NewgenONE OmniDocs RMS

Docker Containers Hotfix Deployment Guide for OpenShift

Version: 4.0 SP1

Newgen Software Technologies Ltd.

Table of Contents

1 Preface 2

1.1 Revision history 2

1.2 Intended audience 2

1.3 Documentation feedback 2

1.4 Third party product information 2

2 CI/CD pipeline 3

2.1 CICD Pipeline for the hotfix of product 3

3 Implementing hotfix deployment pipeline 4

3.1 Build pipeline guidelines 4

3.2 Configuring Jenkins for Build Pipeline 9

3.2.1 Prerequisites 9

3.2.2 Configuring Jenkins jobs 9

3.2.2.1 Pull Docker Image for Hotfix 12

3.2.2.2 Creating a container image for Hotfix 14

3.2.2.3 Push HotFix Docker Image 19

1 Preface

This guide describes how to deploy the hotfix for container based OmniDocs on the OpenShift Container Platform. OmniDocs is Newgen's flagship product. This guide also describes the end-to-end implementation of the product's hotfix deployment pipeline.

1.1 Revision history

Revision Date	Description
April 2024	Initial publication.

1.2 Intended audience

This guide is intended for system administrators, developers, and all other users who are seeking information on the deployment of hotfix for container based OmniDocs. The reader must be comfortable understanding the computer terminology.

1.3 Documentation feedback

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

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

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

1.4 Third party product 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 hotfix 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 CICD Pipeline for the hotfix of product

This section described the CICD Pipeline for the hotfix of Product.

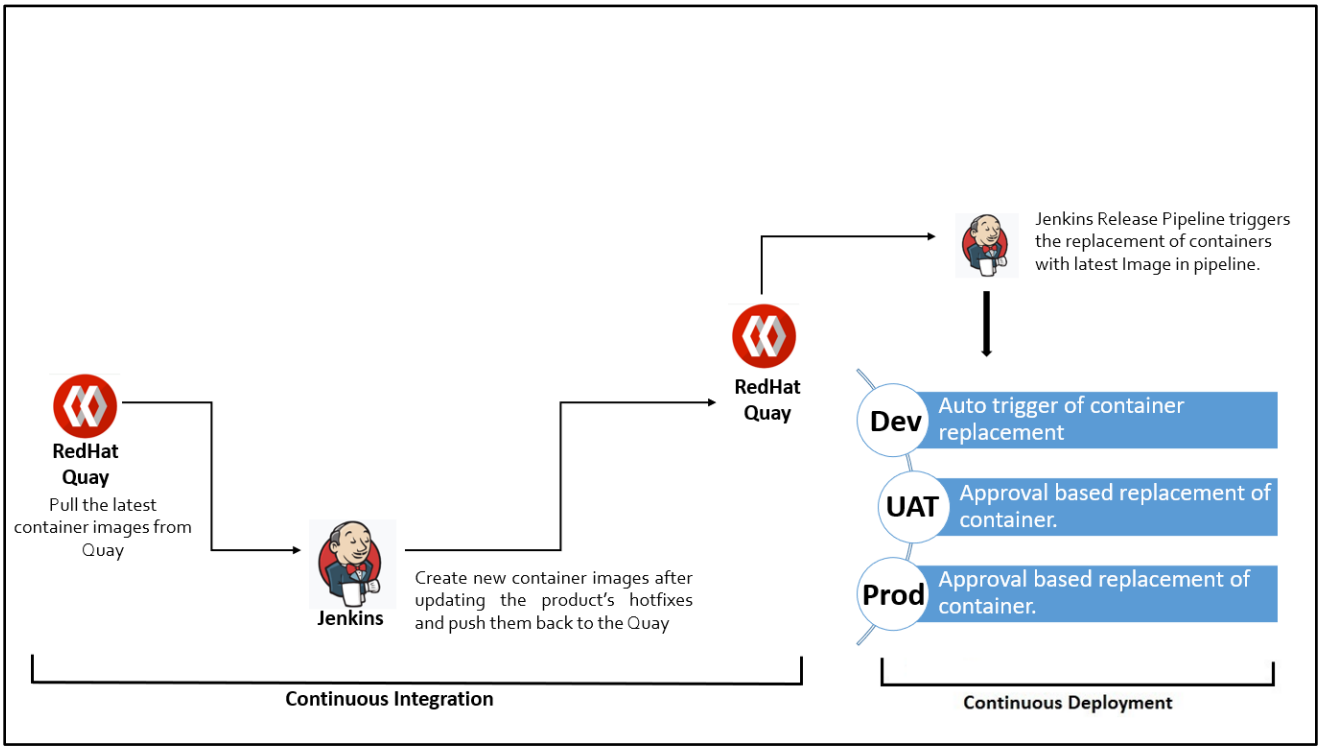

Figure 2.1

To deploy the Newgen product's hotfix, follow the below steps:

- Pull the product's base images or latest images that are already deployed in the current environment, from the container registry.
- Update the hotfix files in the earlier running container images and create new container images. The deployment structure of these hotfix files (Dockerfile) is shared along with the hotfix files, which indicates how container images require update.
- Push the newly created images to the container registry.
- 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 or Production environment, you must trigger the UAT or 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 or Production deployment gets started.

