GIT

Shristi Technology Labs

Contents

Version Control System(VCS)

- A version control system(VCS) helps software team to work together and manage changes to source code over time.

- It keeps track of every modification done to a file or set of files in a repository(a central place)

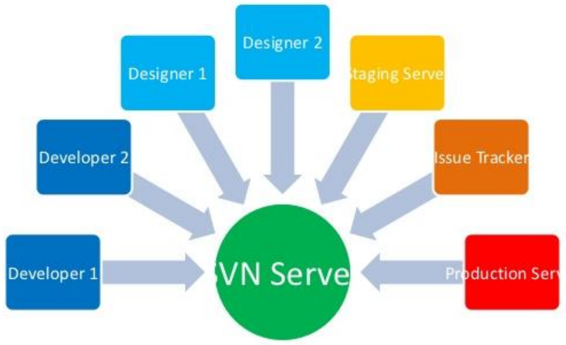
- It also allows to switch back to older versions

Types of VCS

- Centralized Version Control Sytsem
- Distributed Version Control Sytsem

CVCS

- Uses a central server to store all the files and manages different versions of the files
- A developer can checkout a version from the repository to their personal computer

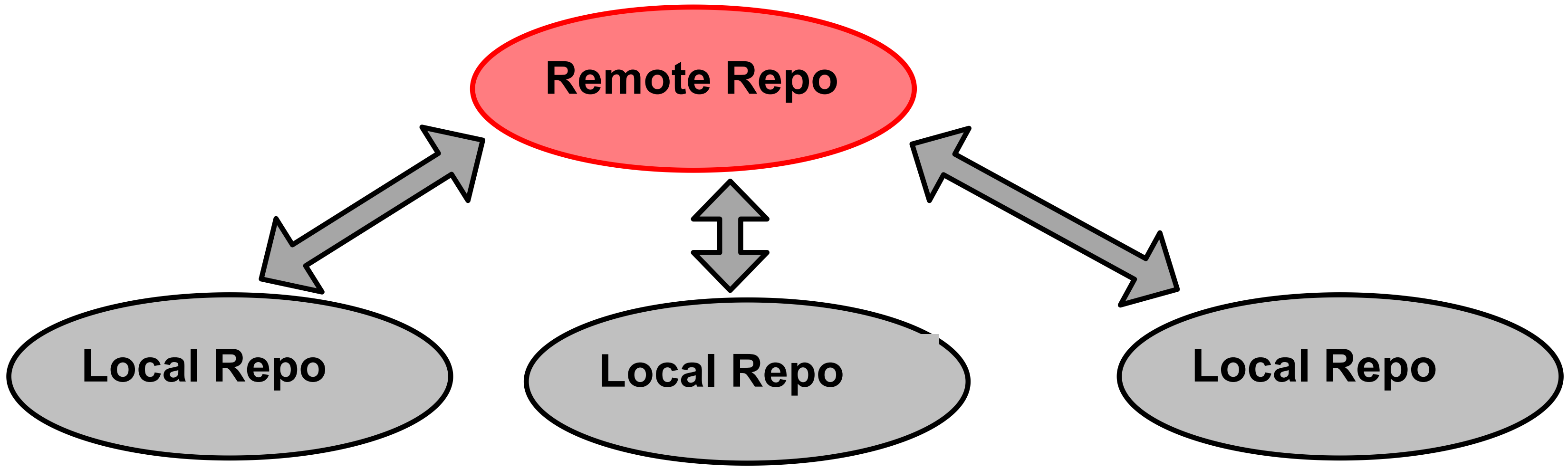


Drawbacks

- repository goes down
- Single point of failure
- Server gets corrupted, no backup, data lost

DVCS

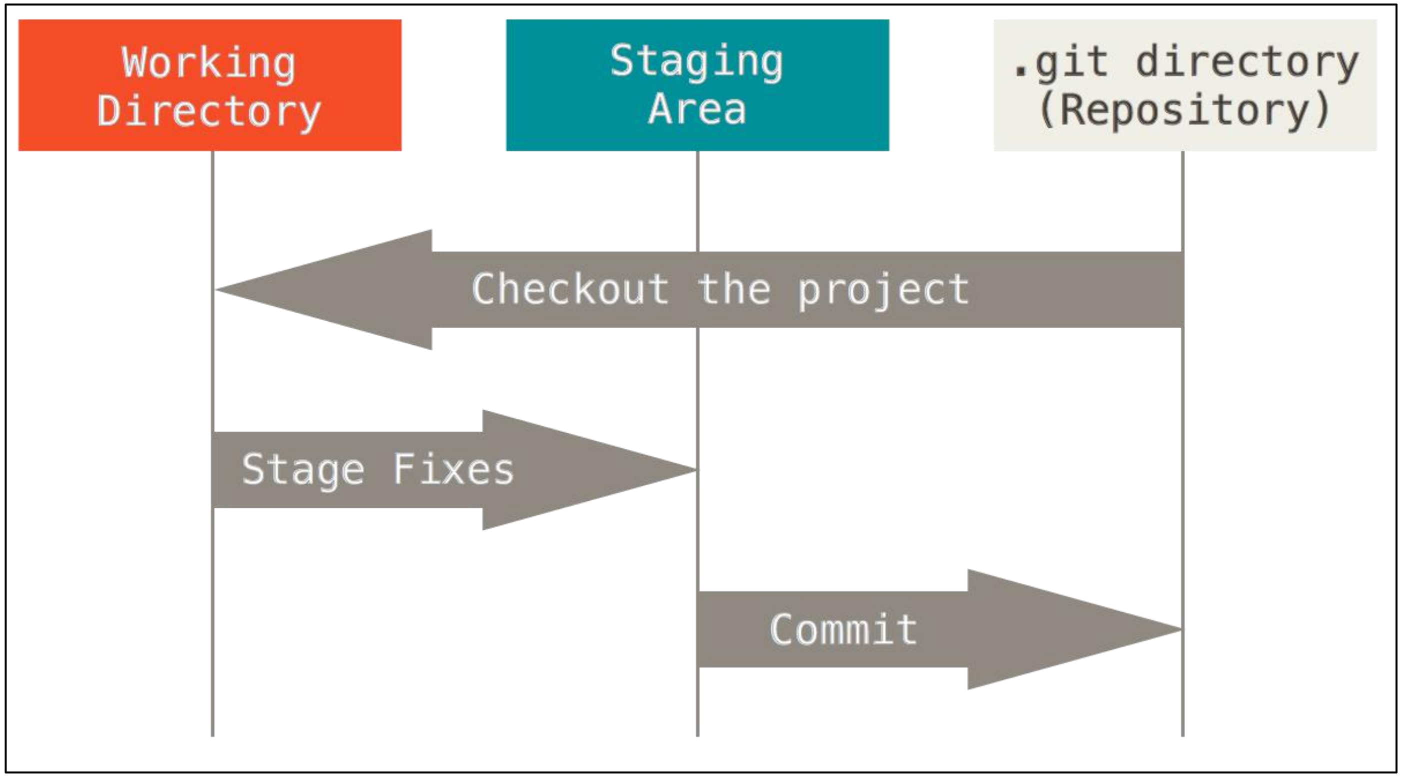
- User can clone the complete repository to his individual computer and get the local copy.
- Every clone has the full history of the collection of files
- Cloned repository has the same functionality as the original repository.
- Each repository can exchange versions of the files with other repositories using the repo running in the server



Introduction to GIT

- Git is a distributed revision control and source code management system.
- Git does not rely on the central server; - no need to interact with the remote server for every operation(add, remove, modify files).
- Facilitates collaborative changes to the files

Working, Staging and Repository



Git Repositories

Get a Git repository in one of two ways:

- Take a local directory that is currently not under version control, and turn it into a Git repository, or
- *Clone* an existing Git repository.

Now, the Git repository is on the local machine, ready for work.

Working Directory

- A local repository with the collection of files which originate from a certain version of the repository .
- It is just a checkout of one version of the repository with potential changes done by the user.
- The developer can change the files in the *working directory* by modifying existing files and by creating and removing files.
- A file in the working tree of a Git repository can have different states.

States in working directory

untracked

- The file is not tracked by the Git repository.
- The file is never staged nor committed.

staged

- staged to be included in the next commit

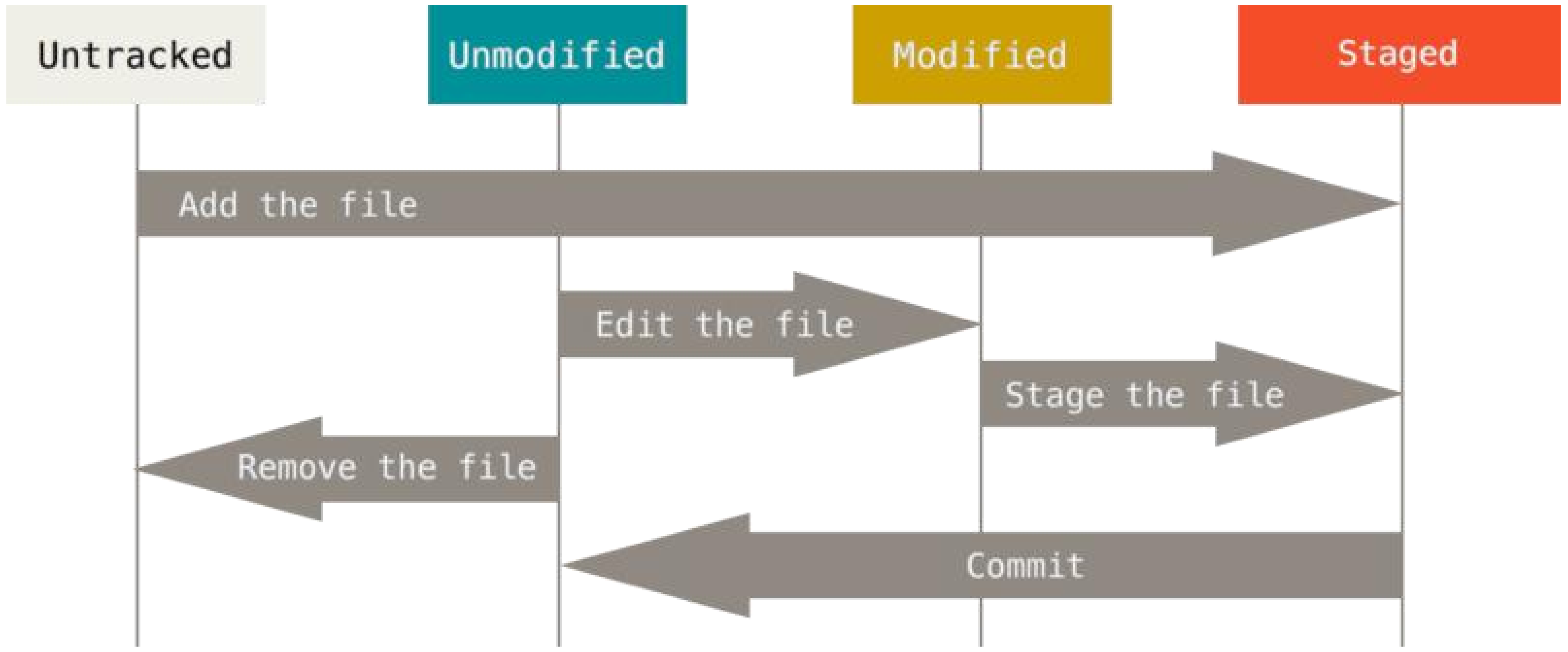
dirty / modified

- the file has changed but the change is not staged

tracked

- committed and not staged

Lifecycle of the status of files



Source:https://git-scm.com/book/en/v1/Getting-Started-Git-Basics

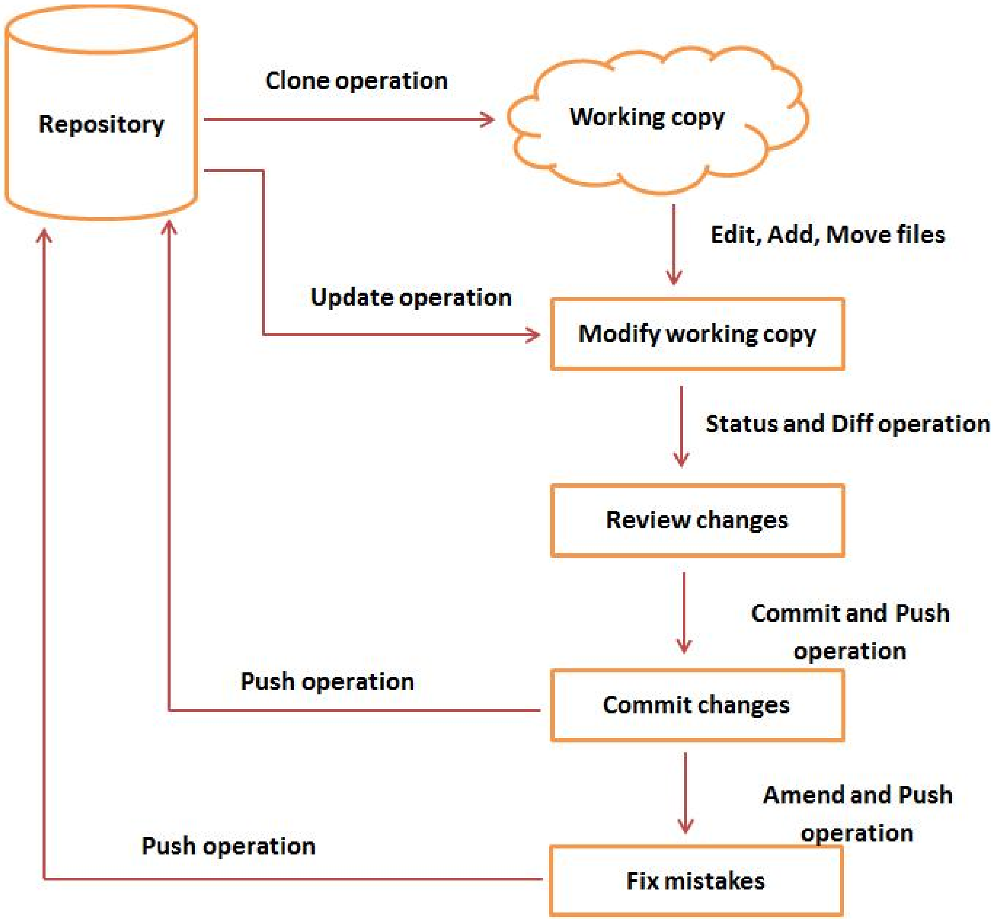
Staging

- The staging area contains a snapshot of the changes in the working tree (changed or new files) relevant for the next commit and stores their mode(file type)
- GIT checks for files in the staging area.
- Only the files in the staging area are committed
- Index is an alternative term for the staging area.

