Assignment Solution

NOTE: Do not forget to see manual page using "--help" option in command when searching for options/commands for a particular task.

1. Install Docker, either on your native OS or on a VM. Make sure it runs. Type "docker -v" to check if it's installed.

If you can't install or configure Docker, you can use the online docker setup to do the assignment.

Step1 Goto:- https://www.katacoda.com/courses/kubernetes/playground

Step2 Click on "continue" button on the left panel

Step3 Click on "launch.sh" button on the left panel

Step4 From the right panel use the top console to execute below command:-

```
docker -v
```

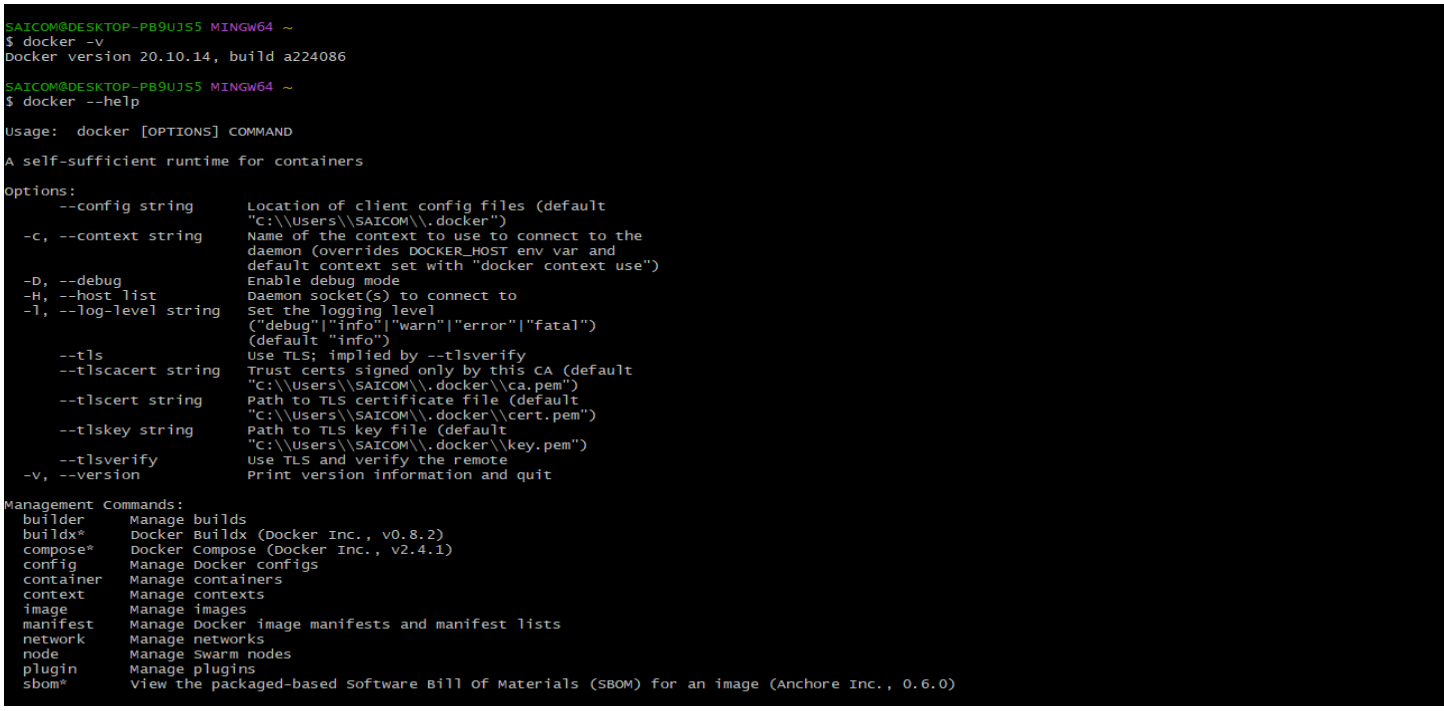
Try below commands for help

docker --help ---> This command shows all available options and commands to work with images and containers

