

7 Git

Techniques and Shortcuts

a quick guide by @meakcodes

Technique 1:

Git Commit

I assume that you already know how to use:

```
git-demo $ git add .  
git-demo $ git commit -m "hi mom"  
[master d6b7949] hi mom  
1 file changed, 1 insertion(+)  
git-demo $ █
```

But theres actually a better way how to do it:

```
git-demo $ git commit -am "that was easy!"  
[master 4bc37a8] that was easy!  
1 file changed, 1 insertion(+)  
git-demo $ █
```

Just get rid of the “**git add .**” and “**git push**”
and just type * **git commit -am** “ “ *

note: new files will be not tracked.

Technique 2:

Git stash

Let's say you have implemented some code, but you don't want to share it yet. You just want to save it without publishing it for now.

That's where git stash comes in:


```
git-demo $ git stash save coolstuff
```

the code now will be hidden and not visible, it's like it never existed.

example: it will be saved under the name “coolstuff”

Then when you're ready to use it again:

use git stash list to view a list of all stashes



```
git-demo $ git stash list  
stash@{0}: On master: coolstuff  
git-demo $
```

Then choose your respective stash:

choose the stash with the current index

```
git-demo $ git stash apply 0
```

Technique 3:

Git log

Imagine you want to view the latest commits from the past. You might currently be using '**git log**' for this.

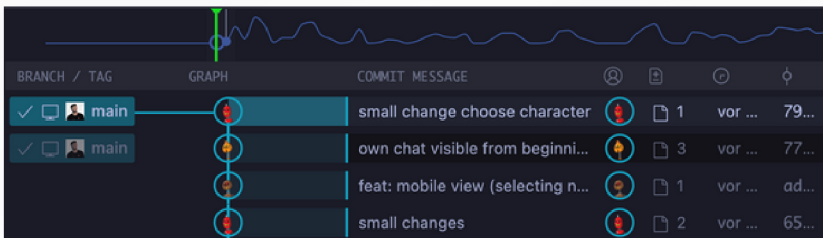
```
fireship $ git log
```

But that's actually not a great idea because it's hard to follow due to all the information being so unstructured: *it will look like this!*

```
commit a8db8aecd3fc5912fb611923d6f0b397fa3a6a1f (orig  
in/master, origin/HEAD, master)  
Author: Jeff Delaney <hello@fireship.io>  
Date:   Fri Sep 3 15:26:31 2021 -0700  
  
    add dispatch to gh actions
```

Just use the VsCode Extension **GitLens**:

there so many features, definitely give it a try!



Technique 4:

Git revert

Consider a scenario where you need to switch to a different commit because you made a mistake and want to revert to a specific section of your commit history:

Use **git log** to show the last commits and copy that commit id: ↴

```
commit d61bb60055c72d74345abe675bcb169fc0e69dc8
Author:
Date:   Wed Oct 13 12:30:43 2021 -0500

    Message 2
```

Now, enter the command '**git revert**' followed by the copied ID from the last commit:

```
Message 2
```

```
useful-custom-react-hooks (main)
$ git revert d61bb60055c72d74345abe675bcb169fc0e69dc8
```

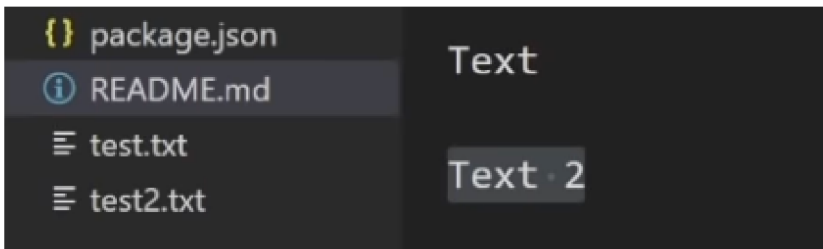
Using '**git revert**' does not delete your previous commit. Instead, it generates a new commit that effectively undoes the changes made in the specified commit. It allows you to selectively undo a particular commit without affecting the rest of your commit history.

Technique 5:

Git log -S “ ”

Picture a situation where you recall having specific words in your code, but now you can't locate them. You wish to investigate where those words are currently within your codebase.

For example our **README.md** file looks like this:



Let's find out where '**Text 2**' was changed and see who made those modifications.

Just use **git log -S “ ”** and in our example with “Text 2” *you will then all commits related to “Text 2”*

```
useful-custom-react-hooks (main)
$ git log -S "Text 2"
|
```

Technique 6:

Git branch --prune

Imagine you have an old branch that you no longer use, but it still exists locally. Learn how to completely delete it:

To view the current and last branches type: **git branch -vv**

```
useful-custom-react-hooks (main)
$ git branch -vv
 21-25 170b8b7 [origin/21-25] Revert "Revert "Message 2""
* main 170b8b7 [origin/main: ahead 8] Revert "Revert "Message 2""
```

As you can see, we have one **old branch** and one **main branch**. Now, we want to delete that old one!

Use **git remote update --prune** , it will delete the reference to your remote repository if the branch no longer exists.

```
useful-custom-react-hooks (main)
$ git remote update --prune
Fetching origin
From github.com:WebDevSimplified/useful-custom-react-hooks
- [deleted]          (none)      -> origin/21-25
```

If you view it again, it will say **gone**:

```
useful-custom-react-hooks (main)
$ git branch -vv
 21-25 170b8b7 [origin/21-25: gone] Revert "Revert "Message 2""
* main 170b8b7 [origin/main: ahead 8] Revert "Revert "Message 2""
```

Technique 7:

Git rebase

Imagine you working in a Team and you need to implement a feature and want to merge it later:

Create another branch, let's name it '**feature1**' for example, and then check out that branch:

```
git branch feature1  
git checkout feature1
```



```
Switched to branch 'feature1'  
* feature1  
master
```

Now let's commit something to that branch:

note: that is not the master branch!

```
touch f1  
git add .  
git commit -m "f1 added"
```

```
touch f2  
git add .  
git commit -m "f2 added"
```

```
touch f3  
git add .  
git commit -m "f3 added"
```

Lets go back to the Master Branch:

```
git checkout master
```



```
Switched to branch 'master'  
feature1  
* master
```

Now, simply use rebase to integrate your features into the master branch.

Simply use '**git rebase**' with your current branch, let's say '**feature 1,**' and indicate the branch where you want to merge it. In our example it's the master branch.

```
git rebase feature1 master
```



your feature branch to master

Before you go!

Please leave a positive review on Gumroad!

good luck & keep learning!