NewgenONE OmniDocs RMS

Docker Containers Custom Code Deployment Guide for OpenShift

Version: 4.0 SP1

Newgen Software Technologies Ltd.

Table of contents

1 Preface

This guide describes the custom code deployment approach guide for container based Newgen's flagship products OmniDocs RMS on OpenShift Container Platform. This guide also describes the end-to-end implementation of the custom code deployment pipeline.

1.1 Revision history

Revision Date	Description
April 2024	Initial publication

1.2 Intended audience

This guide is intended for Cloud Administrators, System Administrators, developers, and all other users who are seeking information on the deployment of hotfix for container-based OmniDocs RMS. The reader must be comfortable understanding the computer terminology. The reader must have administrative rights for deployment and configuration.

1.3 Documentation feedback

To provide feedback or any improvement suggestions on technical documentation, you can write an email to docs.feedback@newgensoft.com.

To help capture your feedback effectively, requesting you to share the following information in your email.

- Document name
- Version
- Chapter, topic, or section
- Feedback or suggestions

1.4 Third-party information

This guide contains third-party product information about configuring OpenShift Container Platform and Jenkins CICD Pipeline for Container Deployment on OpenShift. Newgen Software Technologies Ltd does not claim any ownership of such third-party content. This information is shared in this guide only for the convenience of our users and could be an excerpt from the OpenShift documentation. For the latest information on configuring the OpenShift Container Platform and Jenkins CICD Pipeline refer to the respective official documentation.

2 CI/CD pipeline

The CICD pipeline manages custom code deployments of Newgen products on OpenShift Container Platform. Here, the Build Pipeline and Release Pipeline both are managed by the Jenkins server that can be installed on an on-premises machine or a bastion server.

2.1 CI/CD Pipeline for custom code

This section described the CI/CD Pipeline for Custom Code.

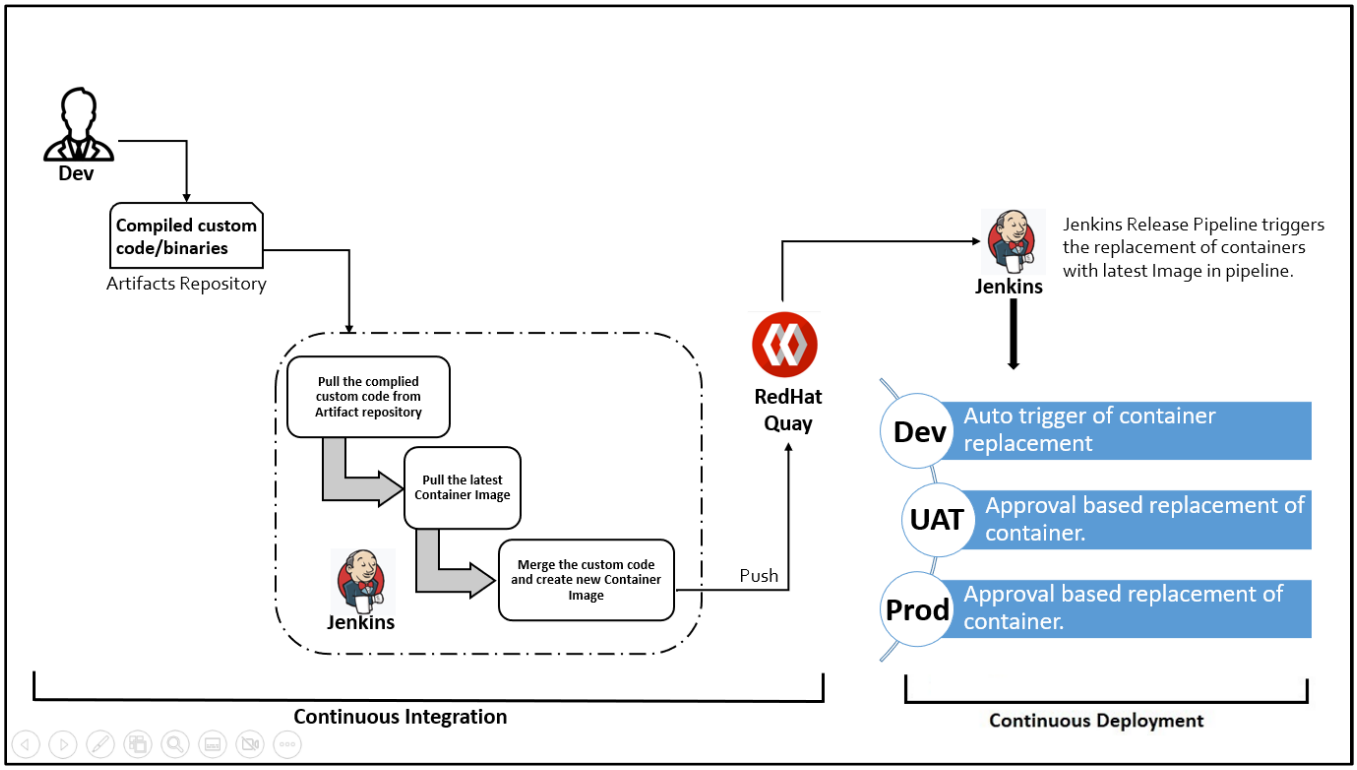

Figure 2.1

- For initiating custom code deployment, the Implementation team pushes the compiled custom code to the artifacts repository. An artifacts repository can be SVN, FTP, a Network drive, and so on.
- After that Jenkins pulls the compiled custom code from the artifacts repository and pulls the latest container image. Then, it creates a new container image after merging the custom code changes and pushes the newly created container images to the RedHat Quay Registry.
- In this architecture, the operation team have full access to initiate the build pipeline configured on the Jenkins server. The Implementation team does not have access to this Jenkins server. However, a common artifacts repository is shared for both teams.

- As soon as any container image is pushed to the RedHat Quay Registry, Jenkins Release Pipeline triggers the deployment to the Dev environment.
- UAT and Production deployments are based on approval and are available on-demand. To deploy to the UAT/Production environment, you need to trigger the UAT/Production deployment. Upon deployment trigger, an approval mail is sent to the project manager or the concerned team. As soon as the project manager or concerned team approves the go-ahead, UAT/Production deployment gets started.

3 Implementation of custom code deployment pipeline

The custom code deployment pipeline is separated into two parts: Build Pipeline (aka Continuous Integration) and Release Pipeline (aka Continuous Deployment). Build Pipeline and Release Pipeline both are configured through the Jenkins server.

For configuration of Jenkins Release Pipeline, refer the *OmniDocs RMS 4.0 SP1 Configuration and Deployment Guide for OpenShift"* document.

3.1 Approach guide for build pipeline

Perform the below steps to build pipeline:

1. There is a pre-defined folder structure for custom code. Only in that folder structure, the Implementation team pushes their custom code. However, all types of possible custom codes in OmniDocs are covered.

NOTE:

For container-based deployment, all custom classes in iForm or any other product module must implement a Serializable interface. It is required for session replication using Redis cache. If Redis cache is being used.

2. It covers 3 different types of custom codes in OmniDocs RMS.
   For example,
   - OmniDocs_CustomCode
   - omniDocs_Hook
   - WebAPI_CustomCode
3. This folder structure is available in the artifacts repository. Artifacts repository can be like AWS S3 Bucket, SVN, FTP, JFrog, and so on. It contains the AWS S3 bucket implementation.
4. Only this folder structure defined in the artifacts repository is accessible from the Implementation team.
5. Jenkins Build Pipeline have the following **5 jobs**:
   - Pull the compiled custom code from the artifacts repository.
   - Pull the latest Docker Image from the container registry in which custom code needs to deploy.
   - Merge custom code changes.

