Docker Registrys & Linking Docker and Jenkins

DevOps Practitioner

transforming performance
through learning

Outline

- **Docker Registry**
  - Creating a docker registry
  - Building from source
- **Using the registry**
  - Local access
  - Searching the repository
  - Registry UI project
- **Jenkins and Docker**
  - Build the docker container
  - Push to the docker hub

Objective

- **By the end of this session you should be able to**
  - Create a docker registry
  - Push and pull from our own private registry
  - Make Jenkins set off the Docker build and push it to the registry

The Docker Registry

- **Docker hub is a copy of the Docker registry**
  - When we use the commands to pull and push to the registry these are shortcuts

```
# docker pull ubuntu
```

  - Is the same as

```
# docker pull registry-1.docker.io/library/ubuntu
```

- **The docker hub can store private images**
  - But you need to pay
- **You can run your own registry**
  - Enterprise – paid for, support provided
  - Open Source – Build yourself or use the docker container

QADEVOPPRAC v1.7

Creating a Docker Registry

- **There is a docker image for the registry**

```
# docker run -d -p 5000:5000 registry:2
# docker run -d -p 5000:5000 registry:0.9.1
```

- **Version 2 was released on April 16th 2015**
  - Previous versions are available to pull and run
  - Version 2 is focused around security and performance
  - Does not have all the functionality of previous versions (such as search)

- **The data is all stored in the container**
  - Not best practice
  - Instead - mount a volume

```
$ docker run -d -p 5000:5000 \
  -e REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY=/var/lib/registry \
  -v /myregistrydata:/var/lib/registry \
  registry:2
```

Git repository for the docker registry: https://github.com/docker/distribution

To configure the docker registry we use environment variables. See https://docs.docker.com/registry/configuration/ for more details

QADEVOPPRAC v1.7

Using the local registry

- **To push to the local registry we need to include the address in the tag**

```
# docker tag UserName/hello-scalatra  \
            localhost:5000/scala/hello-scalatra

# docker push localhost:5000/scala/hello-scalatra
```

- **To pull from a local registry we also use the hostname**

```
# docker pull localhost:5000/scala/hello-scalatra
```

6

To change tags on docker images:

Format:

```
# docker tag [old] [new]
```

Making the registry available

- **To remotely access your own repository you need to tell docker if it is secure or insecure**
  - Secure Docker requires HTTPS
    - Buy a certificate
    - Self sign a certificate
  - Tell docker to use the insecure registry
    - Edit the docker options file in /etc/default/docker
    - Add the line:

```
DOCKER_OPTS="--insecure-registry <IP Address>:5000"
```

    - Restart the service
  - Pull the image using the IP address / host name

```
# docker pull <IP Address>:5000/scala/hello-scalatra
```

Searching the local registry

- **Not yet enabled in version 2.0**
- **For previous versions of the docker registry:**
  - Just add the address of the repository

```
# docker search localhost:5000/UserName
NAME                          DESCRIPTION    STARS        ....
library/UserNamebaseimage                      0           ....
```

- **In v1 of the api this can also be accessed via a GET call to:**

```
http://<YourIP>:5000/v1/search
```

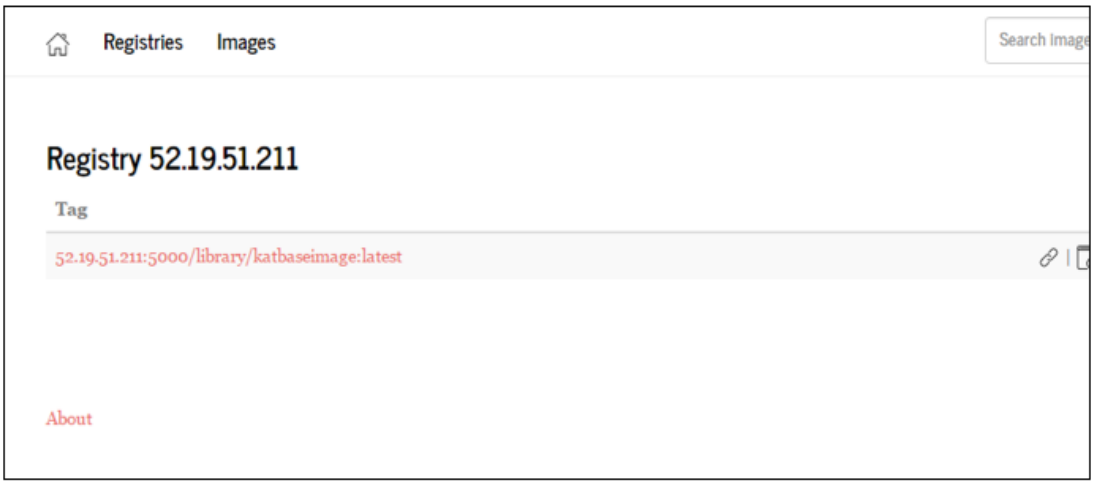
Docker Registry UI

- **The Docker registry uses JSON to communicate**
- **There is a UI available**
  - https://github.com/atc-/docker-registry-ui
  - Based on a (groovy) spring webservice running on port 8080
  - Connects to docker repositories
  - Works best with earlier versions of the docker registry as version 2 is not yet completed
  - https://github.com/atc-/docker-registry-ui
- **To run:**

```
docker run -d -p 8080:8080                    \
      -e REG1=http://<YourIP>:5000/v1/    \
      atcol/docker-registry-ui
```

Docker Registry UI

- **Set up links to multiple registries**
  - List and search the images stored inside



