How to Run/Deploy Spring Boot & EJB in wildfly Docker container.

In this article, I am going to explain how to deploy a RESTful Spring Boot application and EJB 3.2 on wildfly docker container.

## You Will Learn

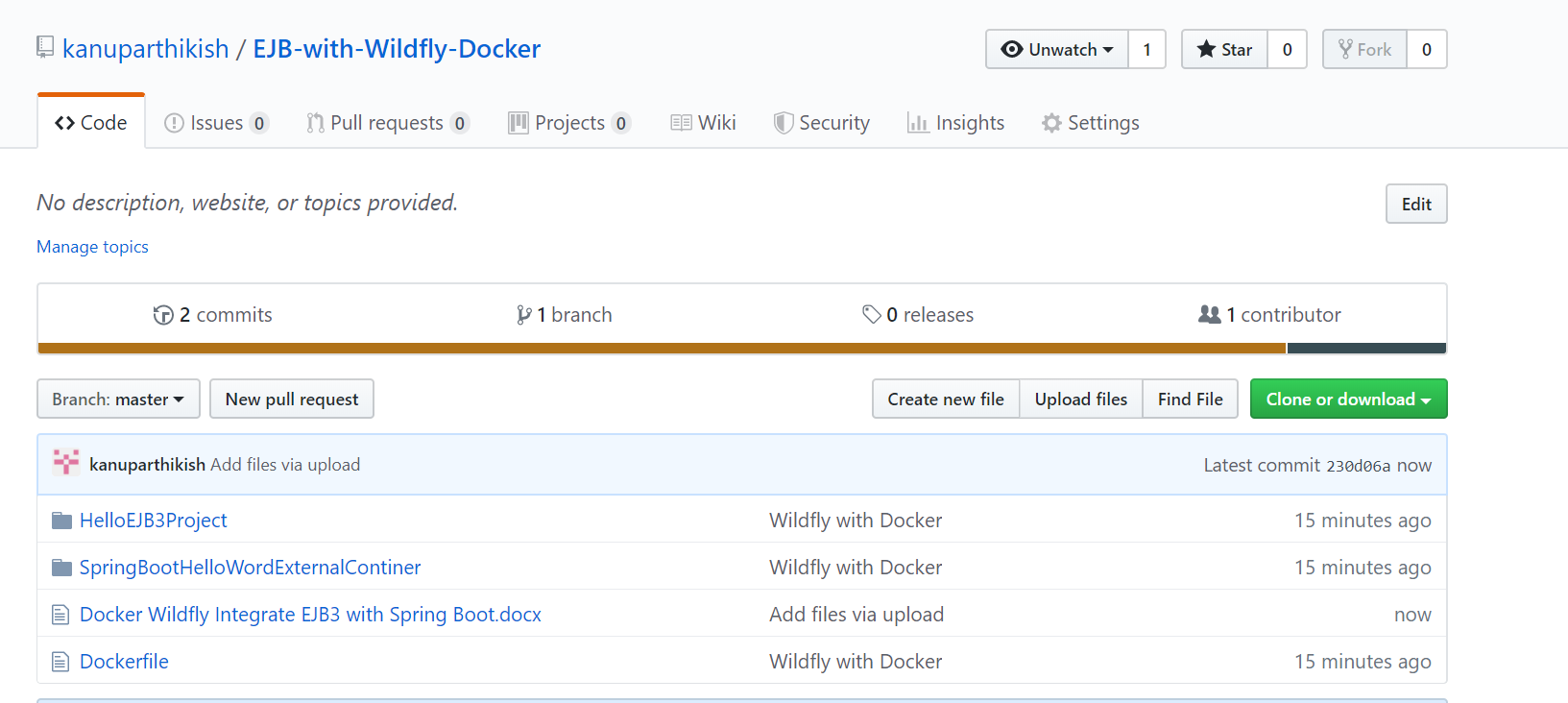
how can we deploy a spring boot REST API and EJB 3.2 in wildfly docker container

## Tools Required

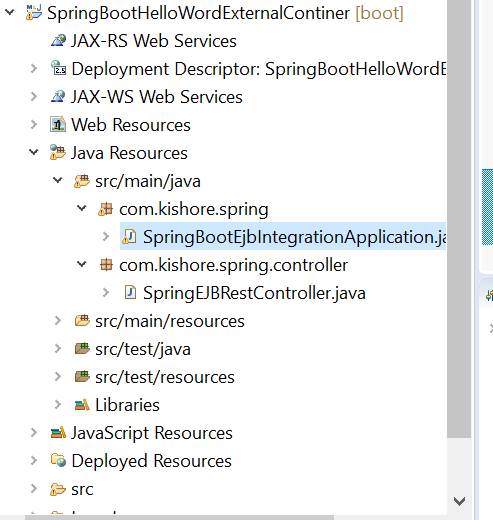
* Maven 3.0+ build tool
* STS or Eclipse as IDE.
* Docker
* Postman

## Complete Maven Project With Code Example is available in Github

<https://github.com/kanuparthikish/EJB-with-Wildfly-Docker>



2. Create the spring boot project structure as per below by using Spring Starter Project in eclipse IDE



3. Create the Dockerfile for Spring Boot and EJB 3.2 with wildfly application so make up your app run together in an isolated environment.

4. Dockerfile for spring Boot & EJB 3.2 application

***FROM jboss/wildfly***

***RUN /opt/jboss/wildfly/bin/add-user.sh admin admin***

***copy /SpringBootHelloWordExternalContiner/target/SpringBootHelloWordExternalContiner.war /opt/jboss/wildfly/standalone/deployments/***

***copy /HelloEJB3Project/target/HelloEJB3Project.jar /opt/jboss/wildfly/standalone/deployments/***

***CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-c", "standalone-full.xml", "-b", "0.0.0.0"]***

5. Rest Controller

@RestController

**public** **class** SpringEJBRestController {

@Autowired

HelloBeanRemote helloBeanRemote;

@GetMapping("/helloword")

**public** String getHello() **throws** Exception

{

**return** helloBeanRemote.getHelloMessage();

}

}

6. Spring Boot application class with EJB Bean Configuration

@SpringBootApplication

**public** **class** SpringBootEjbIntegrationApplication **extends** SpringBootServletInitializer {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringBootEjbIntegrationApplication.**class**, args);

}

@Override

**protected** SpringApplicationBuilder configure(SpringApplicationBuilder application) {

**return** application.sources(SpringBootEjbIntegrationApplication.**class**);

}

@Bean

**public** Context context() **throws** NamingException {

Properties jndiProps = **new** Properties();

jndiProps.put("java.naming.factory.initial", "org.jboss.naming.remote.client.InitialContextFactory");

jndiProps.put("jboss.naming.client.ejb.context", **true**);

// jndiProps.put("java.naming.provider.url", "http-remoting://localhost:8080"); commented for docker

jndiProps.put("java.naming.provider.url", "http-remoting://127.0.0.1:8080");

**return** **new** InitialContext(jndiProps);

}

@Bean

**public** HelloBeanRemote getLookUpObject(Context context) **throws** Exception

{

// return (HelloBeanRemote)context.lookup( "ejb:/HelloEJB3Project-0.0.1-SNAPSHOT/HelloBean!com.kishore.ejb3.HelloBeanRemote");

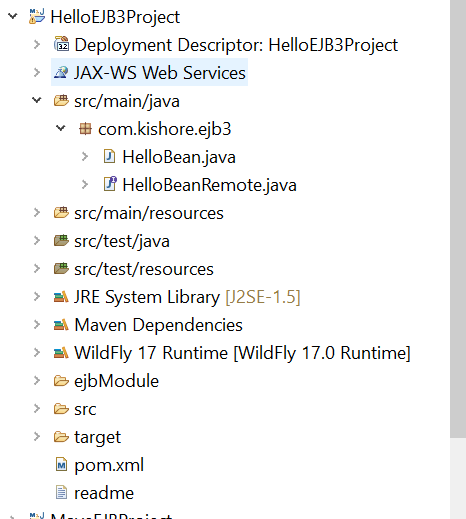
**return** (HelloBeanRemote)context.lookup( "ejb:/HelloEJB3Project/HelloBean!com.kishore.ejb3.HelloBeanRemote");

}

// }

}

7. Create the EJB project structure as per below



8. Create the EJB Remote Interface

@Remote

**public** **interface** HelloBeanRemote {

**public** String getHelloMessage();

}

8. Create the EJB Bean Class

@Stateless

@LocalBean

**public** **class** HelloBean **implements** HelloBeanRemote {

/\*\*

\* Default constructor.

\*/

**public** HelloBean() {

// **TODO** Auto-generated constructor stub

}

@Override

**public** String getHelloMessage() {

// **TODO** Auto-generated method stub

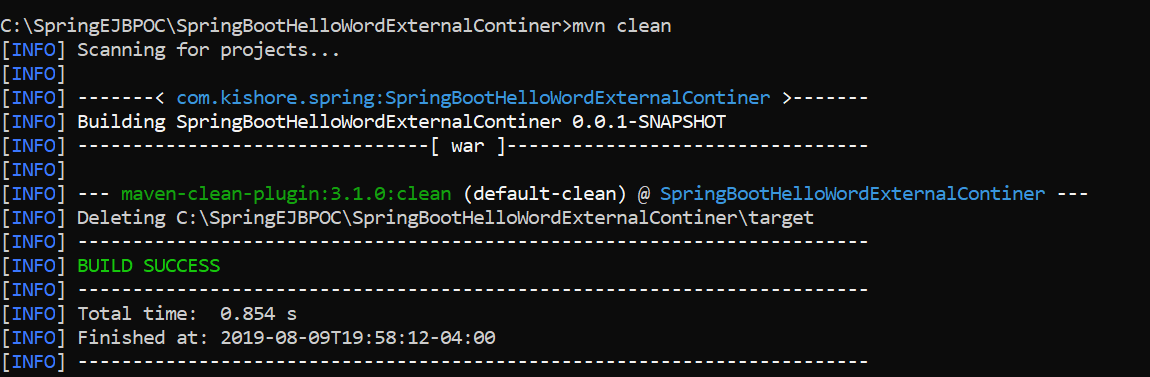
**return** "Hello";

