

DevOps

Product Name: Integration with Jenkins and Docker.

Platform: Node.js.

Phase: Deployment

Date: ☞ Nov 10, 2022

Creator : Fayaz Shaik

Requirements :

1. Jenkins Login credentials.
2. Docker hub Account

Setting up the Jenkins Job first

Add These Files to your VS-Code

- Create a file called `jenkinsfile` in your project and add the below code to the file.

```
1  /* Requires the Docker Pipeline plugin */
2  pipeline {
3      agent { docker { image 'node:16.17.1-alpine' } }
4      stages {
5          stage('build') {
6              steps {
7                  sh 'node --version'
8              }
9          }
10     }
11 }
```

- Create another file called `Dockerfile` in your project and add the below code to the file.

```
1  FROM node
2
3  WORKDIR /rbac
4
5  COPY package.json /rbac/
```

```

6
7  RUN npm cache clean --force
8
9  RUN npm install
10
11  COPY . /rbac/
12
13  EXPOSE 8081
14
15  CMD ["npm","start"]
16

```

Jenkins setup: CI/CD Pipeline for a NodeJS Application with Jenkins :

- First Login into the Jenkins application click on NEW ITEM and select a FREESTYLE project give the item name you want. (ex: your project name) now your job is created.
- Go to the configuration and select the source code manager as shown below.

The screenshot shows the Jenkins 'Configuration' page for a new job. On the left sidebar, 'Source Code Management' is selected. In the main area, 'Git' is chosen as the source code manager. Below this, the 'Repository URL' is set to 'https://github.com/fayaz-faiz/rbac.git'. The 'Credentials' dropdown shows 'fayaz/*****' with a red '2' next to it. There are buttons for '+ Add', 'Advanced...', 'Save', and 'Apply'.

1. In the Repository URL paste the GitHub link where your project is available
2. credentials you need to add the Jenkins credentials to get this you need to set up
The credentials here only admins can add these credentials. To add These follow the below steps.
 - Click on Dashboard and select manage Jenkins.
 - Click on Manage Credentials and click on the system then click on global.
 - Click on add credentials as shown below and click on create button.

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

***give your jenkins youser name**

☐ Treat username as secret ?

Password ?

Enter your Jenkins password here

ID ?

jenkinscredtails

Create

- After setting up Jenkins we need to configure NodeJS on Jenkins.

Install NodeJS plugin:

- Open Jenkins: **Manage Jenkins > Plugin Manager > Install NodeJS plugin.**
- Global Tool Configuration :
 - Open Jenkins: **Manage Jenkins > Global Tool Configuration > NodeJS**
 - Set your compatible node version. We can set multiple NodeJS versions for multiple applications. (here In this we are using 16.16.0 V)

Now go to the job configuration again and select :

Build Environment > Provide Node & npm bin/ folder to PATH

Select the Nodejs version that is compatible with your application.

Dashboard > nodejs > rbac >

Configuration

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

☐ Prepare SonarQube Scanner environment ?

☒ Provide Node & npm bin/ folder to PATH

NodeJS Installation

Specify needed nodejs installation where npm installed packages will be provided to the PATH

elegantproject

npmrc file

- use system default -

Cache location

Default (~/.npm or %APP_DATA%\npm-cache)

☐ Terminate a build if it's stuck

☐ With Ant ?

Build > Execute shell:

```
1  #!/bin/bash
2  echo "-----> Install node modules <-----"
3  npm install
```

