

Enrique Perez

GitHub in a Nutshell



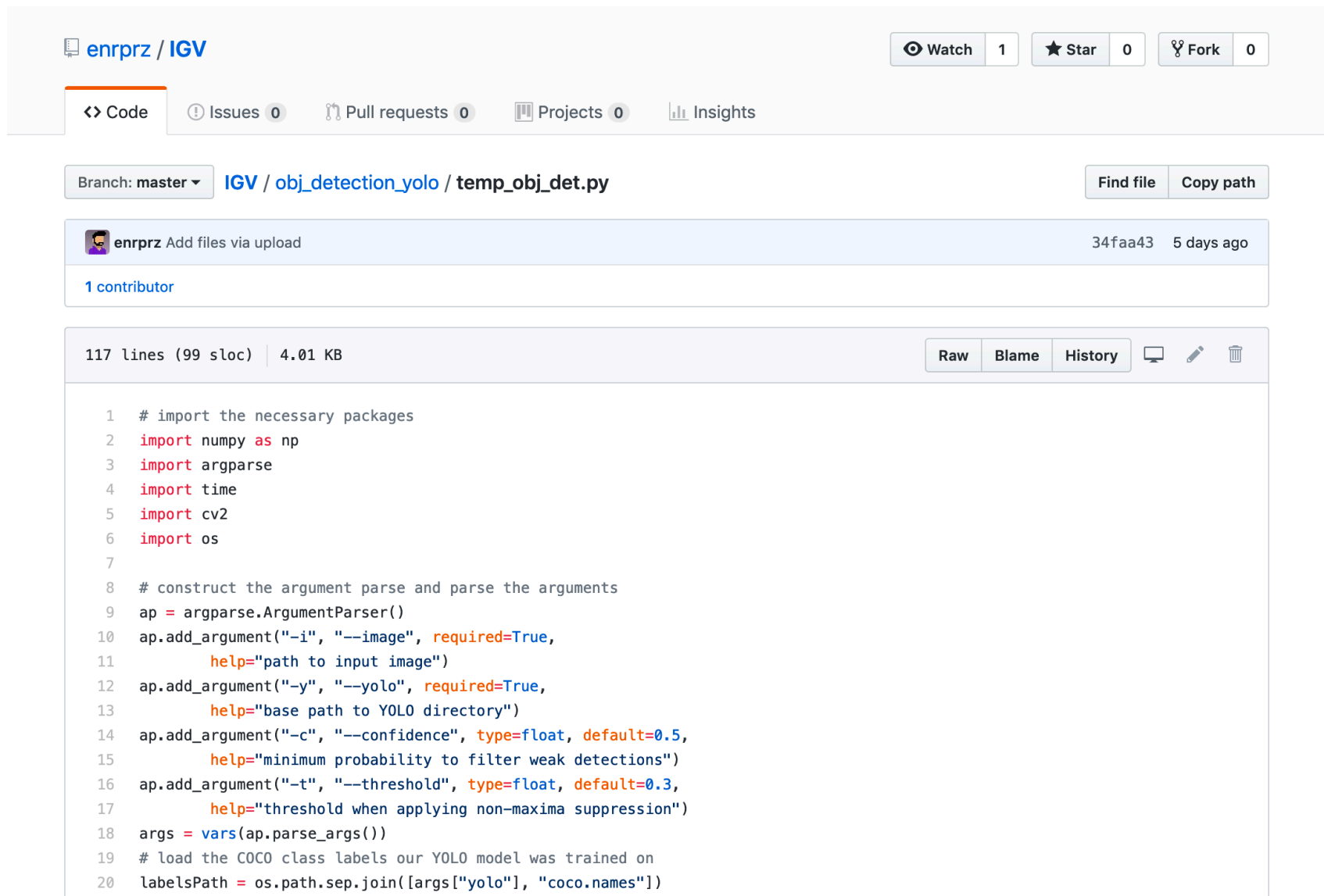


Image 01. Portraits the default view of a users repository with the default website structure.

Why GitHub?

