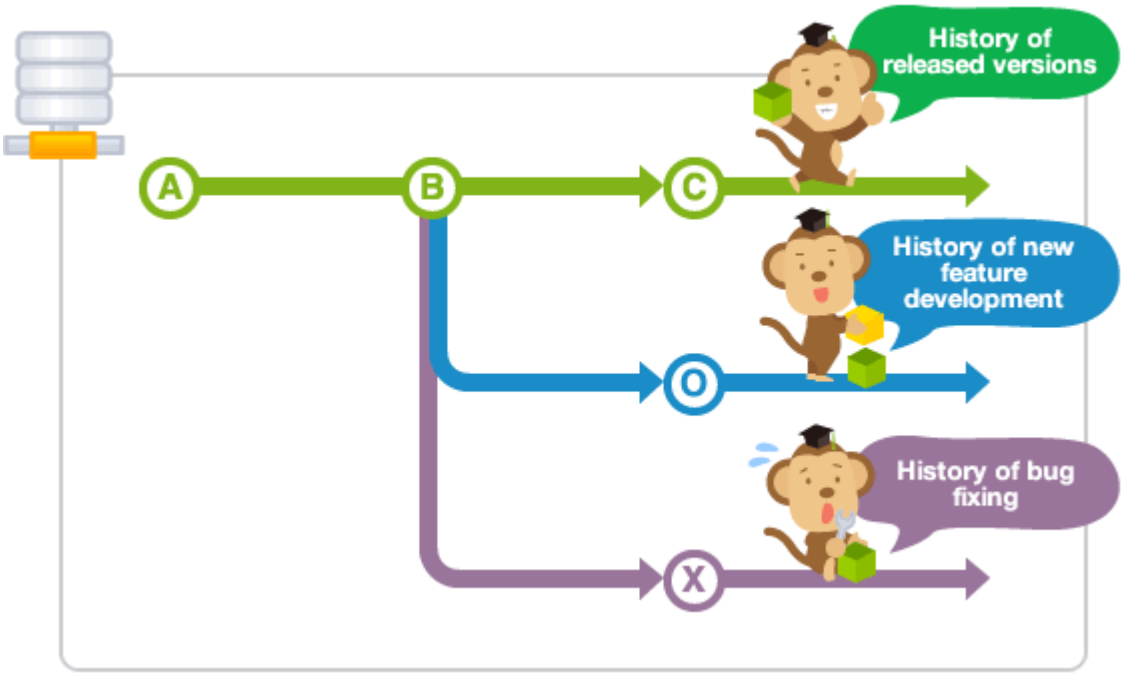
We will understand git branch.



If our code is pointing only to one branch i.e., master, we can work on only one issue. By any chance if we get struck somewhere and unable to work on it, we cannot work on other issues till we make a commit on master branch or reset the present changes - which is undesirable.

If we are on our master branch, we can checkout to a new branch in git and create a sub branch. We can create multiple branches from any commit and from any sub branch. We can switch across branches and work on those branches.

To list the branches, use git branch

```
fms@git_practice:~/dev_flowers/sample-project$ git branch
* master
fms@git_practice:~/dev_flowers/sample-project$


fms@git_practice:~/dev_flowers/sample-project$ git checkout -b
branch_one
Switched to a new branch 'branch_one'
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git branch
* branch_one
```

```
  master
fms@git_practice:~/dev_flowers/sample-project$
```

This means that we have two branches and the active branch is branch_one.

```
 fms@git_practice:~/dev_flowers/sample-project$ git checkout -b branch_two
Switched to a new branch 'branch_two'
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git branch
  branch_one
* branch_two
  master
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git branch -D branch_two
error: Cannot delete branch 'branch_two' checked out at '/home/fms/dev_flowers/sample-project'
fms@git_practice:~/dev_flowers/sample-project$
```

Error: We cannot delete the branch while we are still on it. So move to another branch and then delete.

```
fms@git_practice:~/dev_flowers/sample-project$ git checkout branch_one
Switched to branch 'branch_one'
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git branch -D branch_two
Deleted branch branch_two (was 544e581).
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git branch
* branch_one
  master
fms@git_practice:~/dev_flowers/sample-project$
```

branch_two is deleted. But think twice before deleting branches as you may loose some data.
To get the list of commits, use this command.

```
fms@git_practice:~/dev_flowers/sample-project$ git log --oneline
544e581 (HEAD -> branch_one, origin/master, origin/HEAD, master)
```

```
updated fruits.c
a135ee5 created flowers.cpp
36c5a24 created a file fruits.c
fms@git_practice:~/dev_flowers/sample-project$
```

