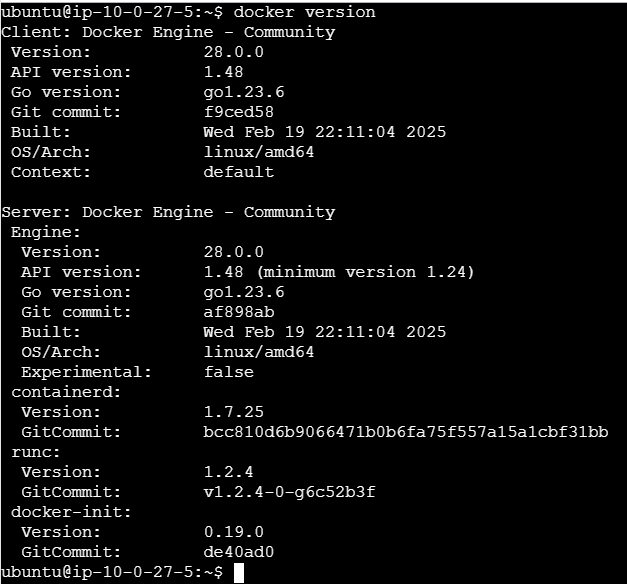
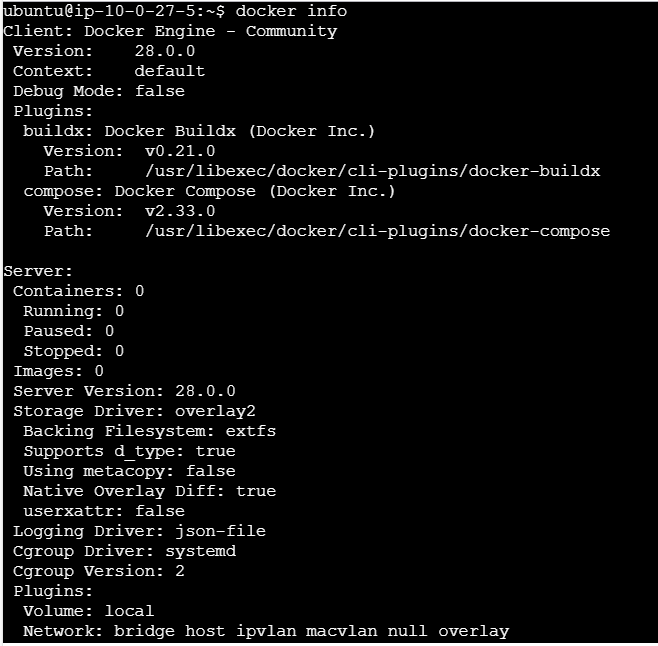
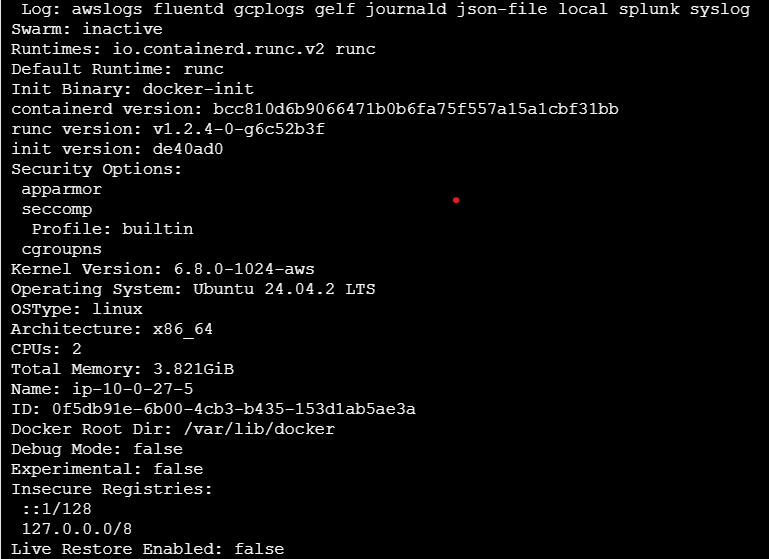
**Docker Lab questions hands-on practice**

