**Docker Solution Installation**

1. Installing a physical machine with Ubuntu 14 with Linux kernel 4.2.0-27-generic
2. Installing Docker Prerequisites on it with those steps:

* Update your apt 🡪 #sudo apt-get update
* ensure that APT works with the https method, and that CA certificates are installed

#sudo apt-**get** **install** apt-transport-https ca-certificates

* determines where APT will search for Docker packages

deb https://apt.dockerproject.org/repo ubuntu-trusty main

run the following command

#**echo** "<REPO>" | sudo tee /etc/apt/sources.**list**.d/docker.**list**

* update your apt 🡪 #sudo apt-get **update**
* Verify that APT is pulling from the right repository

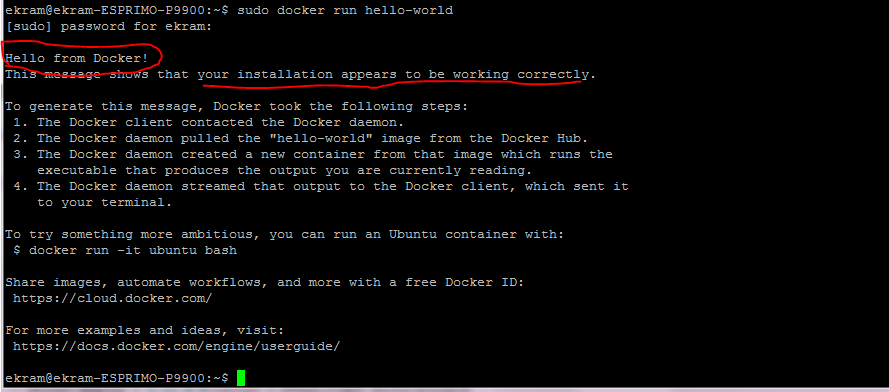
Running this command

#apt-cache policy docker-engine

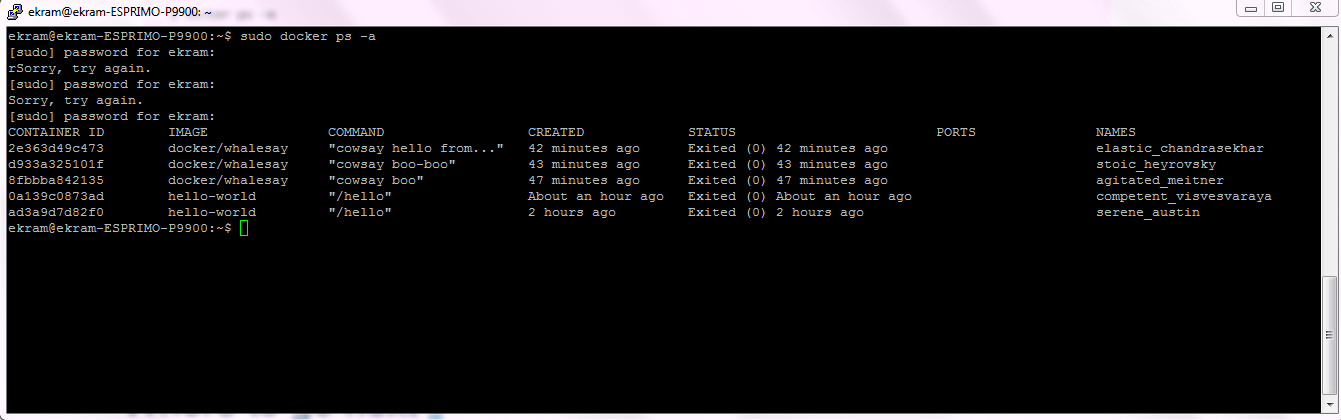
* install the recommended packages 🡪 #sudo apt-get **install** linux-image-extra-$(uname -r) linux-image-extra-**virtual**

1. installing DOCKER

* Update your APT package index 🡪 # sudo apt-get **update**
* Install Docker 🡪 # sudo apt-get install docker-engine
* Start the docker daemon 🡪 # sudo service docker start
* Verify that docker is installed correctly 🡪 # sudo docker run hello-world



* Run # docker ps -a to show all containers on the system.



