# Harshika Pareek

**INTRODUCTION**

As per the Problem Statement we have to create a development environment using Virtual box having ubuntu image in it. Once done, install docker on the Ubuntu machine and build the docker image followed by docker container and perform the Command line Argument task using “Containers” with the Help of Python Script.

**ATTACHED FILES**

1. Assignment.py – Python Script (Command line argument)
2. Test1.json and Test2.json: Json files
3. Dockerfile: A Dockerfile having path /tmp (For Testing Purpose)
4. project3\_Dockerfile: A Dockerfile having path /root (where the actual Json files Located)

**SCREEN\_SHOTS**

Refer Screen-Shot for each and every task performed.

**TASK1**

**Creating a development environment**

1. Install a virtualization tool (VBox / VMware) and create a ubuntu virtual machine.

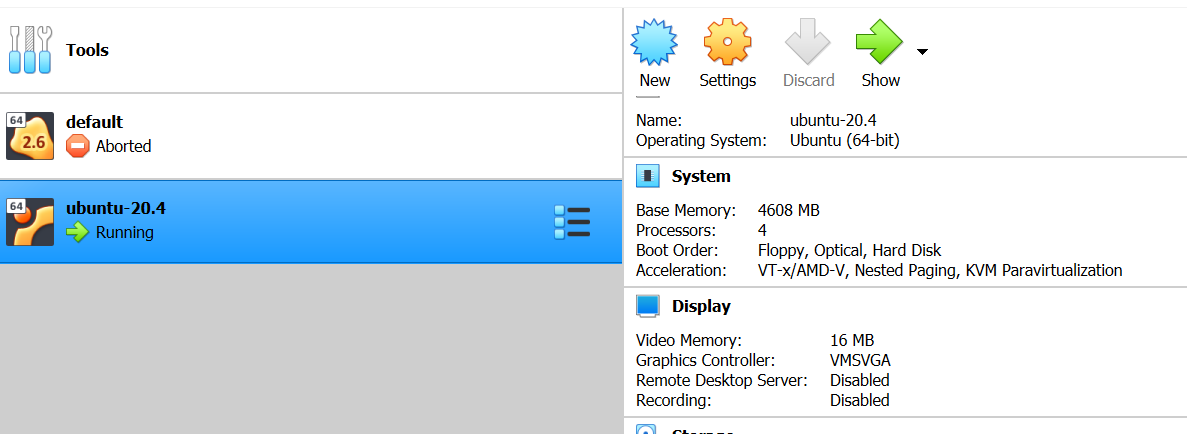
**Steps followed:**

1.Install Oracle VM VirtualBox Manager [1]. After installing the virtual box, we will create a ubuntu virtual machine.

2). In order to create an ubuntu virtual machine, Firstly, download the ubuntu iso image(20.4LTS) from the URL[2],Go to the VM Box manager >New> Create Virtual machine (Hard disk)>Storage on physical hard disk (We will choose Dynamically allocated so that memory space can be provide during the execution of the program if needed)>File location and size> Create.

3)The above process will create an ubuntu machine in the VM. Now go to the settings>General>Advanced>shared option and drag and drop click on “Bidirectional (So that we can copy files from our host machine to virtual machine which is ubuntu)

4)Now we have to Mount out .iso image to our ubuntu machine, go to Storage>Empty>give the path of .iso from host machine>ok Now we are ready with our machine.



1. **Login to the virtual machine, open the terminal and execute commands to find the following**

