Lecture0 : AMS 562 Jupyter/Desktop, Terminal , Git

Lecture 0 Software Requirements

Date: August 27th 2021

Objectives:

- Establish a working development environment
- Docker - ams562_jupyter - ams562_desktop
- Introduction to Git and Github
- Setting up ams562_notes using

Lecture Outline

- [ ] Setting up Docker
- [ ] Intro to Docker Image
  - [ ] ams562_jupyter
    - [ ] Allows you to run code interactively (like python)
    - [ ] Good for testing single statements
    - [ ] All the features you normally get with jupyter notebook
  - [ ] ams562_desktop
    - [ ] linux environment (ubuntu 16.04)
    - [ ] Command Line
    - [ ] VSCode
- [ ] Introduction to Git and Github
  - [ ] Install git and create Github account
  - [ ] Create a local git repository
  - [ ] Add a file to the repo
  - [ ] Add a file to the staging environment
  - [ ] Create a commit
  - [ ] Create a new branch
  - [ ] Create a new repository on Github
  - [ ] Push a branch to Github

- [ ] Create a Pull Request
- [ ] Merge a PR

- [ ] Cloning ams562-notes and compiling
- [ ] Using Git inside VSCode

Getting Started

The basic format of the class will be a series of lectures followed by a set of examples that we will work on together during the course. For that reason, all of you must have a standard development environment that allows you to test code and follow along during the lectures quickly. We provide a standardized development environment by using Docker. So, as a first step, you all need to download Docker, which you can do by following this website. Once you have installed Docker and Python, download these two scripts that will ease using our container.

Desktop driver: ams562_desktop.py Jupyter driver: ams562_jupyter.py

---

AMS562 Jupyter Environment

We provide a Jupyter notebook environment as a method to test code quickly. Using the notebook allows you to follow along during the lectures. In addition, the notebook provides the ability to run C++ code without needing to compile programs.

To run the notebook, you simply run the python script provided. If you have Docker installed, the notebook should open in your web browser.

```
$ python ams562_jupyter
```

AMS562 Desktop Environment

We provide the desktop environment to provide all of you with a standard Linux desktop environment to practice and develop code for this course. In addition, the desktop comes with a Linux command line that serves as the interface to the computer, referred to as the terminal, shell, console. Finally, the environment provides Visual Studio Code (VSCode). VSCode offers valuable tools such as automatic formatting and syntax highlighting. VSCode also comes with easy git and Github integration. Which we will go over in more detail a bit later. These are the essential tools needed to write, build, and test code.

To run the desktop environment for the first time, open a terminal/console/PowerShell session and run

```
$ python ams562_desktop
```

wherever you have saved the scripts

The script will automatically pull the container and create the desktop environment in your default web browser. To see all options:

```
$ python ams562_desktop -h
```