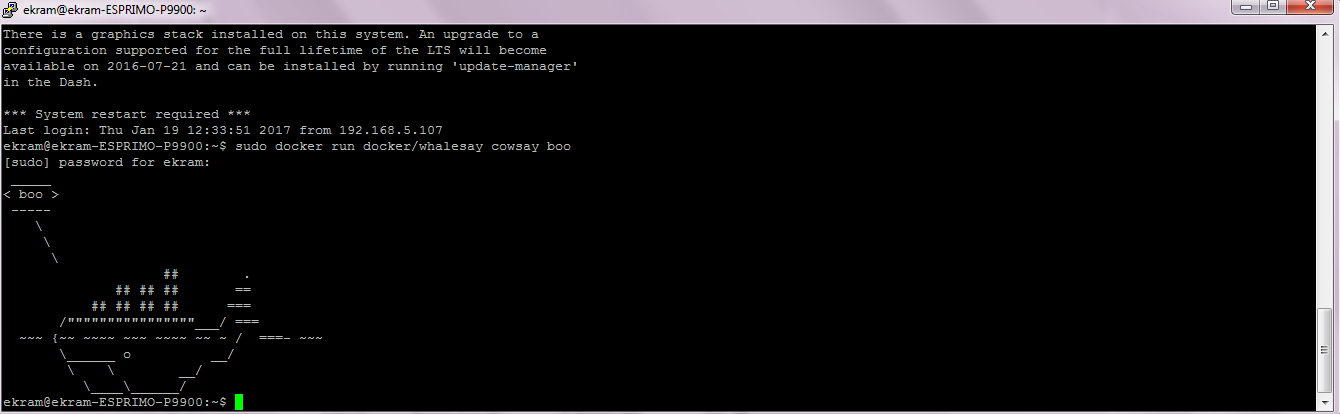
Running an image from Docker Hub

# **Find the whalesay image(found image on Docker hub)**

* Open your browser and browse to docker hub. The Docker Hub contains images from individuals like you and official images from organizations like RedHat, IBM, Google, and a whole lot more.
* Enter the word whalesay in the search bar.
* Click on the **docker/whalesay** image in the results.

1. Run the **whalesay image**

* In command line terminal type 🡪 # docker run docker/whalesay cowsay boo



**Build your own image**

1. Write a dockerfile (directory) 🡪 #mkdir myDockerTESTbuild
2. Change directory 🡪 #cd myDockerTESTbuild
3. Create and edit a new file name 🡪 #nano DockerFile
4. Write those lines

**FROM** docker/whalesay:latest

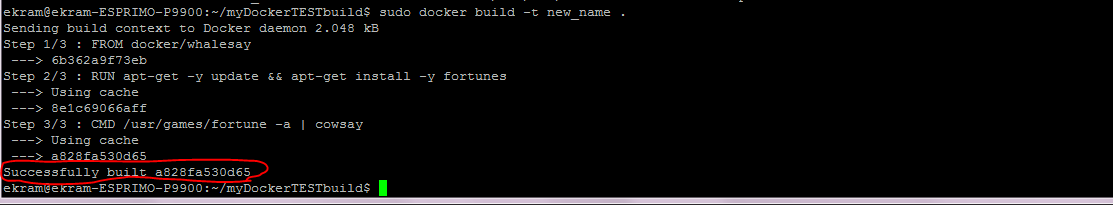
**RUN** apt-get -y update && apt-get install -y fortunes

**CMD** /usr/games/fortune -a | cowsay

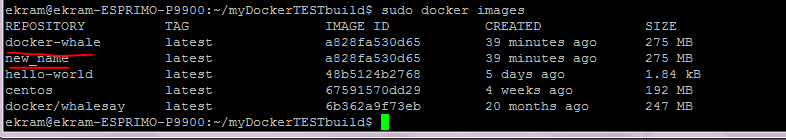
* FROM keyword tells Docker which image your image is based on. Whalesay is cute and has the cowsay program already
* RUN statement which will install the fortunes program into the image
* CMD statement, which tells the image the final command to run after its environment is set up

1. Save file and close the editor
2. #docker build -t “new\_image\_name” .

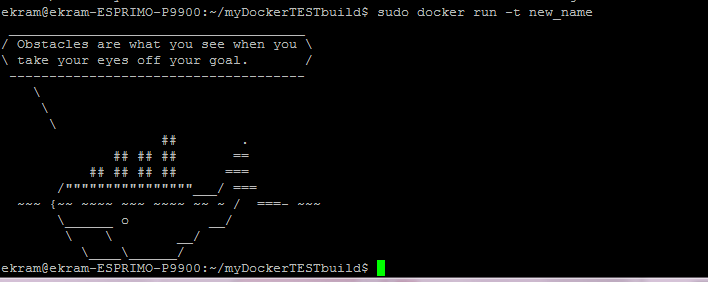
* build the image using the docker build command. The -t parameter gives your image a tag, so you can run it more easily later. Don’t forget the . command, which tells the docker build command to look in the current directory



