The below commands will let you work with images.

List images docker image ls

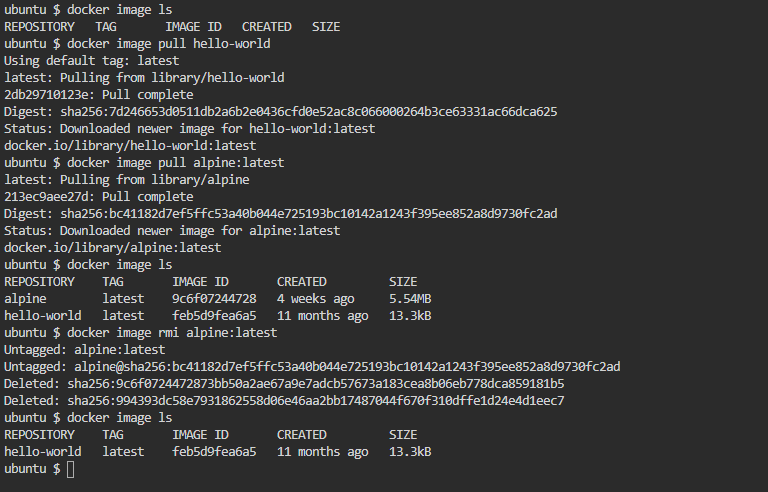
Pull hello-world image from the container image registry (in this case, Docker Hub) docker image pull hello-world

Pull the latest Alpine image from the container image registry docker image pull alpine:latest

List images on the local machine docker image ls

Remove images stored locally docker image rmi alpine:latest

List images again. Note that Alpine is no longer listed docker image ls



The below commands will work with images to create and list containers.

Take inventory of current images and containers

* docker image ls
* docker container ls

Run hello-world image docker container run hello-world Notice how the image did not need to be downloaded since we already downloaded it in step 2.

What if we delete hello-world image from the local machine? docker image rmi -f hello-world .

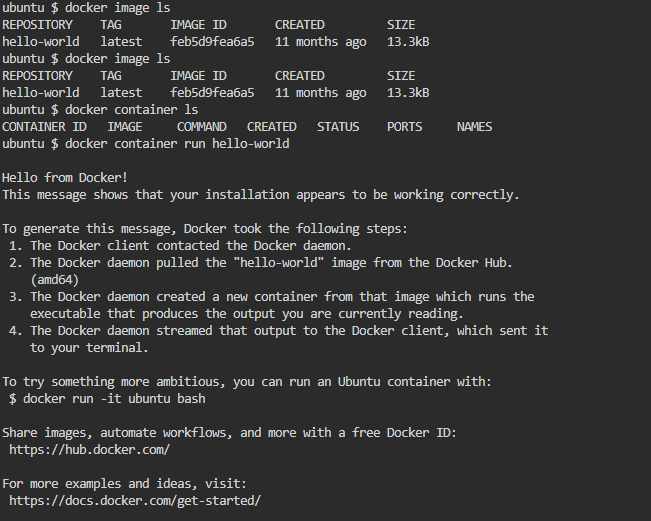
Look at the running containers docker container ls The hello-world container is not listed because it ran to completion.

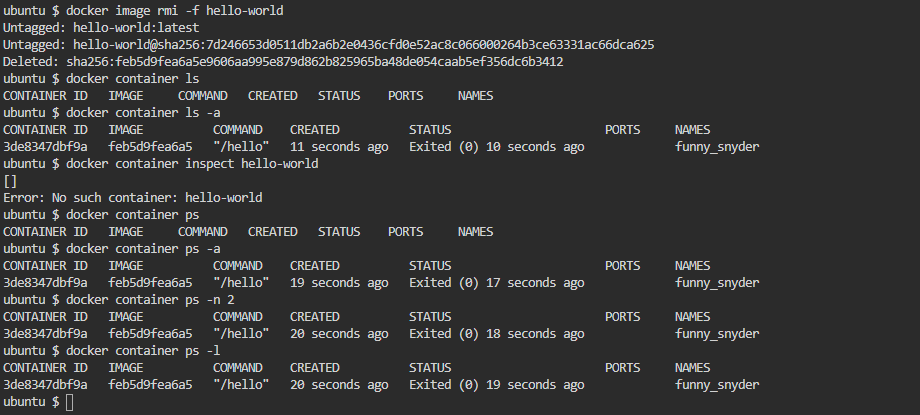
Look at all containers including stopped docker container ls -a Now we see hello-world listed as a stopped container and is named "hello-world" because we defined that container name at runtime.

Display all information about a container. In this case, let's inspect hello-world. docker container inspect hello-world This command displays extensive information about the container. Take some time to look it over.

