Practice for Lesson 7: Jenkins Integration with Docker

Practices for Lesson 7

Overview

In these practices, you will learn how to setup the environment for Docker container job creation on Jenkins instance. Further, create the Pipeline Job for Docker on Jenkins instance using a sample example and then deploying the Docker Container with the Jenkins Pipeline using the GitHub repository

Practice 7-2: Create a Pipeline Job for Docker on Jenkins

Overview

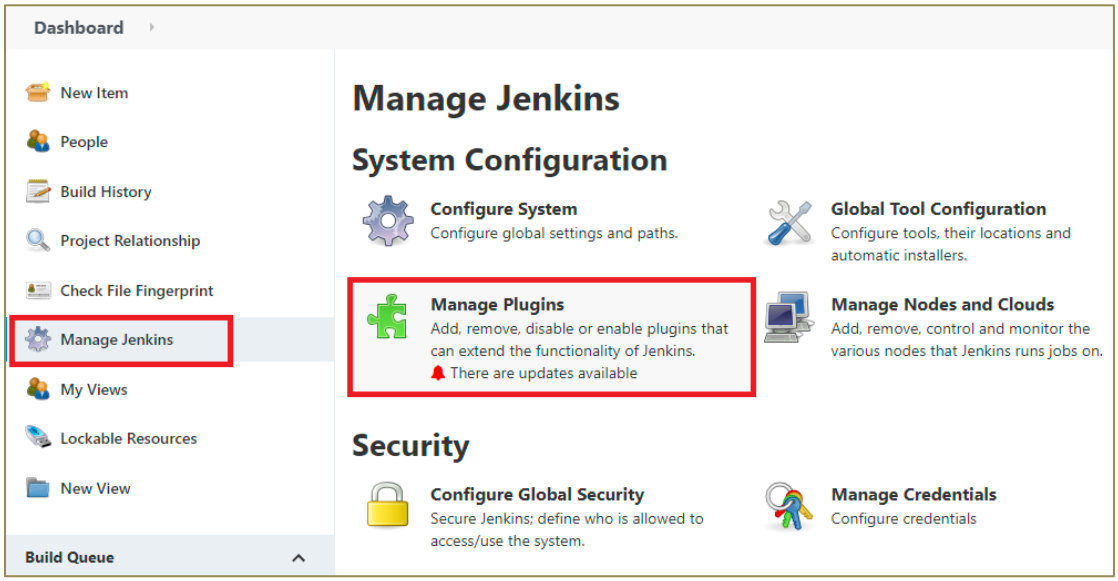
In this practice, you will learn how to create the Pipeline Job for Docker on Jenkins instance using a sample example.