3 Implementing hotfix deployment pipeline

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

For configuration of Jenkins Release Pipeline, refer to the *OmniDocs RMS 4.0 SP1 Docker Containers Custom Code Deployment Guide for OpenShift* document.

3.1 Build pipeline guidelines

The Build pipeline guidelines are as follows:

- A pre-defined folder structure for the product's hotfix is present. The product team shares the hotfixes in that folder structure only.
  For Example,

OmniDocs Hotfix Structure

```
Dockerfile_EJB
Dockerfile_Services
Dockerfile_WEB

Common
└── artifacts
    ├── bin
    └── lib
EJB
└── artifacts
    ├── bin
    ├── deploy
    └── lib
ODServices
├── AlarmMailer
├── Scheduler
├── ThumbnailManager
└── Wrapper
WEB
└── artifacts
    ├── bin
    ├── deploy
    └── lib
```

Figure 3.1

- The WEB components and EJB components are separated into two parts. The Web components are deployed to the underlying WebServer JWS 6.0.x and EJB components are deployed to the underlying AppServer JBoss EAP 7.4.x. The build binaries are segregated like configuration files, deployable files, and dependent libraries for each container. Some binaries are specific to the WEB container, some binaries are specific to the EJB container, and some binaries are common to both containers.
- Along with the hotfix binaries, there are some Docker files. Dockerfile is a text file that contains instructions for building a Docker image. It's like a script file. You must uncomment the first and second lines respective to the cloud provider: AWS, Azure or OpenShift.
  For example,

```
#FROM REGISTRY_ID.dkr.ecr.REGION.amazonaws.com/IMAGE_NAME:IMAGE_TAG   #For AWS
#FROM ContainerRegistryPath/IMAGE_NAME:IMAGE_TAG                      #For Azure
LABEL maintainer="Newgen Software Technologies Limited"

# Install OmniDocs Web Components on JWS 6.0
COPY --chown=docker:newgen WEB/artifacts/bin /Newgen/jws-6.0/tomcat/bin
COPY --chown=docker:newgen WEB/artifacts/lib /Newgen/jws-6.0/tomcat/lib
COPY --chown=docker:newgen WEB/artifacts/deploy /Newgen/jws-6.0/tomcat/webapps
COPY --chown=docker:newgen Common/artifacts/bin /Newgen/jws-6.0/tomcat/bin
COPY --chown=docker:newgen Common/artifacts/lib /Newgen/jws-6.0/tomcat/lib

EXPOSE 8080

#Switch to user: docker
USER 1000

WORKDIR /Newgen/jws-6.0
CMD /bin/run.sh
```

Figure 3.2

```
#FROM REGISTRY_ID.dkr.ecr.REGION.amazonaws.com/IMAGE_NAME:IMAGE_TAG  #For AWS
#FROM ContainerRegistryPath/IMAGE_NAME:IMAGE_TAG                    #For Azure
LABEL maintainer="Newgen Software Technologies Limited"

# Install NewgenOne AutomationStudio EJB Components on JBossEAP 7.4
COPY --chown=docker:newgen EJB/artifacts/bin /Newgen/jboss-eap-7.4/bin
COPY --chown=docker:newgen EJB/artifacts/lib /Newgen/jboss-eap-7.4/modules/omnidocs_library/main
COPY --chown=docker:newgen EJB/artifacts/deploy /Newgen/jboss-eap-7.4/standalone/deployments
COPY --chown=docker:newgen Common/artifacts/bin /Newgen/jboss-eap-7.4/bin
COPY --chown=docker:newgen Common/artifacts/lib /Newgen/jboss-eap-7.4/modules/omnidocs_library/main

EXPOSE 8080 9990

#Switch to user: docker
USER 1000

WORKDIR /Newgen/jboss-eap-7.4
CMD /bin/run.sh
```

Figure 3.3

- Deployable files and dependent libraries are merged inside the container images as there are no dynamic changes in these types of files. Also, they can be merged using Dockerfiles shared along with hotfixes.
- Since configuration files are dynamic, they must be kept outside the container. For this, volume persistence is mapped to the external disk storage like NAS server. So, whenever configuration changes are found in a product's hotfix, update the configuration files located at external disk storage along with updating the container images.
- If database scripts are found in a product's hotfix *DatabaseScripts* folder, execute them manually through Database Client software.
- Jenkins Build Pipeline have three jobs that are as follows:
    i. Pull the latest container image from the container repository in which the hotfix must be deployed.
    ii. Create new container images after updating the hotfix binaries.
    iii. Push the newly created container image to the container registry.