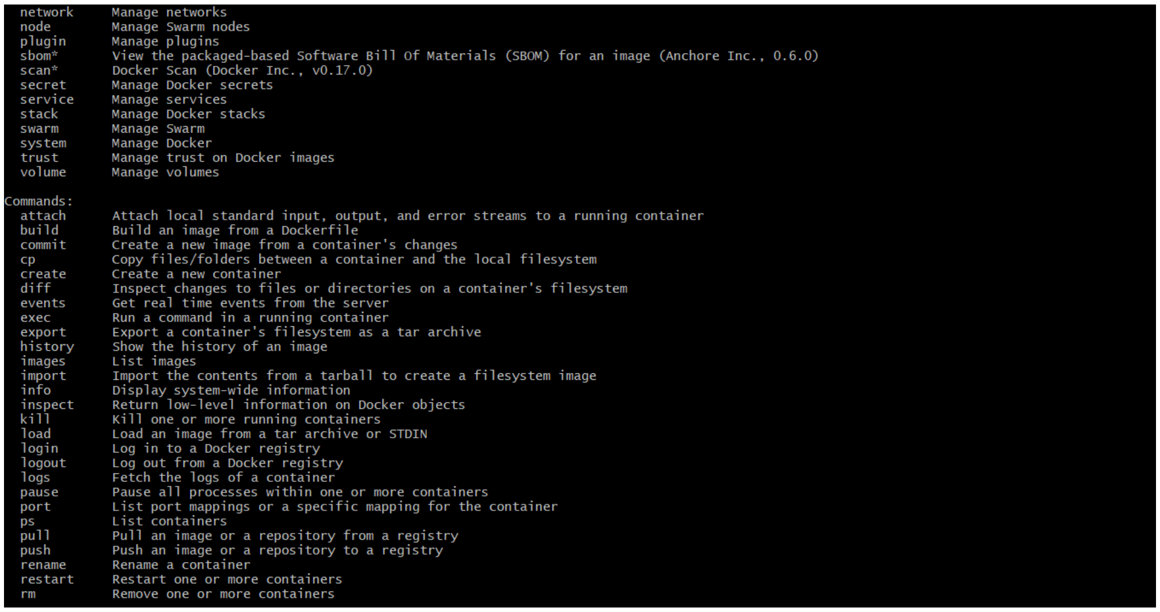
docker image --help ---> This command shows all the avaialble options and commands to work with docker images

docker container --help ---> This command shows all the avaialble options and commands to work with docker containers

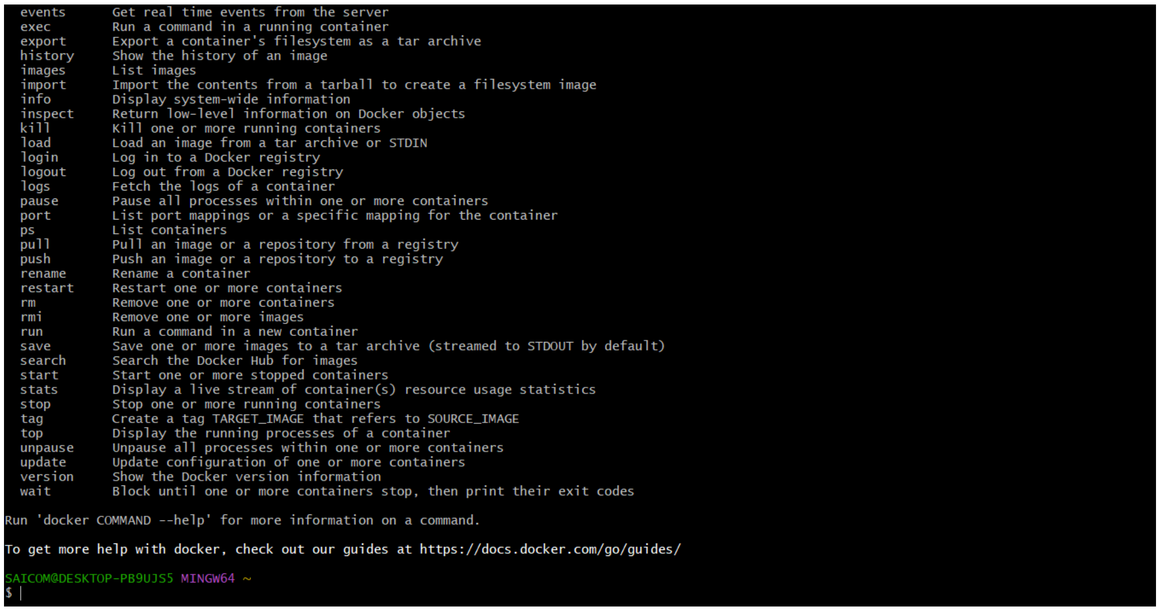
NOTE:- DO NOT TRY TO USE INTERNET TO SOLVE ASSIGNMENT, BETTER USE THE ABOVE --help OPTION TO SEE THE MANUAL OF ANY PARTICULAR COMMAND AND FIGURE OUT THE SOLUTIONS ON YOUR OWN.



