GitHub Documentation

Setting up & version control and sharing

Author: Behdad Shaarbaf Ebrahimi

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

AUCKLAND BIOENGINEERING INSTITUTE

Lung and respiratory group

Setting up

Note: This documentation is mostly compatible with Linux OS.

There are two major ways setting up a repository can happen:

i. Setting it up from scratch using terminal
ii. Cloning a repository
iii. A brief look at documentation by Sphinx

1) Setting up a repository from scratch

The first step is to make sure you have git installed on your system.

In order to check you can open a terminal and type:

git --version

This command will show you the version of the git installed. As a result, if you do not have git on your system it will complain about the input "git" command.

Installing Git on linux Debian/Ubuntu :

- Open terminal
- *Sudo apt-get install git* ------ Enter university credentials (password)

If on any other OS you can use this link:

https://www.linode.com/docs/development/version-control/how-to-install-git-on-linux-mac-and-windows/

1) Setting up a repository from scratch

After finishing with Git installation, defining a directory as GitHub repository use the steps below:

- *cd to directory you want to make it a repository (cd /hpc/<upi>/My_repo)*
- *git init .*

You will see the message:

Initialized empty Git repository in ~dir/.git/

NOTE: In order to check and see if you git initialization files are created, you can do:

- Move to you people drive (~/people/<upi ID>)
- ls -A
- You may find all the folders including git folders created

1) Setting up a repository from scratch

A professional repository should consist of the following parts:

- Source directory (For you scripts and functions and classes)
- Test folder (For test driven scripting)
- Documentation folder (For documenting your scripts functions and classes)
- Examples folder (This is my personal preference and I have seen the group's favor towards having working examples ☺)
- A "README" file as a brief explanation on your GitHuB page.
- Optional but useful: a .gitignore file

1) Setting up a repository from scratch

The last step is to put the repository made on the cloud. Steps are:

- *git remote add origin git@github.com:<your_username>/<repository_name>.git*

Note: The repository needs to be created on you profile first. Then you can add the remote address!

- *git pull origin <branch_name>* : the first branch default name is "master"
- *git remote -v* : after doing this command you should be able to see the address to your repository

You make changes to README.rst file and setup the description to the repository. Write some scripts and functions the same as the structure discussed in the previous slide.

After making all the changes:

- *cd to the directory of your repository*
- *git add .*
- *git commit -m "A brief explanation of the changes"*
- *git push origin master*

Now your repository is set up.

For better understanding of reStructedtext (*.rst) format you can refer to the links below:

- https://github.com/behdadebsh/LungPhD (fork, clone and see the source files as an example)
- http://docutils.sourceforge.net/rst.html#user-documentation

2) Something recommended – Git Prompt

This is a feature that makes you terminal look different when in GitHub repositories. You may find different types and themes for this feature on the internet but, my personal preferred one and easy to setup is provided here:

The easiest way is using *git clone*.

- *cd ~* (To make sure you are on your home directory)
- *git clone https://github.com/magicmonty/bash-git-prompt.git .bash-git-prompt --depth=1*
- *gedit .bashrc*
- *Copy this text into your bashrc file:*

```
GIT_PROMPT_ONLY_IN_REPO=1
source ~/.bash-git-prompt/gitprompt.sh
```

Note: There are other setup methods for git prompt available on:
https://github.com/magicmonty/bash-git-prompt

