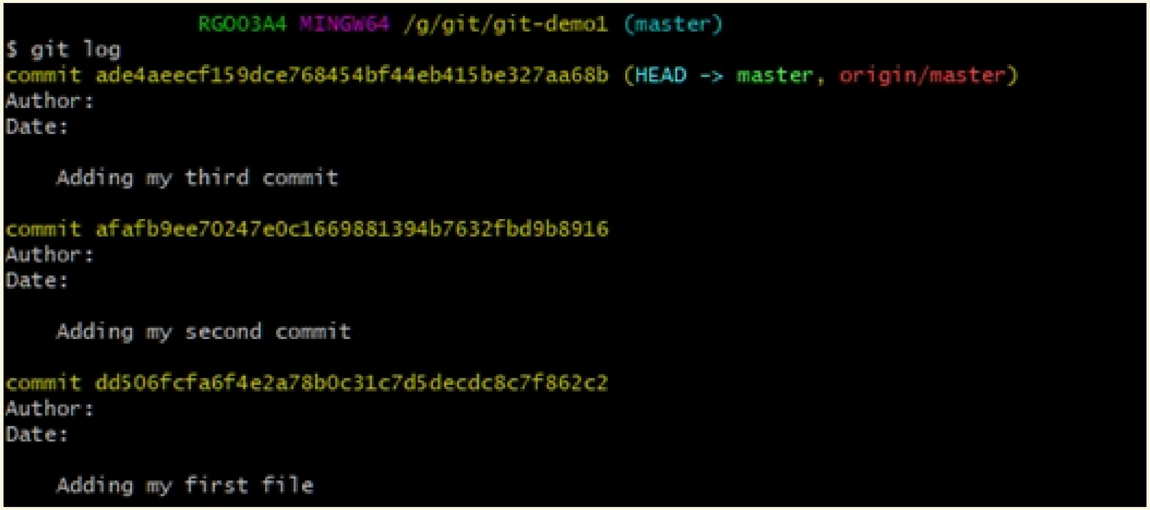
Git Stash

stash ~
to store safely in a hidden or secret place.

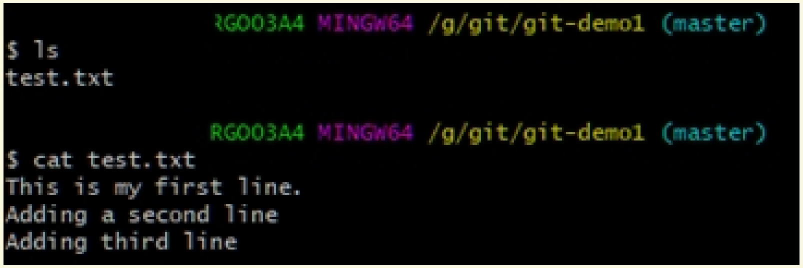
Let's understand with an actual scenario -

You are currently inside the local repository 'git-demo1' which is in-sync with the remote repository and you don't have any local changes.



```
              RG003A4 MINGW64 /g/git/git-demo1 (master)
$ git log
commit ade4aeecf159dce768454bf44eb415be327aa68b (HEAD -> master, origin/master)
Author:
Date:

    Adding my third commit

commit afafb9ee70247e0c1669881394b7632fbd9b8916
Author:
Date:

    Adding my second commit

commit dd506fcfa6f4e2a78b0c31c7d5decdc8c7f862c2
Author:
Date:

    Adding my first file
```

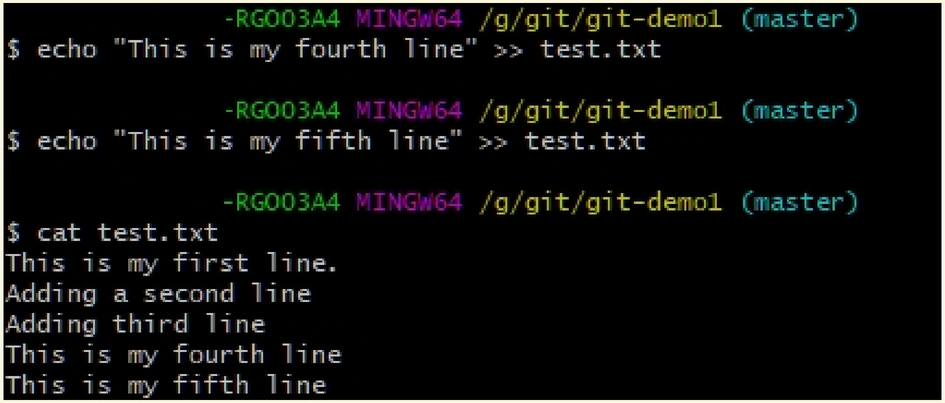
There is a text file 'test.txt' which we will be using for the demo. Printed the content of the file.



```
              RG003A4 MINGW64 /g/git/git-demo1 (master)
$ ls
test.txt

              RG003A4 MINGW64 /g/git/git-demo1 (master)
$ cat test.txt
This is my first line.
Adding a second line
Adding third line
```

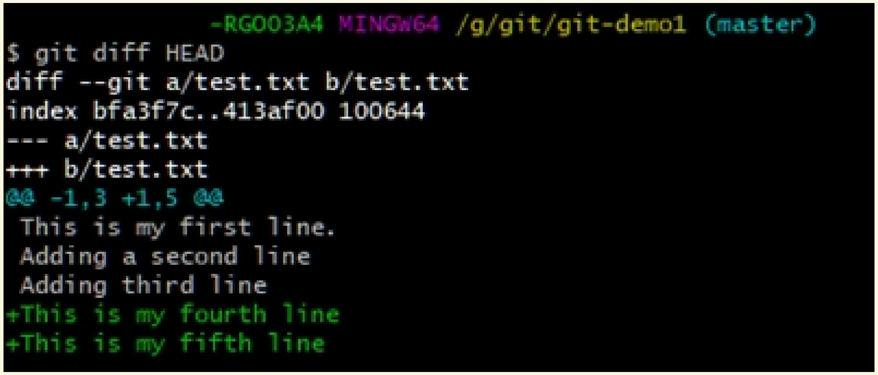
@mayank_ahuja

Now, let's add new lines in the text file. Added 2 new lines.



```
              -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ echo "This is my fourth line" >> test.txt

              -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ echo "This is my fifth line" >> test.txt

              -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ cat test.txt
This is my first line.
Adding a second line
Adding third line
This is my fourth line
This is my fifth line
```

You can see the difference wrt HEAD using command 'git diff HEAD'. Green lines were added.



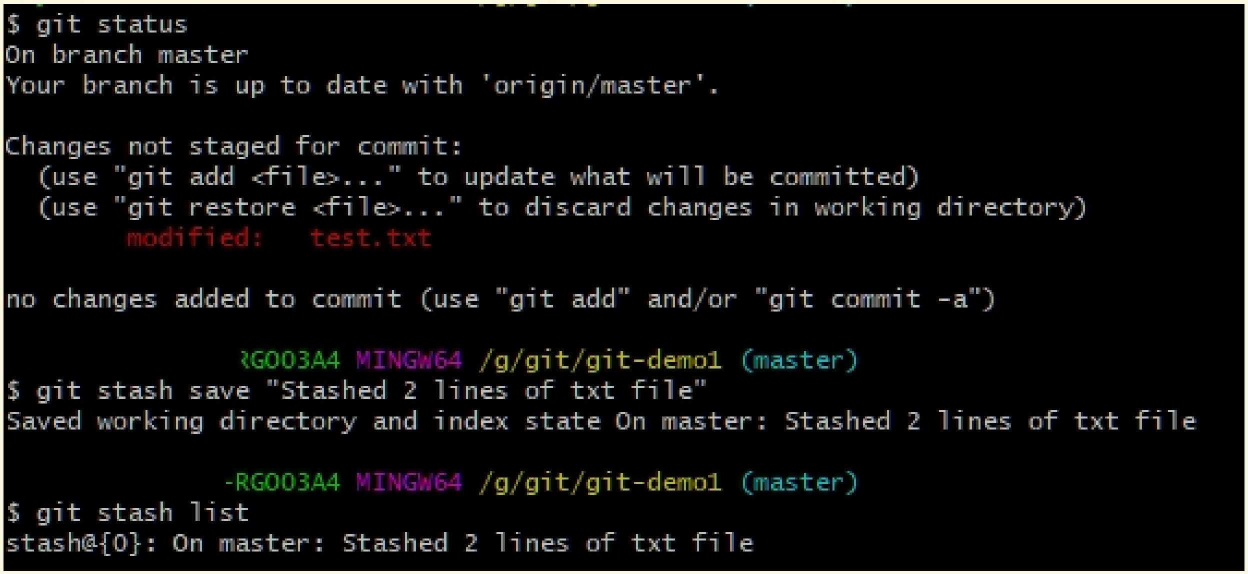
```
              -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git diff HEAD
diff --git a/test.txt b/test.txt
index bfa3f7c..413af00 100644
--- a/test.txt
+++ b/test.txt
@@ -1,3 +1,5 @@
 This is my first line.
 Adding a second line
 Adding third line
+This is my fourth line
+This is my fifth line
```

@mayank_ahuja

Now due to some reason you don't want these new changes in the text file now, but you want to keep them stored somewhere locally as you might need them later.

