NewgenONE OmniDocs RMS

Docker Containers Hotfix Deployment Guide for AWS

Version: 4.0 SP1

Newgen Software Technologies Ltd.

Table of Contents

1 Preface

This guide describes how to deploy the hotfix for container based NewgenONE OmniDocs RMS 4.0 SP1 on the AWS (Amazon Web Services). OmniDocsRMS is a 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 cloud administrators, 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, 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:

2 CI/CD pipeline

The CICD pipeline manages the hotfix deployments with Kubernetes orchestration on cloud platforms. Here, the separation of the Build Pipeline and Release Pipeline is done into two parts. The Build Pipeline is done by the Jenkins server that can be installed on-premises or in a cloud VM. The Release pipeline is managed by AWS CodePipeline cloud service. In this architecture, there are three stages Dev, UAT, and Production and on each stage, deployment is quite different. More stages can be added depending on the requirements.

2.1 CICD Pipeline for the hotfix of product

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

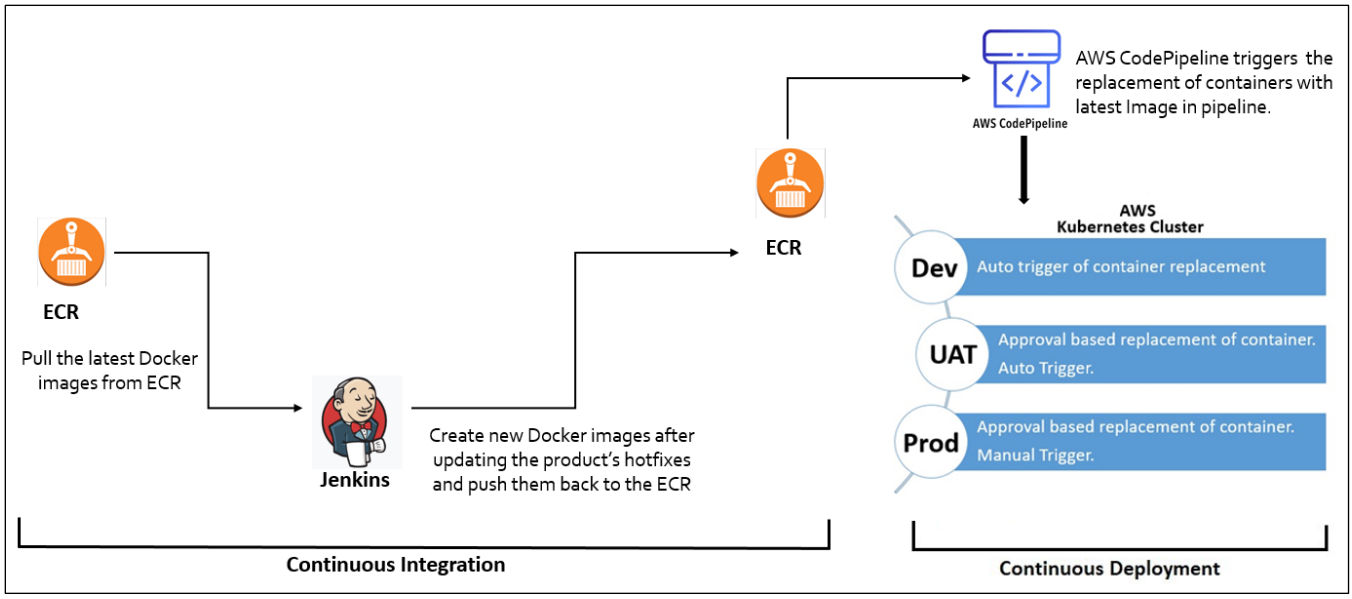

Figure 2.1

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 Docker images and create new Docker images. The deployment structure of these hotfix files (Dockerfile) is shared along with the hotfix files, which indicates how Docker images requires update.
- Push the newly created images to the container registry.
- As soon as any Docker Image is pushed to the AWS ECR (Elastic Container Registry), AWS CodePipeline triggers the deployment to the Dev environment.

- UAT and Production deployments are approval based and they are called on-demand. To deploy the UAT environment, trigger the UAT deployment. Once the deployment is triggered, an approval mail is sent. After receiving the approval, the UAT deployment starts automatically.
- The production deployment is also approval based but it is multi-level approval, to deploy to a production environment the approval of all stakeholders is required, and most importantly once all the approvals from stakeholders are received, deployment to the production environment is not triggered automatically. A manual intervention mail is sent. To deploy to production with a checklist, all the checklist points get verified that they are covered or not. If not, then the deployment to the production gets rejected.

3 Implementation of hotfix deployment pipeline

The hotfix deployment pipeline is separated into two parts: **Build Pipeline and Release Pipeline**. The Build Pipeline is configured on the Jenkins Server and the Release Pipeline is configured on the AWS CodePipeline.
For configuration of the Release Pipeline, refer to the *Configuration of AWS CodePipeline* document.

3.1 Guideline for build pipeline

The guideline for the build pipeline is described below:

