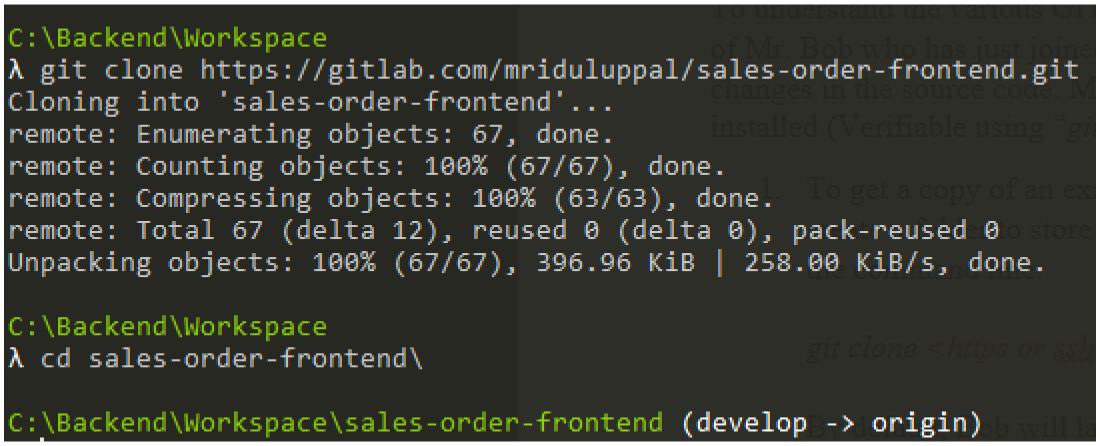
Git Commands

To understand the various GIT commands that are used daily, let us follow a practical example of Mr. Bob who has just joined a development project and wants to start contributing by making changes in the source code. Mr. Bob's routine would follow a pattern like this once GIT is installed (Verifiable using "*git --version*"):

1. To get a copy of an existing remote repository on the local machine, Bob would first create a folder to store his work on the system and then run the following command from the command line:

   git clone <https or ssh url of remote repository>

   By default, Bob will land on the protected branch of the remote repository, commonly known as the "development" branch.

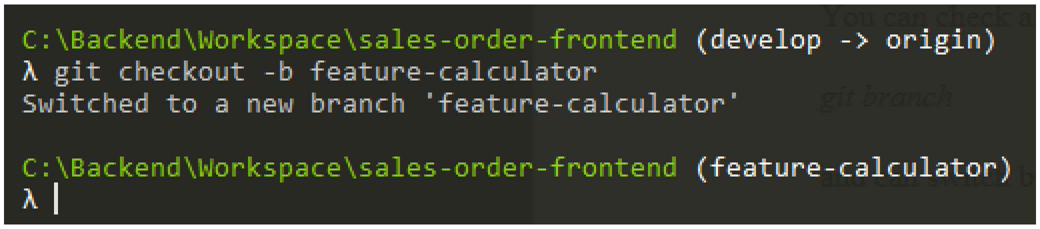


```
C:\Backend\Workspace
λ git clone https://gitlab.com/mriduluppal/sales-order-frontend.git
Cloning into 'sales-order-frontend'...
remote: Enumerating objects: 67, done.
remote: Counting objects: 100% (67/67), done.
remote: Compressing objects: 100% (63/63), done.
remote: Total 67 (delta 12), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (67/67), 396.96 KiB | 258.00 KiB/s, done.

C:\Backend\Workspace
λ cd sales-order-frontend\

C:\Backend\Workspace\sales-order-frontend (develop -> origin)
```

2. Bob wants to add a new feature, but for separating his work he needs to create a branch:

   ***git checkout** -b <branch_name>*

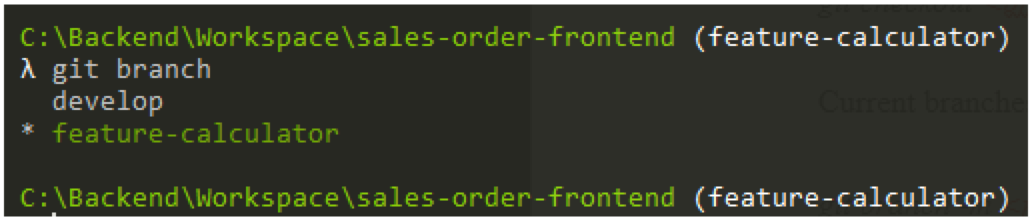
   This creates a new local copy of the "development" branch with the name provided, in which you can make your changes.



```
C:\Backend\Workspace\sales-order-frontend (develop -> origin)
λ git checkout -b feature-calculator
Switched to a new branch 'feature-calculator'

C:\Backend\Workspace\sales-order-frontend (feature-calculator)
λ |
```

