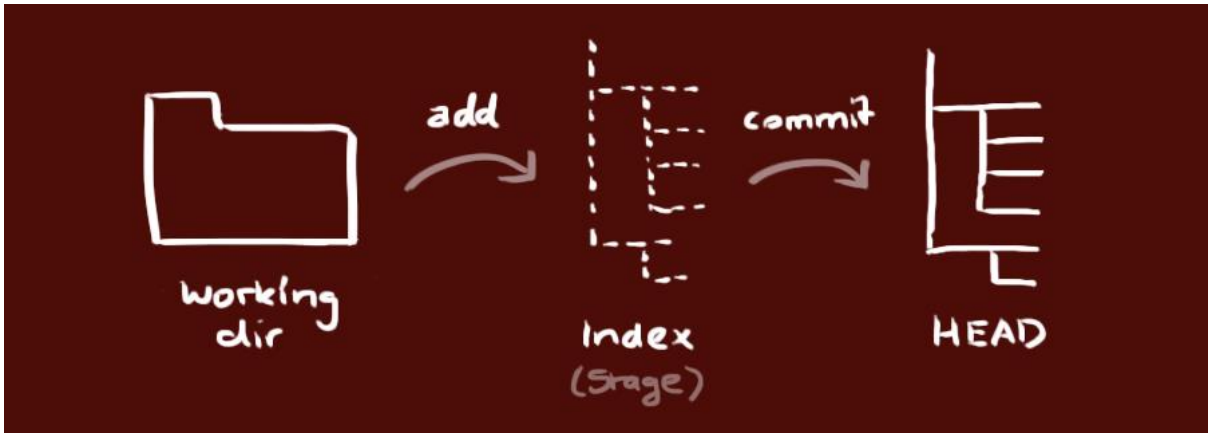
Git Notes:

Git Structure and commands

Your local repository consists of three "trees" maintained by git.

1. Your Working Directory which holds the actual files.
2. The Index which acts as a staging area
3. The HEAD which points to the last commit you've made.



Status

git status	Check the status of the branch you are currently on. It shows two things 1. Current branch name 2. Changed files

Add

You can propose changes (add it to the Index) using

git add <filename>	For single file
git add <filename1> <filename2>	For more than one file
git add *	All the files

Commit

1. This is the first step in the basic git workflow. To actually commit these changes use

```
git commit -m "Commit message"
```

2. Now the file is committed to the HEAD, but not in your remote repository yet.

Pushing changes

Your changes are now in the HEAD of your local working copy. To send those changes to your remote repository, execute

```
git push origin <feature/master>
```

Pulling changes

To get the latest changes from repository for a feature/branch

git checkout <branch name> git pull	

Create a new repository

Steps

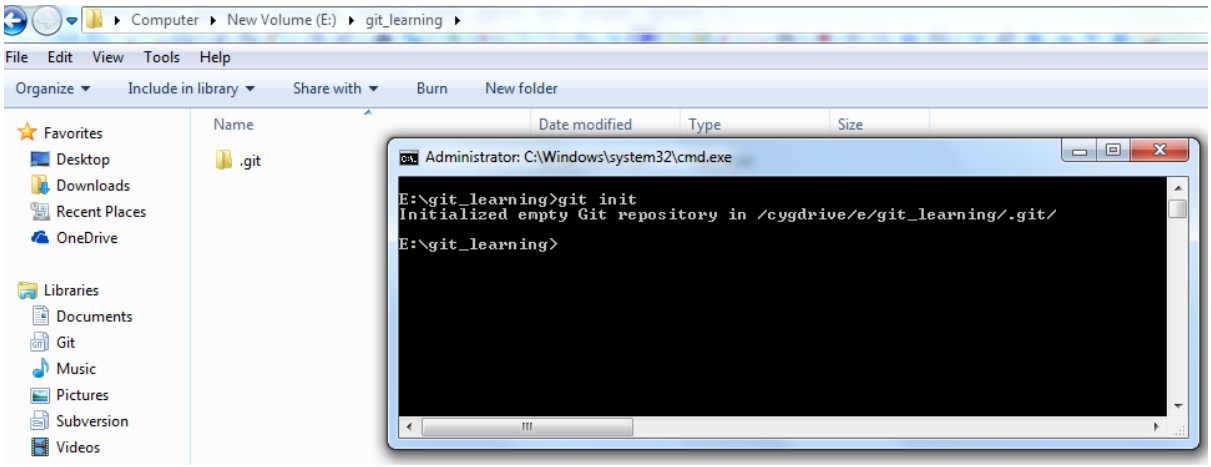
1. create a new directory(folder), open it and perform a

