Local Git Repo - #5 Misc

#git-workshop

Show

Used on a commit, `git show` will display exactly what changes were made.

```
$ git show # Shows changes of most recent commit
$ git show 331d681 # Shows changes of a given commit
```

Diff

`git diff` shows changes. This can be used in a variety of different contexts. Here are a few common use cases.

Unstaged changes

If there are unstaged changes, we used to use `git status` to verify that git sees the changes. `git diff` will show us the actual content of what has been changed.

```
$ echo "spurious change" > pangram1.txt
$ git status # Shows there's a change in pangram1.txt
$ git diff # Shows what change we have made
```

Changes between commits

`git diff` can show the changes made between two different commits. If only one commit id is specified, it will be compared against the current HEAD.

```
$ git diff 331d681 1c2db6b
```

Grep

There's plenty of ways to customize this command to do more advanced stuff. But `git grep` is already surprisingly powerful search tool Simply specify the piece of text you wish to search for, and you'll be given a list of files in which it occurs along with its context.

```
$ git grep "dog"
pangram1.txt:The quick brown fox jumps over the lazy dog
```

Relative References

Tilde `~` is used to look backwards through history on the "first" parent branch. It can be used for multiple steps, depending on the number after the tilde.

Caret `^` is used to look backwards one step. It can only ever be used to look back one step, but a number can be supplied to determine which parent branch to look back on.

Tilde and caret can be chained together. PaulBoxley.com – Git caret and tilde provides a good example: