**Steps to install Docker and Docker Compose**

1. Linux (ubuntu 16.04)

Before you install Docker for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

1. Install packages to allow apt to use a repository over HTTPS:

***sudo apt-get install apt-transport-https ca-certificates curl software-properties-common***

1. Add Docker’s official GPG (GNU Privacy Guard) key2:

***curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -***

1. Verify that the key fingerprint is 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88

***sudo apt-key fingerprint 0EBFCD88 pub***

4096R/0EBFCD88 2017-02-22 Key fingerprint = 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88 uid Docker Release (CE deb) <docker@docker.com> sub 4096R/F273FCD8 2017-02-22

1. Use the following command to set up the stable repository.

***sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"***

1. Now the repository is setup, we can install the Docker. First update the apt package index.

***sudo apt-get update***

1. Install the latest version4 of Docker by using this command.

***sudo apt-get install docker-ce***

1. After the installation is complete you can verify the installation by running this command.

***sudo docker run hello-world***

1. To install docker-compose run the following commands:

***curl -L https://github.com/docker/compose/releases/download/1.13.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose***

***chmod +x /usr/local/bin/docker-compose***

1. In Windows

***https://docs.docker.com/docker-for-windows/install/***

**Steps to execute the exercise**

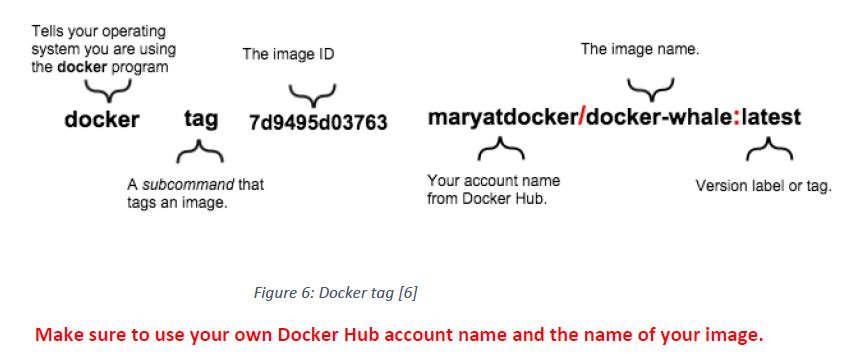
1. Complete the microservice **product-description-service( to get product name and URL)** and **product-price-service(to get product price),** based upon the product id passed as the query parameter. You need to complete the following files
2. **product-price-service/product\_price.js**
3. **server/services/product\_price.js**
4. **product-descp-service/product\_descp.js**
5. **server/services/product\_descp.js**
6. **docker-compose.yml (add your docker hub id)**
7. Install docker and docker compose on the VM.
8. After installation run this application on the VIM as explained above. Check all the services running using docker ps command and testing on the browser.
9. Enable docker remote API:
   1. Edit the file ***/lib/systemd/system/docker.service***
   2. Modify the line that starts with ExecStart to look like this ***ExecStart=/usr/bin/docker daemon -H fd:// -H tcp://0.0.0.0:4243***

Where the addition is ***“-H tcp://0.0.0.0:4243”***

* 1. Save the modified file
  2. Run ***systemctl*** daemon-reload
  3. Run ***sudo service docker restart***
  4. Test that the Docker API is indeed accessible:

***curl http://localhost:4243/version***

1. Enable port 4243 on your VM so that docker API can be accessed from outside network using your VM IP.
2. Run ***docker-compose up –build*** to build images, container and to run it
3. If application stops, then use this command to run it again after the first build - ***docker-compose up***
4. Push images to docker hub
5. As you already have built the image and created the docker hub account, so now it’s the time to tag your image and push it to your docker hub account. Tags are used to identify different versions.
6. Open the terminal and run the ***$ docker images*** command.
7. Find the image ID for the ***“image\_name”*** image, in the third column.
8. Tag the ***“image\_name”*** image using the docker tag command and **the image ID**. The command you must type looks like this:



1. Run docker images again to verify that the ***“image\_name”*** image has been tagged. The same image ID now exists in two different repositories.
2. Before you can push the image to Docker Hub, you need to log in, using the ***docker login*** command. The command doesn’t take any parameters, but prompts you for the username and password, as below:

***$ docker login***

***Username: \*\*\*\*\****

***Password: \*\*\*\*\****

***Login Succeeded***

1. Push your tagged image to Docker Hub, using the docker push command.

***$ docker push “your\_account\_id”/”your\_application\_image\_name”***

1. Go back to the Docker Hub website to see the newly-pushed image in your account.

**Want to learn more (optional):**

1. Delete your images ***“docker rmi “image ID”***

2. Pull the images from the docker hub. ***“docker pull hub\_ID/image\_name”.*** Now you don’t need dockerfile to create the image. As in previous step, you already created the images and pushed into the docker hub.

3. Now same images which you just pulled, can be used in order to run the microservice application.

**Testing the application**

Run the api on the browser, the api format will look something like this (productid can be changed):

***http://YOUR\_VM\_IP:8080/exercises/exercise3?name=CCS&productId=3***

and the output will contain a below message with all the fields values set according to the product id.

***{ "hello": "", "product\_id": "", "productURL": "", "productPrice": "", "productName": "" };***