Github and Pushing

Sharing code through Git

The other way that Git enables collaboration is by making it easy to **share code**.

We've mentioned that Git is *distributed* because everybody has a full copy of the repository.

Sharing code through Git

Everybody is making changes to *their copy*, so each person accumulates changes that the others can't see yet.

At some point, hopefully when the changes are *completed and tested*, a person can share those changes with the rest of the team.

Remote repositories

Normally this is done through a **remote repository**.

A **remote repository** or just **remote** is a repo like the ones each person has on their computer, except it lives on the Internet somewhere.

Since it's on the Internet it's accessible by the entire team.

Remote repositories

To **share your code** with the rest of the team, you add a remote to your local repo and then you **push** to it.

To **retrieve code** shared by other team members, you **pull** from the remote.

Remote repositories

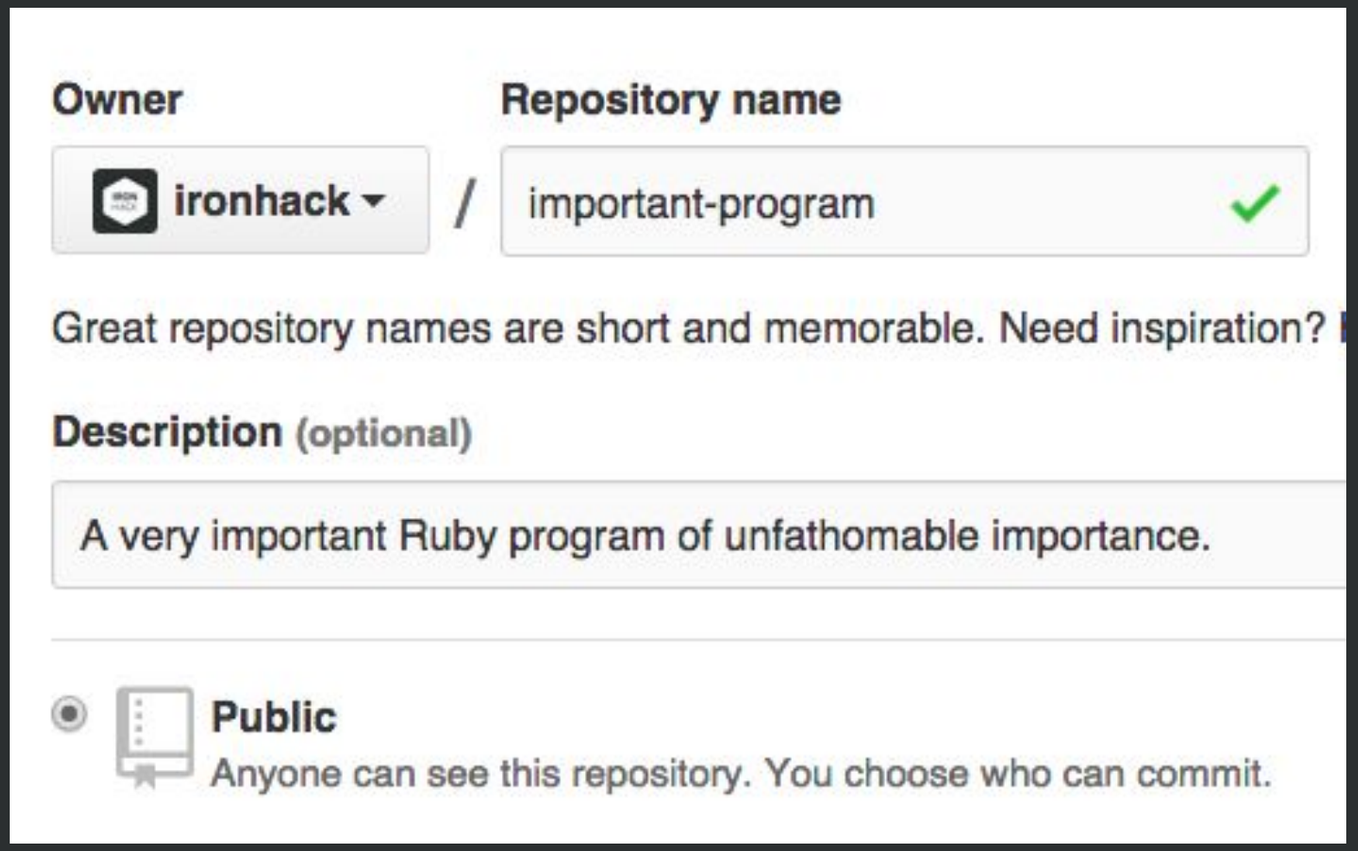
But to do all that you first have to get yourself a remote repository.