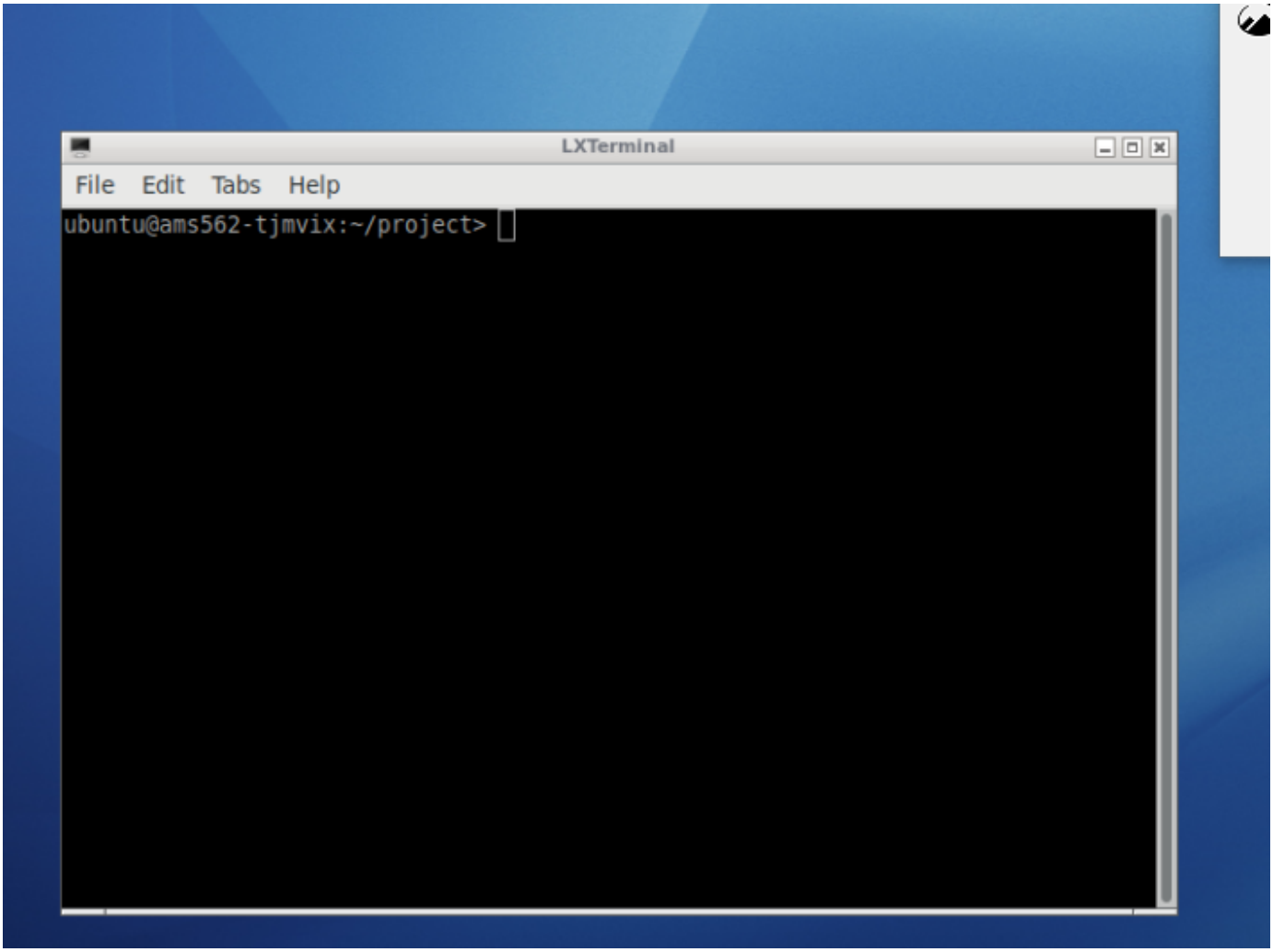
Docker directories	Host directories
`$DOCKER_HOME/shared`	Current working directory
`$DOCKER_HOME/project`	Data volume
`$DOCKER_HOME/.ssh`	`$HOME/.ssh`
`$DOCKER_HOME/.config`	`$HOME/.config`

Introduction to the command-line

Ubuntu Tutorial on terminal

For a more extensive introduction to the command line, you can follow this tutorial giving by Ubuntu. Here all present the bare minimum you need to get through the first view steps.

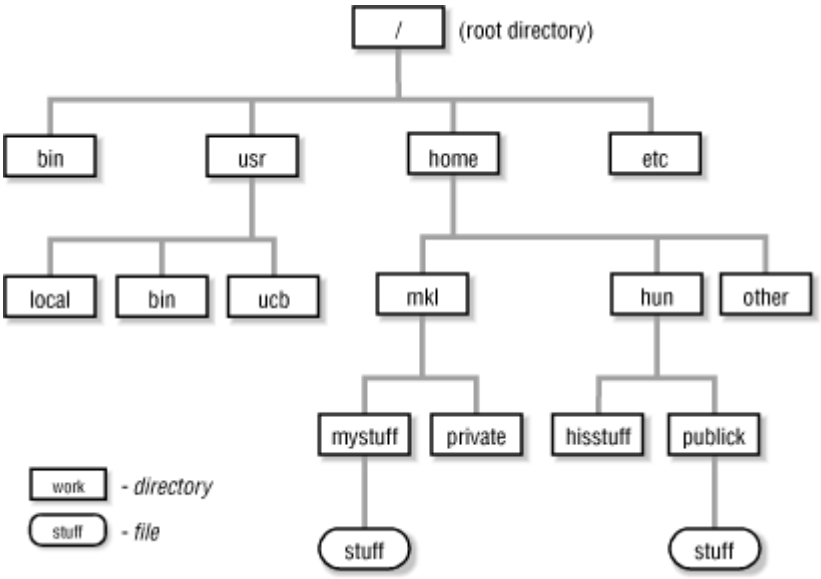
When you first open the desktop, you should find the LXTerminal open. The LXTerminal gives us access to the Linux command line.



Simplistically the command line is a text interface to the computer. It's often referred to as the shell, terminal, prompt. The shell allows us to control the computer with a set of UNIX commands. You can learn a bit of the UNIX shell and get a summary of some basic commands at this link.

Example Tree structure of UNIX File System

At this point, it would be a good idea to explore the filesystem. Here is a schematic of what the file system may look like. The thing to remember is that the file system has a tree structure.