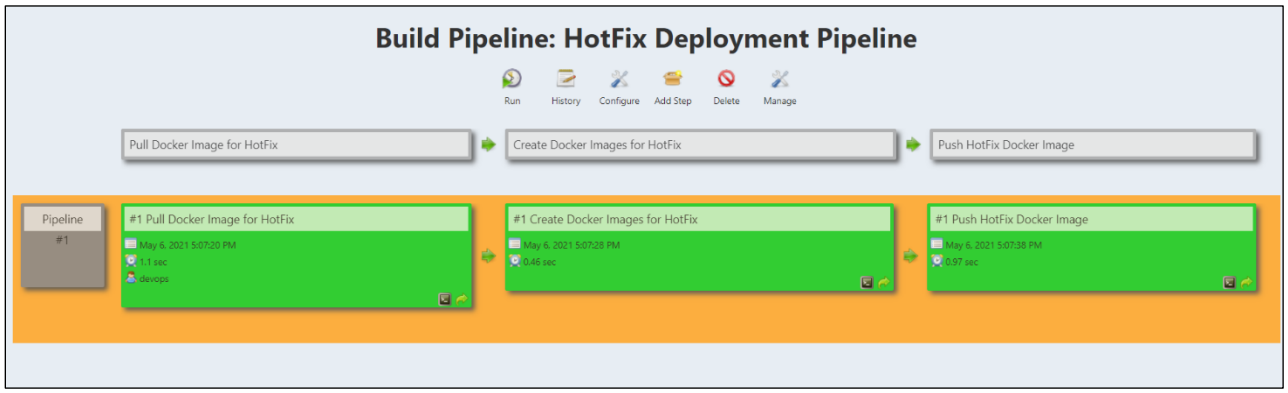

Figure 3.4

- Before pulling the latest container images from the container registry, Jenkins reads the *UserInput.properties* file.
- This properties file contains all the user inputs that are required for condition-based hotfix deployment.
- This property file has multiple sections.
  For example:
    - **#Container Registry Info**
      This section contains the container registry information. Here you must 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 3.5

- **#HotFix Info**
  Deploy the hotfix location.
  For example,
  HotFix_Location=*"C:\Users\Administrator\Downloads\ OmniDocs 11.0 SP1 Patch3_HF01 "*
- **#Container Image to be updated**
  Select the Docker image(s) in which you must deploy hotfix binaries.
  For example, deploy a hotfix in the OmniDocs WEB container, then set the OmniDocs _WEB=Y.
  For example,

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

Figure 3.6

- **# Container Image Info**
  This section contains the information about the source container images in which hotfix binaries get updated 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

OmniDocs_Services_ImageName=od11.0services
OmniDocs_Services_Imagetag=base
```

Figure 3.7

- **#New Container Image Info with Hotfix changes**

  This section contains the information about new container images that get created after updating the hotfix binaries.

  For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#New Docker Image Info with Hotfix changes
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

HotFix_OmniDocs_WEB_ImageName=omnidocs11.0web
HotFix_OmniDocs_WEB_Imagetag=hf01

HotFix_OmniDocs_EJB_ImageName=omnidocs11.0ejb
HotFix_OmniDocs_EJB_Imagetag=hf01

HotFix_OmniDocs_Services_ImageName=od11.0services
HotFix_OmniDocs_Services_Imagetag=hf01
```

Figure 3.8

- **#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 3.9

- Based on the input provided in the *UserInput.properties* file, Jenkins pulls the container images, creates new container images after updating hotfix binaries, and pushes container images to the container repository.

3.2 Configuring Jenkins for 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 Datacenter).
- **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 Configuring Jenkins jobs

For the hotfix deployment pipeline, Jenkins contains the three jobs that are as follows:

1. Pull the latest container image from the container repository in which hotfix must be deployed.
2. Create new container images after updating the hotfix binaries.
3. Push the newly created container images to the container registry.