GitHub

It's very common for developers to use GitHub for their remote repository needs.

Adding a remote repository

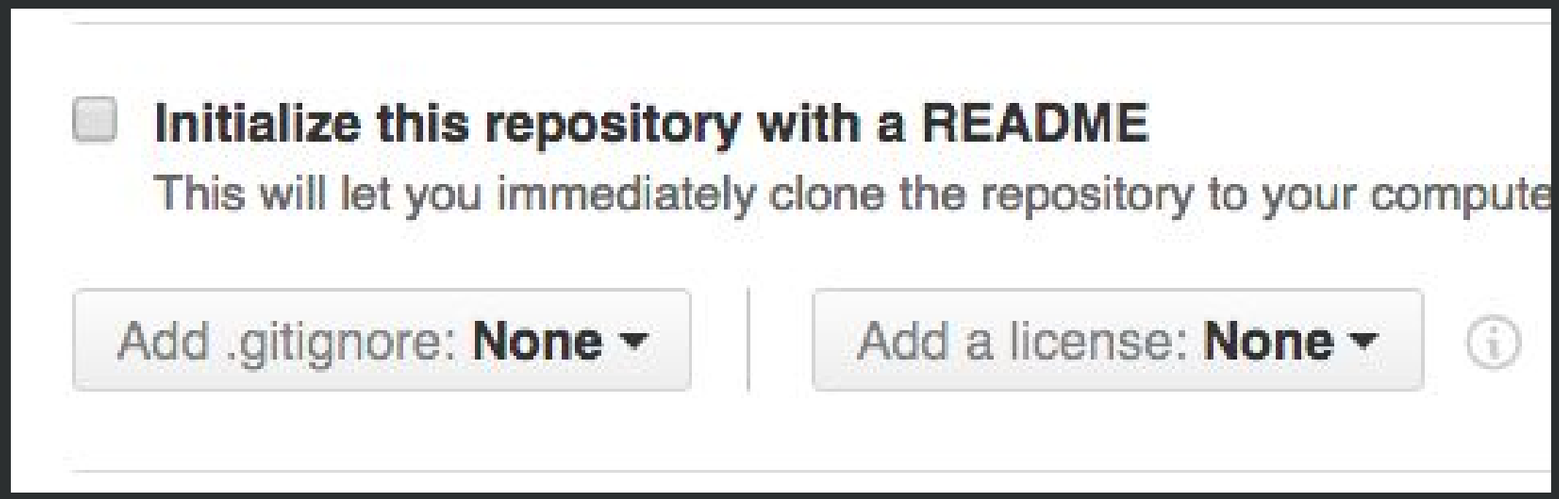
Go to your GitHub accounts and create a new remote repo called `important-program` for your important program.



