/Arpita.git
Cloning into 'Arpita'...
warning: redirecting to https://github.com/ArpitaGalkar/Arpita.git/
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

5.Collaboration And Remote Repositories:

- Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.

Step 1:

- To fetch and rebase the remote repository to local repository, we will move to the already cloned repo.
- Initially before fetching the changes from the remote repo the last commit was “created onefile” after logging the commits.

Step 2:

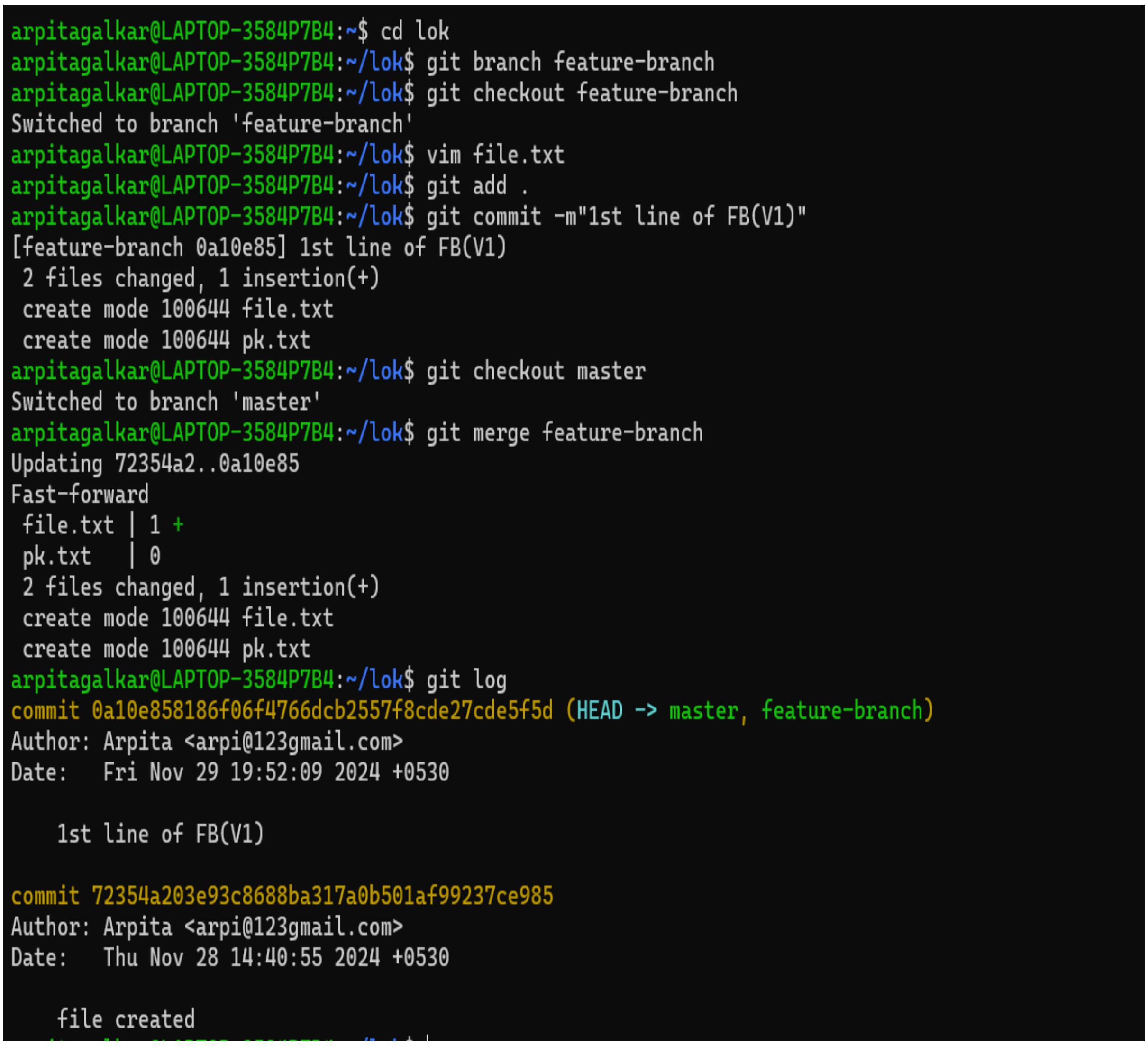
- git fetch origin/ git fetch ->This will fetch the latest changes from the remote repo that is the file named “special_note” which was created and committed.
- These changes after fetching will not be available in the working directory.