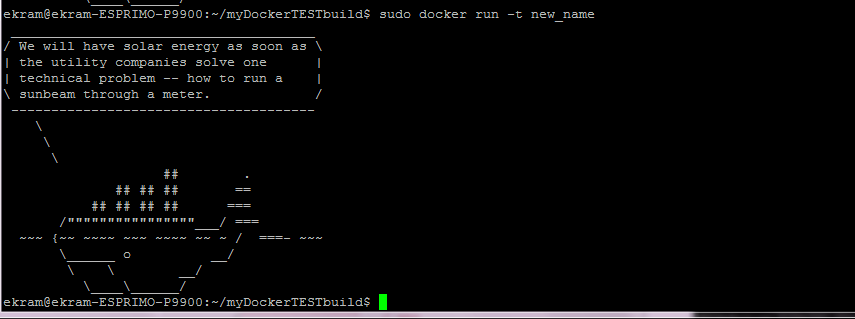
1. now if use the command 🡪 #docker images you will see your image



1. now you can run your image using the command 🡪 # docker run –t new\_name



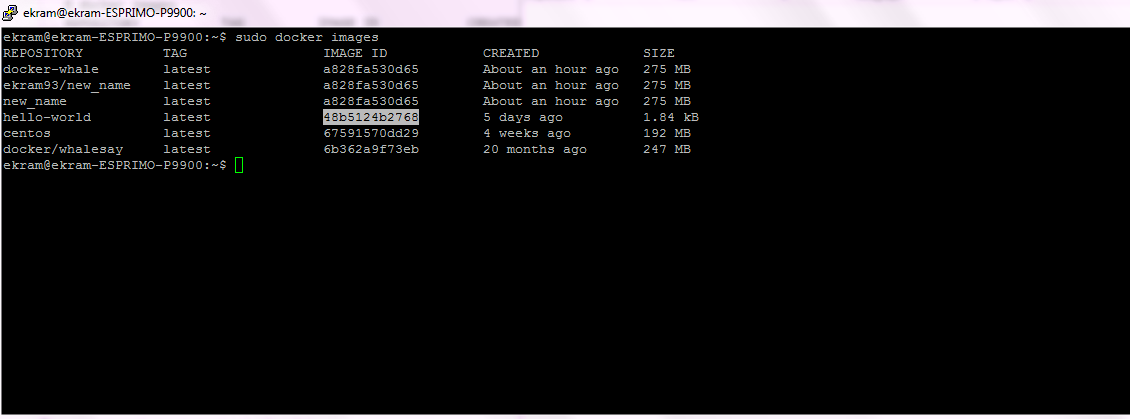
For funny you can run it again, you will see another statement



# **Create a Docker Hub account and repository**

1. go to [the Docker Hub signup page](https://hub.docker.com/register/?utm_source=getting_started_guide&utm_medium=embedded_MacOSX&utm_campaign=create_docker_hub_account) **and signup**
2. verify your email
3. then login in and create repository
4. in command line run the command 🡪 #docker images
5. then find image ID for your image you want to push

for example



1. # docker tag 48b5124b2768ekram93/hello-world:latest

Where

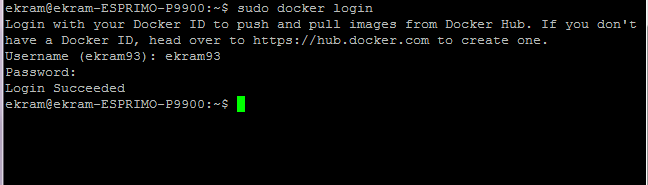
Docker🡪 tells your os that you are using docker program

Tag🡪 a subcommand that tags an image

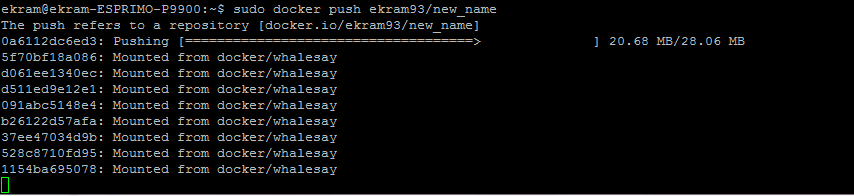
ekram93 is account name from the docker hub

