NewgenONE OmniDocs

Docker Containers Hotfix Deployment Guide for Azure

Version: 11.3

Newgen Software Technologies Ltd.

Table of Contents

1 Preface

This guide describes the hotfix deployment approach for container based OmniDocs on the Azure. OmniDocs is Newgen's flagship product. This document also describes the end-to-end implementation of the product's hotfix deployment pipeline.

1.1 Revision history

Revision Date	Description
July 2024	Initial publication

1.2 Intended audience

This guide is intended for Cloud Administrators, System Administrators, developers, and any other seeking information about the deployment of hotfix for container based OmniDocs. The reader must be comfortable to understand 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, request you to 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 Microsoft Azure CICD Pipeline for Container Deployment on AKS Azure 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 Azure documentation. For latest information on configuring the Azure Kubernetes Cluster and Azure DevOps refer to the Azure documentation.

2 CI/CD Pipeline

The CICD pipeline is used to manage 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 Azure DevOps cloud service. In this architecture, there are three stages that is, 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

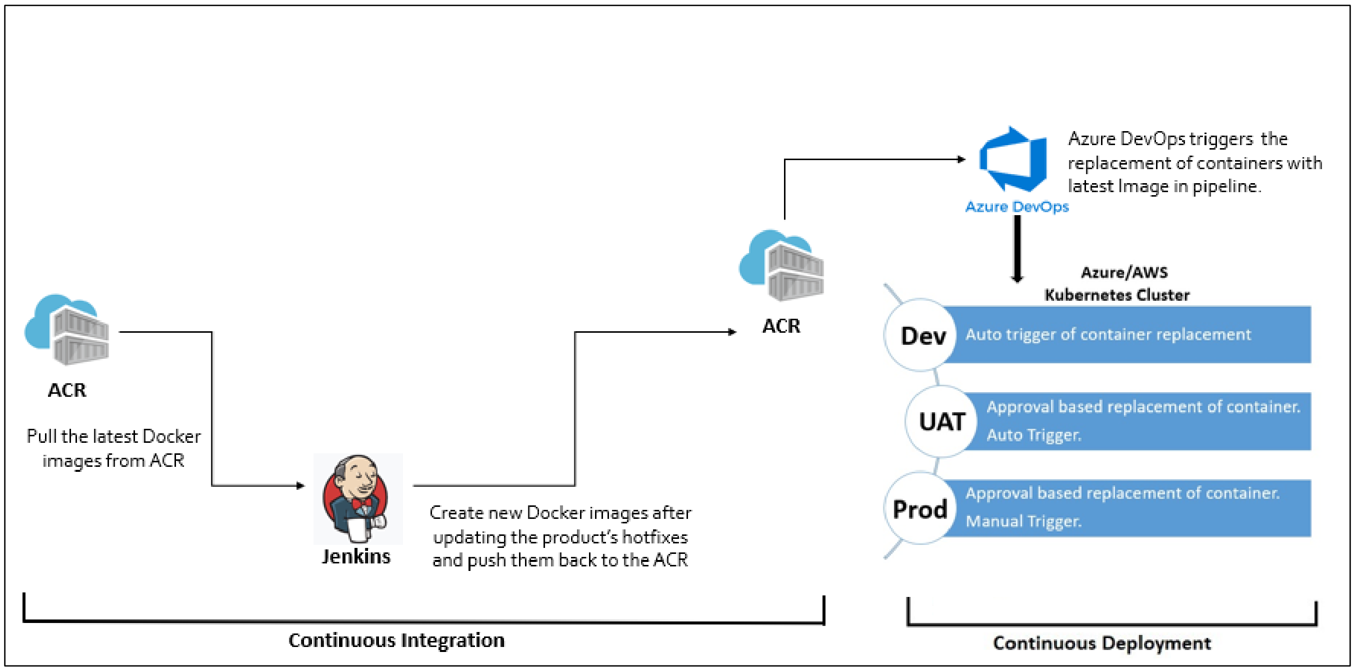

ACR
Pull the latest Docker images from ACR
Jenkins
Create new Docker images after updating the product's hotfixes and push them back to the ACR
ACR
Azure DevOps
Azure DevOps triggers the replacement of containers with latest Image in pipeline.
Azure/AWS Kubernetes Cluster
Dev
Auto trigger of container replacement
UAT
Approval based replacement of container. Auto Trigger.
Prod
Approval based replacement of container. Manual Trigger.
Continuous Integration
Continuous Deployment

Figure 2.1

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

1. Pull the product's base images or latest images that are already deployed in the current environment from the container registry.
2. 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 to update the Docker images.
3. Push the newly created images to the container registry.
4. As soon as any Docker Image is pushed to the Azure Container Registry, Azure DevOps triggers the deployment to the Dev environment.

