

**Docker**

**Exercise Manual**

This page intentionally left blank.

**Table of Contents**

Exercise 1.1: Checking Docker System 1

Exercise 2.1: docker run Part I 5

Exercise 2.2: docker run Part II 9

Exercise 4.1: Volumes 15

This page intentionally left blank.

# Exercise 1.1: Checking Docker System

**Purpose**: To make sure Docker is working correctly.

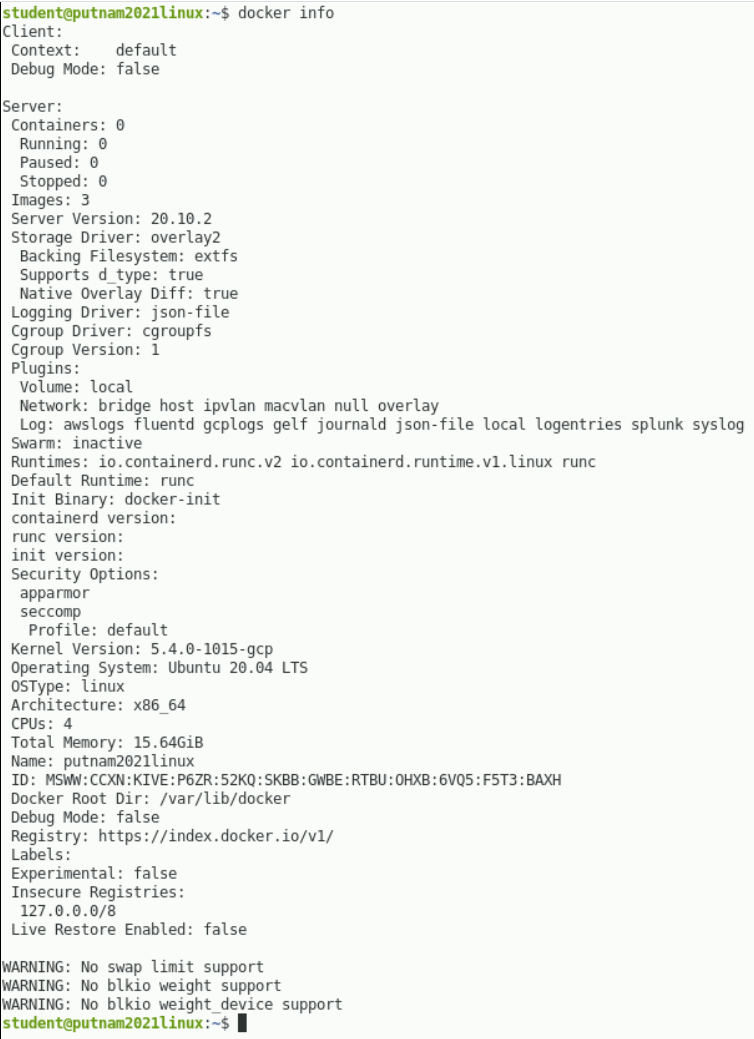
**Steps**

1. Execute: cd ~/ROI.Training/Docker
2. Open a terminal window.
3. Execute: docker version
   1. You should see something similar to:



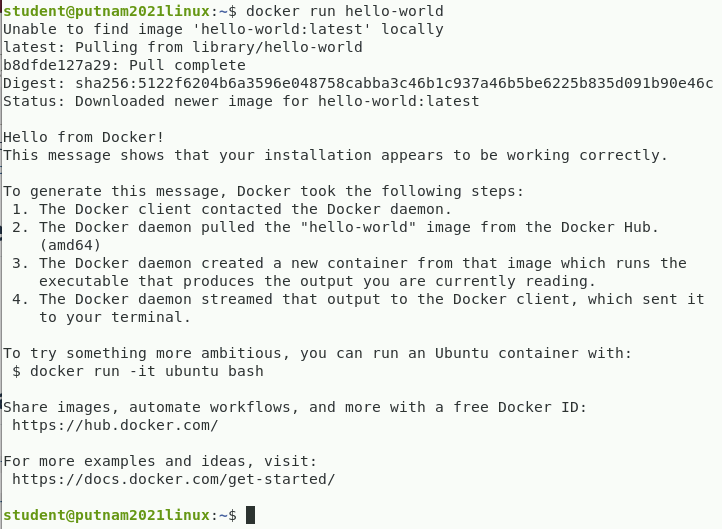
* 1. Docker is written in Go and C.