GitHub has become one of most popular sites used for developers, and organizations that want to keep track of files. Although the website is highly used by programmers, it reaches a broad audience thanks to the use of multiple files ranging from programming languages to word documents, and spreadsheets. To understand GitHub we have to be aware of a couple things. GitHub, and 'Git' are 2 very different things. GitHub is a website that uses the program 'Git' to work. 'Git' is a linux command line program used as a version-control system. Think of it as a way of keeping track of 'who does what'. GitHub makes use of spaces called 'repositories', repo in short. These are places where developers can upload their code, and share it amongst friends, peers or co-workers within the organization. Think of it as a shared 'dropbox' where anyone can contribute, and update the same project files. The following command is an example of how git works in the linux OS environment. This will clone (copy) a repository on your local machine from a url containing the <user>, and <repo> that we want:

```
user@host: ~$ git clone https://github.com/<username>/<repo>.git
```

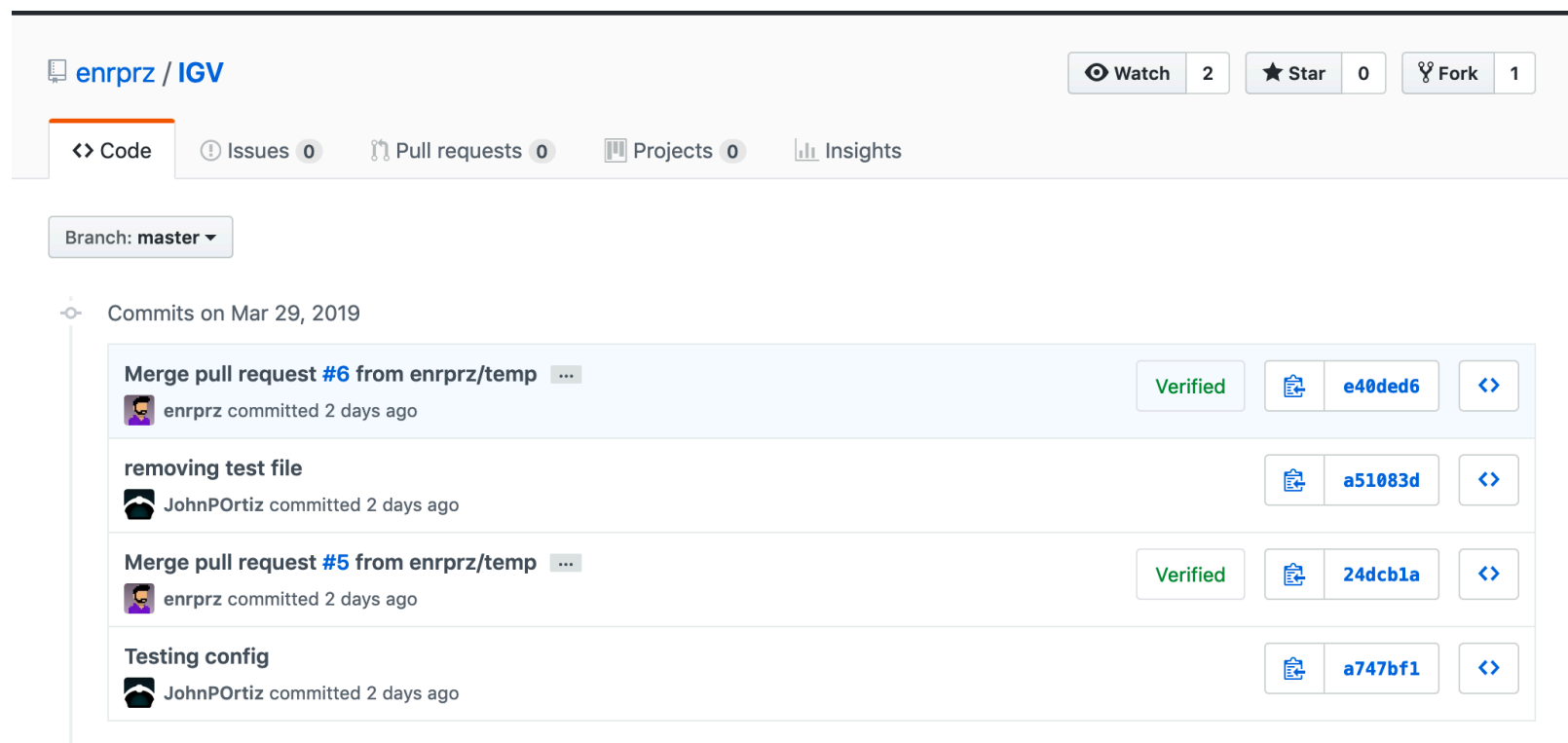


Image 02. Shows examples of a user submitting a 'pull request', and getting approval from the admin to merge with the 'master' branch.