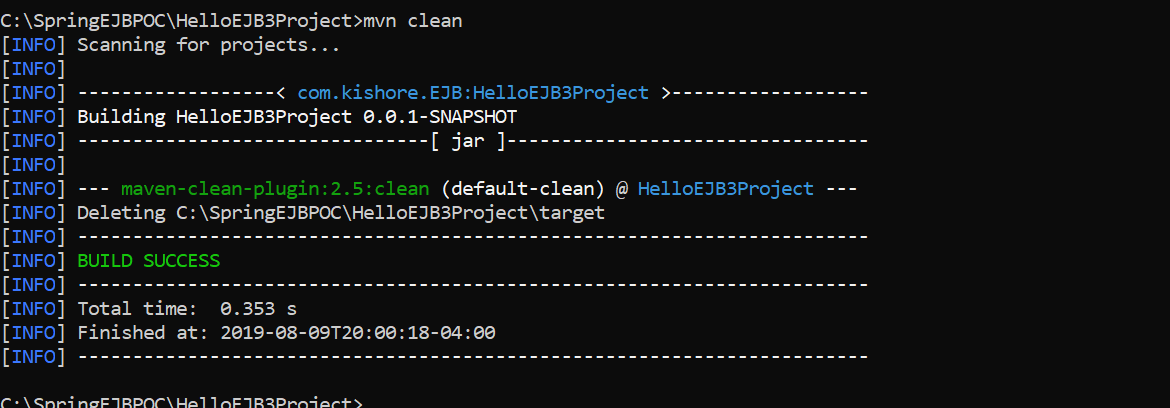
}

}

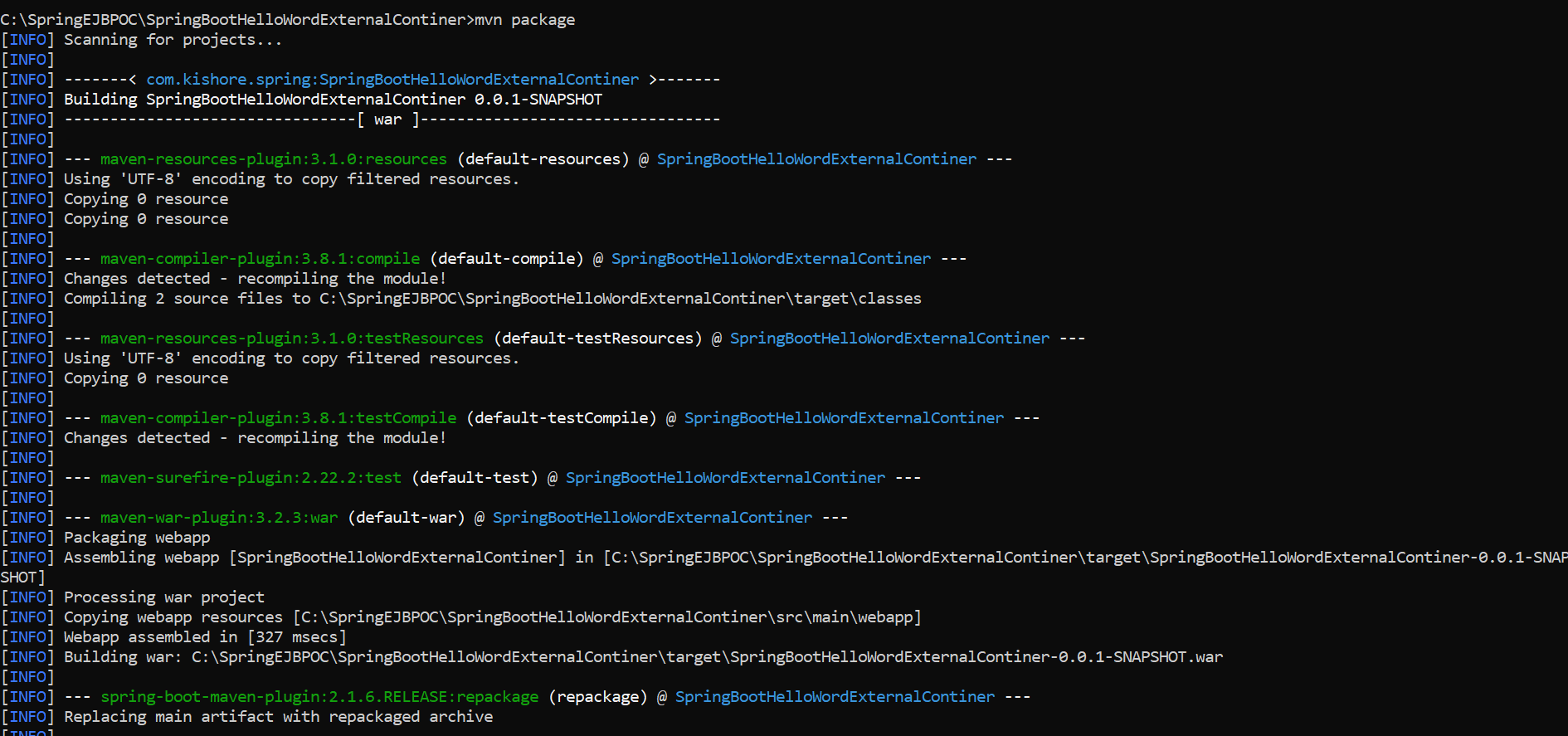
8. create the build package with maven

