README

1. Git

1.1) Install git

git, which is not the same as github, is a software to manage versions of your stuff. You can save changes, go back to previous versions, branch your project in different directions, etc...

1. go to the git website download page
2. download for your OS (windows download, not always up to date)
3. run the installer with defaults

1.2) Initialize your repository

a repository is essentially a project you want git to manage. Open the folder you want to turn into a git repository in a command prompt and run the following command:

```
git init
```

this has to be done only once when you initialize a repository.

1.3) Commit new changes

after you make some changes to your project, you can commit them to a new "version" (called a commit), this is essentially like making a checkpoint.

run the following commands:

```
git add *
```

this adds the changes to the commit, the "*" specifies you want to add all the changes made to the commit, you could specify only specific files, but this is beyond the basics for now. (you can run `git add` multiple times, the commit will not be commited until the next command is ran)

```
git commit -am "commit message goes here"
```

this commands commit the selected changes (using `git add`) to the next commit, you must specify a commit message inside the "". (there are other ways to write bigger and better commit messages, but are beyond the basics)

now you can make new changes, add and commit, then repeat.

1.4) Gitignore

often you might have files you dont want to be part of your git repository, some examples are:

- sensitive configuration files (passwords and tokens)
- very large files (videos, images, etc...)
- build files (built executables)
- every other file that is only needed locally

you could exclude this files every time you do `git add` but a better way is to create a `.gitignore` (`.gitignore` is the actual name of the file, not the extension) file inside your repository. Inside a .gitignore file you can select what to ignore like this:

```
*.bin  # excludes all .bin files from the repository
build/  # excludes the whole /build folder from the repository
...
```

2) Github

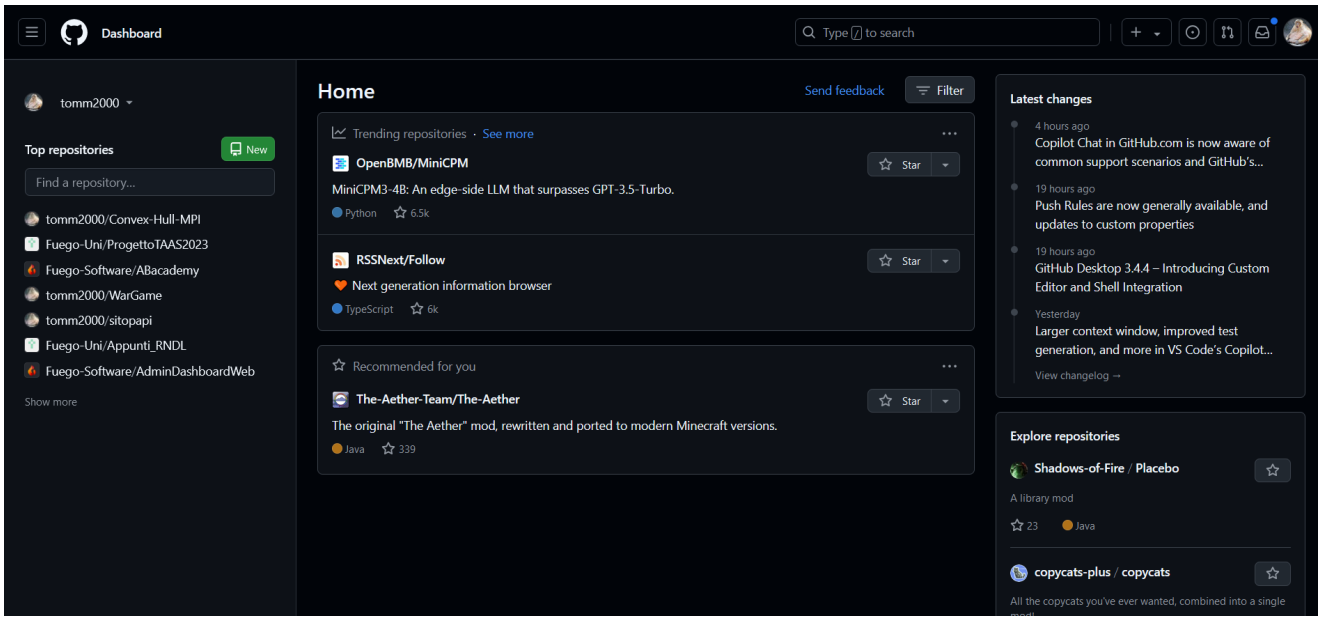
github is a website that hosts your repositories online. You can `push` updates "to the cloud" and `pull` remote changes on your local machine.