You basically need a clean working directory at the moment.

'git stash' can help you in this scenario. The command saves your local modifications away and reverts the working directory to match the HEAD commit.

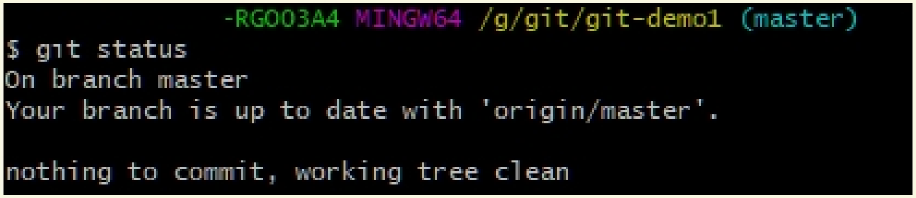


```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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:   test.txt

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

              RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash save "Stashed 2 lines of txt file"
Saved working directory and index state On master: Stashed 2 lines of txt file

             -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: On master: Stashed 2 lines of txt file
```

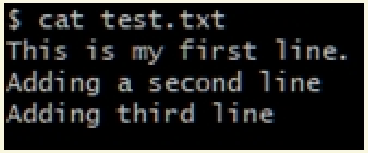
'git status' shows that the test.txt file has been modified.
But as we wanted to stash the changes we ran 'git stash save "message"' command and did 'git stash list' to check the stash and we can see that our change has been saved.

@mayank_ahuja

Now if you do 'git status' you will se clean working directory. The 2 lines that we added in the txt file are gone.

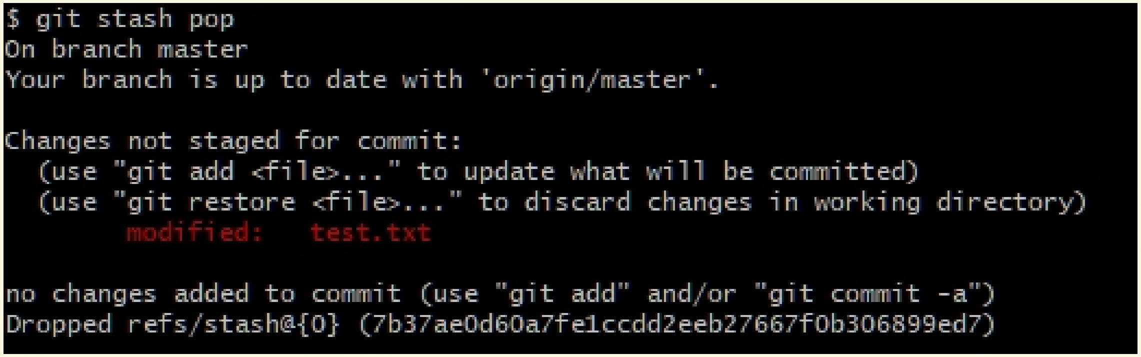


```
              -RG003A4 MINGW64 /g/git/git-demo1 (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

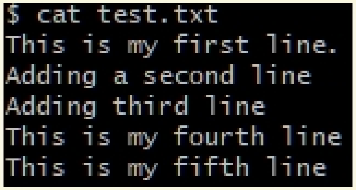


```
$ cat test.txt
This is my first line.
Adding a second line
Adding third line
```

But what if you need the stashed changes back? You need those 2 lines back in the text file.
'git stash pop' command gives you the last stashed change.



```
$ git stash pop
On branch master
Your branch is up to date with 'origin/master'.