Assumptions

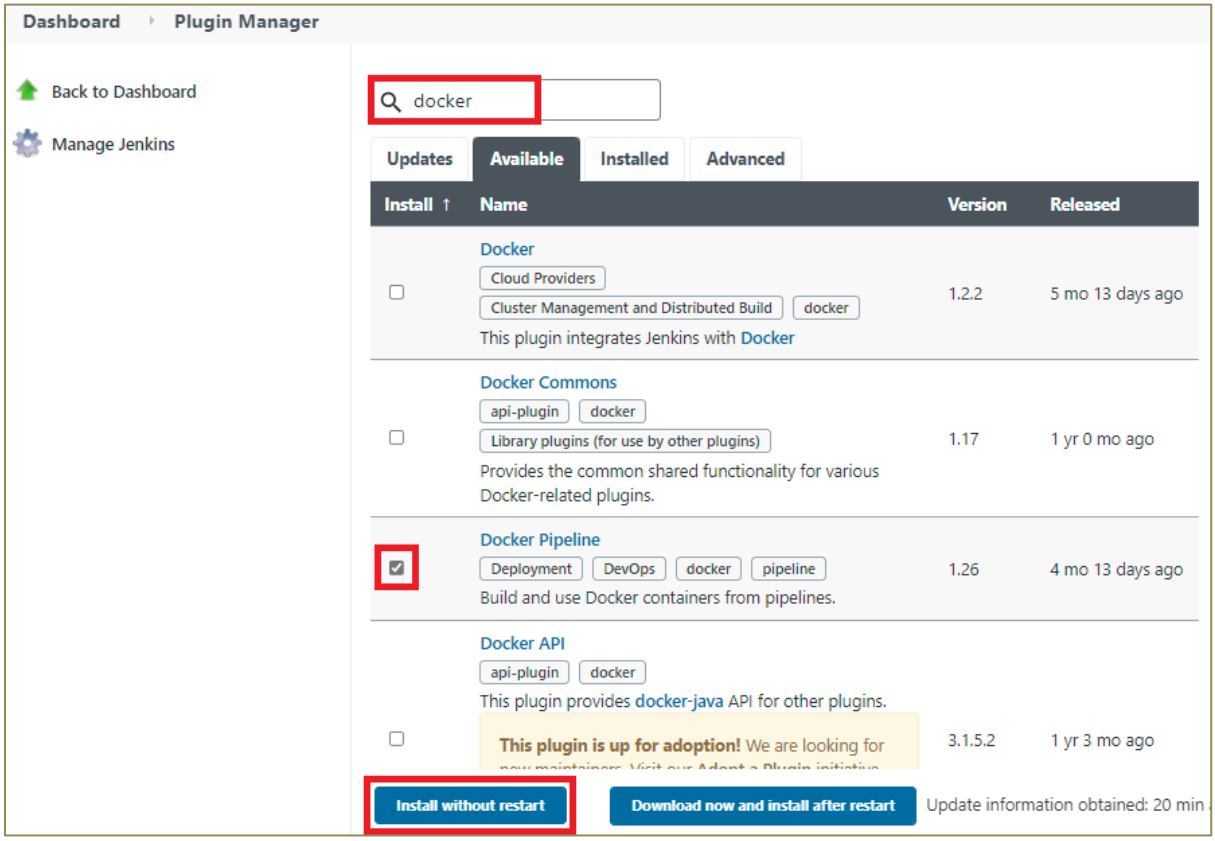
You should have completed the Practice of Lesson 7-1.

Tasks

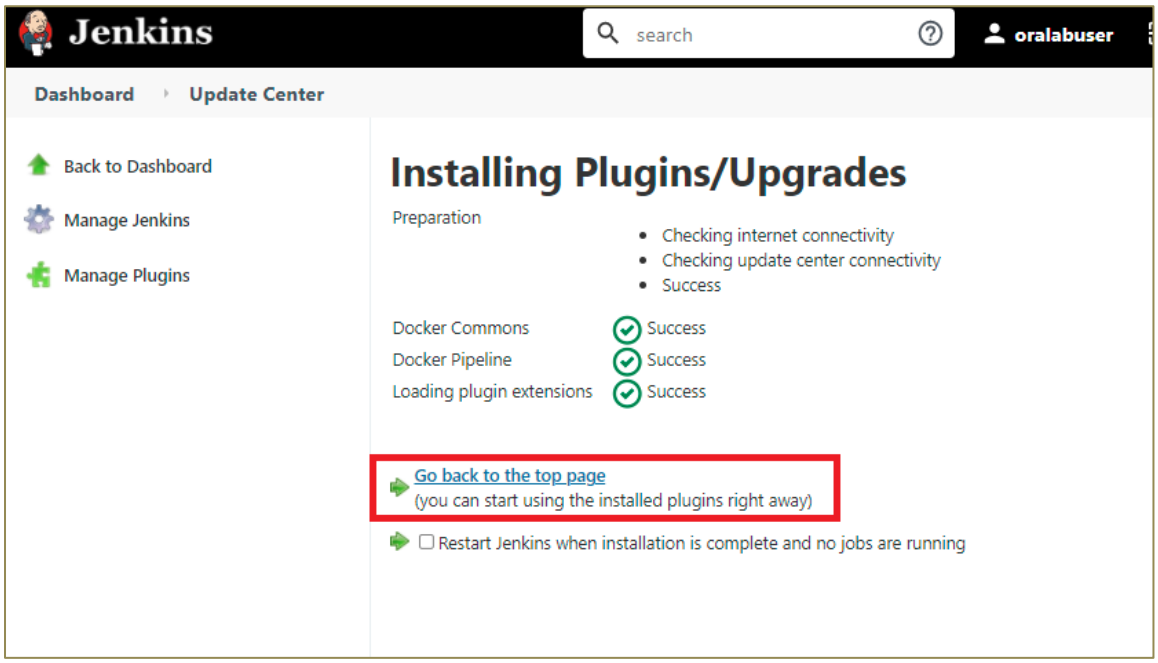
1. Install **Docker Pipeline** plugin on the **Linux instance**.
   a. Navigate to Dashboard, select **Manage Jenkins** and click **Manage Plugins** as shown below.