- Build a new Docker image.
- Push the newly created Docker image to the container registry.
  For example,

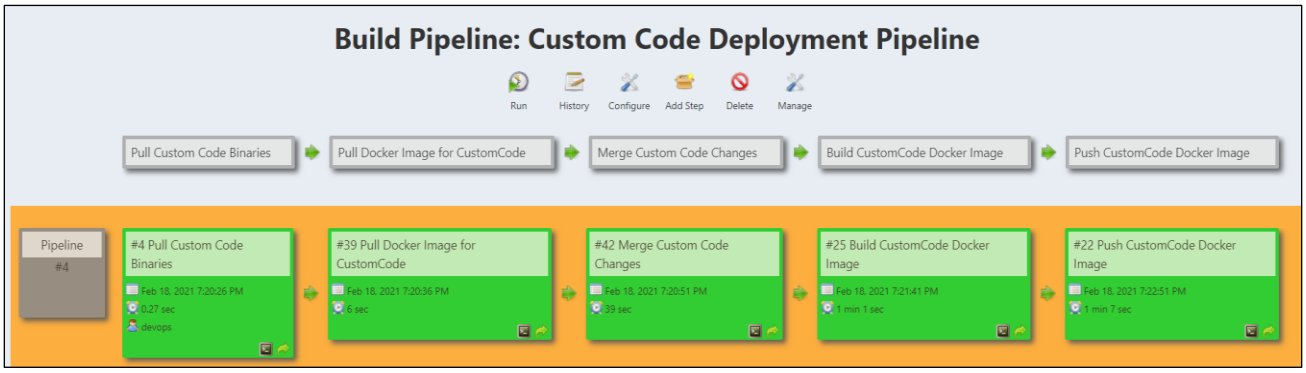

Figure 3.1

6. After pulling the latest custom code from the artifacts repository, Jenkins reads the *UserInput.properties* file.
7. This properties file contains all the user inputs that are required for condition-based custom code deployment.
8. This property file has multiple sections.
   - **#Container Registry Info**
     This section contains the container registry information. Here you need to provide **Quay_Server details** of the RedHat Quay registry. **Quay_User** and **Quay_Password** are used as encrypted environment variables in Jenkins jobs.
     For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Container Registery Info
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Quay_Server=quay.io/newgen
#Quay_User= Used as an encrypted environment variable in Jenkins Jobs
#Quay_Password= Used as an encrypted environment variable in Jenkins Jobs
```

Figure 3.2

   - **#Custom code changes to be deployed**
     In this section, you need to select the components for which you want to deploy the custom code. You can select or deselect the components by setting the component's value as Y/N respectively.
     For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Custom code changes to be deployed
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
OmniDocs_CustomCode=Y
omniDocs_Hook=Y
WebAPI_CustomCode=Y
```

Figure 3.3

- **#Custom code can be deployed to the following Container Images**
  This section just contains the information about the components and their destination Docker images. One component can be deployed to one or more Docker containers. So, we can decide in which container we want to deploy OmniDocs custom code changes.
  For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Custom code can be deployed to the following Docker Images
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#OmniDocs_CustomCode              OmniDocs_WEB
#omniDocs_Hook                    OmniDocs_EJB
#WebAPI_CustomCode                OmniDocs_WEB
```

Figure 3.4

- **#Container Image to be updated**
  In this section, we need to select the Docker image(s) in which we want to deploy custom code changes.
  For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Docker Image to be updated
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
OmniDocs_WEB=Y
OmniDocs_EJB=Y
```

Figure 3.5

- **#Docker Image Info**
  This section contains the information about the source Docker images in which custom code changes is merged or deployed.
  For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Docker Image Info
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

OmniDocs_WEB_ImageName=omnidocs11.0web
OmniDocs_WEB_Imagetag=base

OmniDocs_EJB_ImageName=omnidocs11.0ejb
OmniDocs_EJB_Imagetag=base
```

Figure 3.6

- **#New Docker Image Info with Custom Code changes**
  This section contains the information about new Docker images that are going to be created after merging the custom code changes.
  For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#New Docker Image Info with Custom Code changes
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Custom_OmniDocs_WEB_ImageName=omnidocs11.0web
Custom_OmniDocs_WEB_Imagetag=custom

Custom_OmniDocs_EJB_ImageName=omnidocs11.0ejb
Custom_OmniDocs_EJB_Imagetag=custom
```

Figure 3.7

- **#Other user Inputs**
  This section contains other information that can be used in the Jenkins pipeline.
  For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Other user Inputs
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
JAVA_HOME=C:\Program Files\Java\jdk1.8.0_91
```

Figure 3.8

9. Based on the input provided in the **UserInput.properties** file, Jenkins pulls the Docker images, merges the custom code changes, builds the new Docker images, and pushes Docker images to the container registry.
10. In the case of few components, we may require to deploy .JAR file to the EJB components Docker containers like OmniDocs EJB container and this container is running on the underlying AppServer

JBoss EAP. As we know that in the case of JBoss EAP if want to deploy any dependent library then we must make an entry in the module.xml file. Therefore, this build pipeline also handles this case.

11. To handle the above module.xml case, the Implementation team provides a **module.txt** file that contains the name of the new JAR file to deploy it as a dependent library. Rest Build pipeline manages the updates of the **module.xml** file inside the containers.
    For example:

```
<!--    Module.txt  -->
<resource-root path="NewJarFile.jar"/>
```

Figure 3.9

3.2 Configuration Jenkins jobs for the build pipeline

This section describes how to configure Jenkins for Build Pipeline.

3.2.1 Prerequisites

Following are the prerequisites:

- **Operating System**: Windows Server 2019 or above (Edition: Standard or Data Center).
- **Java** 1.11 update 18 and above.
- **Docker Engine** 20.10.10 or later version must be installed.
- **OpenShift CLI** 4.12.8 or a later version compatible with OpenShift cluster.
- **Cygwin** utility must be installed. [This utility is used to execute the Linux commands on Windows].
- **Jenkins** 2.246.0 or a later version must be installed with default suggested plug-ins along with the following plug-ins.
  - Conditional Build Step
  - Credentials Binding
  - Environment Injector

3.2.2 Configuration of Jenkins jobs

For the custom code deployment pipeline, Jenkins have the following 5 Jobs to do:

- Pull the compiled custom code from the artifacts repository.
- Pull the latest container Image from the container registry in which custom code needs to deploy.

- Merge custom code changes.
- Build a new container image.
- Push the newly created container image to the container registry.

Before creating any job, perform the following server-level configurations in the Jenkins.

1. Sign into the **Jenkins Server**.

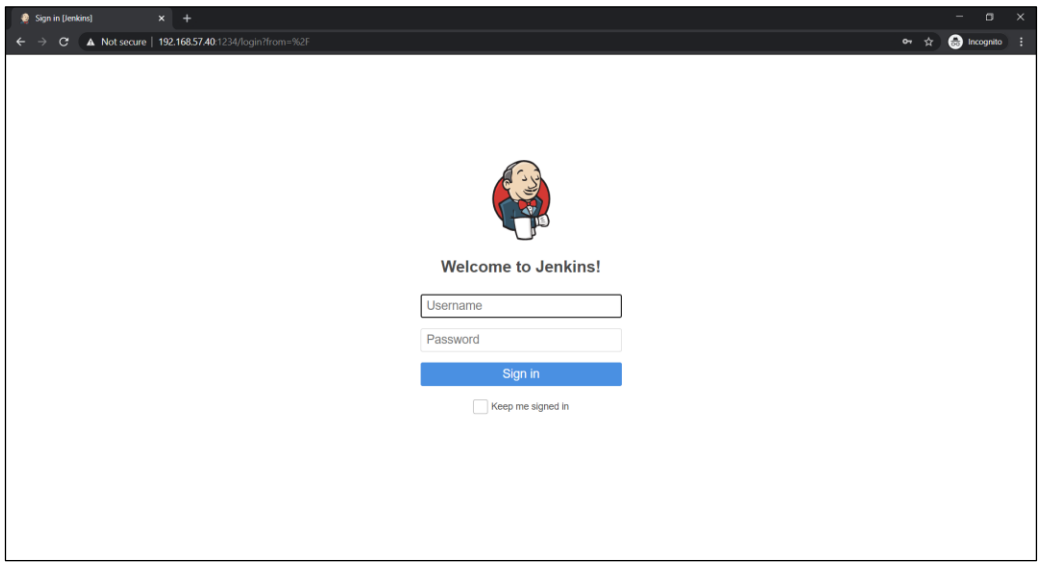

Figure 3.10

2. After the successful login, click **Manage Jenkins** link showing on the left panel.

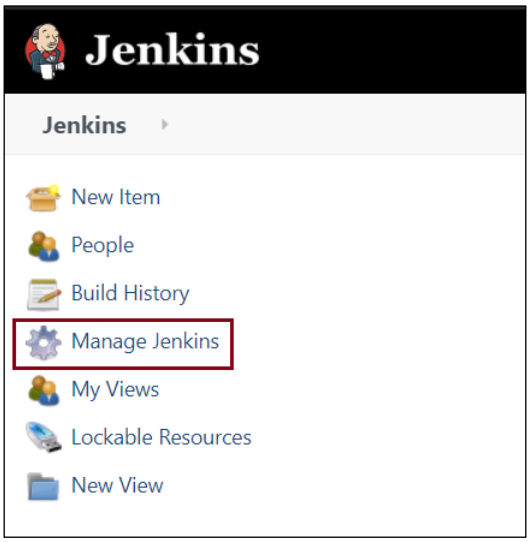

Figure 3.11

3. Click **Configure System** in the **System Configuration** section.

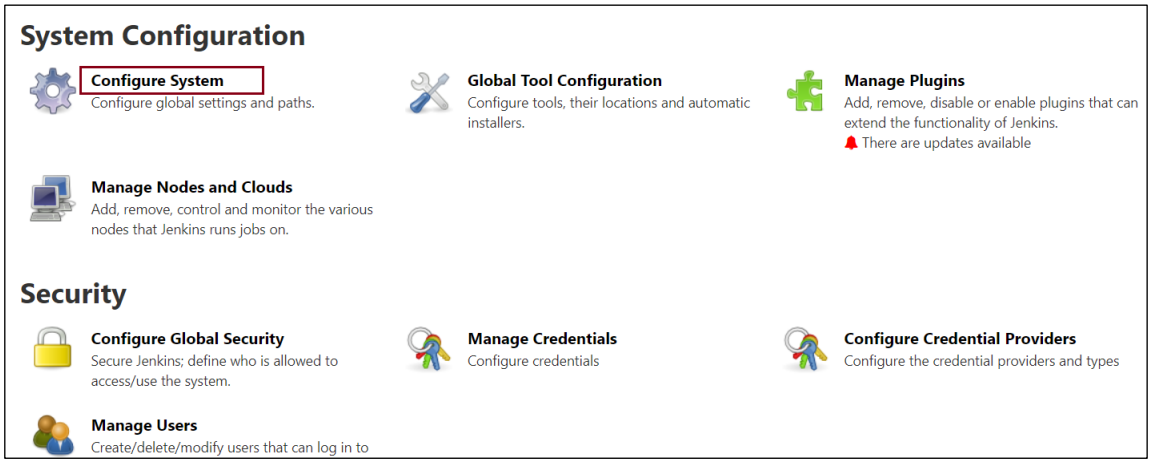

Figure 3.12

4. Under the **Global properties**, define an environment variable **PATH** with the following values separated with a semi-colon:
   - Docker installation path e.g., C:\Program Files\Docker\Docker\resources\bin
   - Cygwin installation path e.g., C:\cygwin64\bin
   - OpenShift CLI installation path e.g., C:\Software\utilties\oc-4.12.8-windows
   - Windows System32 path [C:\Windows\System32]

     For example,