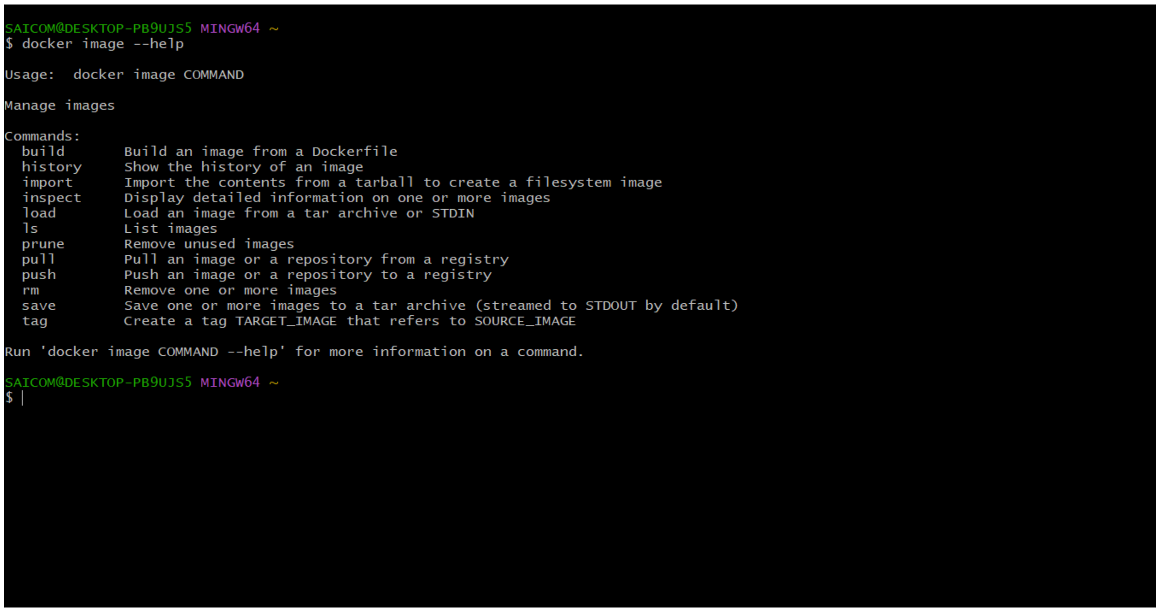
```
SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ docker -v
Docker version 20.10.14, build a224086

SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ docker --help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
      --config string      Location of client config files (default
                           "C:\\Users\\SAICOM\\.docker")
  -c, --context string     Name of the context to use to connect to the
                           daemon (overrides DOCKER_HOST env var and
                           default context set with "docker context use")
  -D, --debug              Enable debug mode
  -H, --host list          Daemon socket(s) to connect to
  -l, --log-level string   Set the logging level
                           ("debug"|"info"|"warn"|"error"|"fatal")
                           (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default
                           "C:\\Users\\SAICOM\\.docker\\ca.pem")
      --tlscert string     Path to TLS certificate file (default
                           "C:\\Users\\SAICOM\\.docker\\cert.pem")
      --tlskey string      Path to TLS key file (default
                           "C:\\Users\\SAICOM\\.docker\\key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.8.2)
  compose*    Docker Compose (Docker Inc., v2.4.1)
  config      Manage Docker configs
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
```



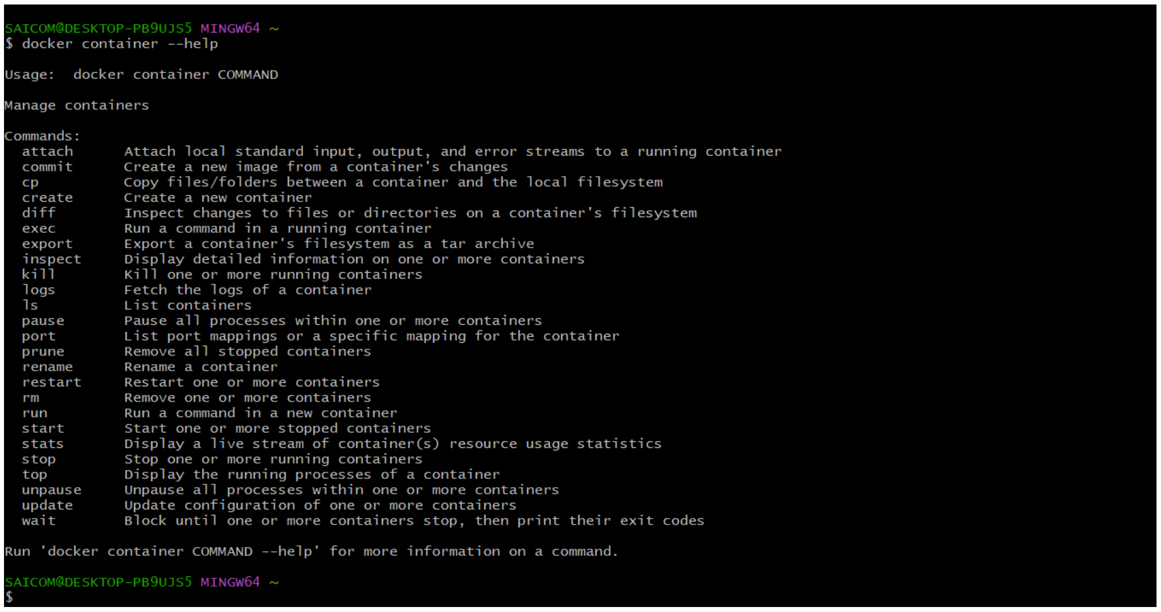
```
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
  scan*       Docker Scan (Docker Inc., v0.17.0)
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
```



```
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker COMMAND --help' for more information on a command.

To get more help with docker, check out our guides at https://docs.docker.com/go/guides/

SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ |
```

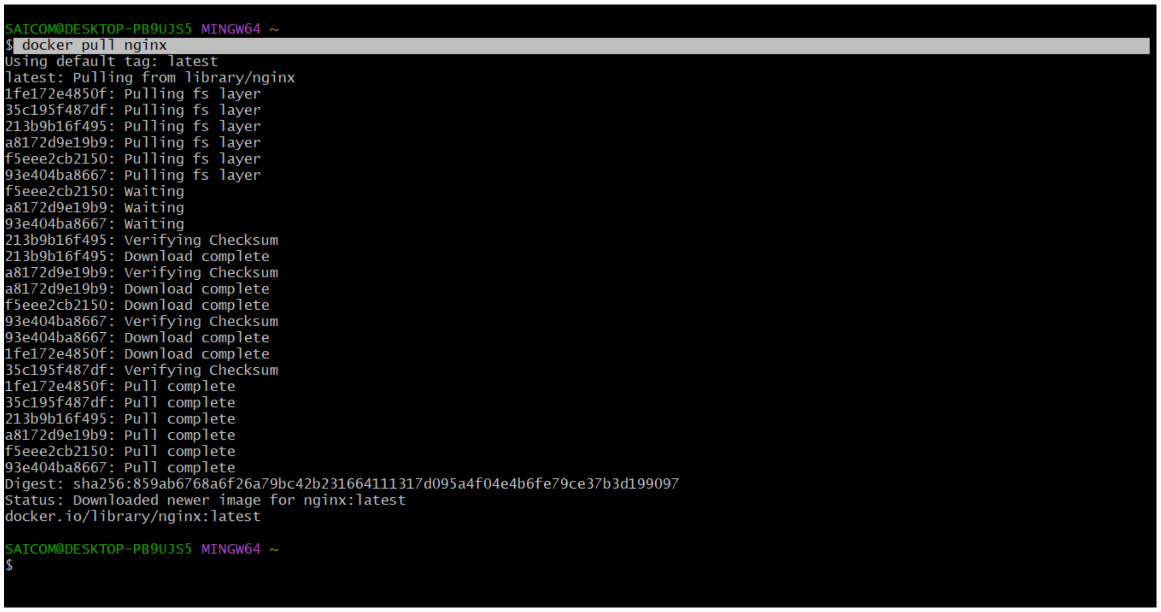


```
SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ docker image --help

Usage:  docker image COMMAND

Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.

SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ |
```


```
SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ docker container --help

Usage:  docker container COMMAND

Manage containers

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  inspect     Display detailed information on one or more containers
  kill        Kill one or more running containers
  logs        Fetch the logs of a container
  ls          List containers
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  prune       Remove all stopped containers
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  run         Run a command in a new container
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  wait        Block until one or more containers stop, then print their exit codes

Run 'docker container COMMAND --help' for more information on a command.

SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$
```