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

1. Log in to the Jenkins Server.

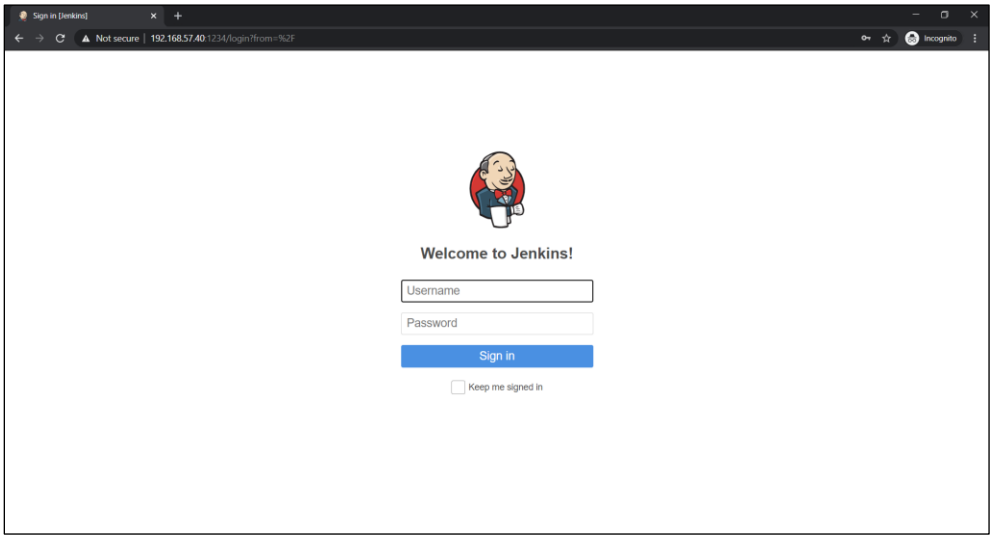

Figure 3.10

2. After successful login, click **Manage Jenkins** link given in the left panel.

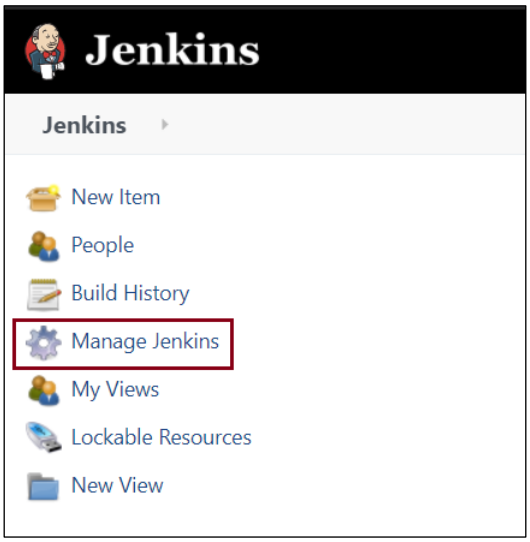

Figure 3.11

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

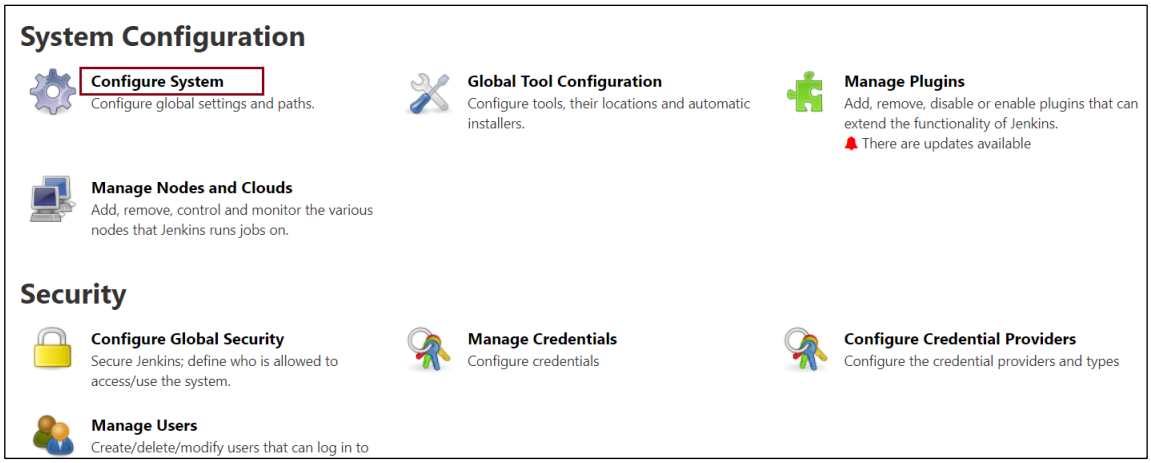

Figure 3.12

4. In **Global properties**, define an environment variable **PATH** with the following values separated with a semicolon:
   - Docker installation path. For example, *C:\Program Files\Docker\Docker\resources\bin*
   - Cygwin installation path. For example, *C:\cygwin64\bin*
   - OpenShift CLI installation path. For example, *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** the changes.

3.2.2.1 Pull Docker Image for Hotfix

Perform the below steps to pull the Docker image for the hotfix:

1. Click the **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. Specify the project description.
4. Select the checkbox **Inject passwords to the build as environment variables** in the **Build Environment** section.
5. Specify two Job passwords: **Quay_User** and **Quay_Password** and provide the appropriate username and password of the RedHat Quay Registry.