**2.1 Current CPU load**

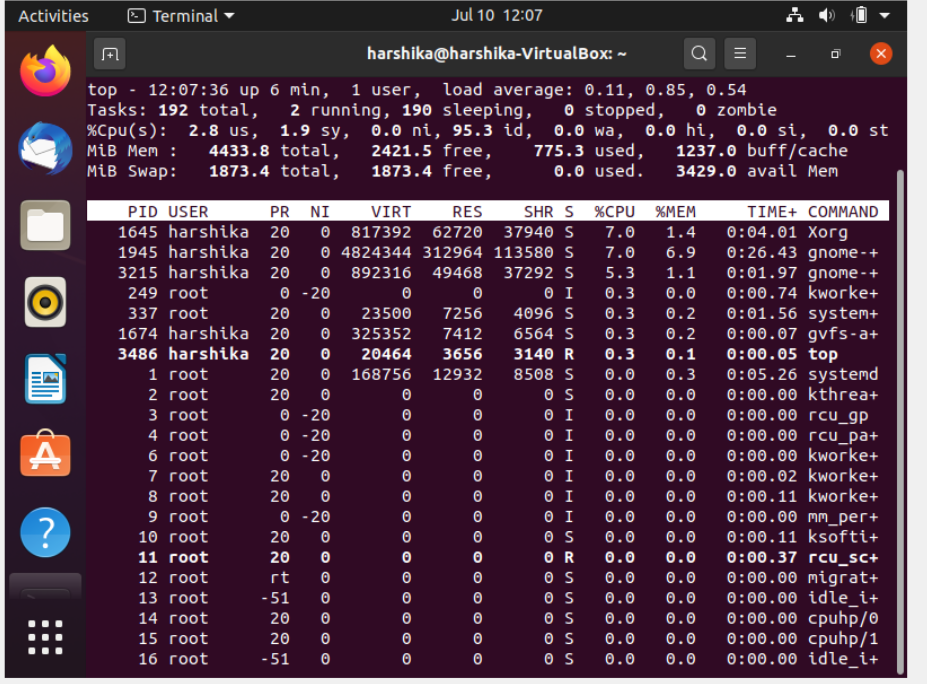
Why Checking CPU Utilization is Important

Once CPU is being used by one process, it is unavailable for processing other requests. Rest of pending requests must wait until the CPU is free. This becomes a bottleneck in the system. Following are the commands which can help us to check the CPU Utilization

1.$top

Number of CPU allocated :4

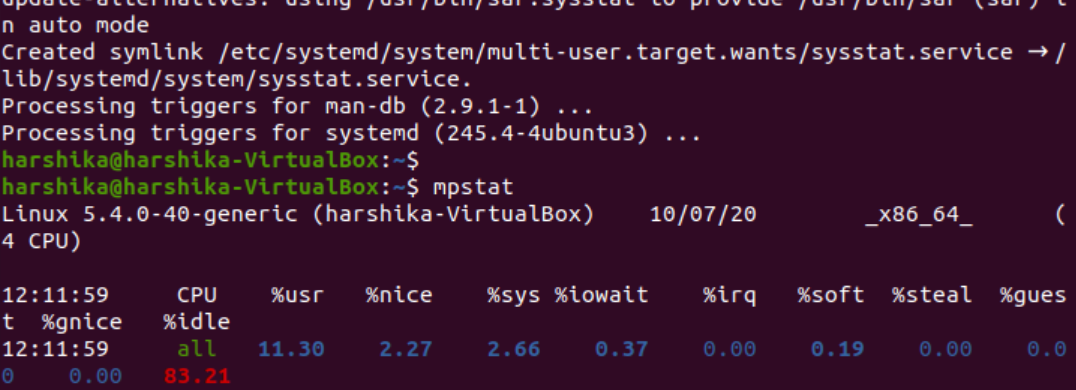
Sample output (Screen-shot)



**Output Description**

Top command provides the summary statistics of the resources being used currently. Looking at the output 95.3 id which means 95.3% of my CPUs are idle and 2.8% in used now. Point to note here, we can customise the options available such as Press “S” to change the Refresh window time, Press “I” to display only processes which are currently using CPU.

1. In order to display each CPU utilization, we can use the command mpstat, as per prerequisites we need to install apt-get install sysstat



**2.2 Available memory**

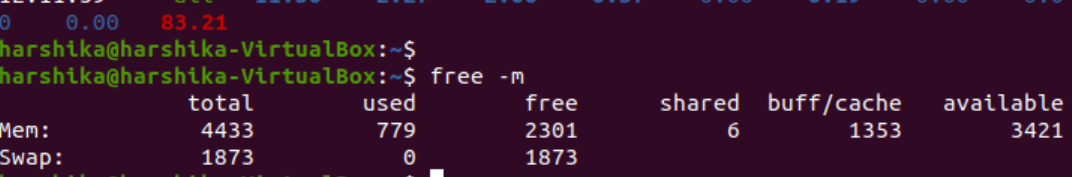
As the Linux users, we need to make sure how much RAM resources our system uses and how much it is free now.