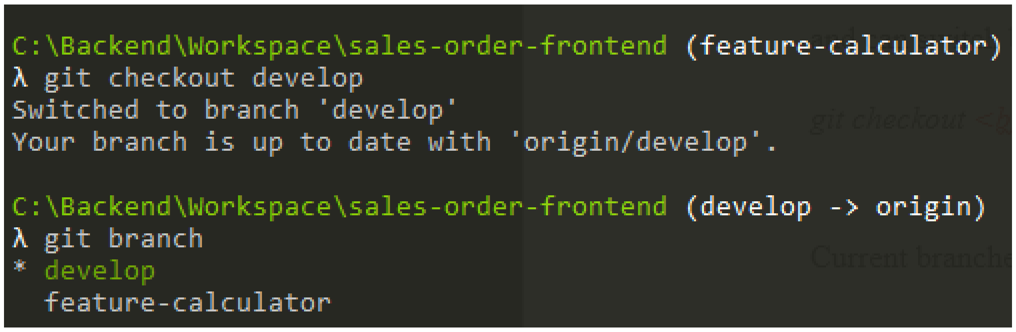
You can check all your created local branches using:

git branch



```
C:\Backend\Workspace\sales-order-frontend (feature-calculator)
λ git branch
  develop
* feature-calculator

C:\Backend\Workspace\sales-order-frontend (feature-calculator)
```

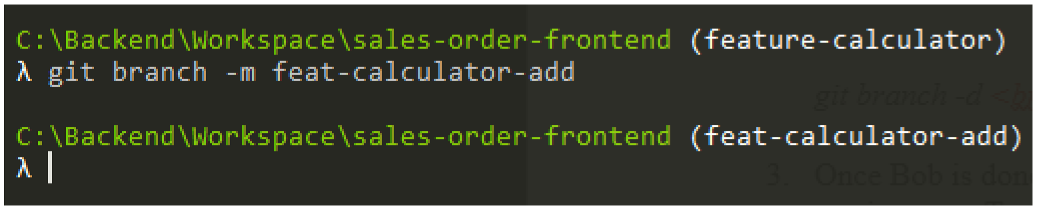
and can switch between them using: *git checkout <branch_name>*



```
C:\Backend\Workspace\sales-order-frontend (feature-calculator)
λ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

C:\Backend\Workspace\sales-order-frontend (develop -> origin)
λ git branch
* develop
  feature-calculator
```

Current branches can be renamed using:

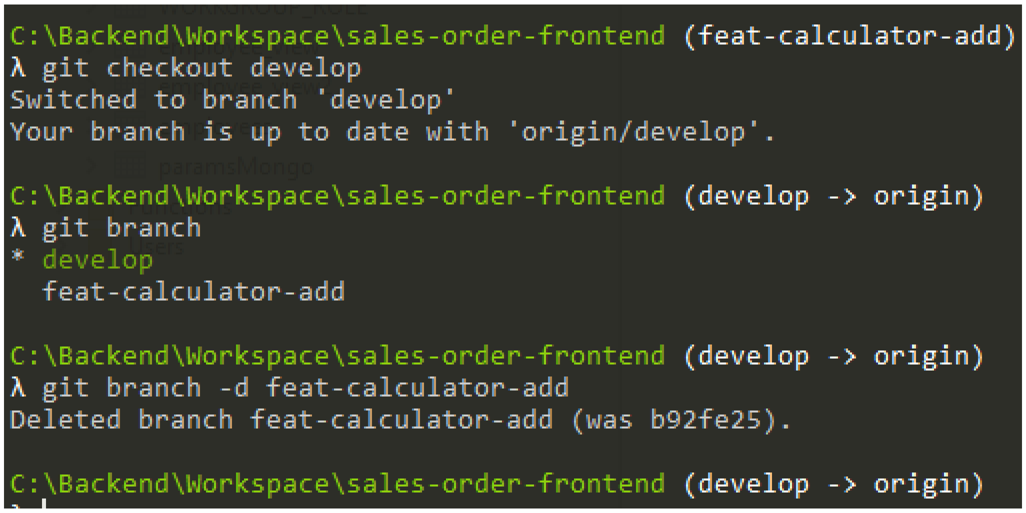
***git branch -m** <new_branch_name>*



```
C:\Backend\Workspace\sales-order-frontend (feature-calculator)
λ git branch -m feat-calculator-add

C:\Backend\Workspace\sales-order-frontend (feat-calculator-add)
λ |
```