1. **Verify Docker Version**

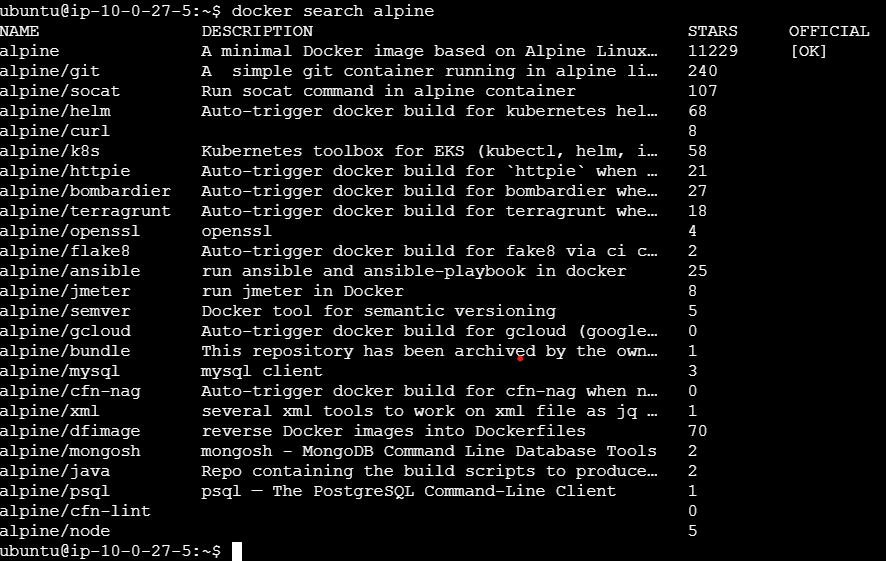


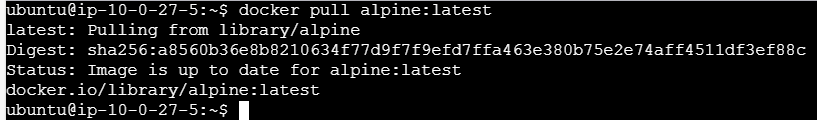
1. View system wide Information

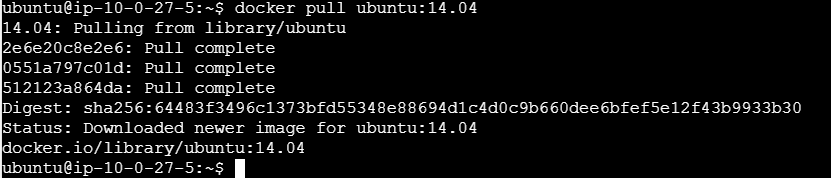


b.   


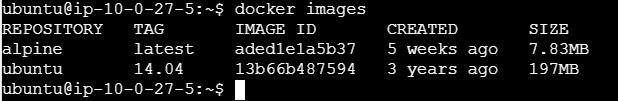
1. Search Docker Images in its Registry Server – alpine



1. Download the latest version of Image alpine  
   
2. Download Ubuntu:14.04 Docker Image

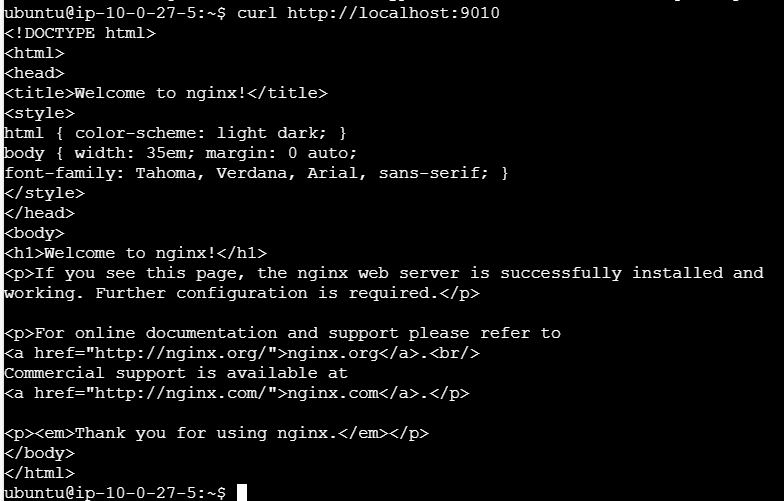


1. List all Downloaded Docker Container Images



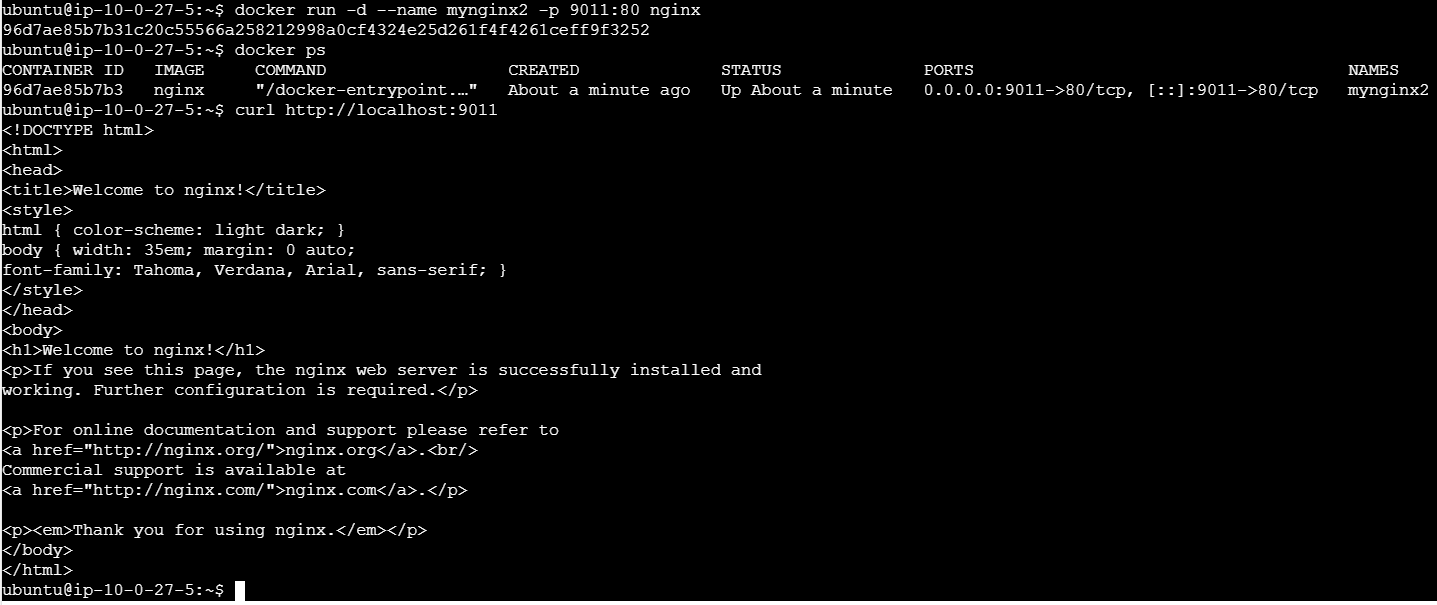
1. Spinup a docker container for nginx and name it as mynginx1 and