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:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (7b37ae0d60a7fe1ccdd2eeb27667f0b306899ed7)
```

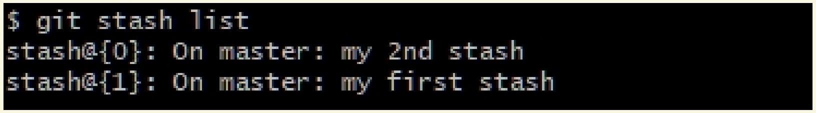


```
$ cat test.txt
This is my first line.
Adding a second line
Adding third line
This is my fourth line
This is my fifth line
```

That's mainly how stashing works. Now let's learn some more commands wrt git stash.

@mayank_ahuja

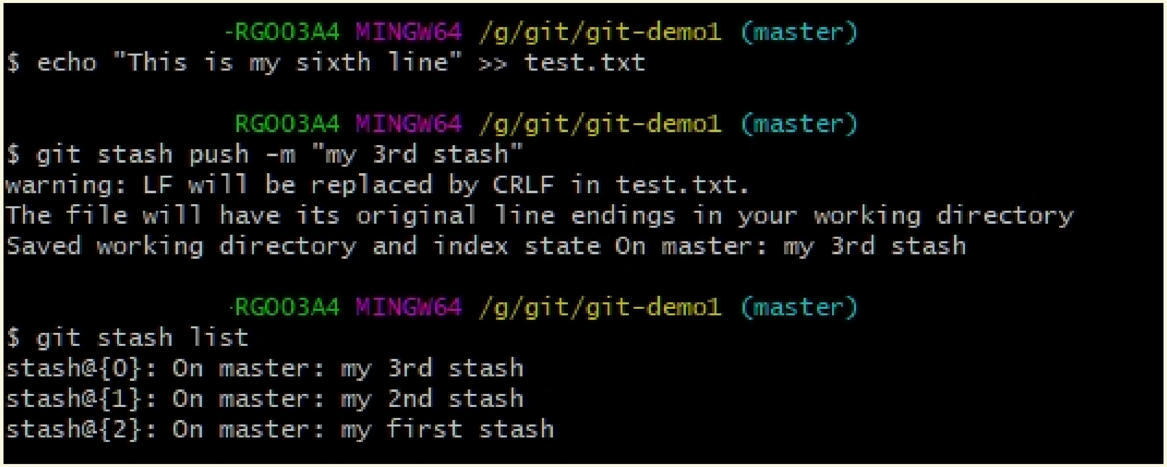
'git stash list' - to know all stashed changes that are available/stored



```
$ git stash list
stash@{0}: On master: my 2nd stash
stash@{1}: On master: my first stash
```

'git stash save' is now deprecated and hence you can use 'git stash push' instead to stash a change.

git stash push -m "message"

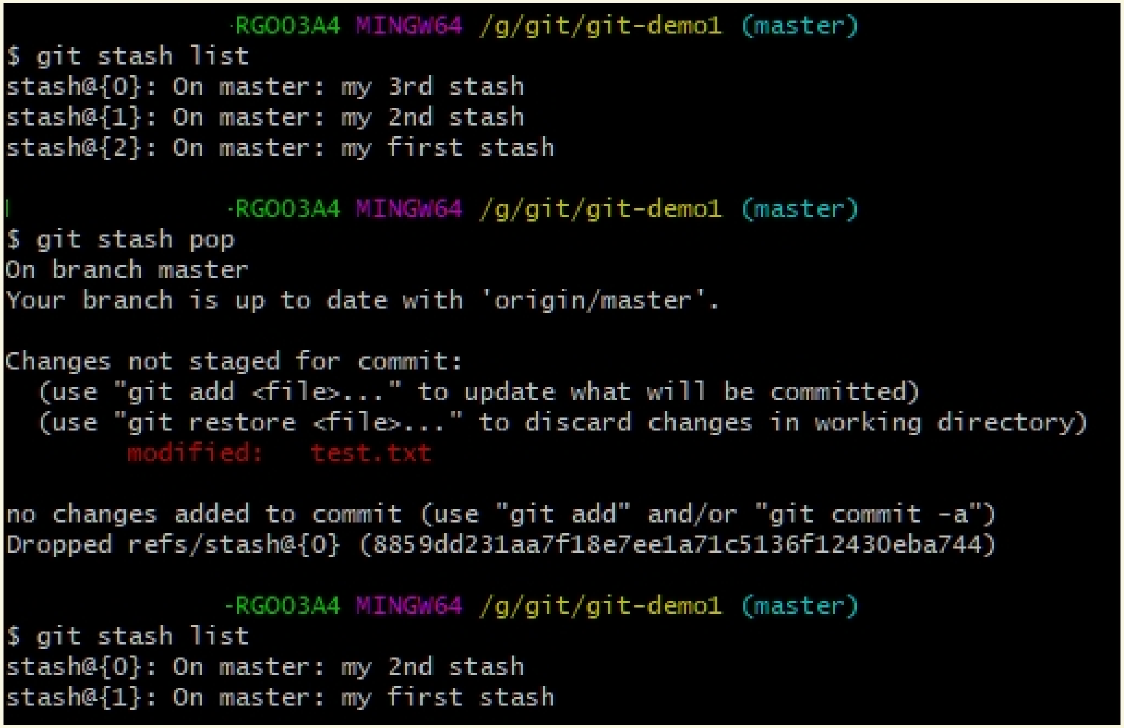


