Task 01: build it

Building deployment packages in your own development environment is risky. Your environment is probably slightly different from the one used to prepare this guide. Fortunately, we can use docker images to collect and install all required dependencies.

Luckily Docker provides us with an official CentOS image based upon 64-bit Linux, which is binary rebuild of RHEL 7, therefore matches the RedHat. By running the installation inside Docker, we have a standard workflow that delivers an artifact matching our target runtime environment. In order to do so please clone repository and checkout proper branch:

```
$ git clone https://github.com/tcodeshare/devopschalange.git
$ cd devopschalange/
$ git checkout task_01
```

To build zip that will contains complete development environment for purpose of this guide, just type:

```
$ cd servletapp/src/main/docker/build-time-env
$ ./BUILDME
```

To copy & extract your zip file with development environment, just type:

```
$ ID=$(docker create ubs/build-time-env /bin/true)
$ docker cp $ID:/devenv.tar .
$ tar xfv devenv.tar -C /
```

Once the all required tools are available under **/opt/ubs** the next step will be to setup maven environment. In order to do so please type the following command from build-time-env folder:

```
$ source /opt/ubs/bin/build-time-setup.sh
$./setup-settings-security.py
Enter maven security settings master password:
Enter (maven/docker) repository password:
```

When: ”maven security settings master password” - is the **master password**, used to encrypt other passwords and the ”(maven/docker) repository password” correspond to the password of the user: ubs-uploader. After execution of the script the following two files will be created in your home directory in folder ~/.m2:

```
 $ ls ~/.m2/
settings-security.xml  settings.xml
```

Now you will be able to build and deploy application, please change the current working directory to one that contains top level pom.xml and type the following command:

```
 $ cd ../../../../ ; mvn deploy
```

The application will be deployed into the: http://nexus.ubs.net:8081/nexus/content/repositories/releases. You can run the application with the newly generated script:

```
 $ ./target/bin/servletApp
# Open new terminal and try to connect to application
$ curl localhost:16384
Hello World
```

Task 02: dockerize it

In this step application has been migrated to use Tomcat over Spring Boot asThe Spring Boot runtime is tested and verified to run with the Embedded Apache Tomcat Container.

This will ease the both docker image creation as well as further application evolution to Kubernetes. This is because of Spring Boot runtime gives you the advantages and convenience of the Kubernetes platform:

- service discovery
- rolling updates
- canary deployments
- ways to implement common microservice patterns: externalized configuration, health check, circuit breaker, and failover

In addition the farbic8 maven plugin configuration has been added to pom.xml configuration. Fabric8 is microservice platform on top of Kubernetes and OpenShift. It includes also a set of Docker images for creating Microservices. All images provided by fabric8 include run-java.sh, which is a generic startup script for starting Java Applications. With using of this approach we get rid of development of self-written Dockerfiles and startup scripts as the one provided by Fabric8 fits to requirement of this guide.

In order to build it we can simply use docker image created during task_01 – so we will run compilation and packaging of our servletApp docker in context of our docker image that describes build environment. By using this technique we are able to use any server in the development server farm. The only one requirement/dependencies is to having the docker daemon intstalled over there.

Please clone repository and checkout proper branch with new version of application:

```
$ git clone https://github.com/tcodeshare/devopschalange.git
$ cd devopschalange/
$ git checkout task_02
$ cd servletapp/
$ source /opt/ubs/bin/build-time-setup.sh
```

Please adjust the paths in script BUILDME, in particular

```
* /home/tomasz/.m2 – Should correspond to your local ~/.m2 directory
* /home/tomasz/work/guide/devopschalange/servletapp – Shuold correspond to your directory used to
extract the task_02.tar
```

Once paths in script adjusted please simply run script ./BUILDME. The script will create image (which you can cech with docker images command)

```
./BUILDME
…

$  docker images | grep servletapp
nexus.ubs.net:18081/ubs/servletapp   1.0                 3a932432dd08        2 minutes ago 112MB
$ docker run -p 16384:16384 -d nexus.ubs.net:18081/ubs/servletapp:1.0
d3dda9bd9318cc70a7c0c144cdc4aa85ef9bca50435abaff2eef3da8016fbd50
$ curl localhost:16384/
Hello World!
$ docker rm -f d3dda9bd9318cc70a7c0c144cdc4aa85ef9bca50435abaff2eef3da8016fbd50
```

Deployments notes

Please note that URL in format of http://nexus.ubs.net:8081/nexus/content/repositories/docker could not be used by the docker daemon, as the on that do not follows dockver V1 or V2 registry layout and don't provide TLS connection. Therefore this guide assume the Nexus Repository Connector Configuration", is available and configured for port: 18081. For details please refer to https://help.sonatype.com/repomanager3/private-registry-for-docker/ssl-and-repository-connector-configuration (**Figure 10.1. Repository Connector Configuration**)

Please note that this build might generated left-overs in your workspace . In order to clean them up, please simply edit BUILDME and pass "clean" as target instead of default "deploy fabric8:push".

Task 03: compose it

In order to run application with using of postgresql over docker-compose please extract the content of the tarball and change directory to one that containse docker-compose file.

```
$ git clone https://github.com/tcodeshare/devopschalange.git
$ cd devopschalange/
$ git checkout task_03
$ cd servletapp/
$ source /opt/ubs/bin/build-time-setup.sh
$ cd src/main/compose/
```

Please adjust the variables in files datasources-cm.env. Please note that this file is shared across containers so you don't need to maintain two separate files. Once completed please run docker-compose. Please be sure that docker trigger in task_02 has been removed (details in task_02).

```
$ docker-compose up # Please add -d to bypass terminal blocking. Otherwise use CTRL+C to stop it.
```

Please note that application is parameterized by redefinition of command. With using of this approach we don't need to maintain our own image, but use the one provided by fabric8 out-of-the-box. The postgres image that has been choose for this exercise will create required database and users so there is no need to do it over JDBC URL. Beside that the image is part of Kubernetes aware Crunchy suite (details: https://github.com/CrunchyData/crunchy-containers)

In order to verify port exposition please type command on hosts as follows. The connection to 5432 should fail:

```
$ telnet localhost 5432
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
$ curl localhost:16384
Hello World!
```

In order ports DB port is expose only to linked service without publishing it to the host machine, please run the same telnet command from servletapp container with using of service name

```
$ docker ps
CONTAINER ID        IMAGE                                                COMMAND
CREATED              STATUS              PORTS                                        NAMES
8952da8fab2f        nexus.ubs.net:18081/ubs/servletapp:1.0               "/bin/sh -c '\nexport…"
About a minute ago   Up About a minute   8778/tcp, 9779/tcp, 0.0.0.0:16384->16384/tcp
compose_servletapp_1
0b03c566cbeb        crunchydata/crunchy-postgres:centos7-9.6.10-2.1.0    "/opt/cpm/bin/start.…"
About a minute ago   Up About a minute   5432/tcp
compose_db-primary_1
$ docker exec -ti 8952da8fab2f sh
/ # telnet db-primary 5432
exit
Connection closed by foreign host
/ # exit
```

Once completed please terminate environment with:

```
$ docker-compose down
```