1. Execute: docker info
   1. You should see something similar to:



* 1. Notice the line Containers is followed by a number—the number of containers on the host. Containers stay around until deleted.
     1. To see all of the containers, use docker ps -a.
     2. To see only the active containers, use docker ps.
     3. To see the last created container, use docker ps -l.
  2. Notice the line Images is followed by a number—the number of images on the host. Images stay around until deleted.
     1. To see locally stored images, execute: docker images
  3. How much memory is available? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  4. How many CPUs? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Execute: docker run hello-world
   1. This executes the container hello-world.
   2. This is used by Docker to check the software was installed correctly.



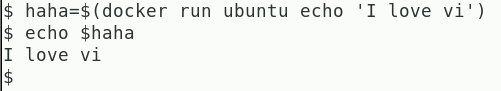
1. Make sure you read the “To generate this message…” on the screen.
2. There is no need to run the command docker run -it ubuntu /bin/bash.
   1. If you do, when done type in exit to stop the container. The container stops because the application stopped.

# Exercise 2.1: docker run Part I

1. Open a terminal window.
2. To see what images are available:
   1. Execute: docker images
   2. You should see something similar to:



1. When executing a single command, the STDOUT and STDERR of the command being executed is returned from the container as STDOUT.
   1. Execute: haha=$(docker run ubuntu echo 'I love vi')
   2. Execute: echo $haha
   3. You should see something similar to:

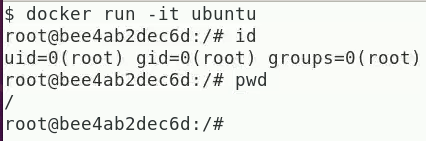


* 1. The STDOUT and STDERR is returned from the container.

1. When you build an image, you can specify a default command. To execute the default command, do not add a command to the end of the docker run command.
   1. The default command for the ubuntu image is /bin/bash. This is an interactive command. To work with this command, you need to tell the container that the command being executed is interactive, use -i or --interactive=true. You will need a tty terminal connected to the terminal. The option -t or --tty=true will attach a terminal to the container.
   2. Execute: docker run -it ubuntu
2. You should see something similar to:



1. Execute: id and pwd
   * 1. You should see something similar to:

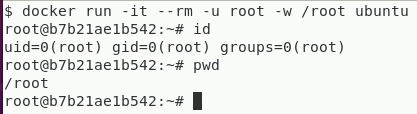


* + 1. The user id (uid) is root and the initial working directory is /.

1. Execute: exit
2. This will log you out of this container.
3. The container does not get deleted. We will see how to restart the container shortly.
4. Execute: docker run -it --rm -u root -w /root ubuntu
   1. You should see something similar to:



* 1. Execute: id and pwd
     1. You should see something similar to:



* + 1. The user id (uid) is root and the initial working directory is /root.
  1. The –u <username> option sets the user for this container.
  2. The --rm option removes the container when it exits.
     1. It will not be seen with a docker ps -a command.
  3. The -w <path> sets the starting directory for the container
     1. Execute: hostname
     2. It is the same as the string of characters after the @ in the prompt. This is the short id for the container.
  4. In a second window, execute: docker ps -l
     1. You should see something similar to:



* + 1. The -l option shows the most recently created container.
    2. By default, docker ps only shows the containers that are running.
    3. To see all containers, use the -a option.
    4. Notice that the CONTAINER ID is the same as the hostname.
    5. To give the image in the container a different hostname, add the option -h <hostname> or --hostname=<hostname> to the docker run command.
    6. The last field of the docker ps command is the NAME field. This name is generated by the Docker daemon and can be used to identify the container instead of the CONTAINER ID.
    7. If your container has a special purpose, it is a good idea to name the container with a name telling you what that purpose is.
    8. Create your own name by using the --name <name> option with the docker run command.
  1. In the container (window 1), execute: exit
     1. The container will be stopped and then will be removed because the   
        --rm option was used with the run command.

