GIT Stash – Create, Save, list & Show

- We can see few examples on GIT Stash commands.
- Stash command is used to temporarily stored un committed work in order to clean out your working directory without having to commit unfinished work on a branch.
- Sometimes while working on a part of our project things can be in a messy stateand we might want to switch to a different branch to work on something else.
- But before moving on to another work we do not want to commit our unfinished work.
- Basically we want to return back our unfinished work later on so the solution is this problem is git stash command.
- By executing git stash command we can save our stack of un finished work that includes modified and tracked files and staged changes after we save our dirty state of our working directory we can retrieve it at any point of time or whenever we need it. For examples please follow at below screen shots

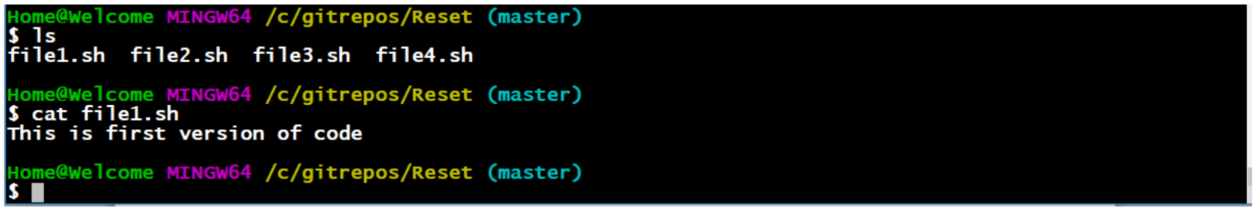
So we can just move to our working repository on our system and checking how many files are available using ls command. Basically for this stash am going to have four files in my repository.

cd stash

git init

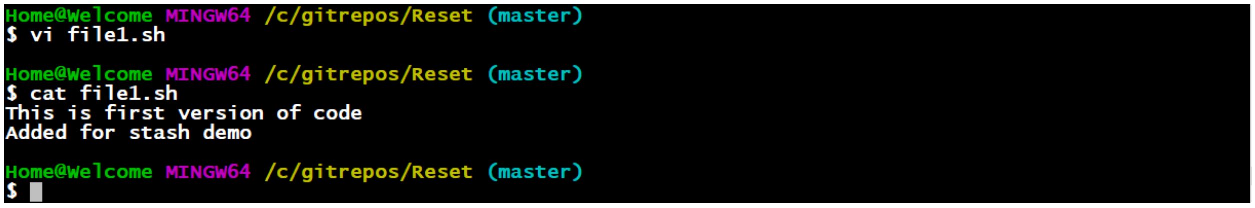
git status

ls



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ ls
file1.sh  file2.sh  file3.sh  file4.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file1.sh
This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$
```

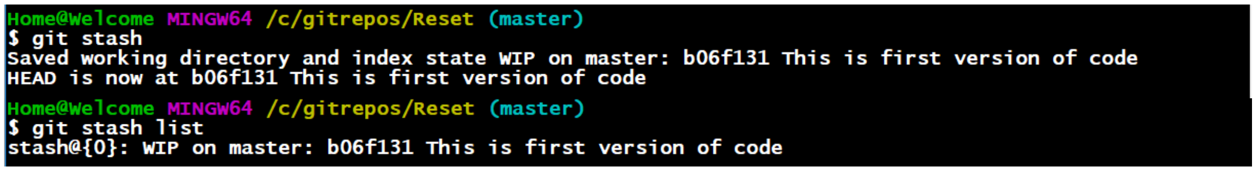
So we are going to modify file called file1.sh here like below



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file1.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file1.sh
This is first version of code
Added for stash demo

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$
```

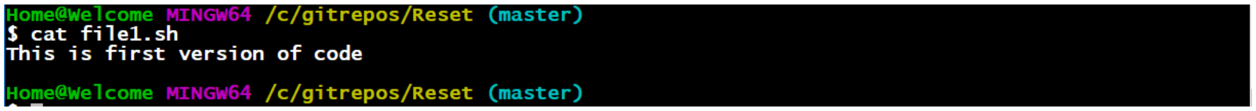
Lets have a scenario here we just modified one file called file1.sh and now there is a requirement to work on some urgeant hotfix on our project so that time we have to move to work on different things menas we have to levae this work environement and moves to some other place like new branch, now we have to save this un finished work and then we have to move another work or else this work can be removed or else need to be commited so with the help of stash we don't do that.

It will be very useful on real cases.



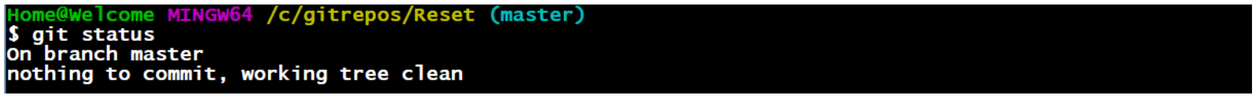
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash
Saved working directory and index state WIP on master: b06f131 This is first version of code
HEAD is now at b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: WIP on master: b06f131 This is first version of code
```

To check the existing stashes we have to use command called **git stash list**



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file1.sh
This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```

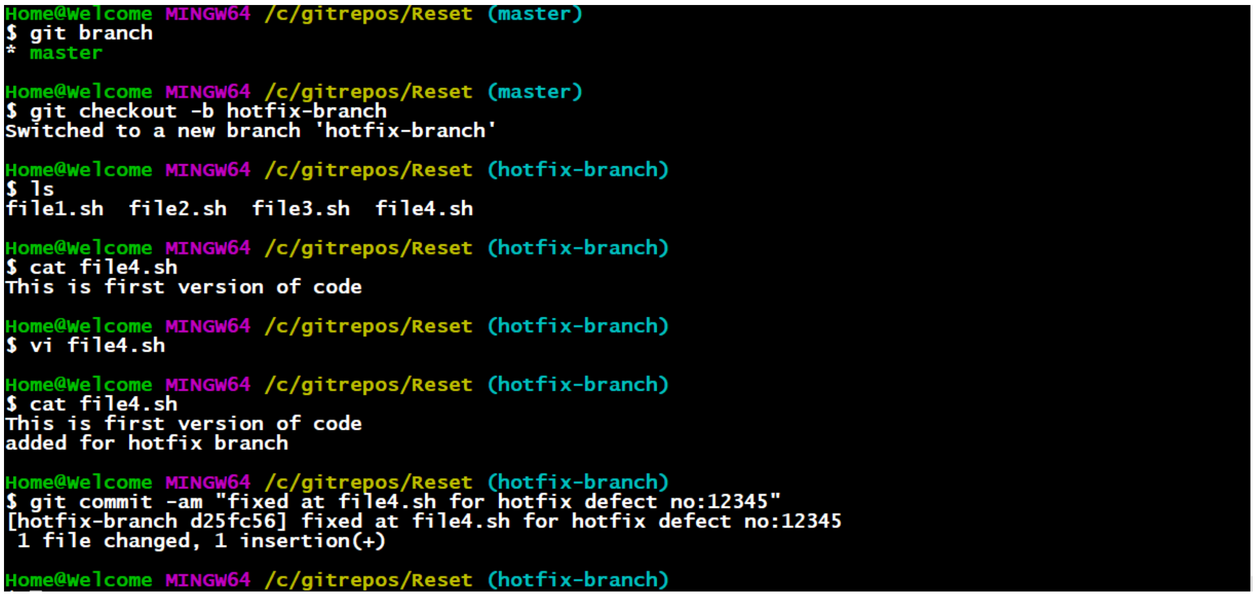
This case we don't see newly added lines in our file means we have kept a side our un finished work.



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status
On branch master
nothing to commit, working tree clean
```