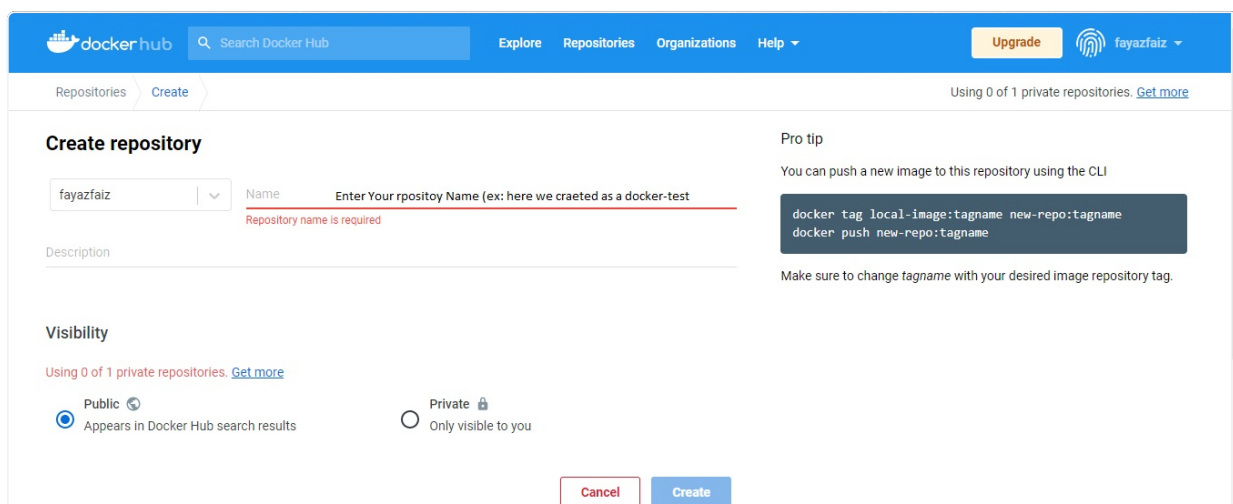
Apply and Save

*** Build Application with Jenkins Pipeline ***

Integrate your application with Docker To create a Docker Image.

Before we create a new build, we require **two** elements:

- Name of the Docker **Repository**
- Dockerhub **credentials** on Jenkins
- Create your Account by using this Link <https://hub.docker.com/>
- Open up a new tab and Login to your **Dockerhub** account. Click on **Create Repository**, enter a name and click **Create**. This will indicate to Jenkins **where** the Docker Image will be pushed.
-

The screenshot shows the Docker Hub 'Create repository' page. The top navigation bar includes the Docker Hub logo, a search bar, and links for Explore, Repositories, Organizations, and Help. A user profile for 'fayazfaiz' is visible in the top right. The main content area is titled 'Create repository' and features a 'Name' field with a dropdown menu showing 'fayazfaiz'. Below the name field, there is a red error message: 'Repository name is required'. To the right of the form, a 'Pro tip' section provides CLI commands for pushing a new image to the repository: 'docker tag local-image:tagname new-repo:tagname' and 'docker push new-repo:tagname'. Below the CLI commands, it says 'Make sure to change tagname with your desired image repository tag.' The 'Visibility' section shows two options: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Appears in Docker Hub search results'. The 'Private' option is described as 'Only visible to you'. At the bottom of the form, there are 'Cancel' and 'Create' buttons.

- In order for Jenkins to communicate with Dockerhub, we need to enter our credentials. Make sure the **Credentials** plugin is installed and navigate to **Manage Jenkins > Manage Credentials > System > Global Credentials > Add Credentials**.
- Enter your **Dockerhub credentials** including **username** and **password**. Assign it with an **ID** name called `dockerhub` or `dockerhubcredentials` and give it a memorable description.

The screenshot shows the Jenkins 'Global credentials (unrestricted)' configuration page. The 'Kind' is set to 'Username with password'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' is 'docker_username', the 'Password' is masked with '***', the 'ID' is 'dockerhubcredentials', and the 'Description' is 'DockerHub Credentials'. An 'OK' button is at the bottom.

- Once saved, navigate back to the home page and click on **New Item**.
- Enter your **project name** and select **Pipeline** and click **OK**:
- In the **General Section**, enter the same project URL as the previous Jenkins build we configured earlier.
-

The screenshot shows the Jenkins 'Configuration' page for a Pipeline project. The 'General' tab is selected. The 'Description' is 'docker-test'. There are three unchecked checkboxes: 'Discard old builds', 'Do not allow concurrent builds', and 'Do not allow the pipeline to resume if the controller restarts'. The 'GitHub project' checkbox is checked. The 'Project url' is 'https://github.com/fayaz-faiz/rbac.git/'. At the bottom, there are 'Save' and 'Apply' buttons.

- In the **Build Triggers** section, select **Build after other projects is built** and select the **name** of the Continuous Integration build we configured earlier. In my case, the build is called `rbac`. This build will be automatically triggered once the freestyle project has passed the tests successfully. Next, select **Trigger only if the build is stable**.

Dashboard > nodejs > docker-test >

Configuration

- General
- Advanced Project Options
- Pipeline

Build Triggers

☒ Build after other projects are built ?