2.1) Github account

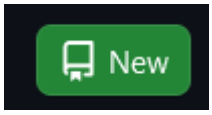
firstly create an account at the github website

2.2) Creating remote repository

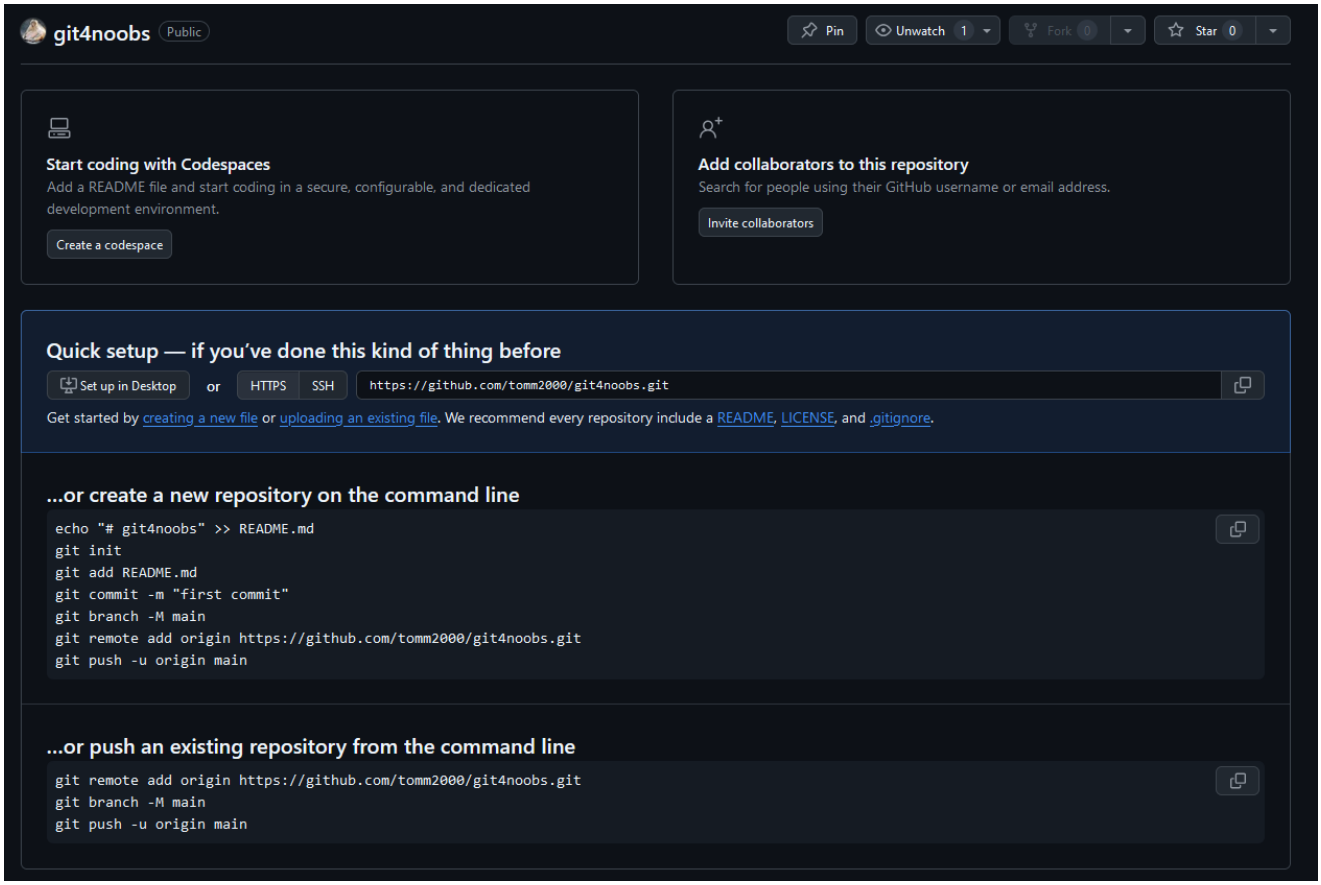
when navigating to https://github.com/ the page should look something like this:



click on the green "new" button



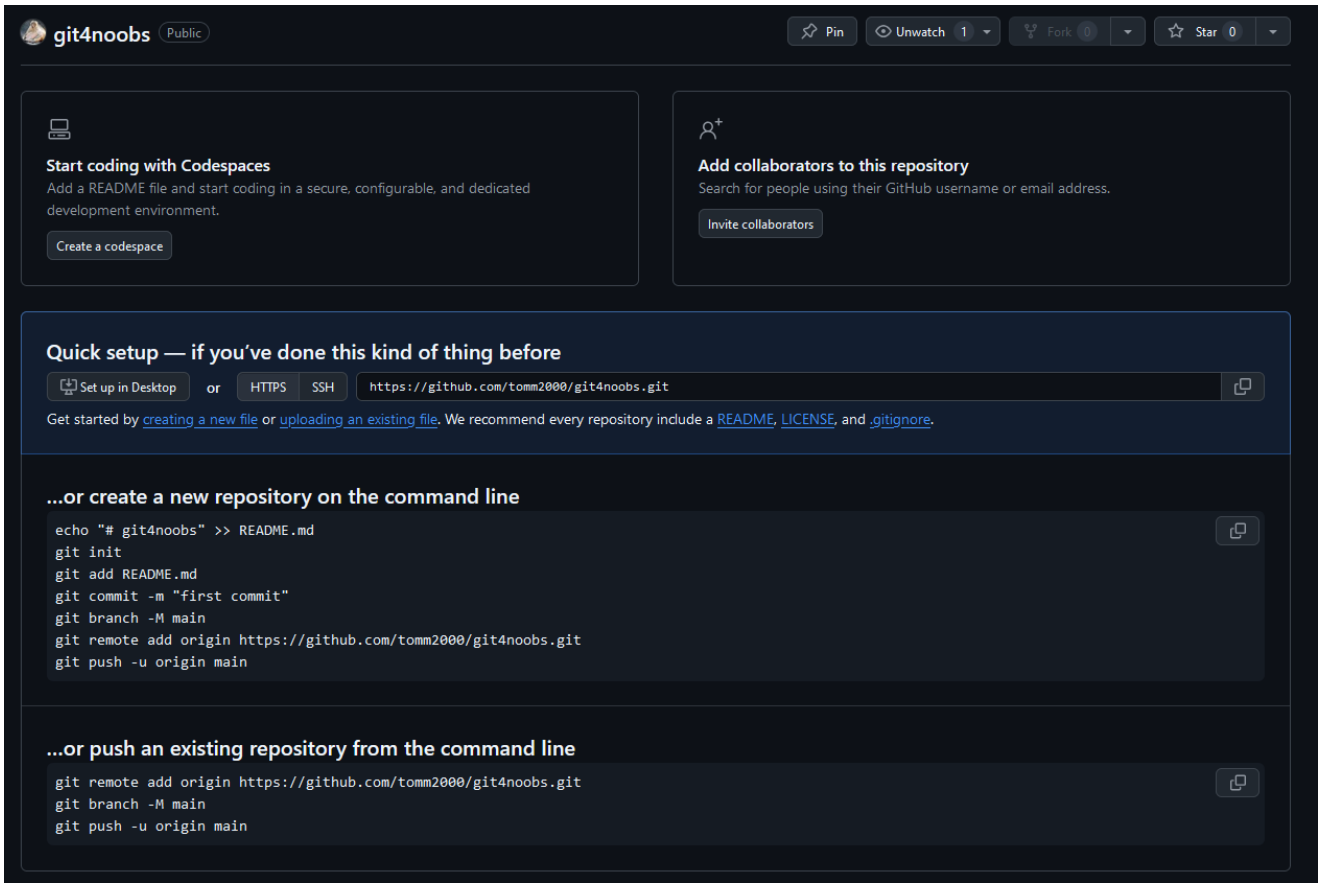
you should get a page like this:



Public and private options are explained in the tooltip, keeping it public is fine, and can be changed at any time.

the rest of the options are not relevant right now and can be left as defaults.

