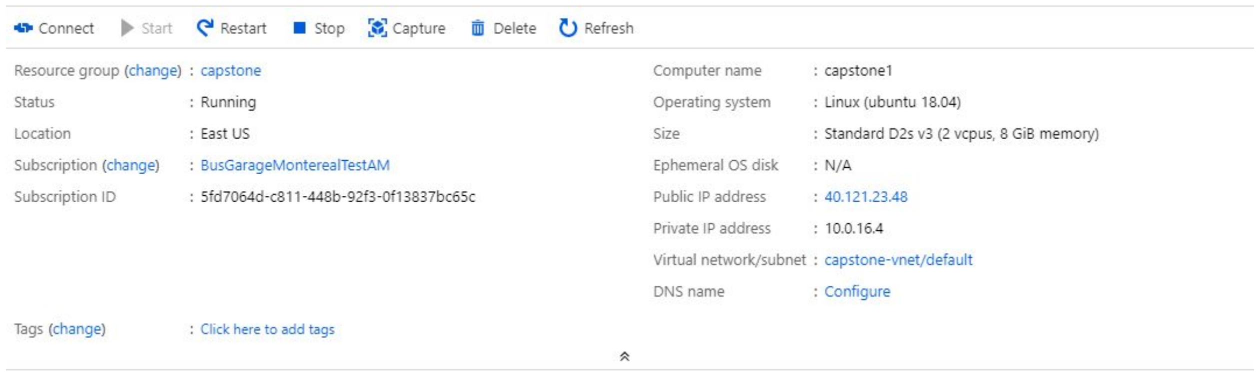
SkeYe360 Setup Instruction

There are two VMs used for this project:

VM1(Jenkins):
SSH : 40.121.23.48



Resource group (change)	: capstone	Computer name	: capstone1
Status	: Running	Operating system	: Linux (ubuntu 18.04)
Location	: East US	Size	: Standard D2s v3 (2 vcpus, 8 GiB memory)
Subscription (change)	: BusGarageMonterealTestAM	Ephemeral OS disk	: N/A
Subscription ID	: 5fd7064d-c811-448b-92f3-0f13837bc65c	Public IP address	: 40.121.23.48
		Private IP address	: 10.0.16.4
		Virtual network/subnet	: capstone-vnet/default
		DNS name	: Configure
Tags (change)	: Click here to add tags		

Username: Soen490
Password: Soen490skeye

VM2(dashboard + backend + database):
168.62.183.116
Username: Soen490
Password: Soen490skeye

Docker compose start frontend + dashboard + database:

Prerequisites

1. local dev environment [https://github.com/vincentsun870530/Soen490/tree/master/dev_env]
2. docker-compose [https://docs.docker.com/compose/install/]

Start

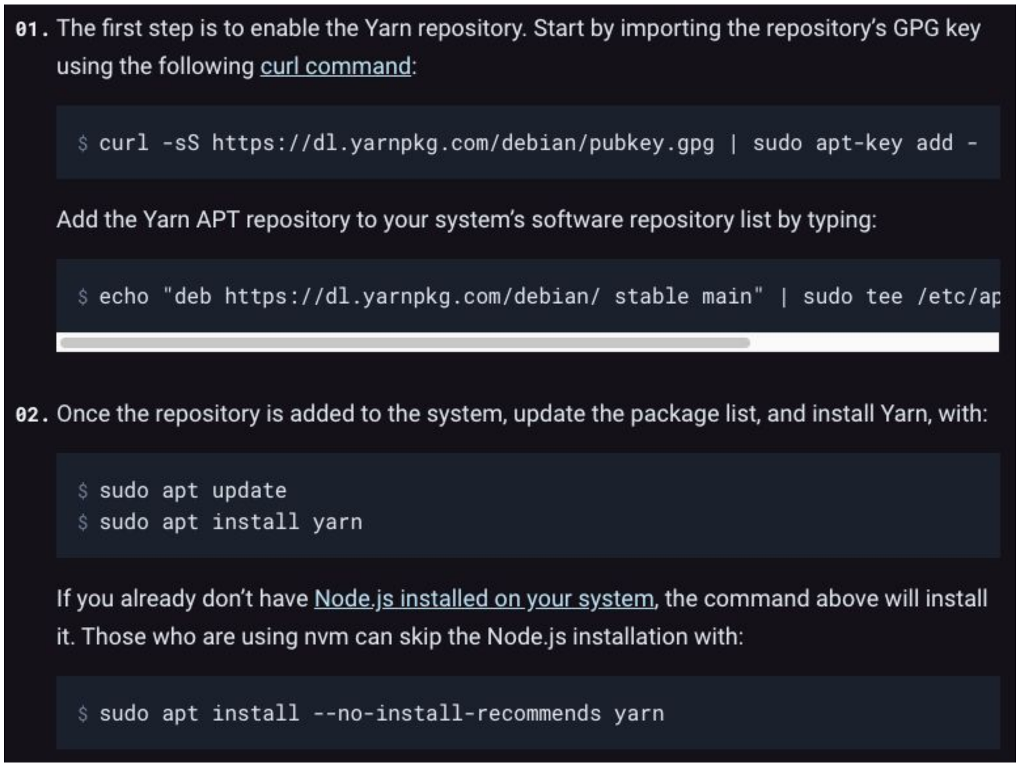
Under project folder:

```
docker network create --gateway 172.16.0.1 --subnet 172.16.0.0/24 app_net // This
command only needs to do the first time

cd project/


docker-compose up --build
```