5. UAT and Production deployments are approval based and they are called on-demand. Once you are ready to deploy to the UAT environment, trigger the UAT deployment. When that deployment is triggered, an approval mail is sent to the concerned team. Upon receiving approval, the UAT deployment starts automatically.
6. The production deployment is also approval-based, but it is multi-level approval. To deploy to a production environment, approvals from all stakeholders are required. After getting approvals from all the stakeholders, deployment to the production environment cannot be triggered automatically. A manual intervention mail is sent to deploy to production with a checklist. After evaluating whether all the checklist points are covered or not. If not, then the deployment to the production gets rejected.

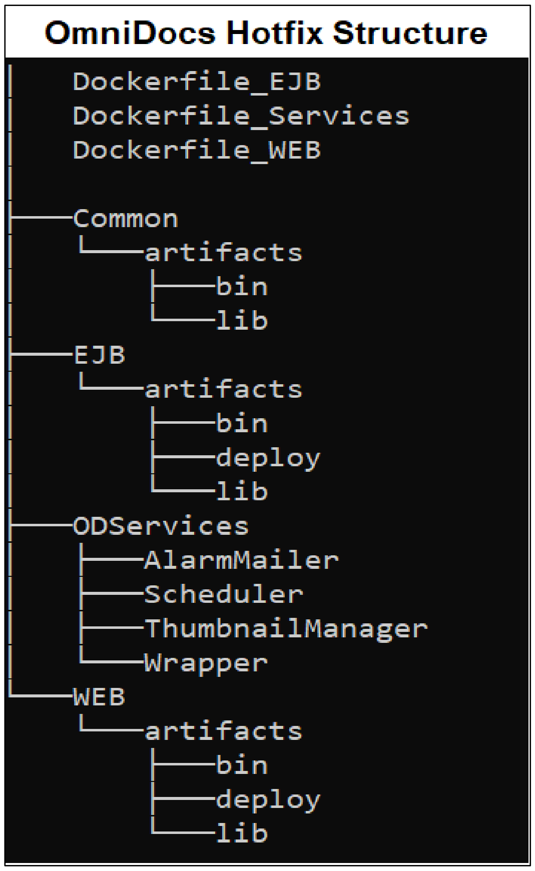
3 Implementation of Hotfix Deployment Pipeline

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 Azure DevOps.
For configuration of the Release Pipeline, refer to the *OmniDocs 11.3 Configuration and Deployment* guide.

3.1 Approach Guide for Build Pipeline

Following are the steps for an approach guide for the build pipeline:

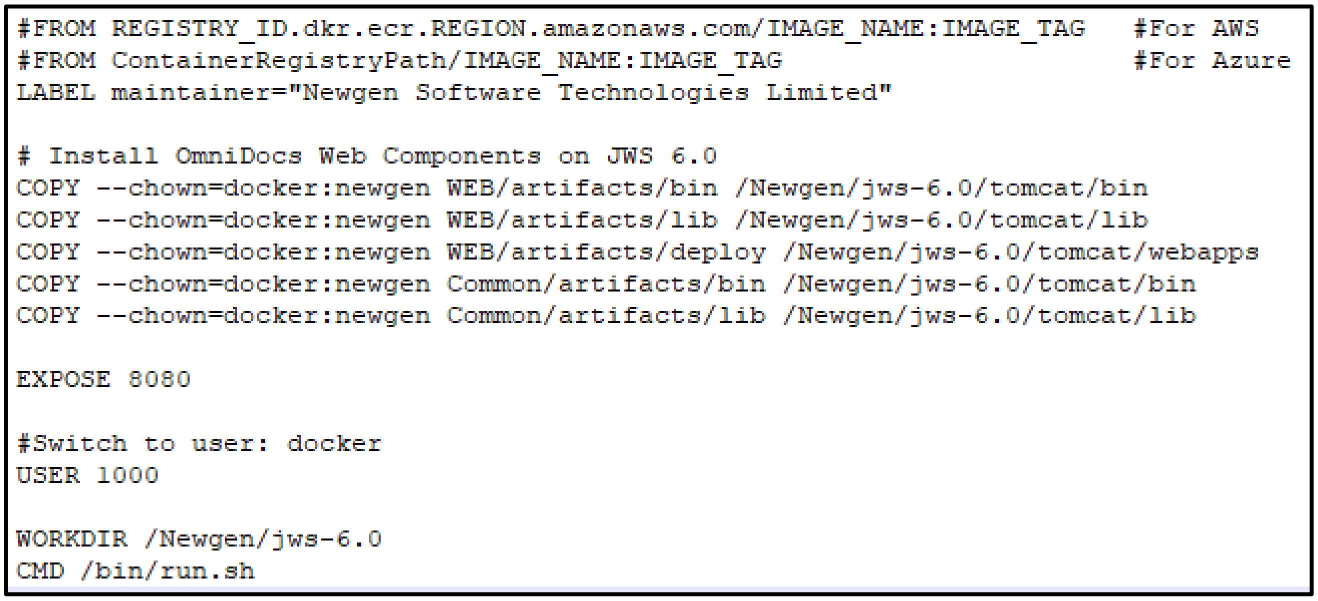
1. A pre-defined folder structure for the product's hotfix is present. For that folder structure, the hotfixes have shared.
   For example,



```
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

2. Since the WEB components and EJB components is separated into 2 parts, the Web components are deployed to the underlying WebServer JWS 6.0k.x and EJB components are deployed to the underlying AppServer JBoss EAP 7.4.x. The build binaries is 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.
1. Along with the hotfix binaries, a Dockerfile is shared for each Docker Image. 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 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

3. Docker containers have merged the deployed files and dependent libraries as there are no dynamic changes in these types of files. Also, they can be merged using Dockerfiles shared along with hotfixes.
4. Since configuration files are dynamic in nature, they must be kept outside the container. For which the volume persistence is used and mapped them to external disk storage like Azure FileShare. So, whenever configuration changes are found in a product's hotfix, update the configuration files located at external disk storage along with updating Docker images.
5. If database scripts are found in a product's hotfix's **DatabaseScripts** folder, then execute them manually through Database Client software.
6. Jenkins Build Pipeline have **three jobs**:
   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,

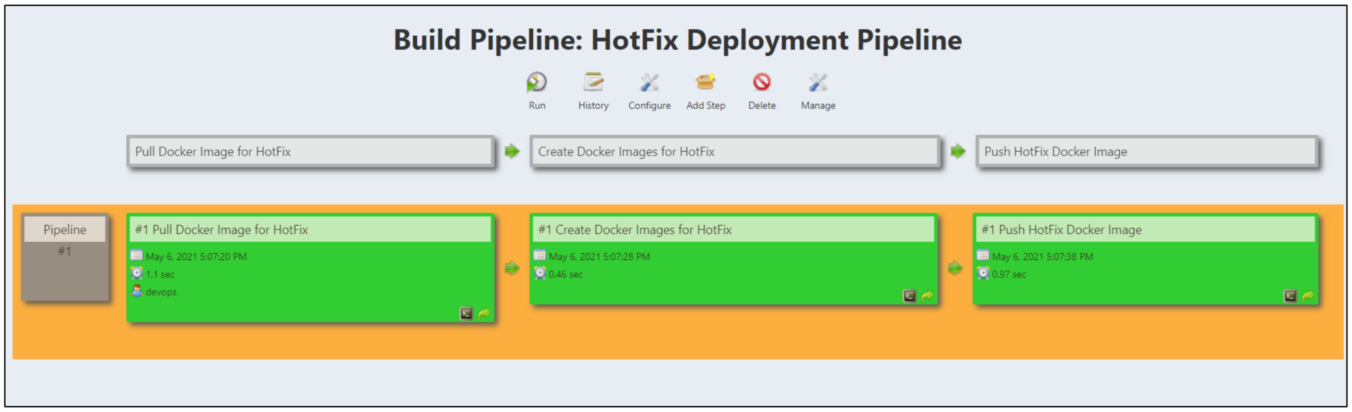
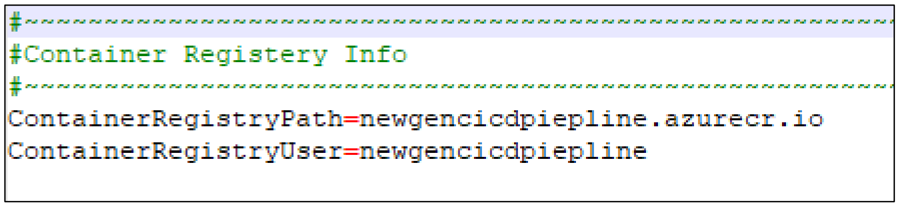

Build Pipeline: HotFix Deployment Pipeline
Run
History
Configure
Add Step
Delete
Manage
Pull Docker Image for HotFix
Create Docker Images for HotFix
Push HotFix Docker Image
Pipeline
#1
#1 Pull Docker Image for HotFix
May 6, 2021 5:07:20 PM
1.1 sec
devops
#1 Create Docker Images for HotFix
May 6, 2021 5:07:28 PM
0.46 sec
#1 Push HotFix Docker Image
May 6, 2021 5:07:38 PM
0.97 sec

Figure 3.3

7. Before pulling the latest Docker images from the container registry, Jenkins reads the *UserInput.properties* file.
8. This properties file contains all the user inputs that are required for condition-based hotfix deployment.
9. This property file has multiple sections.
   For example:
   - **#Container Registry Info**
     This section contains the container registry information. Here, provide **Azure Container Registry login server** and **Azure Container Registry username** where the container registry is created in. **Azure Container Registry password** is used as encrypted environment variables in Jenkins jobs.
     For example,

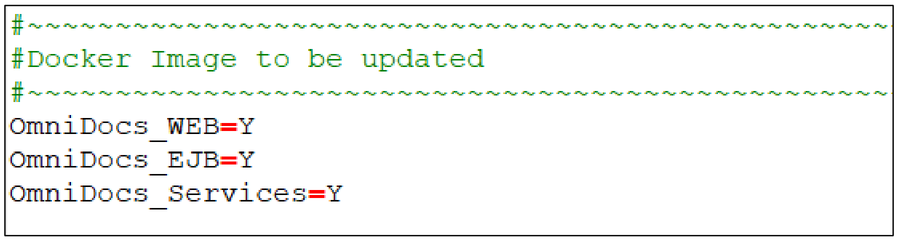