Why is GitHub necessary?

In essence what GitHub does can be done with other websites, what makes GitHub special? GitHub shines thanks to some special features called 'pull request', and 'fork'. These features are very powerful because they allow for simultaneous work on different files without affecting the 'main' ones. GitHub uses something that is called 'branches' to keep the integrity of files throughout many changes. The point is to keep a 'master' branch for final products, and multiple sub-branches to make changes. People can modify files on the sub-branches, and once these changes are final or tested they can 'commit' the update they are working on. But how does anyone get to apply the updated files to the 'master' branch? At that point the user has to submit a 'pull request'. A 'pull request' goes to the administrator of the branch (like a supervisor or a team leader), and it awaits for a review. The administrator will then look at the changes, and if accepted then it gets added to the 'master' branch. This keeps a high level of security by adding protection, and allowing for checks before anything becomes official. The following commands are used to add the file we are updating, update the file (also called 'commit'), and submit the changes to the 'temp' branch on the repo:

```
user@host: ~$ git add file_01.docx
user@host: ~$ git commit -m "Added cool pictures"
user@host: ~$ git push origin temp
```



Pro tip!

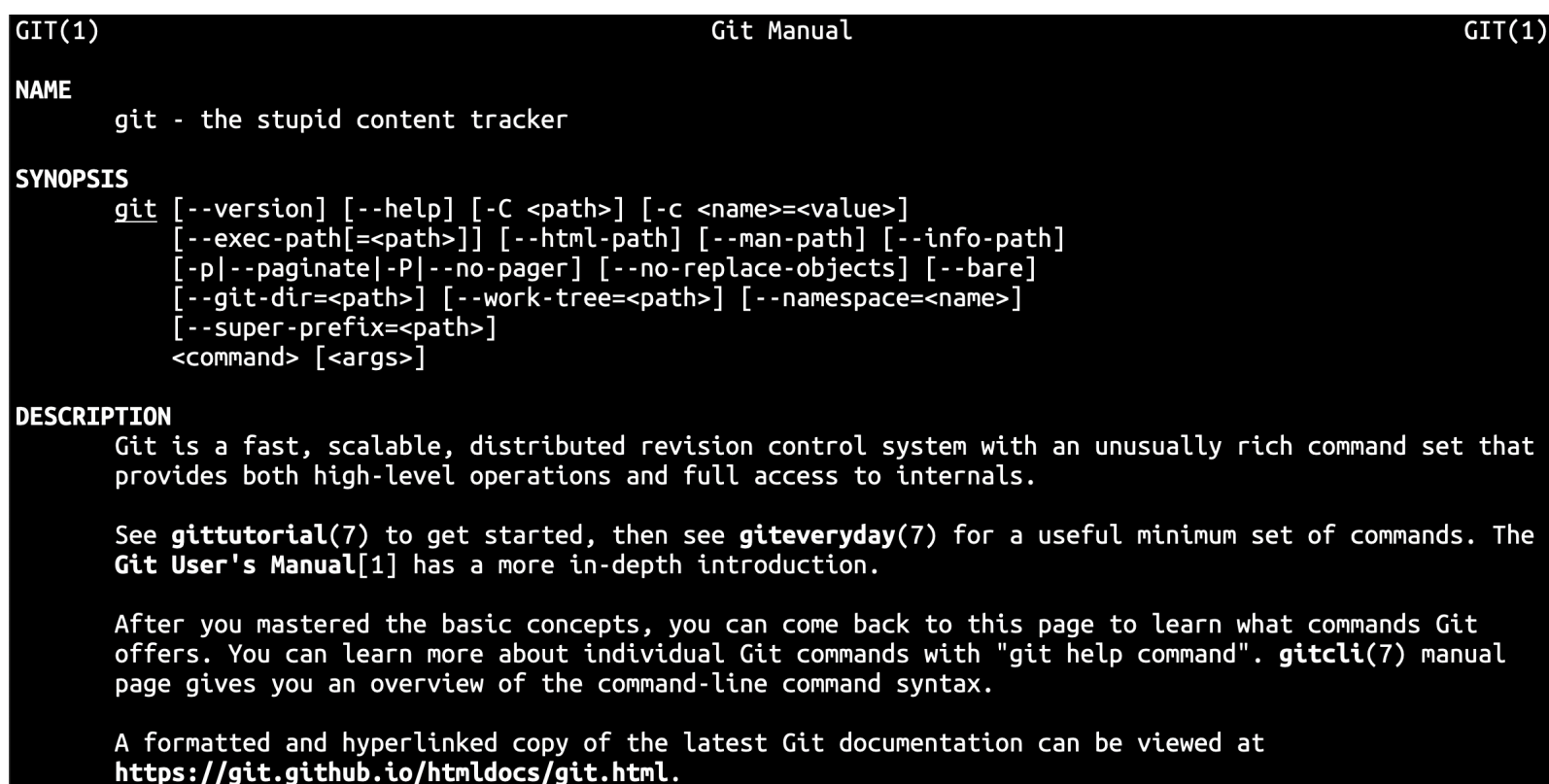
If coding is not your thing, you can also remove, add, and update the documents directly on the website GUI.