The free command

Command: free -m

Here, m flag means that the information will be displayed in MBs.

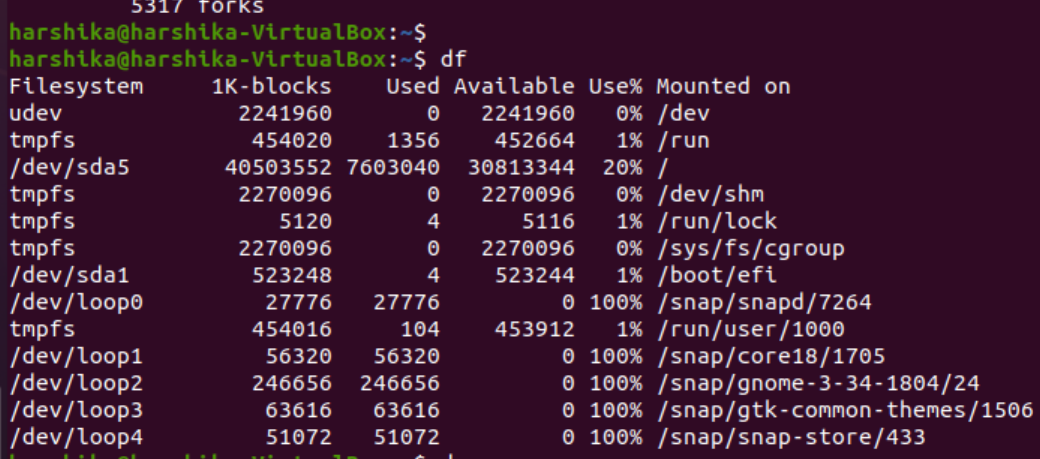
The available column indicates the available memory. The used column in the swap entry is also 0 which means that it is all unused and thus free.



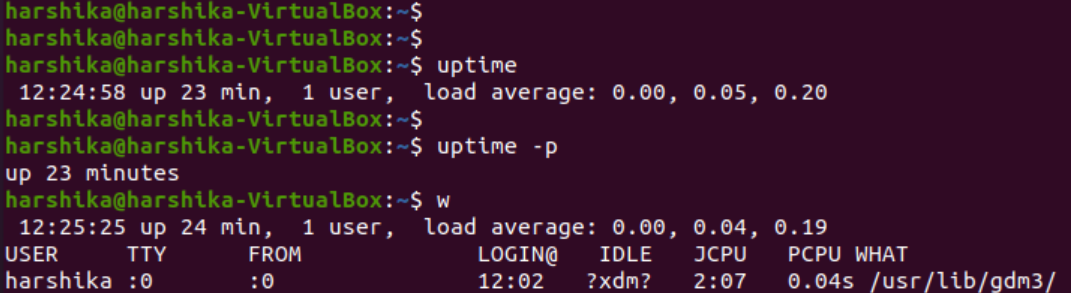
**2.3 Available disk space**

Command: df

Shows the amount of disk space used and available on Linux file system.



* 1. **Uptime**

Command: $uptime

**Description:**

12:24:58: State the current time

Up 23 min: How long the system has been running.