Workflow of GIT

- Clone the Git repository as a working copy.
- Modify the working copy by adding/editing files.
- Update the working copy by taking other developer's changes if needed
- Review the changes before commit.
- Commit changes.
- If good, then push the changes to the repository.
- After committing, if something is wrong, then correct the last commit and push the changes to the repository.

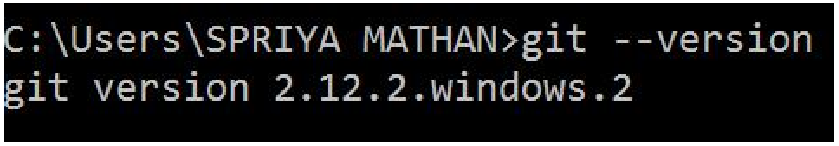
Workflow of GIT



Install GIT in windows

- Go to *http://git-scm.com/download/win* and the download will start automatically.
- This is a project called Git for Windows (also called msysGit)
- Select git-cheetah plugin.
- Complete the installation

- Add GIT installation path to the environment variables
- Open cmd and check git version as **`git --version`**

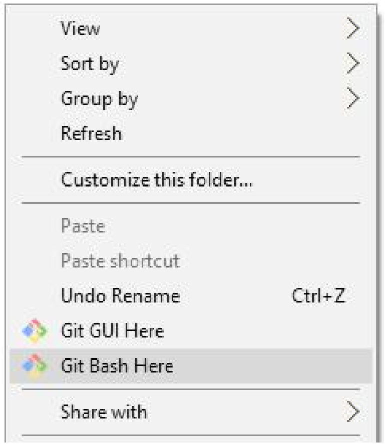


```
C:\Users\SPRIYA MATHAN>git --version
git version 2.12.2.windows.2
```

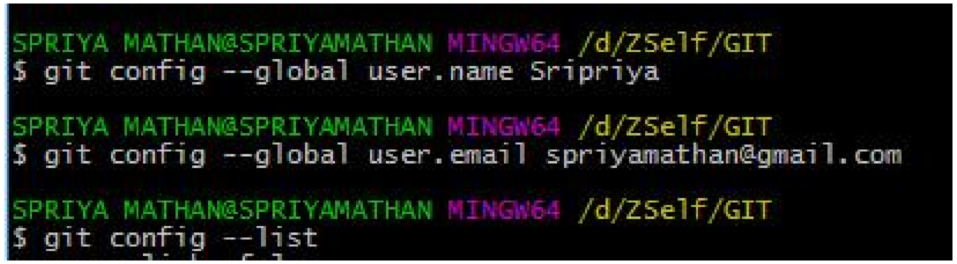
Configure GIT

- Open folder of your choice (D:/Zself/GIT in my case)
- Right click inside the folder and select **Git Bash Here**
- GIT–BASH is opened
- Configure the username, email

```
git config --global user.name "<your name>"
git config --global user.email "<your mailid>"
```



- Use `git config --list` command to list the settings of Git



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT
$ git config --global user.name Sripriya

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT
$ git config --global user.email spriyamathan@gmail.com

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT
$ git config --list
```

GIT Help

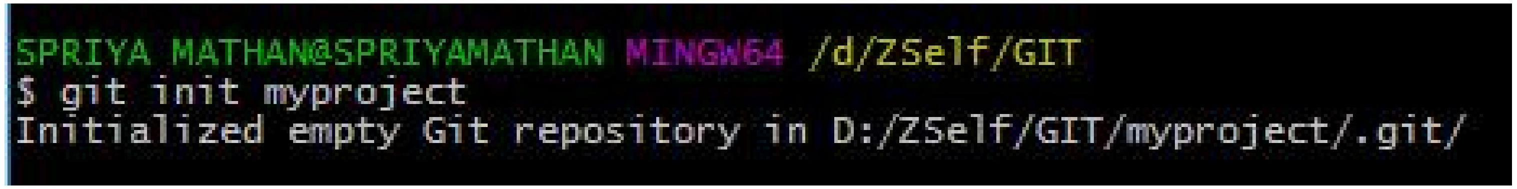
- To get help while using GIT, use
  - `git help <verb>`
  - `git <verb> --help`

as

```
git help config
```

Create a local repository

- To create a local repository. Type `git init myproject`
- This creates an empty Git repository - a **.git** directory inside myproject folder with subdirectories for objects, refs/heads, refs/tags
- An initial HEAD file that references the HEAD of the master branch is also created.

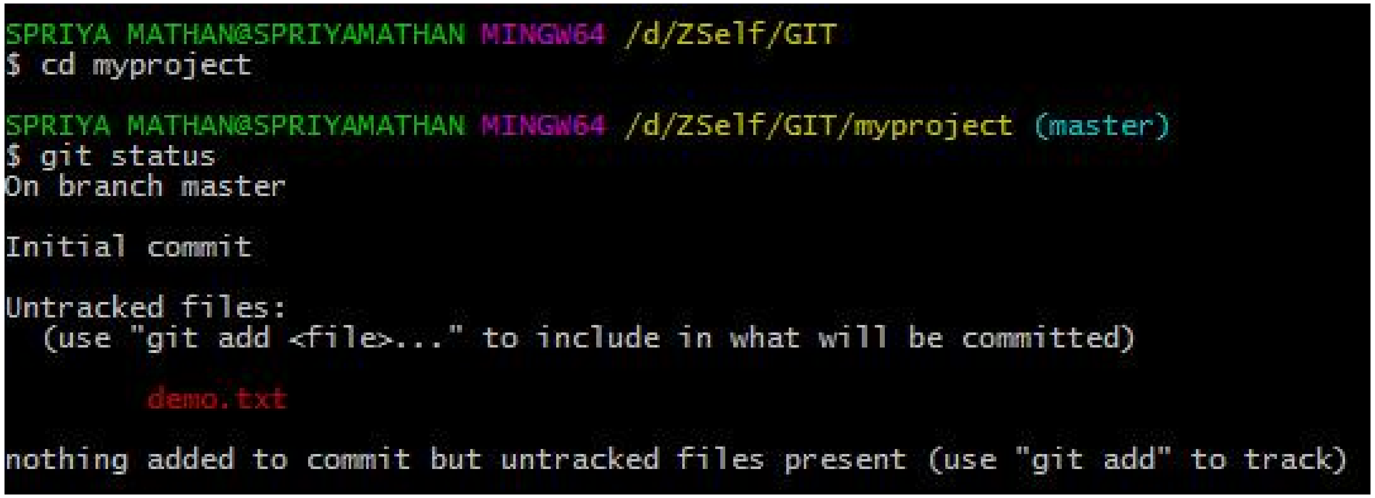


```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT
$ git init myproject
Initialized empty Git repository in D:/ZSelf/GIT/myproject/.git/
```

Create a file - untracked

- Create a file **demo.txt** inside **myproject** directory
- Move into myproject folder and check the status using

  `git status`
- This will show as *untracked files*



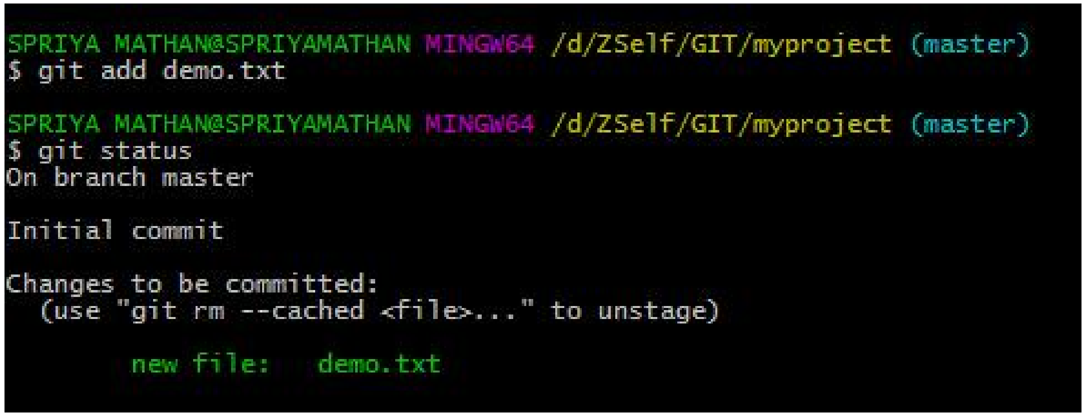
```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT
$ cd myproject

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git status
On branch master

Initial commit

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

        demo.txt

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

Add file to repo - staged

- To add the file to the repository use

  `git add <path to file>`
- To check the files in the folder use `ls`
- Check the status using

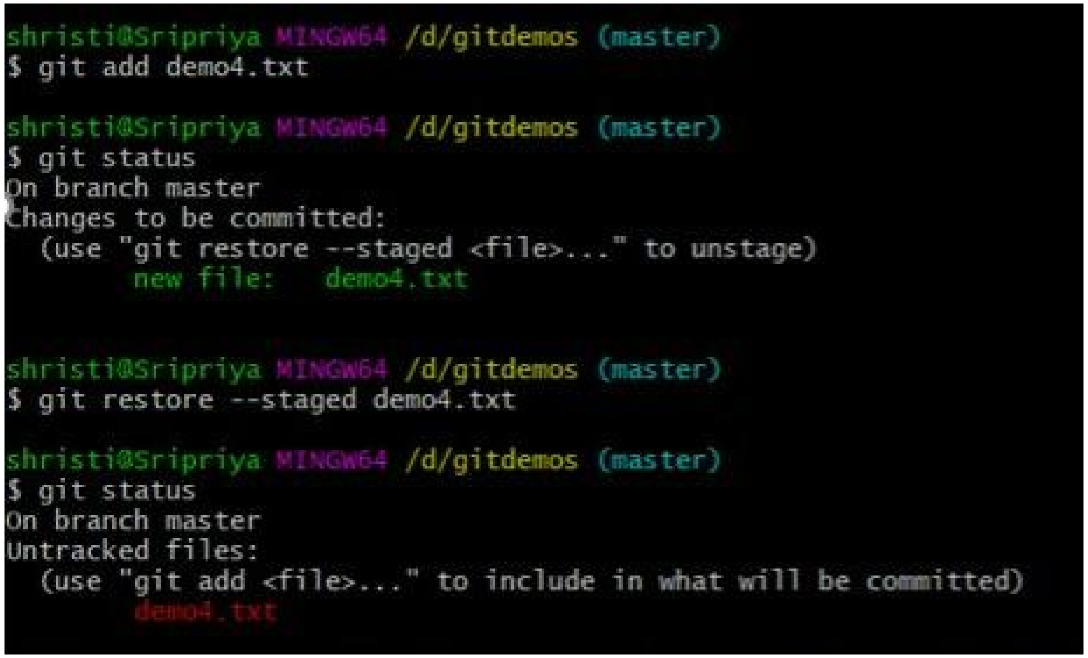
  `git status`
- This will show as *staged files*



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git add demo.txt

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git status
On branch master

Initial commit

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

        new file:   demo.txt
```

From staged to unstaged – using restore (from 2.23.0)

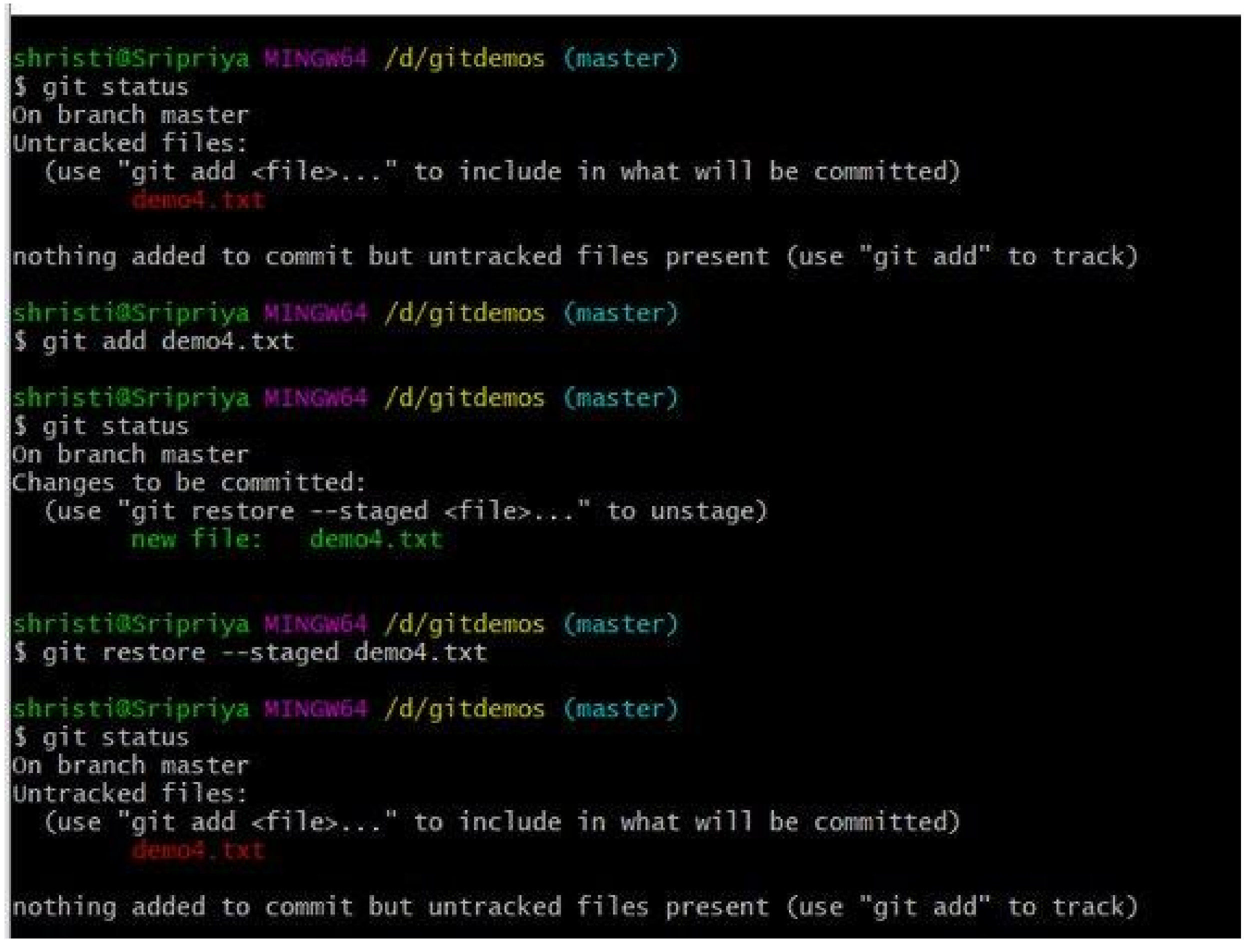
```
git restore --staged demo4.txt
```



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git add demo4.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   demo4.txt


shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git restore --staged demo4.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo4.txt
```

```
git add demo4.txt
git status
git restore --staged demo4.txt
git status
```