Branches can be deleted from local repository using:

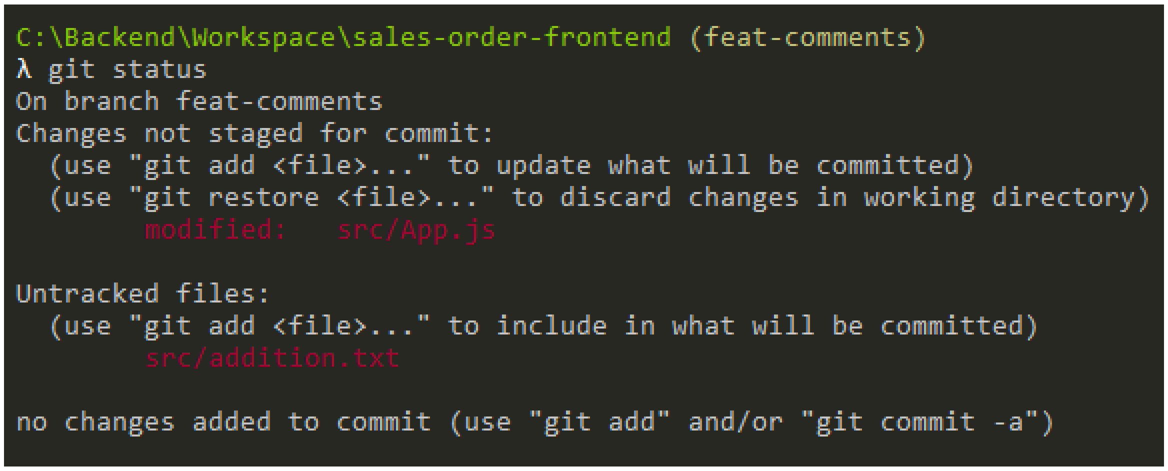
***git branch -d** <branch_name>*


```
C:\Backend\Workspace\sales-order-frontend (feat-calculator-add)
λ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

C:\Backend\Workspace\sales-order-frontend (develop -> origin)
λ git branch
* develop
  feat-calculator-add

C:\Backend\Workspace\sales-order-frontend (develop -> origin)
λ git branch -d feat-calculator-add
Deleted branch feat-calculator-add (was b92fe25).

C:\Backend\Workspace\sales-order-frontend (develop -> origin)
```

3. Once Bob is done with his changes, he first needs to add these changes to his local staging area. To review all the changes made in current branch use:

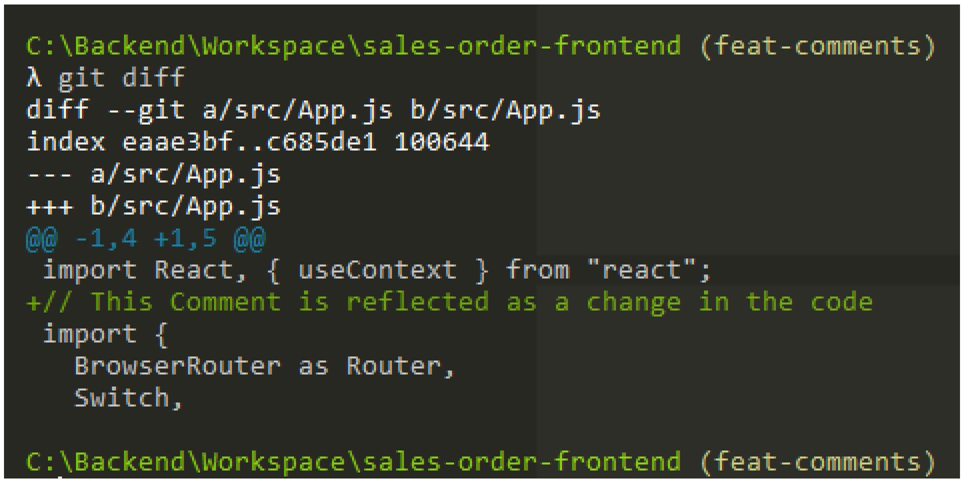
   git status

   (This will show the names of the files changed or added)


```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/App.js

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

no changes added to commit (use "git add" and/or "git commit -a")
```

