git
reset / revert

Let's UNDO

Current state - The local repository is in-sync with the remote, we have 2 text files (demoOne & demoTwo) that already exsists on remote and our working directory is clean(no local changes at the moment).

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (HEAD -> master, origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file

\gitdemo Let us learn/g/git/git-demo (master)
$ ls
demoOne.txt  demoTwo.txt

\gitdemo Let us learn/g/git/git-demo (master)
$ cat demoOne.txt
First line
Second line
Third line
Fourth line

\gitdemo Let us learn/g/git/git-demo (master)
$ cat demoTwo.txt
Valid line

\gitdemo Let us learn/g/git/git-demo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Let's add a new line in each file and create a new commit for each change.

```
\gitdemo Let us learn/g/git/git-demo (master)
$ echo "We need to revert this line" >> demoOne.txt

\gitdemo Let us learn/g/git/git-demo (master)
$ git add .

\gitdemo Let us learn/g/git/git-demo (master)
$ git commit -m "adding an invalid line to demoOne file"
[master 3ef3679] adding an invalid line to demoOne file
 1 file changed, 1 insertion(+)
```

```
\gitdemo Let us learn/g/git/git-demo (master)
$ echo "We need to revert this line" >> demoTwo.txt

\gitdemo Let us learn/g/git/git-demo (master)
$ git add .

\gitdemo Let us learn/g/git/git-demo (master)
$ git commit -m "adding an invalid line to demoTwo file"
```

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit f40d19d45ebfc0d75e0d186d4223465239d61fe9 (HEAD -> master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:51:49 2022 +0530

    adding an invalid line to demoTwo file

commit 3ef367977117172a4bcf3aa42d71e83c032d6d25
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:51:25 2022 +0530

    adding an invalid line to demoOne file

commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file
```

This is how the commits look like now

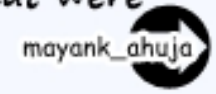
2 new commits added in our local repo.

We are actually now 2 commits ahead of the remote.

git reset --soft <commit id>

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git reset --soft ca54bc9c0486922e6683eaf8442d1e9dea85afb9

\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (HEAD -> master, origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file

\gitdemo Let us learn/g/git/git-demo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   demoOne.txt
        modified:   demoTwo.txt
```

What actually happened?
As the name suggests, reset command is used to reset/undo the changes.
It actually moves the HEAD pointer. Earlier the HEAD was pointing to some 'f40d1..' commit, but now we actually told git to reset the pointer to commit 'ca54bc..'. It actually removed all the commits after 'ca54bc..' ,
in our case 2 commits. 'Soft' flag tells git to not delete the changes that were part of the removed commits and keep them staged(green).



As we saw that after the 'soft' reset, the changes were Staged.
Now let's create a new commit, it will commit the 2 files that were reverted in the soft reset.
Added those 2 files as part of new commit '85221..'. HEAD pointer updated.

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git commit -m "adding back the 2 reverted files"
[master 852215e] adding back the 2 reverted files
 2 files changed, 2 insertions(+)

\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit 852215eedecd486eb6ec897fb2f6a1b1c4c9975c (HEAD -> master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 22:13:38 2022 +0530

    adding back the 2 reverted files

commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file
```

- You can either pass a commit id to perform reset operation, or you can also give the reference wrt HEAD (e.g HEAD~1 -> reset to 1 commit after current HEAD, HEAD~2 -> 2 commits after HEAD etc.)
- When you perform reset, it always undo all commits after specified commit, and not just that commit or a specific range.

git reset --mixed <commit from HEAD>

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git reset --mixed HEAD~1
Unstaged changes after reset:
M       demoOne.txt
M       demoTwo.txt

\gitdemo Let us learn/g/git/git-demo (master)
$ 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:   demoOne.txt
        modified:   demoTwo.txt
```

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (HEAD -> master, origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file
```

What actually happened?

When you pass the flag 'mixed' (which is default), it behaves similar to 'soft' in the sense that the changes are not lost from the local but are unstaged. So, you need to add them again if you need them back before doing commit or may be stash them. Here we moved 1 commit ahead from prev HEAD. Or in other words removed all commits after HEAD-1

git reset --hard <commit from HEAD>

As in the last change we did soft reset, let's add back the unstaged files and also commit them. Added a new commit '8b2ea..'. HEAD pointer updated.

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git add .
```

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git commit -m "adding back again the 2 reverted files"
```

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit 8b2ea3baf12bc627b46a927e732d824d55acc298 (HEAD -> master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 22:14:30 2022 +0530

    adding back again the 2 reverted files

commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file
```

Now, when we use the flag 'hard' in the reset command. It resets the commit but also removes all changes from local which where part of those commit(s).

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git reset --hard HEAD~1
HEAD is now at ca54bc9 adding demoTwo file

\gitdemo Let us learn/g/git/git-demo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Working directory is clean.

git revert <commit id>

This is the current state, we are in sync with the remote repo.

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (HEAD -> master, origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file
```

Now let's see what revert does.

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git revert ca54bc9c0486922e6683eaf8442d1e9dea85afb9
[master 5787afb] This is a Revert command demo ~ Reverting "adding demoTwo file"
 1 file changed, 1 deletion(-)
 delete mode 100644 demoTwo.txt
```

git revert command sounds similar to reset, and yes both are like performing the undo operation, but reset was much cleaner that revert.

Why so? When you perfrom the revert operation, it will undo the commit (or range of commits) but it also adds extra commits to perform this revert.

So, basically git history would have the commit that had the change plus a commit which was used to revert the change.

Let's see that in the next slide, what happened when we fired the revert.

When we fired the revert command in prev slide, it actually asked to add a commit message for the revert we are going to perform.

```
This is a Revert command demo ~ Reverting "adding demoTwo file"

This reverts commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is up to date with 'origin/master'.
#
# Changes to be committed:
#       deleted:    demoTwo.txt
#
~
~
~
~
~
~
```

With revert we can actually undo any specific commit or range of commits. With reset it was resetting everything after the specified commit. We can't specify a range there.

```
\gitdemo Let us learn/g/git/git-demo (master)
$ git log
commit 5787afbe43584d7e583b5689649c1cf1acfe526d (HEAD -> master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 22:15:33 2022 +0530

    This is a Revert command demo ~ Reverting "adding demoTwo file"

    This reverts commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9.

commit ca54bc9c0486922e6683eaf8442d1e9dea85afb9 (origin/master)
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:58 2022 +0530

    adding demoTwo file

commit 38630815df479ac0443ee344706df54e354b4885
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:39:09 2022 +0530

    adding few lines to demoOne file

commit a28a1efde08ac886d7160bea3b085884025b7a1d
Author: mydemo222 <email@example.com>
Date:   Tue Apr 26 21:38:20 2022 +0530

    adding demoOne file
```

A new revert commit has been added.

Now, do a 'git push' if you want this change to appear on the remote as well

So, to summarize -

'git revert' - does the undo operation but it creates a new commit that undoes the changes from a previous commit or range of commits.
So, it doesn't modify existing history. No existing commit is removed.
It just adds new commits. Sounds 'safe' as history is not rewritten.

'git reset' - it also does the undo without adding any new commit.
But when we do reset we are actually moving the pointer based on commit we want to reset to, so yes the existing history will get changed.
It is little complicated and can be risky if not used carefully.

Extras -

to revert range of commits
`git revert --no-commit HEAD~3..HEAD`
(This will actually revert the 3 commits from HEAD)
(You can also use --no-commit option with revert. The --no-commit option tells git to do the revert, but do not commit it automatically. You can perform commit later using 'git commit').

That's it. Thank you.