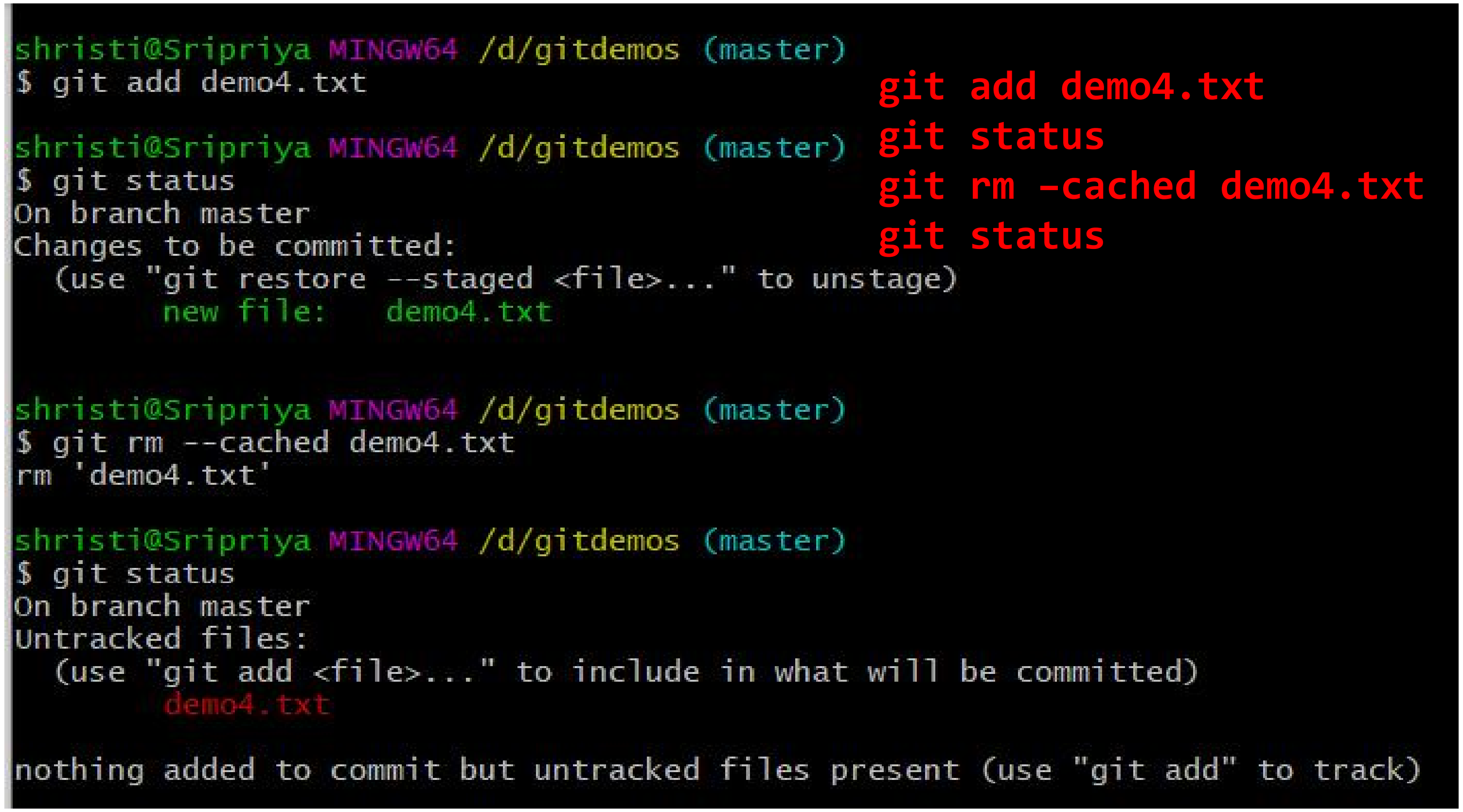
Example



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo4.txt

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

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git add demo4.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   demo4.txt


shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git restore --staged demo4.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo4.txt

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

From staged to untracked – using rm

To remove file from git



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git add demo4.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   demo4.txt


shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git rm --cached demo4.txt
rm 'demo4.txt'

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        demo4.txt

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

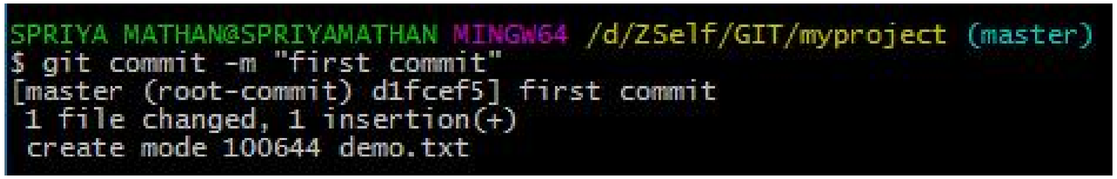
```
git add demo4.txt
git status
git rm -cached demo4.txt
git status
```

Shristi Technology Labs

Store file to repo - committed

- Stores the current contents along with a log message from the user describing the changes.

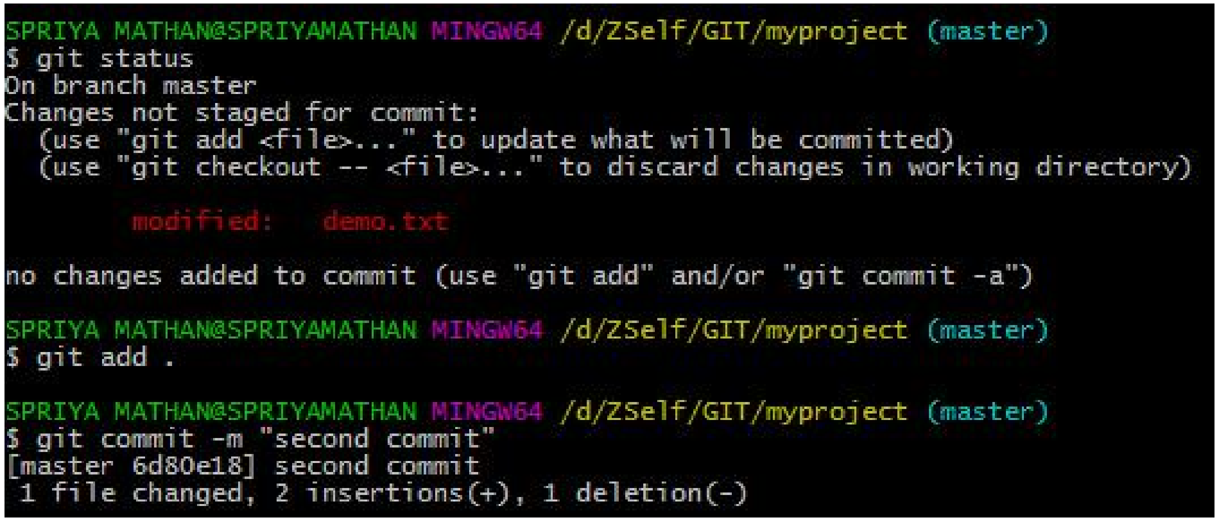
  `git commit -m <message>`
- Now the file is committed to the repository



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git commit -m "first commit"
[master (root-commit) d1fcef5] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 demo.txt
```

Stage modified files

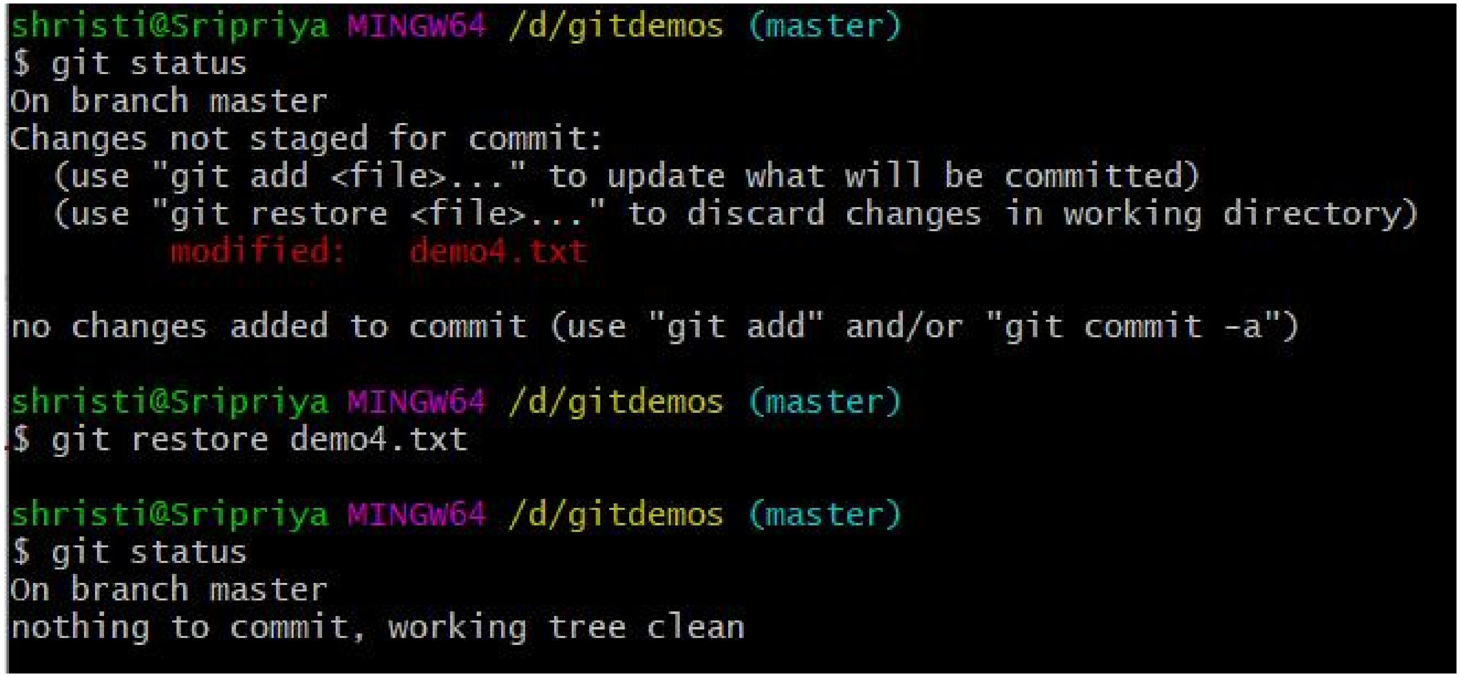
- Check the status using **git status**
- This will show the files modified
- Add the files using **git add <filename>**
- Commit using **git commit**



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (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:   demo.txt

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

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git add .

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git commit -m "second commit"
[master 6d80e18] second commit
 1 file changed, 2 insertions(+), 1 deletion(-)
```

To discard modified changes – using restore

git restore demo4.txt

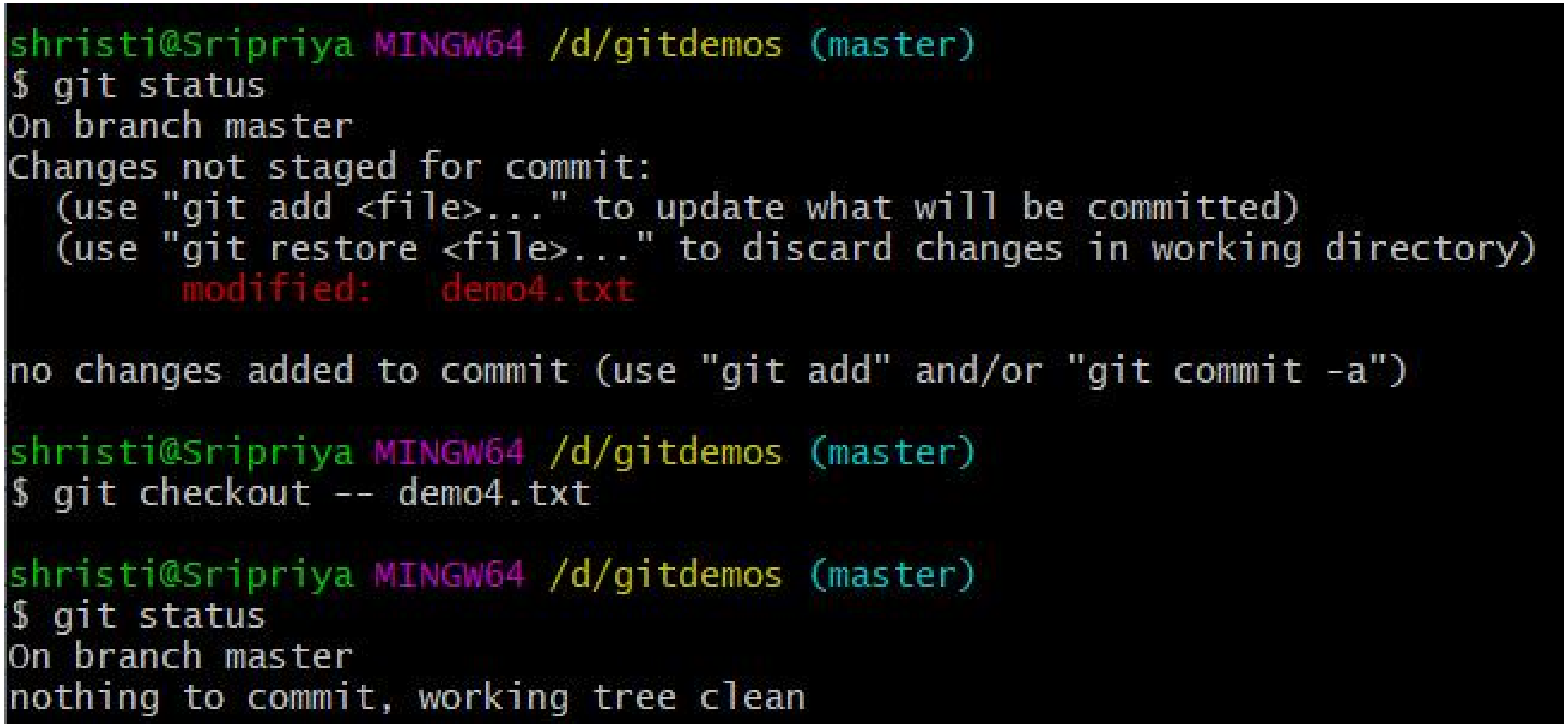


```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   demo4.txt

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

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git restore demo4.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Shriſti Technology Labs

To discard modified changes – using checkout

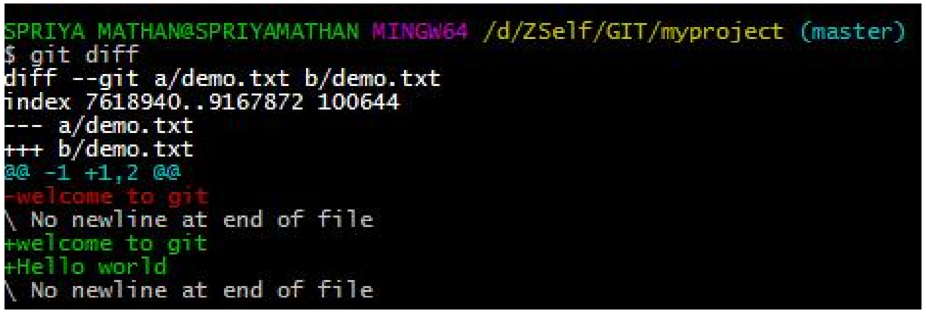
git checkout -- demo4.txt



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch 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:   demo4.txt

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

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git checkout -- demo4.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch master
nothing to commit, working tree clean
```