git init	to create a new git repository

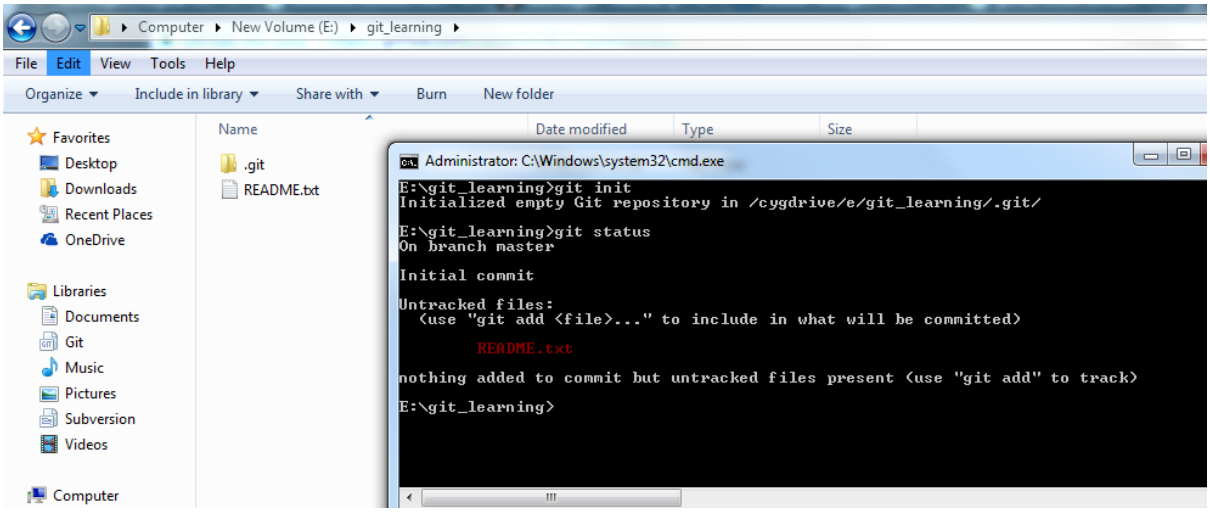
For learning purpose I have created a account samson1487 on github

I have created a directory (folder) called **git_learning** on my local system

- Open the command prompt and type the command git init. After successful completion of command you can see .git folder being created inside git_learning folder



- Create a sample file called README.txt and add comment “This is my first repo”
- I ran the command git status (discussed further) . As of now I do not create any branch . By default I am on master branch.



- It shows README.txt in red. Add and commit the file.

- `git add README.md`
- `git commit -m "first commit"`
- `git remote add origin https://github.com/samson1487/myfirstrepo.git`
- `git push -u origin master`

