DudeWheresMyBike:  
Environment Specifications

Prepared: 17/02/2022, TK

Revisions:

# Cross Platform Issues Mean We Choose To Keep a Text Record of Environment Details

* Python 3.9.7
* pip install requests
* pip install sqlalchemy
* pip install -U Flask-SQLAlchemy
* pip install mysql-connector-python
* pip install flask
* pip install pandas
* pip install sklearn
* pip install gviz\_api

Installing the Environment

## Install MiniConda:

wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86\_64.sh

bash ./Miniconda3-latest-Linux-x86\_64.sh

For the above changes to take effect you need to close and reopen the current shell

# Create/Update Environment on EC2:

conda create -n comp30830py39\_dudeWMB python=3.9

conda activate comp30830py39\_dudeWMB

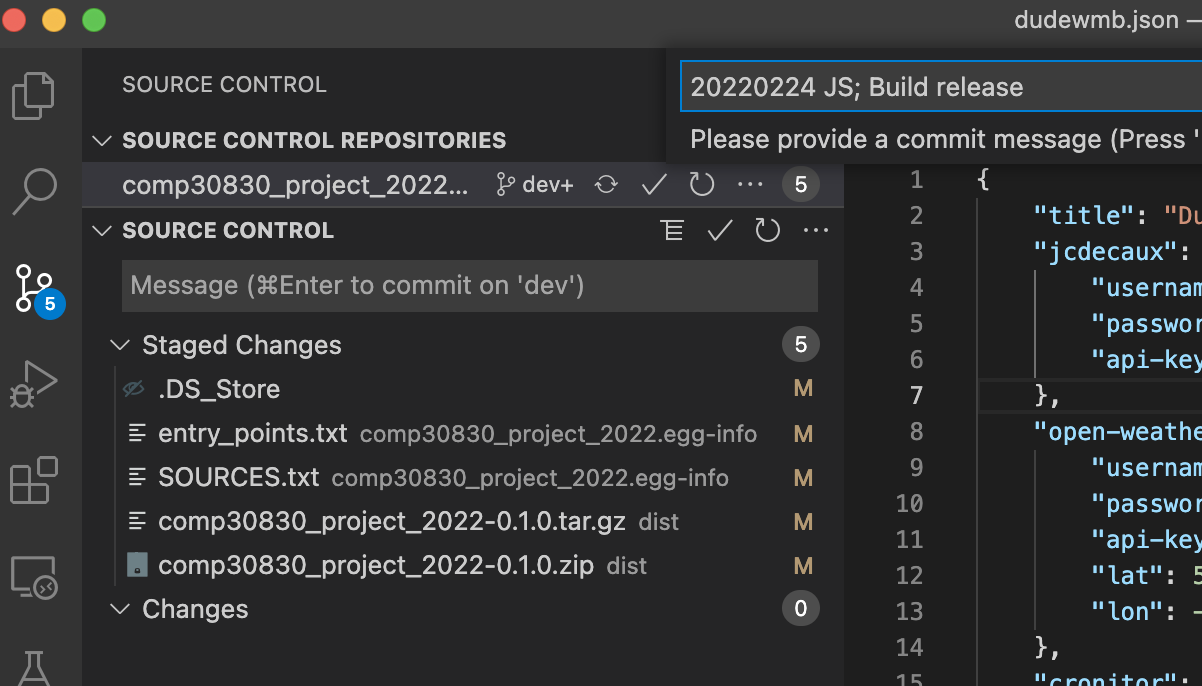
pip install -r requirements.txt (OR install the packages by hand?)

# Preparing a release

1. Activate environment
2. Move to project folder on your local computer
3. Make sure all development branches are complete and merged with the ‘dev’ branch. If they are not - discuss with the team to progress
4. Switch to the ‘dev’ branch
5. Update the application version number in Setup.py if you want to create a new release
6. Run python setup.py sdist --formats=gztar,zip



1. Commit created files to github and synchronise

  
Fig. 1: Committing a Completed Build to dev, before Merge to Main

1. Merge all changes into ‘main’, commit to github and synchronise again
2. You can now delete any complete branches

# Installing the software on EC2

If you’re upgrading from a previous version and the data loader is already running… you need to stop it:

sudo -E env "PATH=$PATH" dwmb\_scheduler.sh stop

Run the following command to install the package, using PIP, directly from github. In the example command below:

* NOTE: The “@dev” part of the URL
* This is the name of the branch in GIT you’re installing from
* If you’re installing from ‘main’ you can leave this bit out
* The branch name ***must come immediately*** after the main GIT url - you can not re-arrange this - it must come before any other parameter.

python -m pip install -e "git+<https://github.com/flintdk/comp30830_project_2022/@dev#egg=comp30830_project_2022>"

## Troubleshooting:

If you see a message along the lines of “Missing or invalid credentials. Error: connect ECONNREFUSED” then most likely what has happened is the security credentials that your VS Code installation are using have expired. You’ll need to re-establish them. To do this you:

### Windows:

Go to:

Control Panel -> Credential Manager -> Windows Credentials

I saw a whole heap of credentials in here beginning with ‘vscodevscode.github’ and ‘vscodevscode.microsoft’ … I removed them all! I then restarted VS Code. I then tried to re-install and when prompted by VS Code authorised the github access.

### macOS:

This process is probably similar (but different) on MAC. Jörg - can you document that process please when it occurs for you?

## 

## Commands to Run the Data Loader & Scheduler

\*\*\* FIRST You must create a file in your login directory “.bashrc\_conda” if it does not already exist

Paste the following text into the file:

# ~/.bashrc\_conda: snippet to enable conda environments in cron'ed jobs

# >>> conda initialize >>>

# !! Contents within this block are managed by 'conda init' !!

\_\_conda\_setup="$('/home/ubuntu/miniconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"

if [ $? -eq 0 ]; then

eval "$\_\_conda\_setup"

else

if [ -f "/home/ubuntu/miniconda3/etc/profile.d/conda.sh" ]; then

. "/home/ubuntu/miniconda3/etc/profile.d/conda.sh"

else

export PATH="/home/ubuntu/miniconda3/bin:$PATH"

fi

fi

unset \_\_conda\_setup

# <<< conda initialize <<<

Go to the folder “/home/ubuntu/src/comp30830-project-2022” and make sure that your properly customised dudewmb.json file is present and correct. This is where it should be - so that the web application AND the data loader can access the same file.

Run the Data Loader once from this directory, it should run fine: dwmb\_dl

Activate the scheduler: sudo -E env "PATH=$PATH" dwmb\_scheduler.sh schedule

Confirm the scheduler is indeed scheduled:

sudo -E env "PATH=$PATH" dwmb\_scheduler.sh show

## Installing a Apache Flask Server on EC2

If you have already installed Apache… don’t do this twice! You only need to restart Apache after an update.

We need a web-server that can serve a Flask application written in Python. There are loads. I chose Apache. There was no bias here - I used the logic ‘it’s a popular one’. Choice of server free to change. To install the server we activate our virtual environment and execute:

sudo apt-get install apache2

TERMINOLOGY: WSGI = Web Server Gateway Application, a standard web servers use to load information about web applications.

NOTE1: Everywhere I looked at the start for installing apache and mod-wsgi recommended doing it all in one go (i.e. install the ‘mod\_wsgi’ module required to support our python app and apache at the same time) as follows (DON’T RUN THIS):

sudo apt-get install apache2 libapache2-mod-wsgi-py3

… making sure to have '-py3' at the end of libapache2-mod-wsgi-py3 or the runtime will default to Python2.7

HOWEVER: the mod-wsgi module installed in that manner \*\*\*did not work\*\*\* with the python 3.9 used in our environment. The most recent mod\_wsgi version at that time had been compiled using python 3.8 and caused a conflict. That may no longer be the case. It might be possible to do this ‘all in one go, simply’. But our team ended up doing the install in steps. If you’ve tried this approach and it didn’t work you can remove the incompatible WSGI plugin using:

sudo apt-get remove libapache2-mod-wsgi-py3

(NOTE2: I had a lot of trouble getting the Python sys.path to point to an actual real directory (it was pointing to python 3.8, when I had python 3.9 installed). If found the following article quite useful in explaining where all the mysterious sys.executable, sys.path etc. variables are defined when you run Python

<https://towardsdatascience.com/python-and-the-module-search-path-e71ae7a7e65f> )

To install mod\_wsgi to the latest version, we must first install apache2-dev (it’s a requirement - see installation line below). Then we install mod\_wsgi with pip (in our virtualenv) and ***then*** install its module to apache system-wide.

