

Exercise – Docker Registries and linking to Jenkins

Objective

While we can use the docker hub to push our images to, this leaves them either open for anyone to view or needing to pay to keep them private. We can create our own docker registry quickly and easily by using a docker container.

The second part of the exercise looks at how to configure Jenkins to build Docker containers. This will be useful for the case study!

Overview

Step 1: Create a Docker Registry

The source code for the **Docker Registry** is available for inspection, but the simplest method for creating a docker registry is to use Docker itself. (See <https://github.com/docker/distribution>)

1. Create a docker container from the Registry:

```
$ docker run -d -p 5000:5000 registry:2
```

This will set up a docker container running the registry code on port 5000.

Step 2: Push to the local registry

Docker pull and push default to the main docker hub when we use them. By telling Docker to use the local registry we can push to our own registry.

2. Create a Dockerfile that does the following:
 - a. Uses ubuntu
 - b. Updates and upgrades everything
 - c. Install git
 - d. Install vim

```
FROM ubuntu
MAINTAINER Lina
RUN apt-get update -y
RUN apt-get upgrade -y
RUN apt-get install -y git yum
```

Build your docker image:

```
$ docker build -t imagename .
```

Tag it "**localhost:5000/<yourname>baseimage**" for example:
localhost:5000/alicebaseimage

If you have built it with a different tag then you can use the following to retag your image:

```
$ docker tag oldtag newtag  
$ docker tag alicebaseimage localhost:5000/<name>baseimage
```

3. Push your image to the local repository

```
$ docker push localhost:5000/<yourname>baseimage
```

Step 3: Pull from the local registry

We want to remove the docker image we just created that is stored locally on our machine so we can see that we are pulling the image from our docker registry.

4. Remove the docker image you have just created

```
# List all the images:  
$ docker images  
  
# Remove your image:  
$ docker rmi -f localhost:5000/<yourname>baseimage  
  
# List all images again, confirm that your image is gone  
$ docker images
```

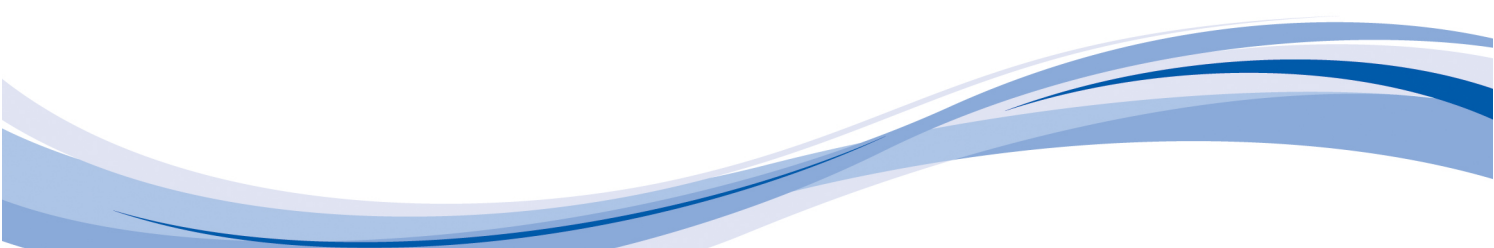
5. Test - pull your image from docker hub. It should show that your image could not be found.

```
$ docker search <yourname>baseimage  
or  
$ docker pull <yourname>baseimage
```

Searching will not work on the local registry, the search command points to the main docker hub

6. Pull the image from the local registry to confirm that we can get the image back!

```
$ docker pull localhost:5000/<yourname>baseimage
```



Create an image locally on Jenkins machine

1. First we need to get Jenkins and Docker on the same machine
 - a. Either install Jenkins on the Docker machine
 - b. Or install **Docker** on the **Jenkins** machine

```
$ sudo yum update -y
$ sudo yum install -y docker
$ sudo service docker start
```

2. Create the **hello-scalatra** project in **Jenkins** pulling from the git repository

```
$ sudo git clone https://bitbucket.org/<User>/\
hello-scalatra
```

3. To check how the proces looks manually, you need to install **Maven** and package **hello-scalatra** project on your **Jenkins** machine:

```
$ sudo yum -y update
$ sudo yum install -y java-1.7.0-openjdk-devel.x86_64
$ wget http://apache.mirrors.ionfish.org/maven/maven-\
3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz
$ sudo tar xzf apache-maven-3.3.9-bin.tar.gz -C /usr/local
$ sudo ln -s /usr/local/apache-maven-3.3.9 /usr/local/maven
$ sudo ln -s /usr/local/maven/bin/mvn /bin/mvn
$ cd hello-scalatra
$ mvn package
```

4. Add a **Dockerfile** in the base directory of your **hello-scalatra** project inside your **Jenkins** machine:

Hint: You may need to clone the project, add the file and commit the changes

```
FROM tomcat
MAINTAINER [you]
COPY target/scalatra-maven-prototype.war \
    /usr/local/tomcat/webapps/scalatra-maven-prototype.war
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

5. Create an image using **Dockerfile**:

```
$ docker build -t [yourusername]/hello-scalatra .  
$ docker run [yourusername]/hello-scalatra
```

Use Jenkins to set off a docker build

1. Create/Copy a **Dockerfile** on your Git project (inside Bitbucket or GitLab):

Dockerfile



2. In Jenkins we need to add another plugin. Install the **CloudBees Docker Build and Publish** plugin
3. Go to the **hello-scalatra** project configuration
4. Click on **post-build steps** and select **Docker Build and Publish**
5. Fill in the boxes with the following
 - a. Your repository name: [username]/hello-scalatra
 - b. Tag: **Jenkins\$BUILD_NUMBER**
The \$BUILD_NUMBER variable will be replaced by Jenkins with the current build. This is a good way to keep track of your versions.
 - c. Docker server URI – leave blank to use the same machine Jenkins is hosted on
 - d. Docker registry URL – leave blank to use Docker hub
 - e. Docker registry credentials – add your username and password for the Docker hub:

Post Steps

☒ Run only if build succeeds ☐ Run only if build succeeds or is unstable ☐ Run regardless of build result
Should the post-build steps run only for successful builds, etc.

Docker Build and Publish

Repository Name

Tag

Docker server URI

Server credentials

Docker registry URL

Registry credentials

6. Now we need to ensure that **Jenkins** is allowed to run **Docker** commands. To do this we will add **Jenkins** to the **Docker** group

On the terminal type

```
# add the group, should exist already
$ sudo groupadd docker
# add Jenkins user to the docker group
$ sudo gpasswd -a jenkins docker
# restart the docker and Jenkins services
$ sudo service docker restart
$ sudo service jenkins restart
```

7. Build your project and watch it push your new container to the Docker hub registry. You will be able to pull down a copy on another machine and run it
- a. If the Docker build fails, reboot your machine and try it again