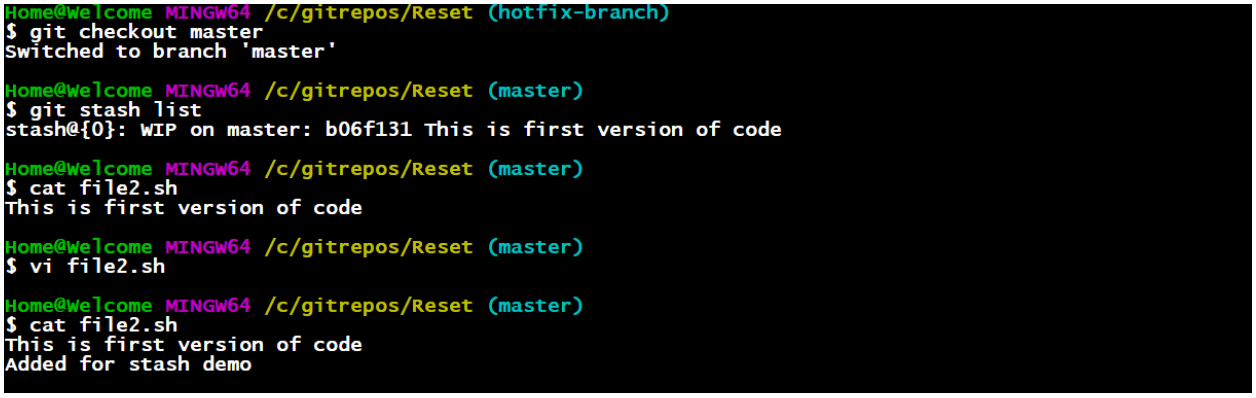
git branch

git checkout –b hotfix-branch

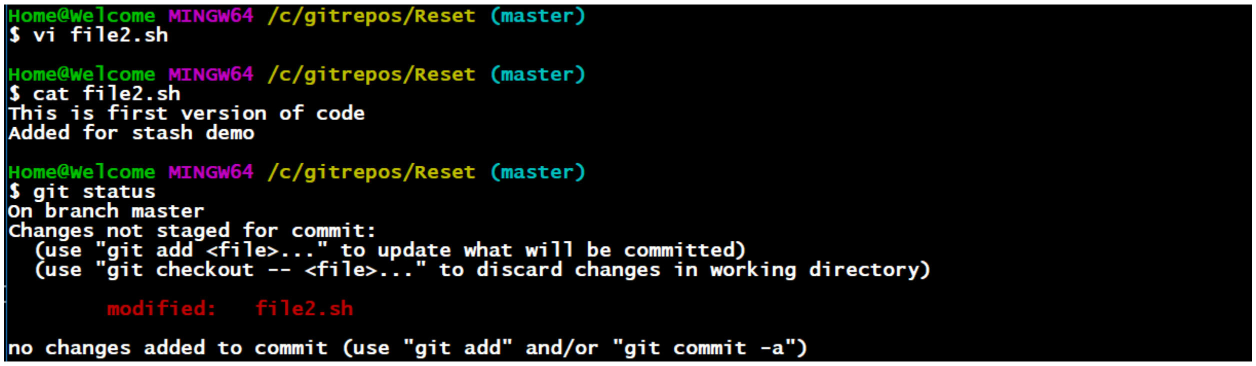


```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git branch
* master

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout -b hotfix-branch
Switched to a new branch 'hotfix-branch'

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ ls
file1.sh  file2.sh  file3.sh  file4.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ cat file4.sh
This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ vi file4.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ cat file4.sh
This is first version of code
added for hotfix branch

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git commit -am "fixed at file4.sh for hotfix defect no:12345"
[hotfix-branch d25fc56] fixed at file4.sh for hotfix defect no:12345
 1 file changed, 1 insertion(+)

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
```

So our work is done for hotfix and we want to go back on our previous work on master branch like below but we don't want to work on same thing we can work on some other file called file2.sh



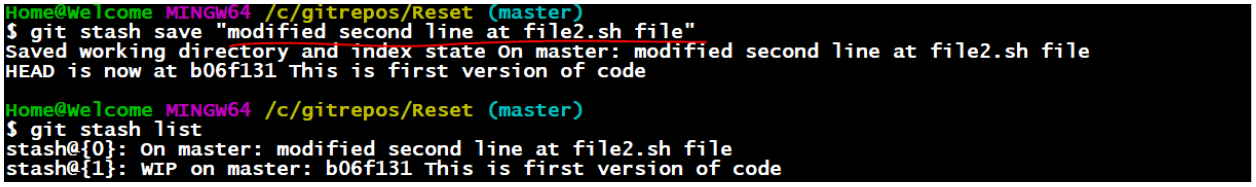
```
Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git checkout master
Switched to branch 'master'

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: WIP on master: b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file2.sh
This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file2.sh
This is first version of code
Added for stash demo
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file2.sh
This is first version of code
Added for stash demo

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file2.sh

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

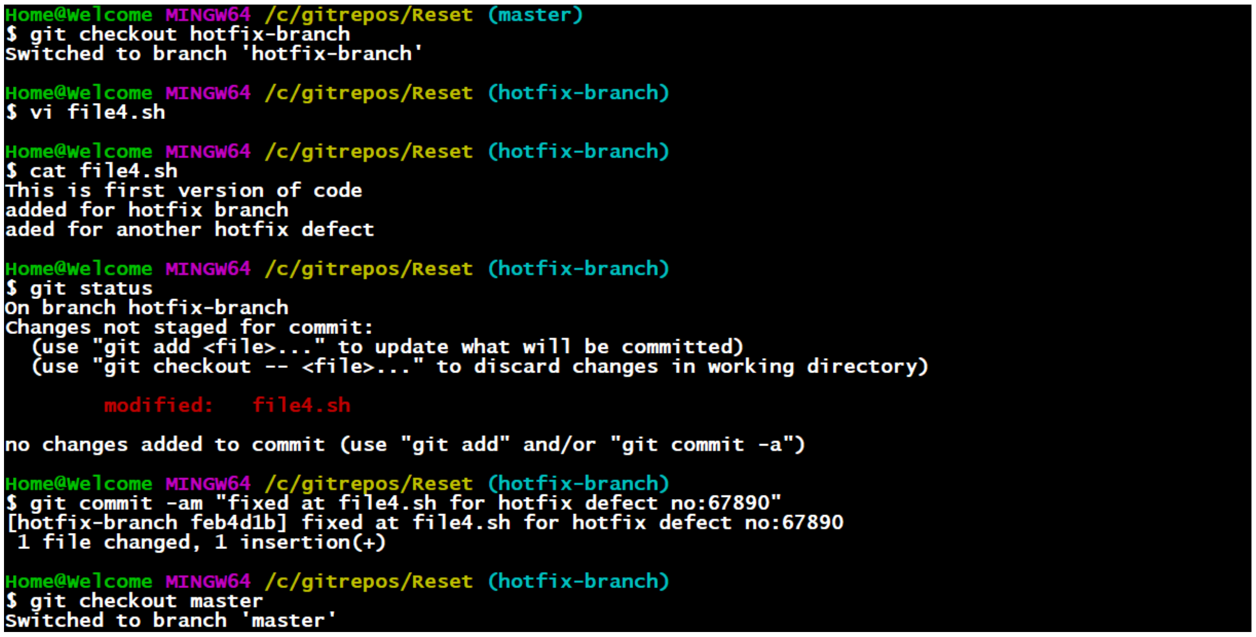
Now again we called to work on another hotfix so we have to move on another branch to work on hot fix defect so we have to save this work as well under stash like first above stash.

While doing second stash we can give message, this message wil help us to identify our stash.



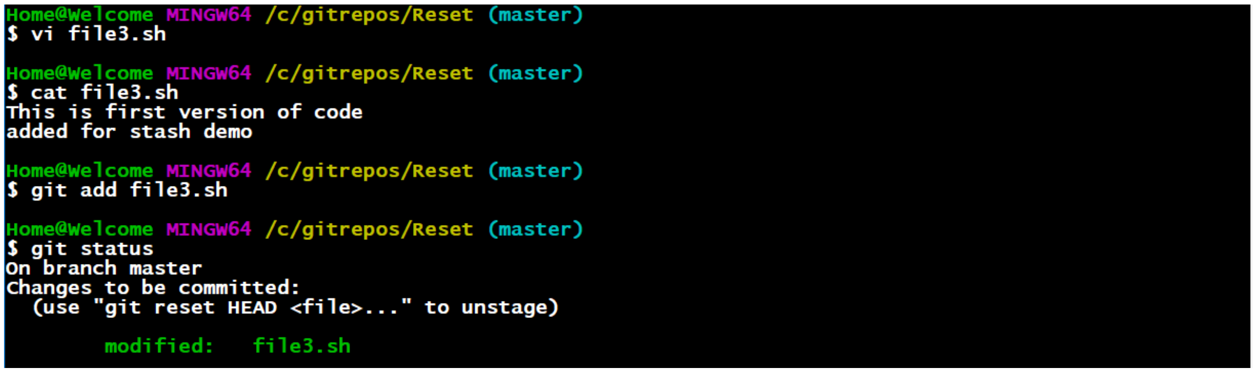
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash save "modified second line at file2.sh file"
Saved working directory and index state On master: modified second line at file2.sh file
HEAD is now at b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file2.sh file
stash@{1}: WIP on master: b06f131 This is first version of code
```

Stash always display will in reverse order. Most recent stash will display with index 0.



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout hotfix-branch
Switched to branch 'hotfix-branch'

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ vi file4.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ cat file4.sh
This is first version of code
added for hotfix branch
aded for another hotfix defect

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git status
On branch hotfix-branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file4.sh

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

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git commit -am "fixed at file4.sh for hotfix defect no:67890"
[hotfix-branch feb4d1b] fixed at file4.sh for hotfix defect no:67890
 1 file changed, 1 insertion(+)