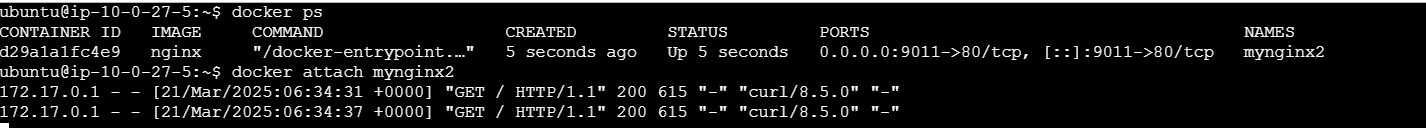
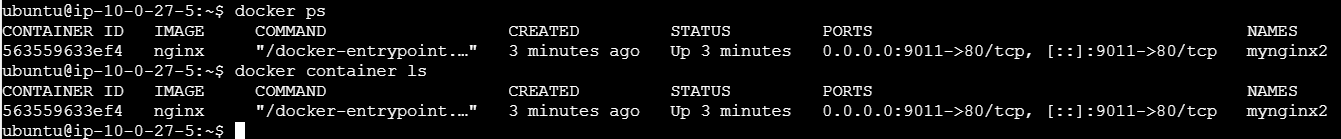
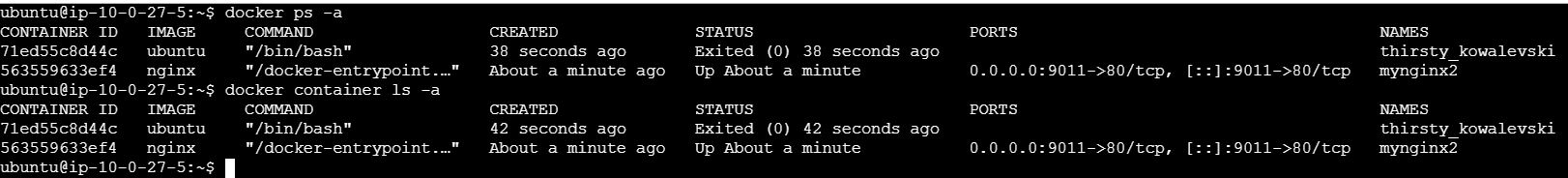
set a rule in such a way that if any request comes to 9010 port on docker host then that request will be redirected to mynginx1 container on 80 port.

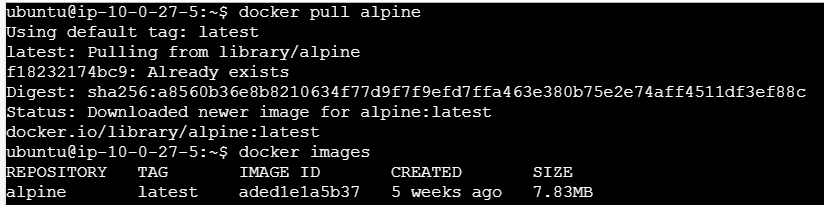
  
 b.   


1. Spinup a docker container in detached mode for nginx and name it as mynginx2 and

set a rule in such a way that if any request comes to 9011 port on docker host then that request will be redirected to mynginx2 container on 80 port and should