Frontend Dashboard:

Setup procedure:

1. Install Yarn:



01. The first step is to enable the Yarn repository. Start by importing the repository's GPG key using the following curl command:

```
$ curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
```

Add the Yarn APT repository to your system's software repository list by typing:

```
$ echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/ap
```

02. Once the repository is added to the system, update the package list, and install Yarn, with:

```
$ sudo apt update
$ sudo apt install yarn
```

If you already don't have Node.js installed on your system, the command above will install it. Those who are using nvm can skip the Node.js installation with:

```
$ sudo apt install --no-install-recommends yarn
```

2. Navigate to the project's dashboard directory in SkeYe360/project/frontend/360_dashboard
3. Run the command yarn to install all the dependencies your machine:

```
yarn
```

4. a)To start the dashboard using yarn locally, run this command:

```
yarn start
```

5. a)To start the dashboard using yarn that would connect to an external backend server, run this command:

```
REACT_APP_API_URL=YOUR_SERVER_IP:8000 react-scripts start
```

Backend

REST API:

Setup procedure:

1. Install python3:

```
sudo apt-get install python3-venv
```

2. Create python virtual environment(not necessary)

```
%% Examples
lisun@devenv:~/courses/profiles-rest-api$ python3 -m venv ~/env
lisun@devenv:~/courses/profiles-rest-api$ source ~/env/bin/activate

%% Check python version, and you should see 3.5.X
(env) vagrant@devenv:~$ python --version
Python 3.5.6

%% Upgrade pip version
(env) vagrant@devenv:~$ pip install --upgrade pip
Successfully installed pip-20.0.2
```

3. Install dependencies:

```
%% Use pip to install dependencies under directory
/SOEN490/project/backend/360_django/djangosite, and you should not have errors.
(env) vagrant@devenv:/SOEN490/project/backend/360_django/djangosite$ pip install -r
requirements.txt
```

4. Start Django application

```
%% Start Django first time, and you need to do data migration
(env) vagrant@devenv:/SOEN490/project/backend/360_django/djangosite$ python manage.py
makemigrations
(env) vagrant@devenv:/SOEN490/project/backend/360_django/djangosite$ python manage.py
migrat(env) vagrant@devenv:/SOEN490/project/backend/360_django/djangosite$ python
manage.py runserver 0.0.0.0:8000

Django version 2.2, using settings 'djangosite.settings'
Starting development server at http://0.0.0.0:8000/
```

```
Quit the server with CONTROL-C.

%% After first time, you can run following command directly
vagrant@devenv:/SOEN490/project/backend/360_django/djangosite$ python manage.py
runserver 0.0.0.0:8000
```

