Git Reset Demo

Created by Guy Van den Nieuwenhof
https://github.com/guyvdn

Initialize Working folder

Initialize git

```
$ git init
```

Create an alias to quickly show history

```
$ git config alias.lo "log --oneline"
```

Create a *Green build*'

Create a new file

```
$ echo green build > file
```

Stage and Commit the file

```
$ git add -A
$ git commit -m "green build"
```

View history

```
$ git lo
```

Create a *Red build*

Change the file content

```
$ echo red build > file
```

Commit changes

```
$ git commit -a -m "red build"
```

View history

```
$ git lo
```

Start fixing the build

Change the file content

```
$  echo partial fix > file
```

Stage the changes

```
$  git add -A
```

Start final fix

Change the file content

```
$ echo final fix > file
```

View status and diff

```
$ git status
$ git diff
```

Somebody wants you to push the *green build*

We don't want to loose our changes in staging and working folder

```
$ git lo
$ git reset --soft <ref green build>
```

View history, status and diff

```
$ git lo
$ git status
$ git diff
```

History has changed, staging and working folder did not.

Commit of green build can now be pushed to origin.

You want to review your changes against *green build*

We don't want to loose our changes in working folder but we don't care about the staged version

```
$ git lo
$ git reset --mixed <ref green build>
```

View status and diff

```
$ git status
$ git diff
```

Staging has changed to *green build* but working folder is the same. Diff will now show *final fix* agaisnt green build.

Final fix is complete

Commit *final fix*

```
$ git commit -a -m "final fix"
$ git lo
```

Someone else fixed the issue.

We need to discard our changes so that they don't get pushed.

We also need to reset our working folder.

```
$ git lo
$ git reset --hard <ref green build>
```

Verify the changes

```
$ git lo
$ git status
$ more file
```

All trees are now back at *green build*