1 user: How many users currently logged on

0.00,0.04,0.19: Load Average

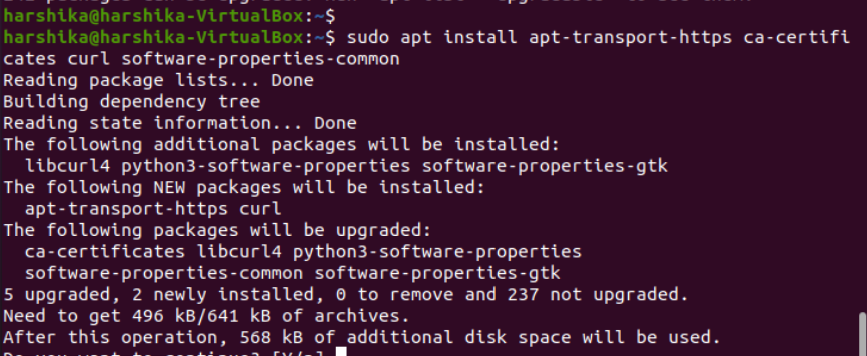
**1.3 Install docker-ce latest version in the created virtual machine. Show the output of `docker info` command**

**Task Description:** Docker is an open source utility which eliminates the repetitive task in software development. There are two versions of docker Docker CE(Community edition) and eDocker EE (Enterprise Edition). For the small-scale project, we use docker ce[3]

**Step to install Docker**

**Step1)** Update software repositories: By using this command to make sure if we update all the existing software in the database.

Command: sudo apt-get update

**Step2)** Download Dependencies

Command: sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

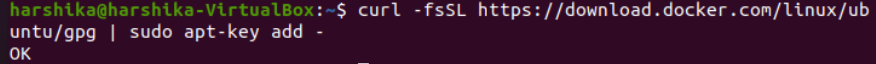
**Breakdown of terms:**

apt-transport-https: Allows the package manager to transfer files and data over https

ca-certificates: Allows the system (and web browser) to check security certificates

curl: This is a tool for transferring data

software-properties-common: Adds scripts for managing software

**Step3:** Add Docker’s GPG Key

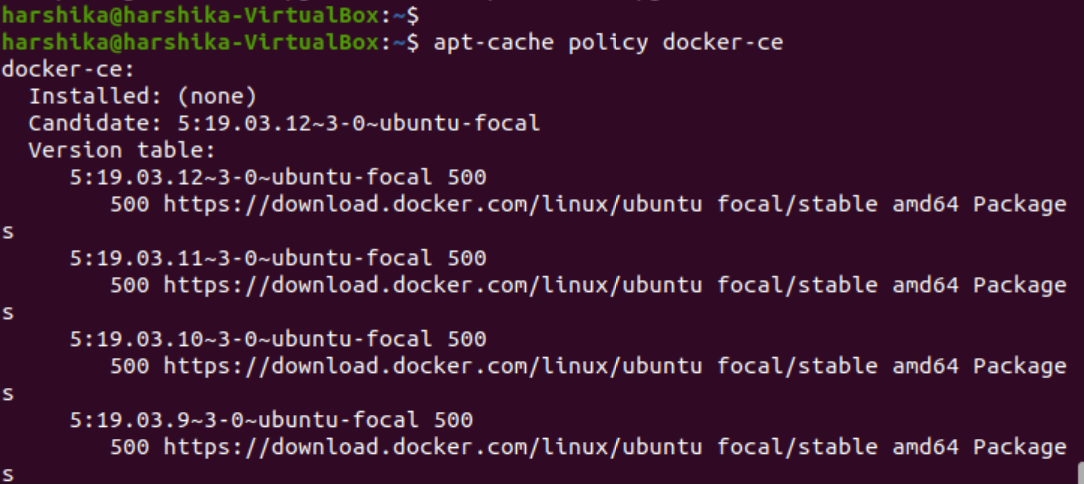
The Docker’s GPG key is the security feature which make sure that the software you are installing is authentic or not. If we get the response as “ok” which means the software is authentic.

