**SOURCE CODE-Setting Up Jenkins Pipeline**

**1.Pipeline**

pipeline {

environment {

registry = "SrimathiElango21/docker\_automation"

registryCredential = 'dockerhub'

dockerImage = ''

}

agent any

stages {

stage('Cloning Git') {

steps {

git 'https://github.com/SrimathiElango21/Setting-Up-Jenkins-Pipeline-to-Deploy-Docker-Swarm'

}

}

stage('Building image') {

steps{

script {

dockerImage = docker.build registry + ":$BUILD\_NUMBER"

}

}

}

stage('Deploy Image') {

steps{

script {

docker.withRegistry( '', registryCredential ) {

dockerImage.push()

}

}

}

}

stage('Remove Unused docker image') {

steps{

sh "docker rmi $registry:$BUILD\_NUMBER"

}

}

}

}

**2.Pipeline explanation**:  
In this pipeline, we have 2 environment variables to change the registry and credentials:

environment {

registry = "SrimathiElango21/docker\_automation"

registryCredential = 'dockerhub'

dockerImage = ''

}

Jenkins will clone a git repository that has a Dockerfile inside

stage('Cloning Git') {

steps {

git 'https://github.com/SrimathiElango21/Setting-Up-Jenkins-Pipeline-to-Deploy-Docker-Swarm'

}

}

In the Docker\_Jenkins\_Pipeline repository on Github, the Dockerfile contains the following configurations:

# Selecting the base image to build our own customised node.js application microservice

FROM node

# Working directory inside the container

WORKDIR /usr/src/app

# Copying dependencies

COPY package\*.json ./

# Installing node package manager

RUN npm install

# Copying everything from current location to default location inside the container

COPY . .

# Expose the port

EXPOSE 3000

# Starting the app with CMD -

CMD ["node", "app.js"]

The following stage will build the image based on the Dockerfile detected in Git repository

stage('Building image') {

steps{

script {

dockerImage = docker.build registry + ":$BUILD\_NUMBER"

}

}

}

Once the Docker image has been created, it will be pushed to Dockerhub

stage('Deploy Image') {

steps{

script {

docker.withRegistry( '', registryCredential ) {

dockerImage.push()

}

}

}

}

Finally, we will remove the previously built image on the local server so that we do not accumulate all the images in our Docker localhost

stage('Remove Unused docker image') {

steps{

sh "docker rmi $registry:$BUILD\_NUMBER"

}

}