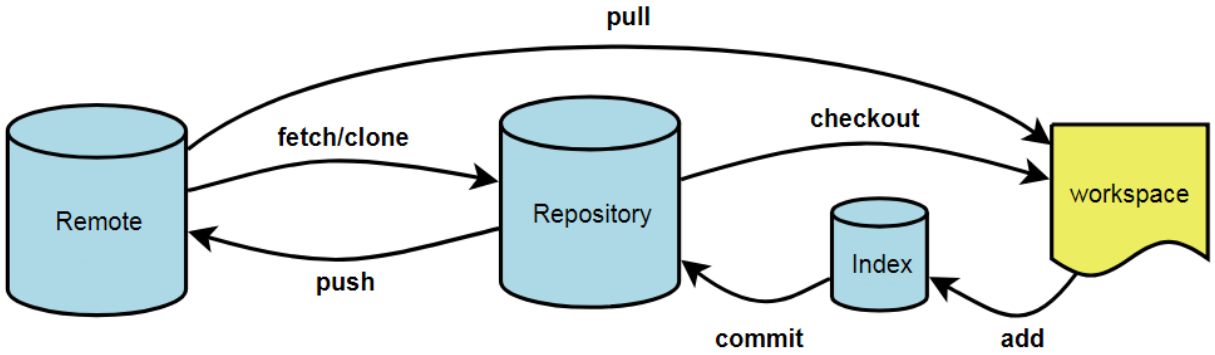
This will display only commit id and commit text.

```
fms@git_practice:~/dev_flowers/sample-project$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git branch
  branch_one
* master
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git log --oneline
544e581 (HEAD -> master, origin/master, origin/HEAD, branch_one)
updated fruits.c
a135ee5 created flowers.cpp
36c5a24 created a file fruits.c
fms@git_practice:~/dev_flowers/sample-project$
```

Ok. We have only 3 commits now. We will try to get all the commits from github.

Git fetch will fetch all the commits from origin but will not merge the changes.
We have to use git merge command to merge the changes.
But git pull will directly pull the commits and merge the commits on to the active branch.

Git pull = git fetch + git merge



```
fms@git_practice:~/dev_flowers/sample-project$ git fetch origin master
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 15 (delta 5), reused 15 (delta 5), pack-reused 0
Unpacking objects: 100% (15/15), done.
From https://github.com/srivalli-projects/sample-project
 * branch            master     -> FETCH_HEAD
   544e581..3d1b34e  master     -> origin/master
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git log --oneline
544e581 (HEAD -> master, branch_one) updated fruits.c
a135ee5 created flowers.cpp
36c5a24 created a file fruits.c
fms@git_practice:~/dev_flowers/sample-project$
```

Now merge the changes.

```
fms@git_practice:~/dev_flowers/sample-project$ git merge origin/master
Updating 544e581..3d1b34e
Fast-forward
 .gitignore              | 1 +
 flowers.cpp => buds.cpp | 0
 fruits.c                | 2 --
 sweets.py               | 1 +
 4 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 .gitignore
 rename flowers.cpp => buds.cpp (100%)
 delete mode 100644 fruits.c
 create mode 100644 sweets.py
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git log --oneline
3d1b34e (HEAD -> master, origin/master, origin/HEAD) Revert "created
colors.java"
803fdf4 renamed flowers.c to buds.c
bcfcc4e remove fruits.c from this project
32960a2 update .gitignore to ignore html files
7d8d1d6 created sweets.py
66c27af created colors.java
544e581 (branch_one) updated fruits.c
```

```
a135ee5 created flowers.cpp
36c5a24 created a file fruits.c
```

Now our number of commits is not three after git merge.

Observe that we are on branch master. We have the commits merged onto this branch because we used the command git push origin/master.

