Git and GitHub

In this section you will:

- create a GitHub account
- create your own fork of a repository
- create a new Git branch
- edit and commit a file on GitHub
- make a pull request
- merge upstream changes into your fork

What is it?

Git is a source code management system, designed to support collaboration.

GitHub is a web-based service that hosts Git projects, including Django itself: https://github.com/django/django.

The key idea in Git is that it's *distributed*. If you're not already familiar with version control systems, then explaining why this is important will only introduce distinctions and complications that you don't need to worry about, so that's the last thing I will say on the subject.

Set up a GitHub account

- sign up at GitHub if you don't already have an account

It's free.

Some basic editing on GitHub

Forking

- visit https://github.com/evildmp/afraid-to-commit/

You can do various things there, including browsing through all the code and files.

- hit the **Fork** button

A few moments later, you'll have your own copy, on GitHub, of everything in that repository, and from now on you'll do your work on your copy of it.

Your copy is at `https://github.com/<your github account>/afraid-to-commit/`.

Note

About angular brackets

> Angular brackets, such as those in the line above, are used as placeholders; they do not need to be included in the command line terminal. Just write in your GitHub account name. The same applies for any angular brackets used throughout the tutorial.

You will typically do this for any Git project you want to contribute to. It's good for you because it means you don't have to sign up for access to a central repository to be permitted to work on it, and even better for the maintainers because they certainly don't want to be managing a small army of volunteers on top of all their other jobs.

Note

Don't worry about all the forks

> You'll notice that there might be a few forks of https://github.com/evildmp/afraid-to-commit; if you have a look at https://github.com/django/django you'll see thousands. There'll even be forks of the forks. Every single one is complete and independent. So, which one is the real one - which one is *the* Django repository?
>
> In a technical sense, they all are, but the more useful answer is: the one that most people consider to be the canonical or official version.
>
> In the case of Django, the version at https://github.com/django/django is the one that forms the basis of the package on PyPI, the one behind the https://djangoproject.com/ website, and above all, it's the one that the community treats as cannonical and official, not because it's the original one, but because it's the most useful one to rally around.
>
> The same goes for https://github.com/evildmp/afraid-to-commit and its more modest collection of forked copies. If I stop updating it, but someone else is making useful updates to their own fork, then in time theirs might start to become the one that people refer to and contribute to. This could even happen to Django itself, though it's not likely to any time soon.
>
> The proliferation of forks doesn't somehow dilute the original. Don't be afraid to create more. Forks are simply the way collaboration is made possible.

Create a new branch

Don't edit the *master* (default) branch of the repository. It's much better to edit the file in a new branch, leaving the *master* branch clean and untouched:

1. select the **branch** menu
2. in *Find or create a branch...* enter `add-my-name`
3. hit **Create branch: add-my-name**

Note

Don't hesitate to branch

As you may have noticed on GitHub, a repository can have numerous branches within it. Branches are ways of organising work on a project: you can have a branch for a new feature, for trying out something new, for exploring an issue - anything at all.

Just as virtualenvs are disposable, so are **branches** in Git. You *can* have too many branches, but don't hesitate to create new ones; it costs almost nothing.

It's a good policy to create a new branch for every new bit of work you start doing, even if it's a very small one.

It's especially useful to create a new branch for every new feature you start work on.

Branch early and branch often. If you're in any doubt, create a new branch.

Edit a file

GitHub allows you to edit files online. This isn't the way you will normally use Git, and it's certainly not something you'll want to spend very much time doing, but it's handy for very small changes, for example typos and spelling mistakes you spot.

1. go to `https://github.com/<your github account>/afraid-to-commit`
2. find the `attendees_and_learners.rst` file
3. hit the **Edit** button
4. add your name (just your name, you will add other information later) to the appropriate place in the file. If you're following the tutorial by yourself, add your details in the *I followed the tutorial online* section.

Commit your changes

- hit **Commit Changes**

Now *your* copy of the file, the one that belongs to *your* fork of the project, has been changed; it's reflected right away on GitHub.

If you managed to mis-spell your name, or want to correct what you entered, you can simply edit it again.

What you have done now is make some changes, in a new branch, in your own fork of the repository. You can see them there in the file.

Make a Pull Request

When you're ready to have your changes incorporated into my original/official/canonical repository, you do this by making a **Pull Request**.

- go back to `https://github.com/<your github account>/afraid-to-commit`

You'll see that GitHub has noted your recent changes, and now offers various buttons to allow you to compare them with the original or make a pull request.

- hit **Compare & pull request**

This will show you a *compare view*, from which you can make your pull request.

When preparing for a pull request, GitHub will show you what's being compared:

```
evildmp:master ... <your github account>:add-my-name
```

On the left is the **base** for the comparison, my fork and branch. On the right is the **head**, your fork and branch, that you want to compare with it.

A pull request goes from the **head** to the **base** - from right to left.

You can change the bases of the comparison if you need to:

1. hit **Edit**
2. select the forks and branches as appropriate

You want your version, the **head branch** of the **head fork** - on the right - with some commits containing file changes, to be sent to my **base repo** - on the left.

1. hit the **Pull Request** button
2. add a comment if you like (e.g. "please add me to the attendees list")
3. hit **Send pull request**

You have now made a pull request to an open-source community project - if it's your first one, congratulations.

GitHub will notify me (by email and on the site, and will show me the changes you're proposing to make). It'll tell me whether they can be merged in automatically, and I can reject, or accept, or defer a decision on, or comment on, your pull request.

GitHub can automatically merge your contribution into my repository if mine hasn't changed too much since you forked it. If I want to accept it but GitHub can't do it automatically, I will have to merge the changes manually (we will cover this later).

Once they're merged, your contributions will become a part of https://github.com/evildmp/afraid-to-commit. And this is the basic lifecycle of a contribution using git: *fork* > *edit* > *commit* > *pull request* > *merge*.

Incorporate upstream changes into your master

In the meantime, other people may have made their own forks, edits, commits, and pull requests, and I may have merged those too - other people's names may now be in the list of attendees. Your own version of afraid-to-commit, *downstream* from mine, doesn't yet know about those.

Since your work is based on mine, you can think of my repository as being *upstream* of yours. You need to merge any *upstream* changes into *your* version, and you can do this with a pull request on GitHub too.

This time though you will need to switch the bases of the comparison around, because the changes will be coming from *my version* to *yours*.

1. hit **Pull Request** once more
2. hit **Edit**, to switch the bases
3. change the **head repo** on the right to *my* version, `evildmp/afraid-to-commit`, branch *master*
4. change the **base repo** to yours, and the **base branch** to *master*, so the comparison bar looks like:

   ```
   <your github account>:master ... evildmp:master
   ```

5. hit **Click to create a pull request for this comparison**
6. add a **Title** (e.g. “merging upstream master on Github) and hit **Send pull request**

You’re sending a pull request to *yourself*, based on updates in my repository. And in fact if you check in your **Pull Requests** on GitHub, you’ll see one there waiting for you, and you too can review, accept, reject or comment on it.

If you decide to **Merge** it, your fork will now contain any changes that other people sent to me and that I merged.

The story of your work is this: you **forked** away from my codebase, and then created a new **branch** in your fork.

Then you **committed** changes to your branch, and sent them **upstream** back to me (with a **pull request**).

I **merged** your changes, and perhaps those from other people, into my codebase, and you **pulled** all my recent changes back into your *master* branch (again with a **pull request**).

So now, your *master* and mine are once more in step.