Project Title: Dockerizing Jenkins Pipeline

Note: This is a solution document on how the demonstration is performed on Docker 16.* version. Install the latest Docker version or >17.+ and Docker edge to allow multiple builds.

Create a Dockerfile and add the following content to it:

```
FROM jenkins/jenkins:lts
USER root
RUN apt-get update && \
apt-get -y install apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    software-properties-common && \
curl -fsSL https://download.docker.com/linux/$(. /etc/os-
release; echo "$ID")/gpg > /tmp/dkey; apt-key add /tmp/dkey && \
add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/$(.
/etc/os-release; echo "$ID") \
    $(lsb_release -cs) \
    stable" && \
apt-get update && \
apt-get -y install docker-ce
RUN apt-get install -y docker-ce
RUN usermod -a -G docker jenkins
USER jenkins
```

Now build the file:

“docker build -t ubuntu .”

Since Jenkins is already installed in your system, start the Jenkins and login. Click on **New Item** and select **Pipeline**. Add the job name as **DockerizeJenkins.**

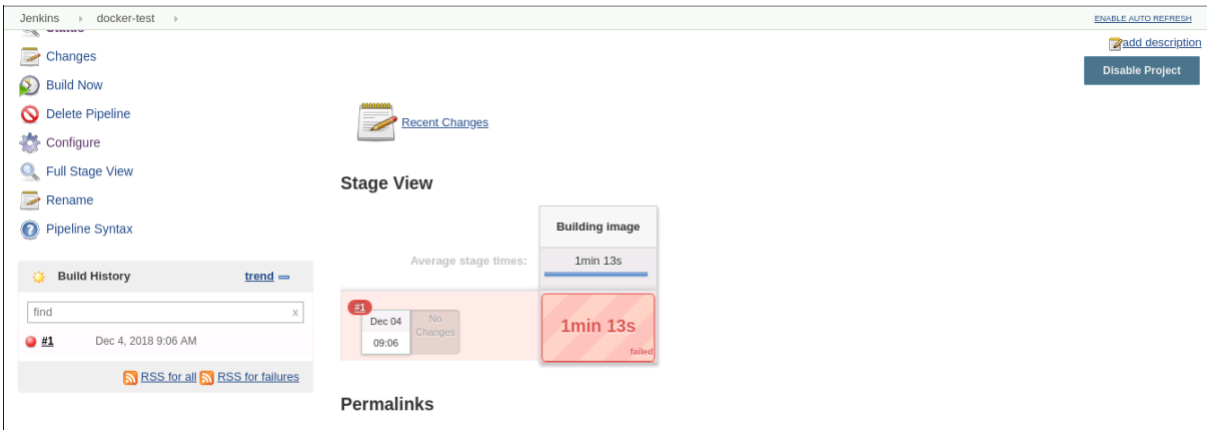
Add the script mentioned below:

```
pipeline {
  environment {
    registry = "docker_hub_account/repository_name"
    registryCredential = 'dockerhub'
  }
  agent any
```

```
  stages {
    stage('Building image') {
      steps{
        script {
          docker.build registry + ":$BUILD_NUMBER"
        }
      }
    }
  }
}
```

Save the job.

Click on **Build Now** in the job menu of Jenkins. The job will fail as the error is due missing the source of Docker Hub file as shown below:



Since you have already created a Docker Hub account, login and click on "**Create Repository**".

Go to Jenkins page and click on **Credentials → Global → Add Credentials**

Add your Docker Hub credentials and save it. ID and Description should be "**dockerhub".**

Also, a Git repository should be cloned in order to use a Node.js application as an example.

Add the image in Docker registry to pull in other machines.

Now change the script in the Jenkins Pipeline as given below. Replace **docker_hub_account** with your Docker Hub account name and **repository_name** with the repository you have created. The final script code will be:

```
pipeline {
  environment {
    registry = "docker_hub_account/repository_name"
    registryCredential = 'dockerhub'
    dockerImage = ''
```

```
  }
  agent any
  stages {
    stage('Cloning Git') {
      steps {
        git
'https://github.com/SimplilearnDevOpsOfficial/DockerizeJenkins.g
it '
      }
    }
    stage('Building image') {
      steps{
        script {
          dockerImage = docker.build registry + ":$BUILD_NUMBER"
        }
      }
    }
    stage('Deploy Image') {
      steps{
        script {
          docker.withRegistry( '', registryCredential ) {
            dockerImage.push()
          }
        }
      }
    }
  }
}
```

Complete the pipeline to the Node.js application:

Go to Manage Jenkins → Manage Plugins → Available. Then, search for **NodeJs**. Check the box and click on **Download now and install after restart.**

Now configure the Node.js tool.

Go to Jenkins home → Manage Jenkins → Global Tool Configuration and search for **NodeJs**.

Name it as **node** and select any 9+ version from the dropdown.

The final script code will be:

```
pipeline {
  environment {
    registry = "docker_hub_account/repository_name"
    registryCredential = 'dockerhub'
    dockerImage = ''
```

```
  }
  agent any
  tools {nodejs "node" }
  stages {
    stage('Cloning Git') {
      steps {
        git
'https://github.com/SimplilearnDevOpsOfficial/DockerizeJenkins.g
it '
      }
    }
    stage('Build') {
       steps {
         sh 'npm install'
         sh 'npm run bowerInstall'
       }
    }
    stage('Test') {
      steps {
        sh 'npm test'
      }
    }
    stage('Building image') {
      steps{
        script {
          dockerImage = docker.build registry + ":$BUILD_NUMBER"
        }
      }
    }
    stage('Deploy Image') {
      steps{
         script {
            docker.withRegistry( '', registryCredential ) {
            dockerImage.push()
          }
        }
      }
    }
  }
}
```

Before you build the application, provide the directory level access to Jenkins and Docker.

"sudo usermod -a -G docker jenkins"

Now click on **Build.** You should be able to view that Jenkins is working with Docker through different stages.