Master branch will have all these commits.
Branch_one will have only 3 commits.

```
fms@git_practice:~/dev_flowers/sample-project$
fms@git_practice:~/dev_flowers/sample-project$ git checkout branch_one
Switched to branch 'branch_one'
fms@git_practice:~/dev_flowers/sample-project$ git branch
* branch_one
  master
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git log --oneline
544e581 (HEAD -> branch_one) updated fruits.c
a135ee5 created flowers.cpp
36c5a24 created a file fruits.c
fms@git_practice:~/dev_flowers/sample-project$
```

If we want to get the changes of one branch to another, make sure we are on the destination branch and give git merge sourcebranch

In our case, our destination branch is branch_one. So we have checkout to that branch to merge the commits on master branch.

```
fms@git_practice:~/dev_flowers/sample-project$ git merge master
Updating 544e581..3d1b34e
Fast-forward
 .gitignore              | 1 +
 flowers.cpp => buds.cpp | 0
 fruits.c                | 2 --
 sweets.py               | 1 +
 4 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 .gitignore
 rename flowers.cpp => buds.cpp (100%)
 delete mode 100644 fruits.c
```

```
 create mode 100644 sweets.py
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git log --oneline
3d1b34e (HEAD -> branch_one, origin/master, origin/HEAD, master)
Revert "created colors.java"
803fdf4 renamed flowers.c to buds.c
bcfcc4e remove fruits.c from this project
32960a2 update .gitignore to ignore html files
7d8d1d6 created sweets.py
66c27af created colors.java
544e581 updated fruits.c
a135ee5 created flowers.cpp
36c5a24 created a file fruits.c
fms@git_practice:~/dev_flowers/sample-project$
```

Now both the branches have all the commits we did so far.

We will create another file to understand how to work on patch files.

```
fms@git_practice:~/dev_flowers/sample-project$ vi candy.sh
fms@git_practice:~/dev_flowers/sample-project$ cat candy.sh
chocolates
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git add candy.sh
fms@git_practice:~/dev_flowers/sample-project$ git commit -m "new file
candy.sh"
[branch_one b52fa47] new file candy.sh
 1 file changed, 1 insertion(+)
 create mode 100644 candy.sh
fms@git_practice:~/dev_flowers/sample-project$
```

We will check only last two commits (latest commits)

```
fms@git_practice:~/dev_flowers/sample-project$ git log -2
commit b52fa475a63ec7138d999039370941483bbcaa66 (HEAD ->
branch_one)
Author: Dev-Flowers <dev_flowers@FLOWERS.com>
Date:   Tue Nov 6 15:41:46 2018 +0530

    new file candy.sh

commit 3d1b34e03c57339e04511ec7d1619929cb5ca54b (origin/master,
```

```
origin/HEAD, master)
Author: Dev-Colors <dev_colors@COLORS.com>
Date:   Tue Nov 6 15:16:49 2018 +0530

    Revert "created colors.java"

    This reverts commit 66c27af89135ff0e8b045d8be0b425165960f704.
fms@git_practice:~/dev_flowers/sample-project$
```

Suppose we need to share this commit with another developer even before pushing the commit to the remote server(github).
Since this commit is local to our machine, we can create a patch file for the commits and share over email/dropbox.

We will now create a patch file for the latest commit.

```
fms@git_practice:~/dev_flowers/sample-project$ git format-patch -1
0001-new-file-candy.sh.patch
fms@git_practice:~/dev_flowers/sample-project$ ls
0001-new-file-candy.sh.patch  buds.cpp  candy.sh  sweets.py
fms@git_practice:~/dev_flowers/sample-project$
```

format-patch –n will create patch files for latest n commits.
The file name of the patch file is same as the commit message with .patch extension

The latest commit is made on branch_one. Patch file is created for this commit.
We will merge this patch file on master branch.

```
fms@git_practice:~/dev_flowers/sample-project$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
fms@git_practice:~/dev_flowers/sample-project$

fms@git_practice:~/dev_flowers/sample-project$ git log -1
commit 3d1b34e03c57339e04511ec7d1619929cb5ca54b (HEAD ->
master, origin/master, origin/HEAD)
Author: Dev-Colors <dev_colors@COLORS.com>
Date:   Tue Nov 6 15:16:49 2018 +0530

    Revert "created colors.java"
```

```
    This reverts commit 66c27af89135ff0e8b045d8be0b425165960f704.
fms@git_practice:~/dev_flowers/sample-project$
```

Apply the patch file here.

```
fms@git_practice:~/dev_flowers/sample-project$ git am 0001-new-file-candy.sh.patch
Applying: new file candy.sh
fms@git_practice:~/dev_flowers/sample-project$
```

Now this will become the latest commit.
At times, there will be issues with git am. It will not apply the patch if codebase is different. It will show conflicts which we need to resolve manually.

