Lesson-End Project

Dockerizing a Java Program and Setting Up a Secure Local Registry

Project agenda: To establish a secure local registry and dockerize a Java program for efficient and consistent deployment across various environments

Description: Your company is experiencing a need to streamline deployment processes and ensure consistency across software environments. To address this, you are undertaking a project to dockerize a Java program and set up a local registry with security measures that enhance scalability and maintain deployment consistency.

Tools required: Docker and Ubuntu OS

Prerequisites: None

Expected deliverables: Dockerized Java program and a secure local registry

Steps to be followed:

1. Write a Java program and create a Dockerfile
2. Create a local Docker registry and run it with an htpasswd file
3. Build and tag the Docker image
4. Log in to the secured registry and push the Docker image to it

Step 1: Write a Java program and create a Dockerfile

1.1 Create a folder for the Java program and navigate into it using the following commands:
mkdir factorial_project
cd factorial_project

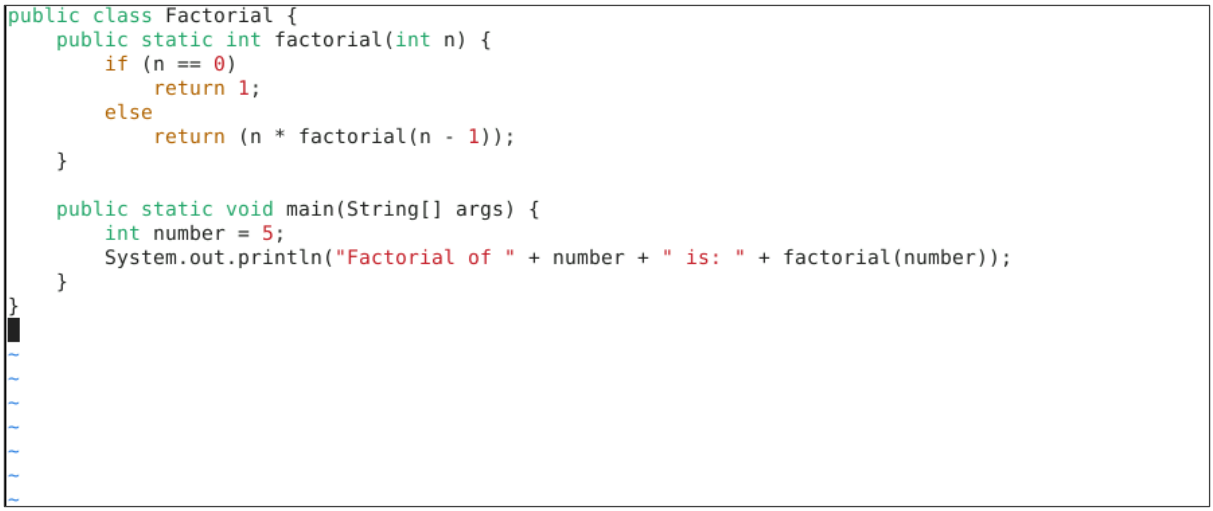
```
sakshiguptasimp@ip-172-31-27-122:~$ mkdir factorial_project
sakshiguptasimp@ip-172-31-27-122:~$ cd factorial_project
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$
```

1.2 Create a Java file named **Factorial.java** using the following command:
vim Factorial.java

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ vim Factorial.java
```

1.3 Write the Java program to calculate factorial using recursion, using the following code:

```
public class Factorial {
    public static int factorial(int n) {
        if (n == 0)
            return 1;
        else
            return (n * factorial(n - 1));
    }

    public static void main(String[] args) {
        int number = 5;
        System.out.println("Factorial of " + number + " is: " + factorial(number));
    }
}
```

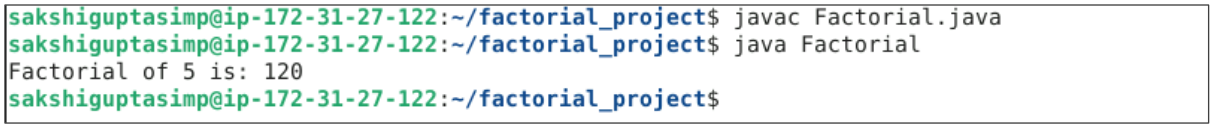


Note: To save the file and exit, press **Esc**, then type **:wq**, and press **Enter**

1.4 In the terminal, compile and run the Java program using the following commands:

javac Factorial.java

java Factorial



1.5 Create a Dockerfile using the following command:

vim Dockerfile

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ vim Dockerfile
```