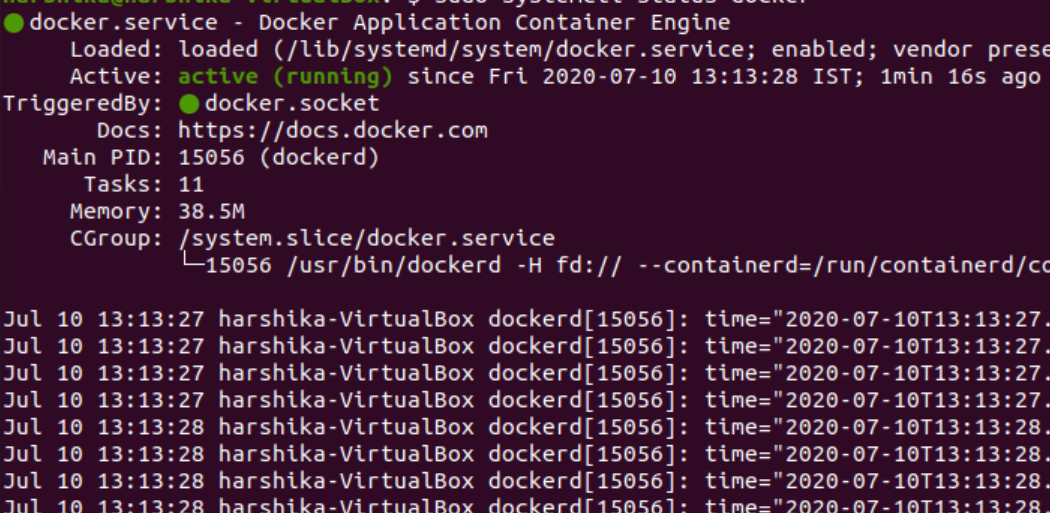
**Step4:** Install the docker Repository

Command: sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

Description: $(lsb\_release -cs) this will scan and return the code name of the ubuntu installation in this case it is Bionic. -stable refers the type of docker release.

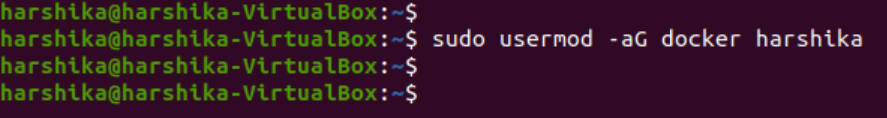
**Step5**: Update Latest Version of Docker

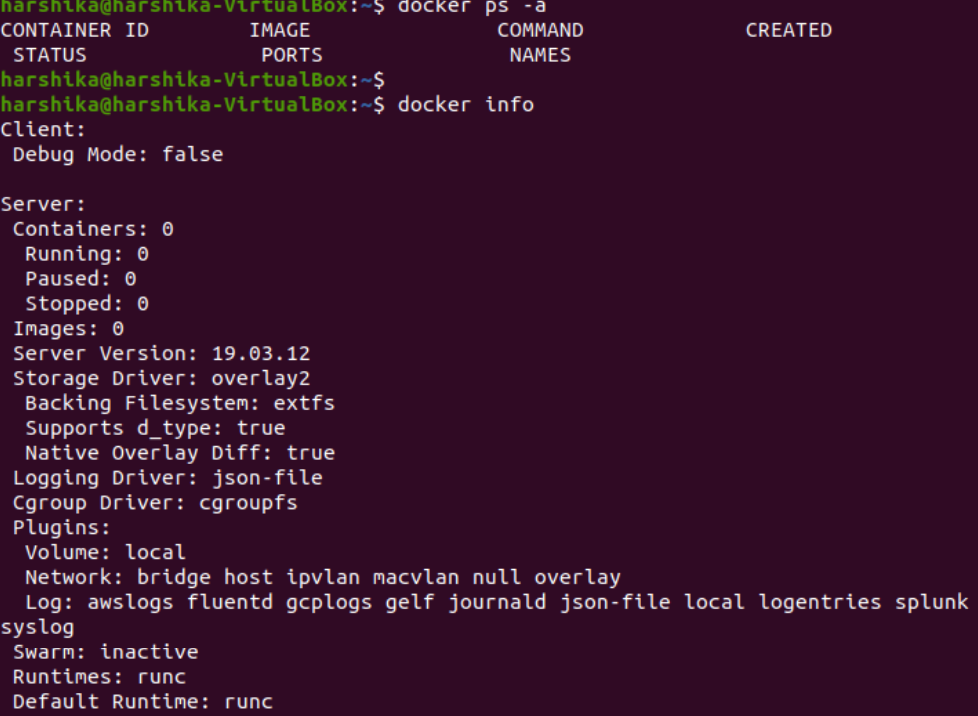
Command: sudo apt-get install docker-ce

**Step6**: Once done with docker -ce installation check the status of docker

The status should be Active.