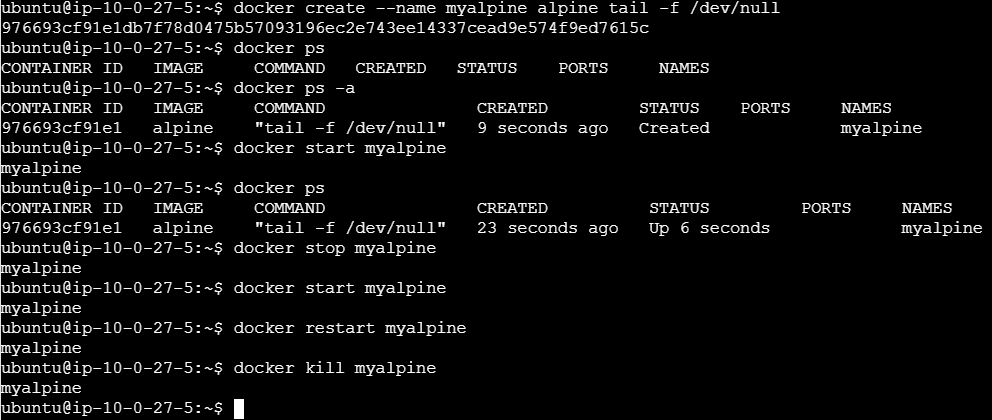


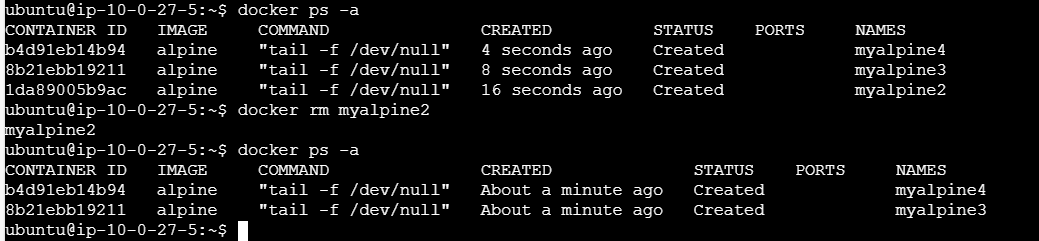
1. In the above example we have provisioned a container in detach mode, we can have its console using  docker attach  command  
   
2. List all the running Containers.  
   
3. List all the containers including stopped /shutdown containers  
   
4. pull image alpine and create container myalpine1 and do following actions:

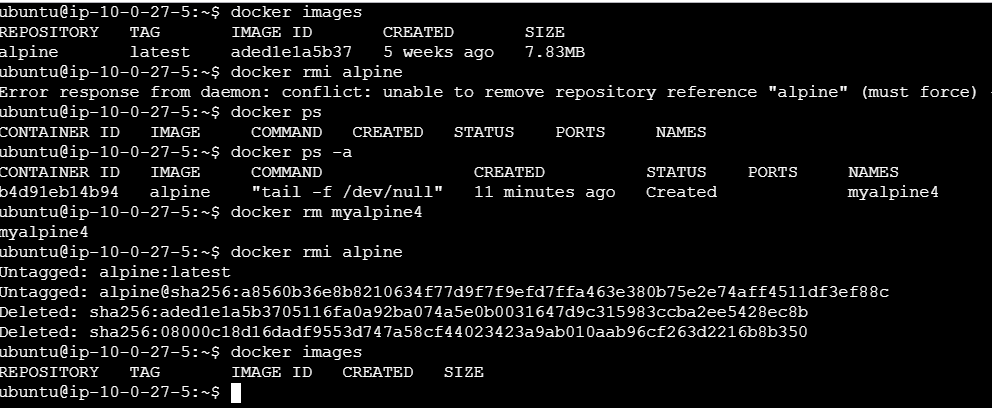
  
a) start the container

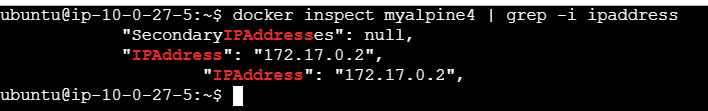
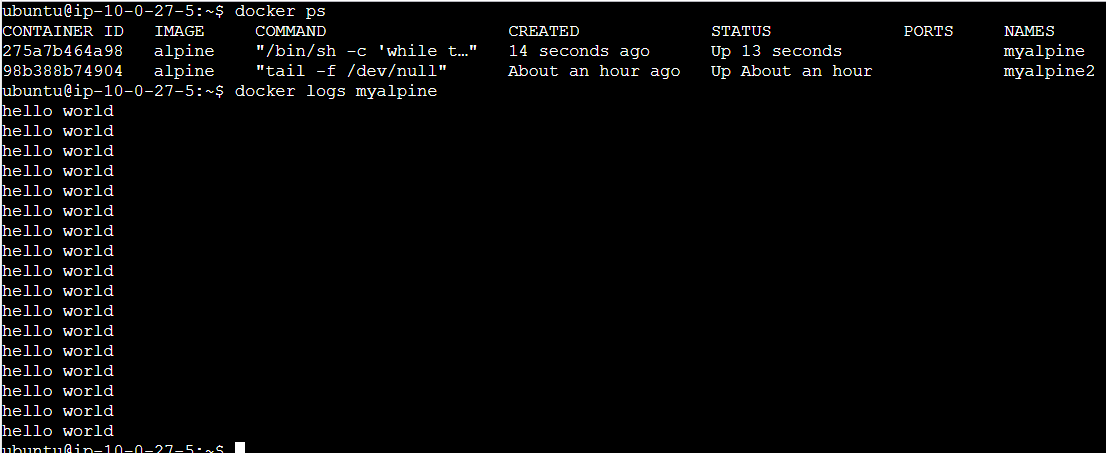
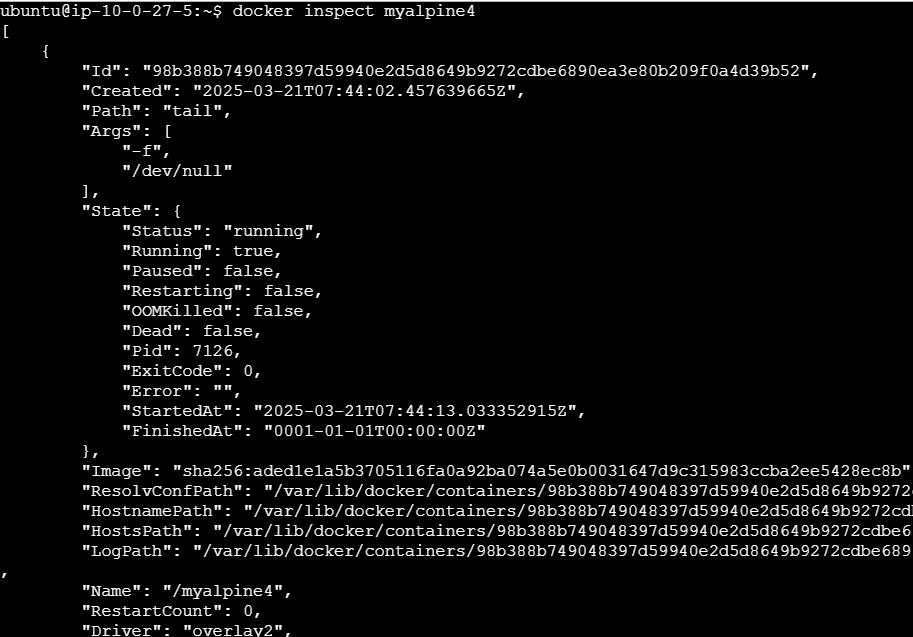
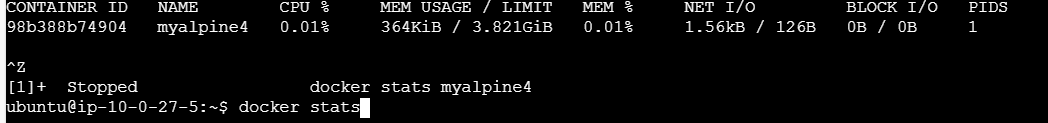
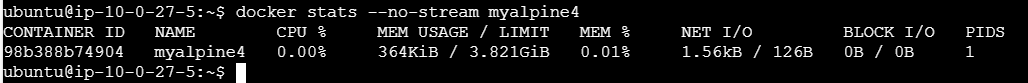
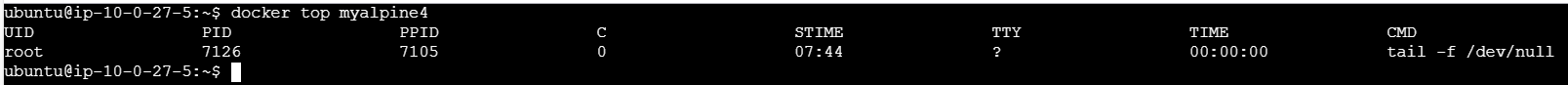
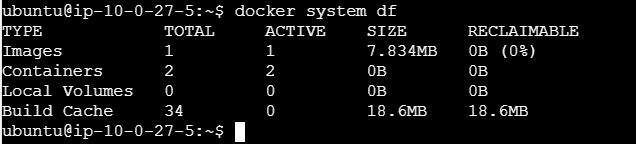
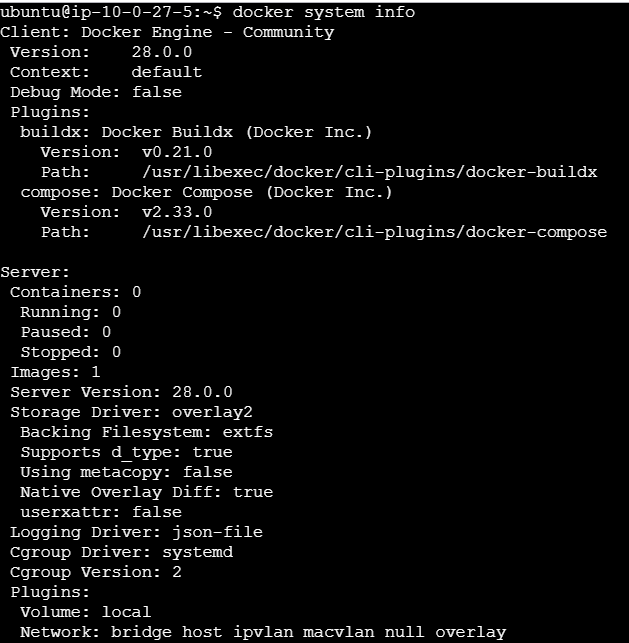
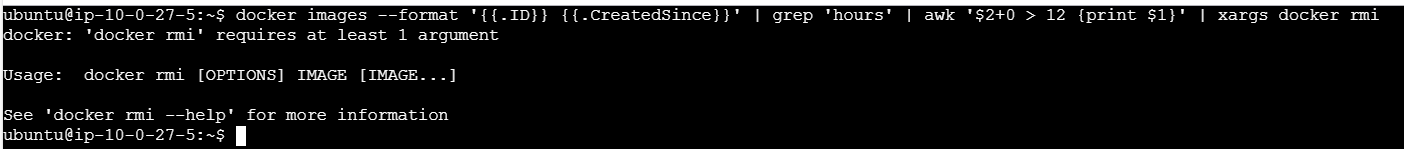
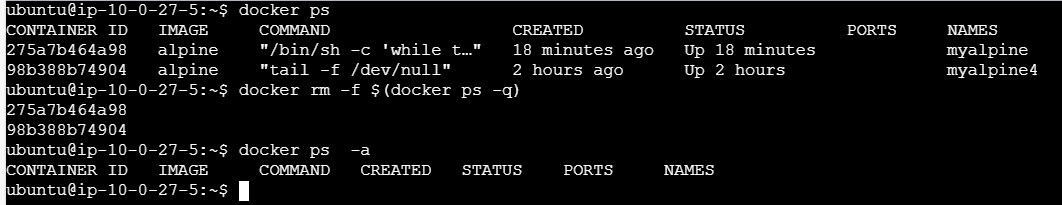
b) stop the container

