software carpentry

Version Control with Git

Remotes in GitHub

Learning Objectives

- Explain what remote repositories are and why they are useful.
- Clone a remote repository.
- Push to or pull from a remote repository.

Version control really comes into its own when we begin to collaborate with other people. We already have most of the machinery we need to do this; the only thing missing is to copy changes from one repository to another.

Systems like Git allow us to move work between any two repositories. In practice, though, it's easiest to use one copy as a central hub, and to keep it on the web rather than on someone's laptop. Most programmers use hosting services like GitHub, BitBucket or GitLab to hold those master copies; we'll explore the pros and cons of this in the final section of this lesson.

Let's start by sharing the changes we've made to our current project with the world. Log in to GitHub, then click on the icon in the top right corner to create a new repository called `planets`:

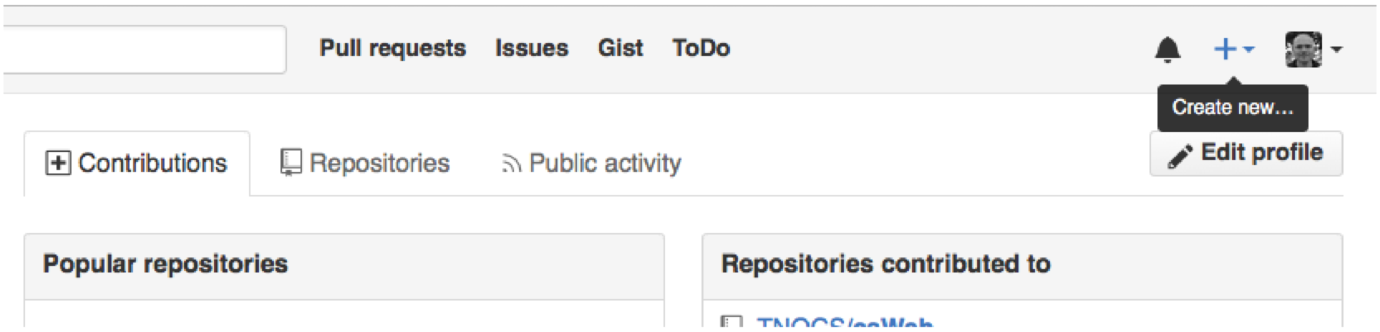
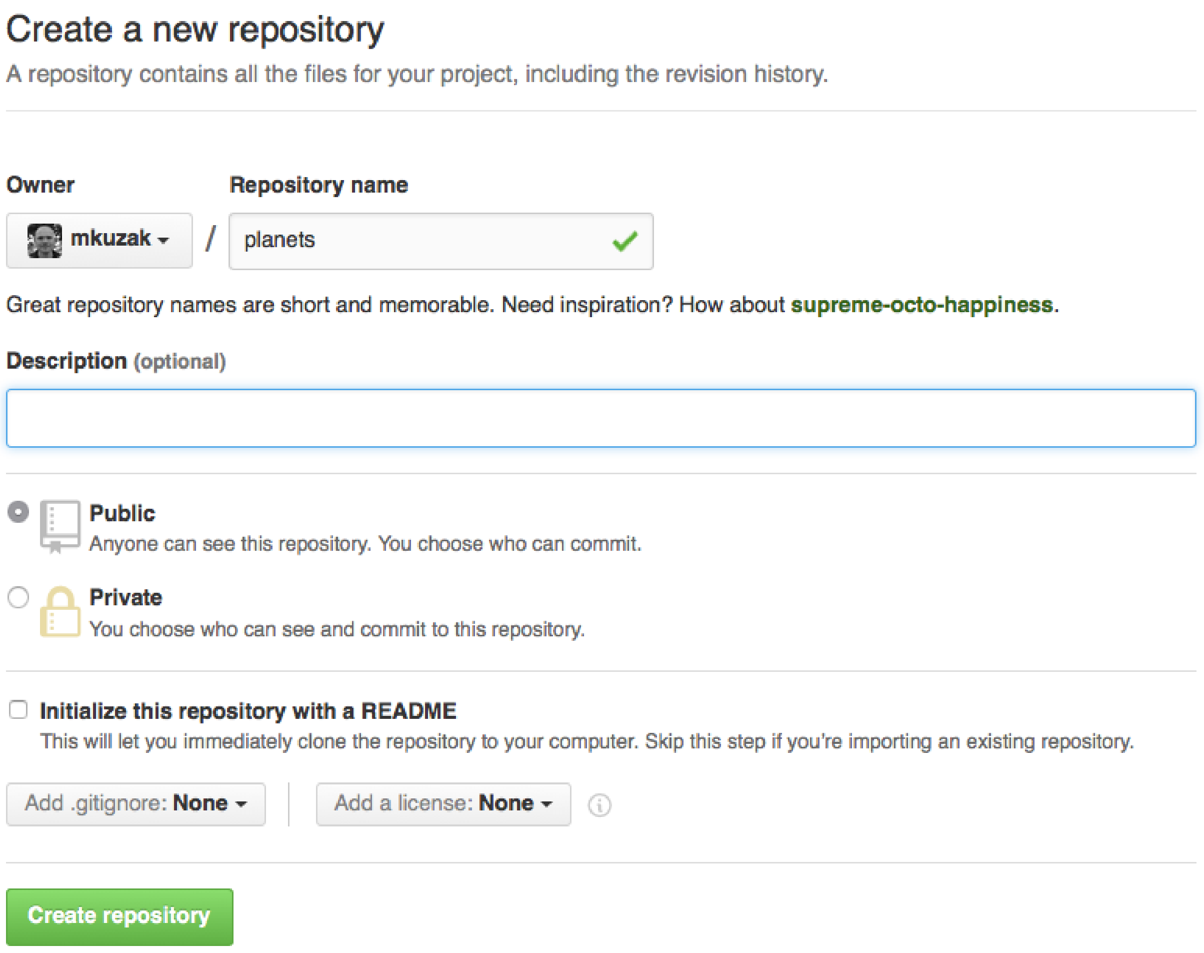