Step 3:

- git rebase origin -> This command is used to bring the changes which are fetched and present in the local repo to the working directory.
- After rebasing the remote branch to local branch, the commit which are made in the remote repo that will added to local branch.

6.Collaboration and Remote Repositories:

- Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge.
- Move to the feature branch and make some changes in the that branch ,stage and commit the changes.
- For committing we can use additional/optional -m “message” , will commit the state with appropriate message.
- Then checkout to the master branch and merge the branch.



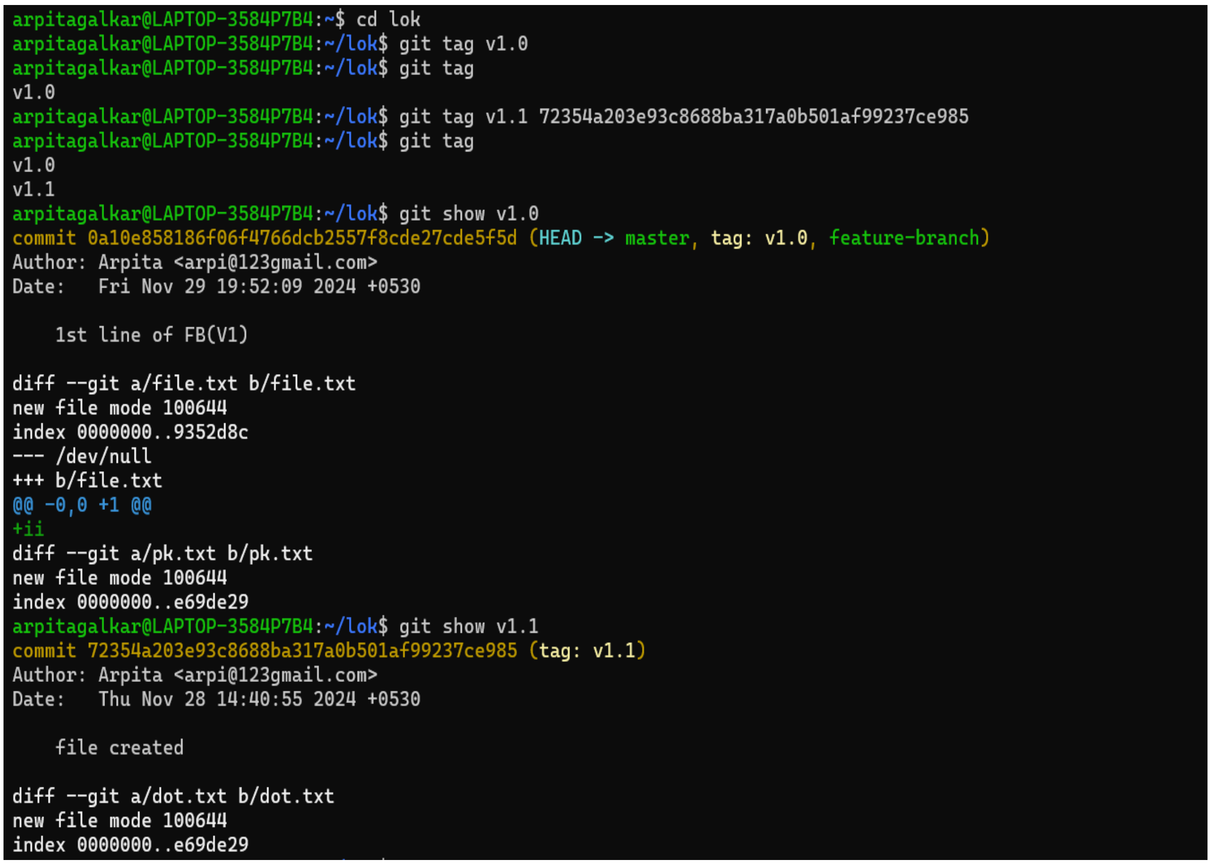
```
arpitagalkar@LAPTOP-3584P7B4:~$ cd lok
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git branch feature-branch
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git checkout feature-branch
Switched to branch 'feature-branch'
arpitagalkar@LAPTOP-3584P7B4:~/lok$ vim file.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git add .
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git commit -m"1st line of FB(V1)"
[feature-branch 0a10e85] 1st line of FB(V1)
 2 files changed, 1 insertion(+)
 create mode 100644 file.txt
 create mode 100644 pk.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git checkout master
Switched to branch 'master'
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git merge feature-branch
Updating 72354a2..0a10e85
Fast-forward
 file.txt | 1 +
 pk.txt   | 0
 2 files changed, 1 insertion(+)
 create mode 100644 file.txt
 create mode 100644 pk.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git log
commit 0a10e858186f06f4766dcb2557f8cde27cde5f5d (HEAD -> master, feature-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 19:52:09 2024 +0530

    1st line of FB(V1)

commit 72354a203e93c8688ba317a0b501af99237ce985
Author: Arpita <arpi@123gmail.com>
Date:   Thu Nov 28 14:40:55 2024 +0530

    file created
```