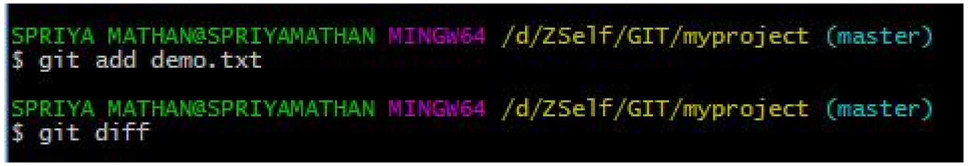
Review changes - diff

- Add new content to the **demo.txt** file
- To view changes between commits, use `git diff`
- This shows only the changes that are unstaged(see before committing)



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git diff
diff --git a/demo.txt b/demo.txt
index 7618940..9167872 100644
--- a/demo.txt
+++ b/demo.txt
@@ -1 +1,2 @@
-welcome to git
\ No newline at end of file
+welcome to git
+Hello world
\ No newline at end of file
```

- Stage the files using `git add`
- Then use `git diff`
- No changes will be seen

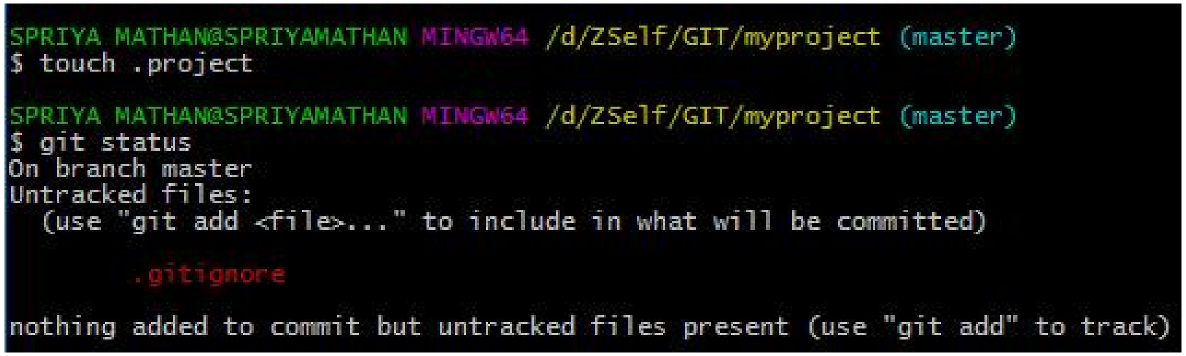


```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git add demo.txt

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git diff
```

Ignoring Files

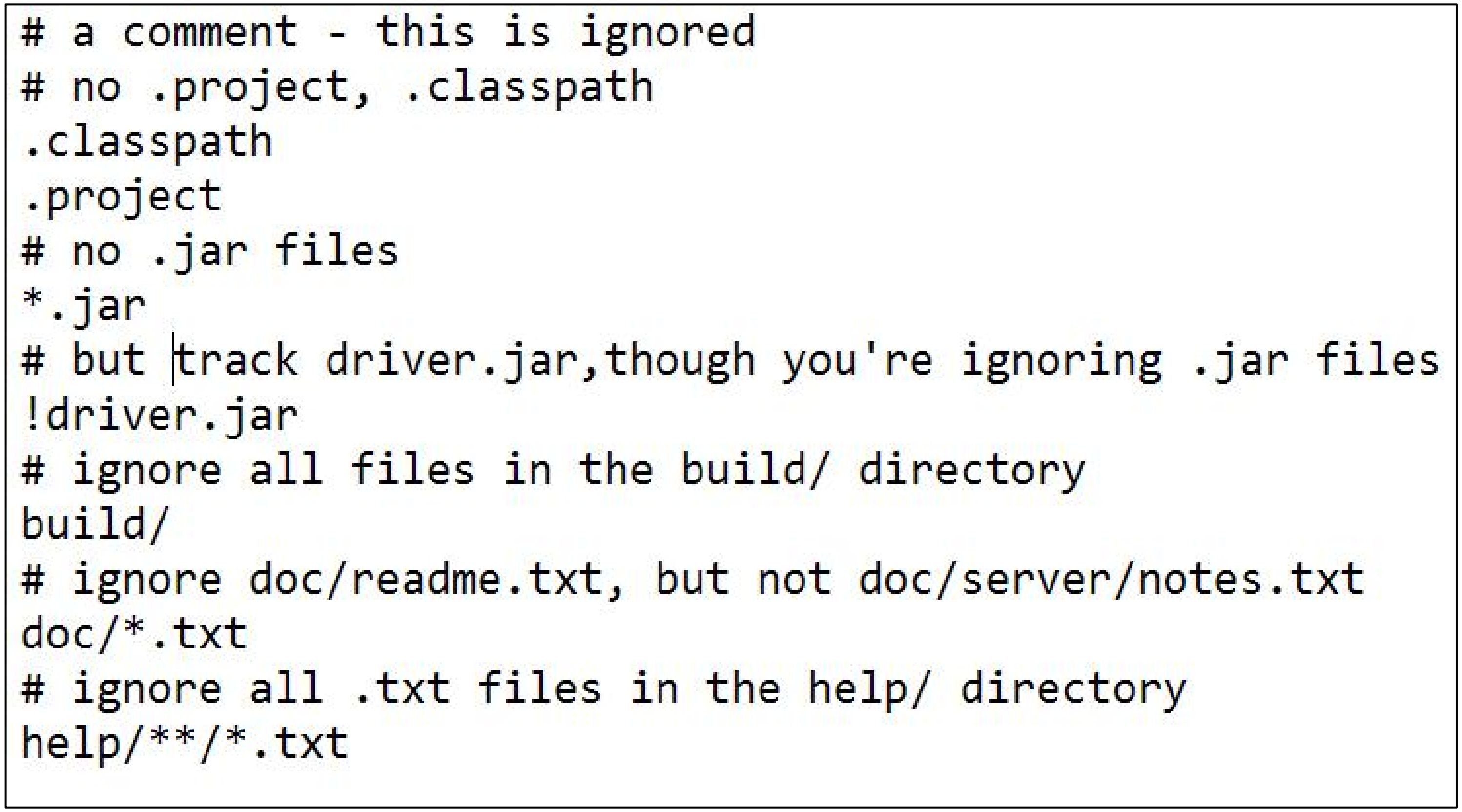
- Few files like log files, automatically generated files should not be added or tracked and should be ignored.
- In git bash type **`touch .gitignore.`** A new file gets created.
- This file is used to ignore the files that should not be committed to the repository.

- Create a new file in git bash using **`touch .project.`**
- Add this file to **.gitignore**



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ touch .project

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

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

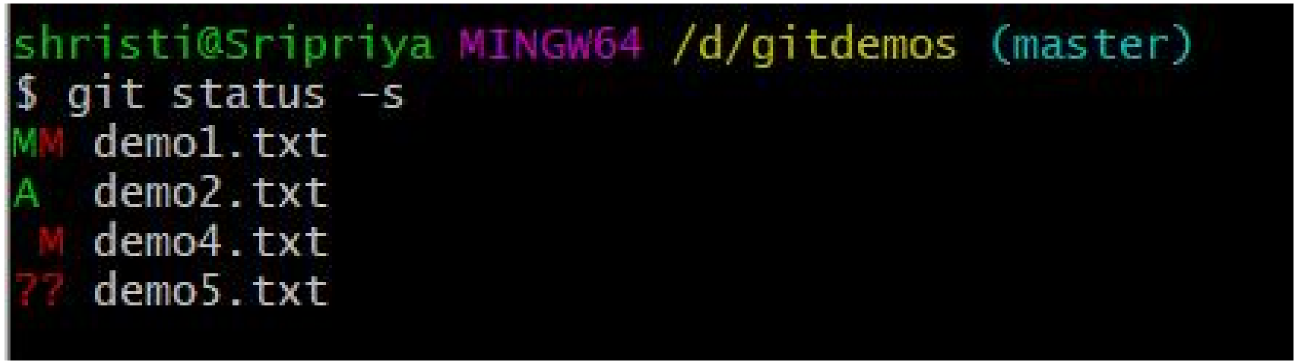
.gitignore file



```
# a comment - this is ignored
# no .project, .classpath
.classpath
.project
# no .jar files
*.jar
# but track driver.jar,though you're ignoring .jar files
!driver.jar
# ignore all files in the build/ directory
build/
# ignore doc/readme.txt, but not doc/server/notes.txt
doc/*.txt
# ignore all .txt files in the help/ directory
help/**/*.txt
```

git status

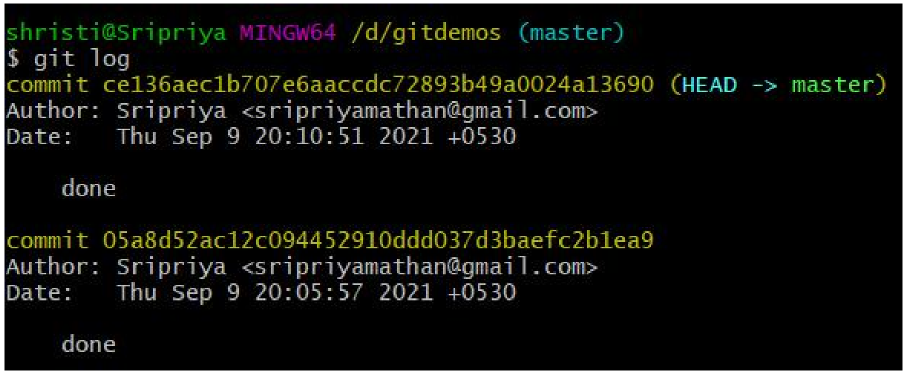
- ?? – untracked
- A – staged
- MM – modified, staged and modified
- M – modified



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status -s
MM demo1.txt
A  demo2.txt
 M demo4.txt
?? demo5.txt
```

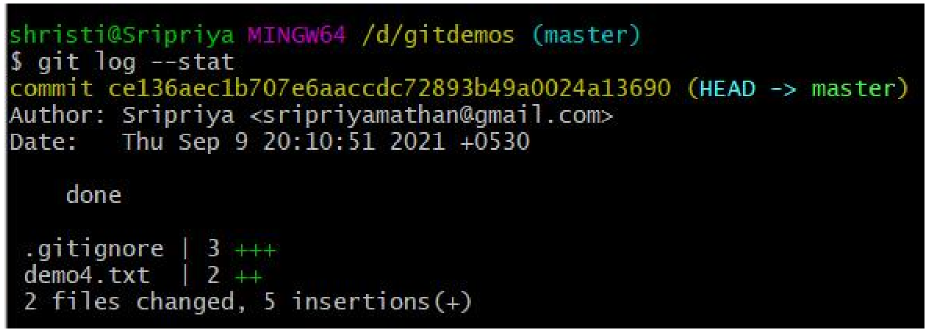
history

git log



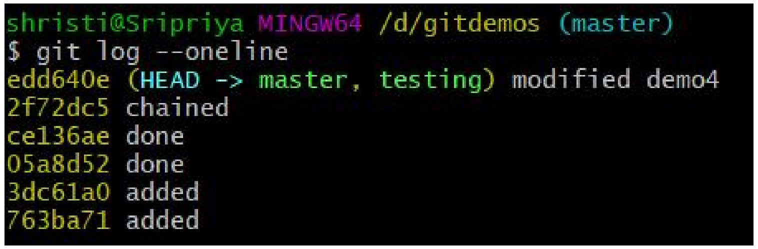
```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git log
commit ce136aec1b707e6aaccdc72893b49a0024a13690 (HEAD -> master)
Author: Sripriya <sripriyamathan@gmail.com>
Date:   Thu Sep 9 20:10:51 2021 +0530

    done

commit 05a8d52ac12c094452910ddd037d3baefc2b1ea9
Author: Sripriya <sripriyamathan@gmail.com>
Date:   Thu Sep 9 20:05:57 2021 +0530

    done
```

git log -stat



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git log --stat
commit ce136aec1b707e6aaccdc72893b49a0024a13690 (HEAD -> master)
Author: Sripriya <sripriyamathan@gmail.com>
Date:   Thu Sep 9 20:10:51 2021 +0530

    done

 .gitignore | 3 +++
 demo4.txt  | 2 ++
 2 files changed, 5 insertions(+)
```

git log --oneline

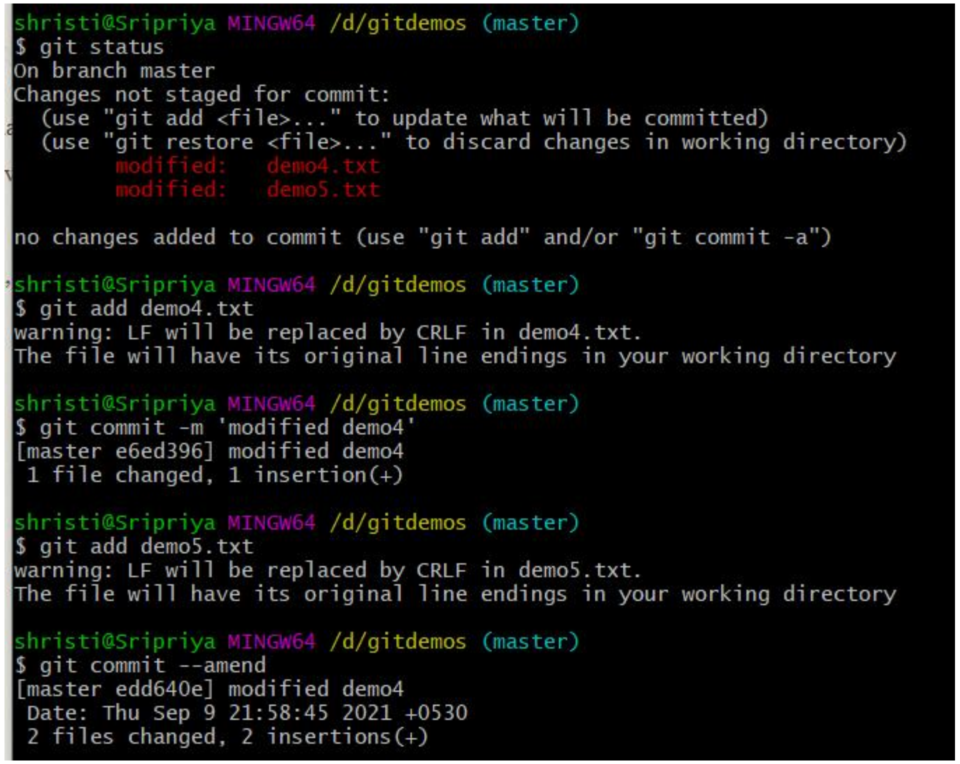