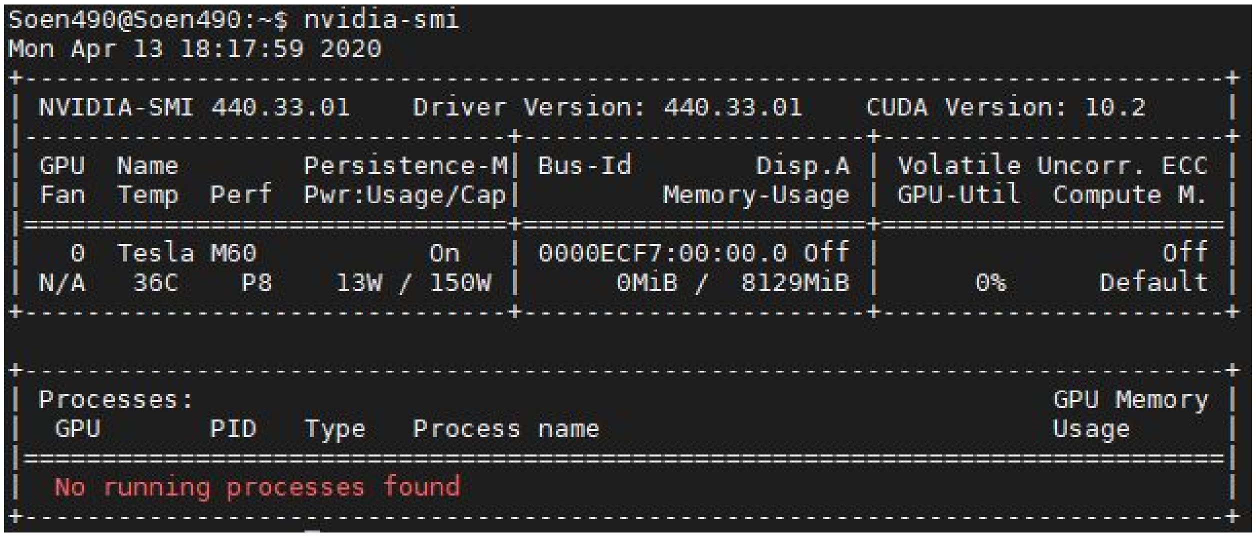
AI module:

Setup procedure:

1. Install nvidia driver:

```
sudo apt-get update
sudo apt install ubuntu-drivers-common
sudo ubuntu-drivers autoinstall
sudo nvidia-smi
```

Expected output when driver is installed and it's version:



```
Soen490@Soen490:~$ nvidia-smi
Mon Apr 13 18:17:59 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla M60           On   | 0000ECF7:00:00.0 Off |                  Off |
| N/A   36C    P8    13W / 150W |      0MiB /  8129MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

2. Install nvidia CUDA:
download deb from:
https://developer.nvidia.com/cuda-10.1-download-archive-base?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=debnetwork

```
sudo dpkg -i cuda-repo-ubuntu1804_10.1.105-1_amd64.deb
sudo apt-key adv --fetch-keys
https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/7fa2af80.pub
sudo apt-get update
sudo apt-get install cuda
```

3. Install python3:

```
sudo apt install python3-pip
sudo apt-get install libopencv-highgui-dev
```

4. Install project dependencies:

```
cd /home/Soen490/Soen490/project/backend/backend_django/camera/
```

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

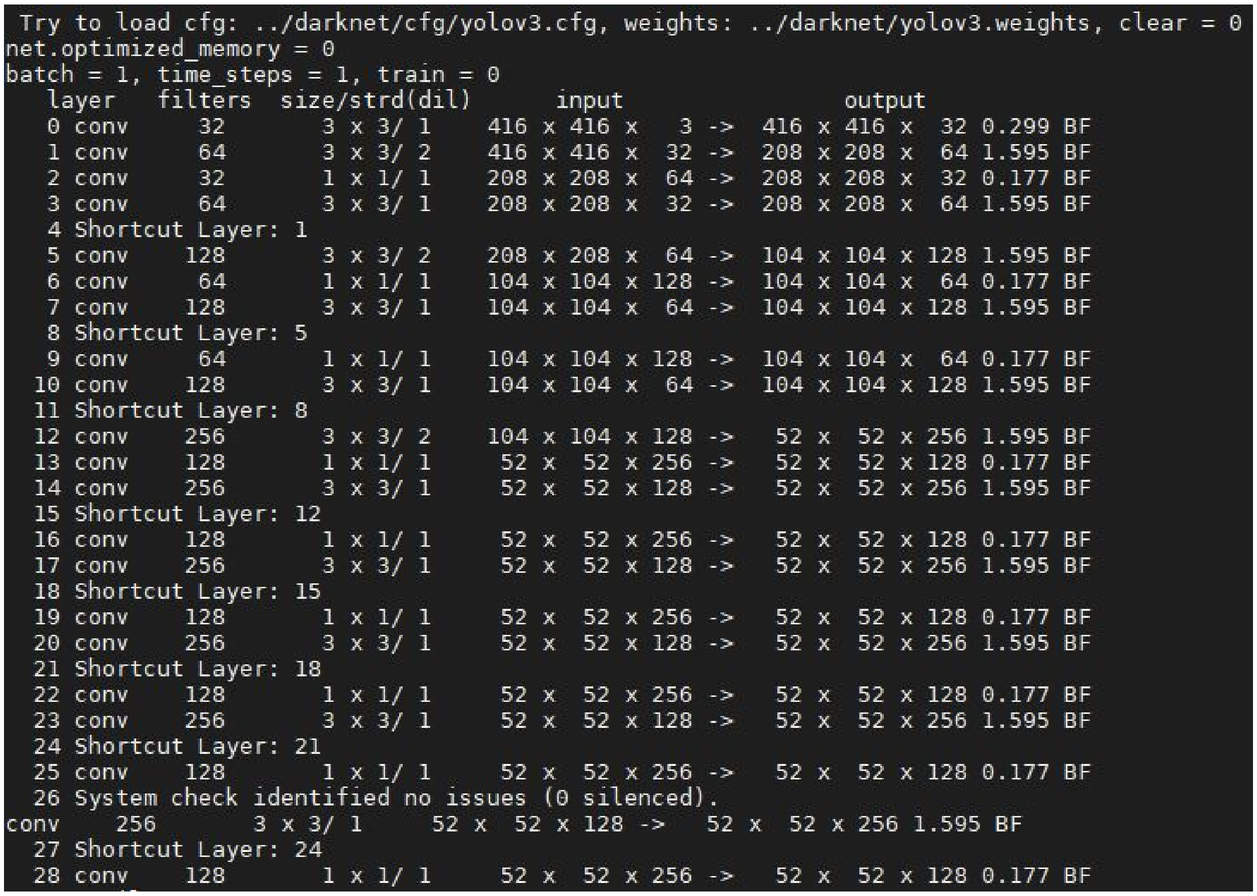
5. Configure:
upload “yolov3.weights” (available on google drive) to
/home/Soen490/Soen490/project/backend/backend_django/darknet/
upload video file “20191117_1600.mp4” (available on google drive) to
/home/Soen490/Soen490/project/backend/backend_django/camera/

Operating instruction:

1. Open terminal

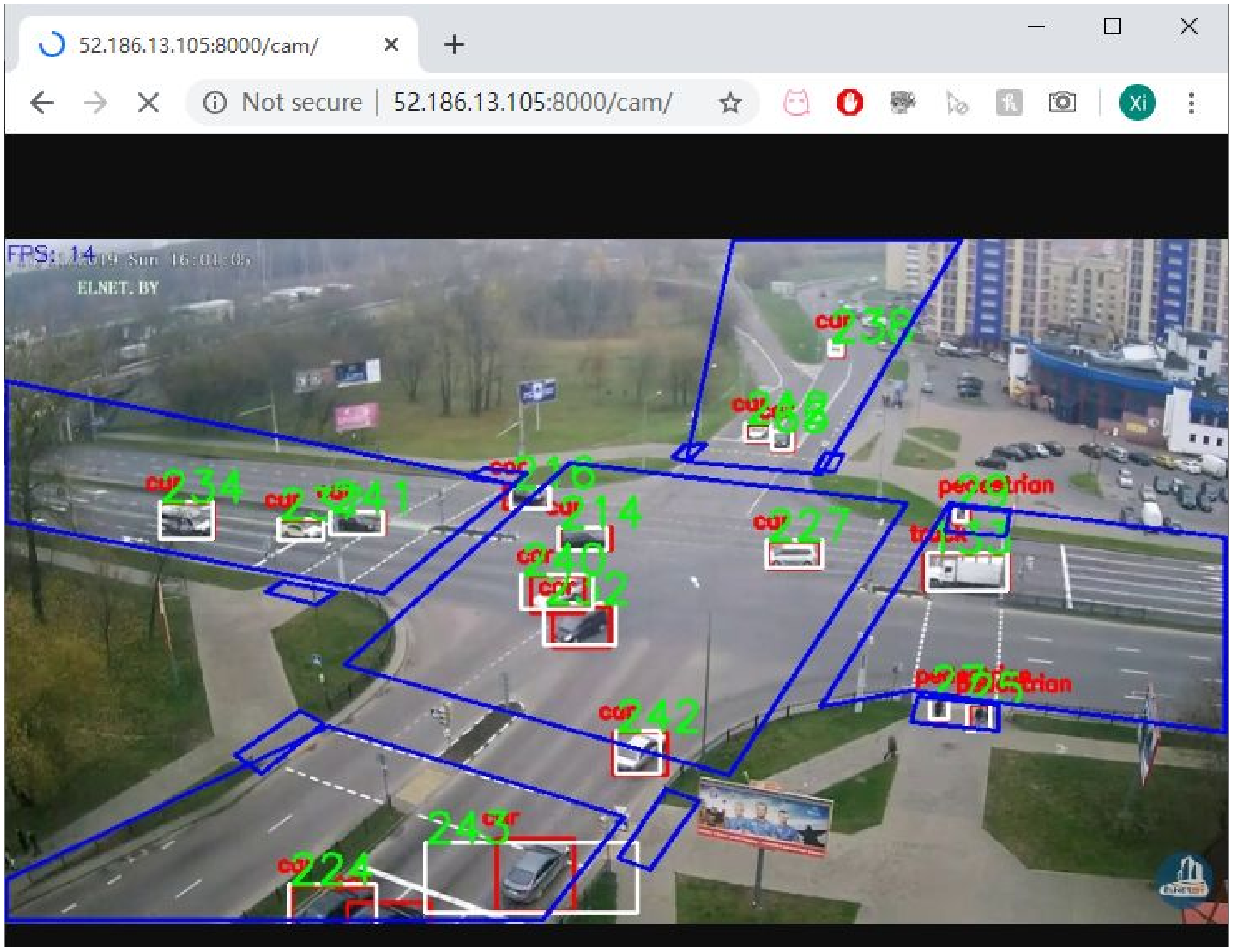
```
cd to /home/Soen490/Soen490/project/backend/backend_django/camera
python3 manage.py runserver 0.0.0.0:8000
```

Expected starting phase of object detection.