   git diff

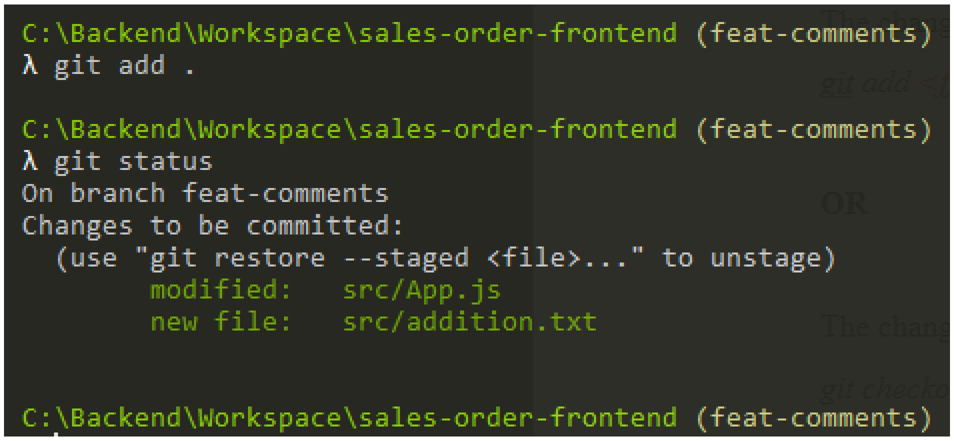
   (This will show the changed content of the files)



```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git diff
diff --git a/src/App.js b/src/App.js
index eaae3bf..c685de1 100644
--- a/src/App.js
+++ b/src/App.js
@@ -1,4 +1,5 @@
 import React, { useContext } from "react";
+// This Comment is reflected as a change in the code
 import {
   BrowserRouter as Router,
   Switch,

C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

The changes can then be added to staging area using:

git add <file_name>



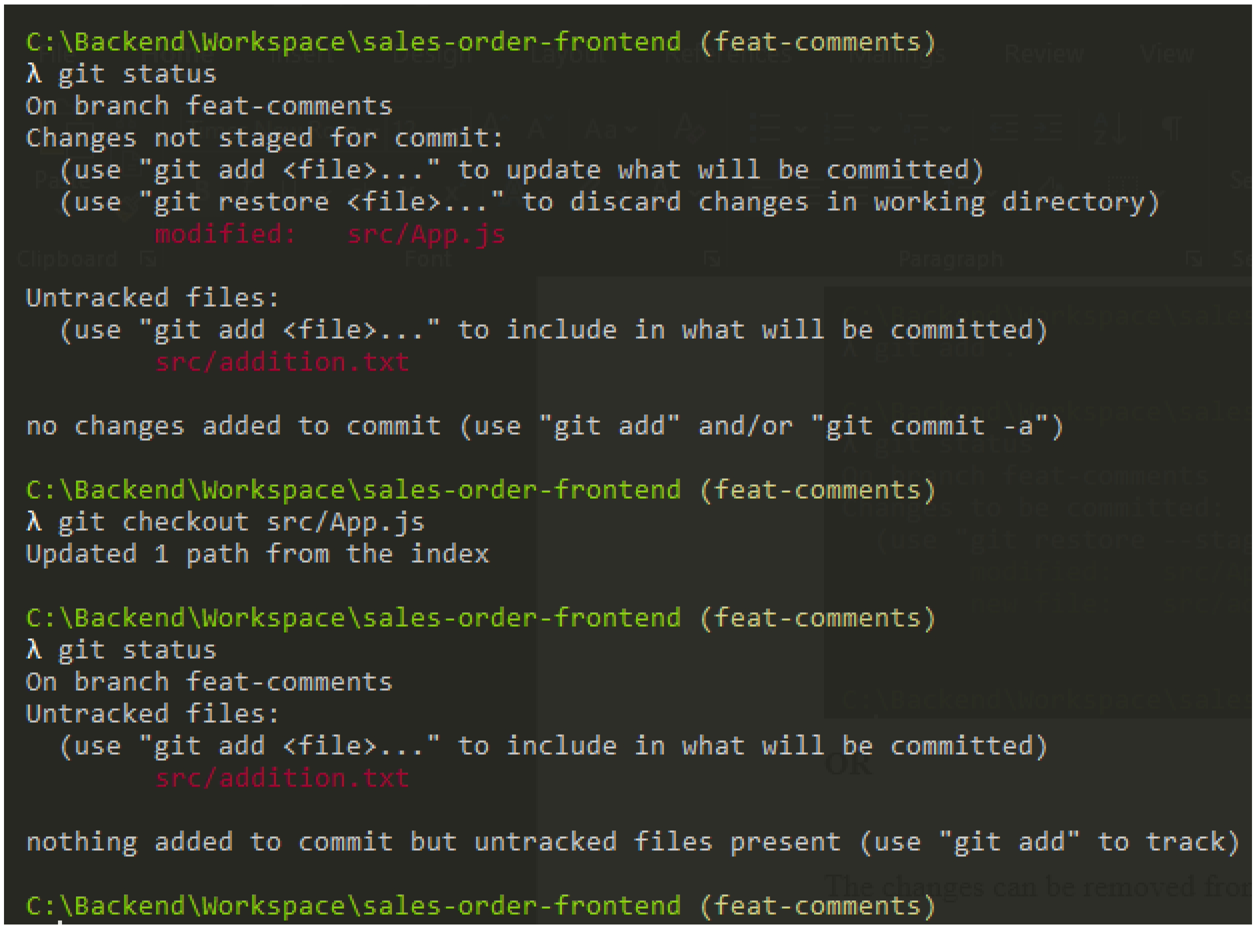
```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git add .