   b. Select **Available** and search for **Docker Pipeline plugin.** Select the check box of the **Docker Pipeline** plugin and click **Install without restart.**

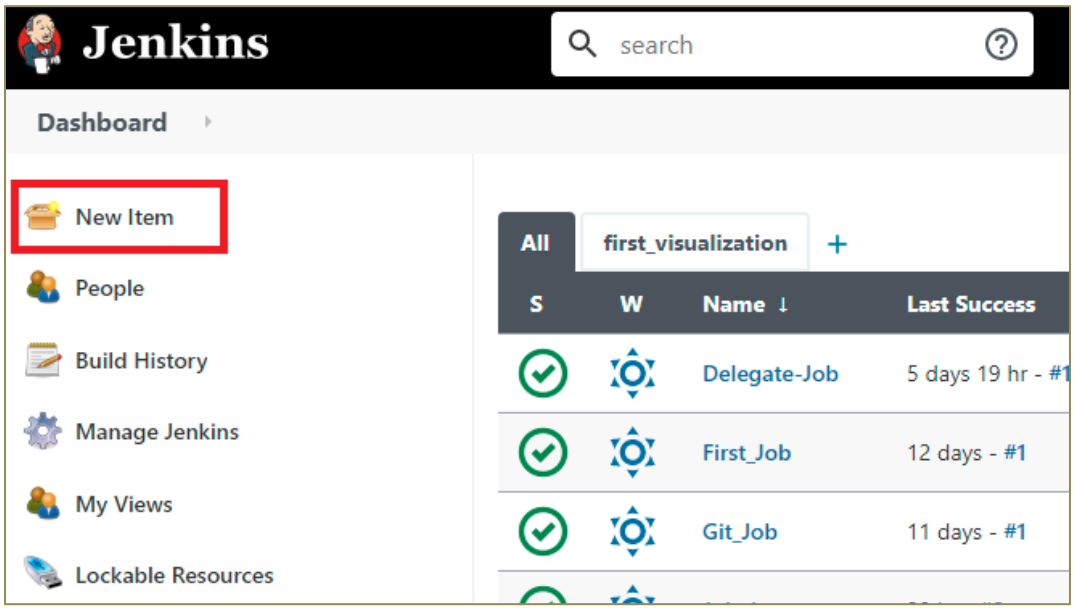


c. The installation process will proceed and Success message will be displayed as shown below.

