#5 Other Cool Stuff to Know

#advanced-git

Get Help

Want to learn about a Git command? Having trouble sleeping at night and need something to read? If any of the above apply to you, you can read the Git manual pages for a command with the following:

```
$ git help <command>
```

For example: `git help add` shows the manual for `git add`.

Hard Reset

By default, reset will preserve the state of your files. If you want to reset the contents of your repo to match the new head, add the `--hard` flag. For example, to reset everything to the current `HEAD` (without changing the location of any branches):

```
$ git reset HEAD --hard
```

Note that this will not clear out untracked files – use `git clean` for that instead (#1 Daily Tools).

Global `.gitignore`

It's possible to set a default `.gitignore` for all your repositories. This is useful if you know you never want to include a certain type of file (for example, `.DS_Store` on macOS).

```
$ git config --global core.excludesfile ~/.gitignore_global
```

Any `.gitignore` in a Git repo will be applied on top of this global `.gitignore` file.

- Ignoring files - User Documentation
- Some common .gitignore configurations · GitHub

The Fixup Commit and Autosquash

Sometimes, it can be difficult to think of a commit message for a tiny change (or even a large change!).

If you're going to just squash the commit later and discard the message, you can take advantage of the `fixup` command in rebase. Even better, you can create a commit in a way such that it will automatically be fixed up.

```
$ git commit --fixup=COMMIT # Specify which commit this should be combined with
```

This will create a new commit with a dummy message. When you do the following rebase command:

```
$ git rebase master --interactive --autosquash
```

Any "fixup" commits will be absorbed into the designated commit, and you don't even need to write a commit message!

Set Upstream

If your repository uses a single remote (like `origin`) and you're tired of typing in `git push origin master`, you can set a default upstream. That way, `git push` and `git pull` will default to the remote you set for the current branch.

For example, this makes it so that whenever you do `git push` on the master branch, it will do a `git push origin master`.

```
$ git push origin master --set-upstream
```

An alternative to this is Bash aliases (see below).

Bash Aliases

If you're tired of typing out long Git commands or if you use a certain command frequently, it's a good idea to create an *alias* for that command. This is a short name that you can type that will refer to the full command – like a keyboard shortcut, but with text.

For example, if you frequently use interactive rebase, you could have the following (in your `~/.bashrc` file:

```
alias grbim='git rebase master --interactive'
```

- Bash Aliases

SSH Keys

If you're tired of typing in your password every time you push or fetch, then SSH authentication is for you. Check out the following instructions from GitHub: Connecting to GitHub with SSH - User Documentation

You'll need to upload your public SSH key to every cloud provider you use (GitHub, Bitbucket, etc.). But afterwards, you'll be able to "magically" push and pull without typing in any passwords.

Other common shortcuts

- To create a branch and switch to it:

```
$ git checkout -b branch_name
```

- To add all modified (but not untracked) files:

```
$ git add --update
```

- To add all files:

```
$ git add --all
```

Hub for GitHub

If you use GitHub a lot, you may find it useful to interact with it from the command line using the Hub tool.

- GitHub - github/hub: hub helps you win at git.

Dry Run

Many commands will let you do a “dry run” (check the help pages to be sure) with the `--dry-run` flag. This will tell you what the command will do without actually doing it.

For example, to see what `git add` would add:

```
$ git add *.txt -n # What text files will get added?
add 'hello.txt'
```