7. Git Tags and Releases:

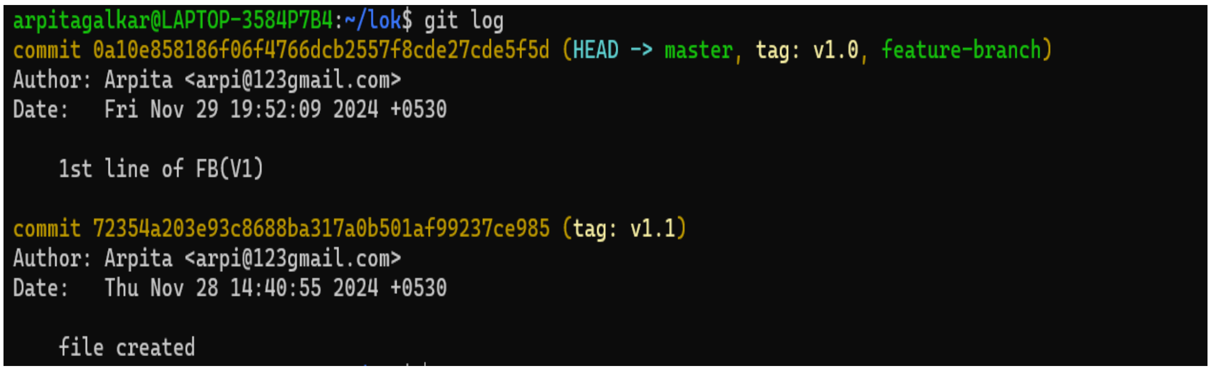
- ❖ Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.

Step1:

- ❖ git tag v1.0 ->this will create a tag of the latest commit (or we can specify the particular commit with commit ID) or we can also add a tag message using -> -m "message".
- ❖ git tag-> this command will show the all tags made i.e , v1.0 created.
- ❖ git show v1.0 -> this will show details in that tag(v1.0) with full description.



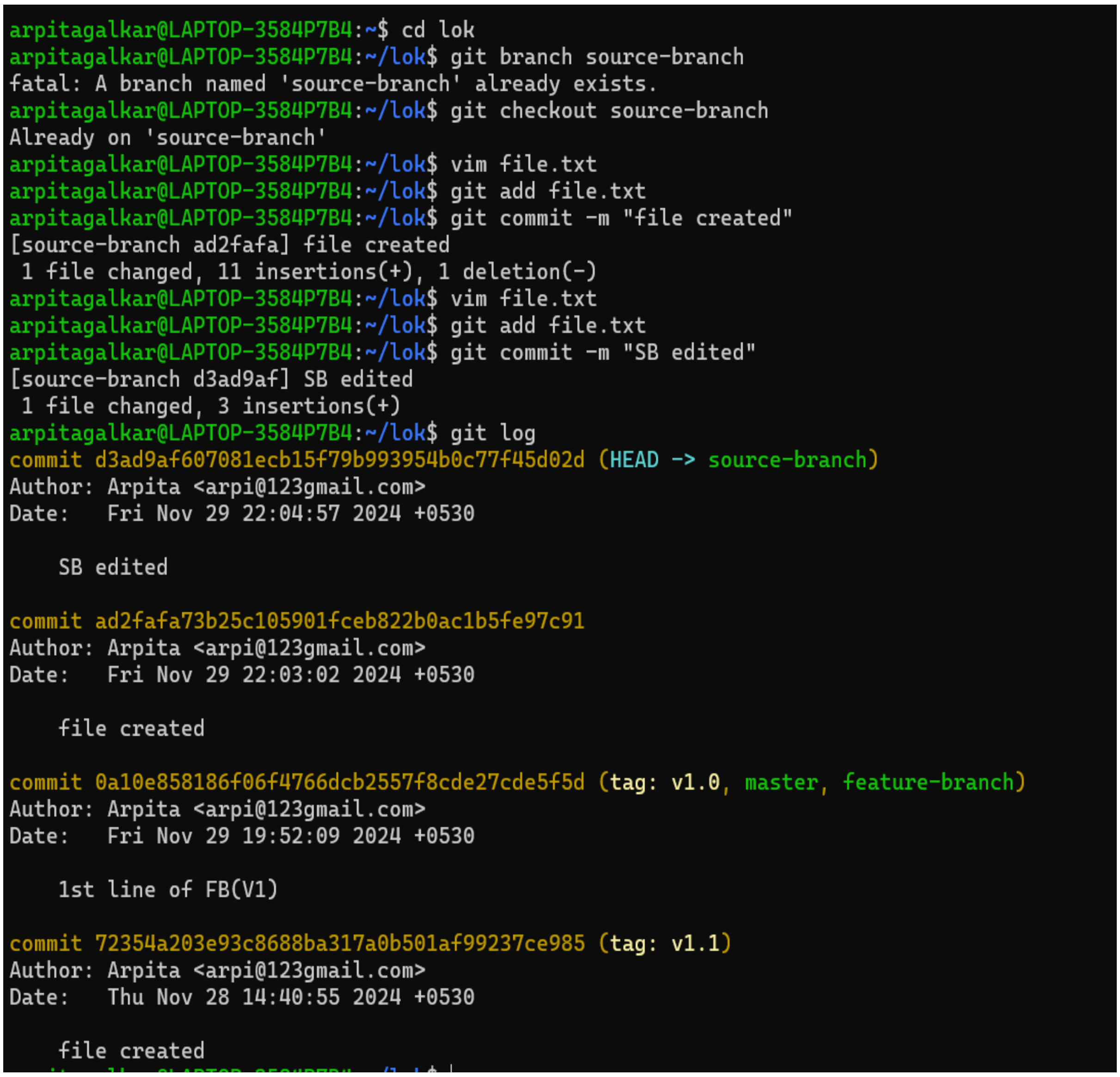
```
arpitagalkar@LAPTOP-3584P7B4:~$ cd lok
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git tag v1.0
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git tag
v1.0
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git tag v1.1 72354a203e93c8688ba317a0b501af99237ce985
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git tag
v1.0
v1.1
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git show v1.0
commit 0a10e858186f06f4766dcb2557f8cde27cde5f5d (HEAD -> master, tag: v1.0, feature-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 19:52:09 2024 +0530

    1st line of FB(V1)

diff --git a/file.txt b/file.txt
new file mode 100644
index 0000000..9352d8c
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+ii
diff --git a/pk.txt b/pk.txt
new file mode 100644
index 0000000..e69de29
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git show v1.1
commit 72354a203e93c8688ba317a0b501af99237ce985 (tag: v1.1)
Author: Arpita <arpi@123gmail.com>
Date:   Thu Nov 28 14:40:55 2024 +0530

    file created

diff --git a/dot.txt b/dot.txt
new file mode 100644
index 0000000..e69de29
```



```
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git log
commit 0a10e858186f06f4766dcb2557f8cde27cde5f5d (HEAD -> master, tag: v1.0, feature-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 19:52:09 2024 +0530

    1st line of FB(V1)

commit 72354a203e93c8688ba317a0b501af99237ce985 (tag: v1.1)
Author: Arpita <arpi@123gmail.com>
Date:   Thu Nov 28 14:40:55 2024 +0530

    file created
```

8. Advanced Git Operations:

- Write the command to cherry-pick a range of commits from "source-branch" to the current branch.
- Create a branch named source branch and check out to the source branch.
- And make the first some change in thefile.txt file.
- Stage and commit the changes with commit message saying “first commit in the source branch.
- Again make some changes in the file or add few more line in the file.txt file.
- Stage and commit the changes with message saying “SB edited”
- Here we want copy the commit ID to cherry-pick the specific state from the git log
- Now move to the master branch.
- Git cherry-pick <commit ID> -> this will take the mentioned commit ID stage and merge to the master branch.
- Main advantage of using cherry pick is we can pick the required snapshot from the branches and add to the master branch.
- Here we can see the content of the file.txt file at that snapshot is added to the master.