2) Something recommended – Git Prompt

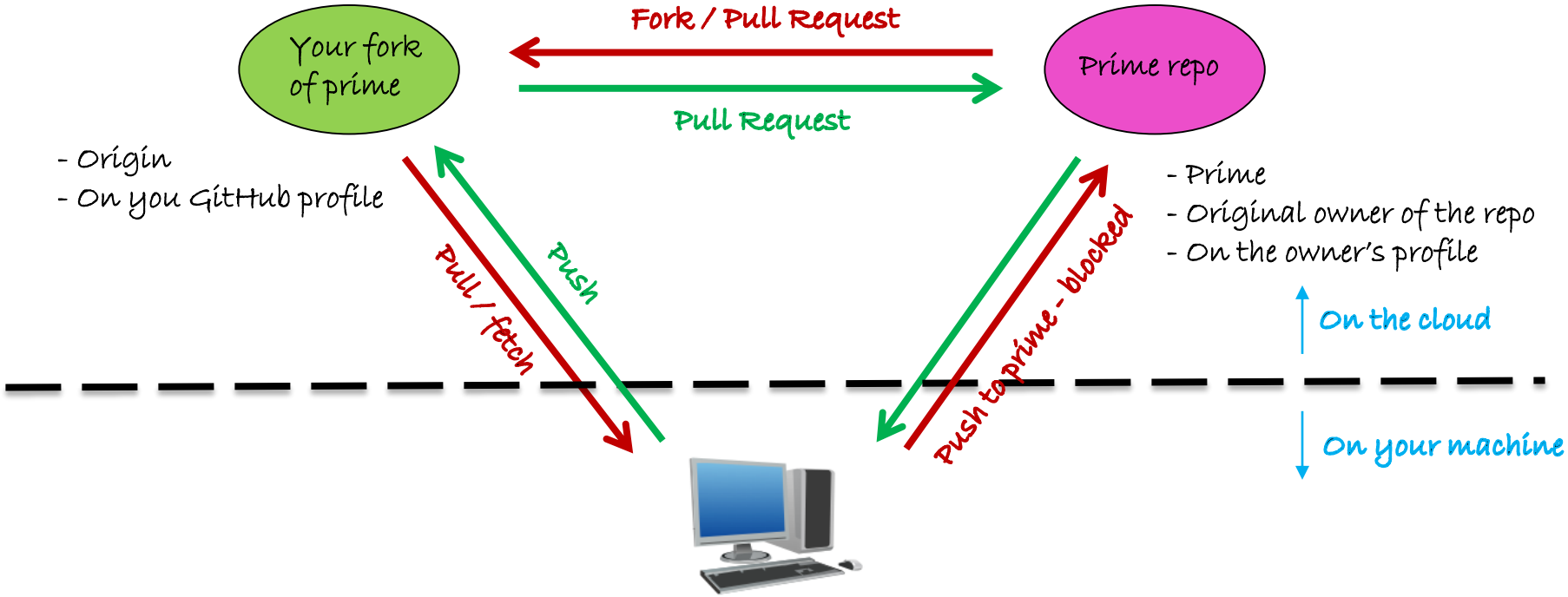
```
✔ /hpc/bsha219/Python/Behdad/lung_segmentation [master L|…33]
15:11 $
✔ /hpc/bsha219/Python/Behdad/lung_segmentation [master L|…33]
15:11 $
✔ /hpc/bsha219/Python/Behdad/lung_segmentation [master L|…33]
15:11 $ cd ..
bsha219@bn371671:/hpc/bsha219/Python/Behdad$ cd lung_segmentation/
✔ /hpc/bsha219/Python/Behdad/lung_segmentation [master L|…33]
15:12 $ git status
On branch master

Initial commit

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

        README.rst
        docs/

nothing added to commit but untracked files present (use "git add" to track)
✔ /hpc/bsha219/Python/Behdad/lung_segmentation [master L|…33]
15:15 $ ls docs/
_build  conf.py  index.rst  make.bat  Makefile  _static  _templates
✔ /hpc/bsha219/Python/Behdad/lung_segmentation [master L|…33]
15:16 $ vi .gitignore
✔ /hpc/bsha219/Python/Behdad/lung_segmentation [master L|…6]
```