For example,

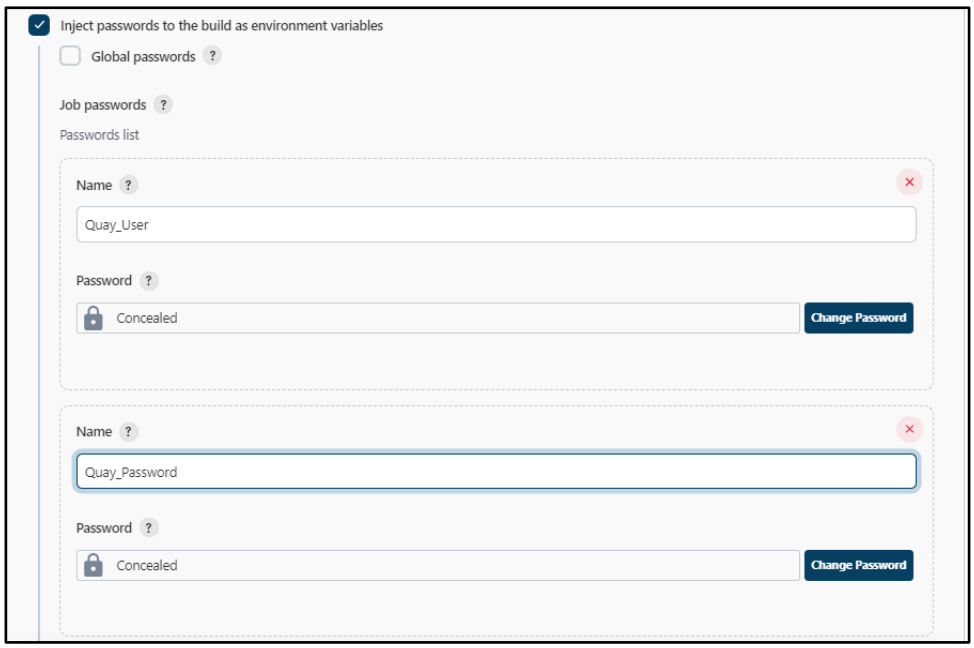

Figure 3.14

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

For example,

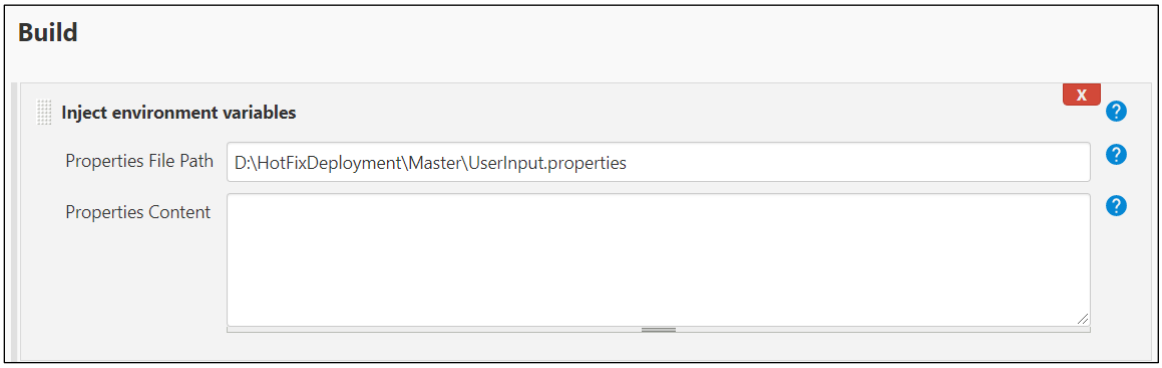

Figure 3.15

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

```
@echo off
findstr /I " OmniDocs _WEB=Y" D:\HotFixDeployment\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,

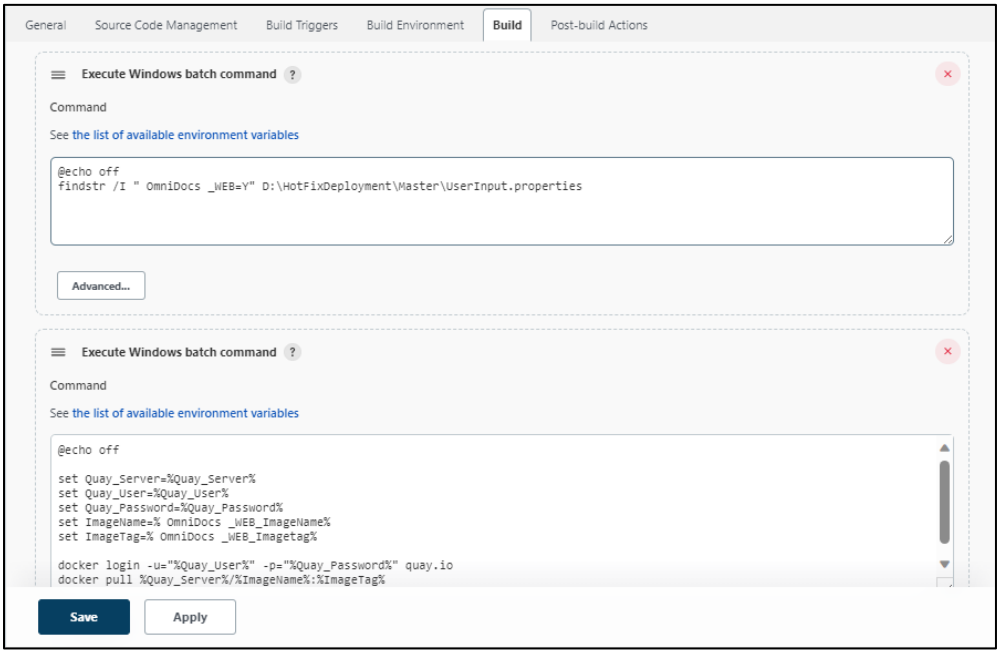

Figure 3.16

12. Click **Save** the changes.
13. Here, the condition and builder are set for the **OmniDocs _WEB** Docker image.
    There are more Conditional steps (single) for other Docker images like:
    - OmniDocs_EJB
    - OmniDocs_Services

3.2.2.2 Creating a container image for Hotfix

Perform the below steps to create the container image for Hotfix:

1. Click **New Item** link appears 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 under the **Build** section.
5. Specify the *UserInput.properties* file path.

For example,

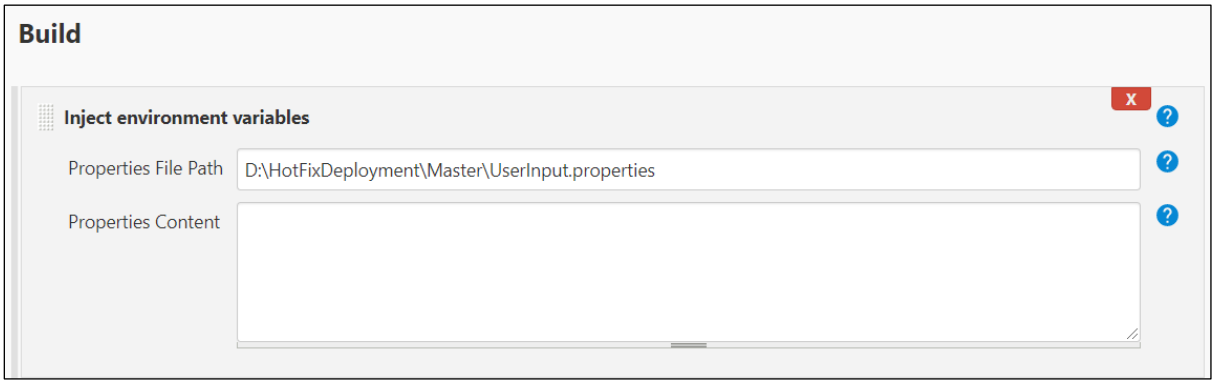

Figure 3.17

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