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git checkout master
Switched to branch 'master'
```

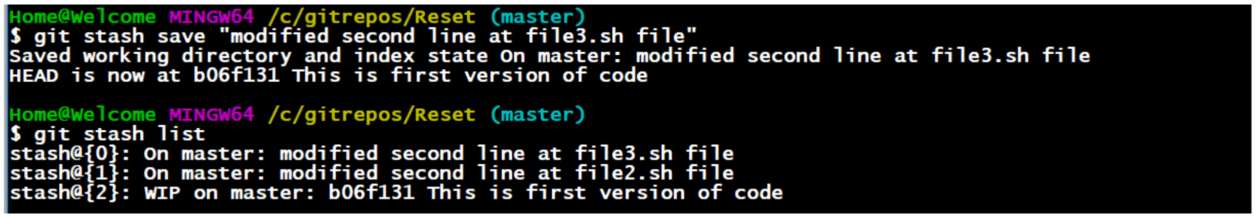
Again we came back to our previous work and we are not going to work on 2 prvious stash works gain we can start new wotk on new file called file3.sh



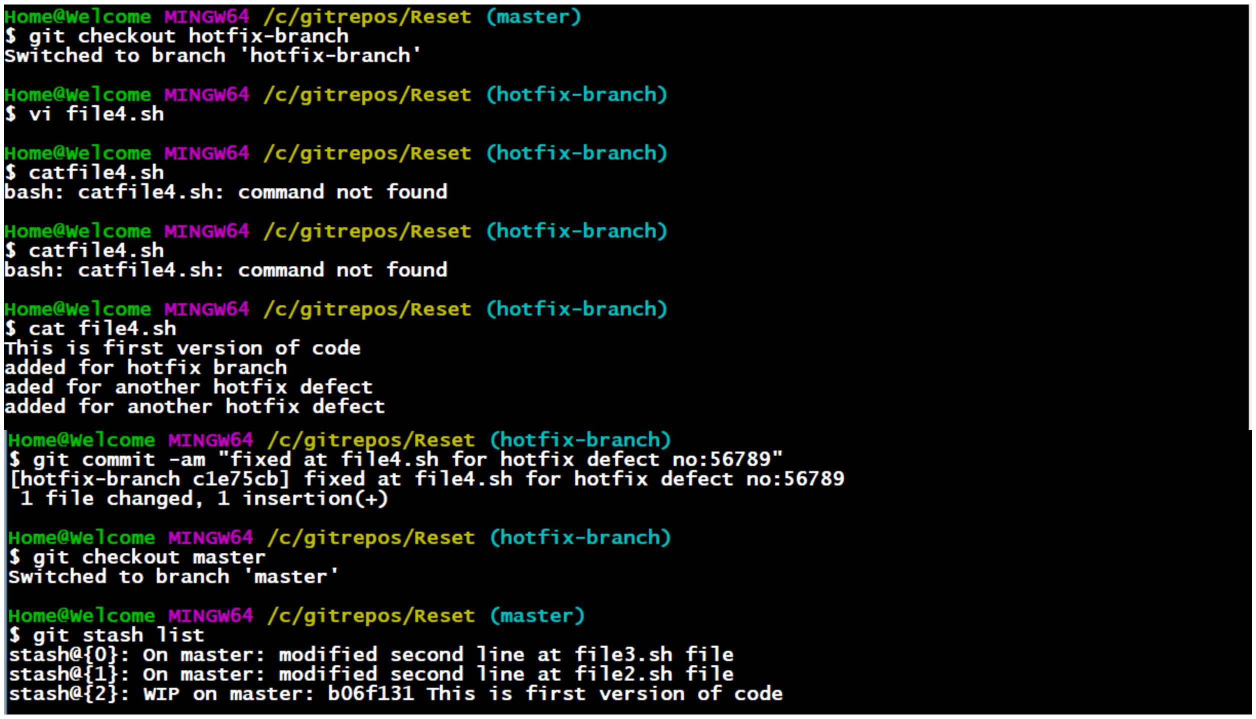
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file3.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file3.sh
This is first version of code
added for stash demo

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git add file3.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file3.sh
```

This time we have been added modified file file3.sh to staging area with git add command.

And we are doing again stash to go back to hotfix-branch



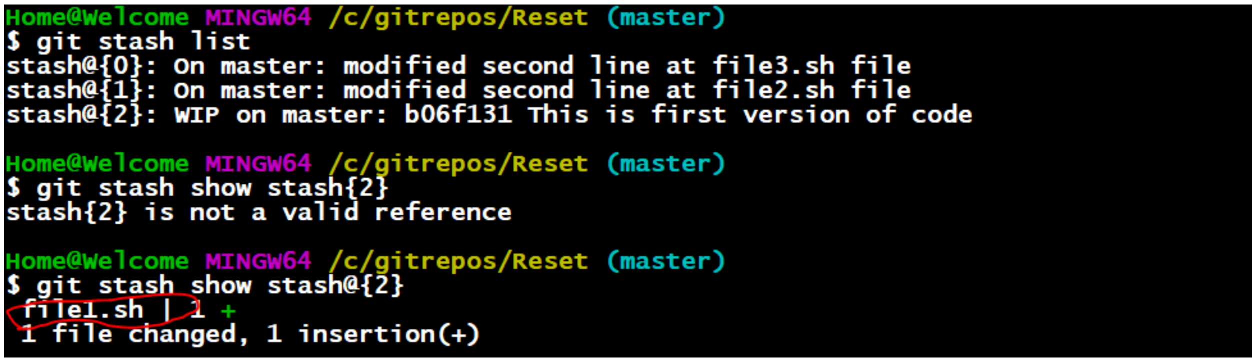
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash save "modified second line at file3.sh file"
Saved working directory and index state On master: modified second line at file3.sh file
HEAD is now at b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file
stash@{1}: On master: modified second line at file2.sh file
stash@{2}: WIP on master: b06f131 This is first version of code
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout hotfix-branch
Switched to branch 'hotfix-branch'

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ vi file4.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ catfile4.sh
bash: catfile4.sh: command not found

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ catfile4.sh
bash: catfile4.sh: command not found

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ cat file4.sh
This is first version of code
added for hotfix branch
aded for another hotfix defect
added for another hotfix defect
Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git commit -am "fixed at file4.sh for hotfix defect no:56789"
[hotfix-branch c1e75cb] fixed at file4.sh for hotfix defect no:56789
 1 file changed, 1 insertion(+)

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git checkout master
Switched to branch 'master'

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file
stash@{1}: On master: modified second line at file2.sh file
stash@{2}: WIP on master: b06f131 This is first version of code
```

Now while checking stashes when I see about stash{2} we are unable to find what changes we have been done so if we want to check about changes we have to use below command

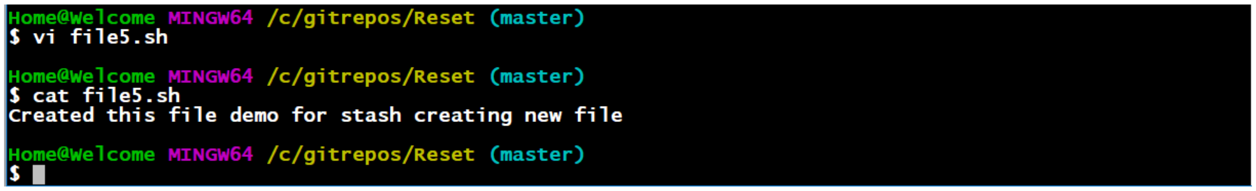
git stash show stash@{2}



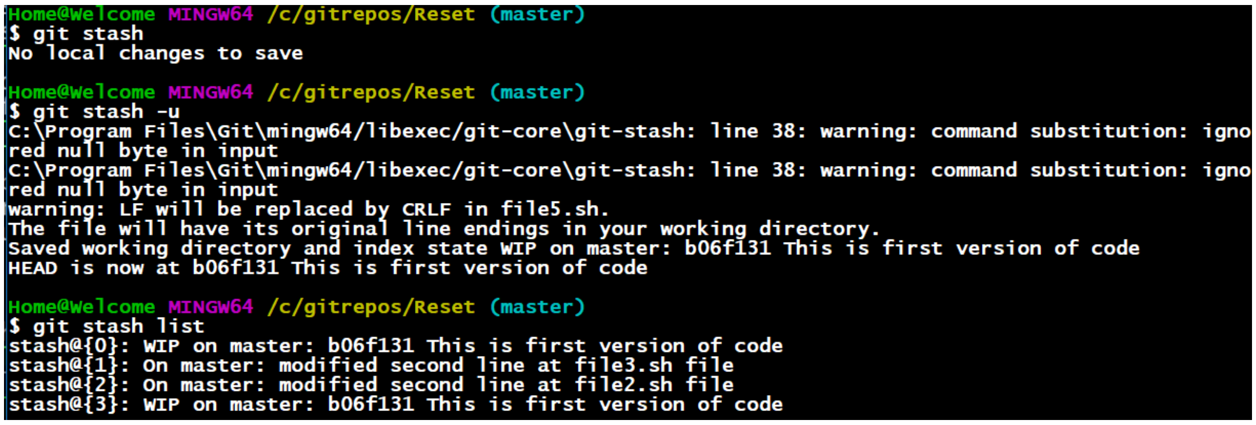
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file
stash@{1}: On master: modified second line at file2.sh file
stash@{2}: WIP on master: b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash show stash{2}
stash{2} is not a valid reference

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash show stash@{2}
 file1.sh | 1 +
 1 file changed, 1 insertion(+)
```

So now we can identify all changes what we have saved on stash number2.

So far we have created 3 stashes for 3 diff files those files are already tracked state once now we can create new file for new stash number 4. So here we have created new file called file5.sh