QADEVOPPRAC v1.7

Authentication

- **Version one of the stand alone registry does not use authentication itself**
  - Instead configure nginx or apache to use basic authentication before allowing access to the resource
- **Version two**
  - Uses certificates and token based authentication
  - If you don't have a signed certificate it is easier to setup security via apache or nginx
- **Docker will only consider a registry secure if it uses certificates**
  - If you want remote access without certificates then we need to run docker with the `--insecure-registry` flag

11

Walkthrough for certificate based registry:

https://github.com/docker/distribution/blob/master/docs/deploying.md#configure-tls-on-a-registry-server

Good tutorial for setting up authentication with nginx:

http://container-solutions.com/running-secured-docker-registry-2-0/

Linking Docker and Jenkins

- **We can create Docker containers which clone a repository and build a project**
  - Docker does not notice if the repo changes
  - You would need to re-build the entire container each time
  - Slow build
  - Best practice is to have the container as small as possible – so only have the deployment environment
- **Jenkins can be used to build the project**
  - Gives us the dashboard and stats
  - Plugin to let us set off Docker builds

The Plugin - Configuration

- **In theory we could just set off Docker builds by adding post-build steps**
  - Run a script which builds the docker container and pushes it
  - Issues with sudo access for the Jenkins user, even if we add it to the docker group
- **Cloudbees Docker build and publish plugin**
  - Adds a post-build action for Docker
  - We can set the Docker server to be anywhere
  - Push to either Docker hub or any repository of our choosing
- **Docker does not do version control by default**
  - We have to set a tag for it
  - Jenkins provides a $BUILD_NUMBER variable we can use

Jenkins Config

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	kizzie/hello-scalatra
Tag	jenkins$BUILD_NUMBER
Docker server URI	
Server credentials	- none - Add
Docker registry URL	
Registry credentials	kizzie/****** Add

Advanced...

Delete

Allow Jenkins to run Docker commands without sudo

- **We still need to enable Jenkins to run Docker commands**
  - A docker group is created when you install Docker
  - Users in this group can run docker commands without running as super user

```
# To add a user to a group
$ sudo gpasswd -a jenkins docker

# restart docker so it picks up on this
$ sudo service docker restart
$ sudo service jenkins restart
```

Build!

- **In the console you will be able to see the Docker build happening after the normal maven compilation is complete**
  - It will push the container to the Docker hub

```
------------------------------------------------------------------------
--------
[INFO] Total time: 9.711 s
[INFO] Finished at: 2015-06-18T10:11:47+01:00
[INFO] Final Memory: 16M/56M
[INFO]
------------------------------------------------------------------------
--------
[JENKINS] Archiving /var/lib/jenkins/jobs/hello-scalatra
/workspace/pom.xml to com.qa/hello-scalatra/1.0-SNAPSHOT/hello-
scalatra-1.0-SNAPSHOT.pom
[JENKINS] Archiving /var/lib/jenkins/jobs/hello-scalatra
/workspace/target/scalatra-maven-prototype.war to com.qa/hello-
scalatra/1.0-SNAPSHOT/hello-scalatra-1.0-SNAPSHOT.war
channel stopped
[workspace] $ docker build -t kizzie/hello-scalatra:Jenkins22
--pull=true .
Sending build context to Docker daemon 557.1 kB
Sending build context to Docker daemon 1.114 MB
Sending build context to Docker daemon 1.671 MB
```

```
Removing intermediate container 73fd6257b1e4
Successfully built d783ef7b52f8
[workspace] $ docker tag --force=true d783ef7b52f8 kizzie/hello-
scalatra:latest
[workspace] $ docker inspect d783ef7b52f8
[workspace] $ docker push kizzie/hello-scalatra:Jenkins22
The push refers to a repository [kizzie/hello-scalatra] (len: 1)
Sending image list
Pushing repository kizzie/hello-scalatra (1 tags)
64e5325c0d9d: Pushing
64e5325c0d9d: Image already pushed, skipping
bf84c1d84a8f: Pushing
bf84c1d84a8f: Image already pushed, skipping
87de57de6955: Pushing
87de57de6955: Image already pushed, skipping
e96a900a1f99: Pushing
e96a900a1f99: Image already pushed, skipping
cea65f1f366a: Pushing
cea65f1f366a: Buffering to disk
cea65f1f366a: Image successfully pushed
4302b8c85571: Pushing
4302b8c85571: Buffering to disk
4302b8c85571: Image successfully pushed
18915de4c9b5: Pushing
18915de4c9b5: Buffering to disk
18915de4c9b5: Image successfully pushed
82abb7bc23ec: Pushing
82abb7bc23ec: Buffering to disk
```

Exercise

- **Pull and run the docker registry image**
  - Push and pull to your repository
- **Get your Jenkins server to start building Docker containers and push them to either the docker hub or your own registry**

Summary

- **Docker Registry**
  - Creating a docker registry
  - Building from source
- **Using the registry**
  - Local access
  - Searching the repository
  - Registry UI project
- **Jenkins and Docker**
  - Build the docker container
  - Push to the docker hub

QADEVOPPRAC v1.7

Other Services – Artifactory (JFrog)

- **Artifactory can be used as a repository manager for many services**
  - Host your own repositories
  - Maven / Docker / Vagrant / NPM / Yum / Apt
  - http://www.jfrog.com/artifactory/features/
- **Need to be running 'pro'**
  - Open source version plus addons to allow different repositories
  - Create an artifactory server with docker

```
docker pull
    jfrog-docker-registry.bintray.io/jfrog/artifactory-pro

sudo docker run
    -p 8081:8081
    jfrog-docker-registry.bintray.io/jfrog/artifactory-pro
```

19

There is a docker image for an artifactory server which will act as a docker hub.

http://www.jfrog.com/confluence/display/RTF/Docker+Repositories