```
@echo off
findstr /I " OmniDocs _WEB=Y" D:\HotFixDeployment\Master\UserInput.properties
```

9. Specify the following commands for the builder:

```
@echo off
set ImageFilePath="%HotFix_Location%"
set SourceImageName=% OmniDocs _WEB_ImageName%
set SourceImageTag=% OmniDocs _WEB_Imagetag%
set DestImageName=%HotFix_ OmniDocs _WEB_ImageName%
set DestImageTag=%HotFix_ OmniDocs _WEB_Imagetag%
set DockerFileName=Dockerfile_WEB

if exist %ImageFilePath% goto found
goto notfound

:found
pushd %ImageFilePath%
copy /y %DockerFileName% %DockerFileName%_temp

if exist %DockerFileName%_temp (
   sed -i s+QUAY_REGISTRY+%Quay_Server%+g %DockerFileName%_temp
   sed -i s+IMAGE_NAME+%SourceImageName%+g %DockerFileName%_temp
   sed -i s+IMAGE_TAG+%SourceImageTag%+g %DockerFileName%_temp
) else (
    goto DockerfileNotFound
)
```

```
pushd %ImageFilePath%
docker build . -t %DestImageName%:%DestImageTag% -f %DockerFileName%_temp
del /Q %DockerFileName%_temp

goto finish

:DockerfileNotFound
echo "%DockerFileName%_temp does not exist."
goto finish

:notfound
echo "HotFix Location does not exist."