**Step7**: Now I want to run my docker with non-root user so that my container can be used to obtain the root privileges. I will add the docker group



**Final Result: Output of “DOCKER INFO”**

**Challenges**

After Installation of Docker, while connecting to it, got stuck with the error “Got Permission denied while trying to connect to docker daemon socket.

Resolution: After browsing this[4], I got to know I would need to log out and log back in so that your group membership is re-evaluated or type the following command:

su -s ${USER}

**TASK2**

**Creating a python application and test it in the development environment created in earlier.**

**Please share the source files with your report.**

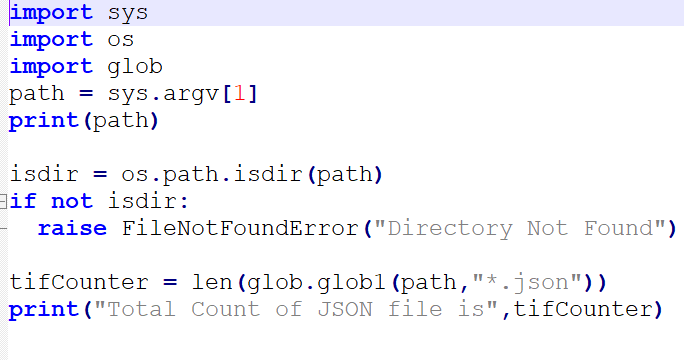
**1. Create a simple python script that does the following.**

**a. It accepts one command line argument. The expected argument should be path to a directory.**

**b. Upon execution the script will check if the directory exists.**

**i. If the directory doesn’t exist, the script should print a warning that directory doesn’t exist and it should terminate gracefully.**

**ii. If the directory exists, the script should check if there are any .json files exists in that directory. It should print the number of .json files exists in that directory to the console output.**

**Solution:** (Attached Python Script as Assignment.py)

**TASK3**

**3.1 Create a simple Dockerfile that will build above application to a docker image.**

Process1: Creating a docker file

1. Project 3:

2. Scope:

a. Create a python script (Assignment.py) to count the .json files of the given directory by argument. (Attached)

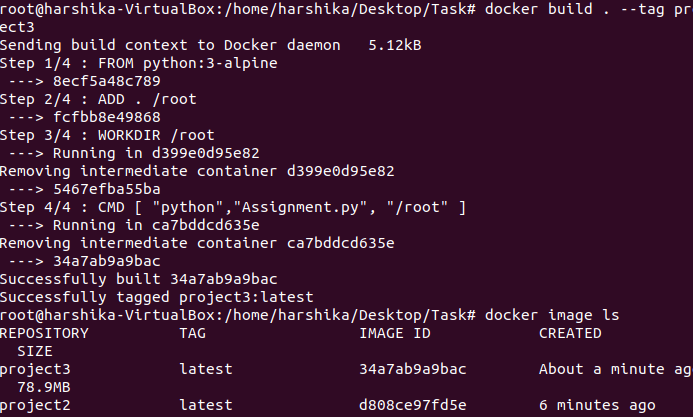
**Script Description**: In this (Assignment.py) script, we will provide the path of directory as an Argument and will check for the condition, if Provided Json file Exist in the directory or not. If yes, the Script Output Should be the Count of the JSON files (In my case it is 2) and if “No” the output should be Zero (0).