Adding a remote repository

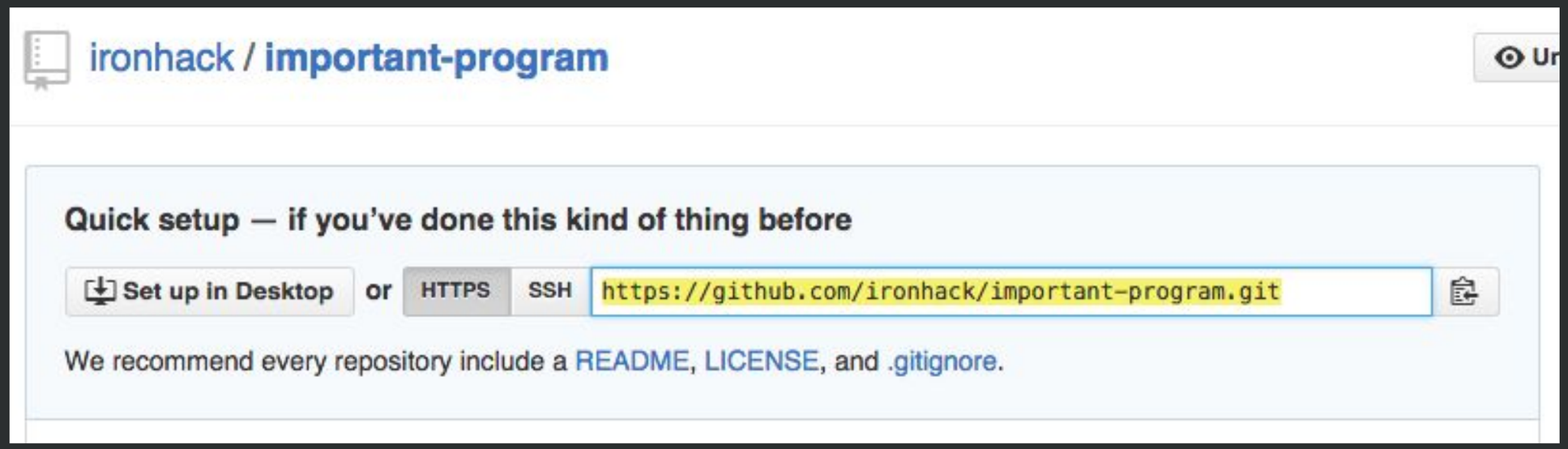
For this time, don't check the *Initialize this repository with a README* box.

We will go over why later.



Adding a remote repository

Now copy the repo's URL. That's what you need to add it to your local repo.



Adding a remote repository

To add your GitHub remote repo to your local repo, use the `git remote add` command:

```
$ git remote add REMOTENAME URL
```

Adding a remote repository

You also have to give the remote repo a name. The typical name for the main remote of your project is `origin`.

So the command we should run is:

```
$ git remote add origin https://github.com/ironhack/important-program.git
```

Adding a remote repository

Now we can run just `git remote` to see the list of remotes we've added.

```
$ git remote add origin https://github.com/ironhack/important-program.git

$ git remote
origin
```

Adding a remote repository

Add the `--verbose` or `-v` option to see what the URL is. The names you give your remotes are specific to your repo.

They don't change the name of the repository on GitHub.

```
$ git remote add origin https://github.com/ironhack/important-program.git

$ git remote -v
origin  https://github.com/ironhack/important-program.git (fetch)
origin  https://github.com/ironhack/important-program.git (push)
```

Pushing to remote repository

Now that we have our remote repository, `origin`, we can now **push** our changes to make them available for the rest of the team.

We do this with the `git push` command:

```
$ git push REMOTE BRANCH
```

Pushing to remote repository

In our case, we are pushing to the remote we called `origin`. What are we pushing? The `master` branch, of course. That leaves us with:

```
$ git push origin master
```

There's an easier way!

If don't specify the remote, it will pick `origin` by default.

If we don't specify a branch, it will pick the current branch by default (and match it with a remote branch with the same name). In our case, it will default to our `master`

```
$ git push
```

There's an easier way!

If don't specify the remote, it will pick `origin` by default.

If we don't specify a branch, it will pick the current branch by default (and match it with a remote branch with the same name). In our case, it will default to our `master`

```
$ git push origin master
```

It's the same!

Pushing to remote repository

Use the `--set-upstream` option to avoid having to specify the remote and the branch on every push in the future.