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

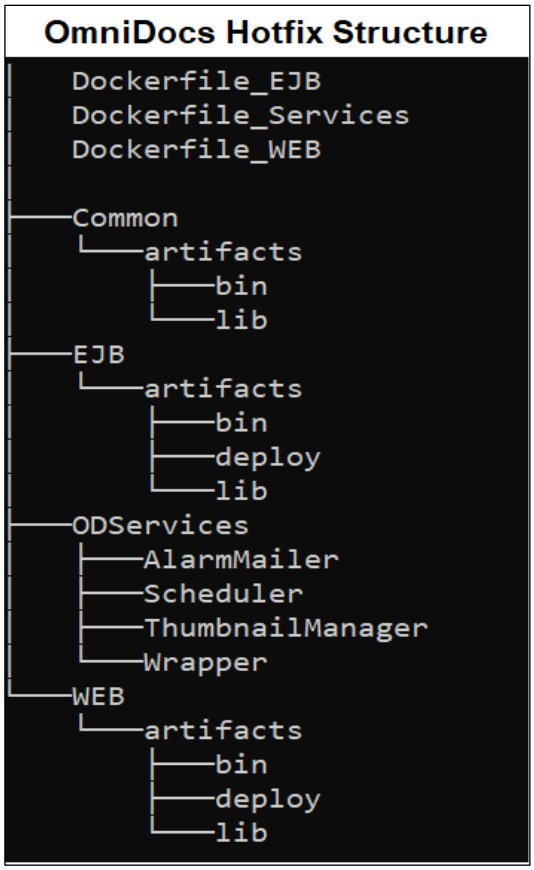

Figure 3.1

- The WEB components and EJB components are separated into two parts. The Web components are deployed to the underlying WebServer JWS 5.7.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 Docker 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, some Dockerfiles is shared. Dockerfile is a text file that contains instructions for building a Docker image. It's like a script file. End-user needs to uncomment the 1st or 2nd line respective to the cloud provider: AWS or Azure.
  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 11.0 Web Components on JWS 5.7
COPY --chown=docker:newgen WEB/artifacts/bin /Newgen/jws-5.7/tomcat/bin
COPY --chown=docker:newgen WEB/artifacts/lib /Newgen/jws-5.7/tomcat/lib
COPY --chown=docker:newgen WEB/artifacts/deploy /Newgen/jws-5.7/tomcat/webapps
COPY --chown=docker:newgen Common/artifacts/bin /Newgen/jws-5.7/tomcat/bin
COPY --chown=docker:newgen Common/artifacts/lib /Newgen/jws-5.7/tomcat/lib

EXPOSE 8080

#Switch to user: docker
USER 1000

WORKDIR /Newgen/jws-5.7
CMD /bin/run.sh
```

Figure 3.2

- Deployable files and dependent libraries are merged inside the Docker containers 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 AWS EFS. So, whenever configuration changes are found in a product's hotfix, update the configuration files located at external disk storage along with updating the Docker images.
- If database scripts are found in a product's hotfix, execute them manually through Database Client software.
- Jenkins Build Pipeline have **three jobs** that are as follows:
    i. Pull the latest Docker image from the container repository in which the hotfix needs to be deployed.
    ii. Create new Docker images after updating the hotfix binaries.
    iii. Push the newly created Docker image to the container registry.
    For example,

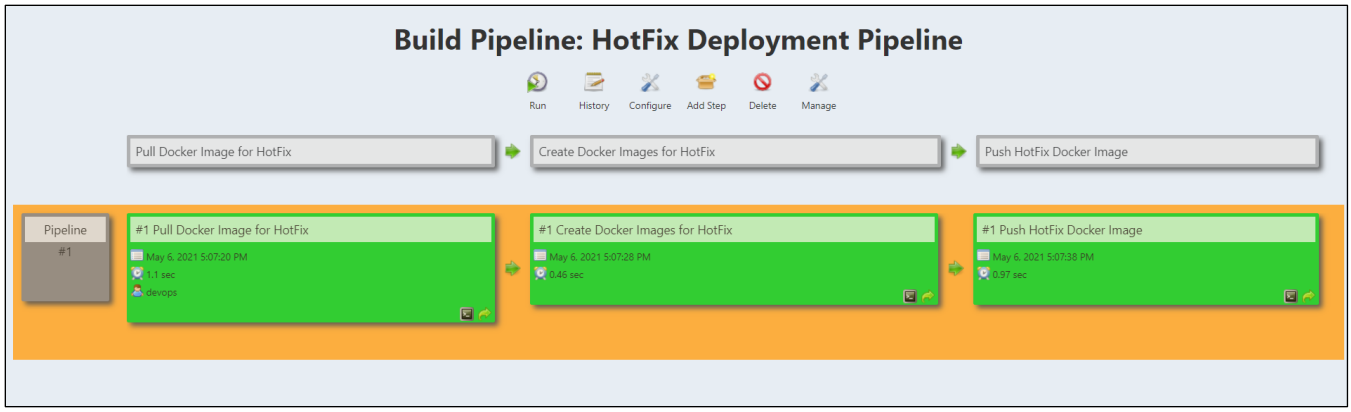

Figure 3.3

- Before pulling the latest Docker 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, provide **AWS_AccountID** and **AWS_Region** where the container registry is created in. **AWS_AccessKey** and **AWS_SecretKey** are used as encrypted environment variables in Jenkins jobs.
    For example,

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Container Registery Info
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#AWS_AccessKey= Used as an encrypted environment variable in Jenkins Jobs
#AWS_SecretKey= Used as an encrypted environment variable in Jenkins Jobs
AWS_AccountID=678035612169
AWS_Region=ap-south-1
```

Figure 3.4

  - **#HotFix Info**
    Deploy the hotfix location.
    For example,
    HotFix_Location="C:\Users\Administrator\Downloads\OD_11.0_SP0_P00_HF01"

- **#Docker Image to be updated**
  Select the Docker image(s) in which 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 3.5

- **#Docker Image Info**
  This section contains the information about the source Docker images in which hotfix binaries gets 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 3.6

- **#New Docker Image Info with Hotfix changes**
  This section contains the information about new Docker images that gets 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 3.7

- **#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

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

3.2 Configuration of 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 (Edition: Standard or Data Center)
- **Java** 1.8 update 91 and above
- **Docker Engine** 20.10.10 or later version must be installed.
- **AWS CLI** 2.0.27 or a later version must be installed.

