

# Message Service Application Setup:


## Technology Stack used are :

- ✓ Jenkins <http://localhost:8080/>
- ✓ Active MQ (docker image) <http://localhost:8161/admin/queues.jsp>
- ✓ SpringBoot (version: 3.3.4)
- ✓ Maven(version: 3.9.3)
- ✓ GitHub ([https://github.com/StutiShrivastava/message\\_service/tree/master](https://github.com/StutiShrivastava/message_service/tree/master))
- ✓ DockerHub Application Image  
(<https://hub.docker.com/repository/docker/stutishrivastava/msgvalidation/gener-al>)

## 1. Active MQ Local setup using Docker Image :

- Cmd → docker pull jenkins/jenkins:lts
- Cmd → docker run -d --name jenkins -p 8282:8080 -p 50000:50000 -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts  
Url to access Active MQ(<http://localhost:8161/>)

## 2. Jenkins Local setup using War file:

- Download the latest Jenkins stable version war file and install it locally  
  
jenkins.war
- Navigate to the Jenkins folder and run it  
-→ Java -jar Jenkins.war
- While running the Jenkins as normal user I got certificate issues while installing the plugins. The issue was solved following the below approach:
  - ✓ There are various urls from where the plugins installations are downloaded, the issue arises when Jenkins server is not able to trust the site and hence cannot download the plugins
  - ✓ To Solve the issue we should download the certificate of site and put it in keystore, Steps to do that are as follows:
    - Step 1: Download the Certificate Manually  
Open a Web Browser:  
Navigate to URLs  
Click on the padlock icon in the address bar (this may vary based on your browser).

And download the certificate

- Step 2: Add  
keytool -importcert -file  
C:\Users\A5143522\Stuti\jenkins\_home\ftp.halifax.rwth-  
aachen.de.crt -keystore  
"C:\Users\A5143522\Java\openjdk-  
17.0.12\lib\security\cacerts" -alias halifax-cert
- Certificate of following urls were added manually:
  - ✓ <https://updates.jenkins.io>.
  - ✓ [https://ftp.halifax.rwth-aachen.de/jenkins/plugins/script-security/1366.vd44b\\_49a\\_5c85c/](https://ftp.halifax.rwth-aachen.de/jenkins/plugins/script-security/1366.vd44b_49a_5c85c/)
  - ✓ <https://ftp.belnet.be/mirror/jenkins/plugins/docker-commons/>

- While running the Jenkins using admin user, there was no certificate issue.
- Proceed with the required plugins installation which are sufficient for pipeline creation
- If your application is using Docker images then you need to install Docker plugin as well in the Jenkins server.
- You can view all the installed plugins and tools in the Jenkins Dashboard → Manage Jenkins.
- Jenkins Server is available on <http://localhost:8080/>
- User: Jenkins Password: Jenkins

### **3. Jenkins Local setup using Docker Image (Optional):**

- docker pull jenkins/jenkins:lts
- docker run -d --name jenkins -p 8282:8080 -p 50000:50000 -v jenkins\_home:/var/jenkins\_home jenkins/jenkins:lts
- docker logs jenkins // to get the password

Refer command.txt for more details



command.txt

#### 4. **Spring Boot Application :**

Create a Spring Boot application with all the starter dependencies from the <https://start.spring.io/>

Add all the starter dependencies and write your application classes and business logic.

#### 5. **Application Containerization**

- To create an image of your application

**Step 1:** For creating a jar with specific name

>> Add final name in pom.xml

>> <finalName>< message-service-docker></finalName>

>> now do mvn install

This will create jar in the target folder of source code.

**Step 2:** Docker file with name Dockerfile

Create a Dockerfile, it is a text file with a set of instructions used to build a Docker image. Each instruction in a Dockerfile describes a step for creating the image, such as defining the base image, installing packages, copying files, setting environment variables, and specifying the command that runs when the container starts.