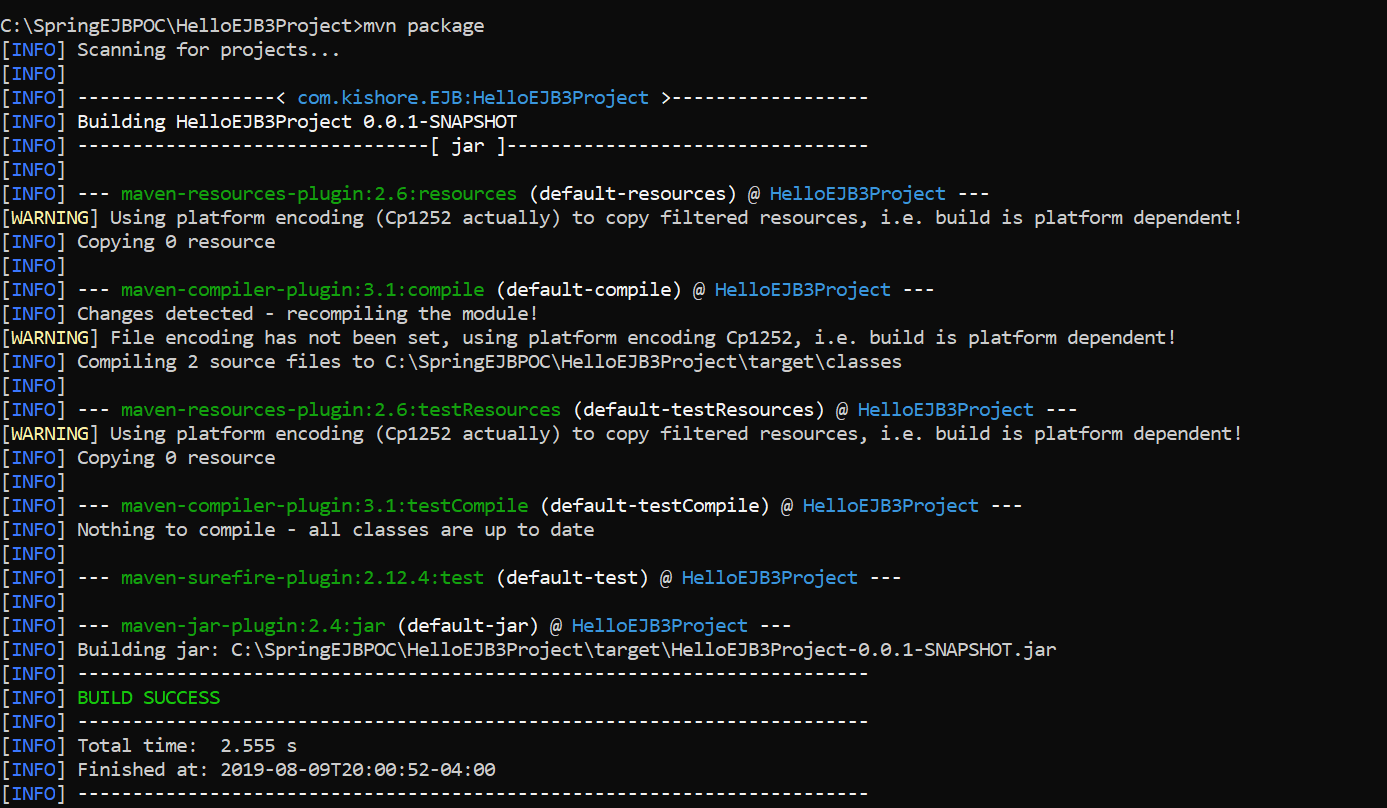
mvn clean



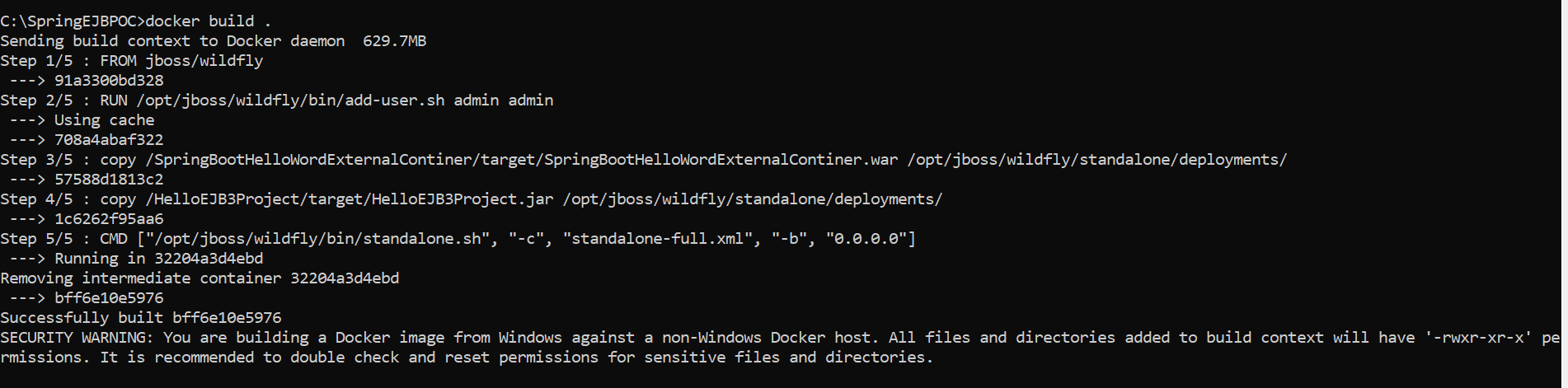


mvn package

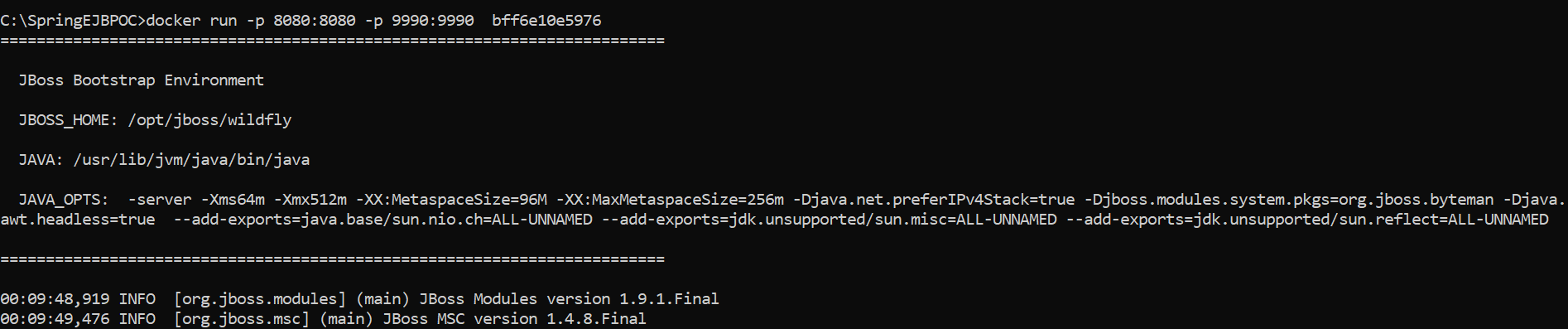