hello-world is the image name

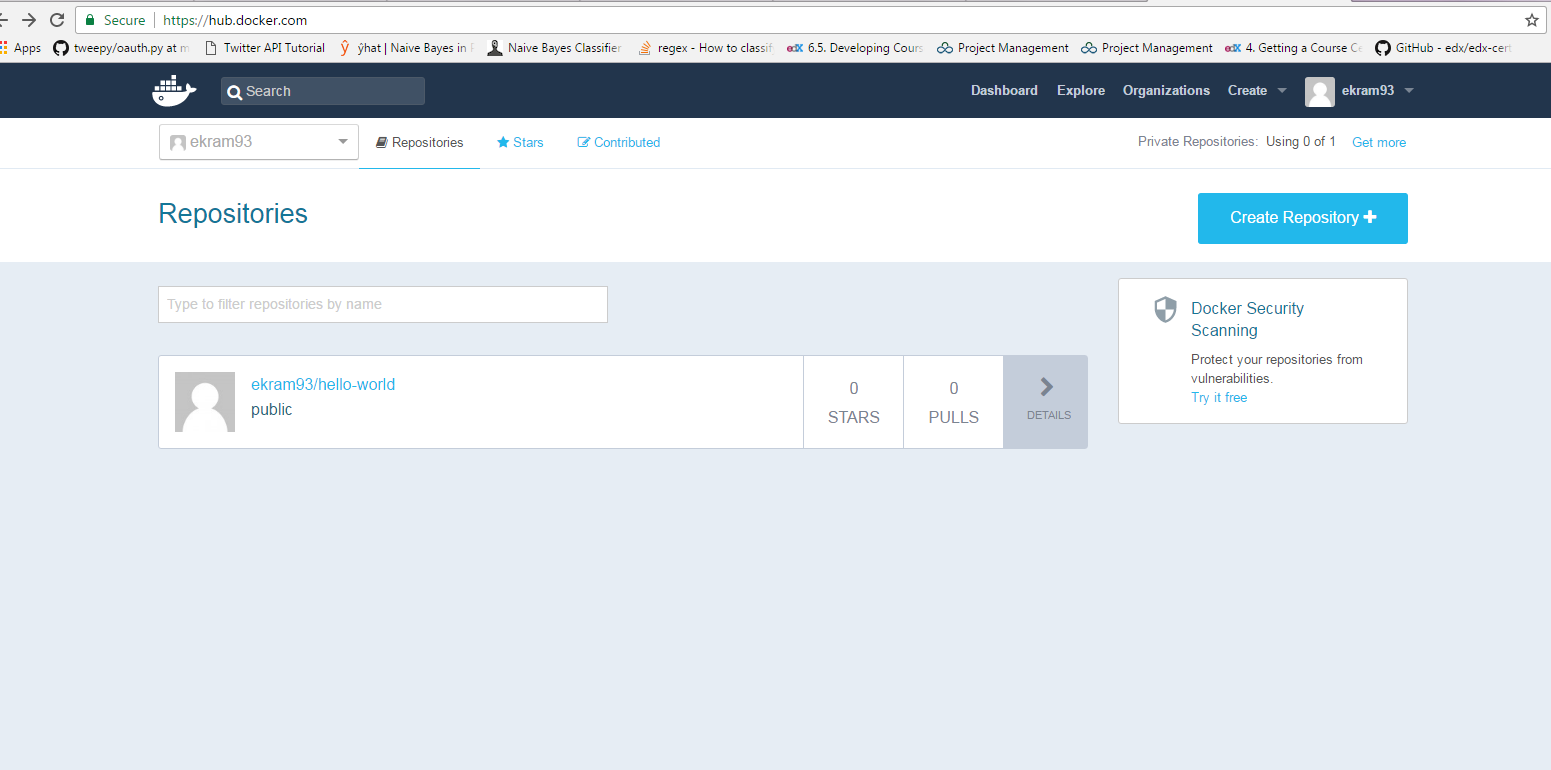
1. Login using 🡪 # docker login



1. Push the image # sudo docker push ekram93/hello-world



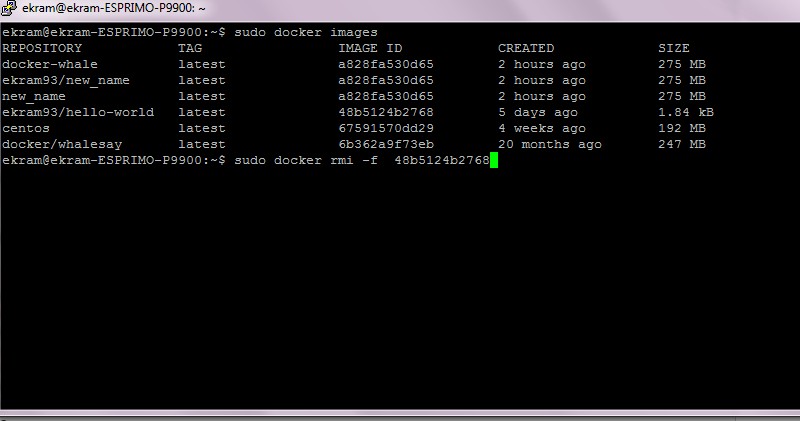
After pushing