```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git log --oneline
edd640e (HEAD -> master, testing) modified demo4
2f72dc5 chained
ce136ae done
05a8d52 done
3dc61a0 added
763ba71 added
```

git commit --amend

Usage

- Done a commit without adding a file
- **`git commit -m 'changed'`**
- Then use **`git add`**
- Use **`git commit -amend`**
- This commit will have the same message as the previous commit and also can edit it
- So thers is a single commit — the second commit replaces the results of the first.



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git status
On branch 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:   demo4.txt
        modified:   demo5.txt

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

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git add demo4.txt
warning: LF will be replaced by CRLF in demo4.txt.
The file will have its original line endings in your working directory

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git commit -m 'modified demo4'
[master e6ed396] modified demo4
 1 file changed, 1 insertion(+)

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git add demo5.txt
warning: LF will be replaced by CRLF in demo5.txt.
The file will have its original line endings in your working directory

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git commit --amend
[master edd640e] modified demo4
 Date: Thu Sep 9 21:58:45 2021 +0530
 2 files changed, 2 insertions(+)
```

Commit

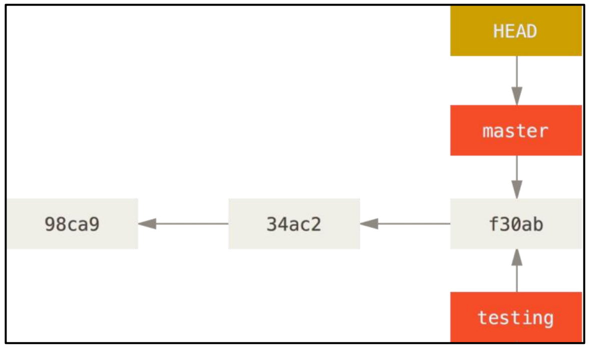
- Commit holds the current state of the repository.
- A commit is also named by **SHA1** hash.
- Like a node of the linked list.
- Every commit object has a pointer to the parent commit object.
- From a given commit, you can traverse back by looking at the parent pointer to view the history of the commit.
- If a commit has multiple parent commits, then that particular commit has been created by merging two branches.

Branches

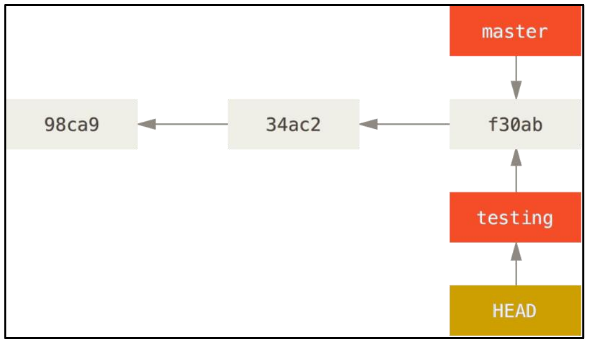
- Are used to create another line of development.
- By default, Git has a master branch
- A branch is created to work on a new feature .
- Once the feature is done, merge it back with the master branch and delete the branch.
- Every branch is referenced by HEAD, which points to the latest commit in the branch.
- Whenever you make a commit, HEAD is updated with the latest commit..

Git Branching

- **git branch testing**



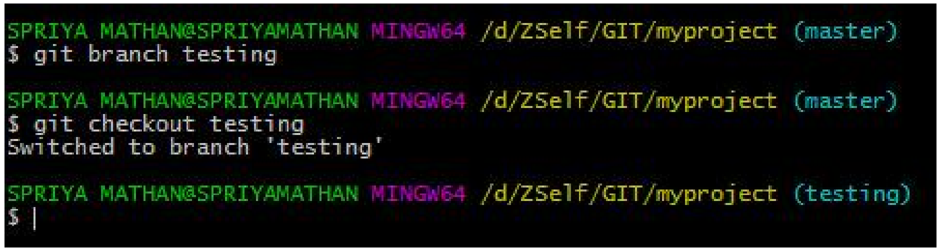
- **git checkout testing**



Create branch

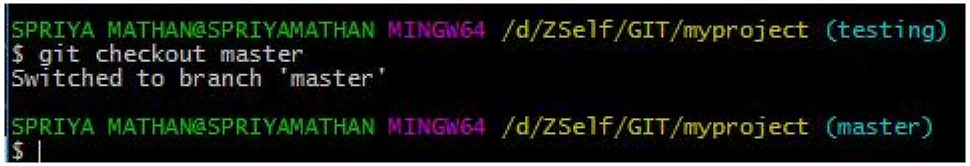
- To create a branch use `git branch testing`
- To checkout to this branch use `git checkout testing`
- Together it can be done as

  `git checkout –b testing`



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git branch testing

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git checkout testing
Switched to branch 'testing'

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (testing)
$ |
```

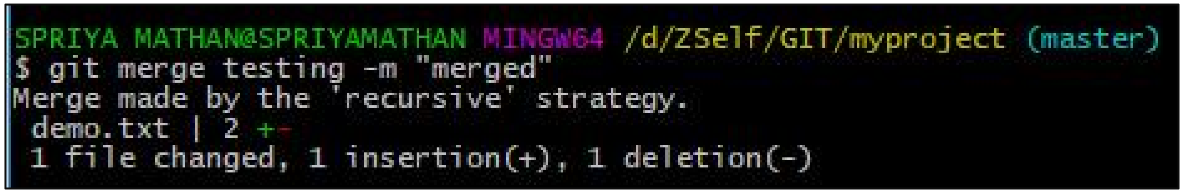
- Add a file to this branch and commit it
- To switch to master use `git checkout master`



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (testing)
$ git checkout master
Switched to branch 'master'

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ |
```

merge

- To merge the branch to the master, use **`git merge`**



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GIT/myproject (master)
$ git merge testing -m "merged"
Merge made by the 'recursive' strategy.
 demo.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

- Use **`git mergetool`** in case of conflicts

Vim commands

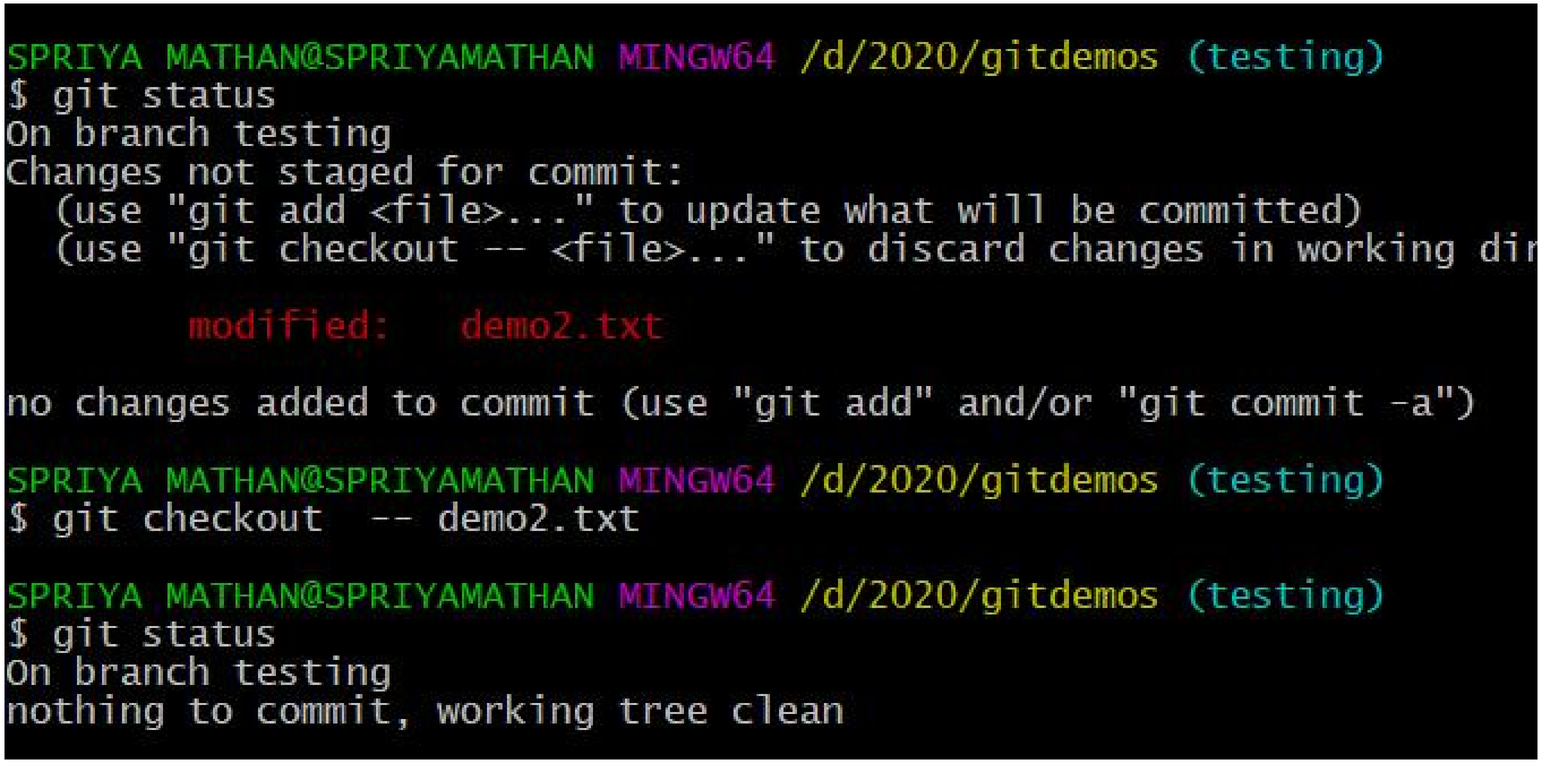
:diffget LO

:diffget BA

:diffget RE

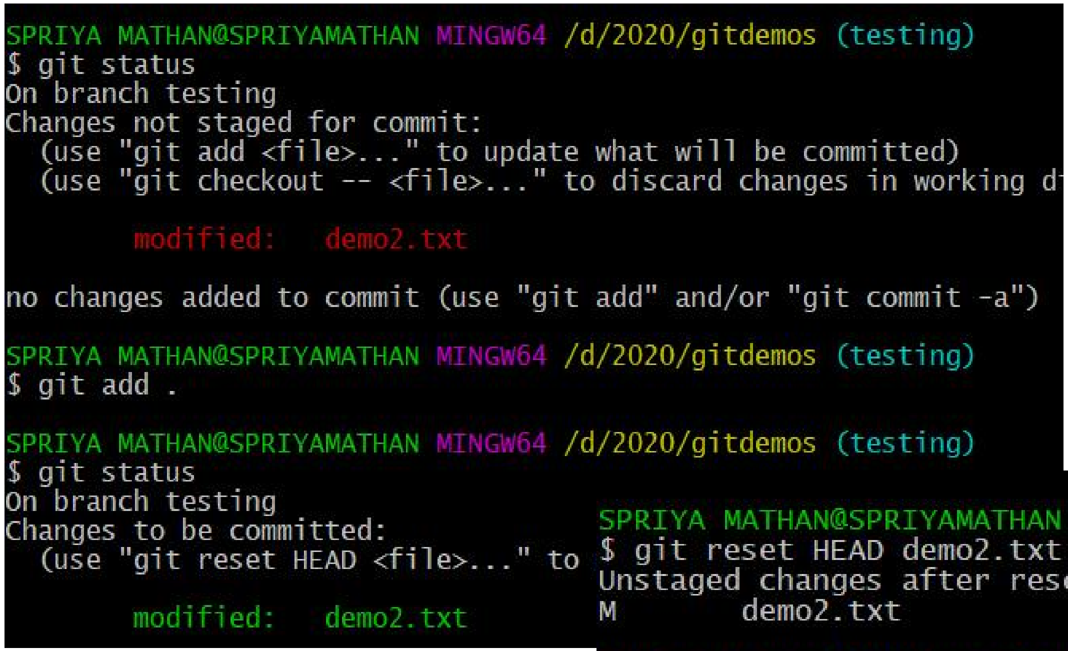
To switch between windows use ctrl W and save using :wqa