- **Cygwin** utility must be installed. This utility is used to execute Linux commands on Windows.
- **Jenkins** 2.235.0 or a later version must be installed with default plug-ins along with the following plug-ins:
  - Conditional Build Step
  - Credentials Binding
  - Environment Injector

3.2.2 Configuration of Jenkins Jobs

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

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

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

1. Log in to the Jenkins Server.

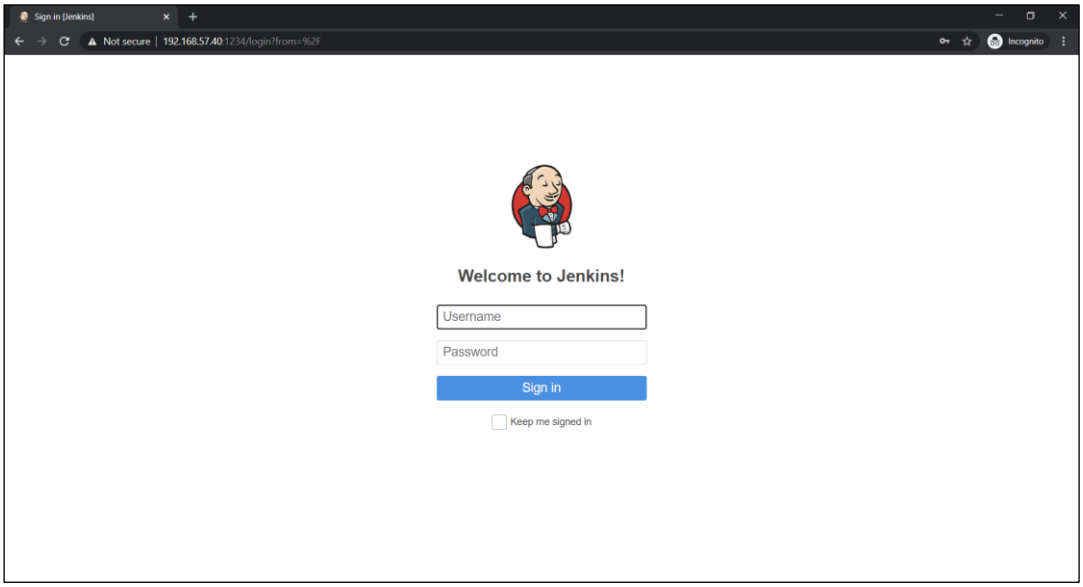

Figure 3.9

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

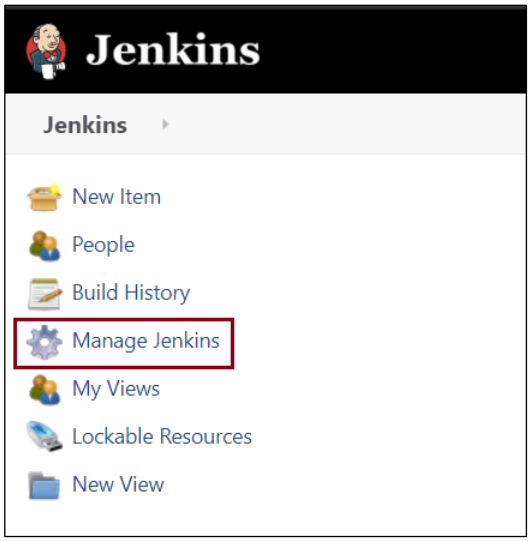

Figure 3.10

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

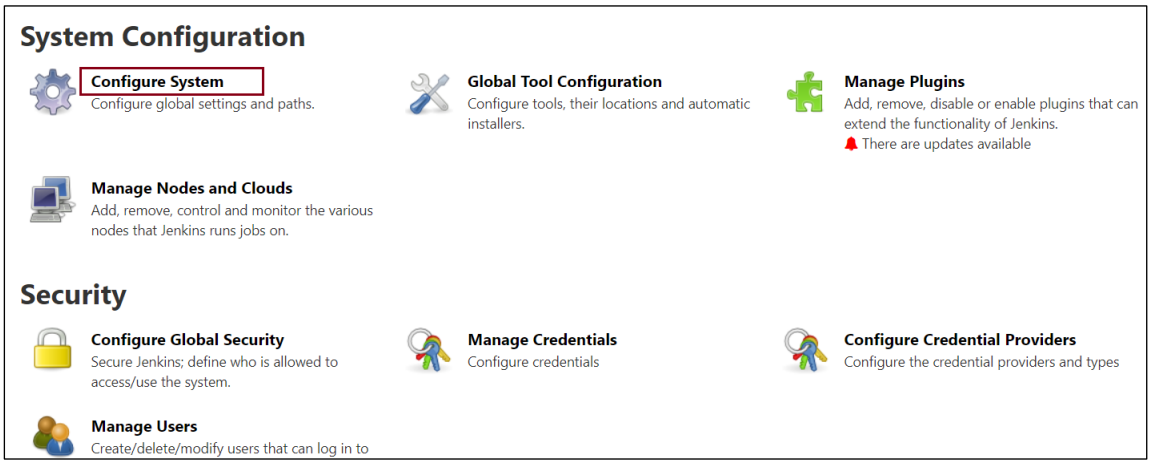

Figure3.11

4. In Global properties, define an environment variable **PATH** with the following values separated with a semicolon:
   - Docker installation path [C:\Program Files\Docker\Docker\resources\bin]
   - Cygwin installation path [C:\cygwin64\bin]
   - AWS CLI installation path [C:\Program Files\Amazon\AWSCLIV2\]
   - Windows System32 path [C:\Windows\System32]

     For example,

     PATH= C:\Program Files\Docker\Docker\resources\bin;C:\cygwin64\bin;C:\Program Files\Amazon\AWSCLIV2\;C:\Windows\System32