**Pulling image from Docker Hub**

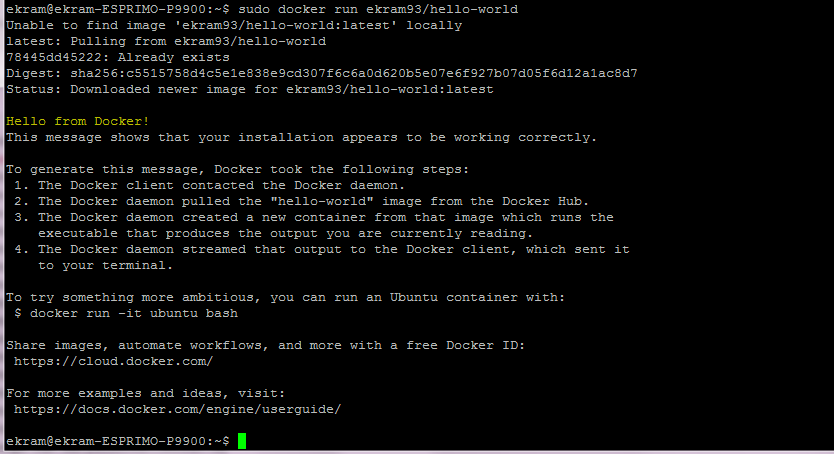
1. Make sure the image is not found, you can delete it using -- >

# docker rmi –f 48b5124b2768



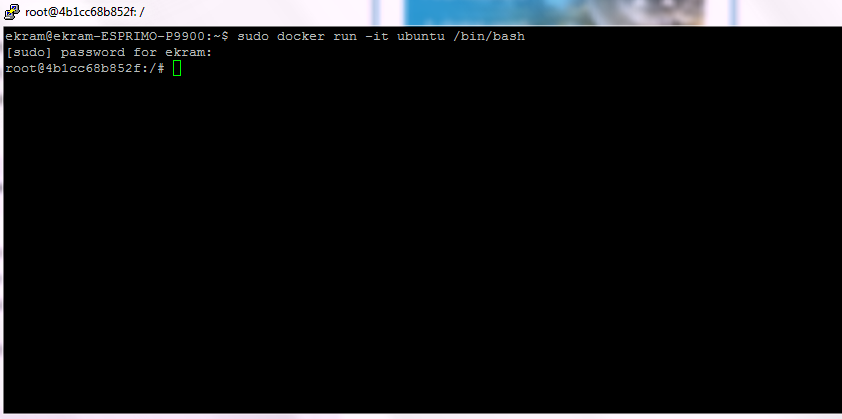
1. Pull and load a new image from your repository using the docker run command