Some other container commands:

* List running containers docker container ps
* List all containers docker container ps -a
* List most recent 2 containers docker container ps -n 2
* List the last container run docker container ps -l





We will now run containers in an interactive way. We will also explore immutability.

Create a container using the Ubuntu 16.04 image and connect to the terminal docker run -it ubuntu:16.04 bash This command runs the container, attaches to standard input stream, and get a pseudo-terminal. For the container process, we specify bash to get the terminal.

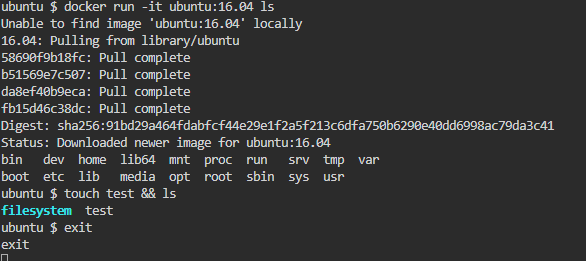
Create a file using the touch command: touch test && ls

You should see the file created in the root directory of the container. Now exit: exit

Run the container once again: docker run -it ubuntu:16.04 ls

Notice that the file is gone. This is because each time use the run command against an image, it creates a new container from the image. Any changes made to a container based on an image are not automatically saved back to the image.

This is know as immutability. This is on of the key principles of containers, they are immutable. Once built, it’s unchangeable, and if you want to make changes, you will get a new image as a result.

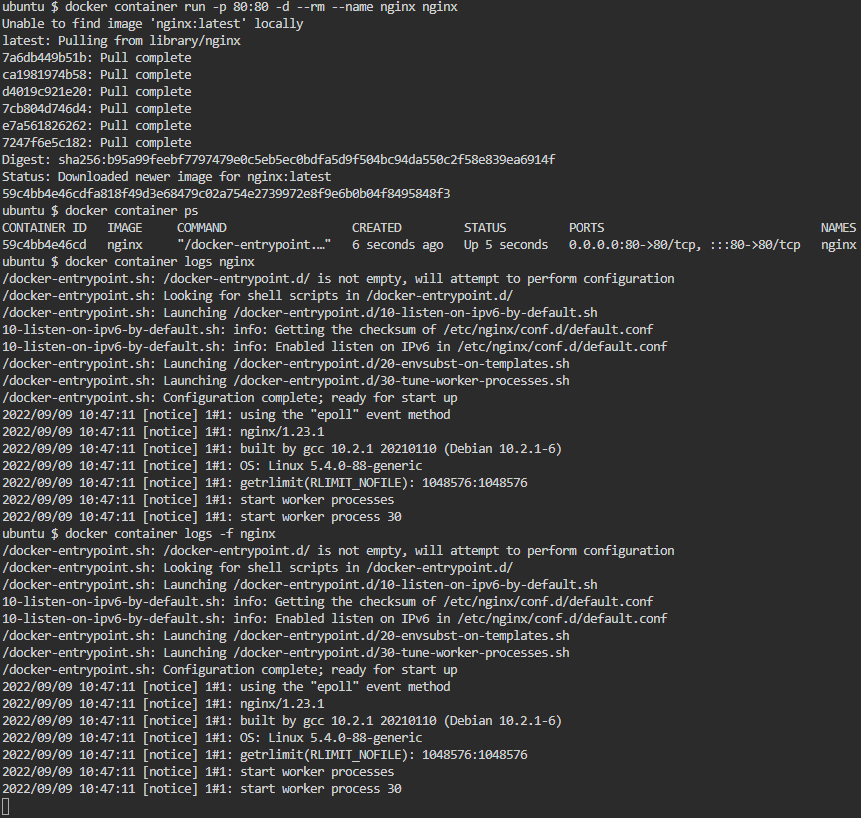


Now let's take a look at some containers logs. This is another good way to troubleshoot container issues.

Let's run NGINX container again and look at the logs:

* docker container run -p 80:80 -d --rm --name nginx nginx
* docker container ps
* docker container logs nginx
* docker container logs -f nginx

By using the -f switch, we are now live tailing the logs. Visit the NGINX page and you will see the logs update.



In this section we will work with container images.

Lets list all containers that exist on the local system:

docker image ls Now, pull some additional images from the Docker Hub:

docker image pull nginx

docker image pull busybox

docker image pull busybox:1.28

docker image pull ubuntu:18.04

Run the following command to see how much space the images are taking:

docker system df

docker image ls

docker image rmi busybox:latest

docker image rmi alpine:latest

docker image rmi ubuntu:18.04

docker image ls

Running the docker system df command shows us that we have freed up space by removing images.

Challenge

Delete all the running containers and containers in the environment.