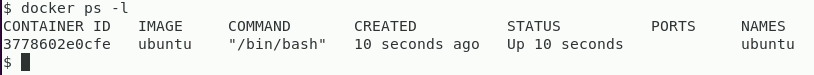
1. In window 1, execute:   
   docker run -it --rm -e envVar="This is a shell variable" ubuntu
   1. In the container, execute: echo $envVar
      1. The return should be: This is a shell variable.
   2. In then container, execute: env | grep -i envVar
      1. You should see: envVar=This is a shell variable
   3. In the container, execute: exit

# Exercise 2.2: docker run Part II

1. Open two windows, one on top of each other
   1. In the top window, execute:

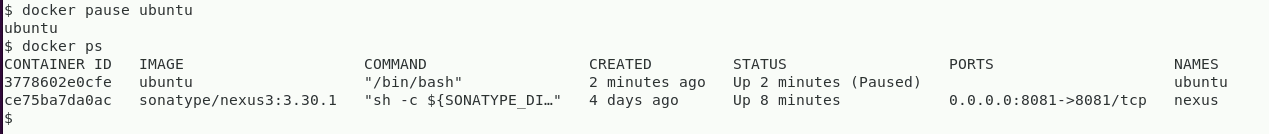
docker run -it --name ubuntu ubuntu

* 1. In the bottom window, execute: docker ps –l
     1. The bottom window should look similar to:



* + 1. Note that the image and container have been given the same name: ubuntu.

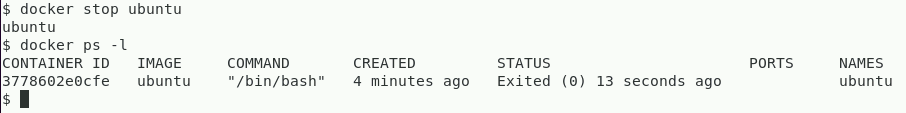
1. The docker pause command freezes the container by taking it out of the run queue. The docker unpause command causes the container to continue running from where it left off and the container is put back into the run queue.
   1. In the bottom window, execute: docker pause ubuntu
      1. This will pause the ubuntu container.
2. In the bottom window, execute: docker ps
   * 1. You should see something like:



1. In the top window, try to execute some Linux commands, e.g., ls, who, pwd.

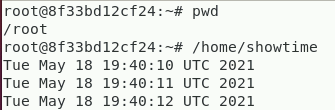
You will not see any output. Just type in the commands, one per line.

1. In the bottom window, execute: docker unpause ubuntu
2. In the top window, you should see the output from the Linux commands you typed in the previous step (2.c).
3. The docker stop <name> command is used to stop a running container.
   1. It causes the Docker daemon to first send a SIGTERM signal to the container. After a small grace period, the Docker daemon will send a SIGKILL signal. The SIGTERM signal is caught by the container and gives it the opportunity to gracefully shut down. The SIGKILL signal forces the container to shut down immediately. Only stopped containers can be removed.
   2. In the bottom window, execute: docker stop ubuntu
      1. This should terminate the container named ubuntu.
   3. In the bottom window, execute: docker ps -l
      1. You should see the status of work container is “Exited”.



* + 1. If you didn’t, it means that this container is not the last one created. Use docker ps -a to find the data.

1. Using the docker start command, it is possible to restart a container that has been stopped.
   1. In the top window, execute: docker start -ia ubuntu
   2. In bottom window, execute: docker cp showtime ubuntu:/home
   3. In the top window, execute: /home/showtime



* 1. In the bottom window, execute: docker kill -s "SIGINT" ubuntu
     1. What happened? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     2. You should see the command continue.

1. You can see simple usage statistics with the docker stats command.
   1. In the bottom window, execute: docker stats ubuntu
      1. Notice that this is in real time.



* + 1. The docker stats command does not return a summary or any statistics on stopped containers.
  1. In the top window, enter <Ctrl-C>. This sends a SIGINT to the foreground process, showtime, which terminates the process and returns to the command line.
  2. In the top window, enter exit. This will shut down the container because the /bin/bash program that the container is running will terminate.
     1. Notice that the bottom window is still running the docker stats ubuntu command. Nothing is being returned as the container is not running. The docker stats command only returns statistics for running containers.
  3. Enter <Ctrl-C> in the bottom window to cancel out the docker stats command.

1. In the top window, execute: docker start ubuntu
   1. The container will start in the background since it was not started in interactive mode (-ia).
   2. In the top window, execute: docker attach ubuntu
      1. Execute: ls /home
         1. You should see the listing of the /home directory.
         2. The docker attach command causes the container to run in the foreground.
         3. The process that the container is running, /bin/bash, displays its interactive prompt.
2. In the bottom window, execute: docker top ubuntu
   1. The docker top command displays the running processes of the work container in a format that is similar to the Linux ps -ef command.