# docker run ekram93/hello-world

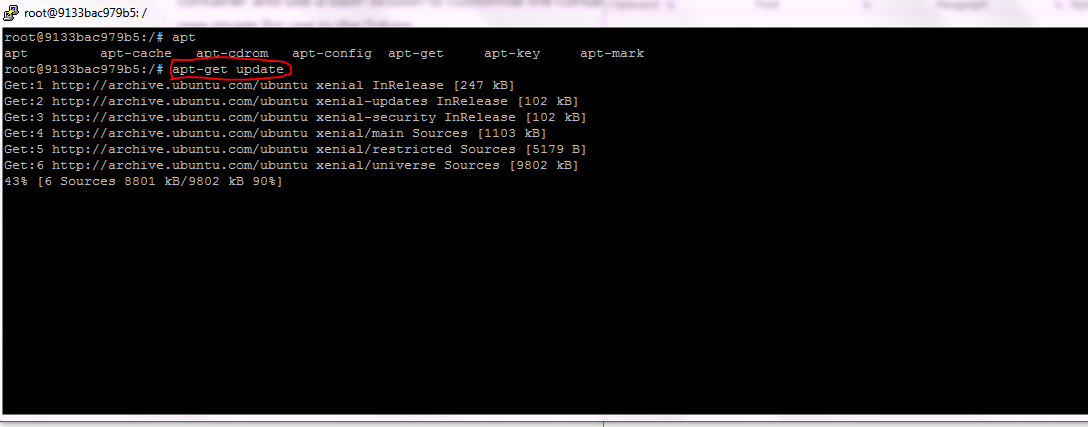


**Creating Container**

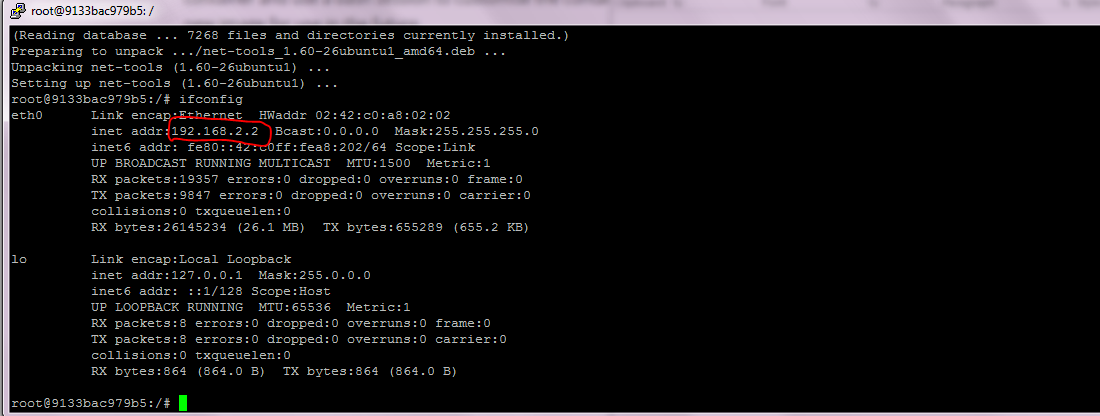
Create the Docker container with the **run** command and specify the **bash** shell to be executed on completion



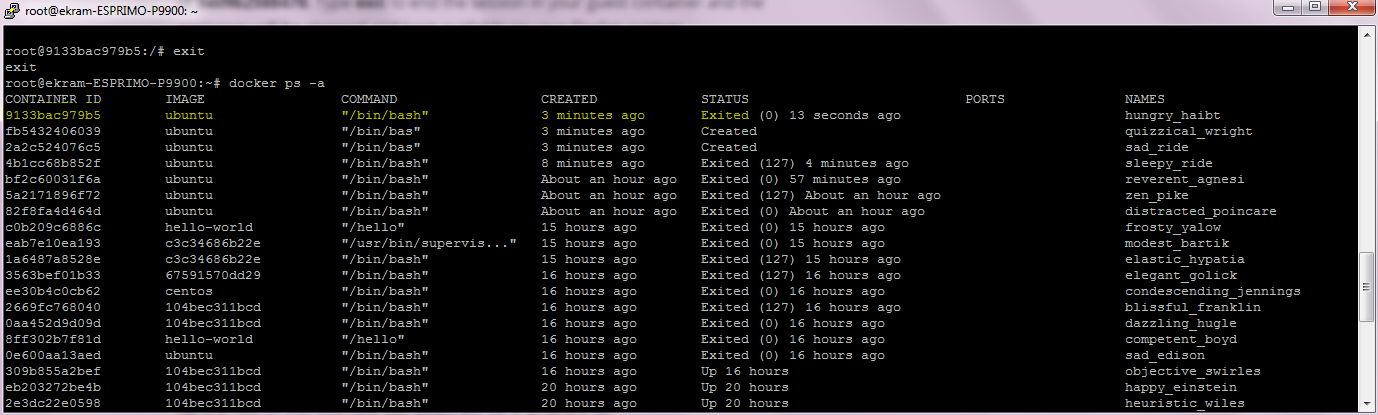
**Now** you can run any commands on it for example, I installed net-tool, apt-get and nano then update apt-get and gave the Ubuntu machine ip from my new network I created “mynet” 192.168.2.0



If you use #ifconfig you will see Ubuntu ip as shown



Exit and Run the **ps**Docker command to see what containers are known to your Docker system.

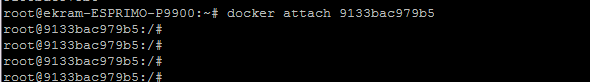


**Start a container**

#docker start “container id”



We can now **attach** to the container to create a shell where we can make our modifications.