1. Create a docker file with python 3.x (Attached)

Creating a Docker File:

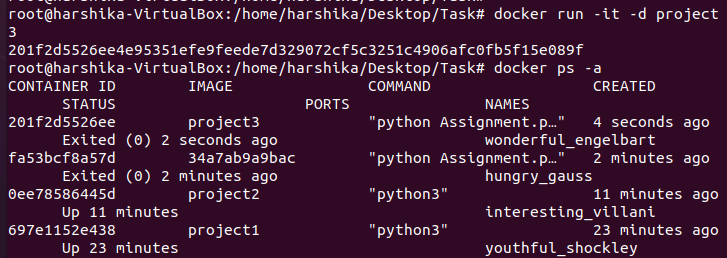
* Add python script (Assignment.py) and few .json files to /root
* Change the working directory to /root
* executes the python script (Assignment.py) with directory argument (/root)

**3.2** **Build a docker image from the Docker file**



**Command: docker build - -tag “name of the image” (project3 in my case)**

1. Once the image will download (The image will be based on the alpine python docker image), we can check the images available in our Repo by using command docker image ls, this command will provide the output with the Image ID.
2. Now, we will run the image to create a container out of it.

Command:

**Description:** we will run docker image in the interactive mode(-it), the benefit of using the interactive mode is we can execute commands at the time of running the container. Here -d stands for detached mode which means Docker containers runs in the background of your terminal. After this we can expect our container to be listed, we can verify all the running containers by using command docker ps -a.

NOTE: The Status of the running containers Must be “UP”.

**3.3**” **During the execution of the docker container, it should accept command line argument and pass it to the script created in the previous step”**

**Solution**: In order to perform the above task, we will,

Execute the docker logs command

Docker Logs Command: To get the Console Output

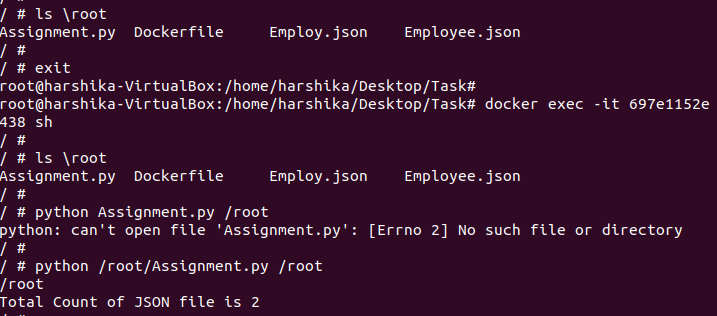
Command: docker logs $container id /path (where our json files will be stored/placed) (Screen-shot)

**OUTPUT:**



Here the output above shows the total number of Json files available in our /root directory.

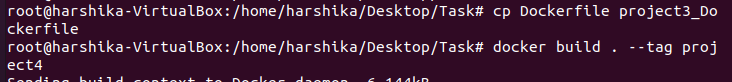
NOTE: In order to cross verify the output, we can go inside the path and check manually using the command /sh or /bin/bash

**Note: I have renamed the JSON files as Test1.json and Test2.json.**

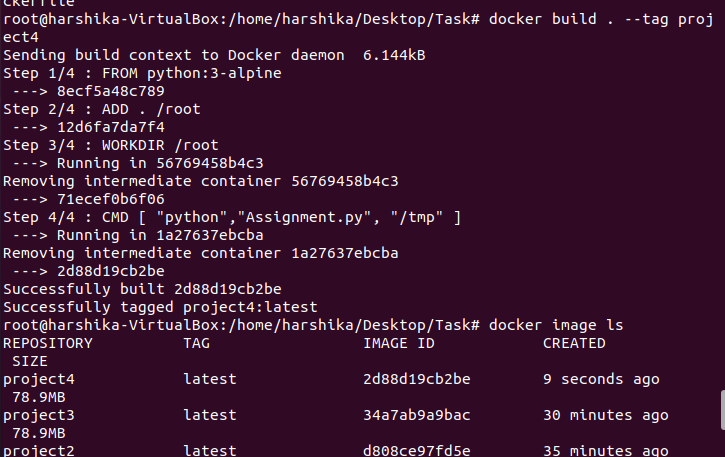
**TESTING**

As per the above discussed process we have got the expected result where our JSON files were located inside the /root directory and we got the JSON file Count as 2.

