GIT WORKFLOW CHEATSHEET

From Git & GitHub Workflow for Visual Learners by Marc Nischan

Here are three scenarios and the workflow for each. Type the lines beginning with "$" in Terminal.

Working locally in Git (assuming you have already installed Git)

In Terminal, find the directory you want to put your project in. For this example, I'm going to put it in my Home folder in another folder called "my-repo". Then initialize the repo.

```
$ cd ~/
$ mkdir my-repo
$ cd my-repo
$ git init
```

Now you can start the branch, create, stage, commit, merge process. I'm going to call the branch "my-feature."

```
$ git checkout -b my-feature
```

Create some content, for this example I'll say I created a file called index.html and I'm ready to commit it.

```
$ git add index.html
$ git commit -m "Adding code for the index page"
```

Now merge it back in to the master branch in preparation for adding the next feature.

```
$ git checkout master
$ git merge my-feature
```

Now you're ready to repeat the cycle again for your next bit of work.

Cloning your GitHub repo down to your local development environment

On the page for your GitHub repository, copy the clone URL from the bottom of the right-hand column. Then in the Terminal, find the directory that you want your project to live in. Remember that when you clone, a folder is created with the project name. For this example I am going to put the project in a "workspace" folder in my "home" folder.

```
$ cd ~/workspace
$ git clone https://github.com/Your-user-name/your-project.git
$ cd your-project
```

Now you are in the cloned repo on your computer, and you can start the branch, create, stage, commit, merge process. I'll use the same conventions as in the previous example.

```
$ git checkout -b my-feature
```

Now create some content, like a file called index.html.

```
$ git add index.html
$ git commit -m “Adding code for the index page”
```

Now you need to push it up to your GitHub repo.

```
$ git push origin HEAD
```

(optionally you could $ git push origin my-feature)

Now you go up to your GitHub repo and create a pull request, and approve and merge it. After that, you want to pull it back in to your local development environment by executing a “pull.”

```
$ git checkout master
$ git pull origin master
```

Now you’re ready to create a new branch and repeat the cycle again for your next bit of work.

Forking a GitHub repo

From your logged in GitHub account, navigate to the repo that you would like to fork, and click the “Fork” button in the upper right corner. Enjoy the nice amination while GitHub copies the repo.

Now, from your newly created fork, copy the clone URL from the bottom of the right-hand column and paste it in to clone the repo down to your computer.

```
$ git clone git@github.com:repo-you-forked/whatever-the-project-is.git
```

Now you can push and pull from your repo, but you will also need to pull from the original repo, so add it as a remote called “upstream.”

```
$ git remote add upstream git@github.com:original-repo/whatever-the-proj-
ect-is.git
```

You’re ready to start the branch, create, stage, commit, merge cycle.

```
$ git checkout -b my-feature
```

Now create some content, an index.html file in this example, and stage and commit it.

```
$ git add index.html
$ git commit -m “Adding code for the index page”
```

Now you can push it to your origin remote on GitHub. Push it using HEAD or the branch name.

```
$ git push origin HEAD or $ git push origin my-feature
```

Now go to your GitHub repo and click the big green button to create a pull request. A PR will be created on the repo that you forked. You will have to wait for the owner to merge the content, but in the mean time you can keep working on your next feature by creating a branch from the feature you just finished.

But before you do that, grab the latest version of the master branch from the project you forked and merge it in to your local master branch.

```
$ git checkout master
$ git pull upstream master
```

Now merge whatever new stuff the upstream remote gave you into the branch you just finished.

```
$ git checkout my-feature
$ git merge master
```

And begin your next feature from there.

```
$ git checkout -b my-next-feature
```
You are ready to start the cycle again.

Some more useful commands

```
$ git status
```
See the branch you're on, and relevant status messages.

```
$ git remote -v
```
See all of the remotes associated with your repo and their URLs.

```
$ git branch
```
See all of the branches in your local repo.

```
$ git branch -D my-feature
```
Delete a branch in your local repo after you no longer need it ("my-feature" in this example).

```
$ git add . or $ git add --all
```
Stage all of your unstaged files at once.

```
$ git reset
```
Undo your last commit, but leave all of the work unchanged.

```
$ git reset --hard
```
Undo the last commit, and erase all of the work back to the commit you made before.