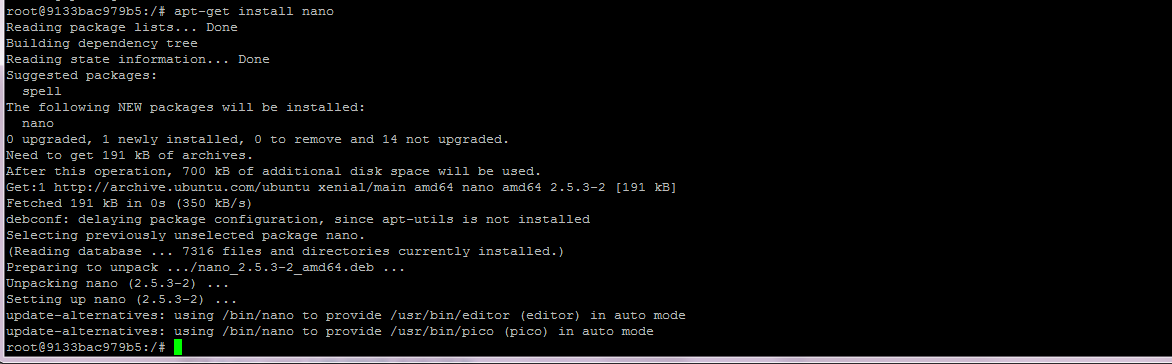


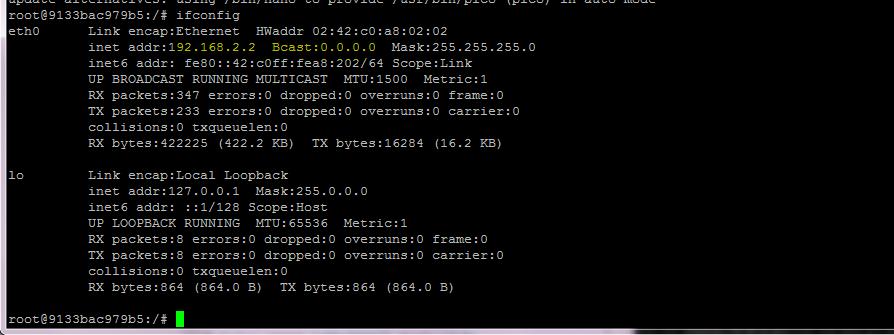
You now have a shell running on the container which you can use to make your changes to the container

Run #apt-get update

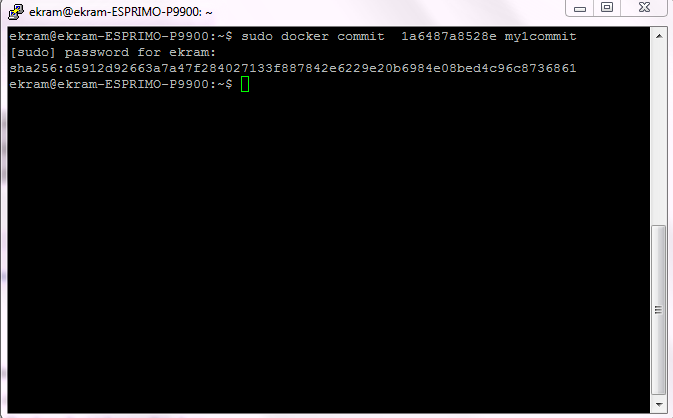
#apt-get install nano

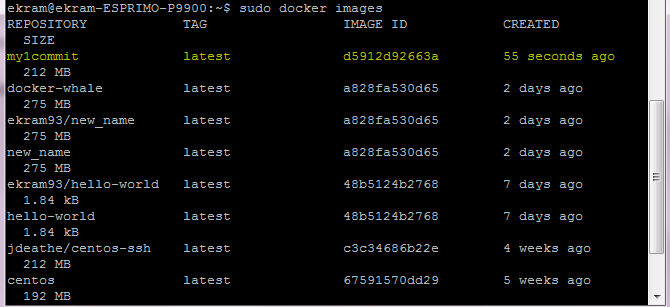
#ifconfig





The last step in our example is to save the container as a new image which can be used to create future Docker containers. You’ll need to specify the container ID as well as the name of the image to use. You can specify a new image name or overwrite the existing image name.