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/App.js
        new file:   src/addition.txt


C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

OR

The changes can be removed from the branch using:

git checkout <file_name>

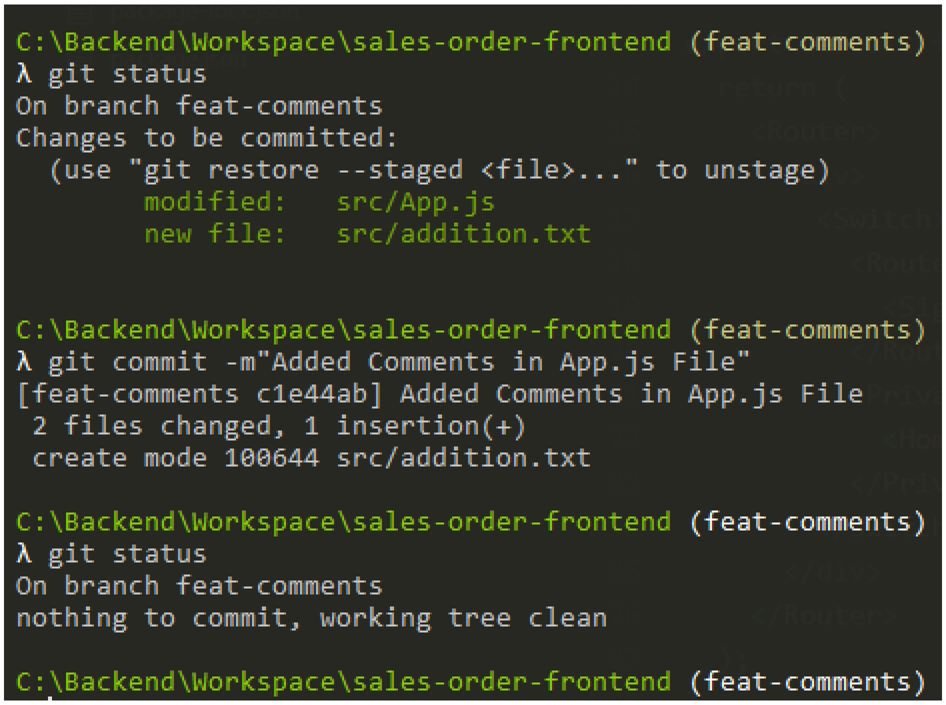
(The changes made will be lost forever!)



```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/App.js

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

no changes added to commit (use "git add" and/or "git commit -a")

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git checkout src/App.js
Updated 1 path from the index

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src/addition.txt

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

C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

4. Once the changes are reviewed by Bob in the staging area and Bob is ready to commit them to his local branch, he can use:

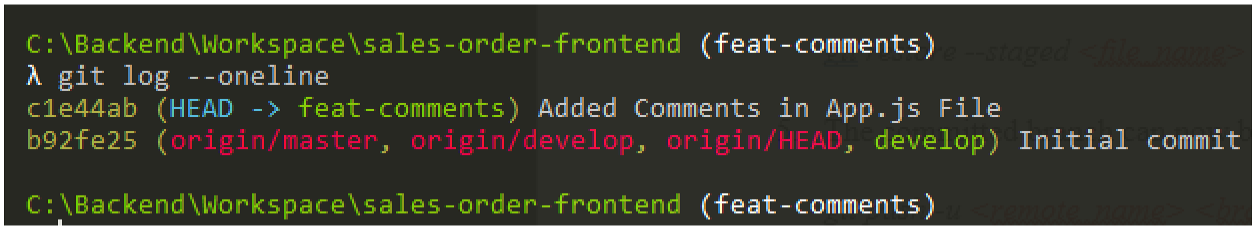
   git commit -m"<A nice message to remember what all changes were done>"



```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/App.js
        new file:   src/addition.txt


C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git commit -m"Added Comments in App.js File"
[feat-comments c1e44ab] Added Comments in App.js File
 2 files changed, 1 insertion(+)
 create mode 100644 src/addition.txt

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
nothing to commit, working tree clean

C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

The commits made on the local branch can be seen by using the command:

git log --oneline

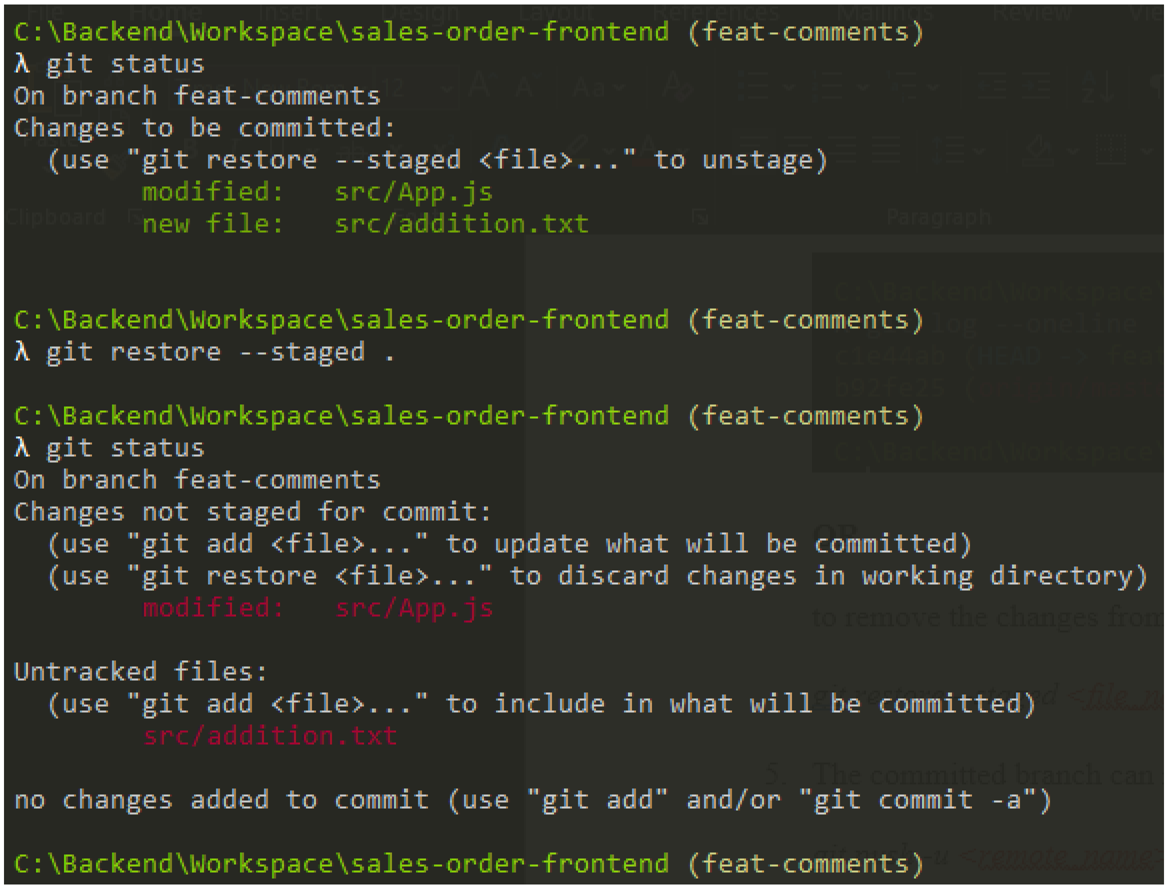


```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git log --oneline
c1e44ab (HEAD -> feat-comments) Added Comments in App.js File
b92fe25 (origin/master, origin/develop, origin/HEAD, develop) Initial commit

C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

OR

to remove the changes from the staging area and bring them back to the working area use:

git restore --staged <file_name>

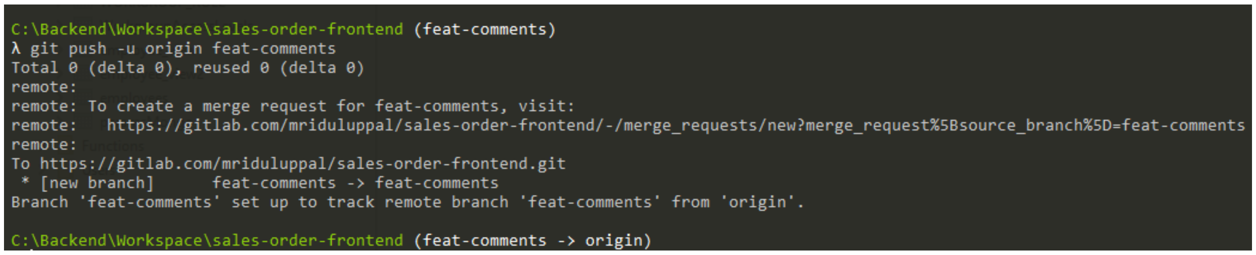