1) Install mod\_wsgi in virtualenv with pip

conda activate comp30830py39\_dudeWMB

sudo apt-get install -y apache2-dev

pip install mod\_wsgi

3) Install into Apache (system-wide)

sudo /home/ubuntu/miniconda3/envs/comp30830py39\_dudeWMB/bin/mod\_wsgi-express install-module

sudo vi /etc/apache2/mods-available/wsgi\_express.load

… and add the following single line…

LoadModule wsgi\_module /usr/lib/apache2/modules/mod\_wsgi-py39.cpython-39-x86\_64-linux-gnu.so

sudo vi /etc/apache2/mods-available/wsgi\_express.conf

… and add the following single line…

WSGIPythonHome /home/ubuntu/miniconda3/envs/comp30830py39\_dudeWMB

4) Enable the module and restart Apache. Apache (once it’s installed) comes with a command to enable and disable modules (a2enmod ≈ apache2-enable-module) . The command looks as follows:

sudo a2enmod wsgi\_express

sudo service apache2 restart

If you mess up and need to remove the software because, for example, you forgot to activate your virtual environment, you can always:

sudo apt-get remove apache2

… go ahead… ask me how I know…

Our project \*\*\*needs\*\*\* to have a dudeWMB.WSGI file to tell the server where to find everything. This file is committed to GIT. Just be aware it’s there - and if stuff “just won’t work” that is one of the places you may have to look.

The server runs all its websites from a folder called /var/www/html (this is like the ‘htdocs’ folder we saw in XAMPP). But our project isn’t in that folder. So we have a choice. We can either copy the project into that folder every time we install, or we can create a ‘link’ (symbolic link) on our EC2 instance so that the project directory *appears* to be in /var/www/html. I chose to do the one off ‘link’ so we wouldn’t have to worry about it going forward.

sudo ln -sT /home/ubuntu/src/comp30830-project-2022 /var/www/html/dudeWMB

At this point we have apache up and running and our WSGI config ready to go. All we need to do is configure Apache to use our website as the ‘root’ or ‘default’ site. We have to edit the Apache configuration file to do this - it’s no biggie, just a matter of pasting in some text once you have the file open for edit:

sudo vi /etc/apache2/sites-enabled/000-default.conf

Paste in the following just after the initial comment in the very first “<VirtualHost \*:80>” section of the file. There should be nothing really except other comments in this section once you’re done:

ServerName www.dudewheresmybike.ie

ServerAlias dudewheresmybike.ie

ServerAdmin webmaster@localhost

DocumentRoot /var/www/html

WSGIDaemonProcess dudeWMB home=/var/www/html/dudeWMB/dudeWMB/ python-home=/home/ubuntu/miniconda3/envs/comp30830py39\_dudeWMB python-path=/var/www/html/dudeWMB:/home/ubuntu/miniconda3/envs/comp30830py39\_dudeWMB/lib/python3.9/site-packages threads=5

WSGIScriptAlias / /var/www/html/dudeWMB/dudeWMB/dudeWMB.wsgi

# Following line ensures you don't get the:

# UserWarning: NumPy was imported from a Python sub-interpreter but NumPy does not properly support sub-interpreters.

# ... error. See https://stackoverflow.com/questions/68849673/importing-numpy-shows-warning-when-running-in-mod-wsgi

WSGIApplicationGroup %{GLOBAL}

<Directory dudeWMB>

WSGIProcessGroup dudeWMB

WSGIApplicationGroup %{GLOBAL}

Order deny,allow

Allow from all

</Directory>

Now all you have to do is restart the server!!

sudo service apache2 restart (OR “sudo apachectl restart”)

### Debugging the Server

Internal Server Error? If anything goes wrong, first check the error logs using:

vi /var/log/apache2/error.log

Show the Apache Service Status:

systemctl status apache2

If you’ve made changes that are not showing up, e.g. the page still displays "Hello, World" or whatever and you’ve given your all trying to rationalise it… try restarting your apache server!

$ sudo apachectl restart