**Step 3:** Building the Image

We can build the image manually by running the build command as follows:

run the docker desktop and following command

>> navigate to application folder

**Cmd** >> docker build -t message-service:1.0 .

this will build the docker image ,

we can view all the images

**Cmd** >> docker images

We can also use Docker compose file to build our images. Docker Compose file is more useful if we are using various services(images) in our application , it will bind them together in the same network and hence they can interact with one another.

**Step 4:** Running the image

If we have to run the individual image manually we can do that by

**Cmd** >> docker run -p 8181:8181 message-service:1.0

## Running the image by compose file

Cmd >> docker-compose up -d

### 6. Application image in Docker repository:

First login to Docker hub portal and create your own repository.

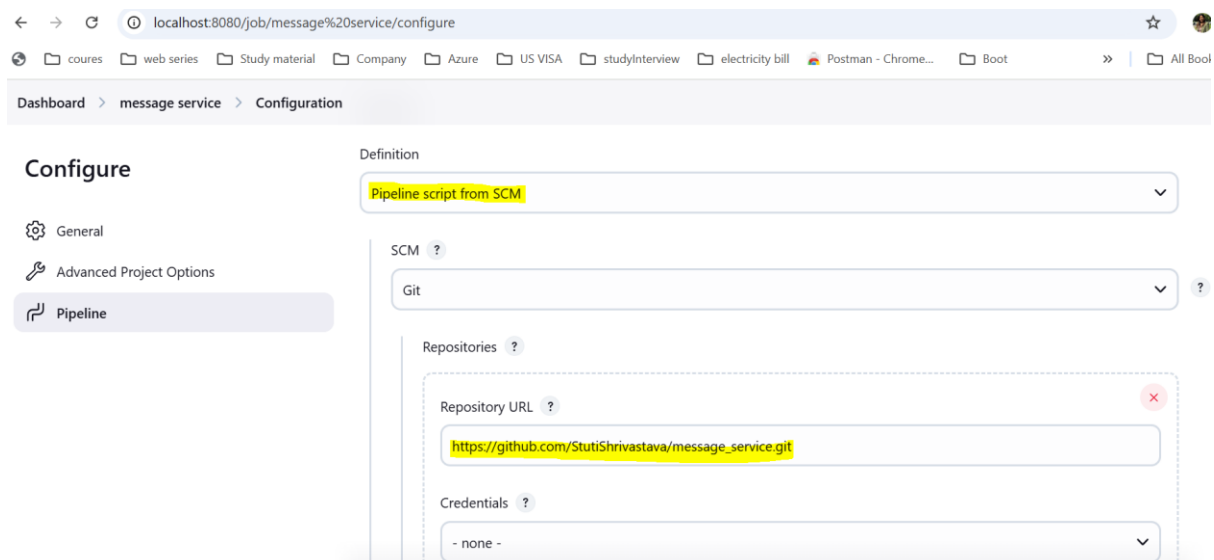
<https://hub.docker.com/>

Then open the command prompt with admin rights and do the following steps:

- ✓ docker login (Using the user and password of docker hub portal)
- ✓ docker tag message-service:1.0 stutishrivastava/msgvalidation-service:latest
- ✓ docker push stutishrivastava/msgvalidation-service:latest
- ✓ We can view the latest image  
<https://hub.docker.com/layers/stutishrivastava/msgvalidation-service/>

### 7. Running application via Jenkins build:

- ✓ **Step 1: Create a pipeline (simple pipeline or multibranch depending upon the requirement)**
- ✓ **Step 2: Configure the pipeline with the basic setup like repository url, branch to be scanned, the pipeline script to be enabled as shown in screenshots below**



Branches to build ?

Branch Specifier (blank for 'any') ?