```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/App.js
        new file:   src/addition.txt


C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git restore --staged .

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/App.js

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

no changes added to commit (use "git add" and/or "git commit -a")

C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

5. The committed branch can now be pushed to the remote repository using the command:

git push -u <remote_name> <branch_name>

(The remote name most used is "*origin*", which is a name for your remote repository)



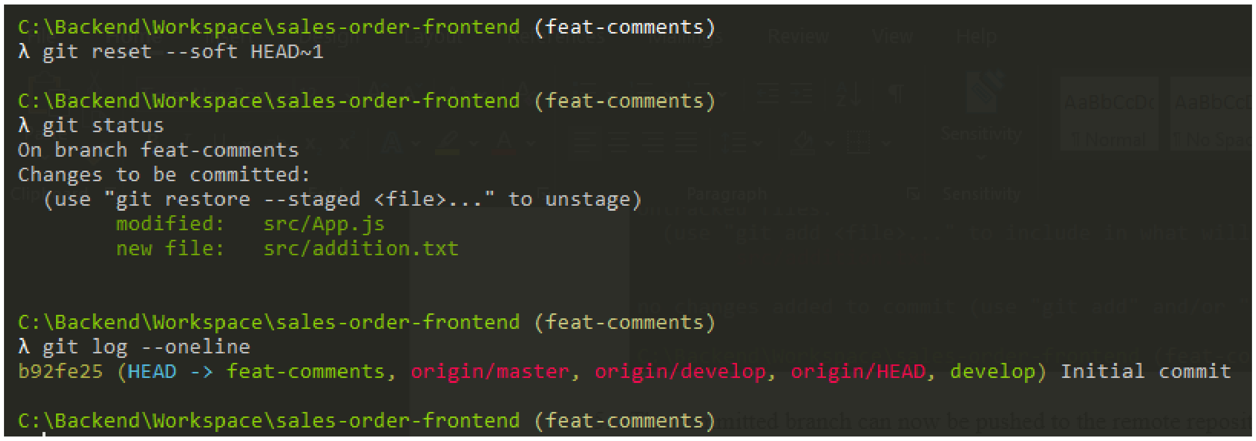
```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git push -u origin feat-comments
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for feat-comments, visit:
remote:   https://gitlab.com/mriduluppal/sales-order-frontend/-/merge_requests/new?merge_request%5Bsource_branch%5D=feat-comments
remote:
To https://gitlab.com/mriduluppal/sales-order-frontend.git
 * [new branch]      feat-comments -> feat-comments
Branch 'feat-comments' set up to track remote branch 'feat-comments' from 'origin'.

C:\Backend\Workspace\sales-order-frontend (feat-comments -> origin)
```