```
 Try to load cfg: ../darknet/cfg/yolov3.cfg, weights: ../darknet/yolov3.weights, clear = 0
net.optimized_memory = 0
batch = 1, time_steps = 1, train = 0
   layer   filters  size/strd(dil)      input                output
   0 conv     32       3 x 3/ 1    416 x 416 x   3 ->  416 x 416 x  32 0.299 BF
   1 conv     64       3 x 3/ 2    416 x 416 x  32 ->  208 x 208 x  64 1.595 BF
   2 conv     32       1 x 1/ 1    208 x 208 x  64 ->  208 x 208 x  32 0.177 BF
   3 conv     64       3 x 3/ 1    208 x 208 x  32 ->  208 x 208 x  64 1.595 BF
   4 Shortcut Layer: 1
   5 conv    128       3 x 3/ 2    208 x 208 x  64 ->  104 x 104 x 128 1.595 BF
   6 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
   7 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
   8 Shortcut Layer: 5
   9 conv     64       1 x 1/ 1    104 x 104 x 128 ->  104 x 104 x  64 0.177 BF
  10 conv    128       3 x 3/ 1    104 x 104 x  64 ->  104 x 104 x 128 1.595 BF
  11 Shortcut Layer: 8
  12 conv    256       3 x 3/ 2    104 x 104 x 128 ->   52 x  52 x 256 1.595 BF
  13 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  14 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  15 Shortcut Layer: 12
  16 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  17 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  18 Shortcut Layer: 15
  19 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  20 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  21 Shortcut Layer: 18
  22 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  23 conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  24 Shortcut Layer: 21
  25 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
  26 System check identified no issues (0 silenced).
conv    256       3 x 3/ 1     52 x  52 x 128 ->   52 x  52 x 256 1.595 BF
  27 Shortcut Layer: 24
  28 conv    128       1 x 1/ 1     52 x  52 x 256 ->   52 x  52 x 128 0.177 BF
```

Expected video stream output of object detection:



Data Analytics:

Setup procedure:

Installation

1. Navigate to the 360_data_analytics folder:
   ```
   cd Soen490/project/data_analytics/360_data_analytics
   ```
2. Install the common requirements:
   ```
   pip3 install -r requirements.txt
   ```
3. Install the specific dependencies:
   ```
   pip3 install pmdarima
   ```

Operating instruction:

Execution:

1. Navigate to the arima folder:
   ```
   cd analytics_models/arima
   ```
2. Execute arima:
   a. Running it only once:
      ```
      python3 runArimaOnce.py
      ```
   b. Running arima as a server:
      ```
      python3 arimaScheduler.py
      ```

The python file arimaScheduler will wait until 2am to start the process.

Database:

Setup procedure:

How to start mongo container manually?

1. Under project/database/360_mongo folder
2. Build MongoDB Image
   `docker build -t [Create a image name] .`
   Example: docker build -t mongo-test . (PS: don't forget '.' symbol)
3. Build MongoDB Container
   `docker run --name [Create a container name] -d -v /data/db:/data/db -p 27017:27017 [Image name you created]]`

Example: docker run --name my-mongo -d -v /data/db:/data/db -p 27017:27017 mongo-test

3. Execute your MongoDB container `docker exec -it [Container name you created] bash`
   Example: docker exec -it my-mongo bash
4. Create a MongoDB admin user

- `mongo`