- Use `pwd` to list your current directory.
- Use `ls` to view the files in your current directory.
- Use `cd [absolute/relative path]` to change directory.
- Use `cd ..` to go up a level.
- Use `cd .` to go to current directory (aka do nothing)
- `cp [OPTION] ... SOURCE DEST` to copy file or directory from source to direction
  - `cp my_file ../my_file`
- `mv` moves a file from source to destination
- 'touch file' creates an empty file
- `mkdir` makes a directory

You can use `--help` at the end of any command to show more information about a command

---

Example

Use `ls --help` to help you figure out what

```
ls -ltr
```

is doing in the command line.

Example

Make a new directory in the projects folder. Create a new file using the `touch` command.

Introduction to Git and Github

Version control tracks and manages changes to your software code. Git is a free and open-source version control system. Github is an online tool that allows you to host your project remotely. It serves as a place to back up code that you write if your local code gets lost. It's primarily used as a way for teams to share and work on code together. Lastly, Github is a place to share open-source projects with the community. This tutorial will set you up with the basics of using git and Github within the command line. This tutorial is based on what I found online but customized to use the terminal inside our `ams562-desktop.` We will follow the following steps.

Step 0 Install git and create Github account

- If you aren't working inside the ams562-desktop you will need to install git on your local machine.
- Once you have git you will need to create a Github account using the following link
- For the rest of the lecture I am going to assume to all of you are going to use the ams-desktop but you if you want to install git here is a link

Step 1 Create a local git repository

We are going to create a new project on your local machine. For example we can move to the project directory and create a new folder called `ams562_test` .

- [ ] move into project directory using `cd` command
- [ ] make directory using `mkdir` command

```
ubuntu@ams562-wrjvhq:~/project> mkdir ams562_test
ubuntu@ams562-wrjvhq:~/project> cd ams562_test
ubuntu@ams562-wrjvhq:~/project/ams562_test>
```

- [ ] We can initialize a git repository in the root folder of the project using the `git init` command.

```
ubuntu@ams562-wrjvhq:~/project/ams562_test> git init
Initialized empty Git repository in /home/ubuntu/project/ams562_test/.git/
```

Note

By using the `git init` command inside of your the `ams562_test` directory you are creating a new Git repository. Repo for short. Git will begin to track all of the changes inside of the folder.

Step 2 Add new file to the repo

Now we can go ahead and make some changes to the repository.

- [ ] Add a file to the repo using touch command or using text editor

```
ubuntu@ams562-wrjvhq:~/project/ams562_test> touch newfile
ubuntu@ams562-wrjvhq:~/project/ams562_test> ls
newfile
```

Git will not automatically track the changes to a file.

- [ ] Use `git status` to see which files knows up

```
ubuntu@ams562-wrjvhq:~/project/ams562_test> git status
On branch master

No commits yet

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

    newfile
```

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

Note

You choose the files you want git to keep track of by using the `git add` command. If there are certain files you don't want tracked you can add it to the `.gitignore` file. https://git-scm.com/docs/gitignore

Note about staging environment and commit

- A commit is a record of changes you have made to project since last commit
- Commits document the incremental changes to a project
- git status shows you the status of the
  - which files have changed
  - untracked files
- To add a file to a commit you need to use the `git add <filename>` command
- You can make a commit by using the `git commit` command

Step 3 Add file to staging environment

- [ ] Add file to staging environment `git add <filename>`
- [ ] Use `git status` to check the change in the repo

```
ubuntu@ams562-wrjvhq:~/project/ams562_test> git add newfile
ubuntu@ams562-wrjvhq:~/project/ams562_test> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   newfile

ubuntu@ams562-wrjvhq:~/project/ams562_test>
```

Step 4 Create a commit

- [ ] Create commit using `git commit -m "Your message"` command

```
ubuntu@ams562-wrjvhq:~/project/ams562_test> git commit -m "Hi Class ! :)"
[master (root-commit) d1b53f0] Hi Class ! :)
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 newfile
```

- [ ] Try typing git commit again

```
ubuntu@ams562-wrjvhq:~/project/ams562_test> git commit
On branch master
nothing to commit, working tree clean
```

- `-m` is an option flag that allows you to write a message about the changes
- It's a good idea to make the message be useful

Step 4a (optional)

- [ ] On your own time make a change to a file or add a newfile and commit it yourself. Try reproducing the steps
- [ ] Use `git log` to see a log of your repo

```
ubuntu@ams562-wrjvhq:~/project/ams562_test> git log
commit 340c2d73ffecb37ee3977dda38fbe6fa89fad98f (HEAD -> master)
Author: Adrian Hurtado <ahurta92@gmail.com>
Date:   Wed Aug 25 17:50:36 2021 -0400

    I forgot what I changed

commit d1b53f090719682917ec1a24335924b9916fcfac
Author: Adrian Hurtado <ahurta92@gmail.com>
Date:   Wed Aug 25 17:44:18 2021 -0400

    Hi Class ! :)
```