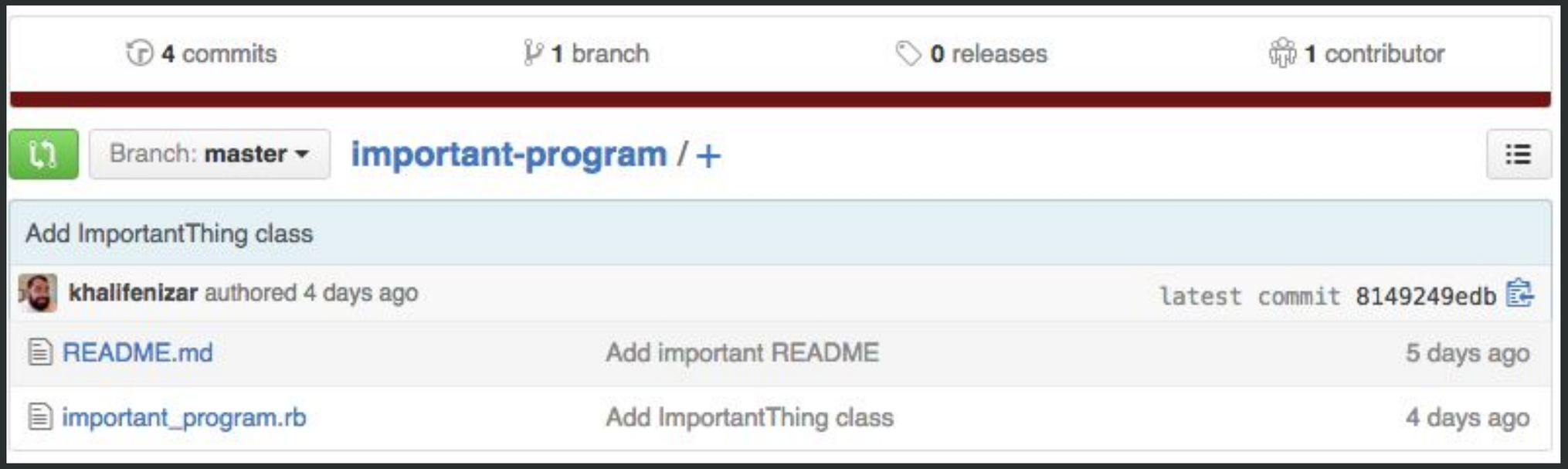
```
$ git push --set-upstream origin master
Branch master set up to track remote branch master from origin.

$ git push
```

Now you can just `git push` and it knows what to do.

Pushing to remote repository

Now if you visit the URL of your GitHub repository, you should see the current status of your project, as well as links to your list of commits.



IRON HACK

Pushing to remote repository

Essentially, a push transfers the commits you have made locally to the remote repository.
Use the `--remote` option with `git branch` to see where the remote's `master` branch is at.

```
$ git branch --verbose --remote
  origin/master 8149249 Add ImportantThing class
```

It's pointing to the same commit as our local `master` branch.

```
$ git branch --verbose
* master 8149249 Add ImportantThing class
```

Pulling from a remote repository

Now that we've learned to share our changes, we can use a similar command to retrieve the changes others have shared. Instead of a push, we now **pull** changes from the remote.

We do this with the git pull command:

```
$ git pull origin master
```

Pulling from a remote repository

Since we've already used `--set-upstream` previously to link our branch with a remote branch, we can shorten this to just:

```
$ git pull
```

Pulling from a remote repository

Of course, in our case we don't have any changes to retrieve so this doesn't really do anything for us.

Keep `git pull` in mind for when you collaborate with others later.

```
$ git pull
Current branch master is up to date.
```

Your workflow with GitHub

Generally, you should be pushing everything you do in the course to a GitHub repo.

It's good to get in the habit of making changes, adding the changes, making commits and pushing to GitHub.