3) Clone a repository

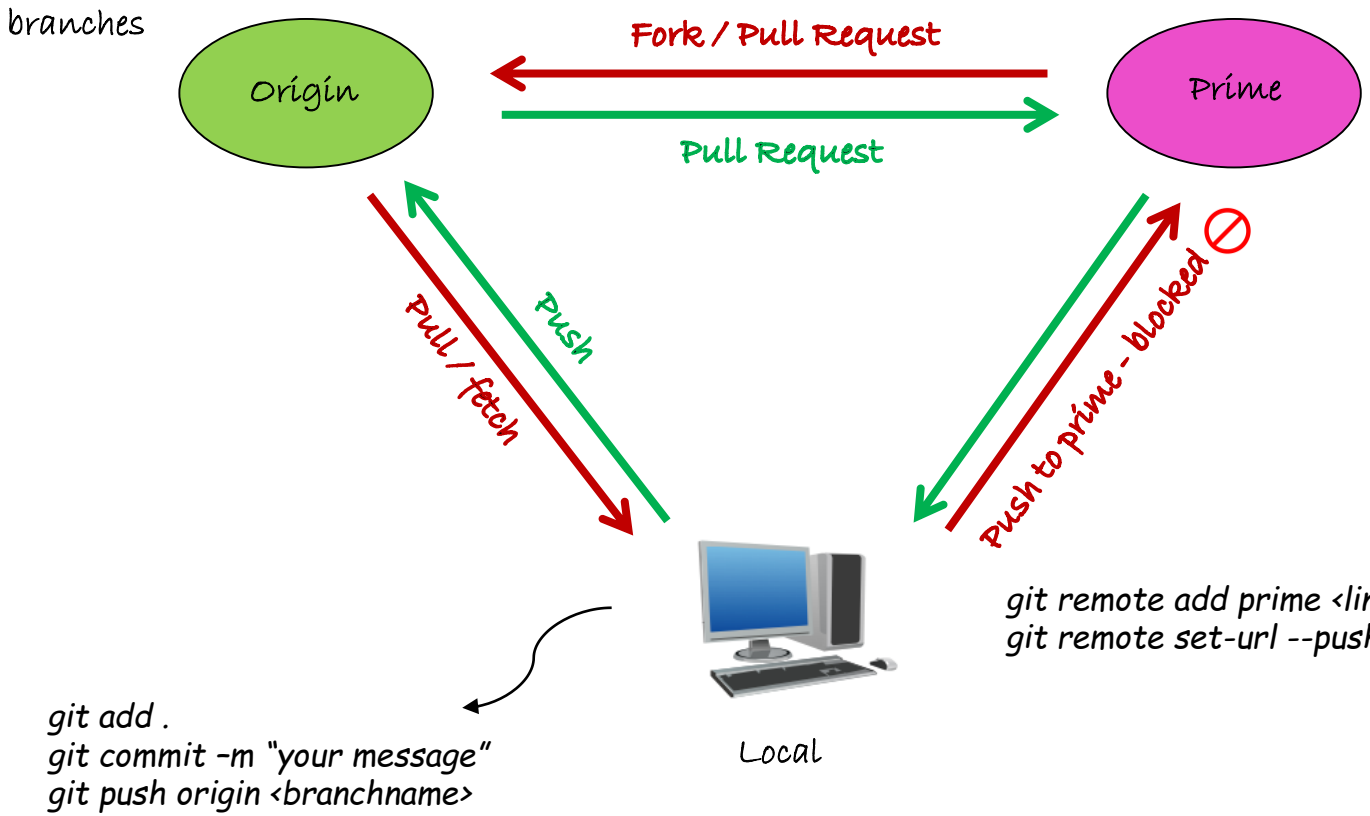
The easiest way to acquire a repository is clone it on your local machine. Some terms that are needed to be defined for consistency between the group:

- Origin
- Prime
- Fork
- Pull
- Push
- Local
- Branch
- Add
- Commit

3) Clone a repository



3) Clone a repository



4) Useful commands

Now that we have familiarized ourselves with useful phrases some git commands are helpful to know:

git remote -v : *shows the list of remotes to the repository i.e. origin, prime and any other remotes that defined. Remember origin is known word to GitHub but prime is defined by us.*

git branch <branch_name> : *creates a new branch with the name given*

git checkout <branch_name> : *switch between branches*

git status : *shows the status of the files added and the commits. This step is recommended before doing a push command to check for the last time what is going on the cloud. You do not want to add something that creates a bug in your script.*

git fetch : *practically only updates the history from the remote since your last pull, therefore, you can do a diff command and see the differences between your local working version and the version on remote (origin, prime, or…)*

git merge : *merges the changes on you local repository (after fetch) and the changes on your local working*

5) Documentation by Sphinx

How to install sphinx:

- Different methods for different OS available:
- pip install -U sphinx
- git clone .... (via source)
- apt-get install python(2/3)-sphinx -- For Debian/Ubuntu
- conda install
- MacOS with Homebrew (brew install ...) & MacPorts (sudo port install ...)