c) restart the container

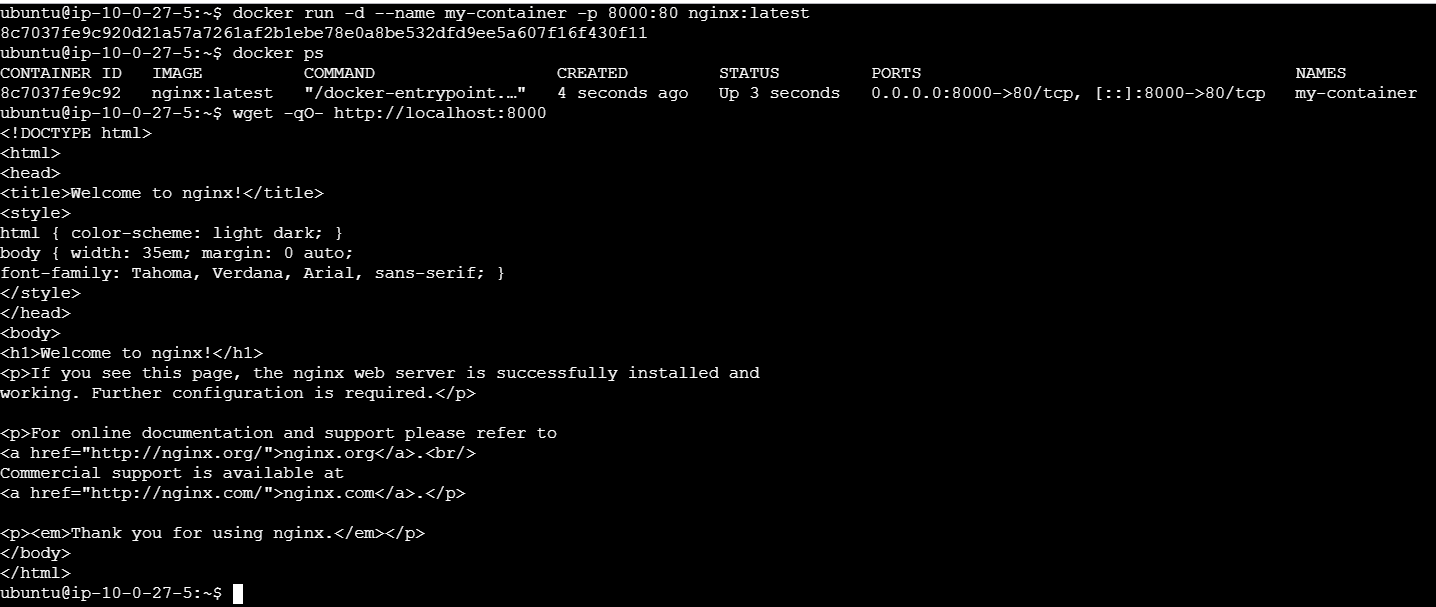
d) kill the container  
 

1. create containers from above alpine image called myalpine2/myalpine3/myalpine4  
   
2. Remove a container name myapline2  
   
3. Remove a running container forcefully named myalpine3  
   
4. Delete the image  Ubuntu:14.04  which you have earlier pulled/downloaded.

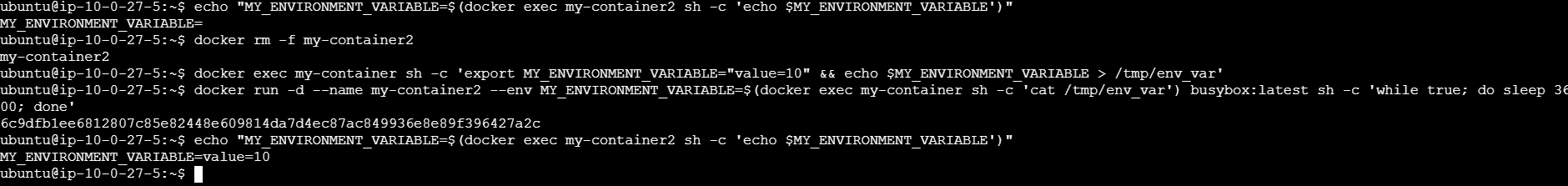


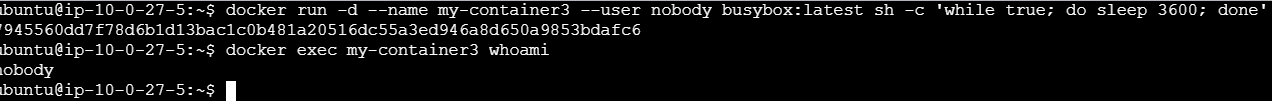
1. Display Container IP address for myalpine4  
   
2. Display the logs for container myalpine  
   
3. Inspect the container myalpine4  
   
4. Display the stats for myalpine4 (live streaming /non-live streaming)  
   a.  
     
   b.   
   
