**Getting to Know Docker & Kubernetes**

**Objectives:**

By the end of this lab you will know how to set up and run a Kubernetes environment with Docker containers implementation, with the help of the following steps:

1. Set up Docker & Kubernetes environment.
2. Understand how projects run in a containers environment.
3. Create a full Docker containers workload environment with Kubernetes.

**Environment Description:**

Log in to your physical station with:

Username: student

Password: student

Use SSH to connect to the following servers with user "student" password "student":  
Use "sudo su –" to become root.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Kubernetes Master** | **Kubernetes Node** | **Docker Only** |
| **Student 1** | student-kube-master1 10.90.41.31 | student-kube-node1  10.90.41.32 | student-docker1  10.90.41.33 |
| **Student 2** | student-kube-master2 10.90.41.37 | student-kube-node2  10.90.41.38 | student-docker2  10.90.41.39 |

**Lab Description**

The following milestones describes the proper way to complete this lab successfully

* Follow the lab steps in order.
* Each step relies on the outcome of the one before.
* There are multiple student environments. Each student should work on their own environment.
* The servers you will be working on are installed with RHEL 7.2 and registered to Red Hat.

**Lab 1 Steps:**

In this lab you will first log in to your student Docker server (“student-dockerX”), install Docker and pull an image from Red Hat’s remote registry.

The image includes a basic installation of RHEL 7, and the image is registered to Red Hat.

You will install a web server on that image, and access it from your physical station browser.

Log in to your Docker server and check if the “Extras” repository is enabled:  
*# subscription-manager repos --list-enabled*

If not, enable it:

*# subscription-manager repos --enable=rhel-7-server-extras-rpms*

Install Docker:  
*# yum install -y docker*

* Note: you cannot install Docker from other repositories.

Start and enable the Docker Service:

*# systemctl start docker*

*# systemctl enable docker*

Get an Image from a remote Docker registry:

*# docker pull registry.access.redhat.com/rhel7/rhel*

To view all images on your system, use the following command:  
*# docker images -a*

In order to run the container and log in to it’s shell:

*# docker run -it <image\_id> /bin/bash*

* Note: “*docker run -it <****image\_id****> /bin/bash*” is used when creating a new container from an image. In order to login to an existing container’s shell, use this:  
  “*docker exec -it <****container\_id****> /bin/bash*”.

Now you are inside the container’s shell.

Install httpd on the container:

*# yum install httpd -y*

Create an “index.html” file. Write anything you want inside (e.g. Docker is awesome):

*# vi /var/www/html/index.html*

Start HTTPD:

*# httpd -k start*

* If this is the output you got: “AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using X.X.X.X . Set the 'ServerName' directive globally to suppress this message”- **It’s OK !**

Use “curl” to check your webpage:

*# curl localhost:80*

* Note: try to run “curl localhost:80” from your Docker server. It should fail =) .

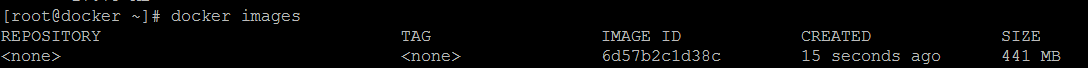
Open a new ssh terminal to your Docker server.

Now we will create a new image from the modified container:

*# docker commit <container\_id>*

Run “docker images” to view your newly created image.

Should be something like this:



Let’s assign a name for this new image:

*# docker tag <image\_id> <name>*

* Note: The default tag is “latest”, so when you use “*docker tag <name> <image\_id>*”, the repository name will be <name> and the tag name will be “latest”. To assign a different tag name, use “*docker tag <name>:<tag> <image\_id>*”

Run a new container from the new image, this time give it a port to expose. This command will map the container’s port 80 out to your server’s port 8080:

*# docker run -it -p 8080:80 <name>:<tag> /bin/bash*

Run HTTPD again:

*# httpd -k start*

Try to run “curl localhost:**8080**” again on your Docker machine. it should work =) .

If it did work, open a browser on your physical station, and type in the URL box:

10.90.41.X:8080

If you type “docker ps” you can see what ports you are exposing.

**Lab 2 Steps:**

In this lab you will log in to your student Kubernetes-Master server (“student-kube-masterX”) and your Kubernetes-Node server (“student-kube-nodeX”), which **already has Docker & Kubernetes installed.**

Run the following command on your **master** server to see that the cluster works:

*# kubectl get nodes*

On your **Master** server, the image from RedHat’s registry is already pulled.  
Run the following command to see it:

*# docker images*

Create a directory called “web”:

*# mkdir /root/web*

Create a docker file to build a new image from. Create it under /root/:

*FROM <image\_id>*

*EXPOSE 80*

*RUN yum install httpd -y*

*RUN echo "This is my server" > /var/www/html/index.html*

*CMD /usr/sbin/httpd -D FOREGROUND*

Great. Now Let’s build it!

*# docker build -t <new\_image\_name>:v1.0 . \_* (The dot is very important…)

Watch as your new image is being built. :)

Run *“docker images”* to see your newly created image.

This time, we will deploy a container using “kubectl”:

*# kubectl run mycontainer --image=<image\_id>*

Can you see your new container on your **Master**? (docker ps).

Check your **Kube-Node**…..

Go back to your **Kube-Master**, and create a new directory called “web2”

*# mkdir /root/web2*

and create a new dockerfile in it with the following content:

*FROM <NEW\_image\_id>*

*EXPOSE 80*

*RUN yum install httpd -y*

*RUN echo "This is my NEW server" > /var/www/html/index.html*

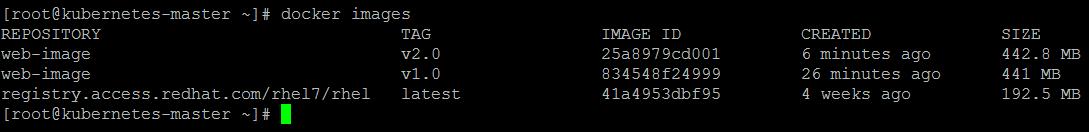
*CMD /usr/sbin/httpd -D FOREGROUND*

save it and run:

*# docker build -t <new\_image\_name>v2.0 .*

This time it should be a lot faster, since the only thing that is being changed is /var/www/html/index.html .

Check for your new image:



Run a new container:

*# kubectl run mycontainer --image=<New\_image\_id>*

Now we will play around with Kubernetes capabilities.

On your **Kube-Node** run:

*# docker ps*

Back to your **Kube-Master**, run the following:

*# kubectl get deployment mycontainer*

Remember this ?

Edit this deployment:

*# kubectl edit deployment mycontainer*

search for the value of the **image**, and change it from **v1.0** to **v2.0**.

On your **Kube-Node**, run “*docker ps*” again. Has your container changed?

Edit the deployment again:

*# kubectl edit deployment mycontainer*

Search for the value of **replicas**. Change it from **1** to **3**.

On your **Kube-Node**, run “*docker ps*” again. Is your container multiplied?

Go ahead and play some more!