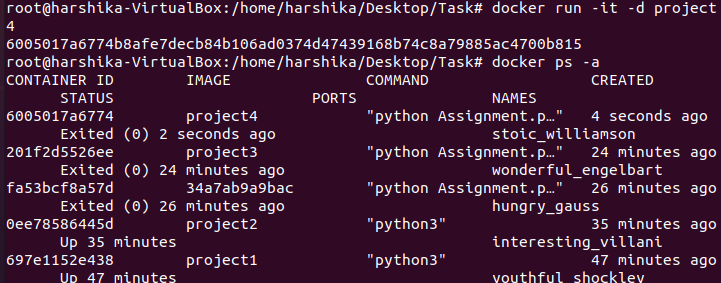
SCENARIO**: Now What if we change the path of our directory from /root to /tmp and check for the expected number of JSON file count?**

**We will copy the same docker file with the different name (Dockerfile)**

**Expected Output: Zero (0)**

1.Create Another Docker image with the image name “project4” with the command

1. Run the Container for “project4”



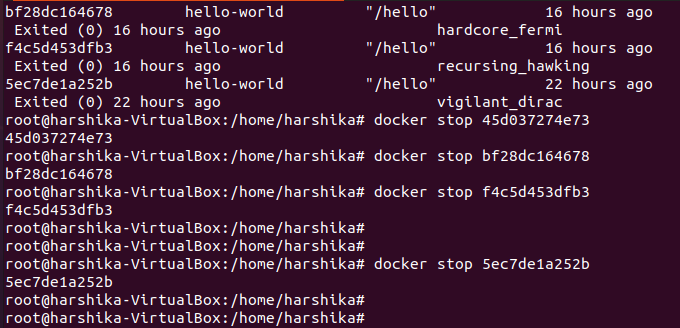
1. Now provide the path /tmp and check for the output

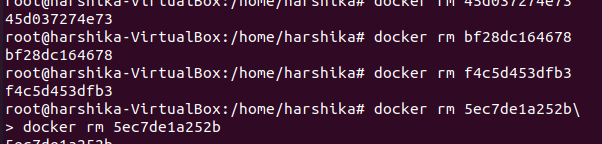


**Actual Output**: Now if we look at the Count of JSON file, resulted Zero (0).

**ASSIGNMENT CHALLENGES:**

1)Firstly, while working with the Docker Image Build, I faced issue downloading the image then I made changes in the Dockerfile and have kept the python3:alpine where it will download the image from the repo.

2)After downloading the image, the Status of Container was continuously showing “Exited” instead of “UP”, in this case I had Start and Stop the docker.

After stopping it, I had to remove the Docker image first followed by docker container.

Resolution: Running Docker in the Interactive mode (-it -d) Helped me. Finally, I could see the output as “Up”.

3) While Working with Ubuntu inside the Virtual Machine the Storage must be well allocated to the machine.

**TAKE AWAY:**

1)While Working with the Given Assignment, I gained insights into Docker so Deeply and get intrigued by the advance feature of “Docker”.

Important Commands:

1. Check the Container Status

docker ps -a

2)Get the console output of Docker

Docker logs $container id

3)docker image ls

Display the list of images available

**REFERENCES**

[1] “Oracle VM VirtualBox.” https://www.virtualbox.org/ (accessed Jul. 13, 2020).

[2] “The leading operating system for PCs, IoT devices, servers and the cloud,” *Ubuntu*. https://ubuntu.com/ (accessed Jul. 13, 2020).

[3] “How to Install Docker On Ubuntu 18.04 {2020 Tutorial},” *Knowledge Base by phoenixNAP*, Oct. 22, 2018. https://phoenixnap.com/kb/how-to-install-docker-on-ubuntu-18-04 (accessed Jul. 13, 2020).

[4] “How to fix docker: Got permission denied while trying to connect to the Docker daemon socket,” *DigitalOcean*. https://www.digitalocean.com/community/questions/how-to-fix-docker-got-permission-denied-while-trying-to-connect-to-the-docker-daemon-socket (accessed Jul. 13, 2020).