```
               -RG003A4 MINGW64 /g/git/git-demo1 (master)
$ echo "This is my sixth line" >> test.txt

                RG003A4 MINGW64 /g/git/git-demo1 (master)
$ git stash push -m "my 3rd stash"
warning: LF will be replaced by CRLF in test.txt.
The file will have its original line endings in your working directory
Saved working directory and index state On master: my 3rd stash

               -RG003A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: On master: my 3rd stash
stash@{1}: On master: my 2nd stash
stash@{2}: On master: my first stash
```

@mayank_ahuja

'git stash pop' vs 'git stash apply'
You can use either of them to re-apply the stashed change to the working directory, but the difference is, pop will remove the change from stash (stack) but when you do apply it's kept intact in the stack.

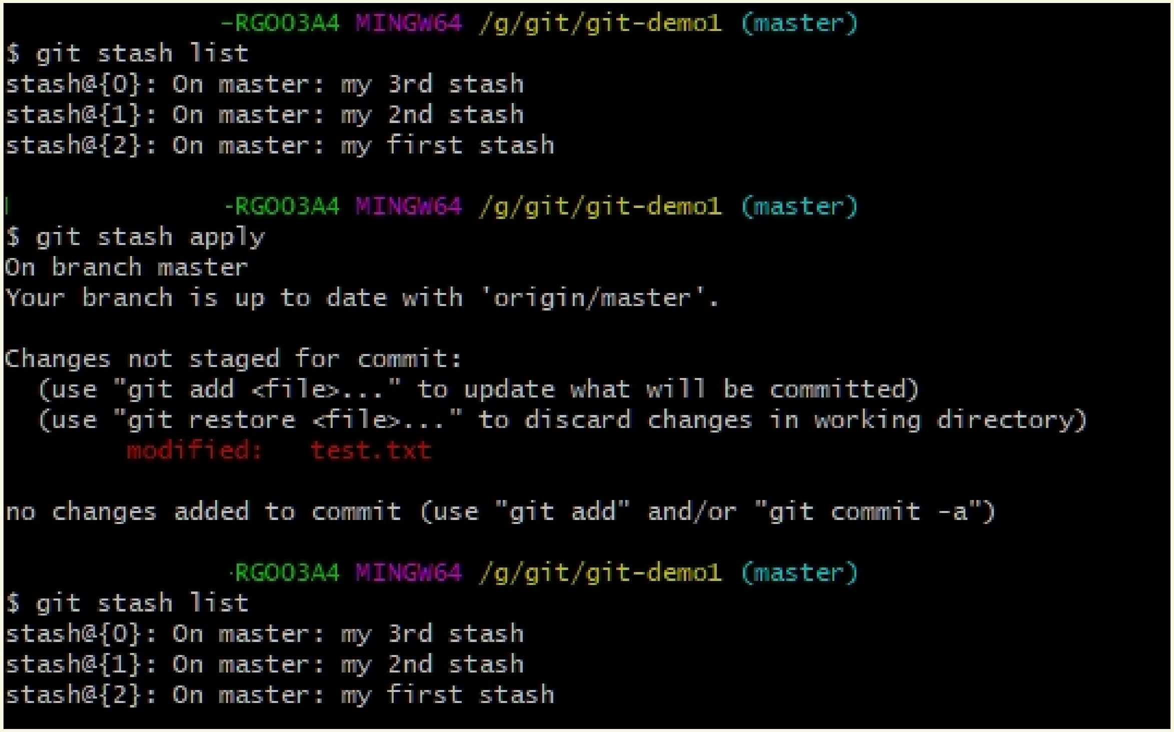
git stash pop



```
               -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: On master: my 3rd stash
stash@{1}: On master: my 2nd stash
stash@{2}: On master: my first stash

               -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash pop
On branch master
Your branch is up to date with 'origin/master'.