5. Display the running processes of a container with  docker top  command  
   
6. Write a command to find out the size of all the Docker objects in your system.  
   
7. Write a command to find out the details of all the Docker objects in your system  
   
8. Remove all images that are created more than 12 hours ago  
   
9. Stop all running containers with a single command.  
   
10. docker run -i alpine /bin/sh ---- here we are not getting terminal

docker run -it alpine /bin/sh ------ here we are getting terminal  


1. Run a new Docker container "my-container" from the "nginx" image (latest tag) in interactive background mode, without pseudo-TTY allocation. Binds port "80" of "my-container" container to port "8000" on localhost. The execution result of "wget -qO- [http://localhost:8000](http://localhost:8000/)" shows a valid nginx "Welcome Page".  
   
2. Run a new Docker container "my-container2" from the "busybox" image (latest tag) in interactive background mode,

without pseudo-TTY allocation, with an existing environment variable "MY\_ENVIRONMENT\_VARIABLE" to the "my-container" container.

print "MY\_ENVIRONMENT\_VARIABLE" value on host.  


1. Run a new Docker container "my-container3" from the "busybox" image (latest tag) in interactive background mode, without pseudo-TTY allocation and under the user "nobody". The result of  whoami must output "nobody".  
   
2. Create a mysql db (rdbms) container named mysqldb1 with a named volume mysqlvol

environment variables - MYSQL\_ROOT\_PASSWORD=H@ckath0n

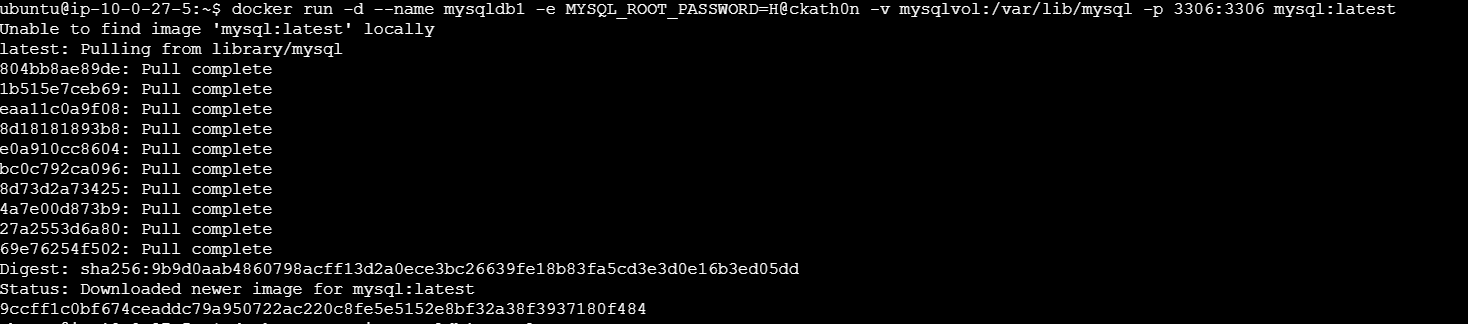
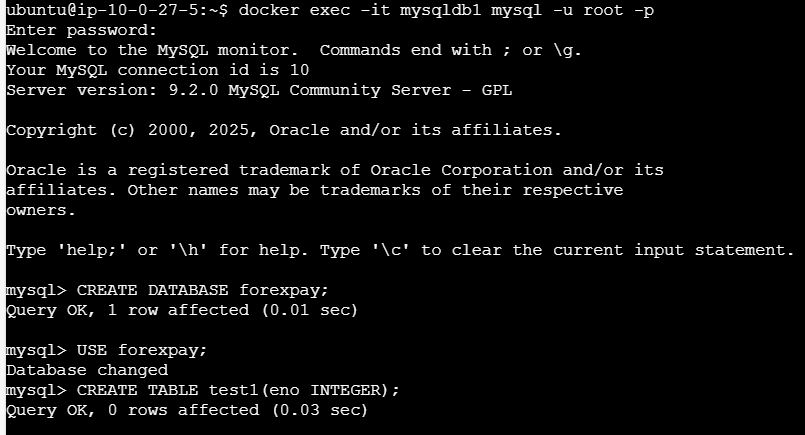
port publishing - 3306:3306