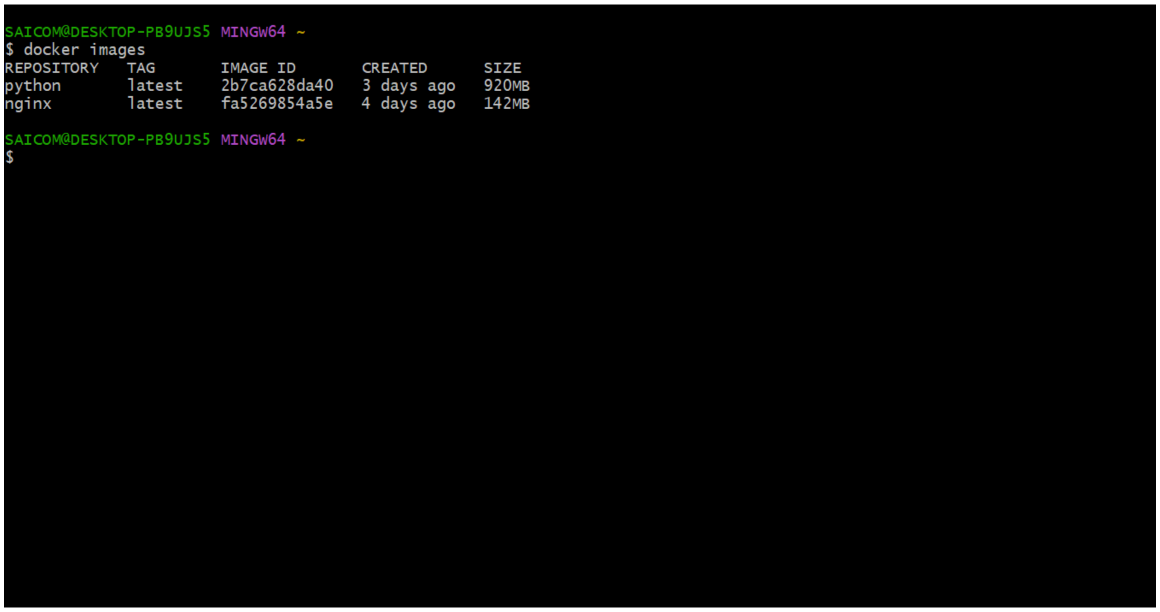
2. Find a image from dockerhub of your choice(recommeded: nginx), don't use browser, pull the official image from dockerhub