Projects to watch

rbac,

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

☐ Always trigger, even if the build is aborted

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

☐ Quiet period ?

Save Apply

- In the **Pipeline** section, we will be creating a script to tell Jenkins to **create** a Docker Image and **deploy** it onto Dockerhub. The pipeline script is based on the **Apache Groovy** programming language.
-

Dashboard > nodejs > docker-test >

Configuration

- General
- Advanced Project Options
- Pipeline

Pipeline

Definition

Pipeline script

Script ?

```

1  > dockerImage = ''
2  }
3  agent any
4  stages {
5  > stage('Cloning Git') {
6  > steps {
7  > git 'https://github.com/fayaz-faiz/rbac.git'
8  > }
9  > }
10 > stage('Building image') {
11 > steps {
12 > script {
13 > dockerImage = docker.build registry + ":$BUILD_NUMBER"
14 > }
15 > }
16 > }
17 > stage('Deploy Image') {
18 > steps {
19 >
20 >
21 >
22 >

```

Save Apply

```

1 //Pipeline Script
2 pipeline {
3     environment {
4         registry = "fayazfaiz/docker-test" // This is docker
        repository name
5         registryCredential = 'dockerhubcredentials'
6         dockerImage = ''
7     }
8     agent any

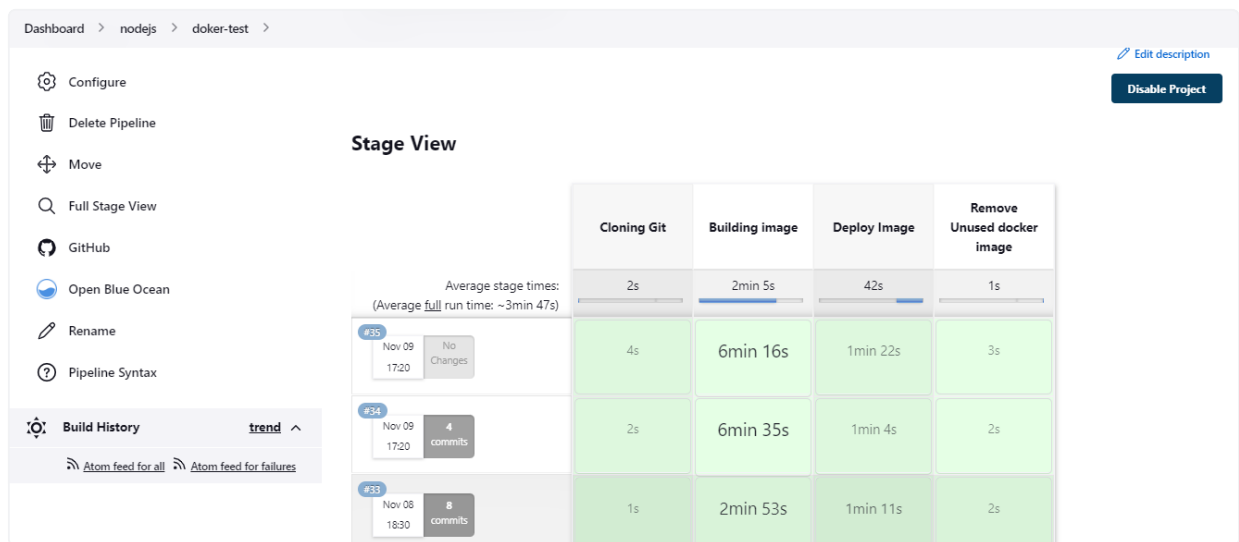
```

```

 9   stages {
10     stage('Cloning Git') {
11       steps {
12         git 'https://github.com/fayaz-faiz/rbac.git'
13       }
14     }
15     stage('Building image') {
16       steps{
17         script {
18           dockerImage = docker.build registry + ":$BUILD_NUMBER"
19         }
20       }
21     }
22     stage('Deploy Image') {
23       steps{
24         script {
25           docker.withRegistry( '', registryCredential ) {
26             dockerImage.push()
27           }
28         }
29       }
30     }
31     stage('Remove Unused docker image') {
32       steps{
33         sh "docker rmi $registry:$BUILD_NUMBER"
34       }
35     }
36   }
37 }

```

- Apply and save and Click on **BUILD NOW**



Build now if success

- If You got any errors go to the output see the error and fix that then come back to Jenkins and start Bui now again.