Unmodifying a Modified File - **git checkout -- <file>**

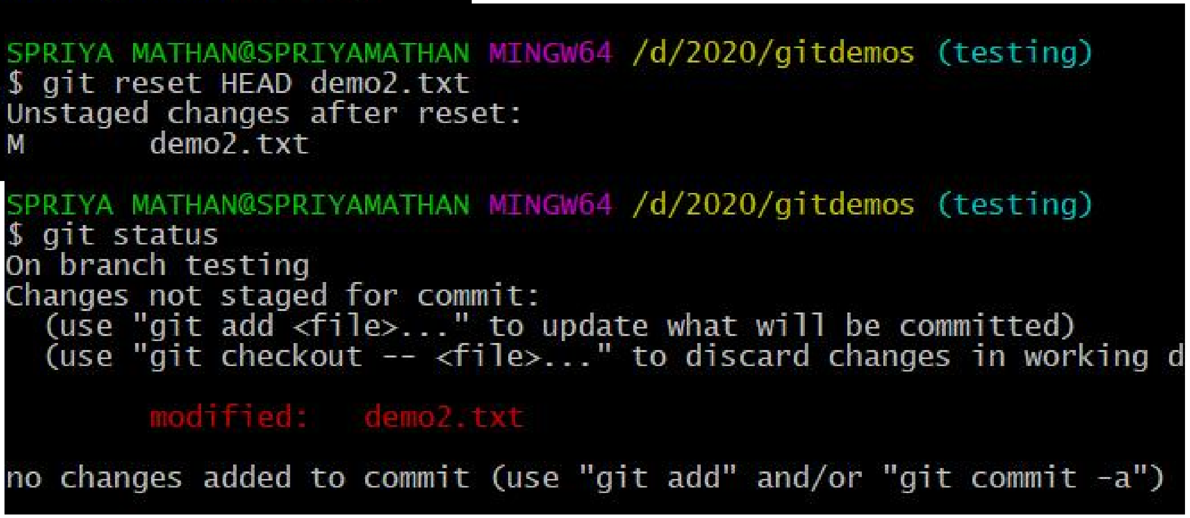


```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git status
On branch testing
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working dir

        modified:   demo2.txt

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

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git checkout  -- demo2.txt

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git status
On branch testing
nothing to commit, working tree clean
```

reset - Is for unstaging a staged file



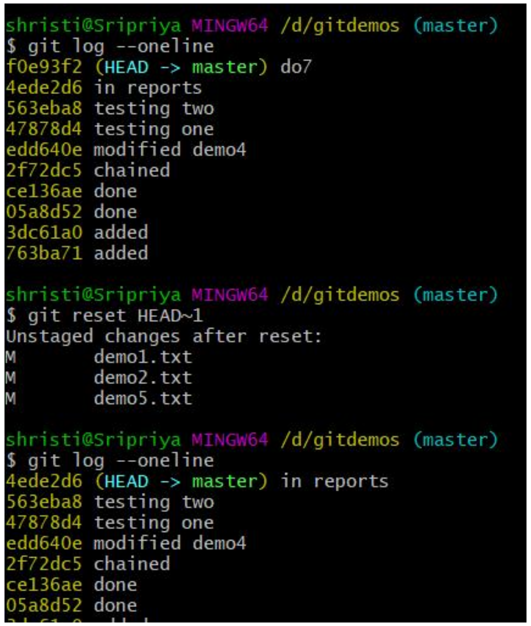
```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git status
On branch testing
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working d

        modified:   demo2.txt

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

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git add .

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git status
On branch testing
Changes to be committed:
  (use "git reset HEAD <file>..." to

        modified:   demo2.txt
```



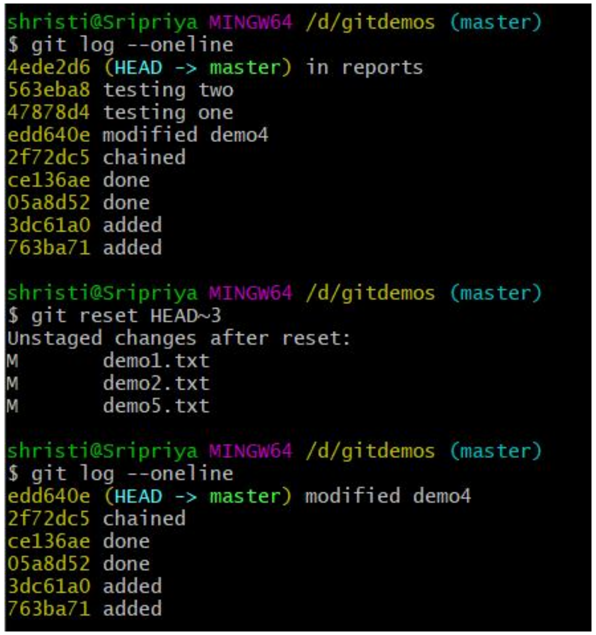
```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git reset HEAD demo2.txt
Unstaged changes after reset:
M       demo2.txt

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/2020/gitdemos (testing)
$ git status
On branch testing
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working d

        modified:   demo2.txt

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

Reset to remove local commits

- Using reset, can remove recent one or two or more commits
- All the files are moved to untracked/ working area
- **`git reset HEAD~1`**
- To move forward again use **`git reset HEAD@{1}`**



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git log --oneline
f0e93f2 (HEAD -> master) do7
4ede2d6 in reports
563eba8 testing two
47878d4 testing one
edd640e modified demo4
2f72dc5 chained
ce136ae done
05a8d52 done
3dc61a0 added
763ba71 added

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git reset HEAD~1
Unstaged changes after reset:
M       demo1.txt
M       demo2.txt
M       demo5.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git log --oneline
4ede2d6 (HEAD -> master) in reports
563eba8 testing two
47878d4 testing one
edd640e modified demo4
2f72dc5 chained
ce136ae done
05a8d52 done
```



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git log --oneline
4ede2d6 (HEAD -> master) in reports
563eba8 testing two
47878d4 testing one
edd640e modified demo4
2f72dc5 chained
ce136ae done
05a8d52 done
3dc61a0 added
763ba71 added

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git reset HEAD~3
Unstaged changes after reset:
M       demo1.txt
M       demo2.txt
M       demo5.txt

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git log --oneline
edd640e (HEAD -> master) modified demo4
2f72dc5 chained
ce136ae done
05a8d52 done
3dc61a0 added
763ba71 added
```

reset and revert

reset

--mixed : takes all the changes and keeps it in working area(untracked)

--soft : keeps the changes in staging area

--hard : changes are removed completely

```
git reset --mixed HEAD~1
```

revert

- To undo the changes in the remote repository and creates a new commit. Old commit is still in history

```
git revert e5456a<commitid>
```

Get the remote url

```
git config --get remote.origin.url
```

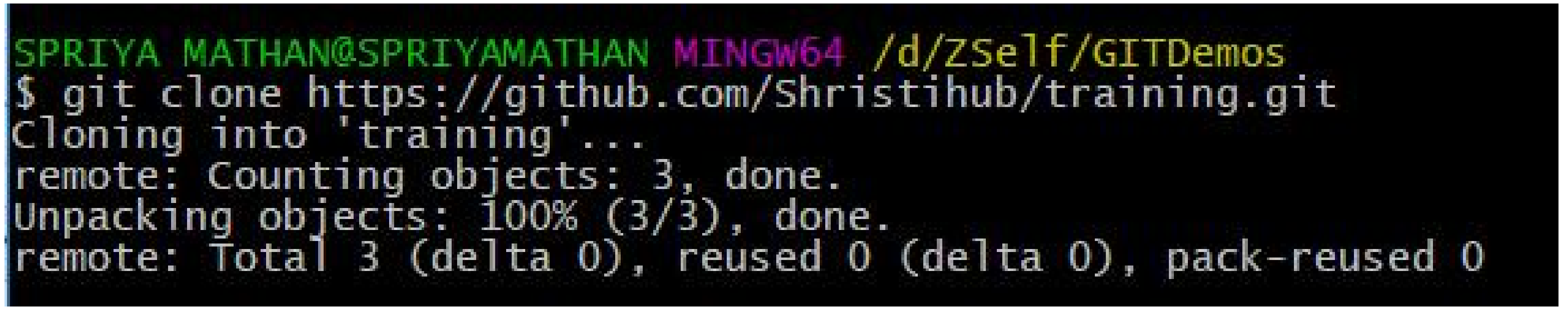
Working with remotes

- Remote repositories are versions of the project that are hosted on the Internet or in some network
- They have either read-only or read/write option.
- Collaborating with others involves managing these remote repositories – add, remove repositories, branches.
- To share data with others, push data into the repository and pull data from repository

Clone a remote repository

- Clone operation creates the instance of the repository.
- Clone checks out the working copy, and mirrors the complete repository.
- Users can perform many operations with this local repository.
- The only time networking gets involved is when the repository instances are being synchronized.

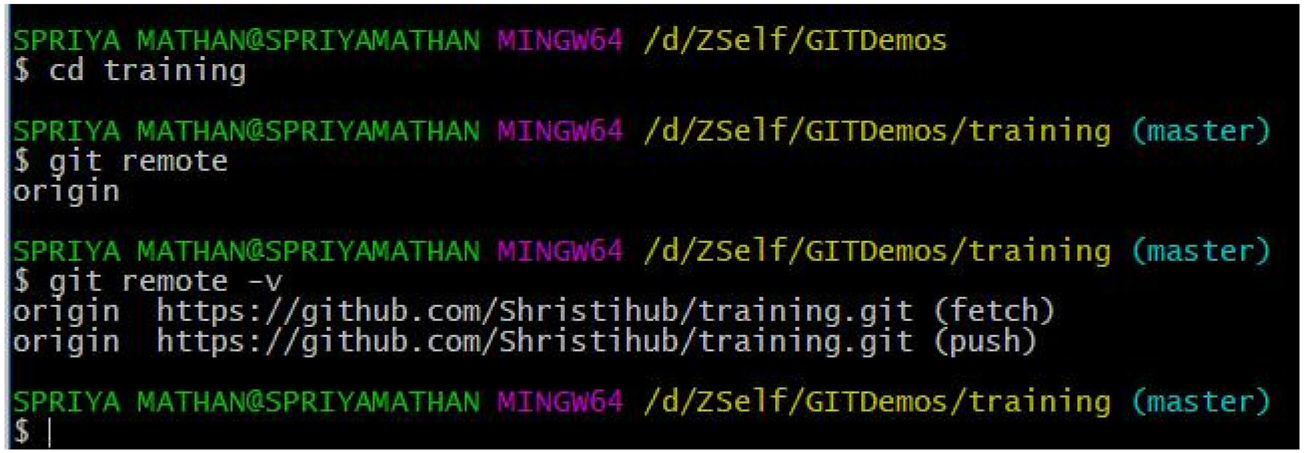
`git clone <url>`



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos
$ git clone https://github.com/Shristihub/training.git
Cloning into 'training'...
remote: Counting objects: 3, done.
Unpacking objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

Get the remotes

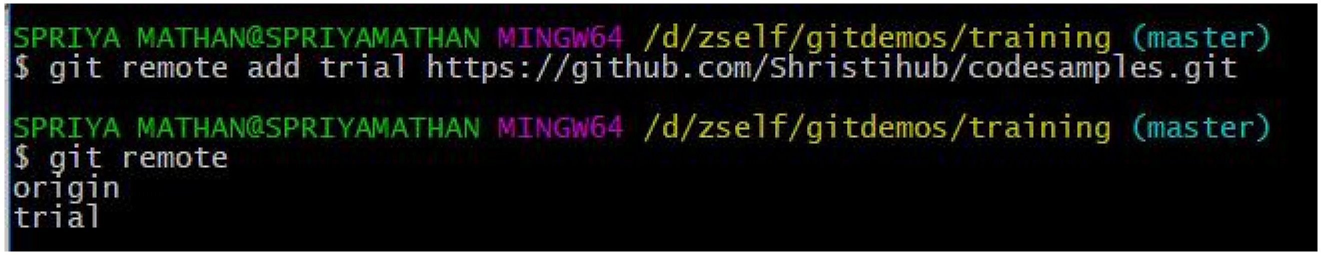
- To see the remote servers that are configured, run the command
  git remote
- It lists the short names of each remote handle.



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos
$ cd training

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ git remote
origin

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ git remote -v
origin  https://github.com/Shristihub/training.git (fetch)
origin  https://github.com/Shristihub/training.git (push)

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ |
```

Add a remote

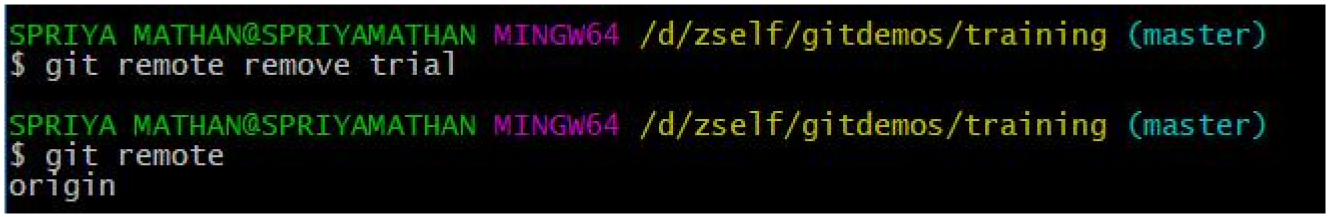
- To add a new remote Git repository as a shortname you can reference easily, run the command
  git remote add <shortname> <url>



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/zself/gitdemos/training (master)
$ git remote add trial https://github.com/Shristihub/codesamples.git