Your workflow with GitHub

There are two ways to approach this:

1. You create the repo on your computer first, start writing code and add the repo to GitHub later.
2. You create the repo on GitHub first, add the repo to your computer and start writing code.

Starting a project from your computer

To start from your computer, use `git init` in a folder as we've been doing:

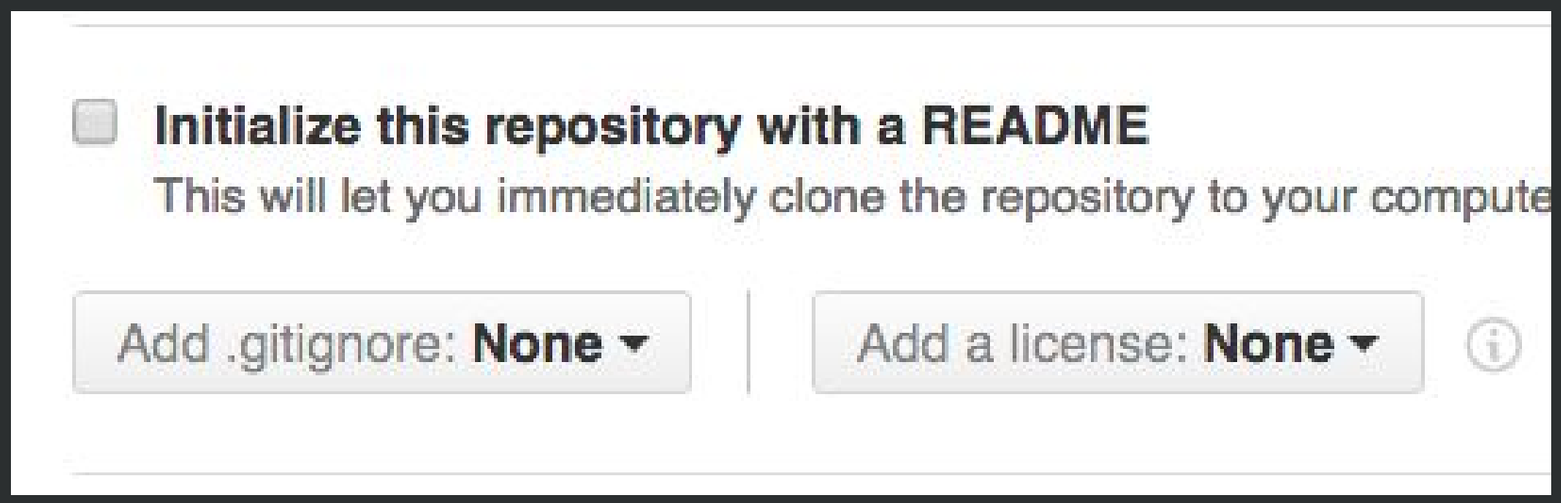
```
$ cd new_project/

$ git init
```

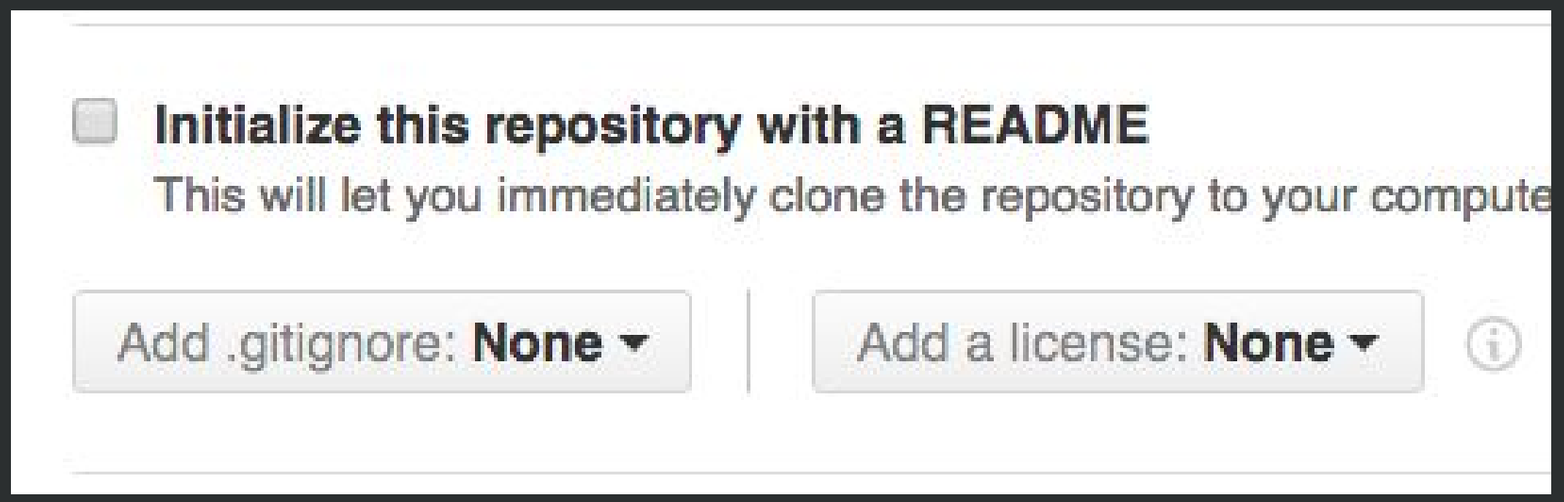
Starting a project from your computer

After you've made a few commits and you want to push, create the repo con GitHub.

Remember to not to check the *Initialize this repository with a README* box.



Starting a project from your computer



If you check it, you will have trouble when you push because it actually *creates commits*.

IRON HACK

Starting a project from your computer

Now add the GitHub remote and push:

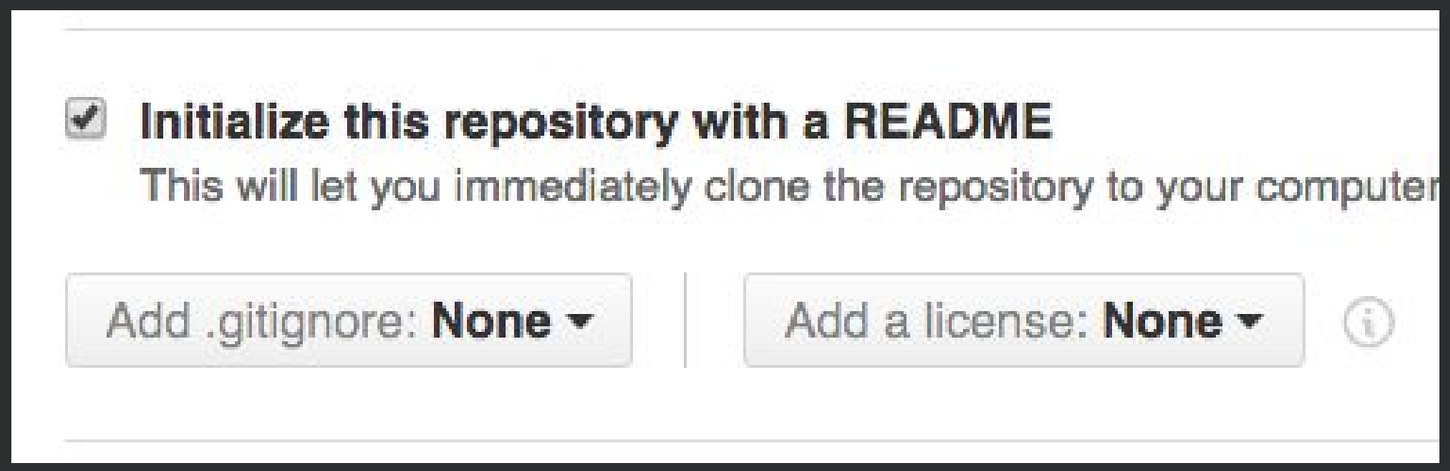
```
$ git remote add origin GITHUB_URL

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

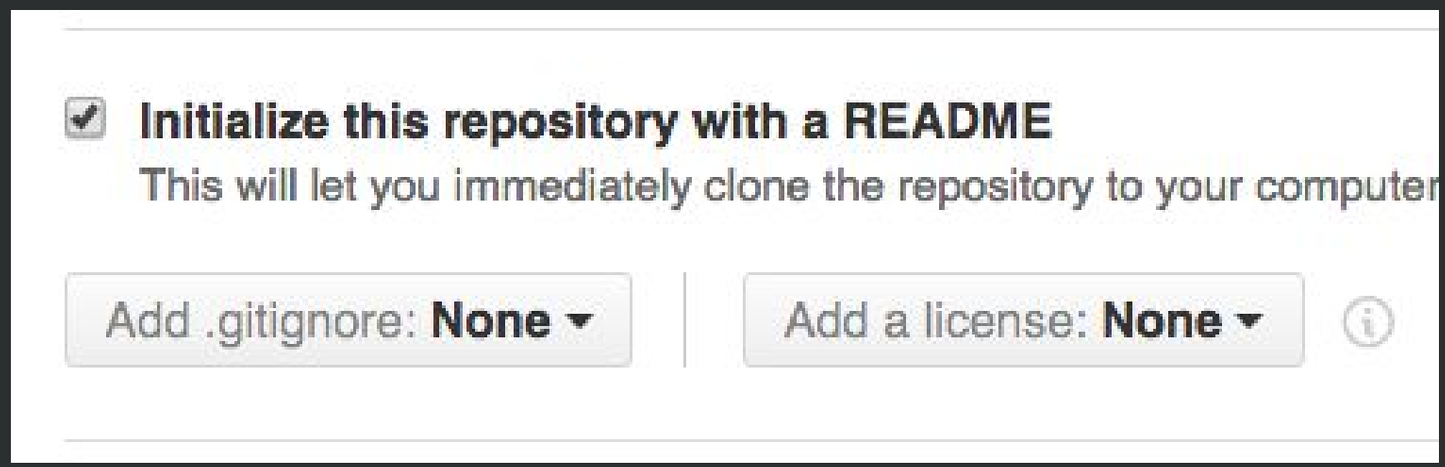
Starting a project from Github

If you create the repo on GitHub first, the procedure is a bit different.

First, you should probably check the *Initialize this repository with a README* box.



Starting a project from Github



The extra commits won't affect you because you haven't made any commits of your own yet.

IRON HACK

Starting a project from Github

Now, since you don't have a repo on your computer yet, you can't `git remote add`.

Instead you will have to do a git clone:

```
$ git clone GITHUB_URL
```

Starting a project from Github

Clone copies an existing remote repository on the computer.

```
$ git clone GITHUB_URL
Cloning into 'new_project'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0),
pack-reused 0
Receiving objects: 100% (3/3), done.
Checking connectivity... done.
```

Starting a project from Github

The `git clone` command automatically creates the folder for the project so you shouldn't create it beforehand.

```
$ git clone GITHUB_URL

$ cd new_project/
```

Starting a project from Github

You'll find that it already has the `origin` remote set up.

```
$ git clone GITHUB_URL

$ cd new_project/

$ git remote --verbose
```

Starting a project from Github

And it has already linked the local `master` branch with the remote's `master`.

```
$ git clone GITHUB_URL

$ cd new_project/

$ git branch --verbose --remote
```

Starting a project from Github

```
$ git clone GITHUB_URL

$ cd new_project/
```

Now you are ready to write your code and commit it!

Conclusion

You should make it a point to commit and push your code often so that others have the latest changes as soon as possible.

Pull often as well!

Remember to make use of branches to avoid affecting other people until your changes are ready.

Conclusion

GitHub is awesome! Get used to it.

It does way more for you than just accept your pushes.

Remember kids: **ABC**

ALWAYS BE COMMITTING