\*/master

Add Branch

---

Dashboard > message service > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

(Auto)

Additional Behaviours

Add

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

**This is the Jenkins file name that needs to be present in our SCM for the Jenkins pipeline to trigger the build and perform the steps given the Jenkinsfile.**

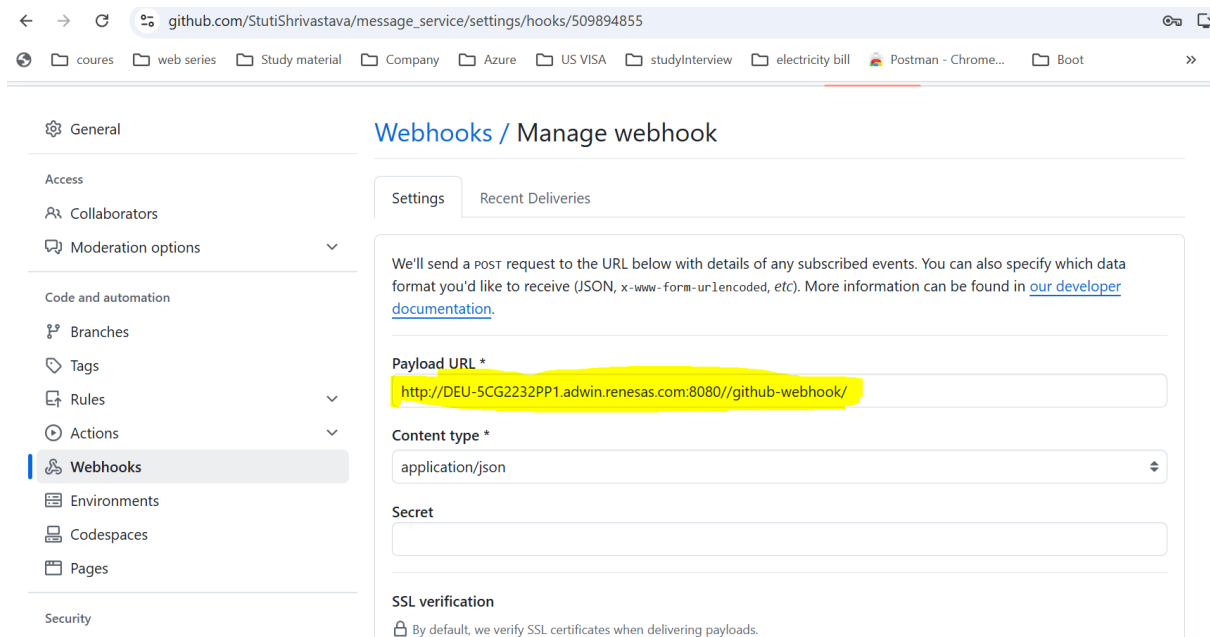
The Jenkinsfile has various stages and the steps that needs to be performed in those stages.

**Also we need to configure the webhook url in our Git repository**

Git hub Repository : [https://github.com/StutiShrivastava/message\\_service](https://github.com/StutiShrivastava/message_service)

**>> Go to Settings → webhook**

[https://github.com/StutiShrivastava/message\\_service/settings/hooks](https://github.com/StutiShrivastava/message_service/settings/hooks)



## 8. **How the application is getting build :**

Basically In the application there is **docker-compose.yml** file provided that build the services for the application

```
container_name: activemq  
image: rmohr/activemq
```

```
container_name: spring-app  
image: message-service:2.0
```

When we run the docker compose file basically these two images are build and run on the respective ports.

We are doing the same through Jenkins file as well, We are using the Docker-compose file to build and start our application.

The command to run build and run the application via docker-compose is  
>> sh/bat "docker-compose -f \${DOCKER\_COMPOSE\_FILE} up -d"

**Also the application image build can be pushed to the Docker Hub repository and can be published to any cloud cluster to deploy the application on cloud.**

## 9. How to run the application locally (Using Docker compose)

via docker (ensure you have docker installed in the system)

- ✓ To run the application locally you can use the docker-compose file provided in the source code.
- ✓ Move to the docker compose file folder and run the command:  
**“docker-compose up -d”**
- ✓ You will get the container id created and then you can view the container in the docker desktop or by using the command : docker ps



docker-compose.yml