OR

The committed files can be reverted to their original state using the following commands:

git reset --soft HEAD~<number of commits to remove>

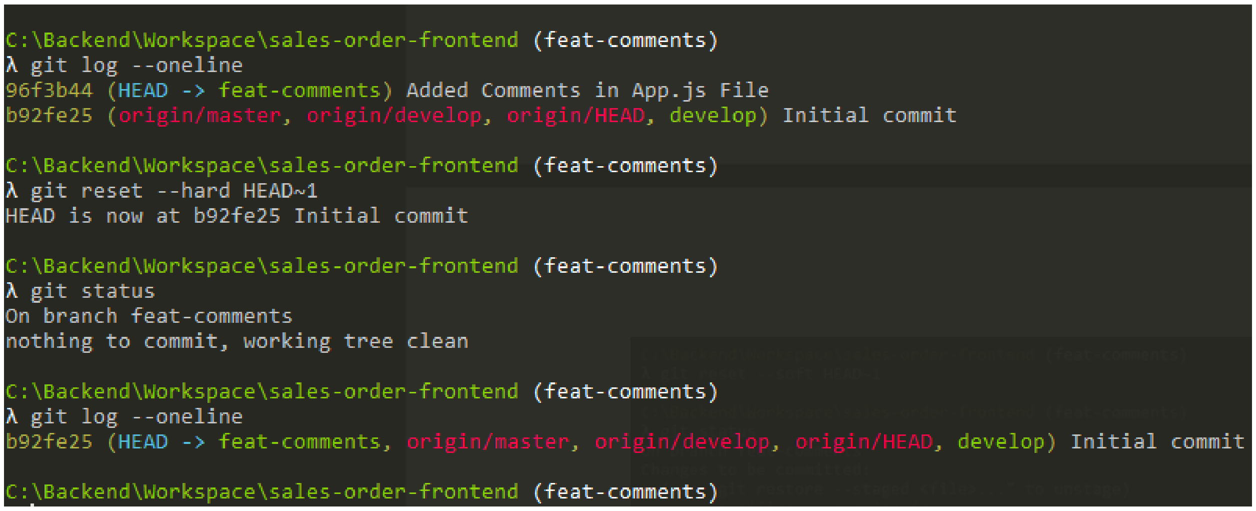
(This will bring the files back to the working area where your changes are preserved)



```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git reset --soft HEAD~1

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/App.js
        new file:   src/addition.txt


C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git log --oneline
b92fe25 (HEAD -> feat-comments, origin/master, origin/develop, origin/HEAD, develop) Initial commit

C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

git reset --hard HEAD~<number of commits to remove>

(This will remove everything from the previous commit made by Bob so be careful!)



```
C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git log --oneline
96f3b44 (HEAD -> feat-comments) Added Comments in App.js File
b92fe25 (origin/master, origin/develop, origin/HEAD, develop) Initial commit

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git reset --hard HEAD~1
HEAD is now at b92fe25 Initial commit

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git status
On branch feat-comments
nothing to commit, working tree clean

C:\Backend\Workspace\sales-order-frontend (feat-comments)
λ git log --oneline
b92fe25 (HEAD -> feat-comments, origin/master, origin/develop, origin/HEAD, develop) Initial commit

C:\Backend\Workspace\sales-order-frontend (feat-comments)
```

This step can also be performed after pushing the changes to the branch. A new push will be needed after performing this change so that it can be reflected on remote repository as well.

Now Bob is finally ready to get his branch reviewed and merged in the code base by creating a Merge Request.