* 1. What command is the ubuntu container running?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. The docker inspect command returns details about a container or an image formatted as a JSON document.
   1. In the bottom window, execute: docker inspect ubuntu | less
   2. Answer the following questions:
      1. What is the PID of the container? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
      2. What is the IPAddress of the container? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
      3. What is the LogPath of the container? (Give just the starting directory.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. What is the username?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. What is the first path in the PATH environmental variable?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* + 1. What is the WorkingDir?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. In the bottom window, execute:

docker inspect --format "{{.LogPath}}" ubuntu

* + 1. You should see the path to the containers log file returned.
  1. In the bottom window, execute:

docker inspect --format \

"{{.NetworkSettings.IPAddress}}" ubuntu

1. In the bottom window, execute:

sudo less $(docker inspect --format "{{.LogPath}}" ubuntu)

* 1. The log file records everything that happens in the container. Unfortunately, it is a rather raw dump and requires a lot of processing to be useful for anything except last ditch debugging.
  2. Enter <Ctrl-D>.
  3. The sudo is needed to access the file returned from the Docker command.

1. In the top window, in the container, enter <Ctrl-D>; this will terminate the /bin/bash command running in the container.
   1. In the bottom window, enter docker ps -a.
      1. You should see that the container ubuntu has been exited.
   2. Execute: docker rm ubuntu
      1. This will remove the container ubuntu.
      2. Execute: docker ps -a
2. The container ubuntu should not be found.

This page intentionally left blank.

# Exercise 4.1: Volumes

**A Docker Created Host Directory as a Container Volume**

1. Open two terminal windows, one on top of each other.
2. In the top window, execute: sudo docker run -i -t --rm ubuntu
3. ls /
   * 1. You should see something like:



1. exit
   * 1. This will terminate the container.
2. In the top window, execute:

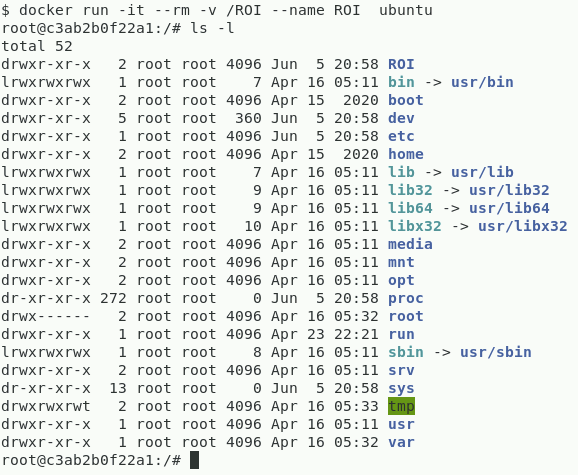
docker run --rm -it -v /ROI --name ROI ubuntu

1. -v is followed by the mount point for a volume.
2. The -v option can be repeated as required.
3. The Docker daemon creates a directory in /var/lib/docker/volumes with a very long unique identifier for each volume added.
4. In the bottom window, execute:

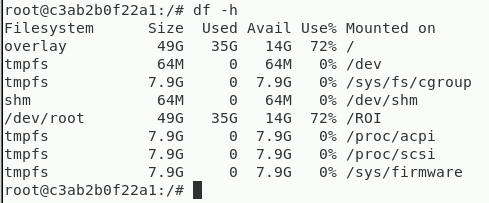
sudo ls -l/var/lib/docker/volumes

* 1. Note the last six digits of the files with very long names:
     1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
     2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. In the top window at the container prompt, execute: ls -l/
2. You should see something similar to:



1. Notice the new /ROI directory.
2. In the top window at the container prompt, execute: df -h
3. See something similar to:



1. Note that the size used, and availability characteristics of the / directory are the same as /ROI.
   * 1. A device driver named “none” represents a file system that cannot be manipulated as a regular file system.
     2. /dev/root is the file system on the host that points to / and /ROI in the container.
2. In the bottom window, execute: docker inspect ROI | less
3. Remember that ROI is the name of the running container.
4. Search for “Mounts”: /Mounts<cr>.
5. You should see something similar to:



1. The "RW": true entry means the volume was mounted for reading and writing.
2. To have a volume be read-only, add :ro to the end of the directory specification; for example, -v /ROI:ro, would be mounted read-only and "RW" would be false in the listing produced by docker inspect.
3. Notice the Destination and Source for the volume.
4. Exit from command.
5. Exit from Container ROI.