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/zself/gitdemos/training (master)
$ git remote
origin
trial
```

Remove a remote

- To remove a remote repository run the command
  `git remote remove <shortname>`



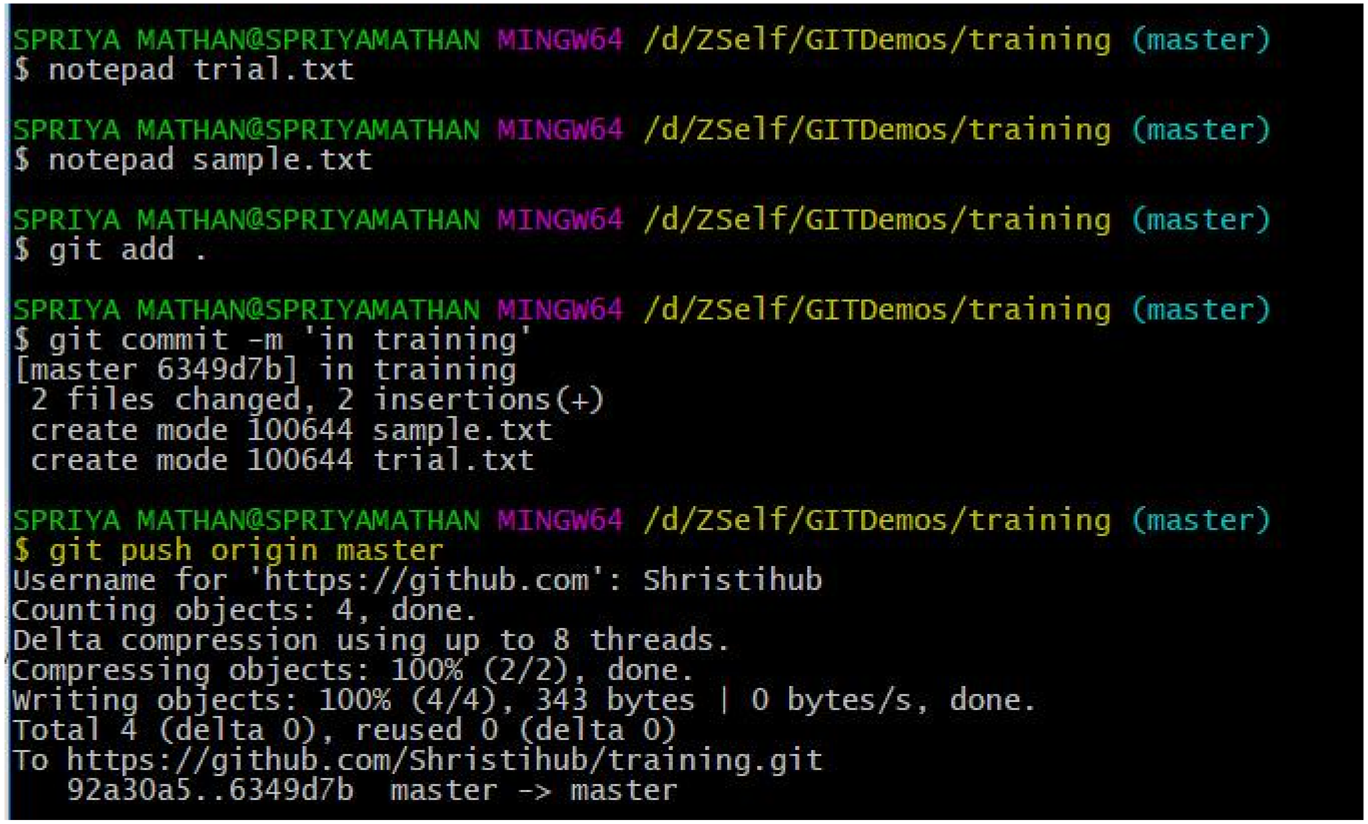
```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/zself/gitdemos/training (master)
$ git remote remove trial

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/zself/gitdemos/training (master)
$ git remote
origin
```

Push Operation

- copies changes from a local repository instance to a remote one.
- Used to store the changes permanently into the Git repository.

```
git push <remote> <branch>
git push origin master
```

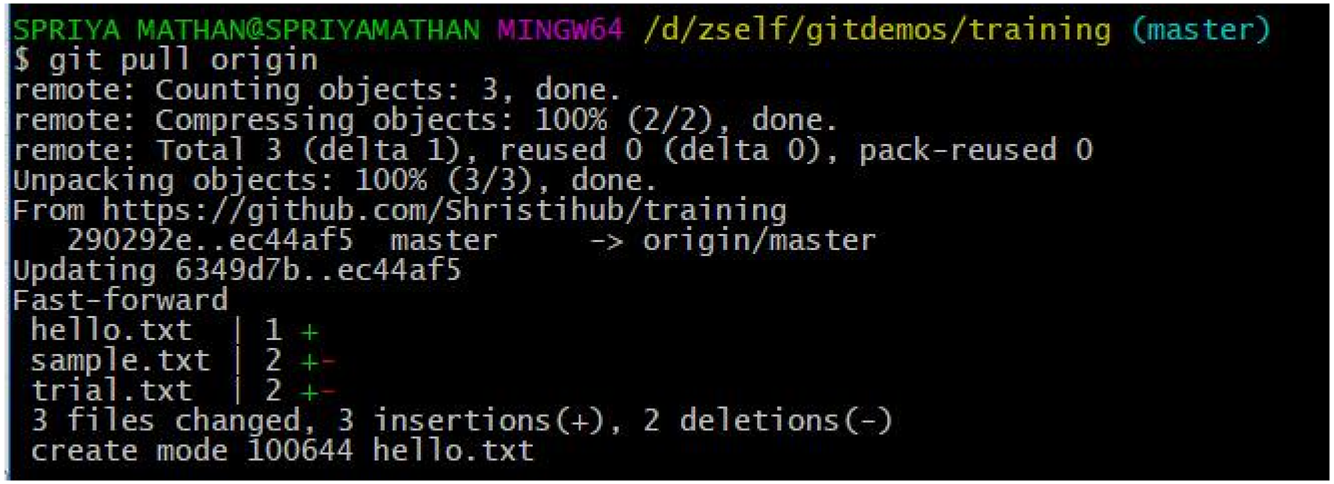
Example - Push



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ notepad trial.txt

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ notepad sample.txt

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ git add .

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ git commit -m 'in training'
[master 6349d7b] in training
 2 files changed, 2 insertions(+)
 create mode 100644 sample.txt
 create mode 100644 trial.txt

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/training (master)
$ git push origin master
Username for 'https://github.com': Shristihub
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 343 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/Shristihub/training.git
   92a30a5..6349d7b  master -> master
```

Pull Operation

- copies the changes from a remote repository instance to a local one.
- is used for synchronization between two repository instances.



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/zself/gitdemos/training (master)
$ git pull origin
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Shristihub/training
   290292e..ec44af5  master     -> origin/master
Updating 6349d7b..ec44af5
Fast-forward
 hello.txt  | 1 +
 sample.txt | 2 +-
 trial.txt  | 2 +-
 3 files changed, 3 insertions(+), 2 deletions(-)
 create mode 100644 hello.txt
```

reset

Modified to unmodified

- `git restore demo1.txt`

Staged to unstaged (modified/ untracked)

- `git restore --staged demo1.txt`

Committed to unmodified/untracked

- `git reset --mixed demo1.txt`
- `git reset demo1.txt`

Committed to staged

- `git reset --soft demo1.txt`

Git Stash

- This command helps to switch to another branch without committing in the current branch.
- Use the command **git stash** to save the changes

- Stashing takes the dirty state of the working directory
- The modified tracked files and staged changes are saved on a stack of unfinished changes so that they can reapplied at any time.

```
git stash apply STASH-NAME
```

Git Stash

- To create a stash use the command `git stash`
- To view the list of stashes use the command `git stash list`
- To view a particular stash, use the command

```
git stash show STASH-NAME
```

- To apply the changes and leave a copy in the stash, use the command

```
git stash apply STASH-NAME
```

- To apply the changes and remove the file from the stash, use

```
git stash pop STASH-NAME
```

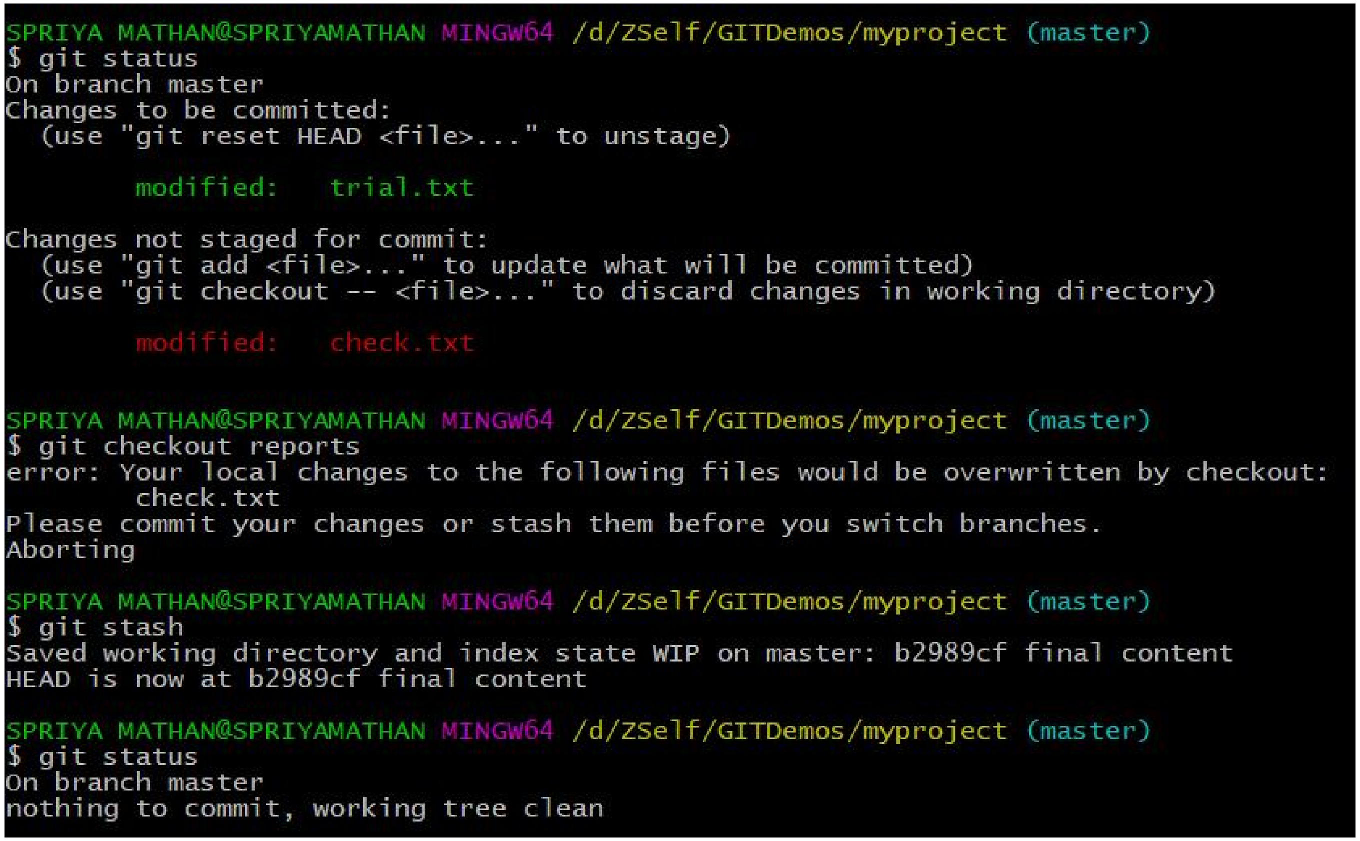
- To remove stashed changes without applying them, use

```
git stash drop STASH-NAME
```

- To clear the entire stash

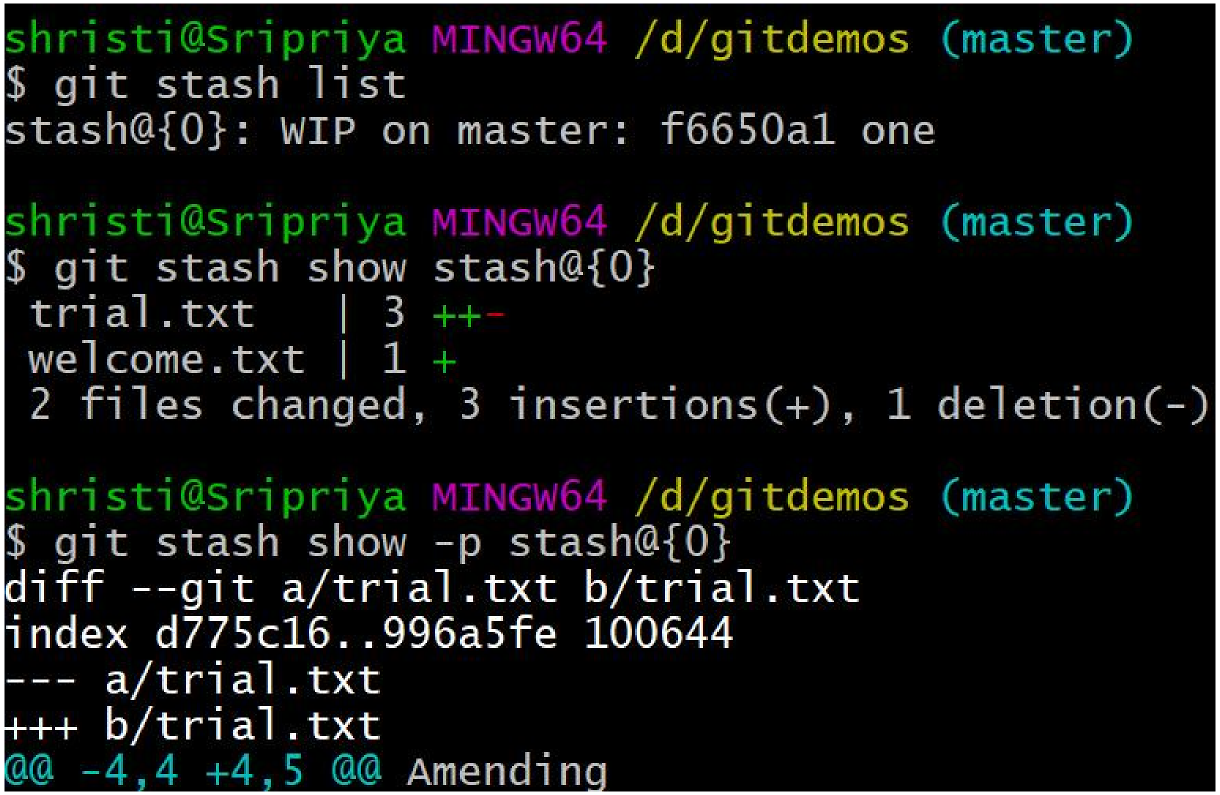
```
git stash drop STASH-NAME
```

Example



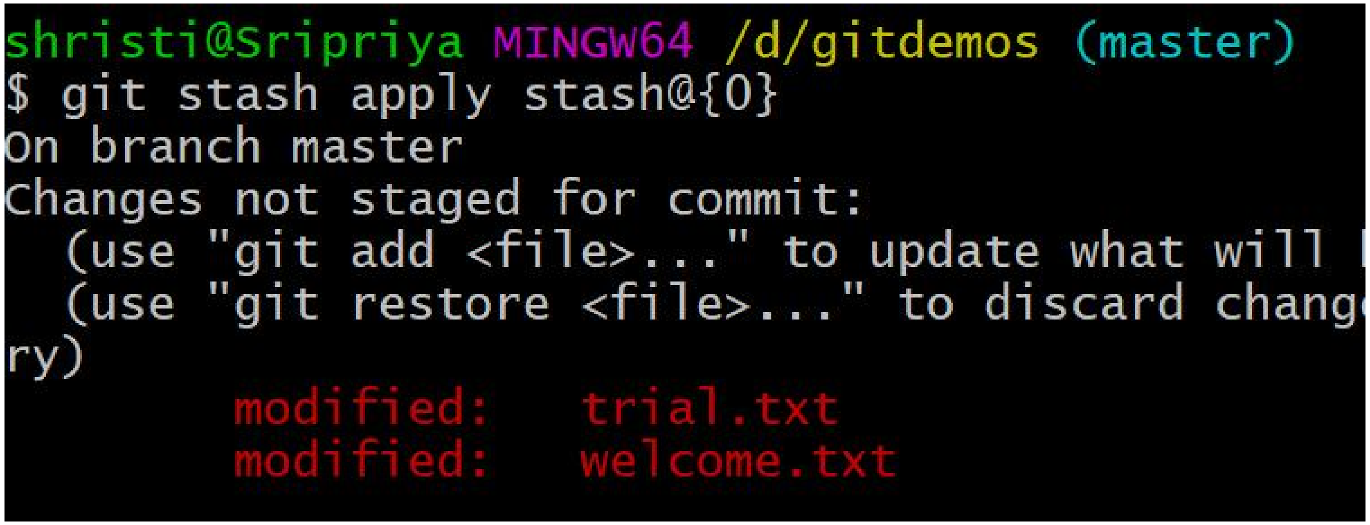
```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   trial.txt