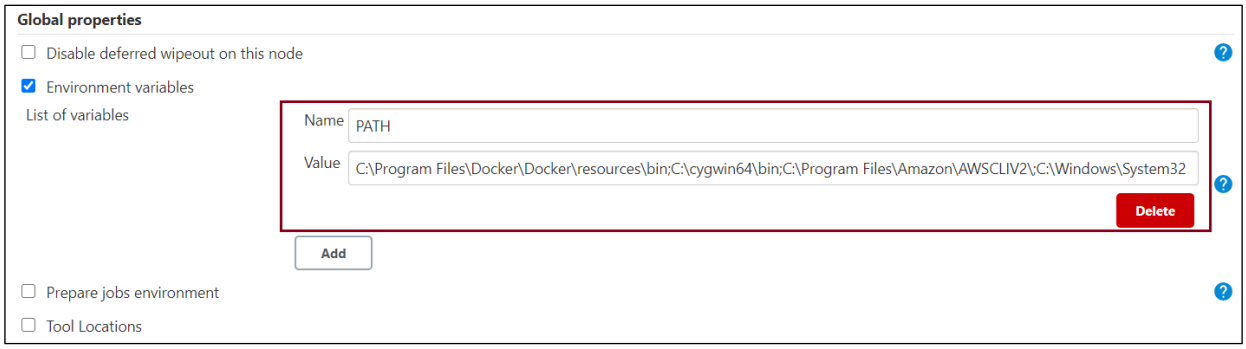

Figure3.12

5. Click **Save** to 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 **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 2 Jobs passwords: **AWS_AccessKey** and **AWS_SecretKey** and specify the AWS access key and AWS secret key of the AWS account where the container registry is created.

For example,

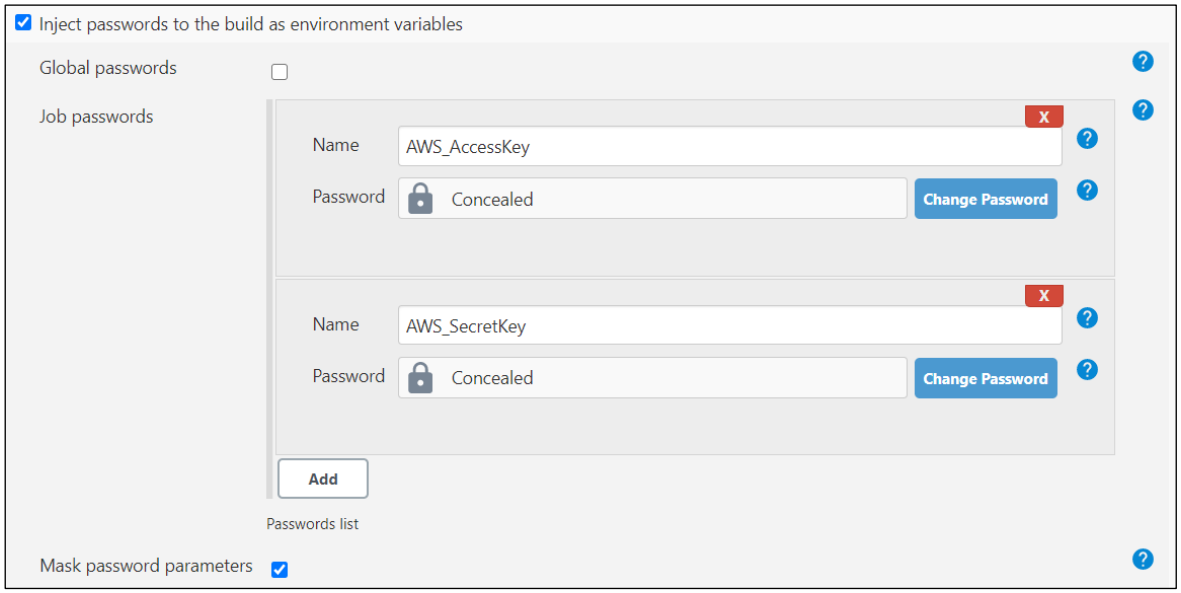

Figure3.13

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

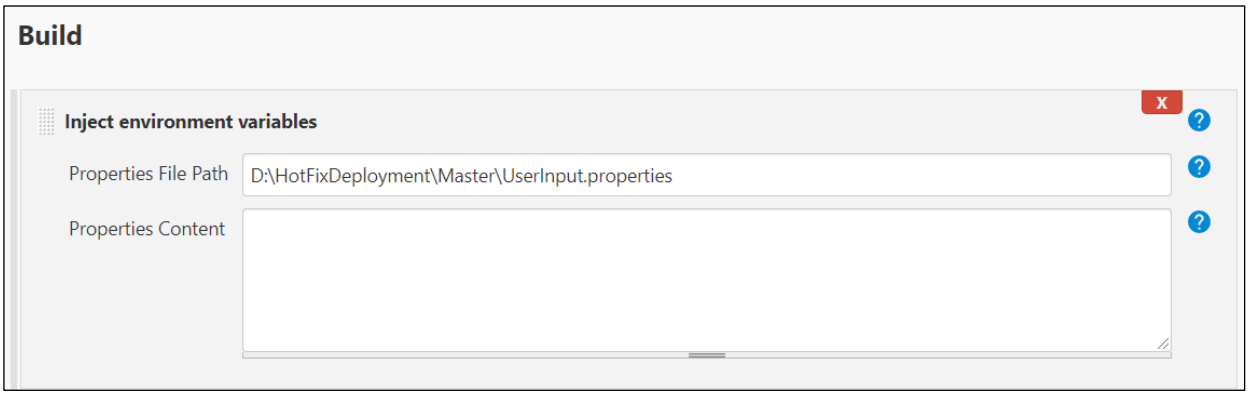

Figure3.14

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 should 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 AWS_AccessKey=%AWS_AccessKey%
set AWS_SecretKey=%AWS_SecretKey%
set AWS_AccountID=%AWS_AccountID%
set AWS_Region=%AWS_Region%
set ImageName=%OmniDocs_WEB_ImageName%
set ImageTag=%OmniDocs_WEB_Imagetag%

aws configure set aws_access_key_id %AWS_AccessKey%
aws configure set aws_secret_access_key %AWS_SecretKey%
aws ecr get-login-password --region %AWS_Region% | docker login --username AWS
--password-stdin %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com