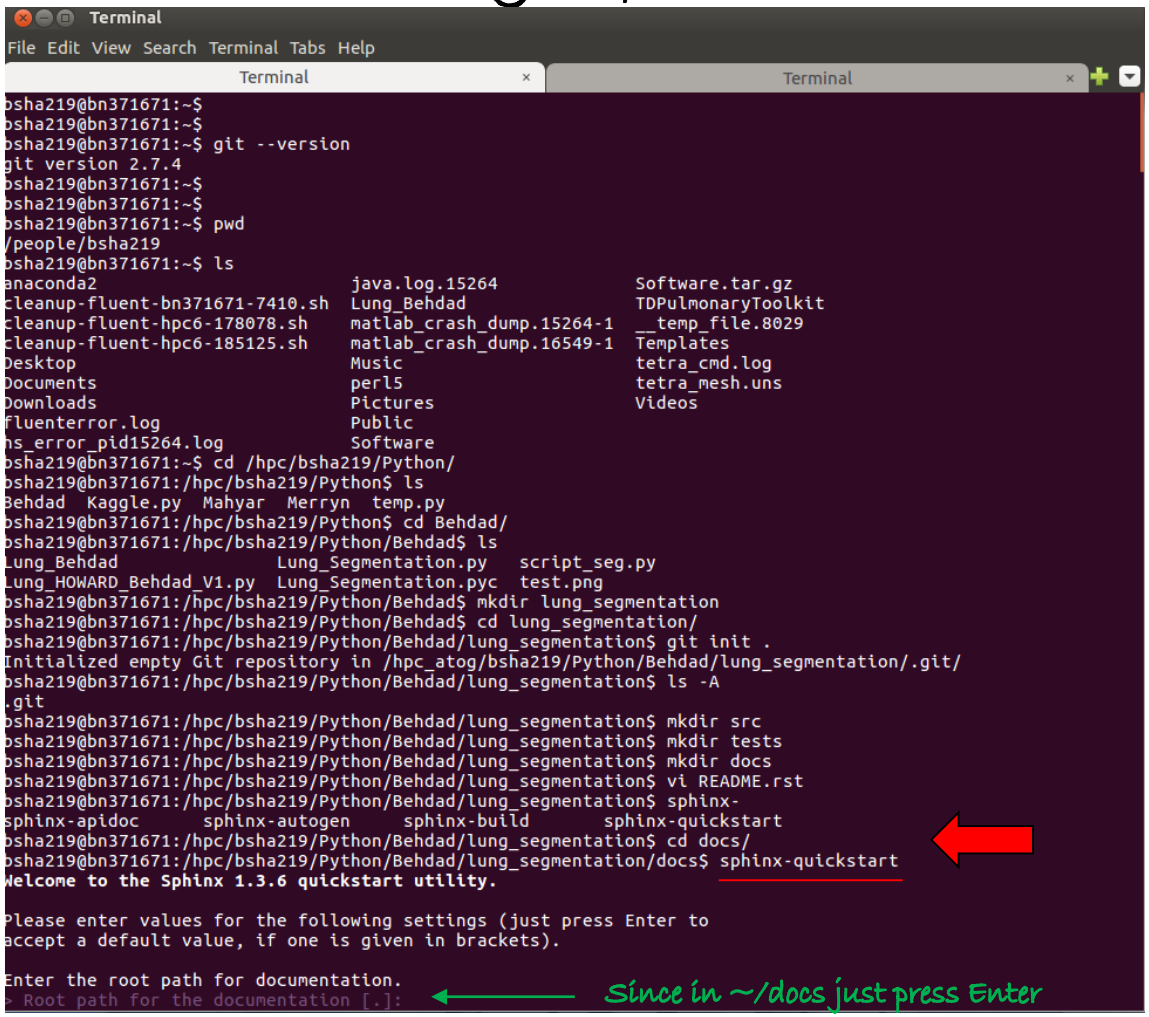
As discussed in GitHub setting up, a "docs" directory is designated for documentation!