1.6 Edit the Dockerfile and add the following snippet:

FROM openjdk:11-jdk
COPY Factorial.java .
RUN javac Factorial.java
CMD ["java", "Factorial"]

```
FROM openjdk:11-jdk
COPY Factorial.java .
RUN javac Factorial.java
CMD ["java", "Factorial"]
~
~
```

> **Note**: To save the file and exit, press **Esc**, then type **:wq**, and press **Enter**

This Dockerfile snippet uses an OpenJDK 11 image to compile and run the Java program **Factorial.java** by copying the source file into the container, compiling it, and executing the compiled class.

Step 2: Create a local Docker registry and run it with an htpasswd file

2.1 Create and run the local Docker registry using the following command:

docker run -d -p 5000:5000 --restart=always --name registry registry:2

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ docker run -d -p 5000:5000 --restart=always --name registry registry:2
212e2e0200f0e10ce150e07e3703054048194493f085e28b8e665dc9458545487
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$
```

2.2 Use the following commands to create a password file with a username and password used for logging in to the local registry:

mkdir auth

htpasswd -Bbn USERNAME PASSWORD > auth/htpasswd

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ mkdir auth
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ htpasswd -Bbn [redacted] > auth/htpasswd
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$
```

Note: Replace USERNAME with the desired username and PASSWORD with the desired password.

2.3 Run the registry container with the htpasswd file using the following command:
docker run --rm -v "$(pwd)/auth:/auth" registry:2

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ docker run --rm -v "$(pwd)/auth:/auth" registry:2
time="2024-03-14T10:18:47.17420911Z" level=warning msg="No HTTP secret provided - generated random secret. This may cause problems with uploads if mu
ltiple registries are behind a load-balancer. To provide a shared secret, fill in http.secret in the configuration file or set the REGISTRY_HTTP_SECR
ET environment variable." go.version=go1.20.8 instance.id=3679f1ff-b56f-40ed-851b-6947fac584e5 service=registry version=2.8.3
time="2024-03-14T10:18:47.174265412Z" level=info msg="redis not configured" go.version=go1.20.8 instance.id=3679f1ff-b56f-40ed-851b-6947fac584e5 serv
ice=registry version=2.8.3
time="2024-03-14T10:18:47.17433211Z" level=info msg="Starting upload purge in 40m0s" go.version=go1.20.8 instance.id=3679f1ff-b56f-40ed-851b-6947fac5
84e5 service=registry version=2.8.3
time="2024-03-14T10:18:47.174453963Z" level=info msg="using inmemory blob descriptor cache" go.version=go1.20.8 instance.id=3679f1ff-b56f-40ed-851b-6
947fac584e5 service=registry version=2.8.3
time="2024-03-14T10:18:47.174695944Z" level=info msg="listening on [::]:5000" go.version=go1.20.8 instance.id=3679f1ff-b56f-40ed-851b-6947fac584e5 se
rvice=registry version=2.8.3
```

Step 3: Build and tag the Docker image

3.1 Build a Docker image using the following command:
docker build -t factorial-java .

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ docker build -t factorial-java .
[+] Building 1.4s (8/8) FINISHED                                                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                                                       0.0s
 => => transferring dockerfile: 131B                                                                                                       0.0s
 => [internal] load metadata for docker.io/library/openjdk:11-jdk                                                                          0.1s
 => [internal] load .dockerignore                                                                                                          0.0s
 => => transferring context: 2B                                                                                                            0.0s
 => [internal] load build context                                                                                                          0.0s
 => => transferring context: 376B                                                                                                          0.0s
 => CACHED [1/3] FROM docker.io/library/openjdk:11-jdk@sha256:99bac5bf83633e3c7399aed725c8415e7b569b54e03e4599e580fc9cdb7c21ab             0.0s
 => [2/3] COPY Factorial.java .                                                                                                            0.1s
 => [3/3] RUN javac Factorial.java                                                                                                         0.9s
 => exporting to image                                                                                                                     0.1s
 => => exporting layers                                                                                                                    0.1s
 => => writing image sha256:fa0a6f43ebd9105a9086efba28fa0bea5a973c061927e889bfbcc1e9bc52b037                                               0.0s
 => => naming to docker.io/library/factorial-java                                                                                          0.0s
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$
```

