Git Commands Explained

Let's get started

mayank_ahuja

git pull is git fetch + git merge -> Part 1

- Assume the local branch is in-sync with the remote branch at the moment. Both 'master' (local copy) and 'origin/master' (pointer to remote master) are pointing to the same commit '54ad..'.

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit 54ad99a28a346989955ea2a192baefe116bd126f (HEAD -> master, origin/master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:38:50 2022 +0530

    Create test.txt
```

- Now, suppose one of your teammate has added/pushed a new file on master remote branch. You local master is not aware of that change.
Let's do 'git fetch' and see what happens.

git fetch

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:mydemo222/git-commands
   54ad99a..b5dcb06  master     -> origin/master
```

What happened? Let's do 'git log' and see :

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit 54ad99a28a346989955ea2a192baefe116bd126f (HEAD -> master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:38:50 2022 +0530

    Create test.txt
```

Oh! I can only see 'master' pointing to '54ad..', but where is 'origin/master' ?
When you did 'git fetch' the origin/master got updated to what's in the remote master. As your teammate added a new change, it must be pointing to it.

git pull is git fetch + git merge -> Part 2

- Let's find out where is 'origin/master' git log origin/master

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log origin/master
commit b5dcb0625534c04c4a8e4c807b1beedf8ea1d030 (origin/master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:41:45 2022 +0530

    Create hello.txt

commit 54ad99a28a346989955ea2a192baefe116bd126f (HEAD -> master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:38:50 2022 +0530

    Create test.txt
```

- See, the origin/master points to the latest commit which is on remote master, but your local master still points to '54ad..' old commit.

How can we sync them again?

we need to do a merge now. git merge

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git merge
Updating 54ad99a..b5dcb06
Fast-forward
 hello.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
```

Awesome, both origin/master and master are now pointing to the same commit.

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit b5dcb0625534c04c4a8e4c807b1beedf8ea1d030 (HEAD -> master, origin/master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:41:45 2022 +0530

    Create hello.txt

commit 54ad99a28a346989955ea2a192baefe116bd126f
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:38:50 2022 +0530

    Create test.txt
```

git pull is git fetch + git merge -> Part 3

- We can perform the complete 'git fetch' + 'get merge' thing by just doing 'git pull'.

Assume, that your teammate again added a new text file on remote master and you local master is unaware of the change.

Current state - 'git log'

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit b5dcb0625534c04c4a8e4c807b1beedf8ea1d030 (HEAD -> master, origin/master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:41:45 2022 +0530

    Create hello.txt

commit 54ad99a28a346989955ea2a192baefe116bd126f
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:38:50 2022 +0530

    Create test.txt
```

- Let's do 'git pull' and see what happens : git pull

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:mydemo222/git-commands
   b5dcb06..e76592e  master     -> origin/master
Updating b5dcb06..e76592e
Fast-forward
 pull.txt | 1 +
```

When we did 'git pull', it first performed 'git fetch' followed by 'git merge'

'origin/master' & 'master' pointing to same new commit.

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit e76592e5404ccdcaa28e9961a8a9d12ed5b3ef95 (HEAD -> master, origin/master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:42:50 2022 +0530

    Create pull.txt

commit b5dcb0625534c04c4a8e4c807b1beedf8ea1d030
Author: Demo User <mak.alex09@gmail.com>
```

Creating and Publishing a Local branch to Remote

- create a local branch 'feature/demo1' and switch to newly created branch.

`git checkout -b feature/demo1` OR `git branch feature/demo1` `git checkout feature/demo1`

```
\gitdemo Let us learn/g/git/git-commands (feature/demo)
$ git checkout -b feature/demo1
Switched to a new branch 'feature/demo1'
```

- remote is unaware of this new branch, let's publish it to the remote

```
\gitdemo Let us learn/g/git/git-commands (feature/demo1)
$ git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
```

`git push -u origin feature/demo1`

```
\gitdemo Let us learn/g/git/git-commands (feature/demo1)
$ git push -u origin feature/demo1
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'feature/demo1' on GitHub by visiting:
remote:      https://github.com/mydemo222/git-commands/pull/new/feature/demo1
remote:
To github.com:mydemo222/git-commands.git
 * [new branch]      feature/demo1 -> feature/demo1
Branch 'feature/demo1' set up to track remote branch 'feature/demo1' from 'origin'.
```

- now if I do 'git pull', your remote is aware of this branch