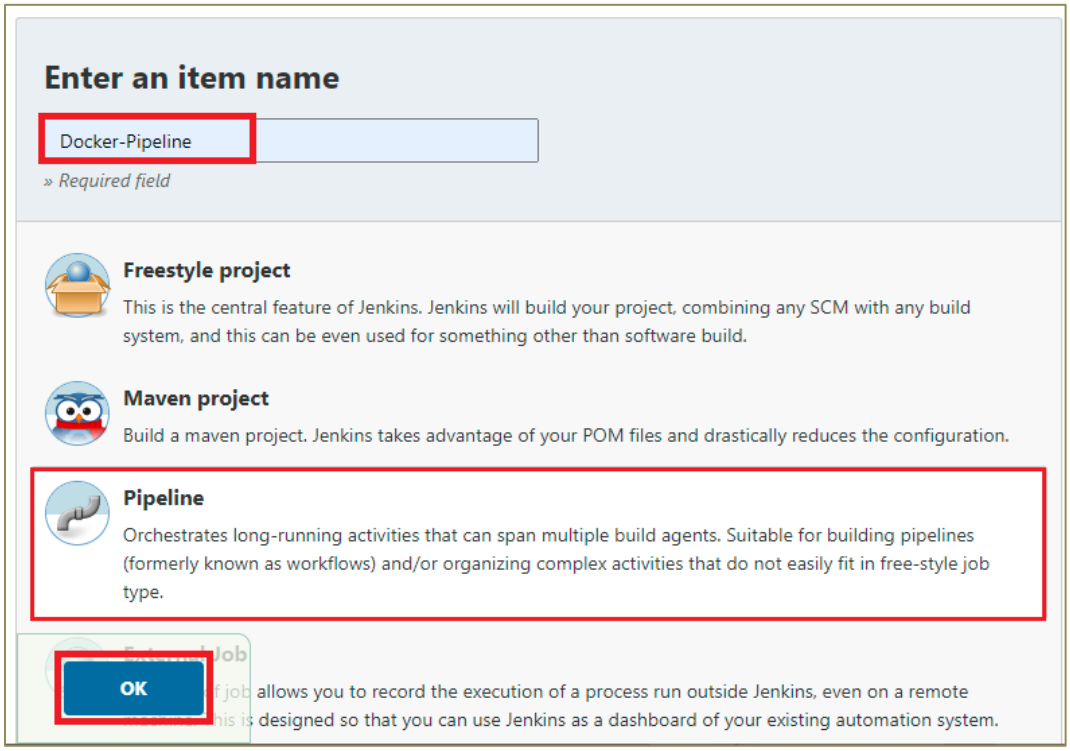


2. Create a Docker Pipeline job in the Jenkins instance.

a. In the Jenkins Dashboard, navigate to main menu and select **New Item** to create a **Docker** pipeline Job as shown below.