docker pull
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%
```

For example,

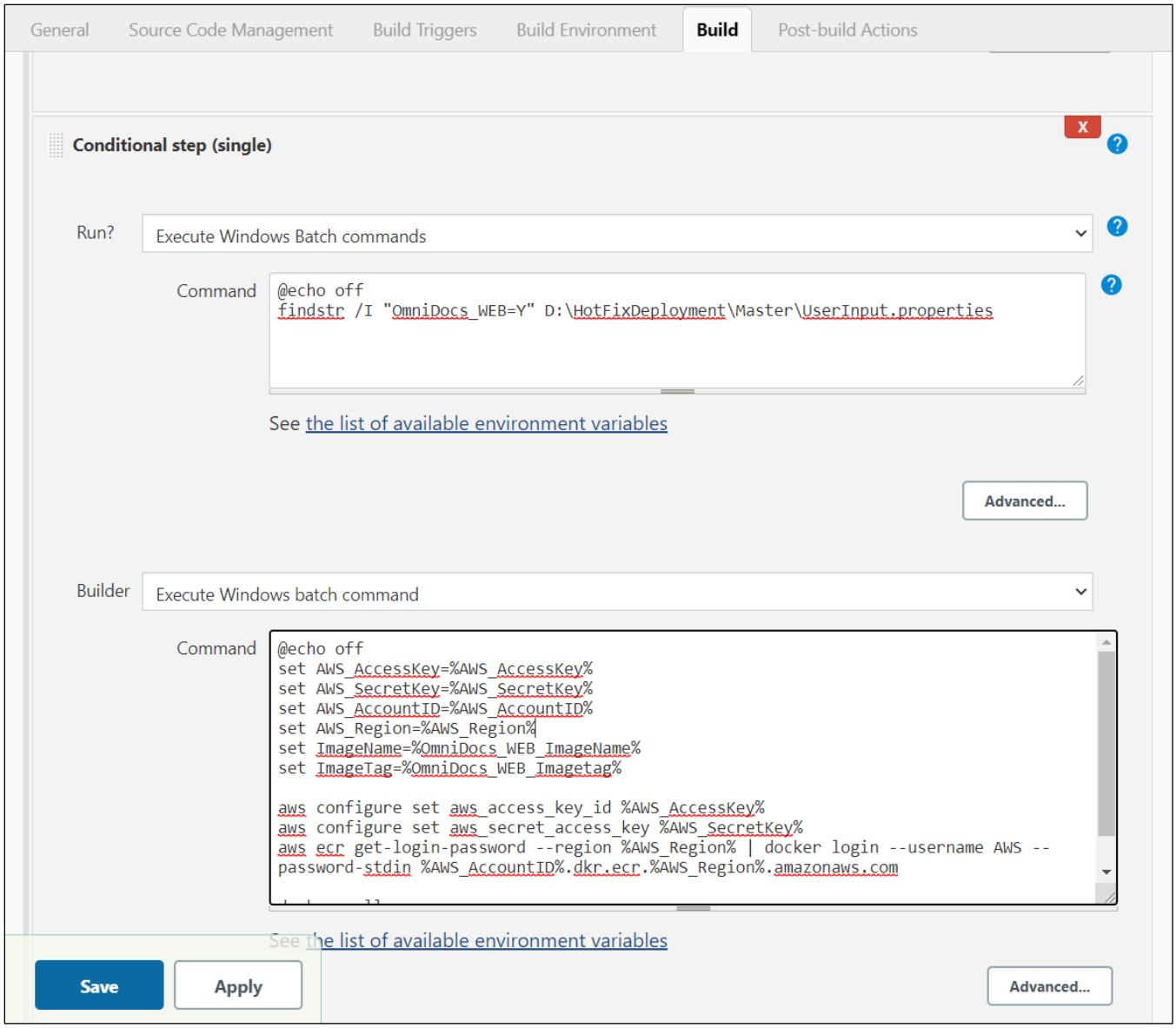

Figure3.15

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

3.2.2.2 Create Docker Image for HotFix

Perform the below steps to create the Docker 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,

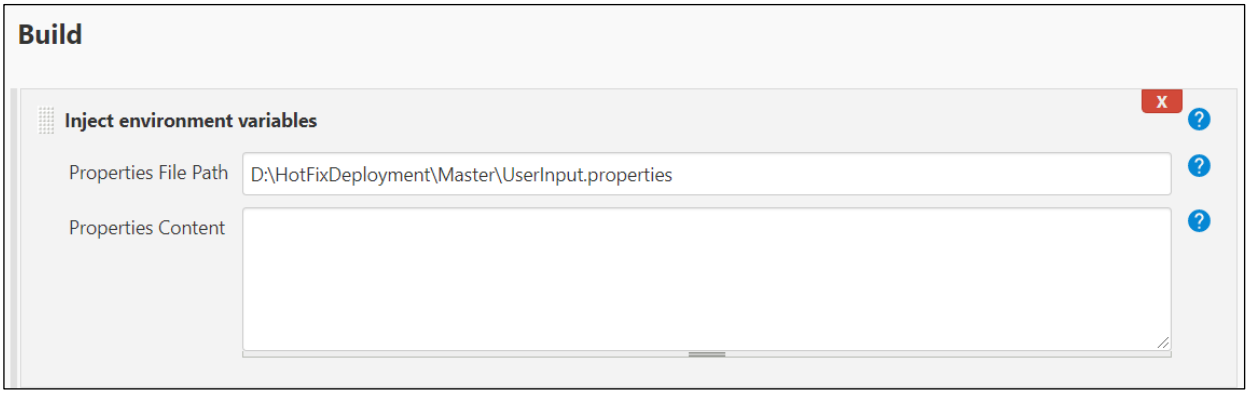

Figure3.16

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 should 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+REGISTRY_ID+%AWS_AccountID%+g %DockerFileName%_temp
   sed -i s+REGION+%AWS_Region%+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
```