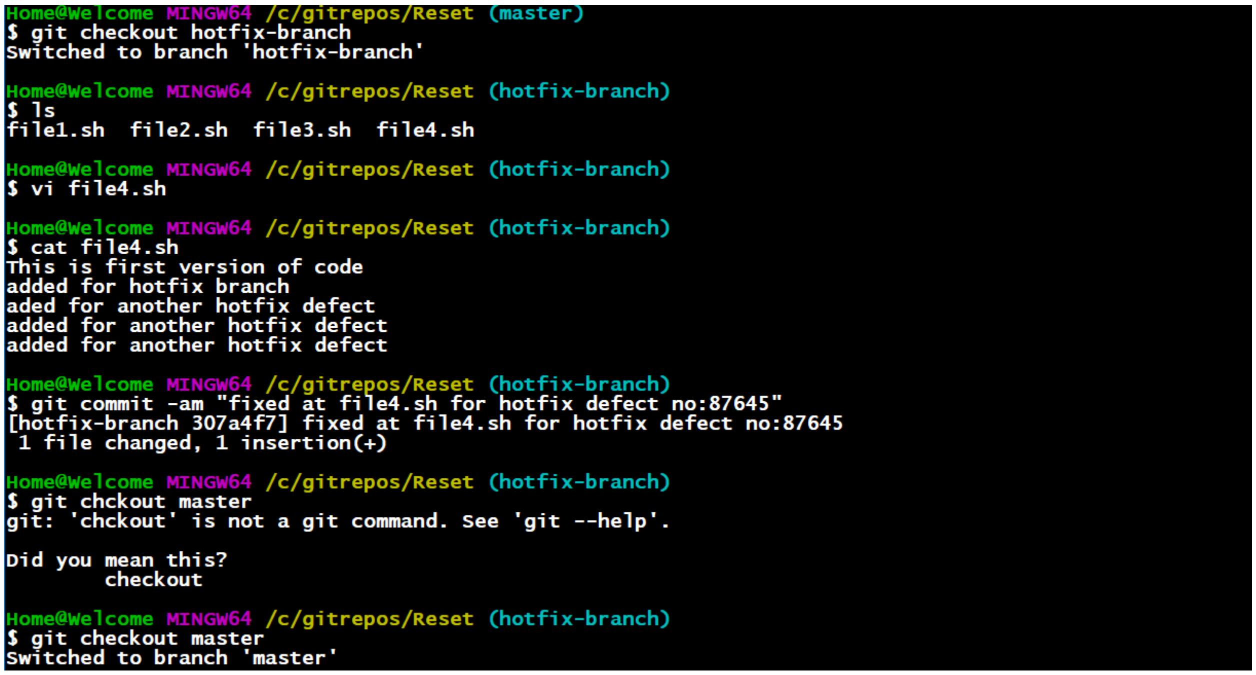


```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file5.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file5.sh
Created this file demo for stash creating new file

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ 
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash
No local changes to save

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash -u
C:\Program Files\Git\mingw64/libexec/git-core\git-stash: line 38: warning: command substitution: igno
red null byte in input
C:\Program Files\Git\mingw64/libexec/git-core\git-stash: line 38: warning: command substitution: igno
red null byte in input
warning: LF will be replaced by CRLF in file5.sh.
The file will have its original line endings in your working directory.
Saved working directory and index state WIP on master: b06f131 This is first version of code
HEAD is now at b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: WIP on master: b06f131 This is first version of code
stash@{1}: On master: modified second line at file3.sh file
stash@{2}: On master: modified second line at file2.sh file
stash@{3}: WIP on master: b06f131 This is first version of code
```

Bydefault git stash command will work only on tracked files so when we create new file not adding to staging area and trying to make as stash it wont allow to make it allow we have to add –u to our stash command like above.



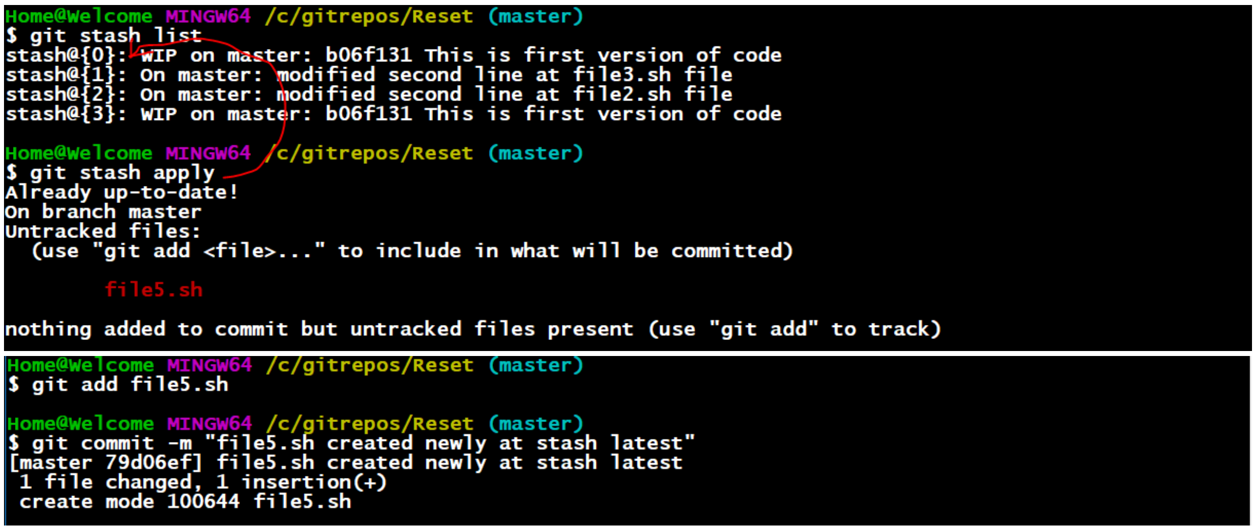
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout hotfix-branch
Switched to branch 'hotfix-branch'

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ ls
file1.sh  file2.sh  file3.sh  file4.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ vi file4.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ cat file4.sh
This is first version of code
added for hotfix branch
aded for another hotfix defect
added for another hotfix defect
added for another hotfix defect

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git commit -am "fixed at file4.sh for hotfix defect no:87645"
[hotfix-branch 307a4f7] fixed at file4.sh for hotfix defect no:87645
 1 file changed, 1 insertion(+)

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git chckout master
git: 'chckout' is not a git command. See 'git --help'.

Did you mean this?
        checkout

Home@Welcome MINGW64 /c/gitrepos/Reset (hotfix-branch)
$ git checkout master
Switched to branch 'master'
```

Till now we have worked on how to create stashes on tracked files and on un tracked files and how to display stash details. Now we can see how to go back to stash and what we are going to do there and how going to do there.

Now we have done all our hot fixes and we can start our work on previous unfinished works on 4 stashes.

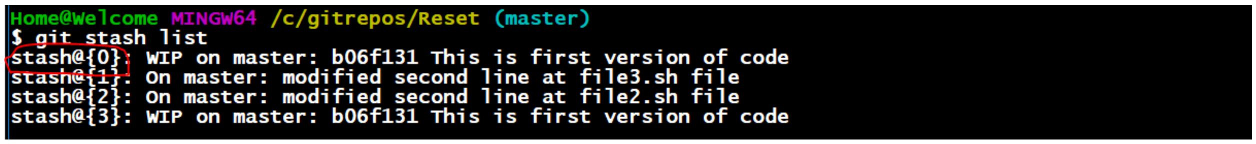
apply : apply is a sub command for stash for resuming our unfinished works on any stash of our’s.

it can have stash number or it can have empty parameter if empty means it will take bydefault the most recent stash ex : stash@{0}



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: WIP on master: b06f131 This is first version of code
stash@{1}: On master: modified second line at file3.sh file
stash@{2}: On master: modified second line at file2.sh file
stash@{3}: WIP on master: b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash apply
Already up-to-date!
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file5.sh

nothing added to commit but untracked files present (use "git add" to track)
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git add file5.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git commit -m "file5.sh created newly at stash latest"
[master 79d06ef] file5.sh created newly at stash latest
 1 file changed, 1 insertion(+)
 create mode 100644 file5.sh
```

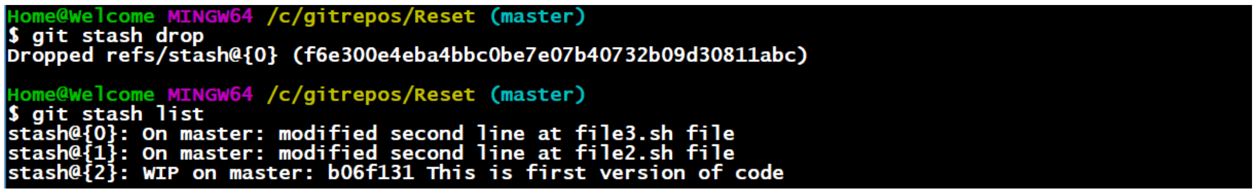
Since it is a untacked file so we have to add first and then commit next like above.



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: WIP on master: b06f131 This is first version of code
stash@{1}: On master: modified second line at file3.sh file
stash@{2}: On master: modified second line at file2.sh file
stash@{3}: WIP on master: b06f131 This is first version of code
```

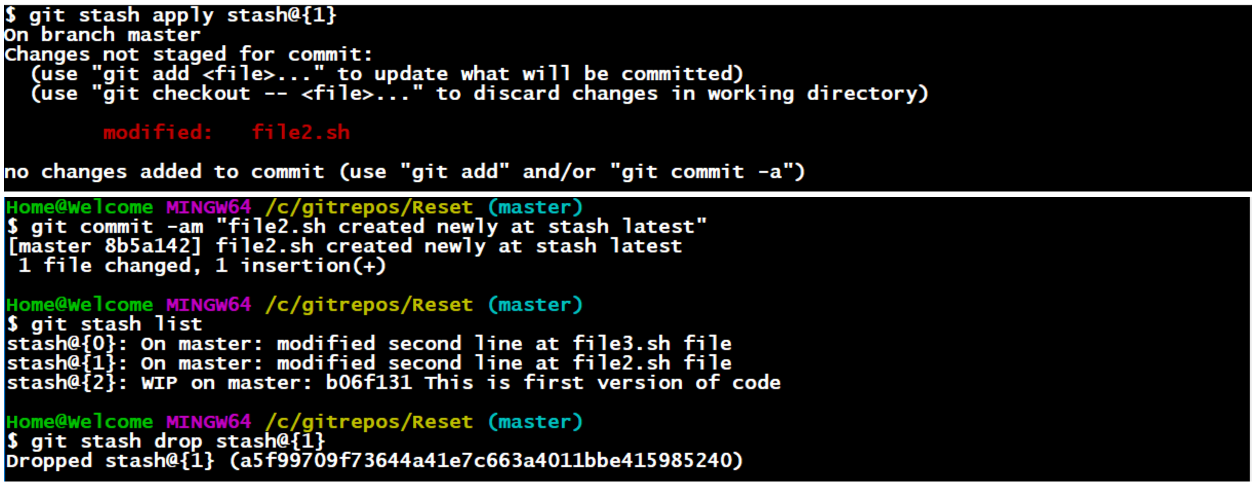
Here still we can see the applied stash as well since it is applied already and commited as well so we don’t need this stash so we can remove this stash using drop command

git stash drop stash@{0}   --- if we don’t give any stash number bydefault will take most recent means 0.



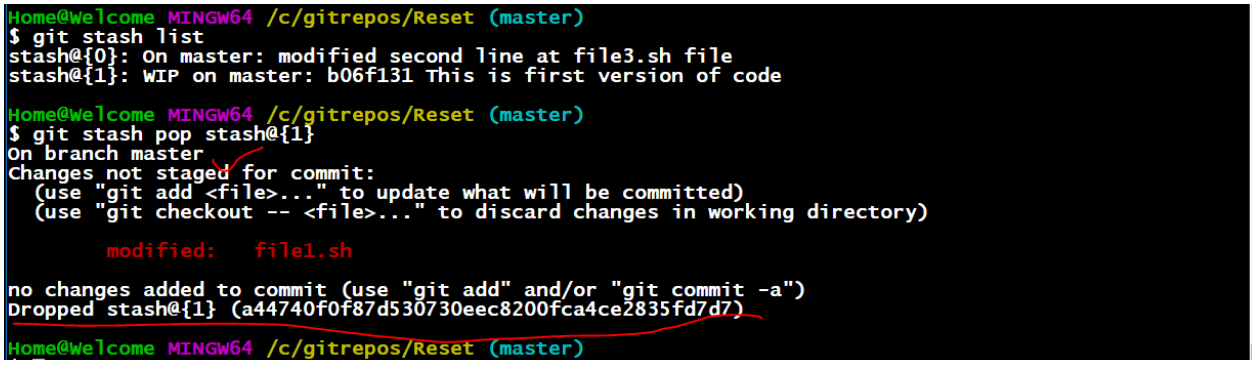
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash drop
Dropped refs/stash@{0} (f6e300e4eba4bbc0be7e07b40732b09d30811abc)

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file
stash@{1}: On master: modified second line at file2.sh file
stash@{2}: WIP on master: b06f131 This is first version of code
```