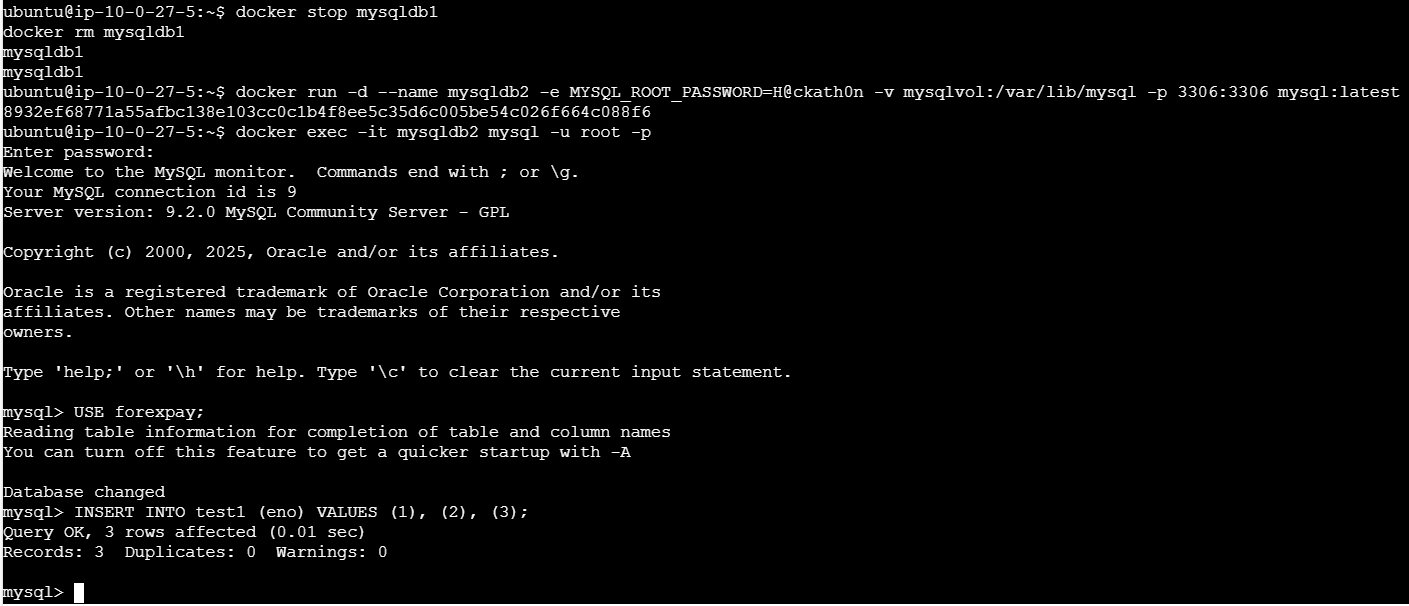
b) Get into the container and perform the below steps:

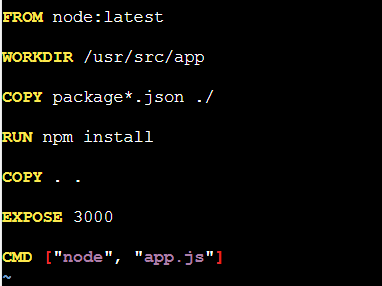
create database forexpay;

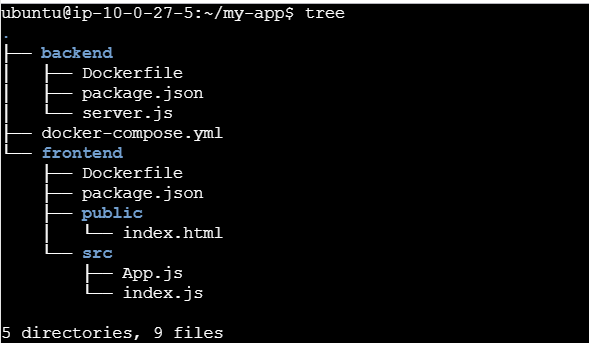
use forexpay; create table test1(eno integer);

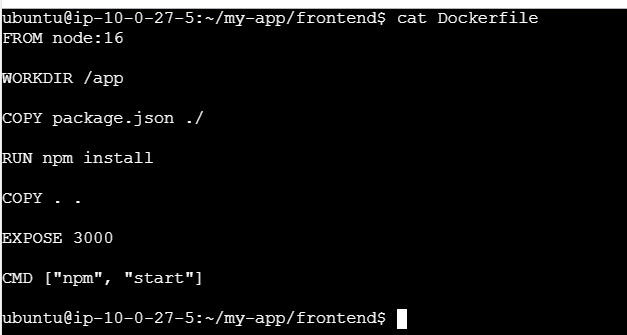
c) Delete the above container mysqldb1

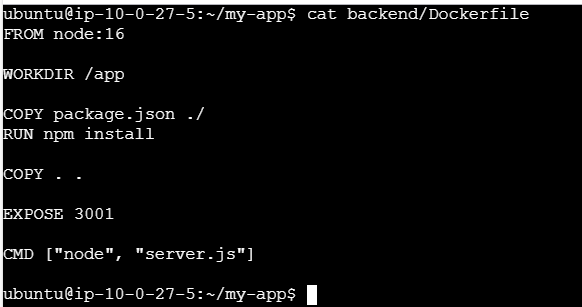
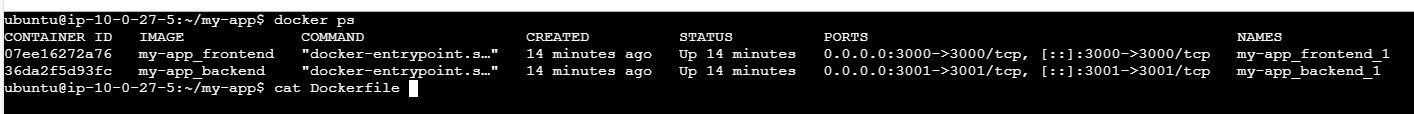
d) spin up a new container mysqldb2 and insert rows into test1 (forexpay database). Use the same mysqlvol  
a.  
  
b.  


c.   


1. Create a Dockerfile for a simple Node.js application. The application should:
2. Use the official Node.js image.
3. Set the working directory to /usr/src/app.
4. Copy the package.json and package-lock.json files to the working directory.
5. Install the application dependencies.
6. Copy the rest of the application code to the working directory.
7. Expose port 3000.
8. Define the command to run the application using node app.js.  
   
9. Create a docker-compose.yml file to set up a simple front-end and back-end application. The front-end should be a React application, and the back-end should be a Node.js application. The front-end should communicate with the back-end via HTTP.

a.  


b.  
 

c.  
   
 d.   
   
 e. 

d.  