For example,

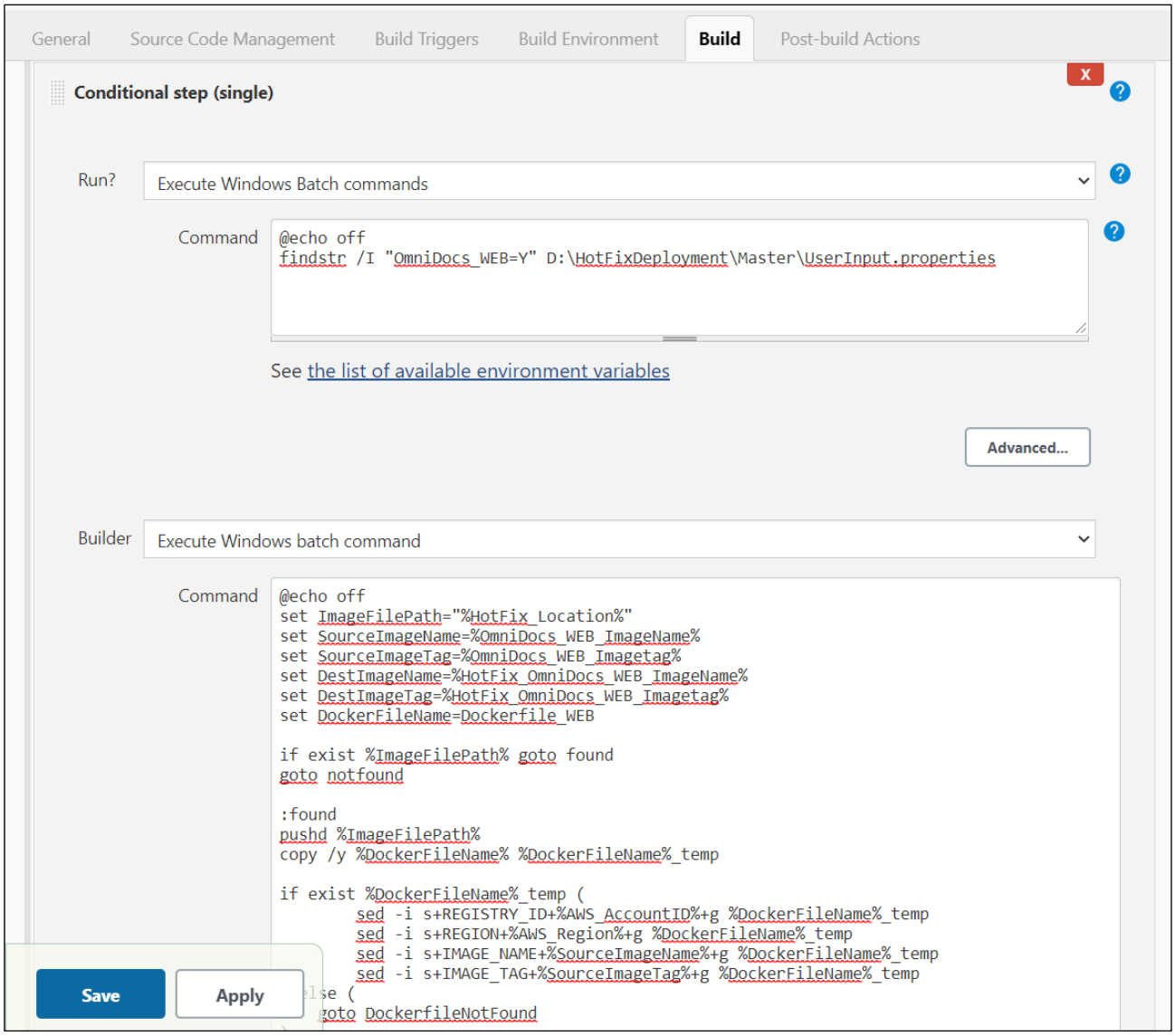

Figure3.17

10. Click **Save** to 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_Services
12. For **OmniDocs_EJB**, use Conditional steps (multiple). In OmniDocs Hotfix, the omnidocs_ejb.jar *omnidocs_ejb.ear* is received. In such a case, extract the *omnidocs_ejb.ear* from the existing Docker images, update the latest *omnidocs_ejb.jar*, and then create a new Docker 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,