Again one more stash am going to apply but most recent we can apply stash@{1} like below

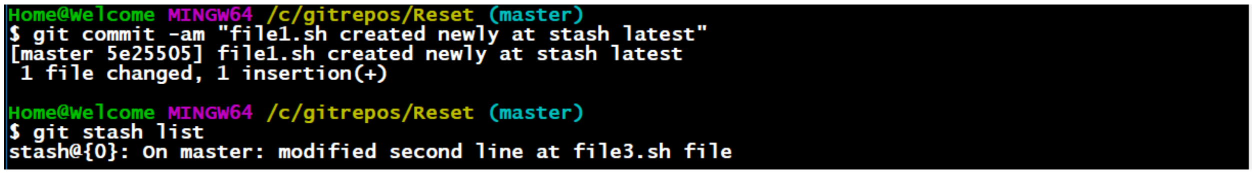


```
$ git stash apply stash@{1}
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file2.sh

no changes added to commit (use "git add" and/or "git commit -a")
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git commit -am "file2.sh created newly at stash latest"
[master 8b5a142] file2.sh created newly at stash latest
 1 file changed, 1 insertion(+)

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file
stash@{1}: On master: modified second line at file2.sh file
stash@{2}: WIP on master: b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash drop stash@{1}
Dropped stash@{1} (a5f99709f73644a41e7c663a4011bbe415985240)
```

So till now we have applied stash and after finished our work we removed our stash but we are doing seperately. But now can use another command called pop it can apply and drop stashes like below

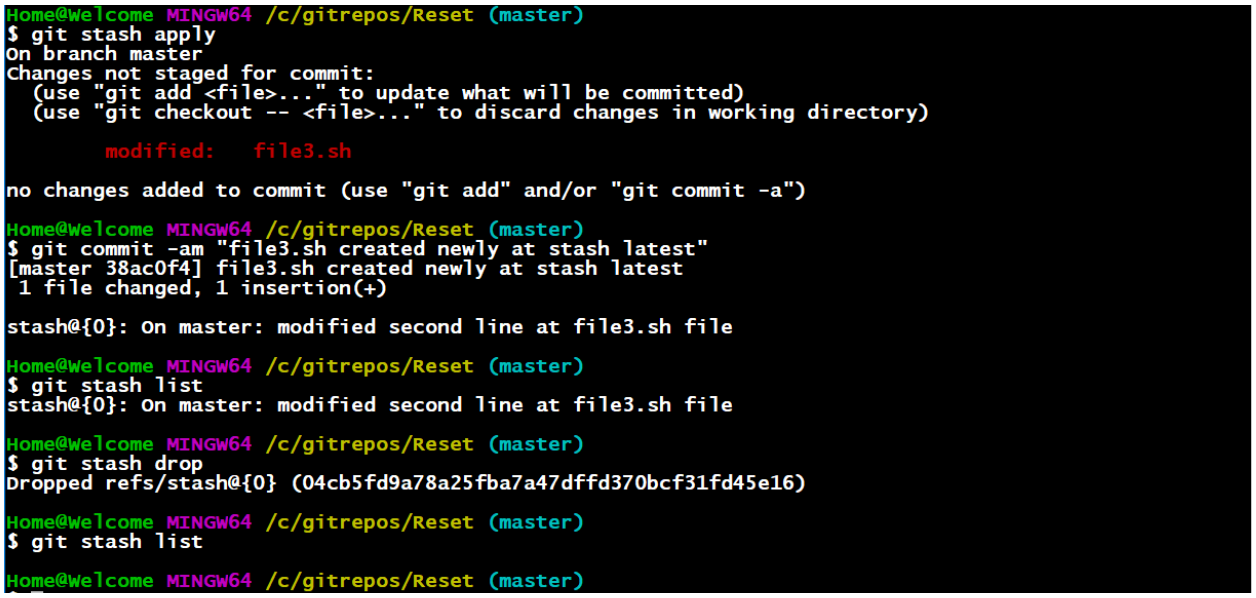


```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file
stash@{1}: WIP on master: b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash pop stash@{1}
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file1.sh

no changes added to commit (use "git add" and/or "git commit -a")
Dropped stash@{1} (a44740f0f87d530730eec8200fca4ce2835fd7d7)

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git commit -am "file1.sh created newly at stash latest"
[master 5e25505] file1.sh created newly at stash latest
 1 file changed, 1 insertion(+)

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file
```

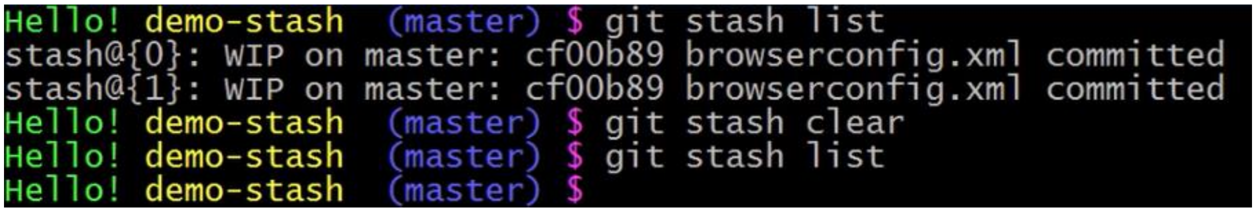
Now we can have only one stash in our work.



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file3.sh

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

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git commit -am "file3.sh created newly at stash latest"
[master 38ac0f4] file3.sh created newly at stash latest
 1 file changed, 1 insertion(+)

stash@{0}: On master: modified second line at file3.sh file

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: modified second line at file3.sh file

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash drop
Dropped refs/stash@{0} (04cb5fd9a78a25fba7a47dffd370bcf31fd45e16)

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```

Now we don't have any stash list on our work.

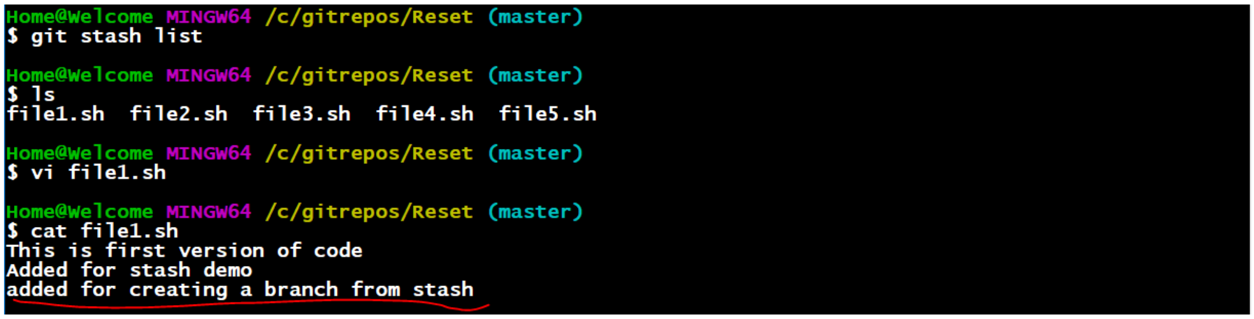
There is powerful command to remove all stashes at a time at one shot of command like below we have to enter. git stash clear