:finish
exit /b 0
```

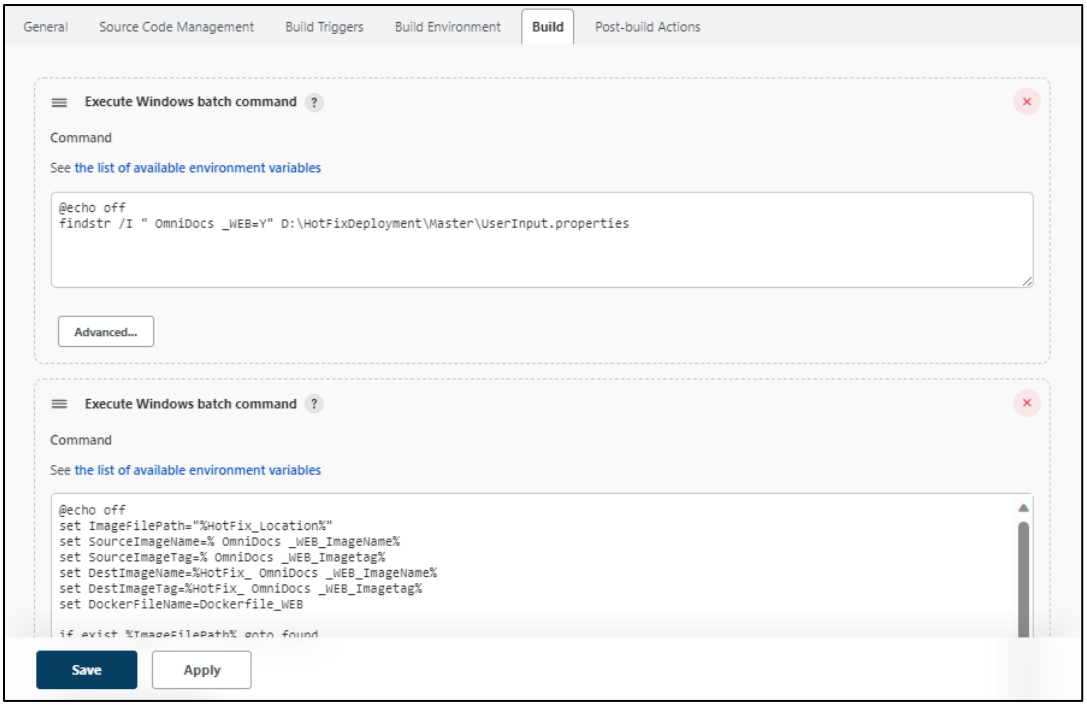

Figure 3.18

10. Click **Save** the changes.
11. Here, the condition and builder are set for the **OmniDocs _WEB** Docker image.

    There are more Conditional steps (single) for other Docker images like:

    - OmniDocs_EJB
    - OmniDocs_Services
12. For **OmniDocs_EJB**, use Conditional steps (multiple). In OmniDocs Hotfix, the OmniDocs_ejb.jar is received instead of *OmniDocs_ejb.ear*. In such a case, extract the *OmniDocs_ejb.ear* from the

existing container image, update the latest *OmniDocs_ejb.jar*, and then create a new container image.

13. Add **Conditional step (multiple)** as a build step task under the **Build** section.
14. Choose **Execute Windows batch command** as **Run?** and **Builder**. ['Run?' is a condition to decide whether a 'builder' command must run or not].
15. Add 2 **Add step to condition** in the **Steps to run if the condition is met** section.

    For example,

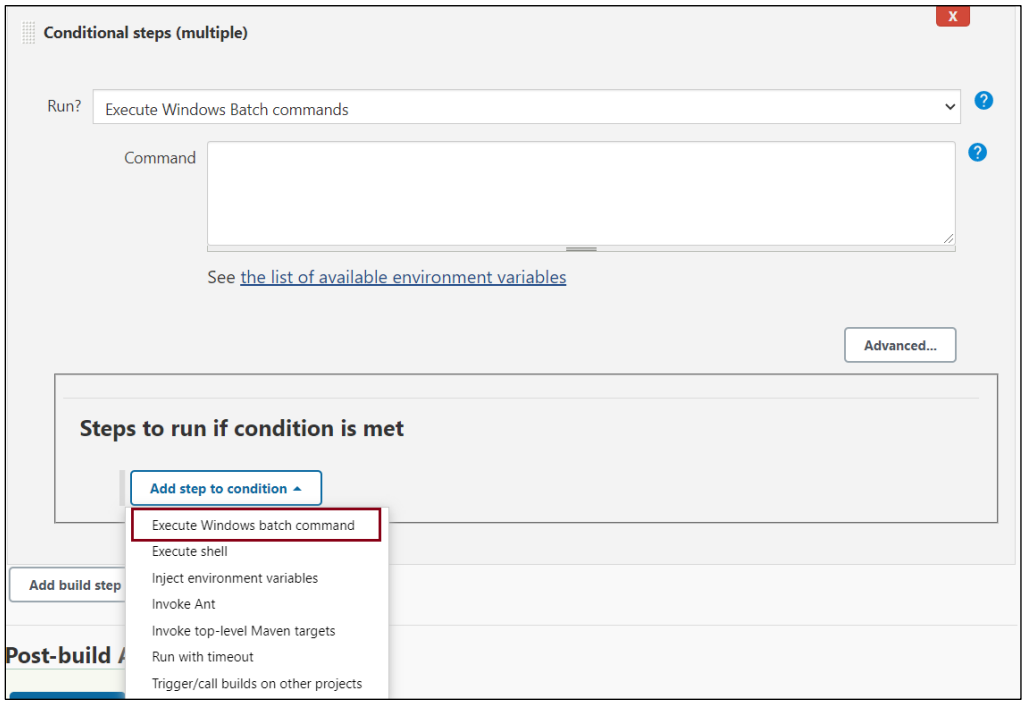

Figure 3.19

16. Specify the following command for the condition:

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

17. Specify the following commands for the first builder:

```
@echo off
for /f %%i in ('docker create %OmniDocs_EJB_ImageName%:%OmniDocs_EJB_Imagetag%')
do set RESULT=%%i
set srcFile= srcFile=/Newgen/jboss-eap-
7.4/standalone/deployments/OmniDocs_ejb.ear
set destDir=D:\HotFixDeployment\TempDir\OmniDocs_EJB
md %destDir%
docker cp %RESULT%:%srcFile% %destDir%
docker rm -f %RESULT%
```

```
set OD_EJB_Location=%HotFix_Location%\EJB\artifacts\deploy
if exist %OD_EJB_Location% goto found
goto notfound

:found
pushd %OD_EJB_Location%

if exist %OD_EJB_Location%\OmniDocs_ejb.jar goto continue
goto filenotfound

:continue
pushd %OD_EJB_Location%
"%JAVA_HOME%\bin\jar.exe" -uvf %destDir%\OmniDocs_ejb.ear OmniDocs_ejb.jar
xcopy %destDir%\OmniDocs_ejb.ear %OD_EJB_Location%\ /I /Y
del /Q %OD_EJB_Location%\OmniDocs_ejb.jar
goto finish

:filenotfound
echo "OmniDocs_ejb.jar could not found."

:notfound
echo "%OD_EJB_Location% does not exist."