Figure: Creating a Repository on GitHub (Step 1)

Name your repository "planets" and then click "Create Repository":


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: mkuzak / Repository name: planets

Great repository names are short and memorable. Need inspiration? How about **supreme-octo-happiness**.

Description (optional)

[✓ Public] Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

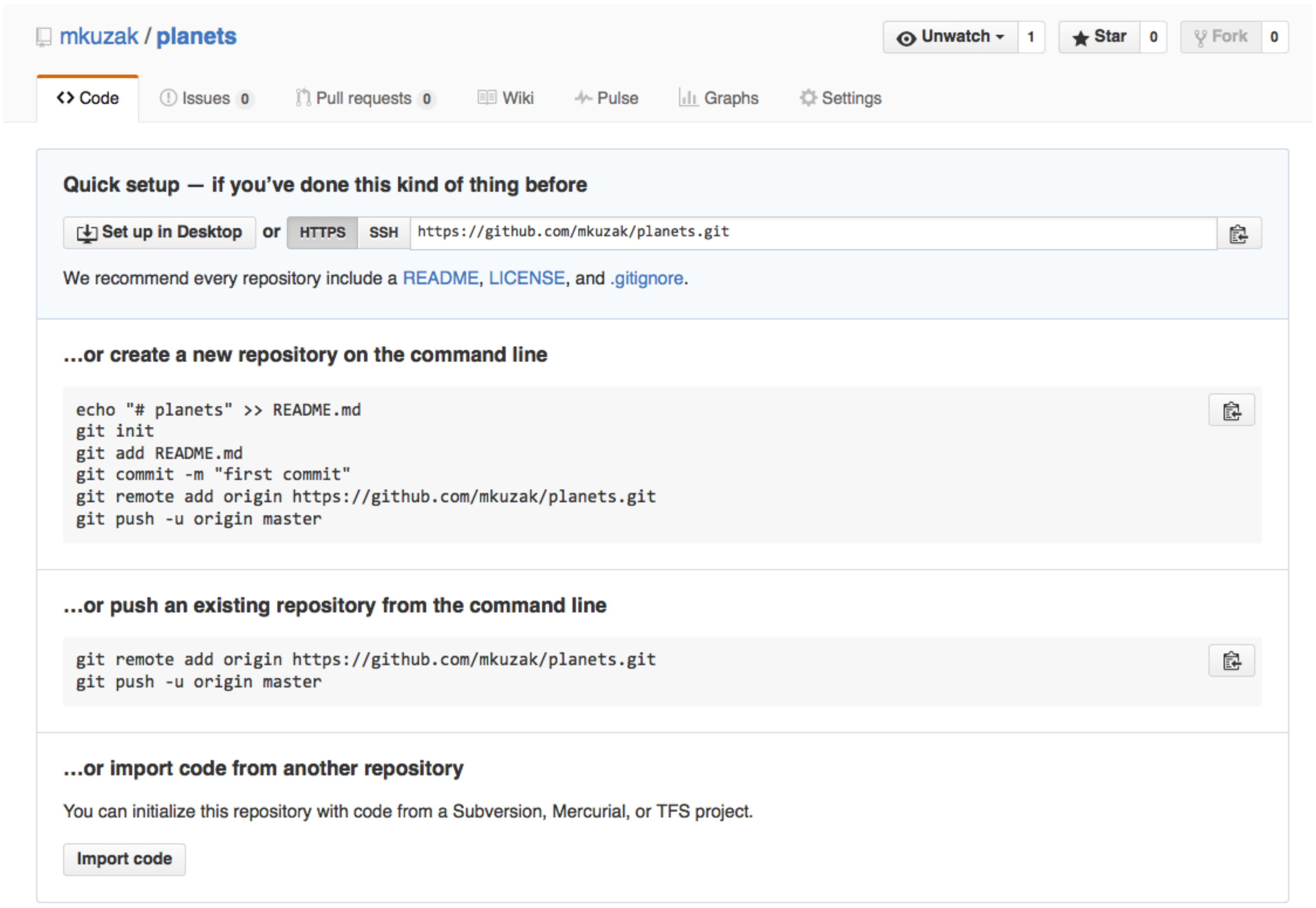
☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Add a license: None

Figure: Creating a Repository on GitHub (Step 2)

As soon as the repository is created, GitHub displays a page with a URL and some information on how to configure your local repository:



Quick setup — if you've done this kind of thing before

```
https://github.com/mkuzak/planets.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

```
echo "# planets" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/mkuzak/planets.git
git push -u origin master
```

…or push an existing repository from the command line

```
git remote add origin https://github.com/mkuzak/planets.git
git push -u origin master
```

…or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Figure: Creating a Repository on GitHub (Step 3)

This effectively does the following on GitHub's servers:

```
$ mkdir planets
$ cd planets
$ git init
```

Our local repository still contains our earlier work on `mars.txt`, but the remote repository on GitHub doesn't contain any files yet:

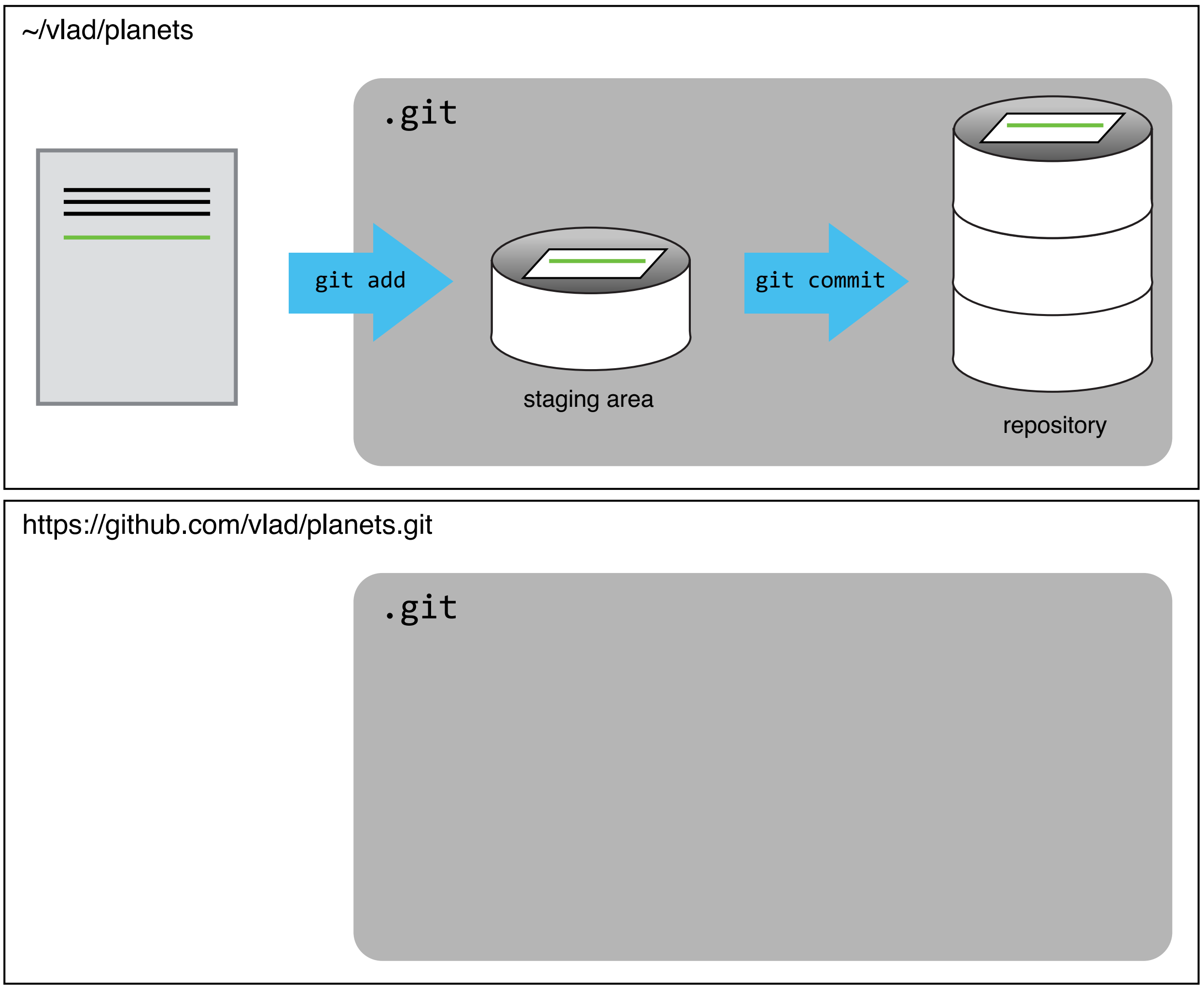
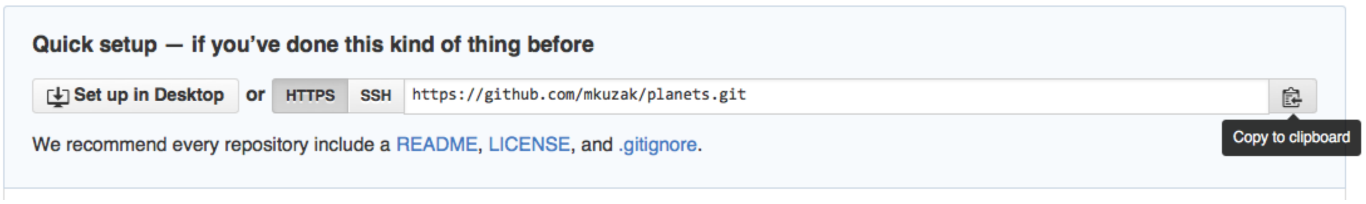


Figure: Freshly-Made GitHub Repository

The next step is to connect the two repositories. We do this by making the GitHub repository a remote for the local repository. The home page of the repository on GitHub includes the string we need to identify it:


Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/mkuzak/planets.git` Copy to clipboard

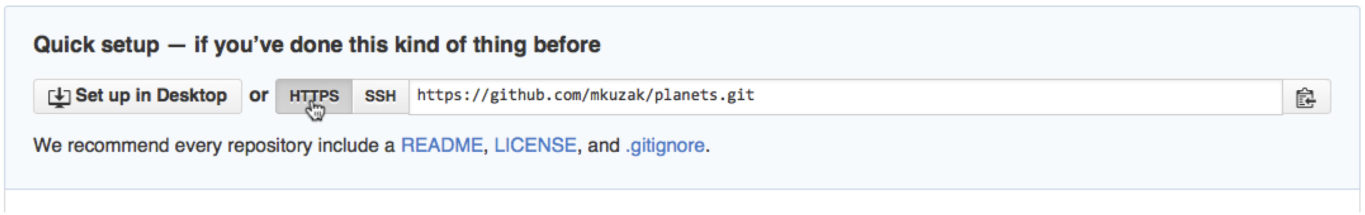
We recommend every repository include a README, LICENSE, and .gitignore.

Figure: Where to Find Repository URL on GitHub

Click on the 'HTTPS' link to change the protocol from SSH to HTTPS.

HTTPS vs SSH

We use HTTPS here because it does not require additional configuration. After the workshop you may want to set up SSH access, which is a bit more secure, by following one of the great tutorials from GitHub, Atlassian/BitBucket and GitLab (this one has a screencast).


Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/mkuzak/planets.git`

We recommend every repository include a README, LICENSE, and .gitignore.

Figure: Changing the Repository URL on GitHub

Copy that URL from the browser, go into the local `planets` repository, and run this command:

```
$ git remote add origin https://github.com/vlad/planets
```

Make sure to use the URL for your repository rather than Vlad's: the only difference should be your username instead of `vlad`.

We can check that the command has worked by running `git remote -v`:

```
$ git remote -v
```

```
origin   https://github.com/vlad/planets.git (push)
origin   https://github.com/vlad/planets.git (fetch)
```

The name `origin` is a local nickname for your remote repository: we could use something else if we wanted to, but `origin` is by far the most common choice.

Once the nickname `origin` is set up, this command will push the changes from our local repository to the repository on GitHub:

```
$ git push origin master
```

```
Counting objects: 9, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 821 bytes, done.
Total 9 (delta 2), reused 0 (delta 0)
To https://github.com/vlad/planets
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Proxy

If the network you are connected to uses a proxy there is an chance that your last command failed with "Could not resolve hostname" as the error message. To solve this issue you need to tell Git about the proxy:

```
$ git config --global http.proxy http://user:password@proxy.url
$ git config --global https.proxy http://user:password@proxy.url
```

When you connect to another network that doesn't use a proxy you will need to tell Git to disable the proxy using:

```
$ git config --global --unset http.proxy
$ git config --global --unset https.proxy
```

Password Managers

If your operating system has a password manager configured, `git push` will try to use it when it needs your username and password. If you want to type your username and password at the terminal instead of using a password manager, type:

```
$ unset SSH_ASKPASS
```

You may want to add this command at the end of your `~/.bashrc` to make it the default behavior.

Our local and remote repositories are now in this state:

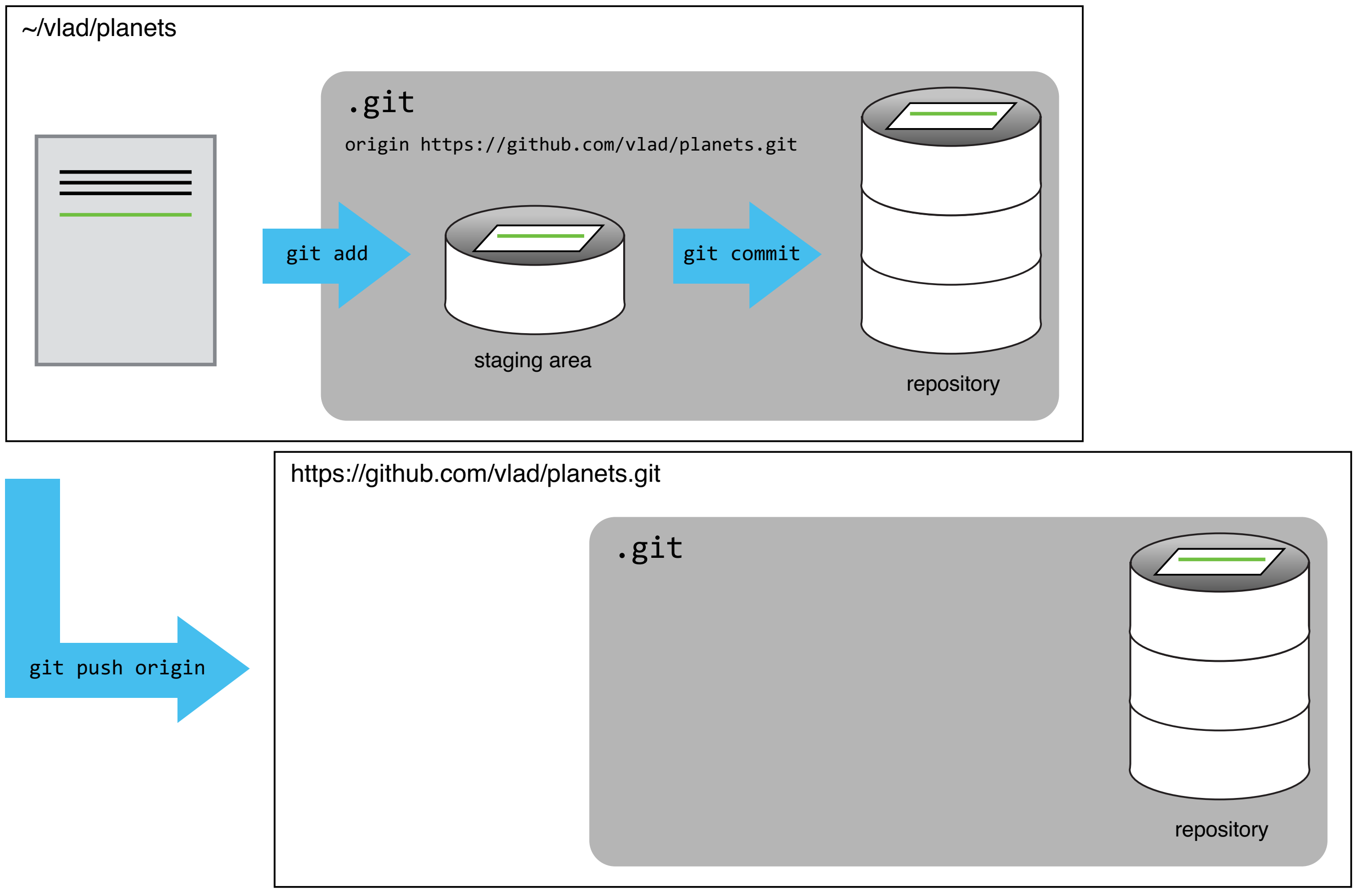


Figure: GitHub Repository After First Push

The '-u' Flag

You may see a `-u` option used with `git push` in some documentation. It is related to concepts we cover in our intermediate lesson, and can safely be ignored for now.

We can pull changes from the remote repository to the local one as well:

```
$ git pull origin master
```

```
From https://github.com/vlad/planets
 * branch            master     -> FETCH_HEAD
Already up-to-date.
```

Pulling has no effect in this case because the two repositories are already synchronized. If someone else had pushed some changes to the repository on GitHub, though, this command would download them to our local repository.

GitHub Timestamp

Create a repository on GitHub, clone it, add a file, push those changes to GitHub, and then look at the timestamp of the change on GitHub. How does GitHub record times, and why?