For start after installing sphinx, change directory to your docs folder in your repository on your local machine. The first step would be:

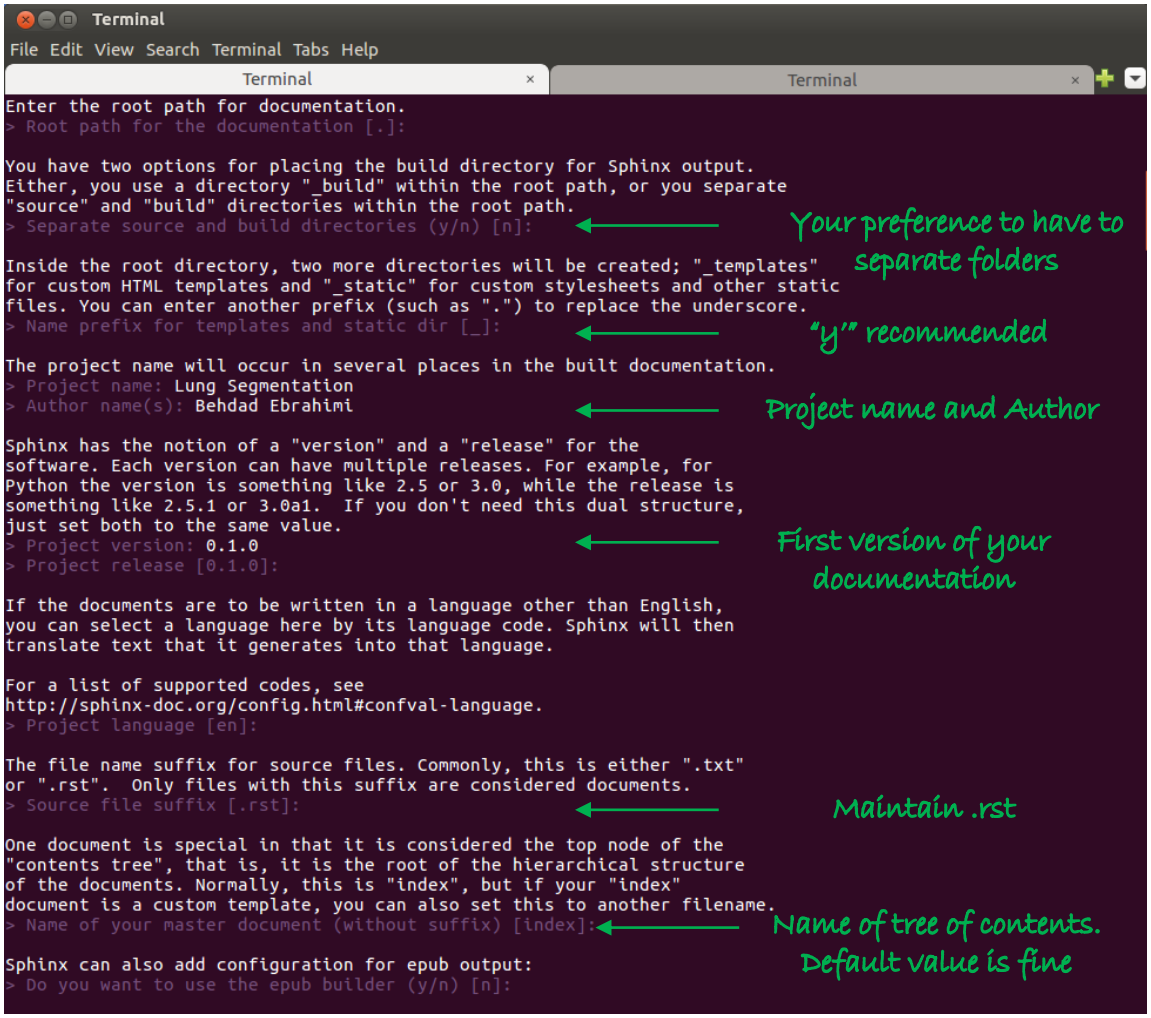
sphinx-quickstart

This command will immediately direct us to setting up a documentation. Basically the settings will be set through questions(This will edit the conf.py made after getting finished with this). In the next slides, a brief description is provided by screenshots of the terminal.