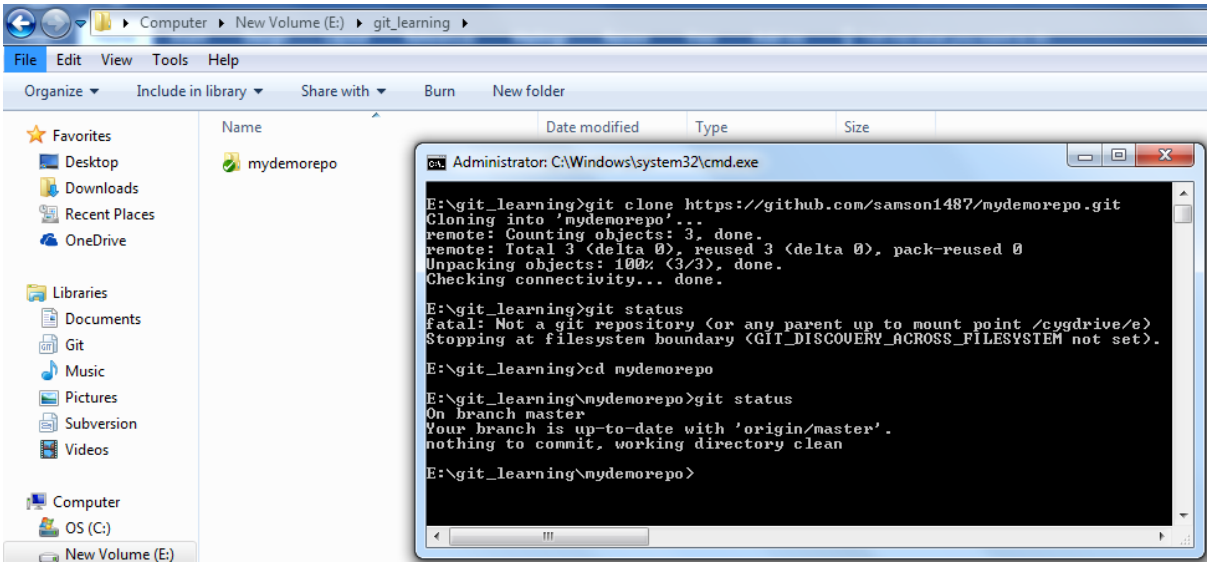
```
E:\git_learning>git status
On branch master

Initial commit

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

        README.txt

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

E:\git_learning>git add README.txt

E:\git_learning>git commit -m "first commit"
[master (root-commit) 3dbc151] first commit
 1 file changed, 1 insertion(+)
 create mode 100755 README.txt

E:\git_learning>git remote add origin https://github.com/samson1487/mydemorepo.git

E:\git_learning>git push -u origin master
git: 'credential-wincred' is not a git command. See 'git --help'.
Username for 'https://github.com':
Password for 'https://samson1487@github.com':
git: 'credential-wincred' is not a git command. See 'git --help'.
Counting objects: 3, done.
Writing objects: 100% (3/3), 231 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/samson1487/mydemorepo.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

E:\git_learning>
```

Checkout (Clone) a repository

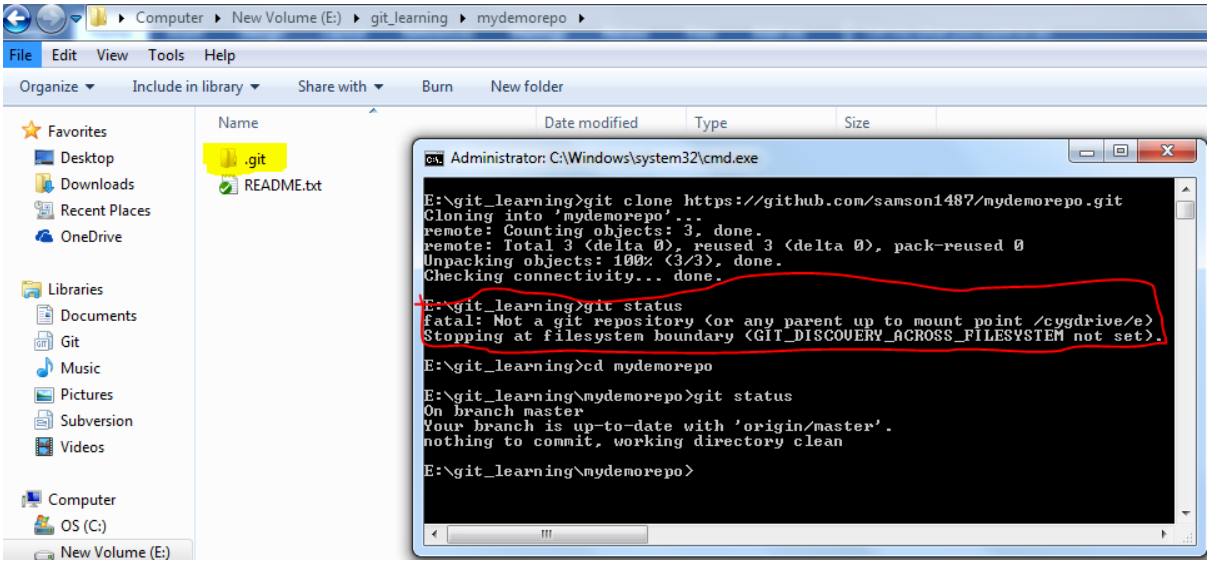
Create a working local copy of a remote repository by running the command