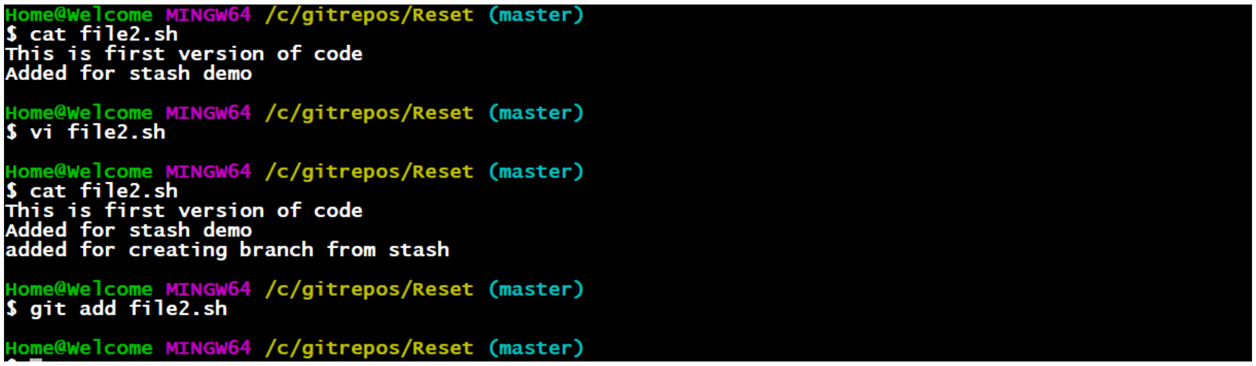
```
Hello! demo-stash  (master) $ git stash list
stash@{0}: WIP on master: cf00b89 browserconfig.xml committed
stash@{1}: WIP on master: cf00b89 browserconfig.xml committed
Hello! demo-stash  (master) $ git stash clear
Hello! demo-stash  (master) $ git stash list
Hello! demo-stash  (master) $
```

GIT Stash : Creating a Branch from a Stash :

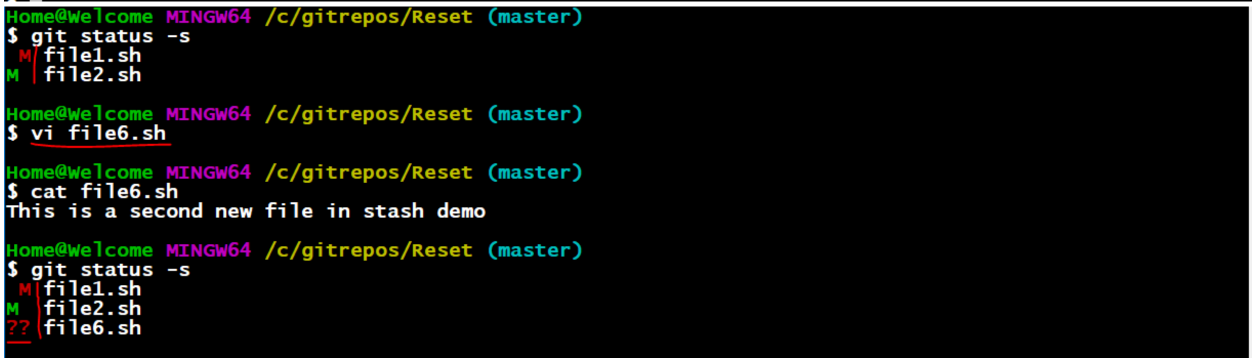
We are going to show how to create a branch using stash changes. Suppose we changed quite lot of changes on our files and stashed all those changes now we decided move to all changes what we done and stashed at master branch those things we want to move and make a new branch.



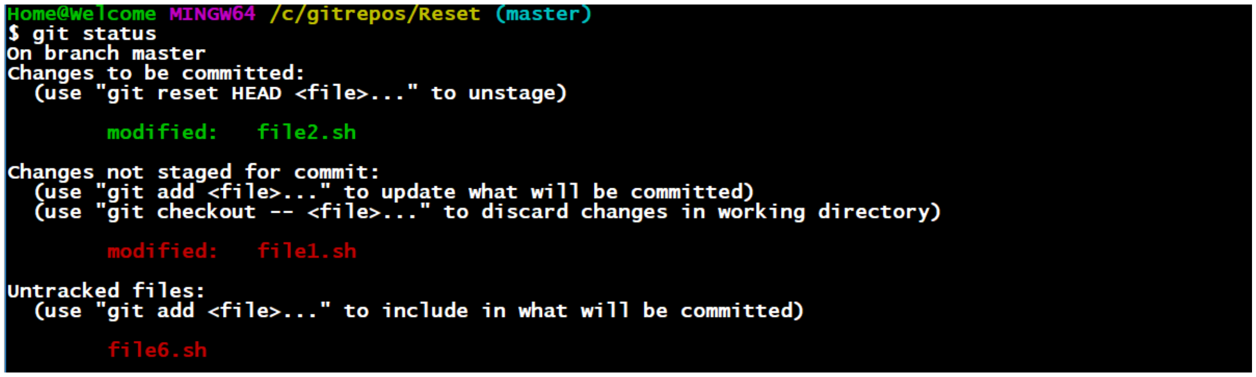
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ ls
file1.sh  file2.sh  file3.sh  file4.sh  file5.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file1.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file1.sh
This is first version of code
Added for stash demo
added for creating a branch from stash
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file2.sh
This is first version of code
Added for stash demo

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file2.sh
This is first version of code
Added for stash demo
added for creating branch from stash

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git add file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```



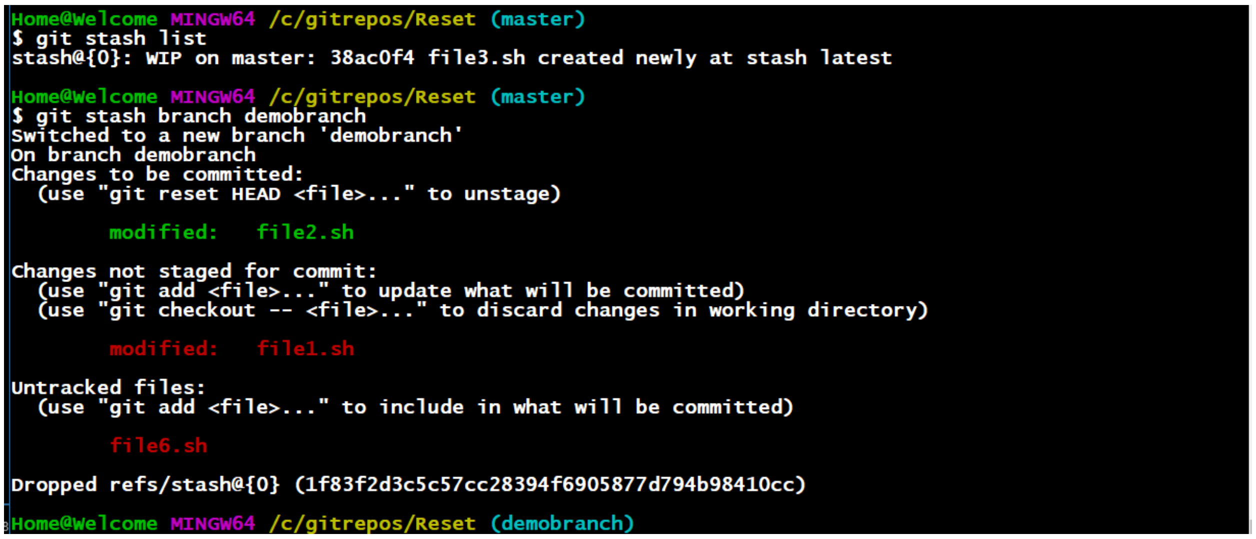
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status -s
 M file1.sh
M  file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file6.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file6.sh
This is a second new file in stash demo

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status -s
 M file1.sh
M  file2.sh
?? file6.sh
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file2.sh

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file1.sh

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

        file6.sh
```

We just added a new file which means untracked file and one file just modified and another modified and added to staging area

-s means abbrevative format.



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: WIP on master: 38ac0f4 file3.sh created newly at stash latest

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash branch demobranch
Switched to a new branch 'demobranch'
On branch demobranch
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file2.sh

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file1.sh

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

        file6.sh

Dropped refs/stash@{0} (1f83f2d3c5c57cc28394f6905877d794b98410cc)

Home@Welcome MINGW64 /c/gitrepos/Reset (demobranch)
```

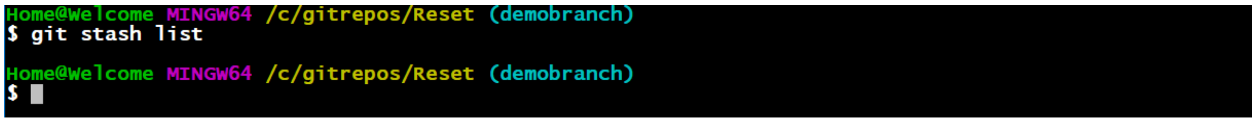
Git has been done several things in a single go does 4 things here.