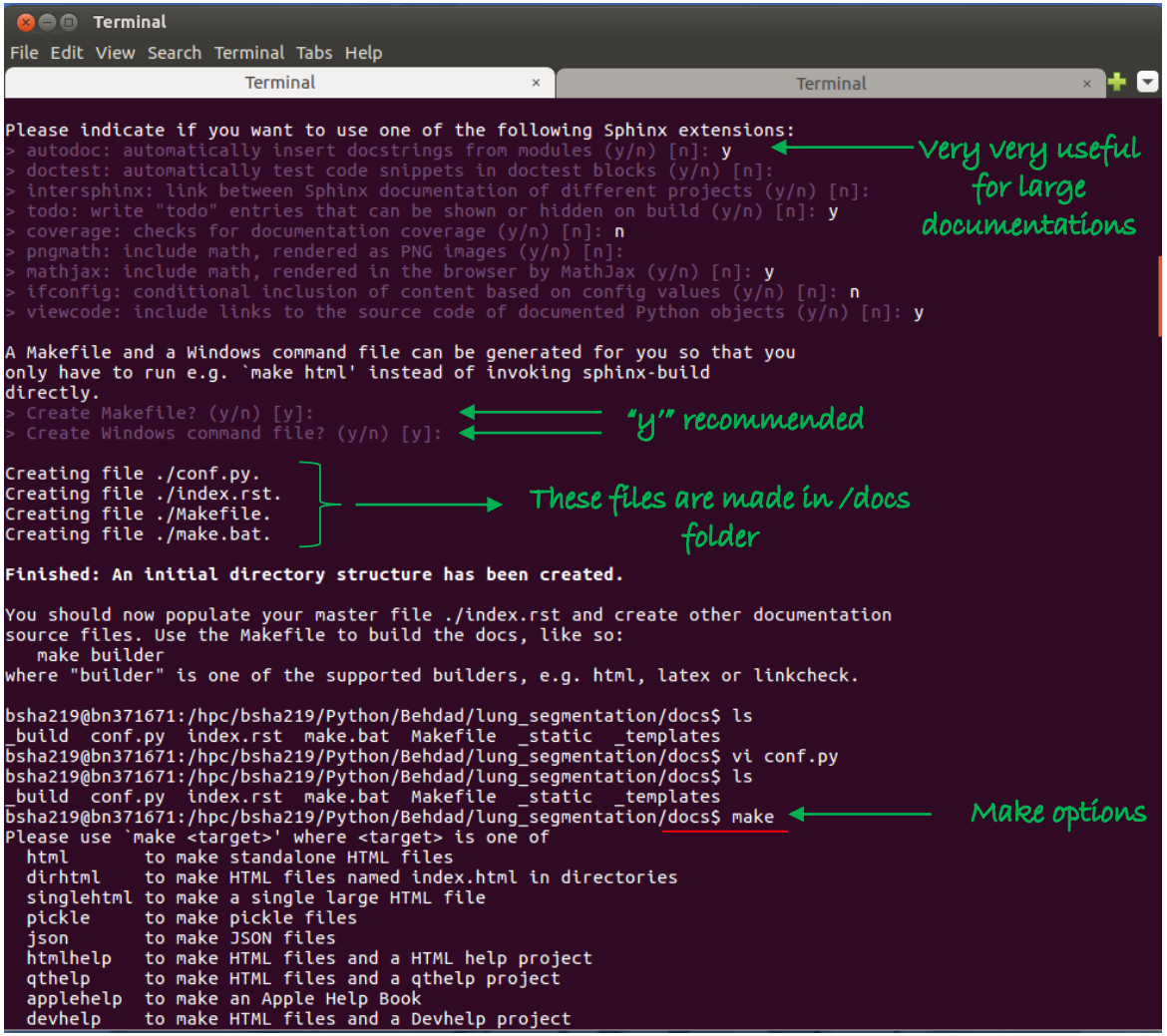
5) Documentation by Sphinx



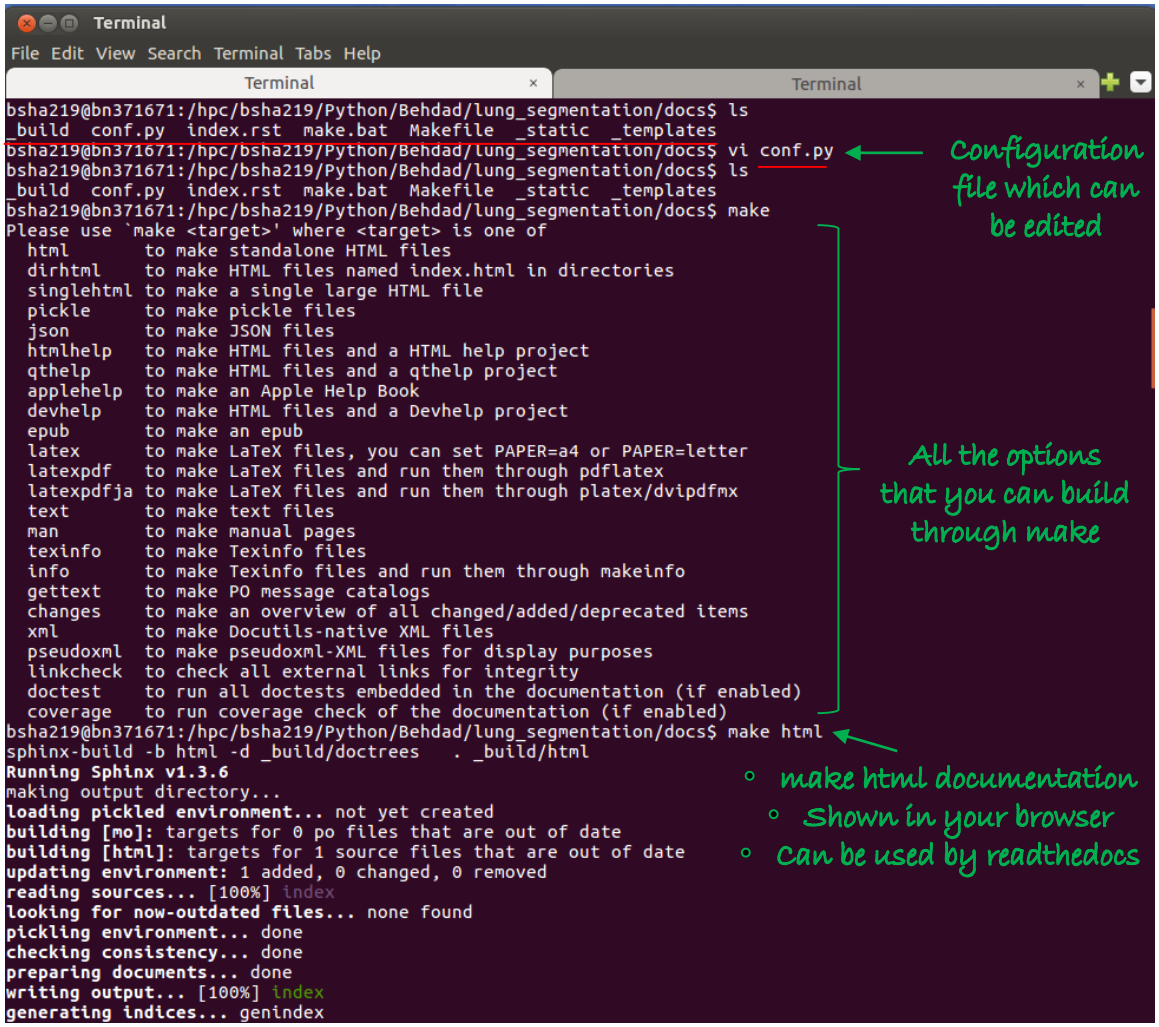
5) Documentation by Sphinx



5) Documentation by Sphinx



5) Documentation by Sphinx



6) Readthedocs

This is a platform made for automatic documentation based on sphinx files.

First you need to make and account in the website if you already do not have one.

Start a project and link it to you GitHub repository. Then you need to build your documentation which is exactly the same version as your html build by sphinx.

As a good example you can refer to documentations below:

- https://github.com/behdadebsh/LungPhD
- https://github.com/LungNoodle/lungsim
- https://github.com/duanemalcolm/morphic

The above are simple examples of documentations which you can make yourself familiar with the basic .rst format ☺