git clone username@host:/path/to/repository	when using a remote server, your command will be

- For e.g in order to create a local copy of the remote repository (e.g https://github.com/samson1487/mydemorepo.git) let us create a local folder named **git_learning.** Open the cmd/terminal in the folder and execute the above git command .



- A new folder named “mydemorepo” created inside the git_learning folder. Inside the this folder all the files resides and the important .**git** folder (highlighted with yellow color)



- Important thing to note here is that all the git commands must be executed from this level i.e at the level where .git folder resides.
  I have highlighted the area with red colour when I tried to execute the git status command from the wrong location.
- Run git command git status to see the on which branch you are on currently.

Branching

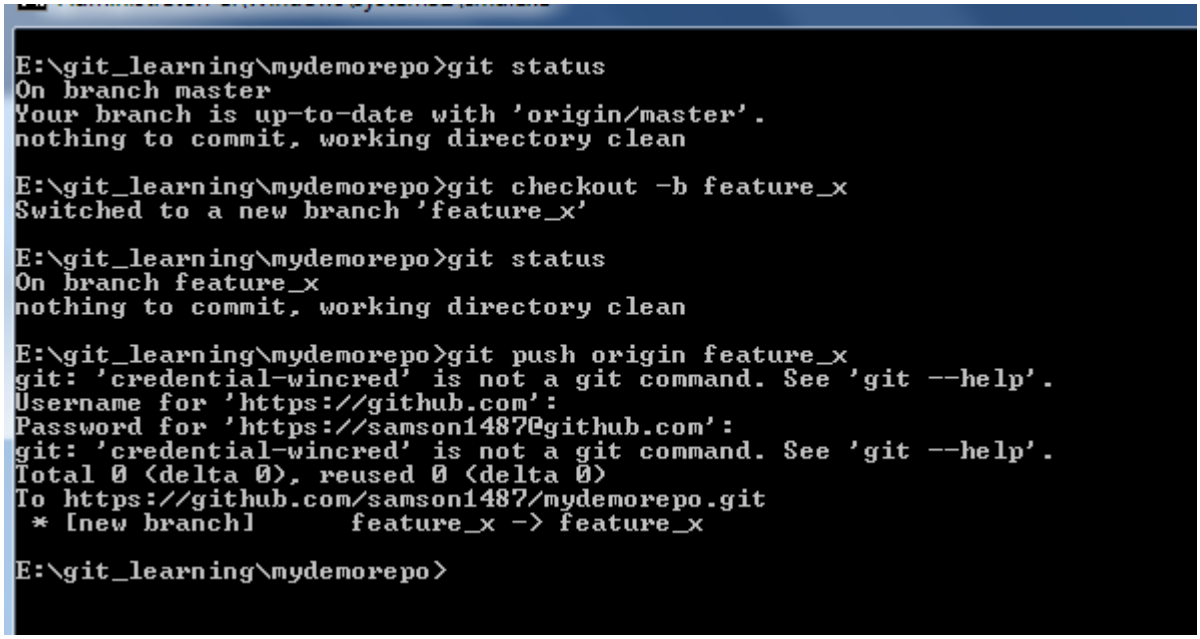
- Branches are used to develop features isolated from each other.
- The master branch is the "default" branch when you create a repository.
- Use other branches for development and merge them back to the master branch upon completion.