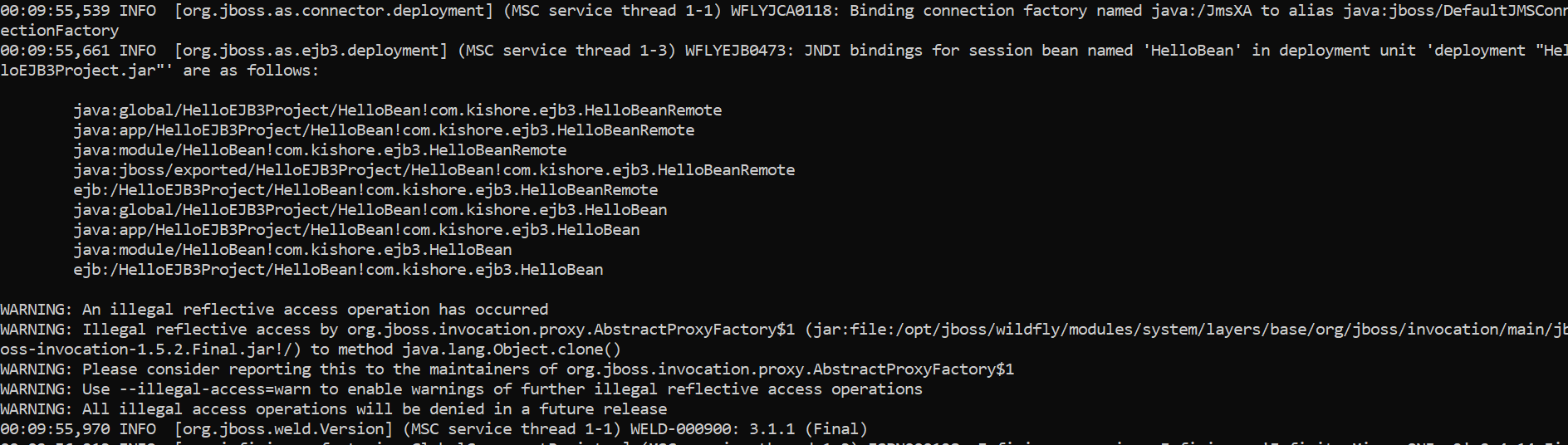
9. Run the docker build command. It will take couple of minutes to build the Image and run the image.