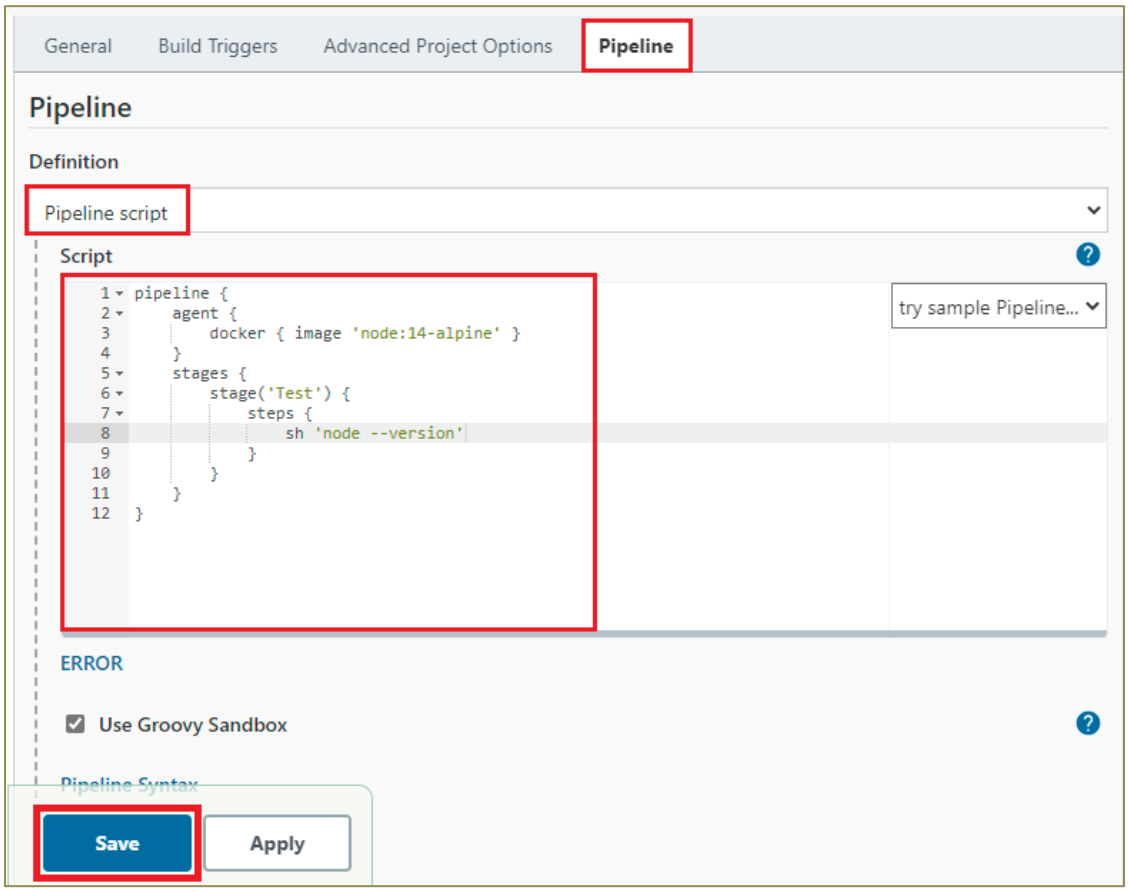


b. Provide the name for Job, select **pipeline** and click **OK.**

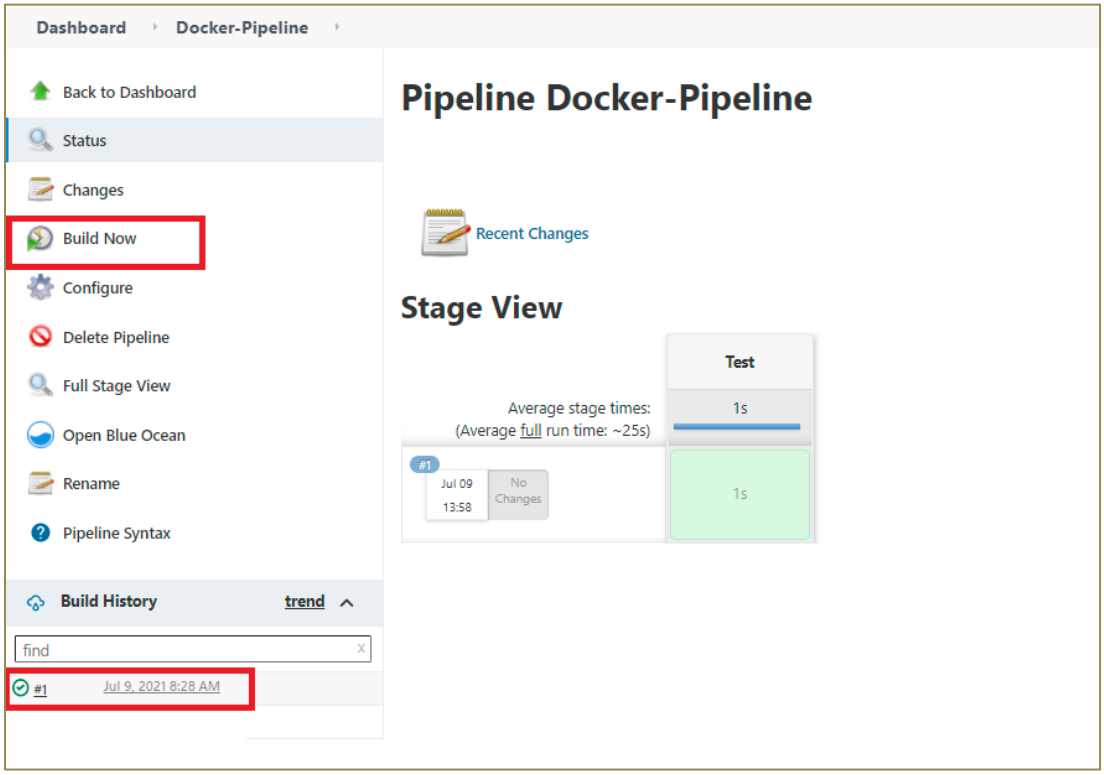


c. Navigate to **Pipeline** and select **Pipeline Script** under **Definition**. Copy the groovy script given below and paste it in the Jenkins Pipeline **Script** block. Click **Apply** and **Save.**