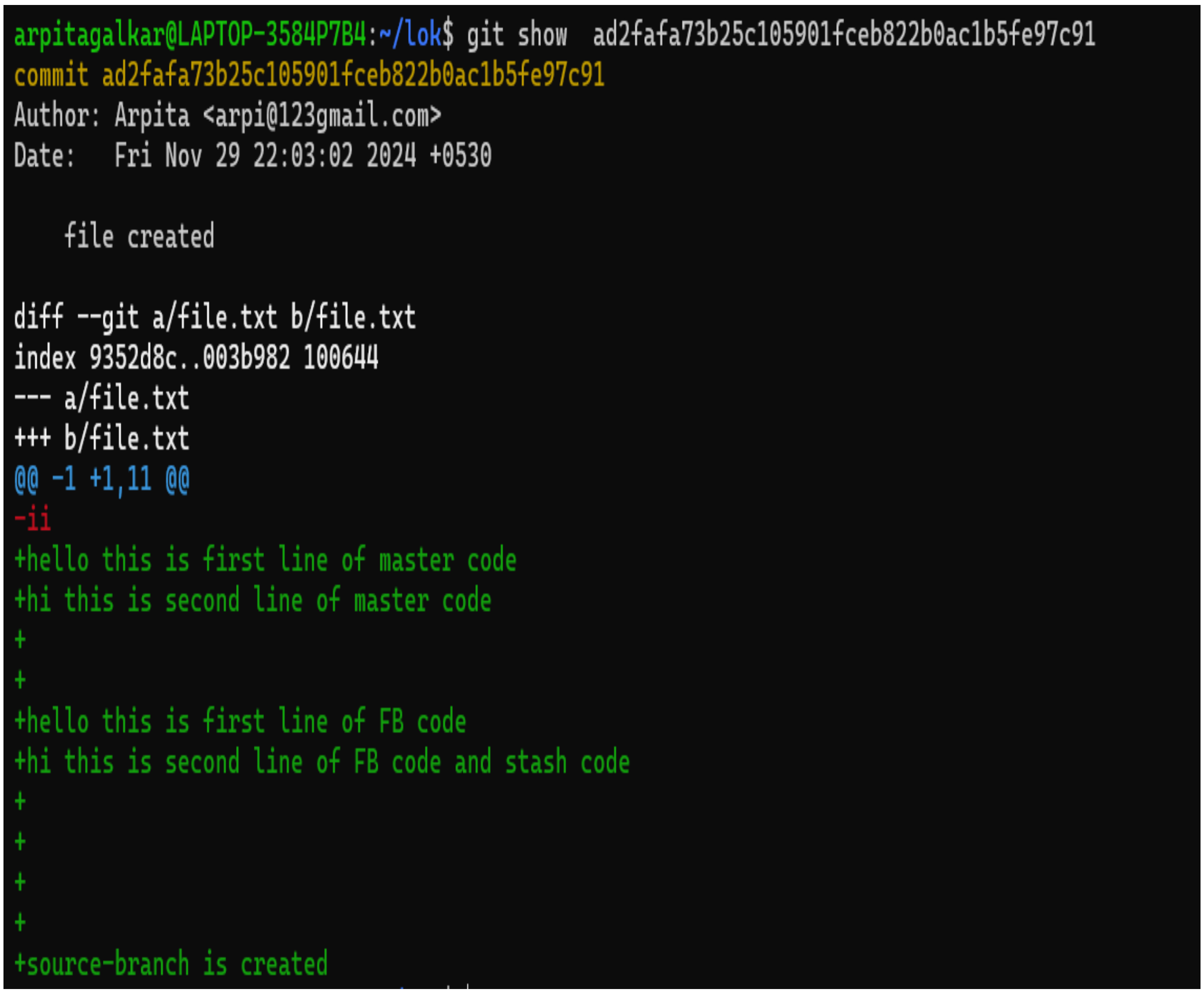
```
arpitagalkar@LAPTOP-3584P7B4:~$ cd lok
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git branch source-branch
fatal: A branch named 'source-branch' already exists.
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git checkout source-branch
Already on 'source-branch'
arpitagalkar@LAPTOP-3584P7B4:~/lok$ vim file.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git add file.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git commit -m "file created"
[source-branch ad2fafa] file created
 1 file changed, 11 insertions(+), 1 deletion(-)
arpitagalkar@LAPTOP-3584P7B4:~/lok$ vim file.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git add file.txt
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git commit -m "SB edited"
[source-branch d3ad9af] SB edited
 1 file changed, 3 insertions(+)
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git log
commit d3ad9af607081ecb15f79b993954b0c77f45d02d (HEAD -> source-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:04:57 2024 +0530

    SB edited

commit ad2fafa73b25c105901fceb822b0ac1b5fe97c91
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:03:02 2024 +0530

    file created

commit 0a10e858186f06f4766dcb2557f8cde27cde5f5d (tag: v1.0, master, feature-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 19:52:09 2024 +0530

    1st line of FB(V1)

commit 72354a203e93c8688ba317a0b501af99237ce985 (tag: v1.1)
Author: Arpita <arpi@123gmail.com>
Date:   Thu Nov 28 14:40:55 2024 +0530

    file created
```