```
use admin

db.createUser(

{

user: "myUserAdmin",

pwd: "abc123",

roles: [ { role: "userAdminAnyDatabase", db: "admin" },

         { role: "dbAdminAnyDatabase", db: "admin" },

         { role: "readWriteAnyDatabase", db: "admin" } ]

}
)
```

How to test mongo container connection?

1. Access local mongoDB container though linux command?

- `mongo 127.0.0.1:8300/admin -u myUserAdmin -p abc123`

2. Access local mongoDB container though python script?

- `pip install -r requirement.txt`
- `python dbConnectionCheck.py`

More References:

https://docs.google.com/document/d/14FUxzs3lFJFjCWdgBlBoZr-zRnH0Xz0L2c_v3M_xxf4/edit

Jenkins:

Setup procedure:

0. Install docker if necessary.

```
sudo apt install docker.io
```

1. Create a Jenkins container and run in the background.

```
sudo docker run \
-u root \
-rm \
-d \
-p 8080:8080 \
-p 50000:50000 \
-v jenkins-data:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
 jenkinsci/blueocean
```

2. Unlock Jenkins. Navigate to http:localhost:8080 and enter Administrator password found at /var/jenkins_home/secrets/initialAdminPassword. The password can be retrieved using the following command.

```
docker exec -it jenkins /bin/bash -c \ "cat /var/jenkins_home/secrets/initialAdminPassword"
```

3. Follow the instructions prompted to create the first admin user and configure the Jenkins URL.

4. From Jenkins' dashboard, navigate to *configuration* > *Manage Jenkins*.

5. Enter github repository URL under **Github Pull Requests** as Published Jenkins URL.

6. From Jenkins' dashboard, create a *New Item*.

7. Enter the desired name and select *Multibranch Pipeline*.

8. From the newly created job, navigate to *Configure*.

9. Under **Branch Sources** section, select Github. Enter the repository HTTPS URL and select proper credentials if applicable.

10. Configure desired Jenkins behaviors (ie. when will Jenkins be triggered) under the *Behaviors* section.

11. Apply and save changes.

Operating instruction:

Using the instructions above, Jenkins will execute the job depending on the preferences set in the job settings (*behaviors* section). Under the main project folder, a Jenkinsfile should be developed to execute the project tests. Jenkins can be accessed through localhost:8080 and localhost:8080/blue/organizations/jenkins.