# Exercise 1: Getting Started with GitHub

## Aim

To get you set up with a GitHub account and **personal access token**. Then to create an empty repository to put your work from the week and get used to using the basic workflow of Git.

### Issues Covered

- Access tokens
- Github
- Git Repos
- Clone/Commit/Push

## 1. Sign up for GitHub

If you don't already have an account go to https://github.com and sign up (top right)

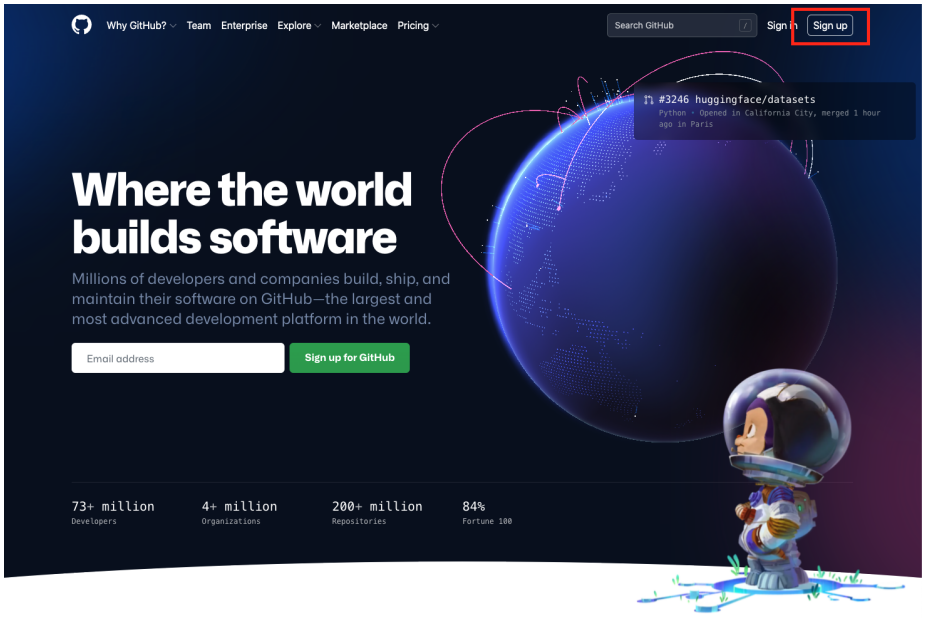


Figure 1: github_homepage

## 2. Generate an access token to use as your password and setup the Git client

GitHub has moved away from username and password as a way of authenticating repo access. Username/Password combos are known to be insecure (how many do you really have, you should probably have 100s) and we will likely see the way we authenticate change across the internet over the next few years.

To generate a token:

1. Login to **GitHub**
2. Click on your name/avatar in the top right corner and select **Settings**
3. On the left, click **Developer Settings**
4. Select **Personal access tokens** and click **Generate new token**
5. Give the token a description/name and select **repo** scope (you can read more about scopes using the **Read more about OAuth scopes**)
6. Click **Generate Token**
7. Copy the token - this is your new password (you might want to put this in a password manager)

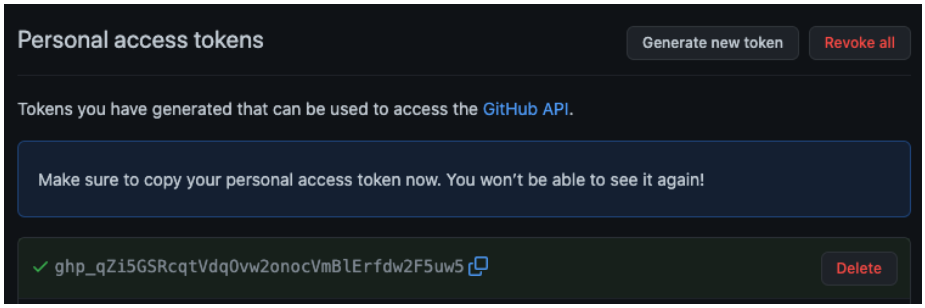


Figure 2: access_token

Configure local git:

This tells the Git client running on your machine who you are.

On linux, the commands look like:

```
git config --global user.name "<your-username>"
git config --global user.email "<your-email>"
git config -l
```

The `git config -l` will show you all the values stored in the config

When you clone a repo, it will ask for your username/password. Enter your username and access token as the password.