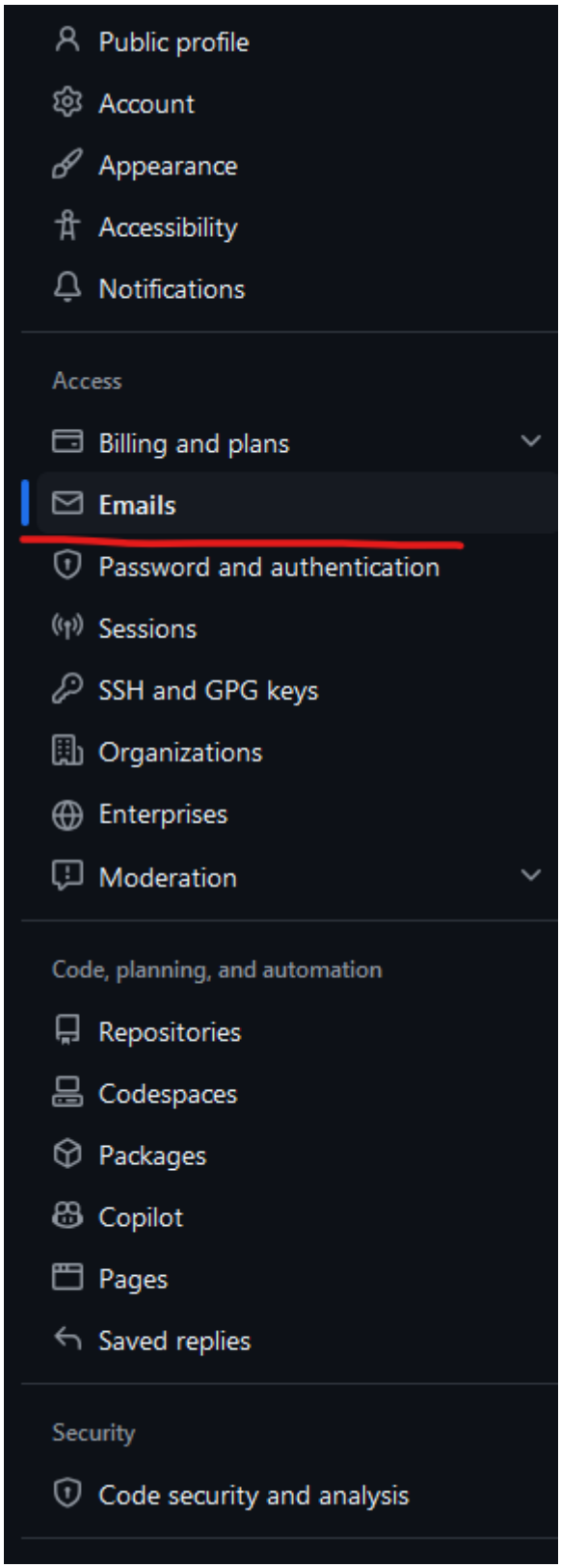
Once you click "create repository" you should land on a page like this:



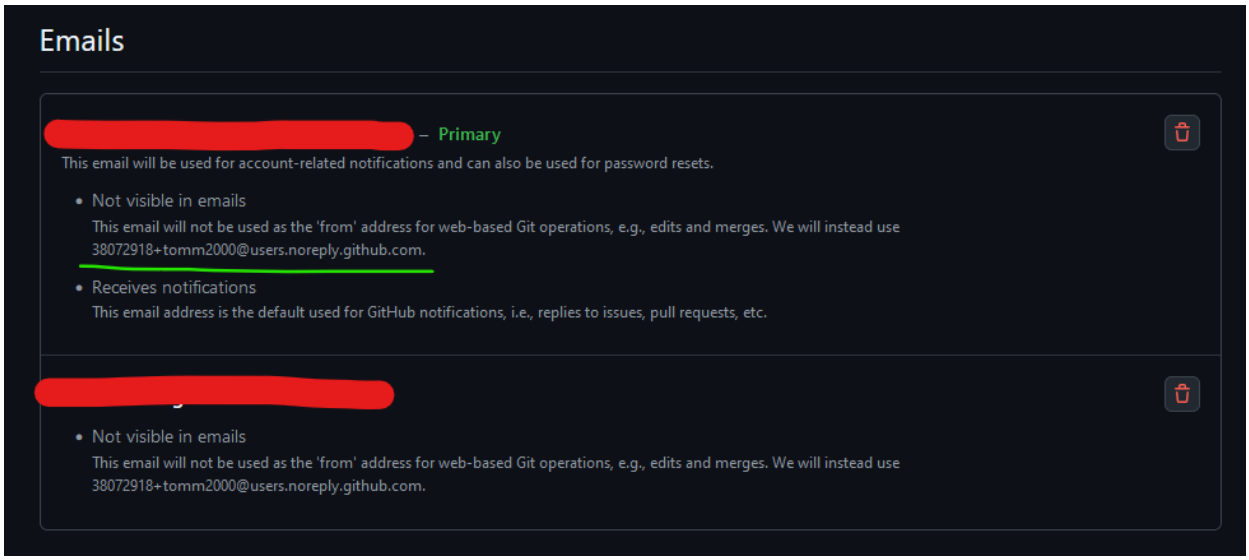
2.3) Email settings

the first time you are going to try pushing a repository, you might run into the issue of git asking you for a username and email. You can choose any username but by default you are not allowed to use your normal email (for privacy reasons). To find the correct github email follow these steps:

1. go to https://github.com/settings/profile
2. navigate to the emails tab



3. take note of the github email (marked in green in the image below):



this is the email that you will use to make pushes.

you can now run this 2 commands (only once per machine usually, when asked):

```
git config --global user.name "Your Name"
```

```
git config --global user.email "youremail@yourdomain.com"
```

again, you can choose any username and should replace the email with the github email (not your personal email). These 2 fields can be changed ad any time with the same commands.

2.3) Linking local repo to remote repo

now you want to "link" your local repository to be saved on this remote repository you just created.

the easiest way is to use this command once in the local repository:

```
git push --set-upstream https://github.com/tomm2000/git4noobs master
```

obviously you should replace `https://github.com/tomm2000/git4noobs` with the url of your repository (should be copied directly from the browser address bar)

if you refresh the remote repository you should now see the files of your local repository.

2.4) Pushing

if you want to push local changes to the remote repository you first need to perform a commit on the local repository.

NOTE: when you push changes you only push the commit, uncommited changes won't be pushed to the remote.

to push simply run this command (in the repository folder):

```
git push
```

NOTE: when pushing, ALL the commits done up to the last push will be pushed. You dont need to push after every single commit.

2.5) Pulling

if for whatever reason the remote repository is ahead of your local repository (ex: you made changes from another machine), you might want to pull the remote changes to your local repository.

to pull remote changes run this command (in the repository folder):

```
git pull
```

2.6) Conflicts

As long as you keep your project synced you should have no issues pulling and pushing, but if you mismatch the versions (ex: you push, make local changes, then try to pull the "old" version) you will run into conflicts, conflicts can be resolved but it's not an easy task often. For basic use it's better to avoid them.

2.7) Cloning

If you have a remote repository you want to "download" you need to clone it.

to clone (any) repository you can run the following command:

```
git clone https://github.com/tomm2000/git4noobs [destination folder]
```

the destination folder is optional, by default a folder with the same name of the remote repository will be created at the location where the command is ran.

3) TLDR

to start:

1. init local repository

```
git init
```

2. create remote repository
3. push the repository first time

```
git push --set-upstream https://github.com/tomm2000/git4noobs master
```

---

workflow

1. make changes to project
2. add changes to commit:

```
git add *
```

3. commit changes:

```
git commit -am "message goes here"
```

4. push changes when feel necessary:

```
git push
```