```
PATH=C:\Program Files\Docker\Docker\resources\bin;C:\cygwin64\bin;C:\Software\utilties\oc-4.12.8-windows;C:\Windows\System32
```

Figure 3.13

5. Click **Save** to save the changes.

3.2.2.1 Pull custom code binaries

Perform the below steps to pull custom code binaries:

1. Click **New Item** link given on the left panel.

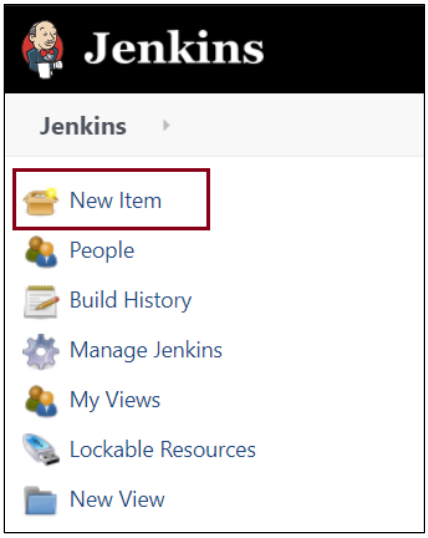

Figure 3.14

2. Specify the item name or job name and select the project type as **Freestyle project**.

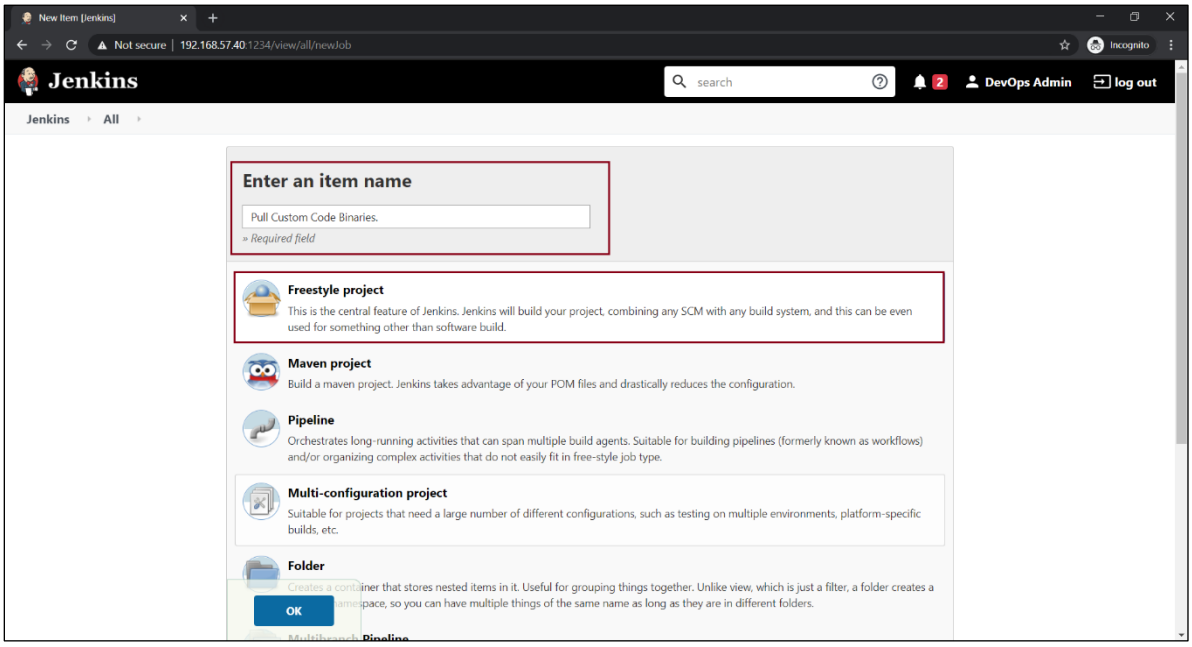

Figure 3.15

3. Specify the project description.

4. Under the **General**, click **Advanced** and specify the **Use custom workspace**.

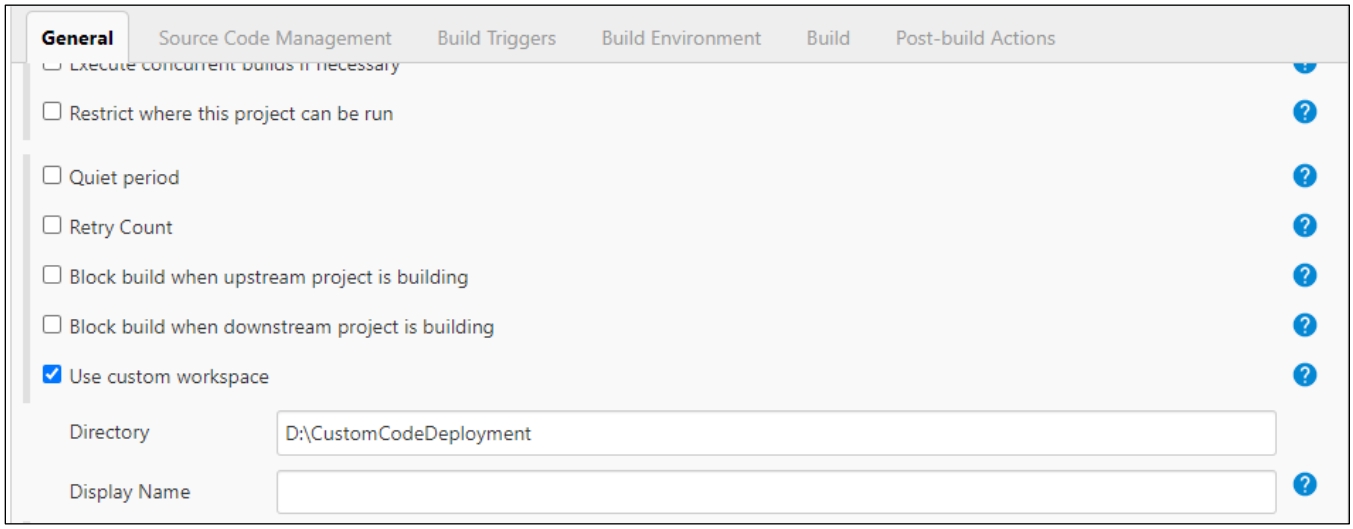

Figure 3.16

Custom workspace is the parent folder for your custom code deployment architecture.

5. Select the **Subversion** radio button under the Source Code Management.
6. Specify the **Repository URL** where your custom code folder structure resides.
7. Specify the **SVN credentials**.
8. Specify the **Local module repository** for checking out your custom code. As defined in step 4, local module repository gets appended to the custom workspace.

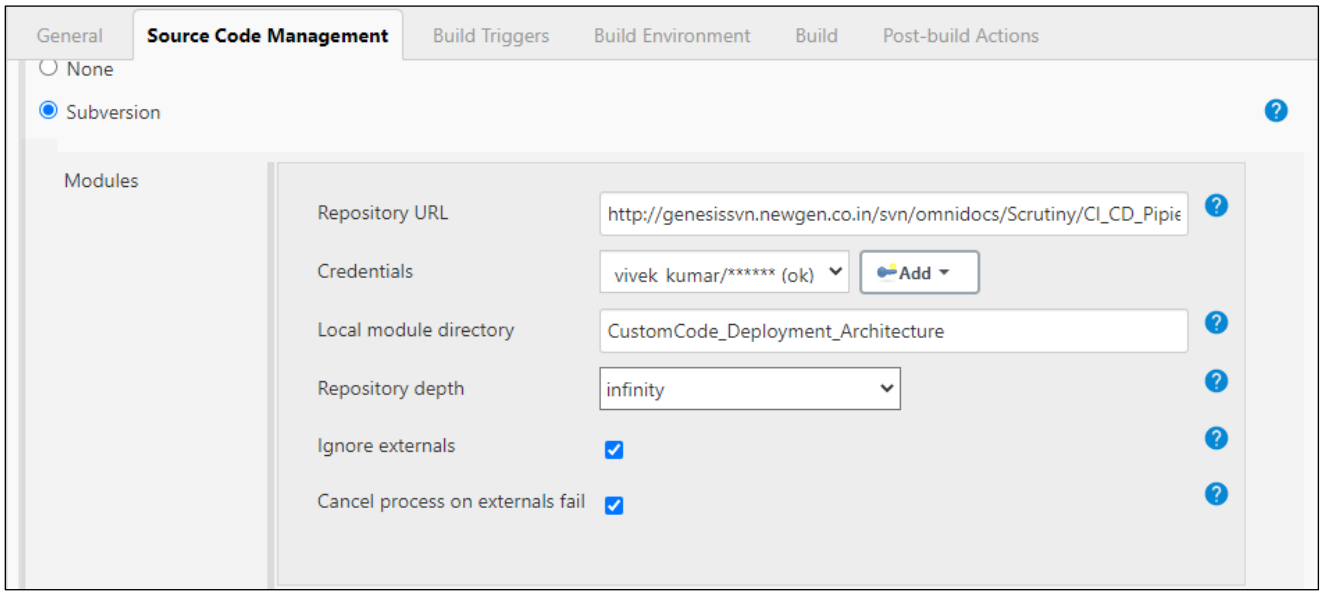

Figure 3.17

9. **Save** the changes.

3.2.2.2 Pull container image for custom code

Perform the below steps to pull container images for custom code:

1. Click **New Item** link given on the left panel.
2. Specify the item name or job name and select the project type as **Freestyle project**.
3. You can specify the project description.
4. Select the checkbox **Inject passwords to the build as environment variables** given in the **Build Environment** section.
5. Specify 2 Job passwords: **Quay_User** and **Quay_Password** and provide the appropriate username and password of the RedHat Quay Registry.
   For example,

Figure 3.18

6. Add **Inject environment variables** as a build step task in the **Build** section.
7. Specify the **UserInput.properties** file path.
   For example,

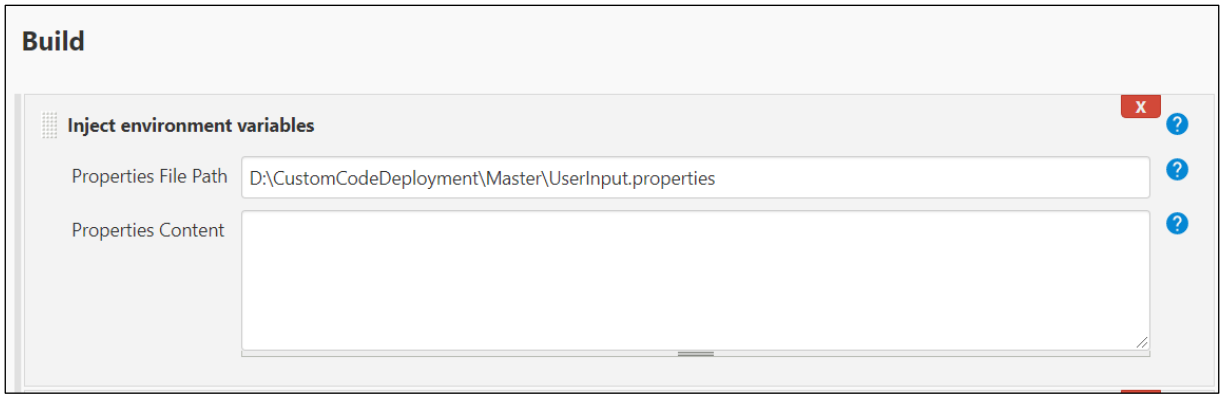

Figure 3.19

8. Add a **Conditional step (single)** as a build step task under the **Build** section.
9. Select **Execute Windows batch command** as **Run?** and **Builder** ('Run?' is a condition to decide whether a 'builder' command should run or not).
10. Specify the following command for the condition:

```
@echo off
findstr /I "OmniDocs_WEB=Y" D:\CustomCodeDeployment\Master\UserInput.properties
```

11. Specify the following commands for the builder:

```
@echo off

set Quay_Server=%Quay_Server%
set Quay_User=%Quay_User%
set Quay_Password=%Quay_Password%
set ImageName=%OmniDocs_WEB_ImageName%
set ImageTag=%OmniDocs_WEB_Imagetag%

docker login -u="%Quay_User%" -p="%Quay_Password%" quay.io
docker pull %Quay_Server%/%ImageName%:%ImageTag%
```

For example,

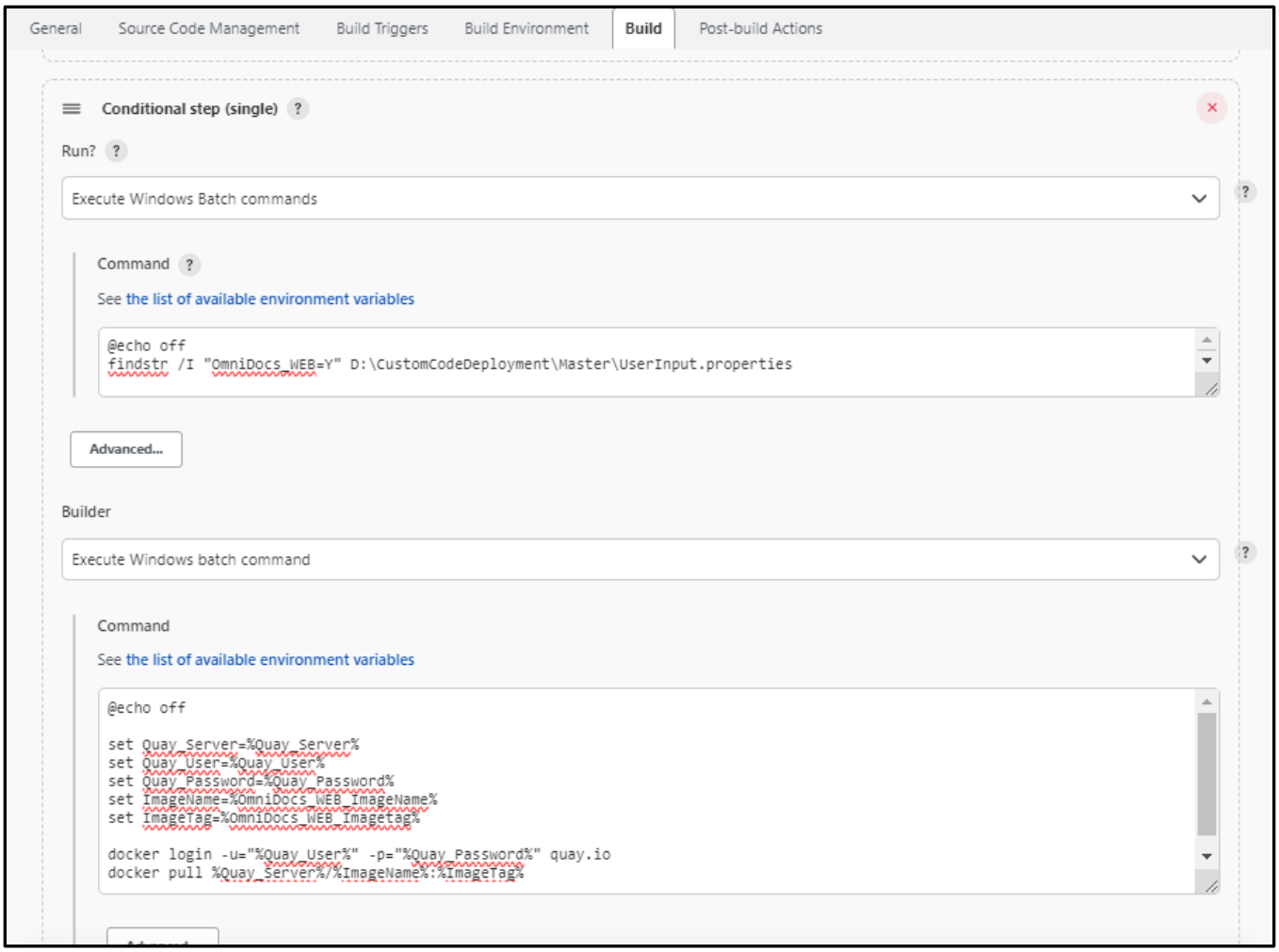

Figure 3.20

12. Click **Save** to save the changes.
    Here, the condition and builder for the **OmniDocs_WEB** container image is set.
    There is more 'Conditional step (single)' for other Docker images such as OmniDocs_EJB.

3.2.2.3 Merge custom code changes

To merge custom code changes, follow the below steps:

Click **New Item** link given on the left panel.

1. Specify the item name or job name and select the project type as **Freestyle project**.
2. You can specify the project description.
3. Add **Inject environment variables** as a build step task under the **Build** section.
4. Specify the **UserInput.properties** file path.

For example,

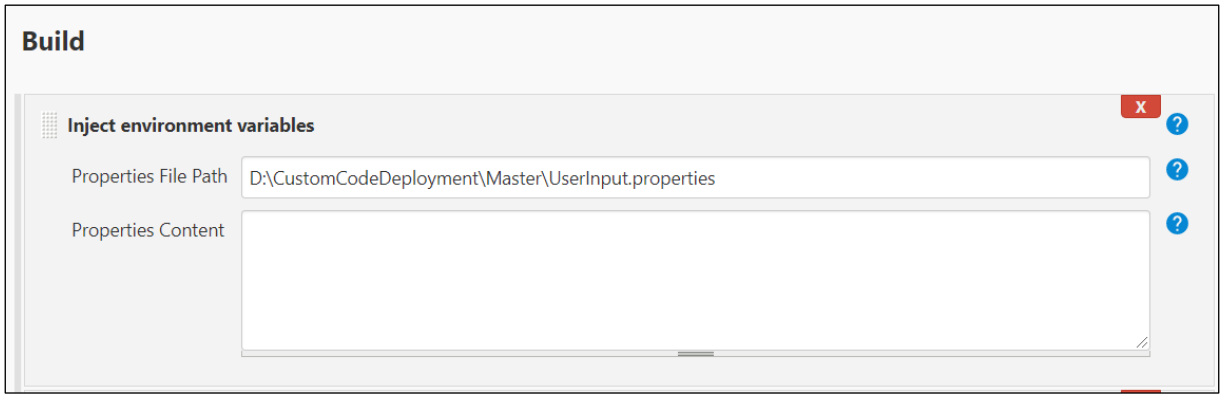

Figure 3.21

5. Add **Conditional step (multiple)** as a build step task under the **Build** section.
6. Select **Execute Windows Batch commands** as **Run?** (Run? is a condition to decide whether a builder command should run or not).
7. Click **Add step to condition** in the **Steps to run if the condition is met** section.
   For Example,

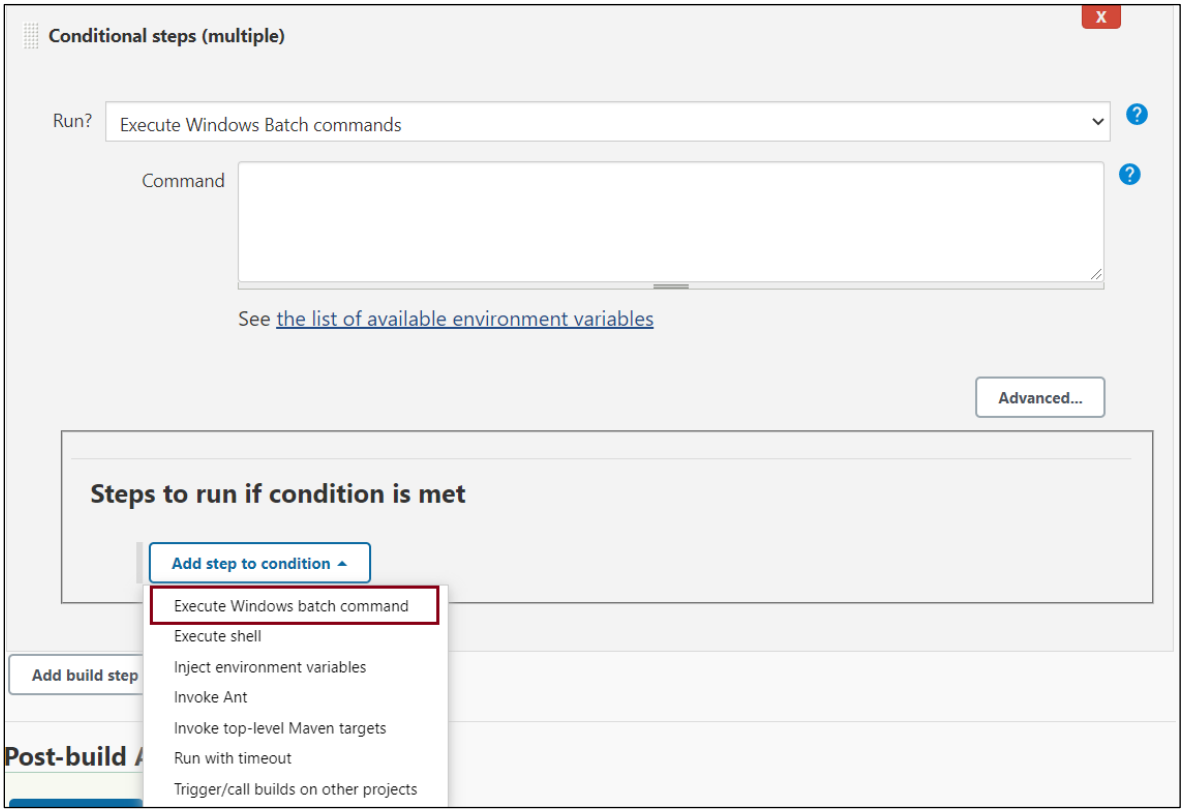

Figure 3.22

3.2.2.4 Merge OmniDocs_CustomCode custom code changes

1. Specify the following command for the condition:

```
@echo off
findstr /I "OmniDocs_CustomCode=Y"
D:\CustomCodeDeployment\Master\UserInput.properties
```

2. Specify the following commands for the 1st builder:

```
@echo off
findstr /I "OmniDocs_WEB=Y" D:\CustomCodeDeployment\Master\UserInput.properties
if %ERRORLEVEL% equ 0 goto found
goto notfound
:found
set
srcCustomCode=D:\CustomCodeDeployment\CustomCode_Deployment_Architecture\OmniDoc
s_CustomCode
set
artifactsDir=D:\CustomCodeDeployment\Build_Docker_Images\OmniDocs_WEB\artifacts\
webapps
pushd %srcCustomCode%\webapps
xcopy *.war %artifactsDir%\ /I /Y
:notfound
exit /b 0
```

3.2.2.5 Merge OmniDocs_Hook custom code changes

1. Specify the following command for the condition:

```
@echo off
findstr /I "omniDocs_Hook=Y" D:\CustomCodeDeployment\Master\UserInput.properties
```

2. Specify the following commands for the 1st builder:

```
@echo off
findstr /I "OmniDocs_EJB=Y" D:\CustomCodeDeployment\Master\UserInput.properties
if %ERRORLEVEL% equ 0 goto found
goto notfound
:found
for /f %%i in ('docker create %OmniDocs_EJB_ImageName%:%OmniDocs_EJB_Imagetag%')
do set RESULT=%%i
set srcFile1=/Newgen/jboss-eap-
7.4/modules/omnidocs_library/main/omnidocs_hook.jar
set srcFile2=/Newgen/jboss-eap-7.4/modules/omnidocs_library/main/module.xml
set destDir1=D:\CustomCodeDeployment\TempDir\omnidocs_hook\OmniDocs_EJB
set
destDir2=D:\CustomCodeDeployment\CustomCode_Deployment_Architecture\omniDocs_Hook
md %destDir1%
md %destDir2%
docker cp %RESULT%:%srcFile1% %destDir1%
docker cp %RESULT%:%srcFile2% %destDir2%
docker rm -f %RESULT%
```

```
set
srcCustomCode=D:\CustomCodeDeployment\CustomCode_Deployment_Architecture\omniDoc
s_Hook
set
artifactsDir=D:\CustomCodeDeployment\Build_Docker_Images\OmniDocs_EJB\artifacts\
modules\omnidocs_library\main\
set war=omnidocs_hook.jar
pushd %srcCustomCode%\omnidocs_hook.jar
"%JAVA_HOME%\bin\jar.exe" -uvf %destDir1%\%war% *
xcopy %destDir1%\%war% %artifactsDir% /I /Y

pushd  %srcCustomCode%
xcopy *.jar %artifactsDir% /I /Y

pushd D:\CustomCodeDeployment\Master
if exist "%srcCustomCode%\module.txt" (
"%JAVA_HOME%\bin\java.exe" -jar AppendModuleXML.jar %srcCustomCode%\module.xml
%srcCustomCode%\module.txt
"%JAVA_HOME%\bin\java.exe" -jar UpdateModuleXML.jar %srcCustomCode%\module.xml
xcopy %srcCustomCode%\module.xml %artifactsDir% /I /Y
)

:notfound
exit /b 0
```

3.2.2.6 Merge WebAPI_CustomCode custom code changes

1. Specify the following command for the condition:

```
@echo off
findstr /I "WebAPI_CustomCode=Y"
D:\CustomCodeDeployment\Master\UserInput.properties
```

2. Specify the following commands for the 1st builder:

```
@echo off
findstr /I "OmniDocs_WEB=Y" D:\CustomCodeDeployment\Master\UserInput.properties
if %ERRORLEVEL% equ 0 goto found
goto notfound
:found
for /f %%i in ('docker create %OmniDocs_WEB_ImageName%:%OmniDocs_WEB_Imagetag%')
do set RESULT=%%i
set srcFile=/Newgen/jws-5.7/tomcat/webapps/omnidocs.war
set destDir=D:\CustomCodeDeployment\TempDir\WebAPI_CustomCode\OmniDocs_WEB
md %destDir%
docker cp %RESULT%:%srcFile% %destDir%
docker rm -f %RESULT%
set
srcCustomCode=D:\CustomCodeDeployment\CustomCode_Deployment_Architecture\WebAPI_
CustomCode
```