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:   check.txt


SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git checkout reports
error: Your local changes to the following files would be overwritten by checkout:
        check.txt
Please commit your changes or stash them before you switch branches.
Aborting

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git stash
Saved working directory and index state WIP on master: b2989cf final content
HEAD is now at b2989cf final content

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git status
On branch master
nothing to commit, working tree clean
```

To view the stashes



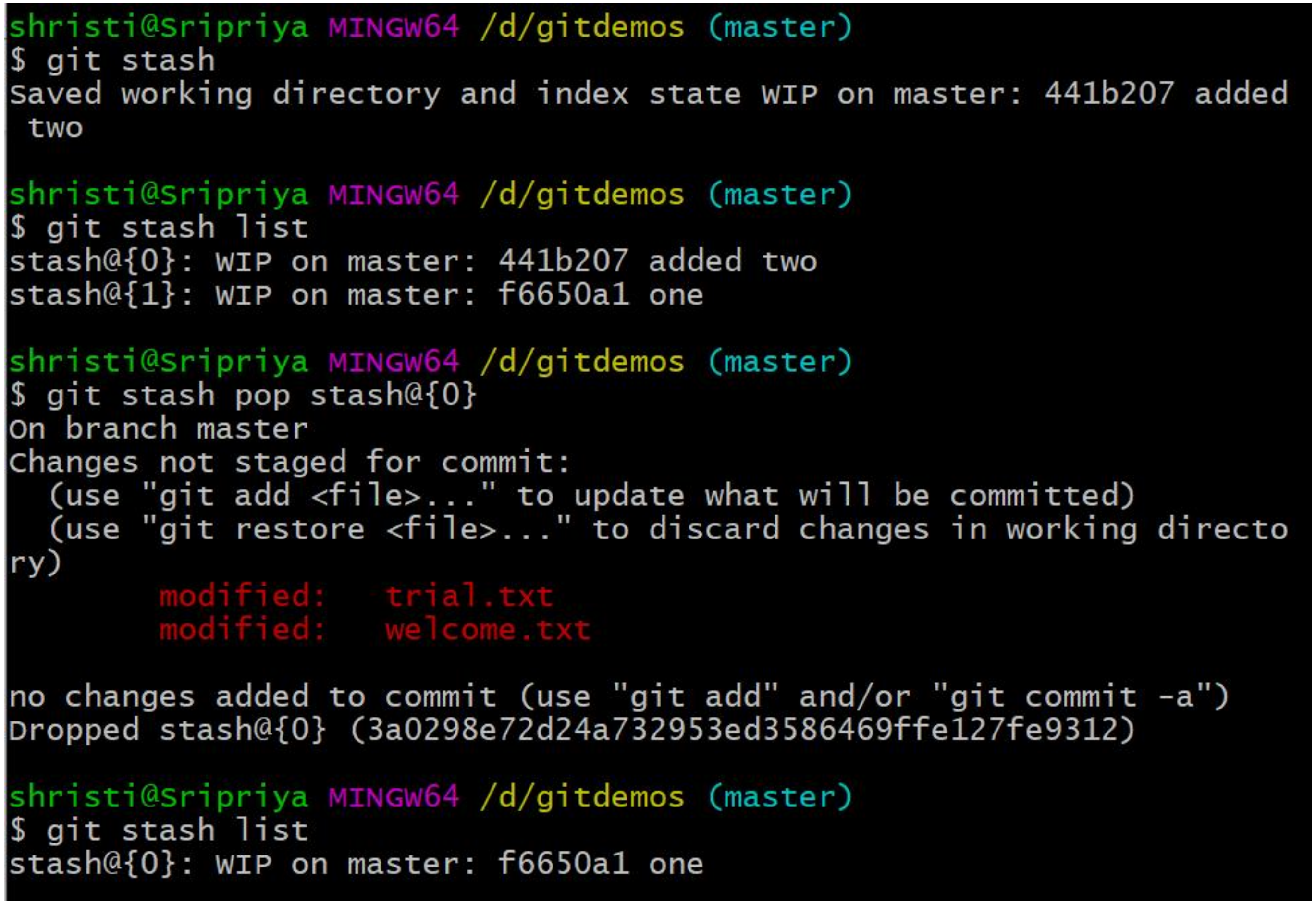
```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash list
stash@{0}: WIP on master: f6650a1 one

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash show stash@{0}
 trial.txt   | 3 ++-
 welcome.txt | 1 +
 2 files changed, 3 insertions(+), 1 deletion(-)

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash show -p stash@{0}
diff --git a/trial.txt b/trial.txt
index d775c16..996a5fe 100644
--- a/trial.txt
+++ b/trial.txt
@@ -4,4 +4,5 @@ Amending
```

To apply the stashes



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash apply stash@{0}
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will
  (use "git restore <file>..." to discard chang
ry)
        modified:   trial.txt
        modified:   welcome.txt
```

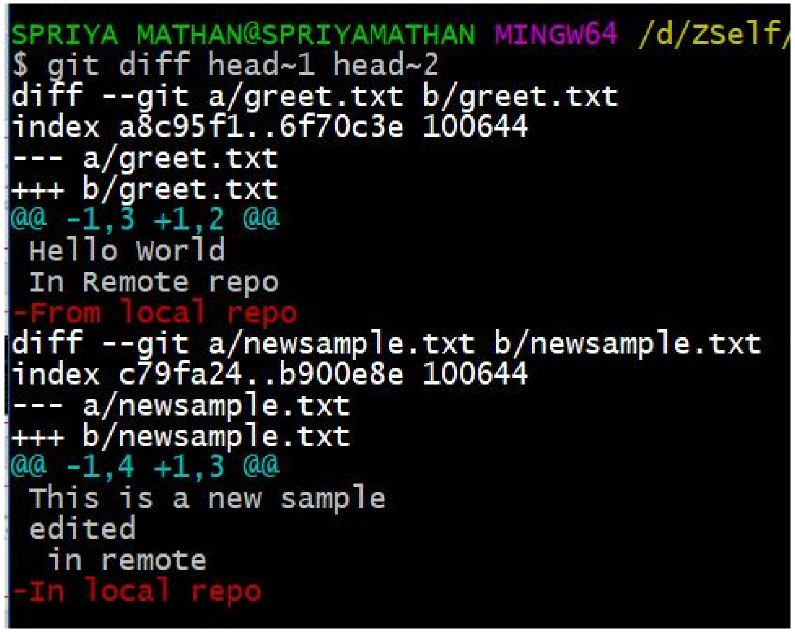
stash pop



```
shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash
Saved working directory and index state WIP on master: 441b207 added
 two

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash list
stash@{0}: WIP on master: 441b207 added two
stash@{1}: WIP on master: f6650a1 one

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash pop stash@{0}
On branch 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 directo
ry)
        modified:   trial.txt
        modified:   welcome.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped stash@{0} (3a0298e72d24a732953ed3586469ffe127fe9312)

shristi@Sripriya MINGW64 /d/gitdemos (master)
$ git stash list
stash@{0}: WIP on master: f6650a1 one
```

Head

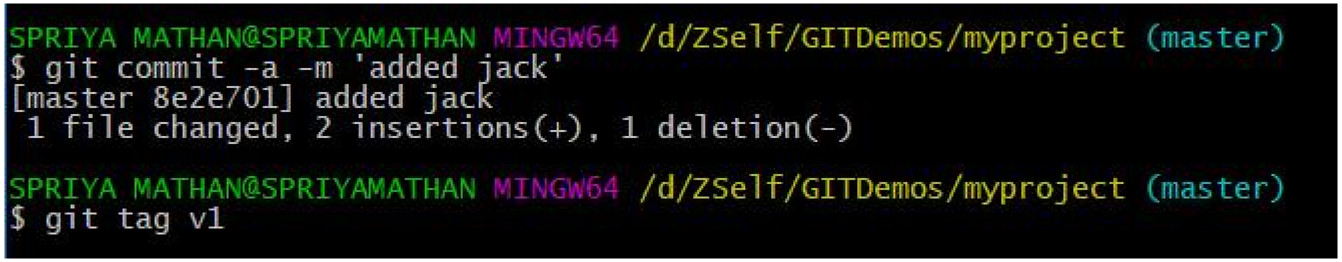
- HEAD is a pointer, which always points to the latest commit in the branch.
- Whenever you commit, HEAD is updated with the latest commit.
- The heads of the branches are stored in **.git/refs/heads/** directory.



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/zSelf/
$ git diff head~1 head~2
diff --git a/greet.txt b/greet.txt
index a8c95f1..6f70c3e 100644
--- a/greet.txt
+++ b/greet.txt
@@ -1,3 +1,2 @@
 Hello World
 In Remote repo
-From local repo
diff --git a/newsample.txt b/newsample.txt
index c79fa24..b900e8e 100644
--- a/newsample.txt
+++ b/newsample.txt
@@ -1,4 +1,3 @@
 This is a new sample
 edited
  in remote
-In local repo
```

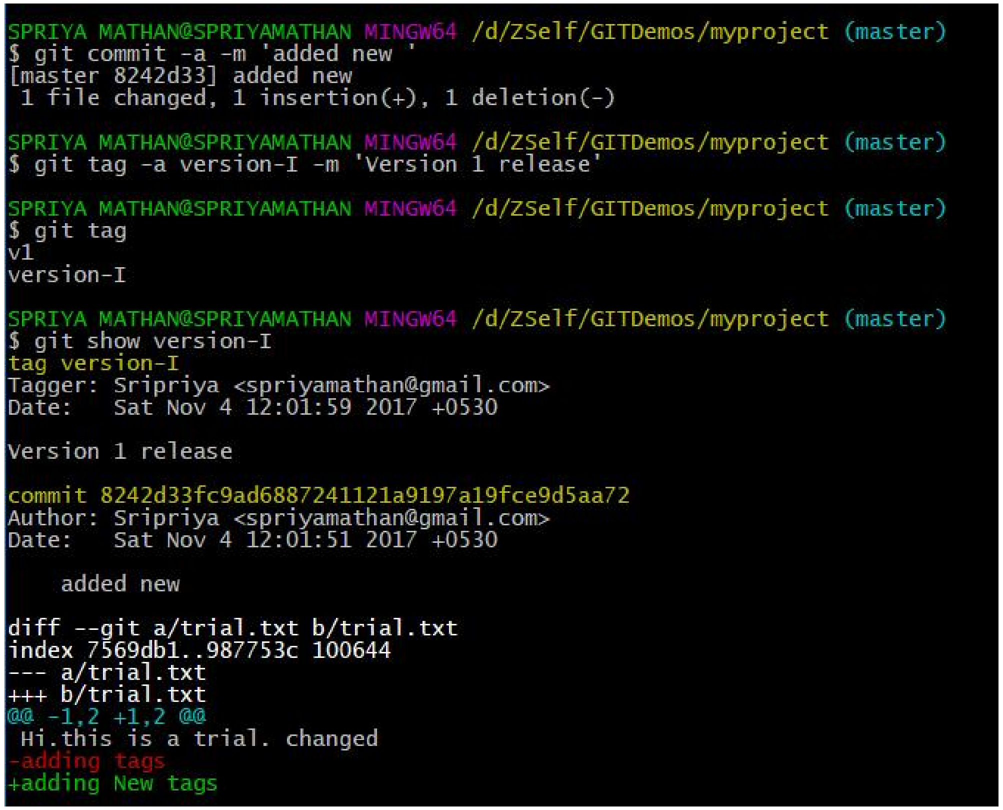
Tags

- Tag assigns a meaningful name with a specific version in the repository.
- Tag is similar to a branch, and should not be modified - immutable
- Once a tag is created for a particular commit, even if you create a new commit, it will not be updated.
- Create tags for product releases.



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git commit -a -m 'added jack'
[master 8e2e701] added jack
 1 file changed, 2 insertions(+), 1 deletion(-)

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git tag v1
```

Example



```
SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git commit -a -m 'added new '
[master 8242d33] added new
 1 file changed, 1 insertion(+), 1 deletion(-)

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git tag -a version-I -m 'Version 1 release'

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git tag
v1
version-I

SPRIYA MATHAN@SPRIYAMATHAN MINGW64 /d/ZSelf/GITDemos/myproject (master)
$ git show version-I
tag version-I
Tagger: Sripriya <spriyamathan@gmail.com>
Date:   Sat Nov 4 12:01:59 2017 +0530

Version 1 release

commit 8242d33fc9ad6887241121a9197a19fce9d5aa72
Author: Sripriya <spriyamathan@gmail.com>
Date:   Sat Nov 4 12:01:51 2017 +0530

    added new

diff --git a/trial.txt b/trial.txt
index 7569db1..987753c 100644
--- a/trial.txt
+++ b/trial.txt
@@ -1,2 +1,2 @@
 Hi.this is a trial. changed
-adding tags
+adding New tags
```

URL

- URL represents the location of the Git repository.
- Git URL is stored in config file.

Summary

- Version Control System(VCS)
- Overview of GIT
- GIT Repositories
- Workflow of GIT
- Creating a local repository
- Staging and committing
- Git Branching and merging
- Working with Remotes
- GIT stashing
- Head, Tags, URL

Shristi Technology Labs

Thank you