It creates the branch

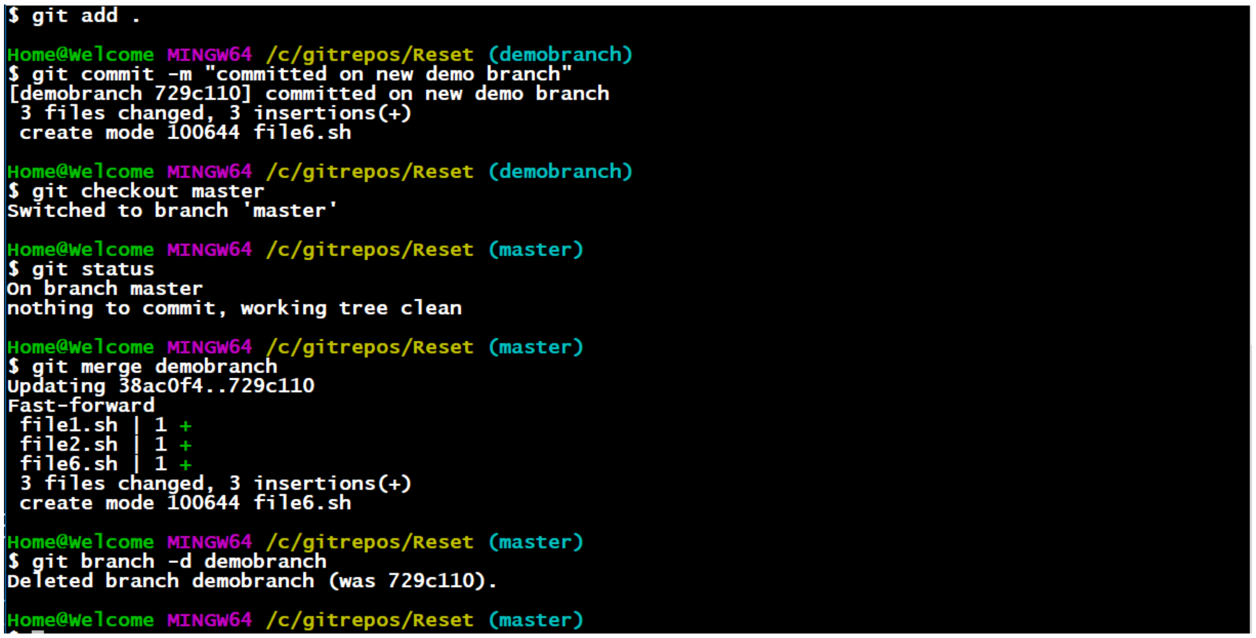
It switches to the new branch

It applies the stash

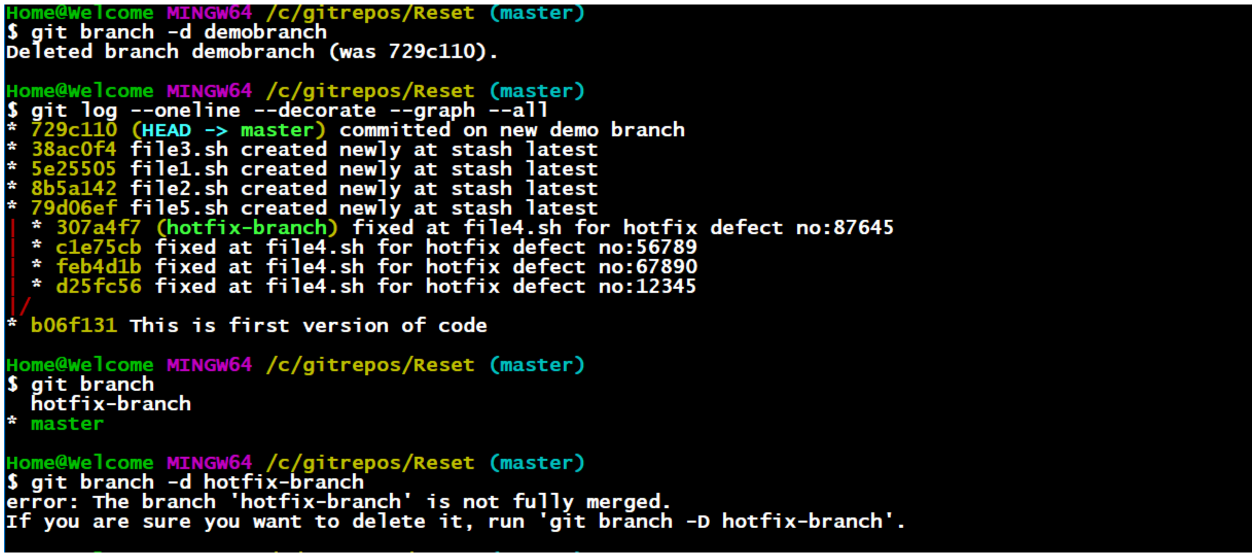
It drops the stash as well



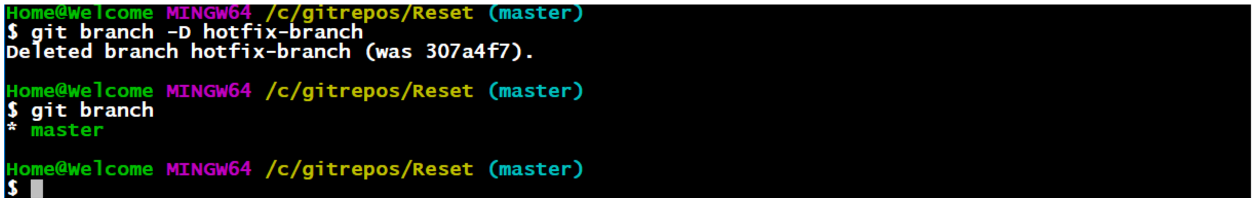
```
Home@Welcome MINGW64 /c/gitrepos/Reset (demobranch)
$ git stash list

Home@Welcome MINGW64 /c/gitrepos/Reset (demobranch)
$ 
```



```
$ git add .

Home@Welcome MINGW64 /c/gitrepos/Reset (demobranch)
$ git commit -m "committed on new demo branch"
[demobranch 729c110] committed on new demo branch
 3 files changed, 3 insertions(+)
 create mode 100644 file6.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (demobranch)
$ git checkout master
Switched to branch 'master'

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status
On branch master
nothing to commit, working tree clean

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git merge demobranch
Updating 38ac0f4..729c110
Fast-forward
 file1.sh | 1 +
 file2.sh | 1 +
 file6.sh | 1 +
 3 files changed, 3 insertions(+)
 create mode 100644 file6.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git branch -d demobranch
Deleted branch demobranch (was 729c110).

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```

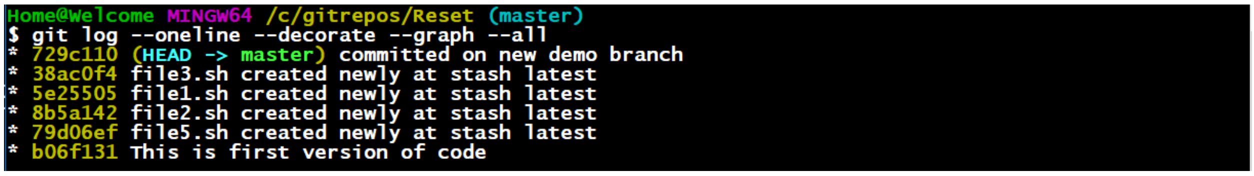


```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git branch -d demobranch
Deleted branch demobranch (was 729c110).

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git log --oneline --decorate --graph --all
* 729c110 (HEAD -> master) committed on new demo branch
* 38ac0f4 file3.sh created newly at stash latest
* 5e25505 file1.sh created newly at stash latest
* 8b5a142 file2.sh created newly at stash latest
* 79d06ef file5.sh created newly at stash latest
| * 307a4f7 (hotfix-branch) fixed at file4.sh for hotfix defect no:87645
| * c1e75cb fixed at file4.sh for hotfix defect no:56789
| * feb4d1b fixed at file4.sh for hotfix defect no:67890
| * d25fc56 fixed at file4.sh for hotfix defect no:12345
|/
* b06f131 This is first version of code

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git branch
  hotfix-branch
* master

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git branch -d hotfix-branch
error: The branch 'hotfix-branch' is not fully merged.
If you are sure you want to delete it, run 'git branch -D hotfix-branch'.
```



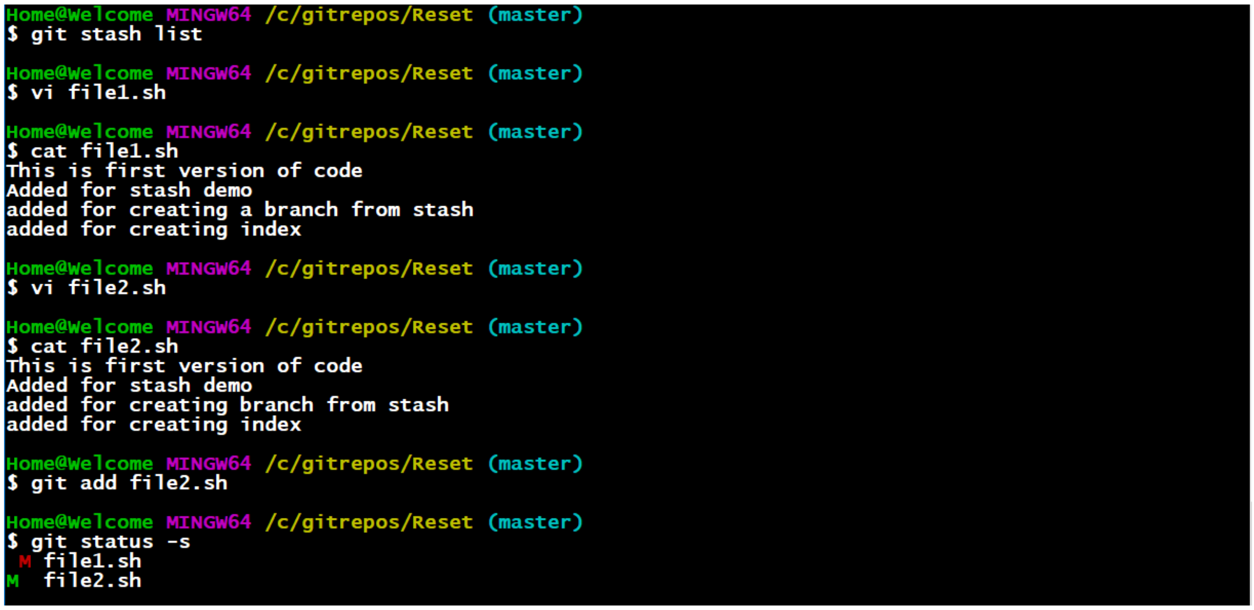
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git branch -D hotfix-branch
Deleted branch hotfix-branch (was 307a4f7).

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git branch
* master

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ 
```

If you don’t merge any branch with master and try to delete that branch cant delete we can delete via capital D not with small d like above.