```
set
artifactsDir=D:\CustomCodeDeployment\Build_Docker_Images\OmniDocs_WEB\artifacts\
webapps\
set war=omnidocs.war

pushd %destDir%
"%JAVA_HOME%\bin\jar.exe" -xvf %destDir%\%war% WEB-INF\lib\webapi.jar

pushd %srcCustomCode%\webapi.jar
"%JAVA_HOME%\bin\jar.exe" -uvf %destDir%\WEB-INF\lib\webapi.jar *

pushd %destDir%
"%JAVA_HOME%\bin\jar.exe" -uvf %destDir%\%war% WEB-INF\lib\webapi.jar

xcopy %destDir%\%war% %artifactsDir% /I /Y

:notfound
exit /b 0
```

3.2.2.7 Build custom code container image

To build the custom code container image, follow the below steps:

1. Click **New Item** link given on the left panel.
2. Specify the item name or job name and select the project type as **Freestyle project**.
3. You can specify the project description.
4. Add **Inject environment variables** as a build step task in the **Build** section.
5. Specify the **UserInput.properties** file path.
   For example,

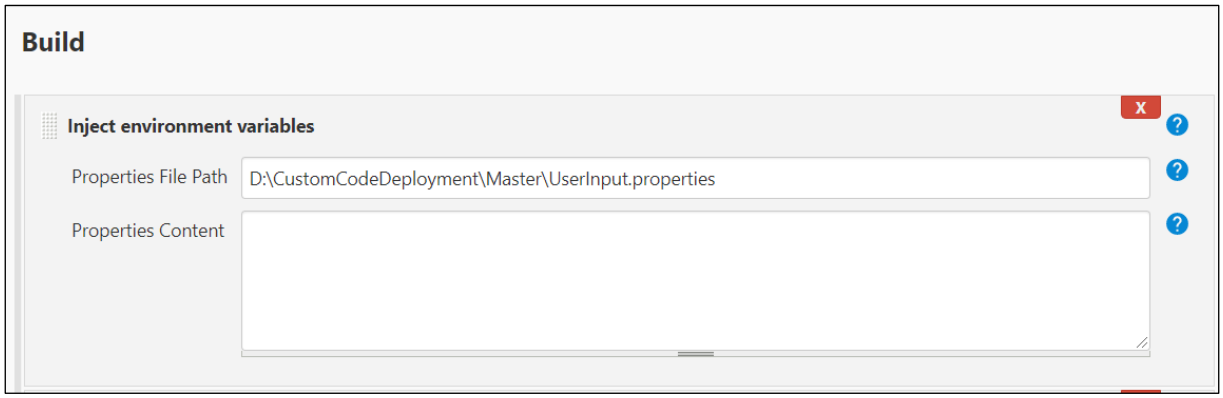

Figure 3.23

6. Add **Conditional step (single)** as a build step task under the **Build** section.

7. Choose **Execute Windows batch command** as **Run?** and **Builder** (Run? is a condition to decide whether a builder command should run or not).
   8. Specify the following command for the condition:

```
@echo off
findstr /I "OmniDocs_WEB=Y"
D:\CustomCodeDeployment\Master\UserInput.properties
```

   9. Specify the following commands for the builder:

```
@echo off
set
ImageFilePath="D:\CustomCodeDeployment\Build_Docker_Images\OmniDocs_WEB"
set ImageName=%Custom_OmniDocs_WEB_ImageName%
set ImageTag=%Custom_OmniDocs_WEB_Imagetag%

pushd %ImageFilePath%
copy /y Dockerfile_Original Dockerfile

if exist Dockerfile (
   sed -i s+QUAY_REGISTRY+%Quay_Server%+g Dockerfile
   sed -i s+IMAGE_NAME+%OmniDocs_WEB_ImageName%+g Dockerfile
   sed -i s+IMAGE_TAG+%OmniDocs_WEB_Imagetag%+g Dockerfile
) else (
    echo File doesn't exist
)

pushd %ImageFilePath%
docker build . -t %ImageName%:%ImageTag%
```

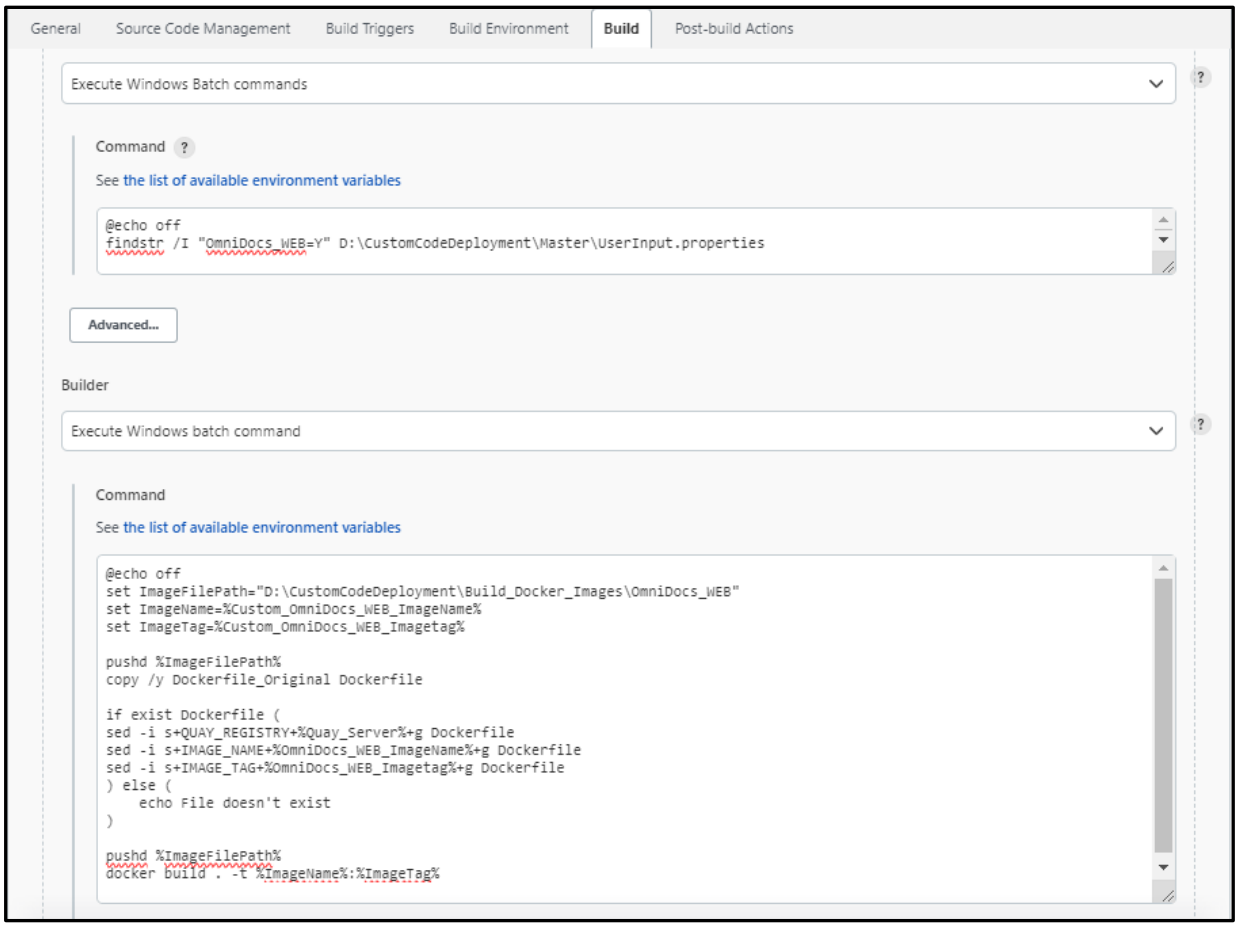

Figure 3.24

10. Click **Save.** The changes get saved.
    Here, the condition and builder for the **OmniDocs_WEB** Docker image is set.
    There are more Conditional steps (single) for other Docker images such as OmniDocs_EJB.

3.2.2.8 Push custom code container image

To push custom code container image, follow the below steps:

1. Click **New Item** link showing on the left panel.
2. Specify the item name or job name and select the project type as **Freestyle project**.
3. Specify the project description.
4. Select the checkbox **Inject passwords to the build as environment variables** given in the **Build Environment** section.
5. Specify 2 Job passwords: **Quay_User** and **Quay_Password** and provide the appropriate username and password of the RedHat Quay Registry.
   For example,

Figure 3.25

6. Add **Inject environment variables** as a build step task in the **Build** section.
7. Specify the **UserInput.properties** file path.
   For example,

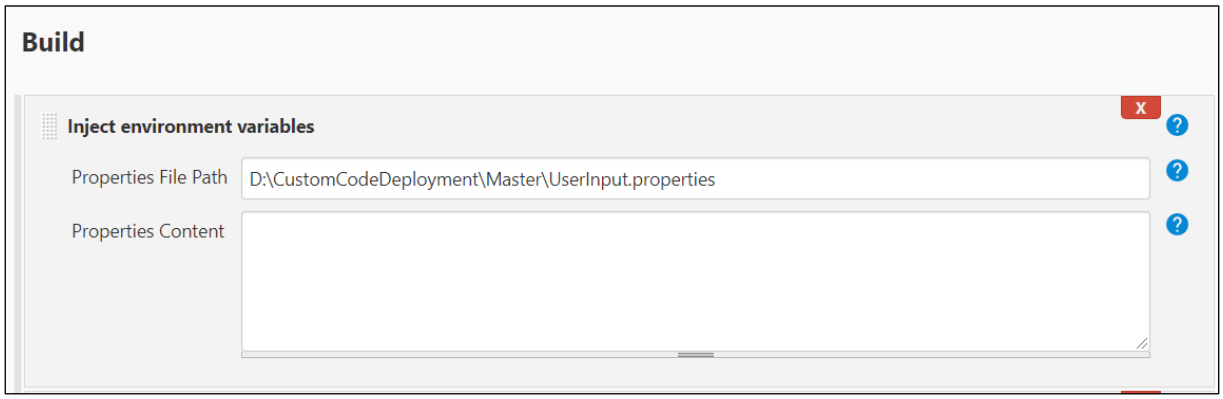

Figure 3.26

8. Add **Conditional step (single)** as a build step task under the **Build** section.
9. Select **Execute Windows batch command** as **Run?** and **Builder** (Run? is a condition to decide whether a builder command should run or not).

10. Specify the following command for the condition:

```
@echo off
findstr /I "OmniDocs_WEB=Y" D:\CustomCodeDeployment\Master\UserInput.properties
```

11. Specify the following command for the builder:

```
@echo off
set Quay_Server=%Quay_Server%
set Quay_User=%Quay_User%
set Quay_Password=%Quay_Password%
set ImageName=%Custom_OmniDocs_WEB_ImageName%
set ImageTag=%Custom_OmniDocs_WEB_Imagetag%
set BuildNumber=%ImageTag%-build-%BUILD_NUMBER%

docker login -u="%Quay_User%" -p="%Quay_Password%" quay.io

docker tag %ImageName%:%ImageTag% %Quay_Server%/%ImageName%:%ImageTag%
docker push %Quay_Server%/%ImageName%:%ImageTag%

docker tag %ImageName%:%ImageTag% %Quay_Server%/%ImageName%:%BuildNumber%
docker push %Quay_Server%/%ImageName%:%BuildNumber%
```

For example,

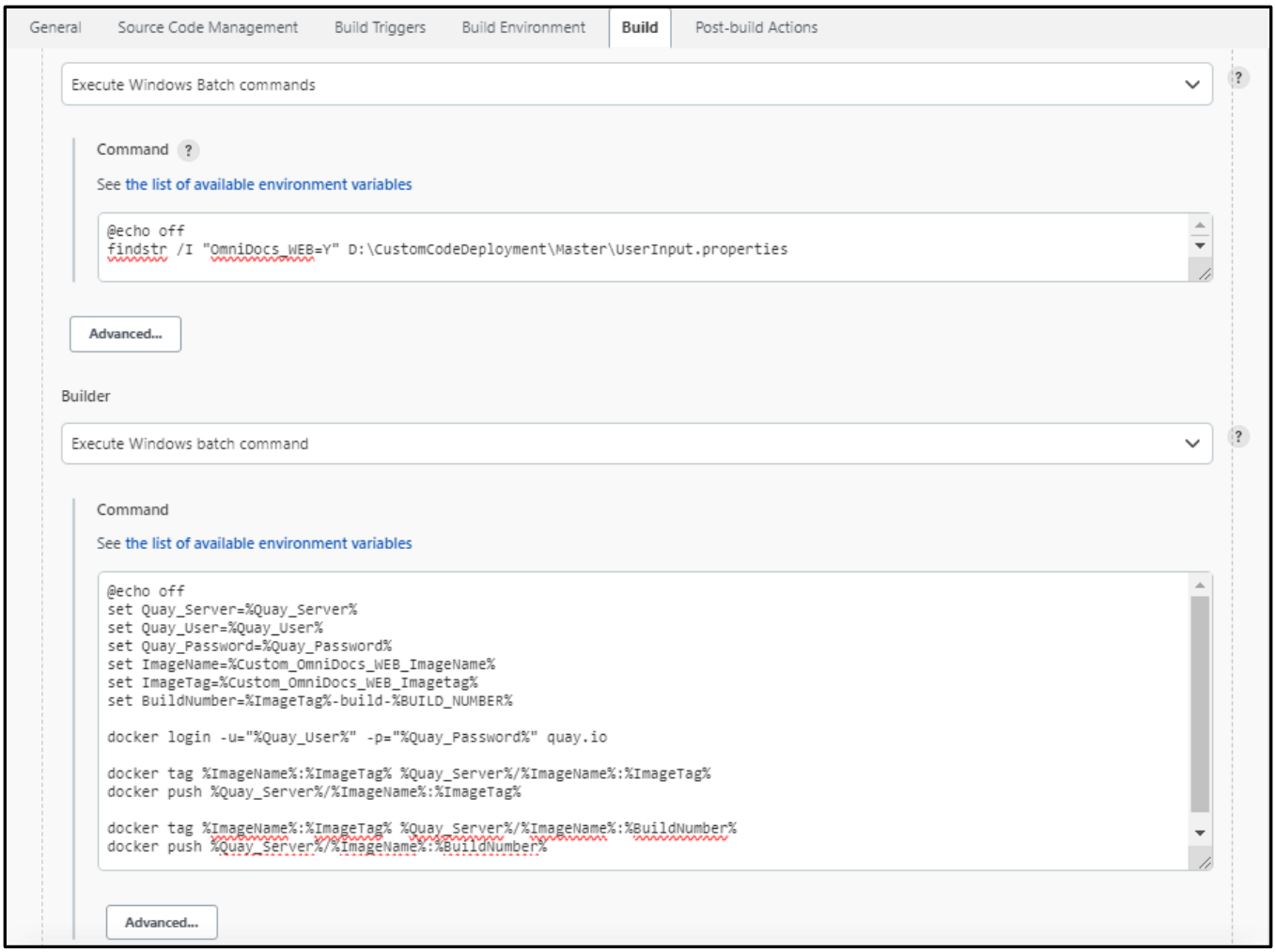

Figure 3.27

12. Click **Save**. The changes get saved.

    Here, the condition and builder for the **OmniDocs_WEB** Docker image is set.

    There are more Conditional steps (single) for other Docker images such as OmniDocs_EJB.