:finish
RD /S /Q %destDir%
exit /b 0
```

18. Specify the following commands for the second builder:

```
@echo off
set ImageFilePath="%HotFix_Location%"
set SourceImageName=%OmniDocs_EJB_ImageName%
set SourceImageTag=%OmniDocs_EJB_Imagetag%
set DestImageName=%HotFix_OmniDocs_EJB_ImageName%
set DestImageTag=%HotFix_OmniDocs_EJB_Imagetag%
set DockerFileName=Dockerfile_EJB

if exist %ImageFilePath% goto found
goto notfound

:found
pushd %ImageFilePath%
copy /y %DockerFileName% %DockerFileName%_temp

if exist %DockerFileName%_temp (
      sed -i s+QUAY_REGISTRY+%Quay_Server%+g %DockerFileName%_temp
      sed -i s+IMAGE_NAME+%SourceImageName%+g %DockerFileName%_temp
      sed -i s+IMAGE_TAG+%SourceImageTag%+g %DockerFileName%_temp
) else (
    goto DockerfileNotFound
```

```
)

pushd %ImageFilePath%
docker build . -t %DestImageName%:%DestImageTag% -f %DockerFileName%_temp
del /Q %DockerFileName%_temp

goto finish

:DockerfileNotFound
echo "%DockerFileName%_temp does not exist."
goto finish

:notfound
echo "HotFix Location does not exist."

:finish
exit /b 0
```

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

3.2.2.3 Push HotFix Docker Image

Perform the below steps to push the hotfix Docker images:

1. Click **New Item** link appears 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** under the **Build Environment** section.
5. Specify two Job passwords: **Quay_User** and **Quay_Password** and provide the appropriate username and password of the RedHat Quay Registry.
   For example,

Figure 3.20

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

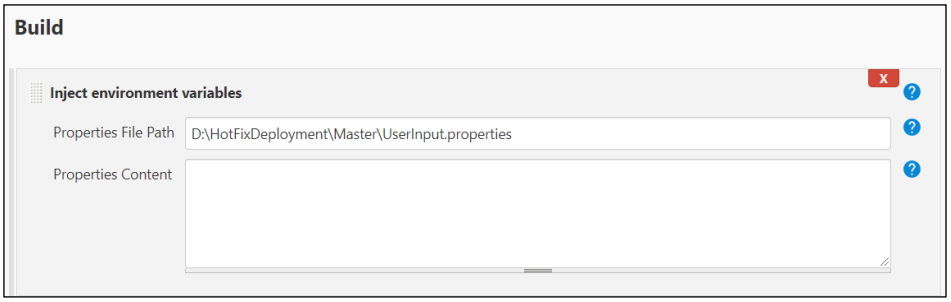

Figure 3.21

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

```
@echo off
findstr /I " OmniDocs_WEB=Y"  D:\HotFixDeployment\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=%HotFix_ OmniDocs_WEB_ImageName%
set ImageTag=%HotFix_ 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,

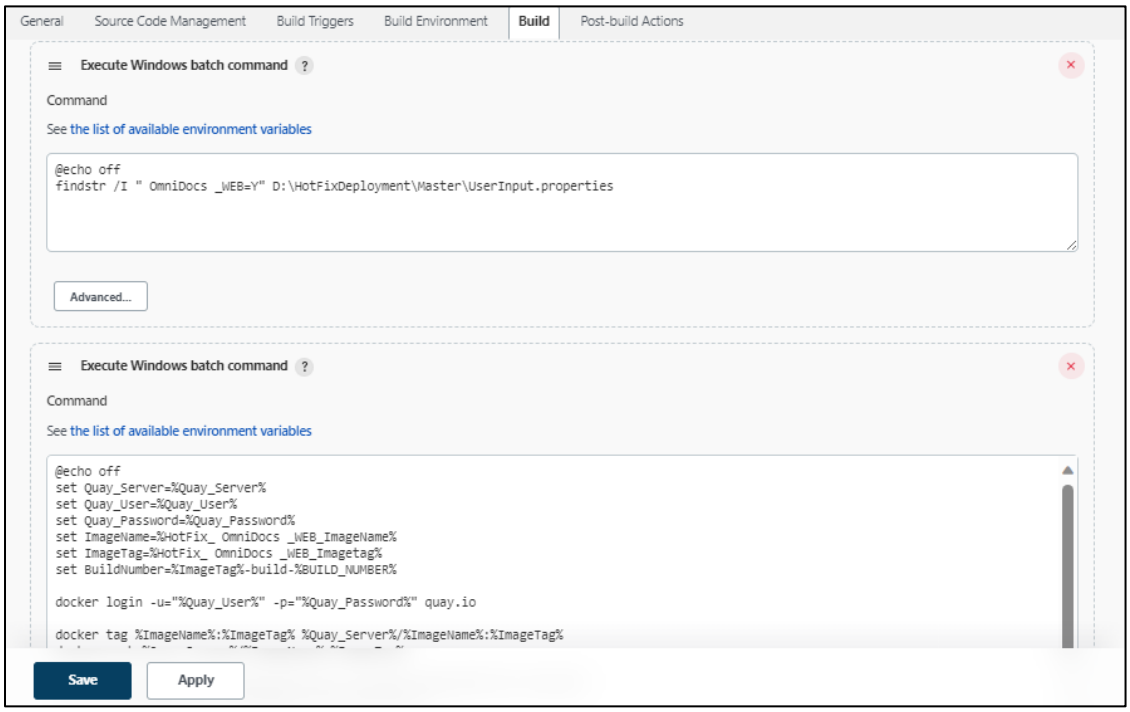

Figure 3.22

12. Click **Save** the changes.
13. Here, the condition and builder are set for the **OmniDocs _WEB** Docker image.

    There are more Conditional steps (single) for other Docker images like:

    - OmniDocs_EJB
    - OmniDocs_Services