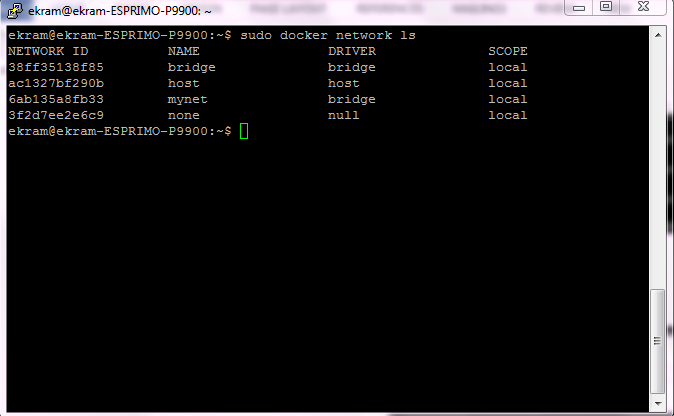




**Viewing networks**

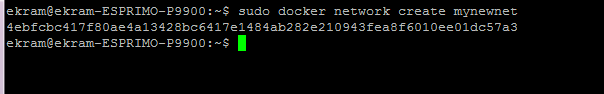
listing all networks

#sudo docker network ls

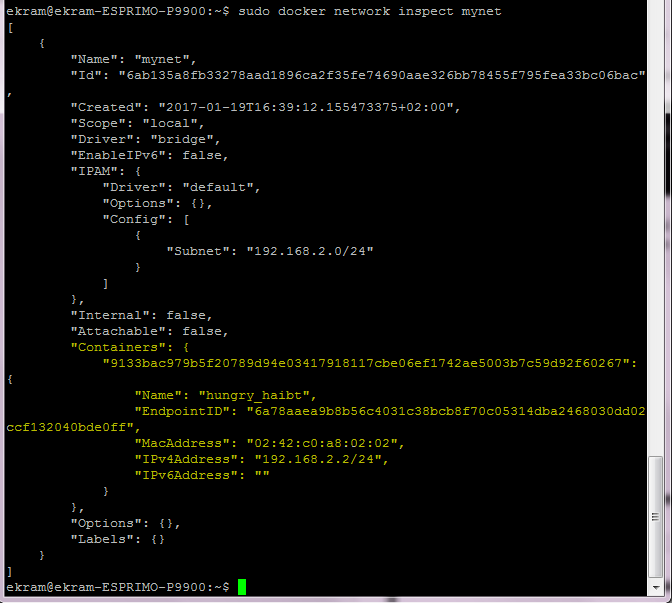


**Create new network**

# sudo docker network create mynewnet

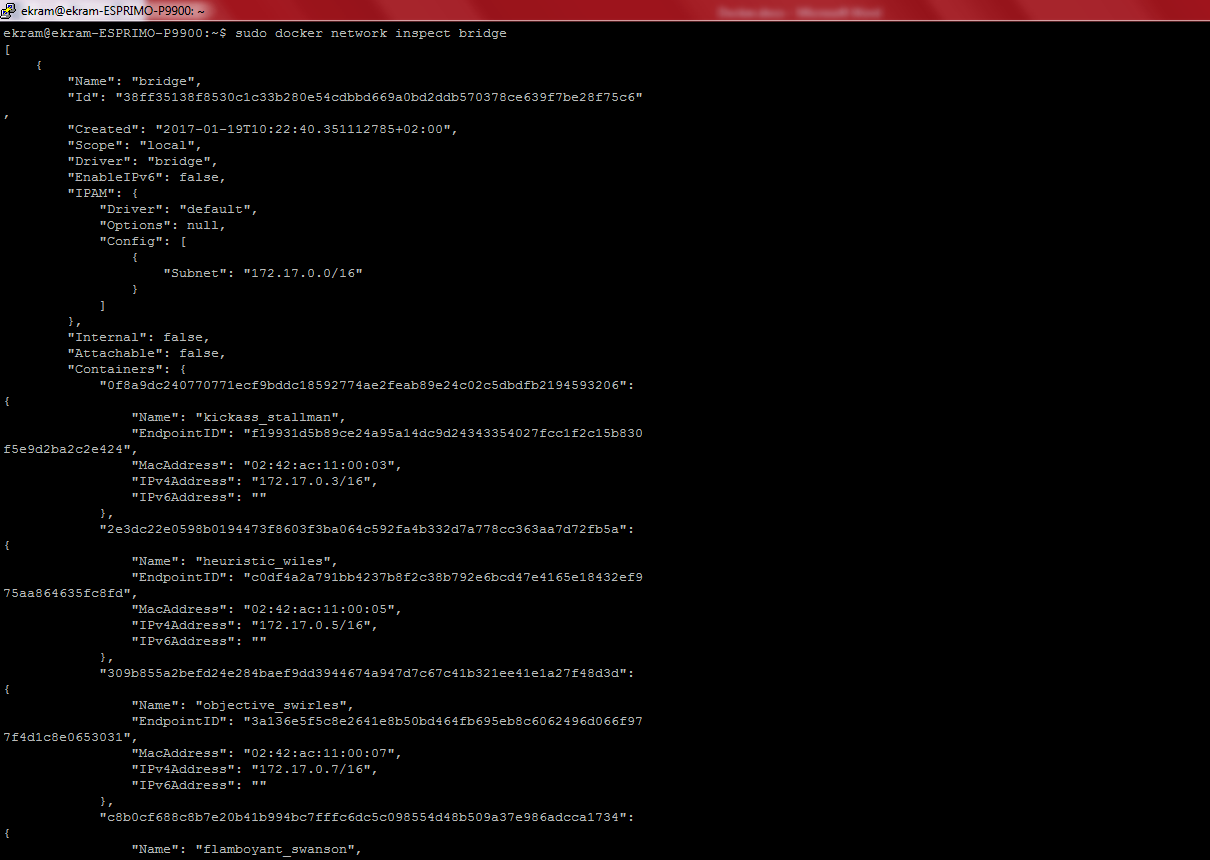


If you didn’t specify network id and gateway, it will take them automatically, but if you specified those argument you can see them with any container that take ip from this network.



You can view all information about any network using this command

# sudo docker network inspect “network name”



to force container to take from network u create use # sudo docker run --network=mynet-it “mynet” /bin/bash