```
\gitdemo Let us learn/g/git/git-commands (feature/demo1)
$ git pull
Already up to date.
```

Exploring git log command to view commit logs

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit 88ea3093a6e932d1d57a3fdfe7e781ed6e13b6d1 (HEAD -> master, origin/master)
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:43:06 2022 +0530

    Third commit

commit 9b4ecd547212dba91968120b3239b9b05d6d959c
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:42:42 2022 +0530

    Second commit

commit 0baba06ee243303430e374a87fe4364a0445f971
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:42:21 2022 +0530

    First commit
```

- viewing last n commits (in this case 2 commits)

git log -n2

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log -n2
commit 88ea3093a6e932d1d57a3fdfe7e781ed6e13b6d1 (HEAD -> master, origin/master)
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:43:06 2022 +0530

    Third commit

commit 9b4ecd547212dba91968120b3239b9b05d6d959c
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:42:42 2022 +0530

    Second commit
```

- one liner info of every commit

git log --oneline

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log --oneline
88ea309 (HEAD -> master, origin/master) Third commit
9b4ecd5 Second commit
0baba06 First commit
```

Exploring git log command to view commit logs

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit 88ea3093a6e932d1d57a3fdfe7e781ed6e13b6d1 (HEAD -> master, origin/master)
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:43:06 2022 +0530

    Third commit

commit 9b4ecd547212dba91968120b3239b9b05d6d959c
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:42:42 2022 +0530

    Second commit

commit 0baba06ee243303430e374a87fe4364a0445f971
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:42:21 2022 +0530

    First commit
```

- skipping last n commits (in this case 2 commits)

  git log --skip=2

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log --skip=2
commit 0baba06ee243303430e374a87fe4364a0445f971
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:42:21 2022 +0530

    First commit
```

- grep commits by string/pattern

  git log --grep="Second"

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git log --grep="Second"
commit 9b4ecd547212dba91968120b3239b9b05d6d959c
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 20:42:42 2022 +0530

    Second commit
```

git cherry-pick is cool

- As the name sounds, it is used to pick changes from existing commit and apply them by creating a new commit.

Let's see a simple example.

We are on 'feature/demo' branch currently and we have added a new change with commit id '32904...'- Created hello file

```
\gitdemo Let us learn/g/git/git-commands (feature/demo)
$ git log
commit 32904343f494309dcc6b3dd8b2fb41f9b3d6c6c7 (HEAD -> feature/demo, origin/feature/demo)
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 22:11:24 2022 +0530

    Created hello file

commit e76592e5404ccdcaa28e9961a8a9d12ed5b3ef95 (origin/master, master)
Author: Demo User <mak.alex09@gmail.com>
Date:   Thu Apr 28 21:42:50 2022 +0530

    Create pull.txt
```

- We can see the 'master' is behind by 1 commit from the feature.

We know one way to updated 'master' is by doing merge with the feature branch.

But here, we will see how cherry-pick works. We can pick a commit and apply.

So let's pick the commit '32904..' and apply it to master.

Let's switch to master and do cherry pick.

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git cherry-pick 32904343f494309dcc6b3dd8b2fb41f9b3d6c6c7
[master b2c0728] Created hello file
 Date: Thu Apr 28 22:11:24 2022 +0530
 1 file changed, 1 insertion(+)

\gitdemo Let us learn/g/git/git-commands (master)
$ git log
commit b2c072893f36151e1bf422c308f46d36dfb934bc (HEAD -> master)
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 22:11:24 2022 +0530

    Created hello file
```

After cherry-pick the same changes from that commit are applied to master branch. But it created a new commit with a different sha1.