3.2 Tag the Docker image using the following command:
docker tag factorial-java localhost:5000/factorial-java:v1.0

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ docker tag factorial-java localhost:5000/factorial-java:v1.0
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$
```

3.3 Push the tagged image to the local registry using the following command:
docker push localhost:5000/factorial-java:v1.0

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ docker push localhost:5000/factorial-java:v1.0
The push refers to repository [localhost:5000/factorial-java]
4268495f1d8a: Pushed
c18d946e323c: Pushed
7b7f3078e1db: Pushing [==================>                                ]  132.7MB/337.3MB
826c3ddbb29c: Pushed
b626401ef603: Pushed
9b55156abf26: Pushing [================================>                  ]  104.5MB/152MB
293d5db30c9f: Pushed
03127cdb479b: Pushed
9c742cd6c7a5: Pushing [========================================>          ]  101.8MB/124MB
```

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ docker push localhost:5000/factorial-java:v1.0
The push refers to repository [localhost:5000/factorial-java]
4268495f1d8a: Pushed
c18d946e323c: Pushed
7b7f3078e1db: Pushed
826c3ddbb29c: Pushed
b626401ef603: Pushed
9b55156abf26: Pushed
293d5db30c9f: Pushed
03127cdb479b: Pushed
9c742cd6c7a5: Pushed
v1.0: digest: sha256:54386e9ab17df67a57b956ff1ff09326e2a015e41a1f7141b1577e1d62ef3352 size: 2209
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$
```

Step 4: Log in to the secured registry and push the Docker image to it

4.1 Log in to the local secure registry using the following command:
docker login localhost:5000

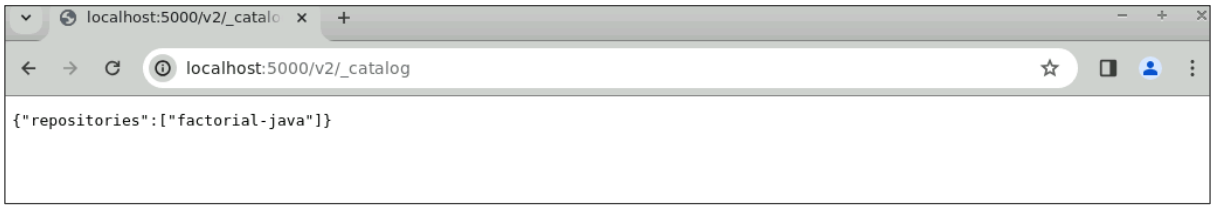
```
sakshiguptasimp@ip-172-31-27-122:~/       $ docker login localhost:5000
Username:
Password:
WARNING! Your password will be stored unencrypted in /home/sakshiguptasimp/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

Note: Enter the created USERNAME and PASSWORD. Refer to step 2.2 of this demo.

4.2 Push the Docker image to the secured registry:
docker push localhost:5000/factorial-java:v1.0

```
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$ docker push localhost:5000/factorial-java:v1.0
The push refers to repository [localhost:5000/factorial-java]
4268495f1d8a: Pushed
c18d946e323c: Pushed
7b7f3078e1db: Pushed
826c3ddbb29c: Pushed
b626401ef603: Pushed
9b55156abf26: Pushed
293d5db30c9f: Pushed
03127cdb479b: Pushed
9c742cd6c7a5: Pushed
v1.0: digest: sha256:54386e9ab17df67a57b956ff1ff09326e2a015e41a1f7141b1577e1d62ef3352 size: 2209
sakshiguptasimp@ip-172-31-27-122:~/factorial_project$
```

4.3 Browse to the following link to verify that the image is available in the local registry **http://localhost:5000/v2/_catalog**



By following these steps, you have successfully established a secure local registry and dockerized a Java program for efficient and consistent deployment across various environments.