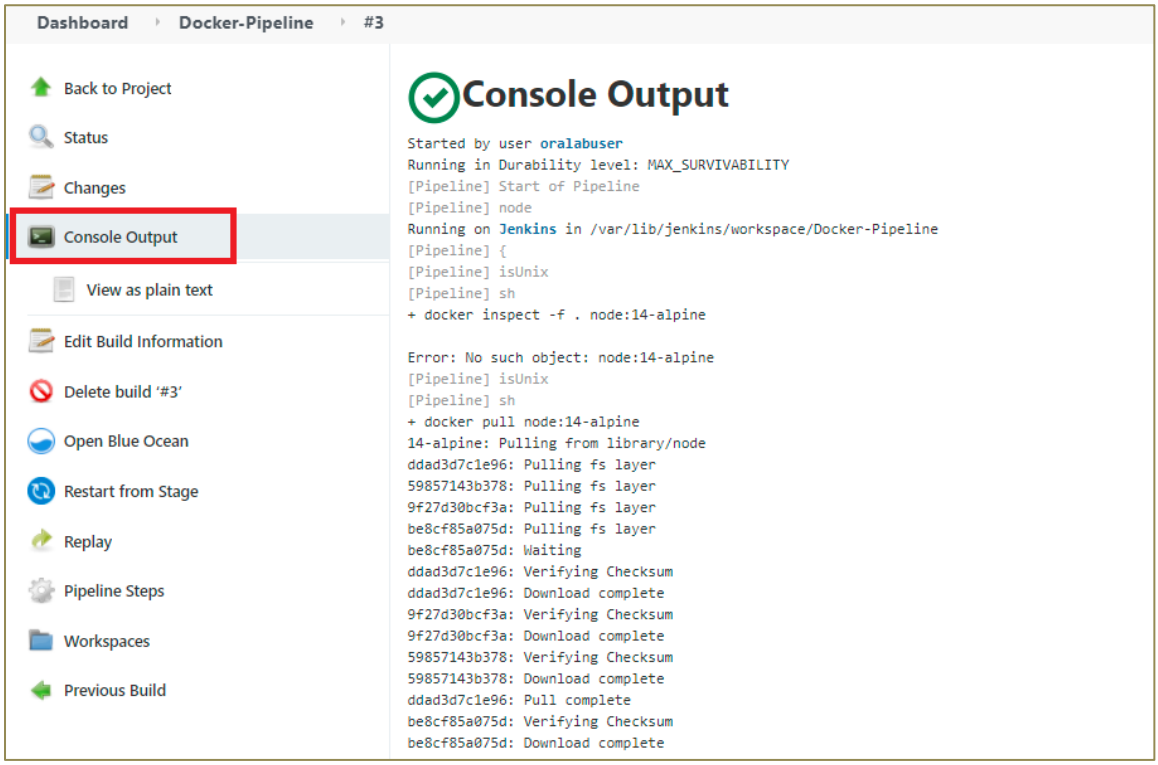
```
pipeline {
    agent {
        docker { image 'node:14-alpine' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
            }
        }
    }
}
```



d. Job is created successfully, click **Build Now** to execute the pipeline. Click on build created under **Build History** as shown below



e. In the **Build** page, click **Console Output** to view the output of the job. The given Docker image is pulled successfully and tested.



f. The Docker image **version** is also displayed and in the end **SUCCESS** message is displayed as shown below.

```
[Pipeline] withDockerContainer
Jenkins does not seem to be running inside a container
$ docker run -t -d -u 995:993 -w /var/lib/jenkins/workspace/Docker-Pipeline -v /var/lib/jenkins/workspace/Docker-
Pipeline:/var/lib/jenkins/workspace/Docker-Pipeline:rw,z -v /var/lib/jenkins/workspace/Docker-
Pipeline@tmp:/var/lib/jenkins/workspace/Docker-Pipeline@tmp:rw,z -e ******** -e ******** -e ******** -e ********
-e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e
******** -e ******** -e ******** -e ******** -e ******** node:14-alpine cat
$ docker top 99a83a08243e8a381af9da4065472a826d0a3c4fe0ffba016f275340918b9a59 -eo pid,comm
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh
+ node --version
v14.17.3
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
$ docker stop --time=1 99a83a08243e8a381af9da4065472a826d0a3c4fe0ffba016f275340918b9a59
$ docker rm -f 99a83a08243e8a381af9da4065472a826d0a3c4fe0ffba016f275340918b9a59
[Pipeline] // withDockerContainer
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

3. Keep the Jenkins Dashboard, terminal and the AWS Management Console open for the next practice.