9. Analyzing and Changing Git History:

- ❖ Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?

Step1:

- ❖ To view the details of the specific commit including author, date and commit message we should copy the specific commit which you want to view in detail.
- ❖ git show <commit ID> -> this will show the full detail of the commit ID mentioned ,added changes will be shown in green colour and deleted changes will be shown in red colour.

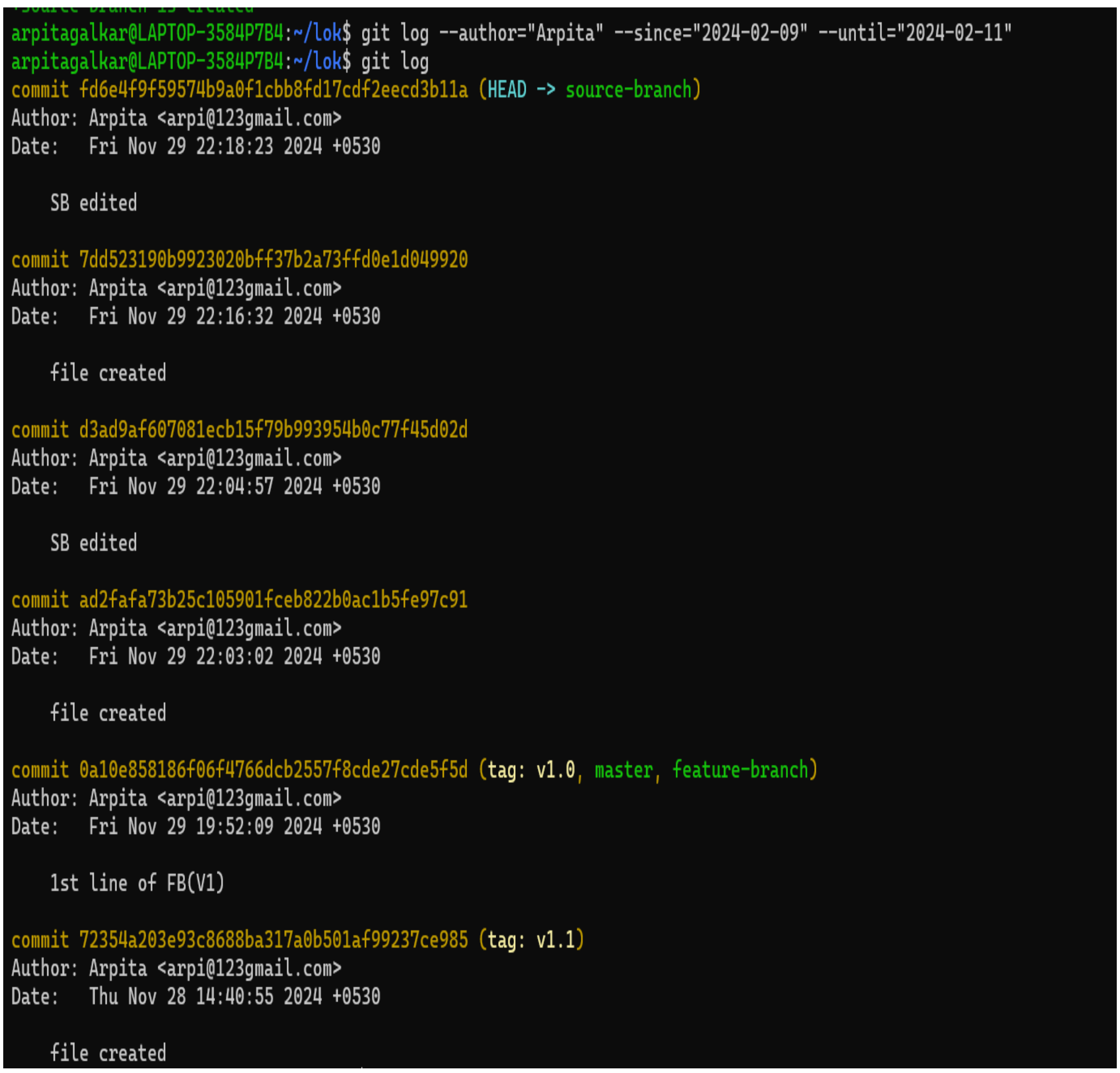


```
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git show  ad2fafa73b25c105901fceb822b0ac1b5fe97c91
commit ad2fafa73b25c105901fceb822b0ac1b5fe97c91
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:03:02 2024 +0530

    file created

diff --git a/file.txt b/file.txt
index 9352d8c..003b982 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1,11 @@
-ii
+hello this is first line of master code
+hi this is second line of master code
+
+
+hello this is first line of FB code
+hi this is second line of FB code and stash code
+
+
+
+
+source-branch is created
```