1. create a new branch named "feature_x" and switch to it using

```
git checkout -b feature_x
```

2. A branch is not available to others unless you push the branch to your remote repository

```
git push origin <branch>
```



Update & merge

- To update your local repository to the newest commit,
    1. Go to the directory(folder) where .git folder resides
    2. Execute below command to fetch and merge remote changes

```
git pull
```

- To merge another branch into your active branch (e.g. master), use

```
git merge <branch>
```

NOTE :: In both the above cases (git pull & git merge) git tries to auto-merge changes]

- Unfortunately, this is not always possible and results in conflicts.

  You are responsible to merge those conflicts manually by editing the files shown by git.

  git add <filename>

  before merging changes, you can also preview them by using

  git diff <source_branch> <target_branch>

  After changing, you need to mark them as merged with

Lets take an example on how to solve merge issues

1. For example purpose let us take the file README.txt that we have created in the above example.
2. Suppose two users ,user1 and user2 both working on branch feature_x (created in the previous example) are working on the same file README.txt. Now user1 make some changes in the file and commit and pushed it into the branch feature_x.
   Contents of file README.txt after user 1 changes

```
1  my first repo
2  line added by code 1
```

```
E:\git_learning\code 1\mydemorepo>git status
On branch feature_x
Your branch is up-to-date with 'origin/feature_x'.
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:   README.txt

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

E:\git_learning\code 1\mydemorepo>git add README.txt

E:\git_learning\code 1\mydemorepo>git commit -m "comment from code 1"
[feature_x 8be9e8b] comment from code 1
 1 file changed, 2 insertions(+), 1 deletion(-)

E:\git_learning\code 1\mydemorepo>git status
On branch feature_x
Your branch is ahead of 'origin/feature_x' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
```

3. User 2 knows nothing about it. He changes his local copy of README.txt and commits it.

```
1   line added by code 2
```

user 2 changes for README.txt

```
E:\git_learning\code 2\mydemorepo>git status
On branch feature_x
Your branch is up-to-date with 'origin/feature_x'.
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:   README.txt

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

E:\git_learning\code 2\mydemorepo>git add README.txt

E:\git_learning\code 2\mydemorepo>git commit -m "comment from code 2"
[feature_x b43819a] comment from code 2
 1 file changed, 1 insertion(+), 1 deletion(-)

E:\git_learning\code 2\mydemorepo>git status
On branch feature_x
Your branch is ahead of 'origin/feature_x' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
```

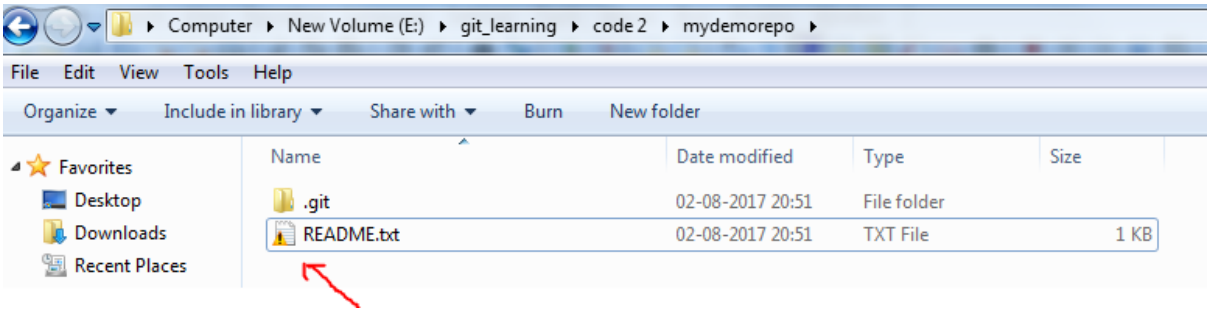
4. But when he tries to push the same file to branch feature_x he will receive and error

```
E:\git_learning\code 2\mydemorepo>git push
git: 'credential-wincred' is not a git command. See 'git --help'.
Username for 'https://github.com':
Password for 'https://samson1487@github.com':
git: 'credential-wincred' is not a git command. See 'git --help'.
To https://github.com/samson1487/mydemorepo.git
 ! [rejected]        feature_x -> feature_x (fetch first)
error: failed to push some refs to 'https://github.com/samson1487/mydemorepo.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

5. Now before pushing his local changes to the remote branch he needs to take a **pull** to get the remote changes of the file and then if any conflicts arises then he needs to manually resolved those conflicts.
   Please note when users does git pull , git tries to automatically resolve the conflicts but if it is not able to resolved it then user has to resolve it manually .

6. User 2 does executes command git pull but CONFLICTS arises in the file