**docker pull nginx**


```
SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
1fe172e4850f: Pulling fs layer
35c195f487df: Pulling fs layer
213b9b16f495: Pulling fs layer
a8172d9e19b9: Pulling fs layer
f5eee2cb2150: Pulling fs layer
93e404ba8667: Pulling fs layer
f5eee2cb2150: Waiting
a8172d9e19b9: Waiting
93e404ba8667: Waiting
213b9b16f495: Verifying Checksum
213b9b16f495: Download complete
a8172d9e19b9: Verifying Checksum
a8172d9e19b9: Download complete
f5eee2cb2150: Download complete
93e404ba8667: Verifying Checksum
93e404ba8667: Download complete
1fe172e4850f: Download complete
35c195f487df: Verifying Checksum
1fe172e4850f: Pull complete
35c195f487df: Pull complete
213b9b16f495: Pull complete
a8172d9e19b9: Pull complete
f5eee2cb2150: Pull complete
93e404ba8667: Pull complete
Digest: sha256:859ab6768a6f26a79bc42b231664111317d095a4f04e4b6fe79ce37b3d199097
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$
```

3. List all the available images in your machine/vm, make sure you see recently pulled image in the list.

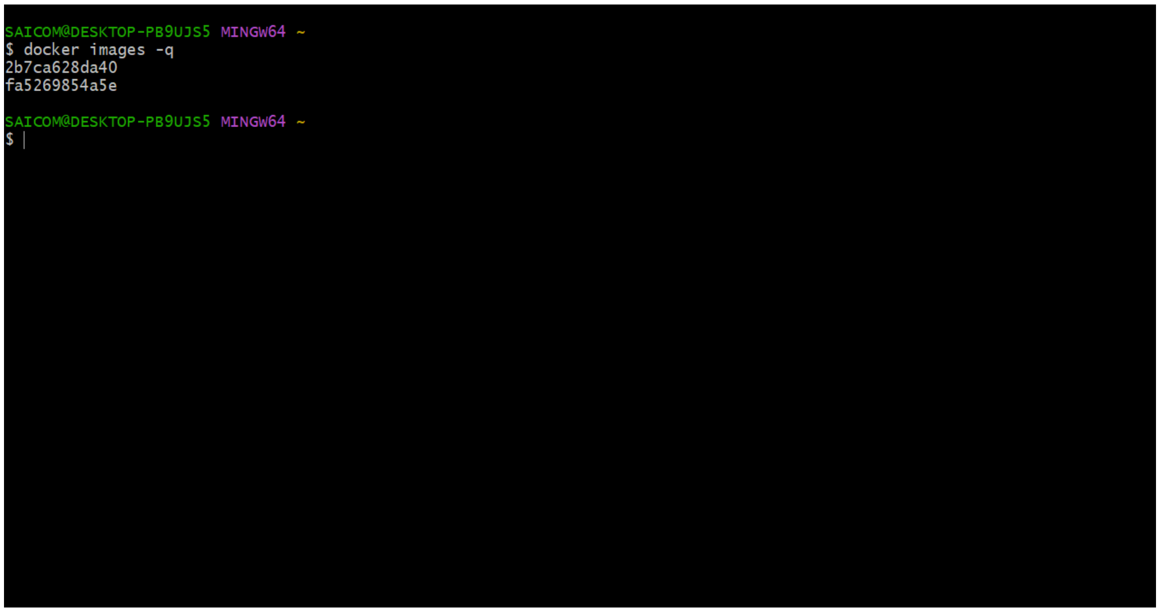
**docker images**



```
SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
python       latest    2b7ca628da40   3 days ago   920MB
nginx        latest    fa5269854a5e   4 days ago   142MB

SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$
```

4. Find out the "Full" ImageId of the image that you pulled and write it below.

**docker images -q**



```
SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ docker images -q
2b7ca628da40
fa5269854a5e

SAICOM@DESKTOP-PB9UJS5 MINGW64 ~
$ |
```

5. Create a container of your image

**docker container create hello-world**

6. List all the running containers

**docker ps**

7. List all the running and stopped containers

**docker ps -a**

8. Find out the "Full" containerId of the container and write it below.

**docker ps -aqf "name=containername"**

9. Find out how many image layers are used to build this image.

**docker image inspect nginx**

10. Get the Apache Tomcat 7 server image from the docker hub.

**docker pull tomcat**:**7**

11. Run the Apache Tomcat 7, I mean create a container of Apache Tomcat.

**docker run tomcat:7**

12. Find out what is the IP Address of the Apache Tomcat Container that it is running on

**docker inspect <containername or id>**

13. Which Port it is using?

**docker port tomcat**

14. Try to access the Tomcat's home page from your machine/vm.

**docker run --name tomcat -P bitnami/tomcat:latest**

15. What is the disk size of Apache Tomcat image?

**docker images**

16. Find out list of all environment variables that is configured for tomcat image, can you see JAVA_HOME and CATALINA_HOME? What did you notice about it?