- ❖

10. Analysing and Changing Git History:

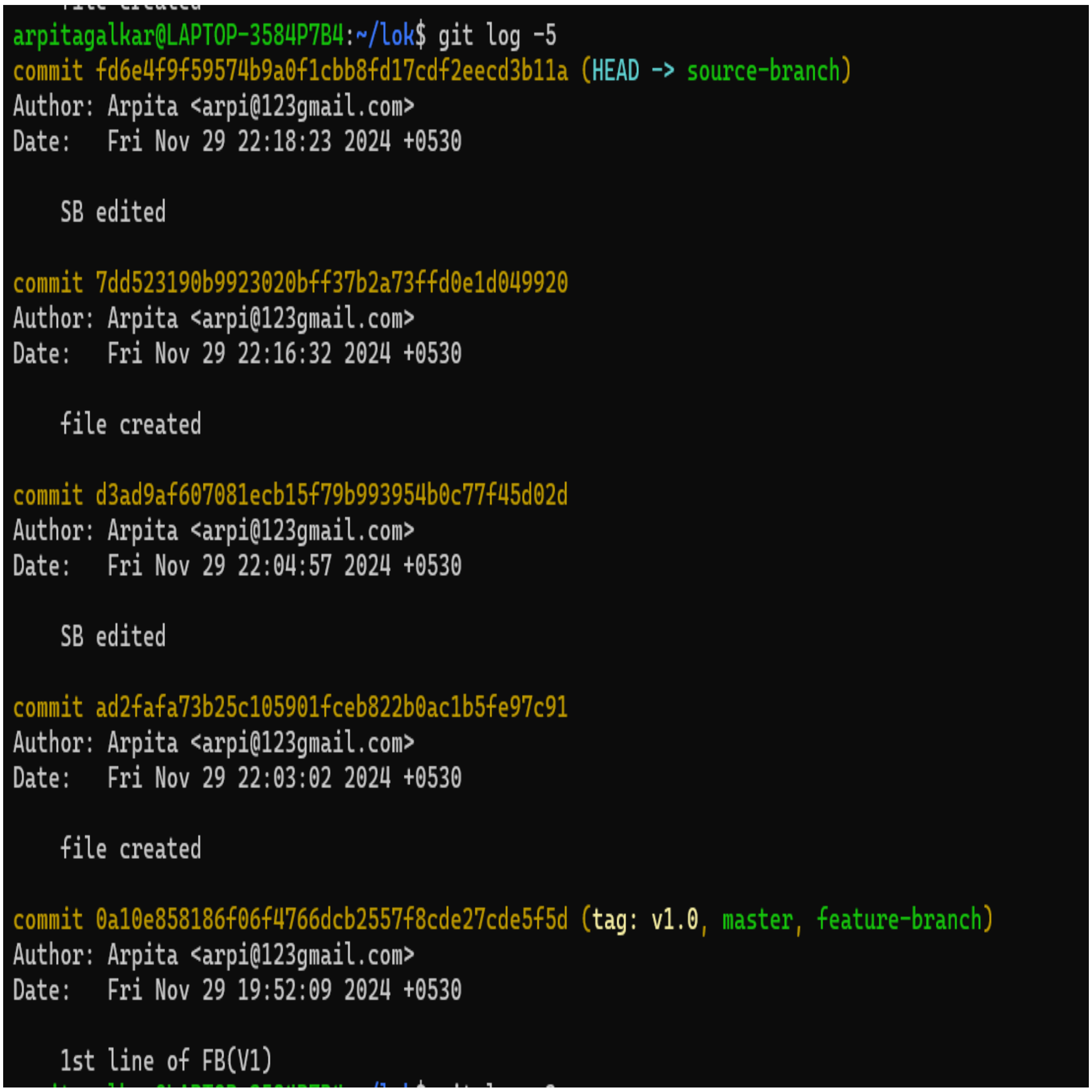
- ❖ Write the command to list all commits made by the author "John Doe" between "2024-01-27" and "2023-01-28."
- ❖ git log --author="Sakshi" --since="2024-02-09" --until="2024-02-11" ->this will show all the commits made by the author “Sakshi” b/w dated “2024-02-09" and "2024-02-11"



```
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git log --author="Arpita" --since="2024-02-09" --until="2024-02-11"
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git log
commit fd6e4f9f59574b9a0f1cbb8fd17cdf2eecd3b11a (HEAD -> source-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:18:23 2024 +0530

    SB edited

commit 7dd523190b9923020bff37b2a73ffd0e1d049920
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:16:32 2024 +0530

    file created

commit d3ad9af607081ecb15f79b993954b0c77f45d02d
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:04:57 2024 +0530

    SB edited

commit ad2fafa73b25c105901fceb822b0ac1b5fe97c91
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:03:02 2024 +0530

    file created

commit 0a10e858186f06f4766dcb2557f8cde27cde5f5d (tag: v1.0, master, feature-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 19:52:09 2024 +0530

    1st line of FB(V1)

commit 72354a203e93c8688ba317a0b501af99237ce985 (tag: v1.1)