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:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (8859dd231aa7f18e7ee1a71c5136f12430eba744)

               -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: On master: my 2nd stash
stash@{1}: On master: my first stash
```

@mayank_ahuja

git stash apply

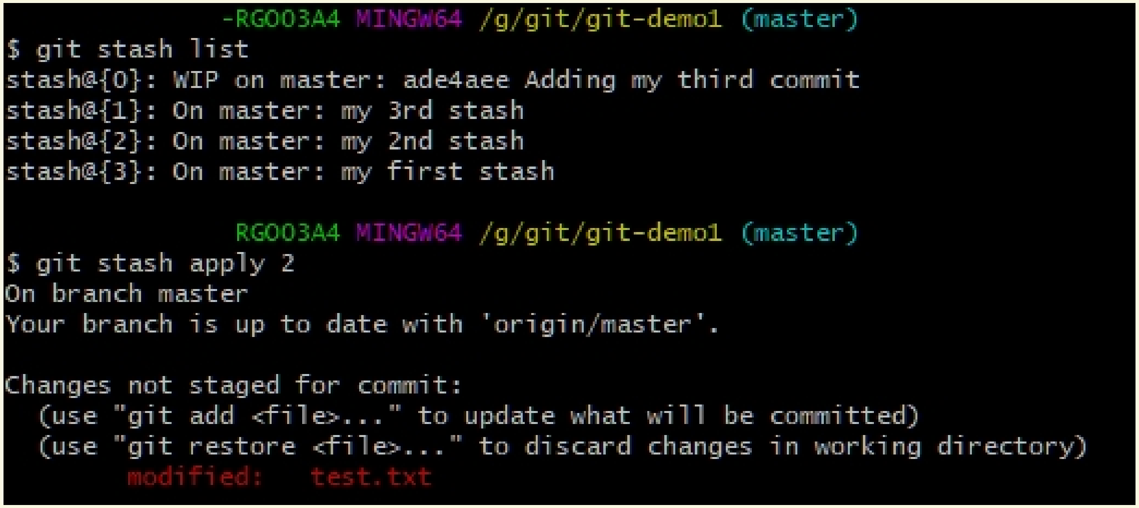


```
              -RG003A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: On master: my 3rd stash
stash@{1}: On master: my 2nd stash
stash@{2}: On master: my first stash

              -RG003A4 MINGW64 /g/git/git-demo1 (master)
$ git stash apply
On branch master
Your branch is up to date with 'origin/master'.

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:   test.txt

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

              -RG003A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: On master: my 3rd stash
stash@{1}: On master: my 2nd stash
stash@{2}: On master: my first stash
```

You can clearly see that even though the change was re-applied the entry is still present in the stash.

@mayank_ahuja

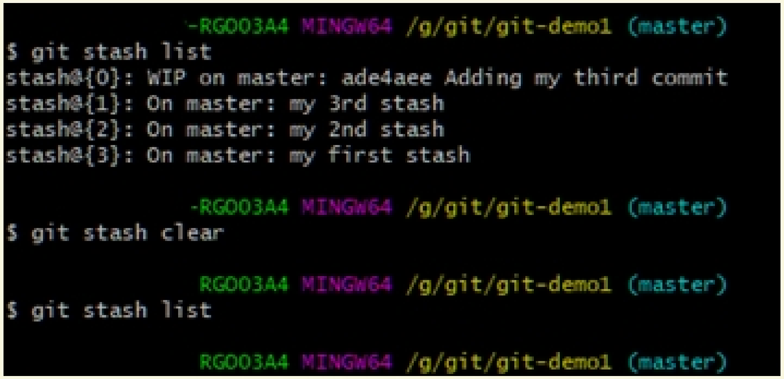
git stash apply <index>



```
              -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: WIP on master: ade4aee Adding my third commit
stash@{1}: On master: my 3rd stash
stash@{2}: On master: my 2nd stash
stash@{3}: On master: my first stash

               RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash apply 2
On branch master
Your branch is up to date with 'origin/master'.

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:   test.txt
```

You can specify a specific index from the list to be applied.

git stash clear



```
              -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list
stash@{0}: WIP on master: ade4aee Adding my third commit
stash@{1}: On master: my 3rd stash
stash@{2}: On master: my 2nd stash
stash@{3}: On master: my first stash

              -RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash clear

               RGO03A4 MINGW64 /g/git/git-demo1 (master)
$ git stash list

               RGO03A4 MINGW64 /g/git/git-demo1 (master)
```

Clears the stashed items.
List shows no entries.

@mayank_ahuja