**docker exec container_id printenv**

**Yes I can see the variables**

**CATALINA_HOME specifies the location of the root directory of the binary distribution of Tomcat**

**CATALINA_HOME=/usr/local/tomcat**

**JAVA_HOME used to specify the location of a Java Runtime Environment that is used to start the environment.**

**JAVA_HOME=/usr/local/openjdk-11**

17. Find out which port is exposed for tomcat?

**docker inspect tomcat**

18. Run multiple conntainers of tomcat on different port and access it's home page.

**TOMCAT_VERSION=10.0.20**

19. Pull ubuntu os from dockerhub, try to pull 2 images of ubuntu, Except the latest one.

**docker pull ubuntu:16.04**

**docker pull ubuntu:18.04**

20. Run the container of ubuntu in attached mode.

**docker run -it --name ubuntu-16 ubuntu:16.04**

**docker start ubuntu-16**

**docker attach ubuntu-16**

21. Run the container of another ubuntu in detached mode.

**docker run -d --name ubuntu2 ubuntu:18.04**

22. Check how many ubuntu containers are running and stopped

**docker ps -a**

23. Is the tomcat container running? If no, start one.

**Yes it is running**

24. Check the logs, generated by tomcat container(don't forget to make request to tomcat's home page to see the log).

**docker logs container_id_of_tomcat**

25. Check if ubuntu conatiner is running? If no, start one in attached mode to the terminal.

**docker attach ubuntu_container_id**

26. Login as root user in ubuntu container

**docker exec -it ubuntu2 bash**

27. Create a file with any name in root directory

**touch file1**

28. Install software of your choice in ubuntu container using "apt-get install"

**apt-get update**

**apt-get install python3**

29. Now exit the ubuntu shell, are you back to your host machine, if not, come back to the host machine.

**exit**

30. Check if the ubuntu container is running.

**NO**

**docker ps**

31. Create a new ubuntu container out of the same image as that previous container in attached mode.

**docker run -it --name ubuntu-new ubuntu:16.04**

**docker start ubuntu-new**

**docker attach ubuntu-new**

32. Login as a root user

**It is already in root user after running the above commands**

33. Check if you can see the file created in previous container, you will not see the file as well as software that you installed in the previous container. Now kill this Container.

**docker kill container-id**

34. Do you have the previous ubuntu container where you created the file and installed the software? If no reapeat step 25 to 29.

**yes**

35. Create an Image out of the existing container.

**docker commit ubuntu-new SAICOM/user_image**

36. Now Create a Container out of this image and login into it to see if you can see the file and software installed by you in the previous container.

**NO**

37. Do you have running tomcat container? If yes, Stop it and kill all tomcat container.

**docker kill tomcat_container_id**

38. Create an index.html file with following code in it:-

    <h1>This is Tomcat Container</h1>

Now, Start a tomcat container in such a way that on hitting its URL for home page it should show the above html page.

**nano index.html**

**write the above content in index.html exit the nano**

**Dockerfile should contain**

**ADD index.html    in /usr/local/tomcat/webapps/**

**docker run -it --name c1 -p 8080:8080 tomcat**

39. type below command:-

    docker images --help

Now, try to run command that proves the concept of following three options:-

1. -a

2. -f

3. -q

write atleast 1 command using each option above and prove their concepts as described in the --help.

**docker images -a --------- shows all images**

**docker images -f     ---------- filter the output based on condition**

**docker images -q     ----------Display only the image id**

40. type below command:-

    docker ps --help

Now, try to run command that proves the concept of following six options:-

1. -a
2. -f
3. -q
4. -n
5. -l
6. -s

**docker ps -a ----------shows all the running and stopped containers**

**docker ps -f ----------used to filter the result based on condition**

**docker ps -q---------used to show only container IDs**

**docker ps -n ----------used to show last n created containers(includes all states)**

**docker ps -l ----------shows the latest created containers**

**docker ps -s ----------Display total file sizes**