What If I don't use GitHub?

The short answer: don't worry! GitHub is not mandatory to have. However, many organizations use it. It is very likely that you will have to use it at your workplace. Learning how to use GitHub is not only to get experience, but also to show companies your 'portfolio'. GitHub nowadays is used as a way to showcase a users coding abilities to potential employers. You will be able to get more attention from a recruiter if they see you have been working on similar projects that they do, and all the contributions you have done to another users repo! So not only will it make your life easier to learn it, and collaborate with your coworkers, but it will also show the quality of updates that you submit to a repo. This also comes with the added benefit of learning a little bit of code. If you are a programmer it would be a good choice to learn the how to use 'git' on the terminal. If you are not a programmer you can still get exposed to more secure, and efficient ways of working by using a terminal instead of the typical GUI.

If you would like to know how to take full advantage of the git command you can always type the following command to take a look (assuming you are using UNIX/macOS or Linux):

```
user@host: ~$ man git
```



```
GIT(1)                                Git Manual                                GIT(1)

NAME
    git - the stupid content tracker

SYNOPSIS
    git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p|--paginate|-P|--no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      [--super-prefix=<path>]
      <command> [<args>]

DESCRIPTION
    Git is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

    See gittutorial(7) to get started, then see giteveryday(7) for a useful minimum set of commands. The Git User's Manual[1] has a more in-depth introduction.

    After you mastered the basic concepts, you can come back to this page to learn what commands Git offers. You can learn more about individual Git commands with "git help command". gitcli(7) manual page gives you an overview of the command-line command syntax.

    A formatted and hyperlinked copy of the latest Git documentation can be viewed at
    https://git.github.io/html/docs/git.html.
```

Image 03. Shows a snippet of the git command manual page with examples on how to use various arguments.

How do I start?

Glad you ask! You can start by creating an account on GitHub, and creating your first repo. After that you will need to see if you have **git** installed on your computer, you can try the following command:

```
user@host: ~$ git --version
```

If you got something like: `-bash: git: command not found` then that means it is not installed. You can install git with the following command (make sure you type your password, and choose 'y' when prompted):

```
user@host: ~$ sudo apt-get install git
```

Great! Next we are going to go to the directory where you wanna have your work on, and create an empty local Git repo on your computer:

```
user@host: ~$ git init
```

Lets test if things work by cloning your GitHub repo with your local repo, and setting ourselves on the 'master' branch:

```
user@host: ~$ git clone https://github.com/<yourUsername>/<repo>.git
user@host: ~$ git checkout master
```

Next we are going to create a file, add a message, and upload it to the repo.

```
user@host: ~$ touch myFile.txt
user@host: ~$ git add myFile.txt
user@host: ~$ git commit -m "This is my first file"
user@host: ~$ git push origin
```

Hurray! You just uploaded your first file to GitHub. There are plenty of combinations to add to the git commands, and an even greater way of using them depending on your OS. Nevertheless, we encouraged you to read about it, practice, and play with the arguments for the commands. This is by no means an extensive tutorial, but more of an intro to get you started with GitHub so you can start collaborating, and sharing your projects with other developers.

Citations:

“Build Software Better, Together.” *GitHub*, github.com/.

Chacon, Scott, and Ben Straub. *Pro Git*. Apress, 2014.

Git, git-scm.com/.

Negus, Chris, et al. *Linux Bible (8th Edition)*. John Wiley & Sons, 2012.