**Credential Caching**

It is annoying to have to react to the prompt every time. Thankfully, you can cache your credentials.

`git config --global credential.helper cache`

This saves your username/access token in memory so you don't need it everytime.

If you need to clear it:

`git config --global --unset credential.helper`

There are other stores (you can look these up another time): - `credential.helper store` - stores credentials on disk - `credential.helper osxkeychain` - can be used with MacOS to store credentials in MacOSs keychain

## 3. Make a repo for your Introduction to Scientific Computing work

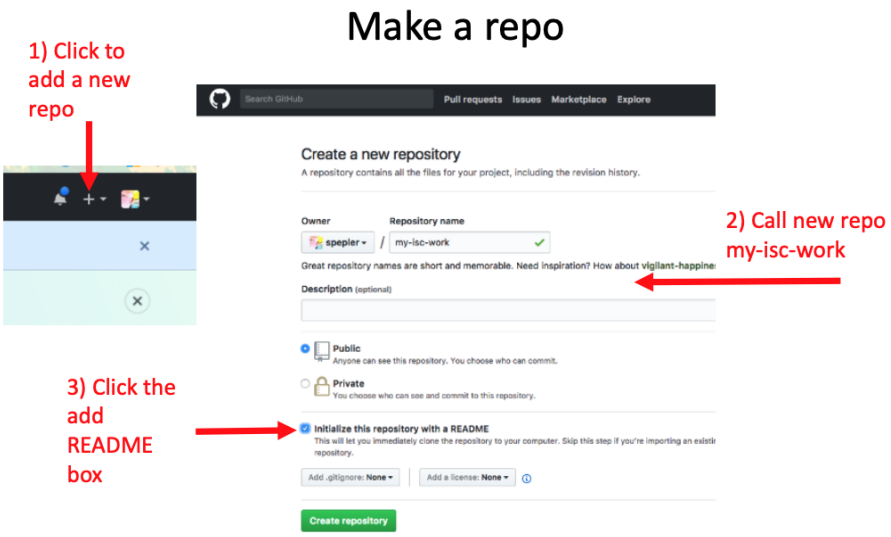


Figure 3: new repo

## 4. Copy the clone link

## 5. Clone the repo

1. Open the terminal
2. make sure you are in your home directory `cd`
3. `git clone <you-clone-link>`

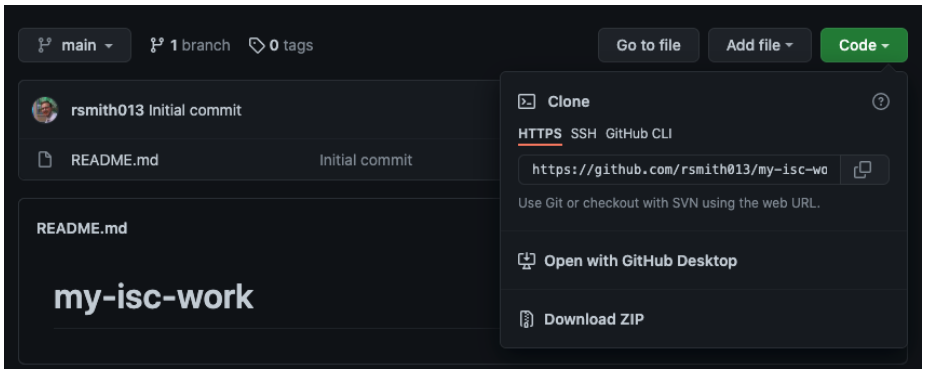


Figure 4: new repo

It will then prompt for username and password (it won't ask again if you have your credentials cached) **NOTE: Password == Access token**

You have now cloned the repo. Now we are going to make a change and push it.

## 6. Make a change

```
bash
cd my-isc-work
touch x
```

If you now run `ls` you will see two files. `README.md` and `x`.

See what has changed.

```
git status
```

```
(base) vpn-149-065:my-isc-work vdn73631$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        x

nothing added to commit but untracked files present (use "git add" to track)
```

Figure 5: git status

Add the changed file to git

```
git add x
```

(can use `git add .` to add all files)

Write a message to say what you have done and make a **commit**.

```
git commit -m "Adding an empty file as part of tutorial"
```

Push the change to github

```
git push origin main
```

**7. Refresh the page on github. You will now see empty file x.**