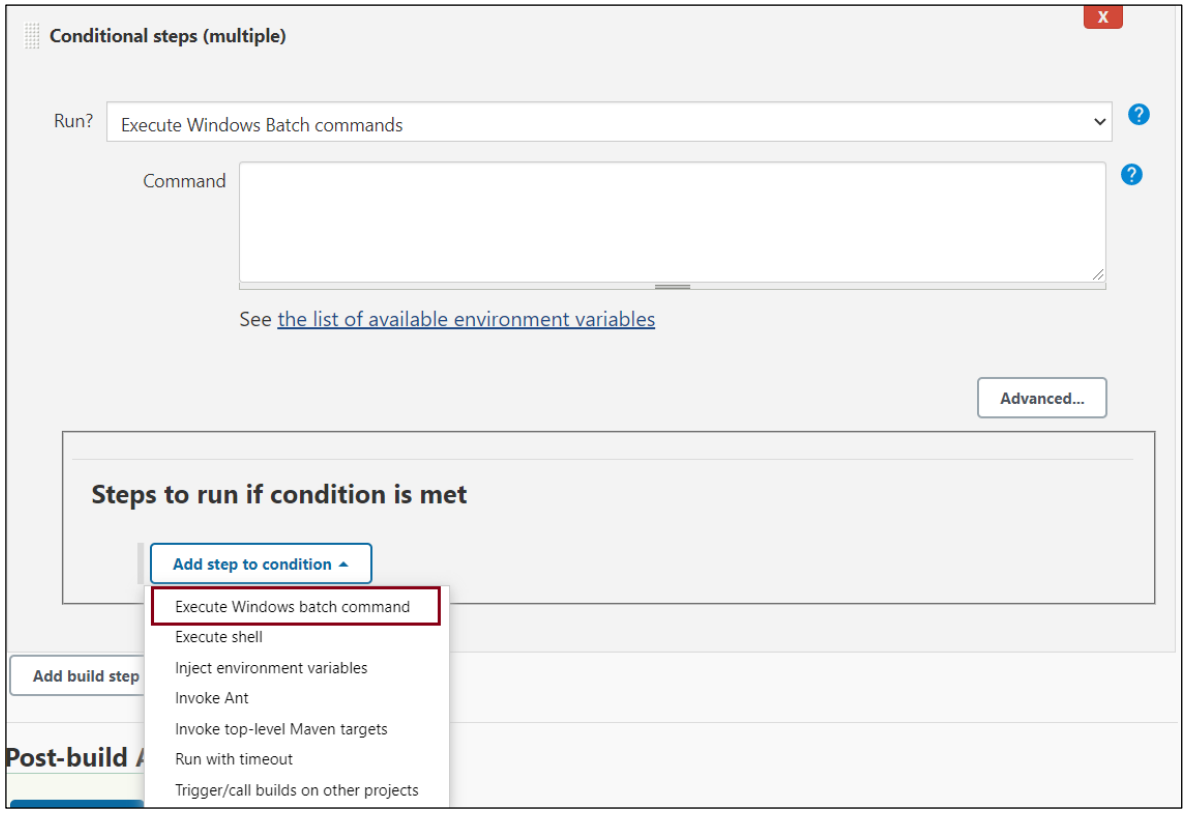

Figure 3.18

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 1st builder:

```
@echo off
for /f %%i in ('docker create
%OmniDocs_EJB_ImageName%:%OmniDocs_EJB_Imagetag%') do set RESULT=%%i
set 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 pushd %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 2nd 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+REGISTRY_ID+%AWS_AccountID%+g %DockerFileName%_temp
      sed -i s+REGION+%AWS_Region%+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** to save the changes.

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 2 Job passwords: **AWS_AccessKey** and **AWS_SecretKey** and specify the AWS access key and AWS secret key of the AWS account where the container registry is created.
   For example,

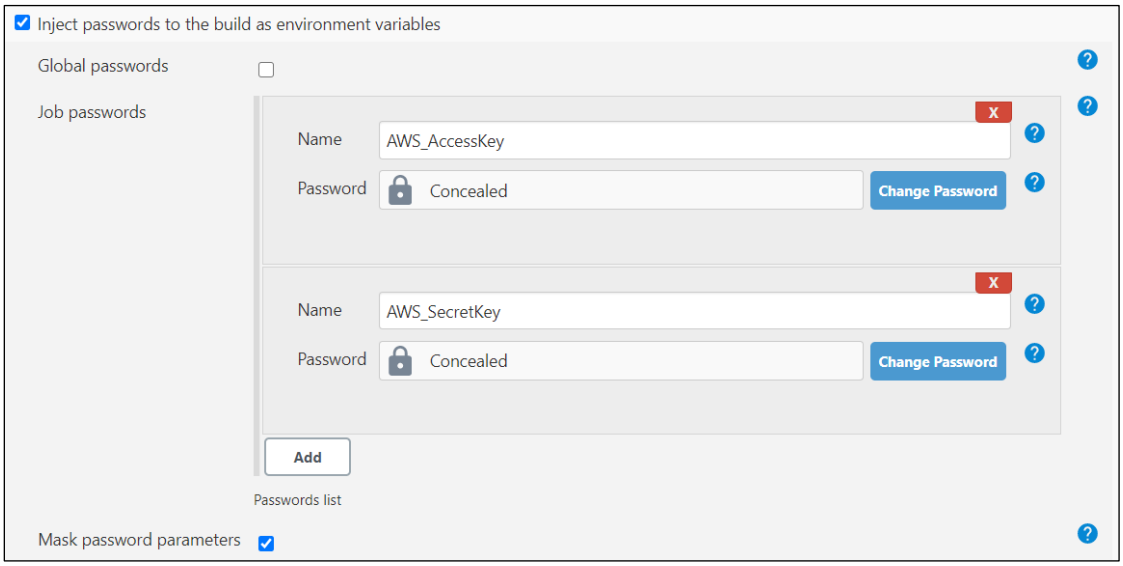

Figure 3.19

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

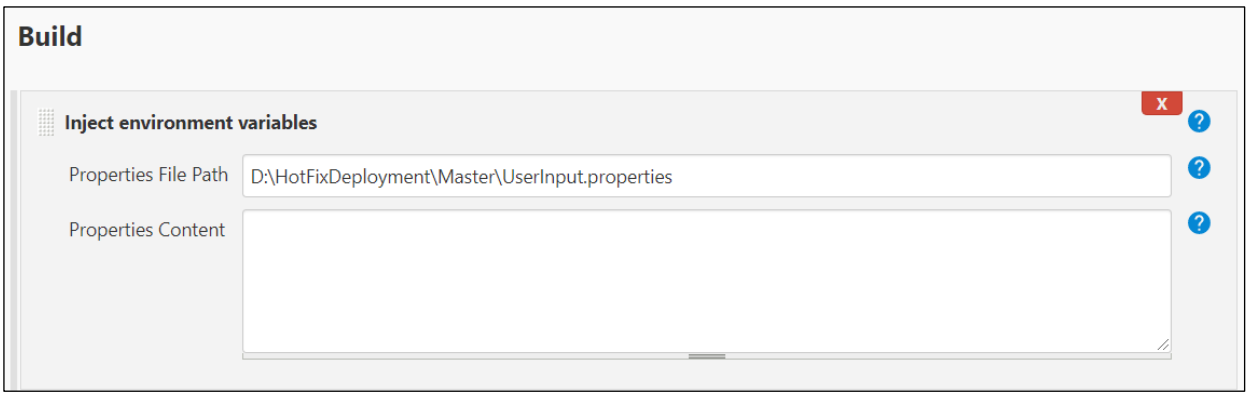

Figure 3.20

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 AWS_AccessKey=%AWS_AccessKey%
set AWS_SecretKey=%AWS_SecretKey%
set AWS_AccountID=%AWS_AccountID%
set AWS_Region=%AWS_Region%
set ImageName=%HotFix_OmniDocs_WEB_ImageName%
set ImageTag=%HotFix_OmniDocs_WEB_Imagetag%
set BuildNumber=%ImageTag%-build-%BUILD_NUMBER%

aws configure set aws_access_key_id %AWS_AccessKey%
aws configure set aws_secret_access_key %AWS_SecretKey%
aws ecr get-login-password --region %AWS_Region% | docker login --username AWS
--password-stdin %AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com
aws ecr describe-repositories --repository-names %ImageName% || aws ecr
create-repository --repository-name %ImageName% --image-scanning-configuration
scanOnPush=true

docker tag %ImageName%:%ImageTag%
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%
```

```
docker push
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%ImageTag%

docker tag %ImageName%:%ImageTag%
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%BuildNumber%
docker push
%AWS_AccountID%.dkr.ecr.%AWS_Region%.amazonaws.com/%ImageName%:%BuildNumber%
```