Author: Arpita <arpi@123gmail.com>
Date:   Thu Nov 28 14:40:55 2024 +0530

    file created
```

11. Analyzing and Changing Git History:

- ❖ Write the command to display the last five commits in the repository's history.
- ❖ git log –n ->this will display last n no. of commits. Here n is 5.



```
arpitagalkar@LAPTOP-3584P7B4:~/lok$ git log -5
commit fd6e4f9f59574b9a0f1cbb8fd17cdf2eecd3b11a (HEAD -> source-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:18:23 2024 +0530

    SB edited

commit 7dd523190b9923020bff37b2a73ffd0e1d049920
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:16:32 2024 +0530

    file created

commit d3ad9af607081ecb15f79b993954b0c77f45d02d
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:04:57 2024 +0530

    SB edited

commit ad2fafa73b25c105901fceb822b0ac1b5fe97c91
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 22:03:02 2024 +0530

    file created

commit 0a10e858186f06f4766dcb2557f8cde27cde5f5d (tag: v1.0, master, feature-branch)
Author: Arpita <arpi@123gmail.com>
Date:   Fri Nov 29 19:52:09 2024 +0530

    1st line of FB(V1)
```

12. Analyzing and Changing Git History:

- Write the command to undo the changes introduced by the commit with the ID "abc123".

Step 1:

- The above image is before reverting.
- git revert <commit ID> ->this will revert to the that stage of commit.
- The above image is after reverting.