Step 5 Create a new branch

- Branches are useful when you have working code but you might just want to add a new feature without breaking the working code.
- Branches allow you to move back and forth between different states of a project
  - Say you want to work on a new feature without affecting the working code
  - You can work on the new feature in a seperate branch of the code
  - Once the code is working and ready you can merge the new feature branch into the main branch

- [ ] Create a new branch using `git checkout -b <branch_name>`

```
ubuntu@ams562-wrjvhq:~/project/ams562_git> git checkout -b "new_feature"
Switched to a new branch 'new_feature'
```

This will create a new branch and place you on the new branch

- [ ] Check that the new branch was created using `git branch`

```
ubuntu@ams562-wrjvhq:~/project/ams562_git> git branch
  main
* new_feature
```

- [ ] Go ahead and make a change to the repo. (e.g) make a change to your file or create a new file using touch
- [ ] Add the change and commit

Step 4 Create a new repository on Github

- [ ] Go to Github website
- [ ] On the left hand side you should find a green `New` button. Click it
- [ ] Give new repository a name and descrption
    - [ ] Do not add a README because you will be push the existing local repository to your new repository

Step 6 Set remote to point to new repository

Before we can actually push changes your local repository into the remote repository online we are going to need to generate a ssh key within

Step 6a Generate a new SSH key

Follow the steps listed on the Github Website. Especially if you are working on your own machine. The steps below will work for us working with ams-desktop

- [ ] Paste text bellow using your Github email

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

This creates a new ssh key, using the provided email as a label.

- [ ] Enter the file in which to save the key (you can use the default location)
- [ ] At the prompt enter a secure passphrase

Step 6b Add key to ssh key to ssh-agent

- [ ] Start ssh-agent in the background

```
eval "$(ssh-agent -s)
> Agent pid 59566
```

- [ ] Add the SSH private key to ssh-agent. With your key name

```
ssh-add ~/.ssh/id_ed25519
```

Step 6c Add your new key to Github account

- [ ] Use `cat` command to display the public key

```
cat ~/.ssh/id_ed25519.pub
```

- [ ] Copy and paste contents into new ssh key location on Github. Follow the instructions here

Step 6b Push changes onto remote

Push our local repo to the remote repo on Github

```
git remote add origin git@github.com:ahurta92/refactored-octo-parakeet.git
```

Here `origin` is pretty much an alias to the remote repository you want to point to.

First we want to push our main branch. We first need to checkout out the main branch which will actually be called master.

```
git checkout master
```

Then we are going to rename the main branch

```
git branch -M main
```

Last we are going to push the main branch into the remote repository.

```
git push -u origin main
```

You can inspect the changes to the remote repo online.

Step 7

Now we can push our branch to the Github repo. This allows other people to see the changes you have made. To push the branch you will use the `git push origin branch_name` command.

```
git push -u origin new_branch
```

Step 8 Create a pull request (PR)

This alerts the owner of the repo that you want to make a change to their code. The owner can then review the code to make sure the code looks good before it is added to the main branch.

Step 9 Merge a PR

Step 10 Get changes on GitHub back to your computer

You can then get the changes from Github back onto your computer using the `git pull origin main` commadn

```
git pull origin main
```

Summary

By doing this tutorial you should be familiar with the basics of git and Github. These are the commands that you should try and remember.

- `git init`
- `git add`
- `git commit`
- `git status`
- `git log`
- `git checkout -b <new branch>`
- `git checkout <branch_ranch>`
- `git remote add origin <remote_address>`
- `git remote add origin <remote_address>`
- `git push origin branch_name`
- `git pull origin branch_name`

For our homework you will need need to create a repository on Github and give me READ access so I can view your work.

Next Steps (git clone ams526notes)

Once in the shared directory you will use the git clone command to clone the ams562-notes repository into the shared directory. This is cloning the remote repository onto your local machine. For those who need a refersher on git or are using Git for the first time here is a useful link. I personally first learned Git from a free coursera class.

```
git clone https://github.com/ahurta92/ams562-notes.git
```

From here can change directories to `ams562-notes` and run the following commands to first download the neccessary packages.

```
pip3 install -r requirements.txt --user
```

Then to compile the notes.

```
make html latex
```

```
cd _build/latex
make &>/dev/null || echo "failed compiling latex"
```

Once the notes are compiles you should find them on your local machine under the `_build` directory. You can open these notes on your web browser by passing finding the `index.html` finding and opening. On my Windows machine the path looks like this `ive/Documents/GitHub/ams562-docker/ams562-notes/_build/html/index.html`

Using git pull to update the notes

From time to time I will be updating the notes. Making admendments, and adding content. In order to download those changes you can use the `git pull` command.

`git pull`

This will pull changes from the remote repository intor your own local repository.

After you pull changes you will need to build using the previous commmands

```
make html latex
```

VSCode Git integration

Optional Homework

1. Create a new repository and add it to Github