```
E:\git_learning\code 2\mydemorepo>git pull
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/samson1487/mydemorepo
   3dbc151..8be9e8b  feature_x  -> origin/feature_x
Auto-merging README.txt
CONFLICT (content): Merge conflict in README.txt
Automatic merge failed; fix conflicts and then commit the result.
```



7. As you can see in the above example (highlighted in the yellow) it is clearly showing conflicts (look at the conflict symbol in second screenshot) while merging the changes from remote to the local repository file README.txt
8. Now user 2 open the README.txt which shows the following content

```
<<<<<<< HEAD
line added by code 2
=======
my first repo
line added by code 1
>>>>>>> 8be9e8b443acacd202b1a2c3fdd768f1f34225bb

```

9. The contents between “<<<<<<< HEAD” and “=======” shows user 2 changes for the README.txt file
10. The contents between “=======” and
    “ >>>>>>> 8be9e8b443acacd202b1a2c3fdd768f1f34225bb” shows user 1 changes for the README.txt file
11. Now its is upto user 2 to select what will be the final contents of the file ,based on this three possibilities arises
    - First he decides to have only his changes in that case the final content of the file will be

```
line added by code 2
```

      .
      Note : The final changes must not include “<<<<<<< HEAD” , “=======” and

" >>>>>>> 8be9e8b443acacd202b1a2c3fdd768f1f34225bb". These are separator used by git to show differences.

- Second user 1 changes

```
1  my first repo
2  line added by code 1
```

- Third both the changes

```
1  line added by code 2
2
3  my first repo
4  line added by code 1
5
6
```

12. Lets say he goes with the third option, he made the changes and commits the file.

```
E:\git_learning\code 2\mydemorepo>git status
On branch feature_x
Your branch and 'origin/feature_x' have diverged,
and have 1 and 1 different commit each, respectively.
  (use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   README.txt

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

E:\git_learning\code 2\mydemorepo>git add README.txt

E:\git_learning\code 2\mydemorepo>git commit -m "merged with the remote branch"
[feature_x a30a95b] merged with the remote branch

E:\git_learning\code 2\mydemorepo>git status
On branch feature_x
Your branch is ahead of 'origin/feature_x' by 2 commits.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean

E:\git_learning\code 2\mydemorepo>git push
git: 'credential-wincred' is not a git command. See 'git --help'.
Username for 'https://github.com':
Password for 'https://samson1487@github.com':
git: 'credential-wincred' is not a git command. See 'git --help'.
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 572 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/samson1487/mydemorepo.git
   8be9e8b..a30a95b  feature_x -> feature_x
```

Finally he pushes the file to branch feature_x

Logs

in its simplest form, you can study repository history using.. git log

You can add a lot of parameters to make the log look like what you want.

To see only the commits of a certain author:

git log --author=bob

See only which files have changed:

git log --name-status

#replace local changes

In case you did something wrong, which for sure never happens , you can replace local changes using the command

git checkout -- <filename> [this replaces the changes in your working tree with the last content in HEAD. Changes already added to the index, as well as new files, will be kept.]

If you instead want to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it like this

git fetch origin

git reset --hard origin/master

##References

http://rogerdudler.github.io/git-guide/