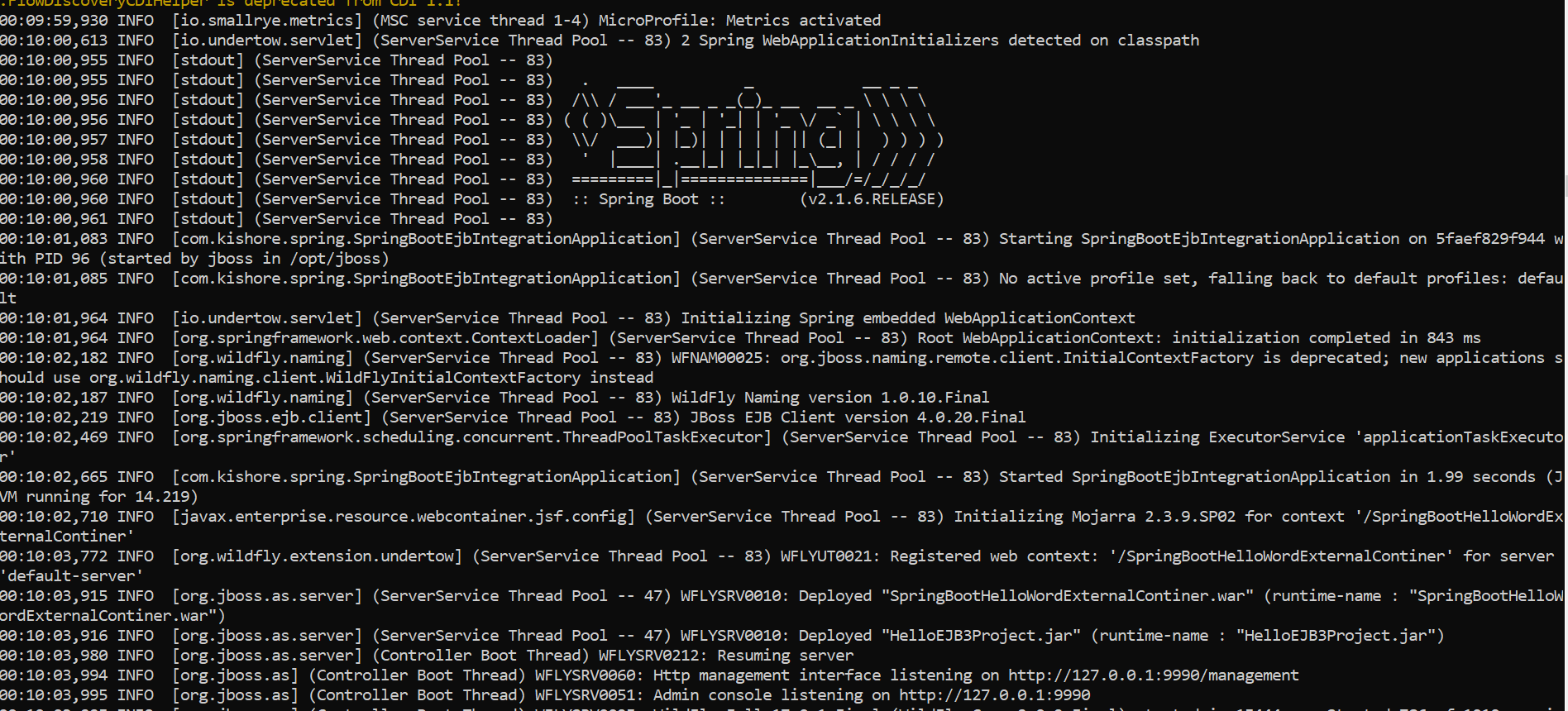
Expose the Port numbers 8080,9990



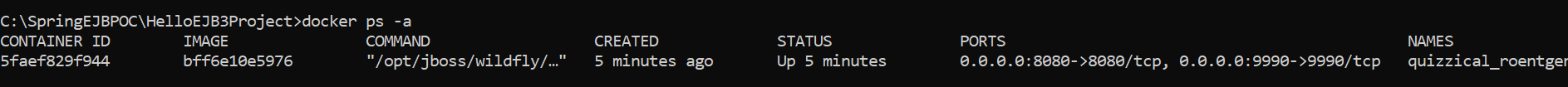
The EJB project is deployed successfully on wildfly continer



Spring boot container is up and running in wildfly



10. the docker containers are up and running



11. in windows 10 we have to use 127.0.0.1 instead of local host to connect service are running on docker container.

<http://127.0.0.1:8080/SpringBootHelloWordExternalContiner/helloword>