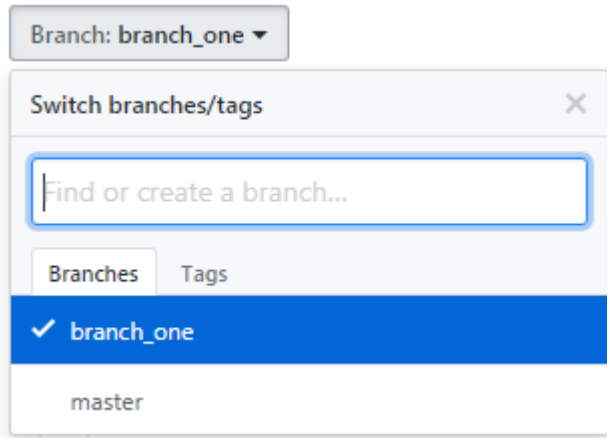
```
fms@git_practice:~/dev_flowers/sample-project$ git log -2
commit 1c6710a620e611532a2e886e396c94df4b50321c (HEAD -> master)
Author: Dev-Flowers <dev_flowers@FLOWERS.com>
Date:   Tue Nov 6 15:41:46 2018 +0530

    new file candy.sh

commit 3d1b34e03c57339e04511ec7d1619929cb5ca54b (origin/master, origin/HEAD)
Author: Dev-Colors <dev_colors@COLORS.com>
Date:   Tue Nov 6 15:16:49 2018 +0530

    Revert "created colors.java"

    This reverts commit 66c27af89135ff0e8b045d8be0b425165960f704.
fms@git_practice:~/dev_flowers/sample-project$
```

So even before pushing the commits to the server, we can create patch files for our commits to merge on the required branches/workspaces.

Now we will switch to branch_one and push the commit.
Observe that all the commits we pushed so far are from master branch.

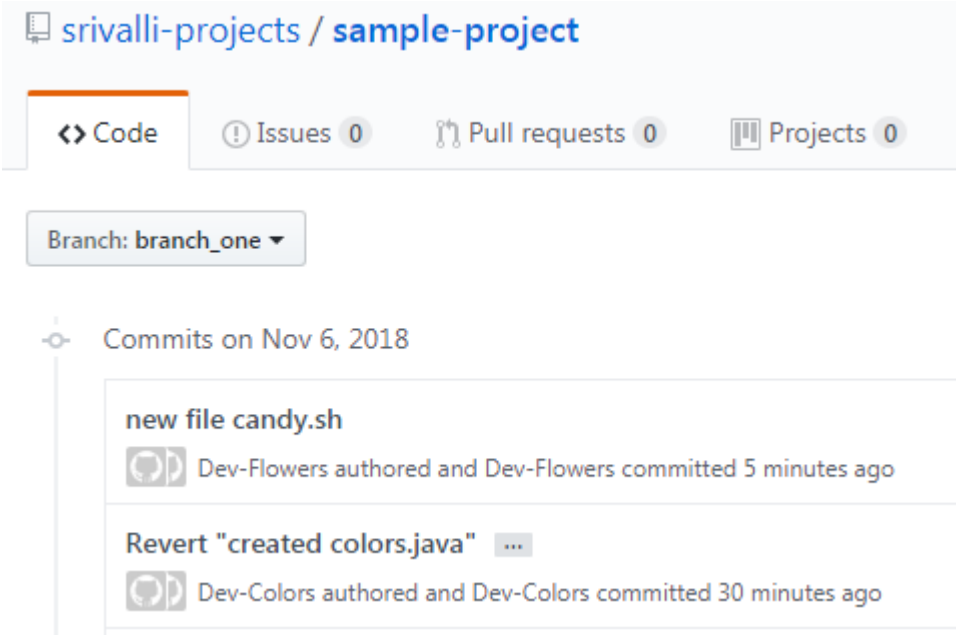
```
fms@git_practice:~/dev_flowers/sample-project$ git checkout branch_one
Switched to branch 'branch_one'
fms@git_practice:~/dev_flowers/sample-project$
```

```
fms@git_practice:~/dev_flowers/sample-project$ git push origin branch_one
Username for 'https://github.com': srivalli-projects
Password for 'https://srivalli-projects@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 368 bytes | 368.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'branch_one' on GitHub by visiting:
remote:      https://github.com/srivalli-projects/sample-project/pull/new/branch_one
remote:
To https://github.com/srivalli-projects/sample-project.git
 * [new branch]      branch_one -> branch_one
fms@git_practice:~/dev_flowers/sample-project$
```



We see two branches in github.

git push origin branch_one will push the changes onto github by creating a new branch called branch_one.



Git commands covered so far:

19. git checkout
20. git fetch
21. git merge
22. git format-patch
23. git am