For example,

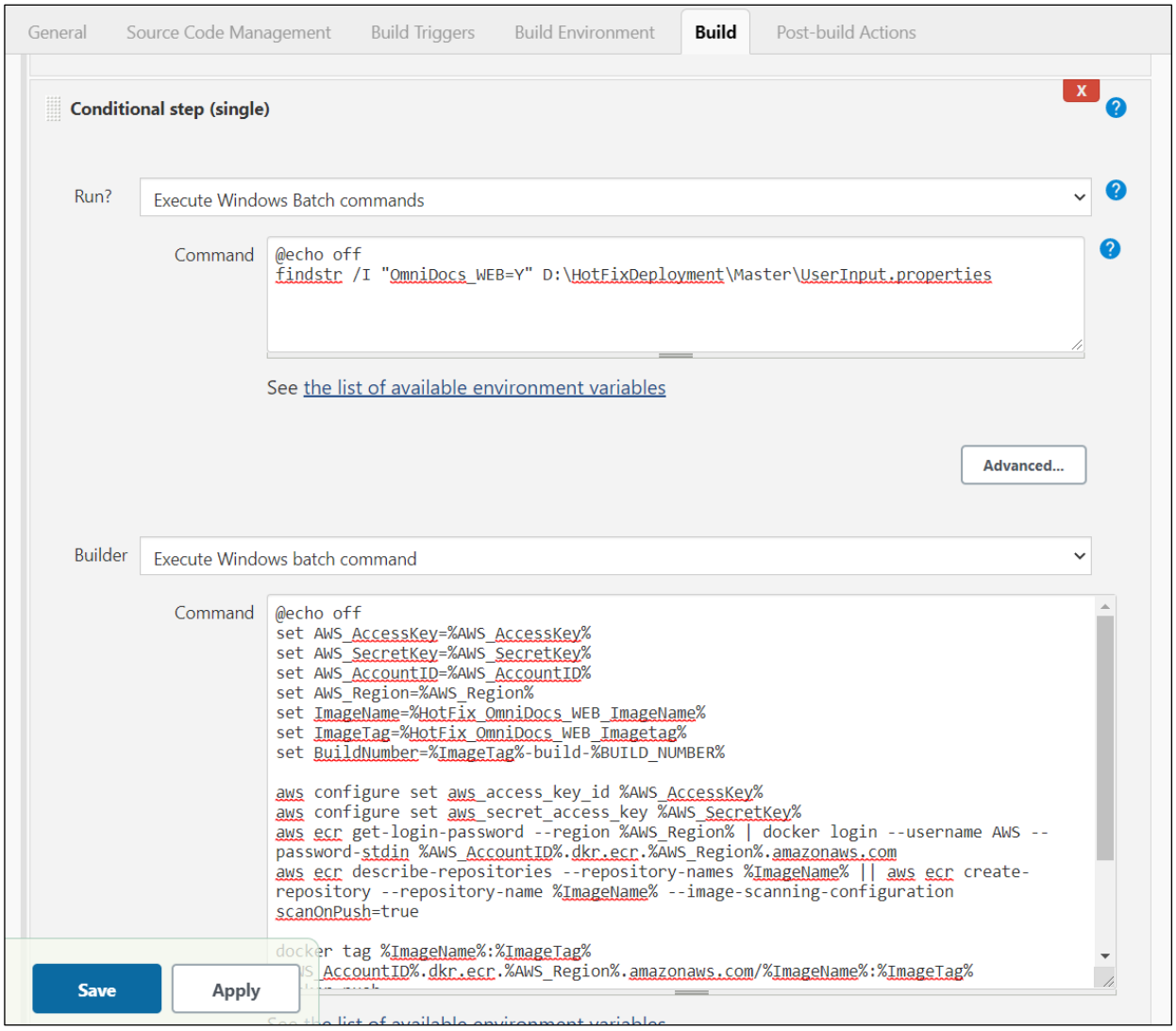

Figure 3.21

12. Click **Save** to 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

Appendix

This guide contains third-party product information about configuring Amazon Web Services (AWS) CodePipeline for Container Deployment on EKS and AWS Kubernetes Cluster. Newgen Software Technologies Ltd does not claim any ownership on such third-party content. This information is shared in this guide only for convenience of our users and could be an excerpt from the AWS documentation. For latest information on configuring the AWS Kubernetes Cluster and AWS CodePipeline refer to the AWS documentation.