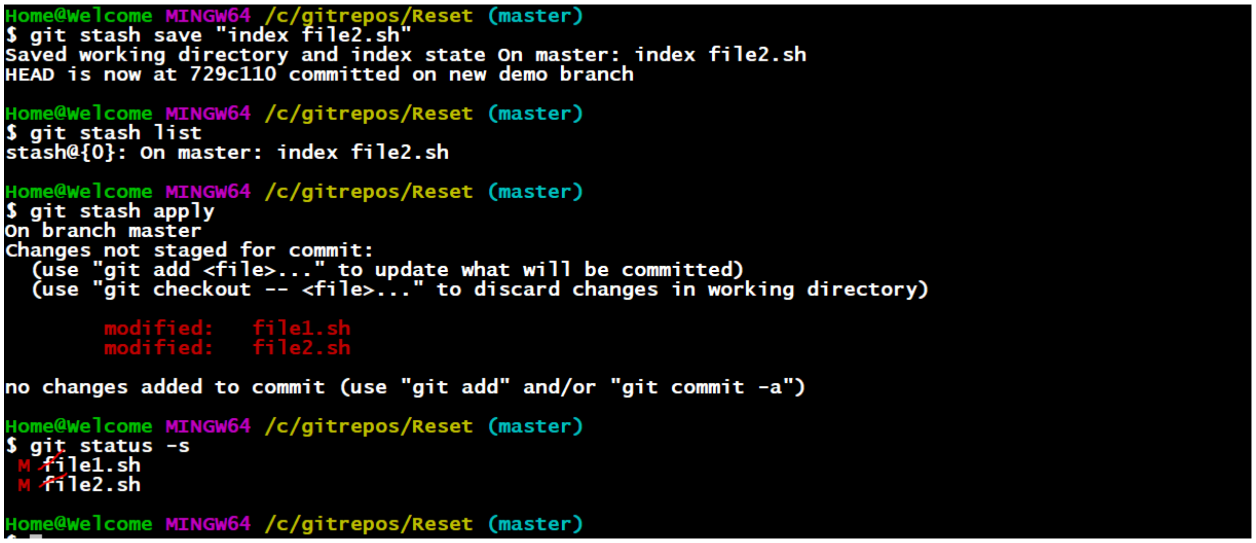


```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git log --oneline --decorate --graph --all
* 729c110 (HEAD -> master) committed on new demo branch
* 38ac0f4 file3.sh created newly at stash latest
* 5e25505 file1.sh created newly at stash latest
* 8b5a142 file2.sh created newly at stash latest
* 79d06ef file5.sh created newly at stash latest
* b06f131 This is first version of code
```

Now we can discuss about index command in stash



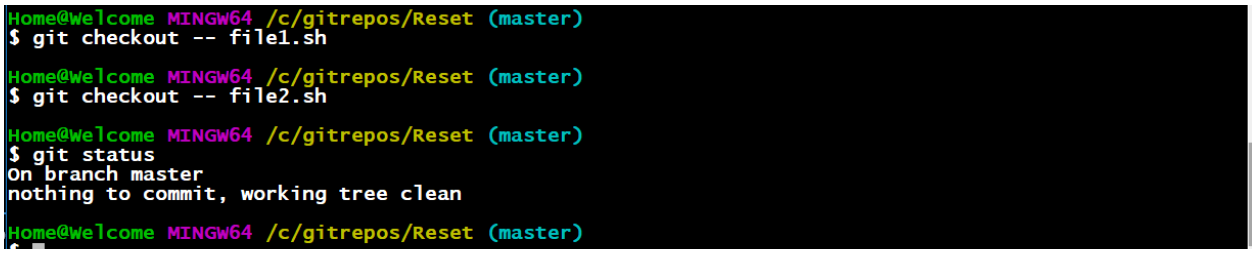
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file1.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file1.sh
This is first version of code
Added for stash demo
added for creating a branch from stash
added for creating index

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ cat file2.sh
This is first version of code
Added for stash demo
added for creating branch from stash
added for creating index

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git add file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status -s
 M file1.sh
M  file2.sh
```



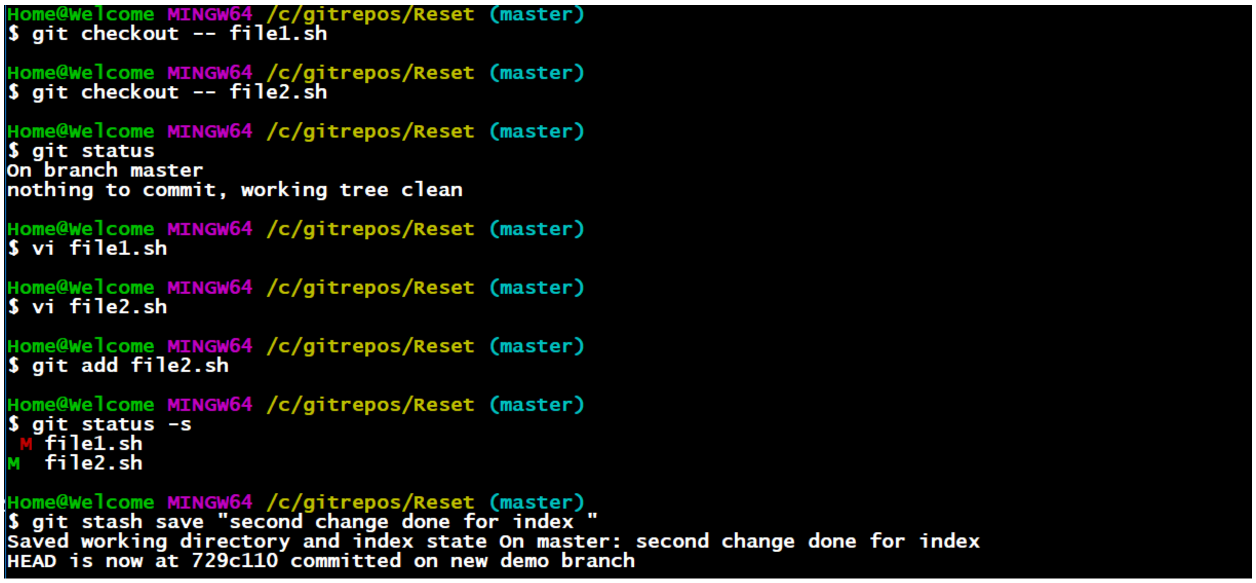
```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash save "index file2.sh"
Saved working directory and index state On master: index file2.sh
HEAD is now at 729c110 committed on new demo branch

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: index file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash apply
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file1.sh
        modified:   file2.sh

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

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status -s
 M file1.sh
 M file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```

Now both files are in just modified state after applying stash but before doing stash file2.sh added to stage as well.

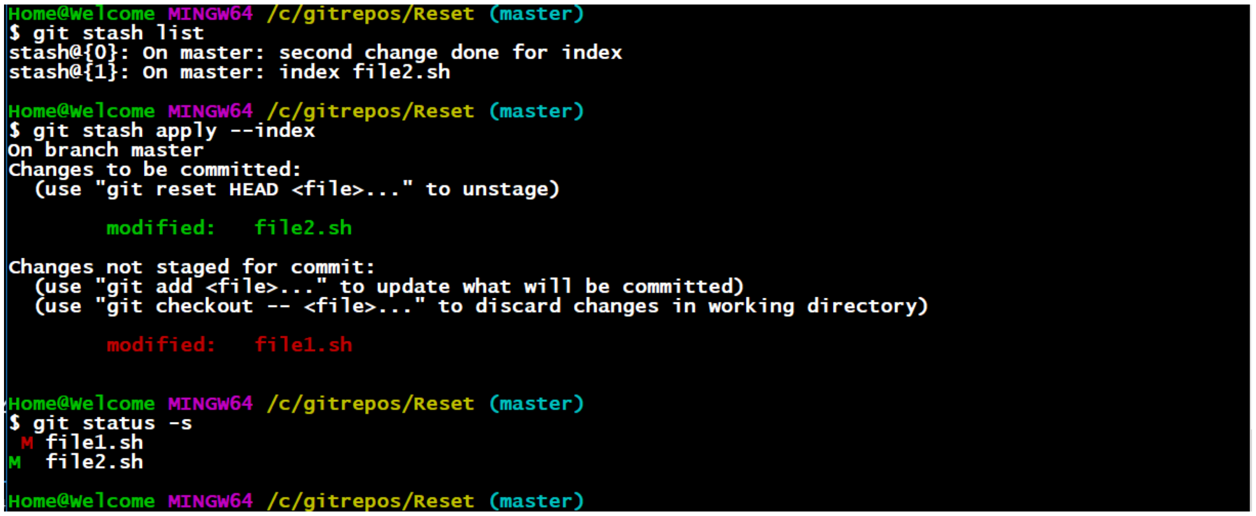
For this we have use index command for this we can rol back our changes to back like below



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout -- file1.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout -- file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status
On branch master
nothing to commit, working tree clean

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout -- file1.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git checkout -- file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status
On branch master
nothing to commit, working tree clean

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file1.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ vi file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git add file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status -s
 M file1.sh
M  file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash save "second change done for index "
Saved working directory and index state On master: second change done for index
HEAD is now at 729c110 committed on new demo branch
```



```
Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash list
stash@{0}: On master: second change done for index
stash@{1}: On master: index file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git stash apply --index
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   file2.sh

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   file1.sh


Home@Welcome MINGW64 /c/gitrepos/Reset (master)
$ git status -s
 M file1.sh
M  file2.sh

Home@Welcome MINGW64 /c/gitrepos/Reset (master)
```

Like this we can create index here.