Let's see the difference

- difference between 2 commits git diff commi1 commit2

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git diff 54ad99a28a346989955ea2a192baefe116bd126f a7e102b00a994cce983109bd2f6e5ca81dba4d38
diff --git a/hello.txt b/hello.txt
new file mode 100644
index 0000000..c6ba641
--- /dev/null
+++ b/hello.txt
@@ -0,0 +1,5 @@
+Helloooo
+Line 2 ..............
+Line 3 ..............
+Line 4 ..............
+Line 5 ..............
diff --git a/pull.txt b/pull.txt
new file mode 100644
index 0000000..7bee91d
--- /dev/null
+++ b/pull.txt
@@ -0,0 +1 @@
+Pull file
```

a/... from commit1 '54ad...' & b/... is from commit2 'a7e10..'. You can see 2 files (hello.txt and pull.txt) were added in commit2.

- similarly you can see diff of tip between two branches
  git diff branchA branchB
- to see difference between your local (unstaged) vs staged(indexed) changes
  git diff
- to see difference between staged(indexed) changes vs what's commited your local repo git diff --staged
  OR
  git diff --cached

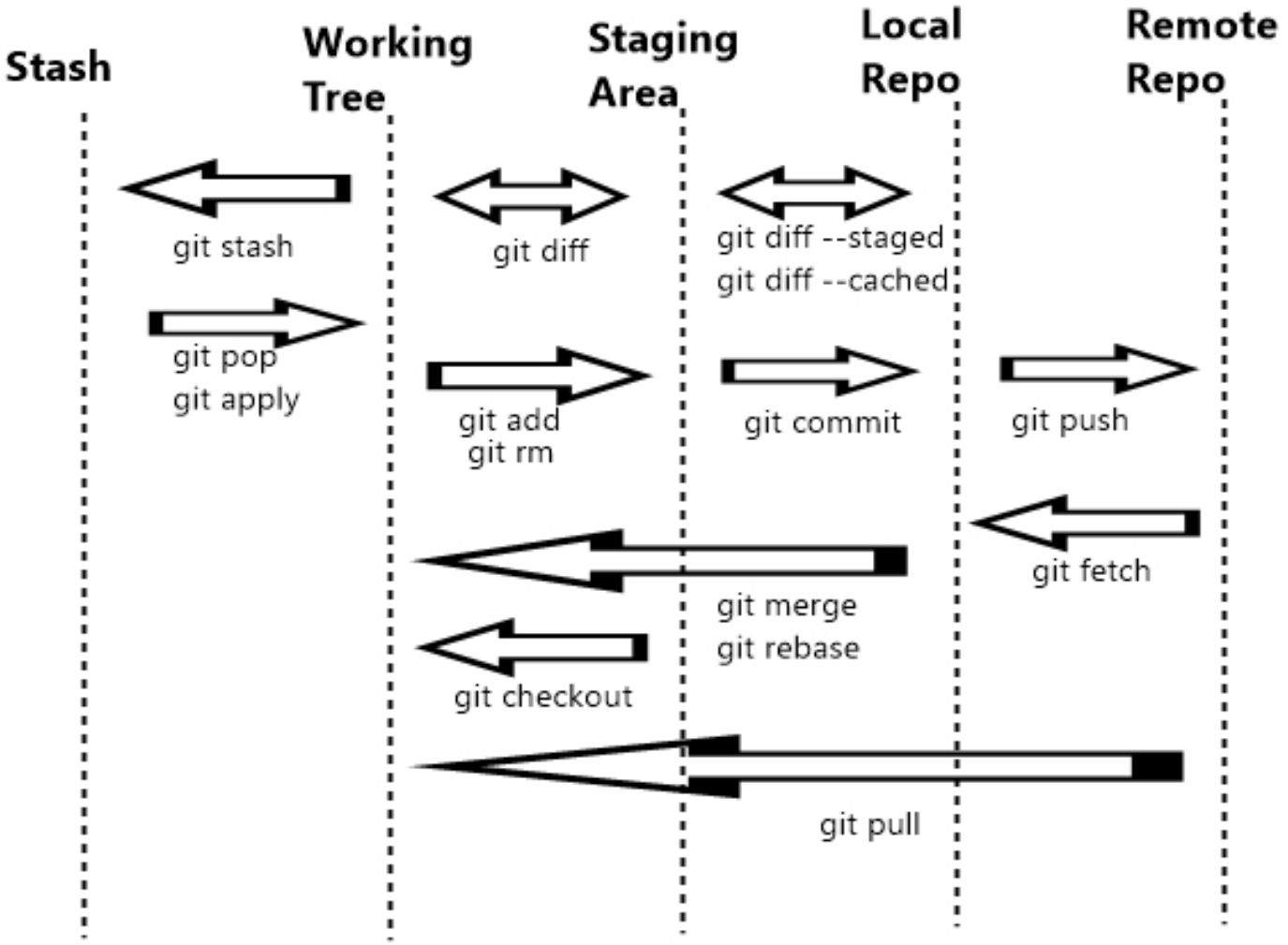
- to see changes in a commit
  git show <commit>

```
\gitdemo Let us learn/g/git/git-commands (master)
$ git show a7e102b00a994cce983109bd2f6e5ca81dba4d38
commit a7e102b00a994cce983109bd2f6e5ca81dba4d38 (HEAD -> master)
Author: mydemo222 <email@example.com>
Date:   Thu Apr 28 22:50:13 2022 +0530

    Adding new lines

diff --git a/hello.txt b/hello.txt
index 44f954a..c6ba641 100644
--- a/hello.txt
+++ b/hello.txt
@@ -1,2 +1,5 @@
-Hello File
 Helloooo
+Line 2 ..............
+Line 3 ..............
+Line 4 ..............
+Line 5 ..............
```

In a nutshell



mayank_ahuja