```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Container Registery Info
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ContainerRegistryPath=newgencicdpiepline.azurecr.io
ContainerRegistryUser=newgencicdpiepline
```

Figure 3.4

   - **#HotFix Info**
     This section contains the location of a hotfix that you want to deploy.
     For example,
     HotFix_Location="C:\Users\Administrator\Downloads\OD_11.0_SP0_P00_HF01"

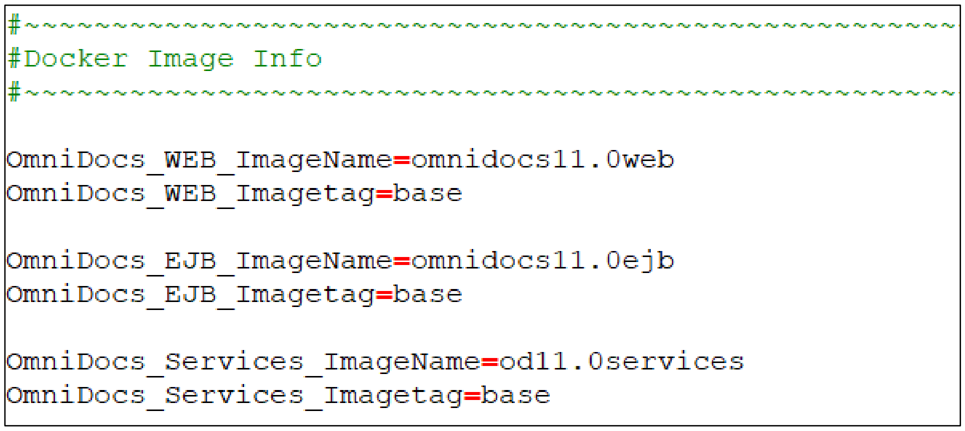
- **#Docker Image to be updated**
  In this section, select the Docker image(s) in which you want to deploy the hotfix binaries. For example, if hotfix is deployed in the OmniDocs WEB container, then set the OmniDocs_WEB=Y.
  For example,



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

Figure 3.5

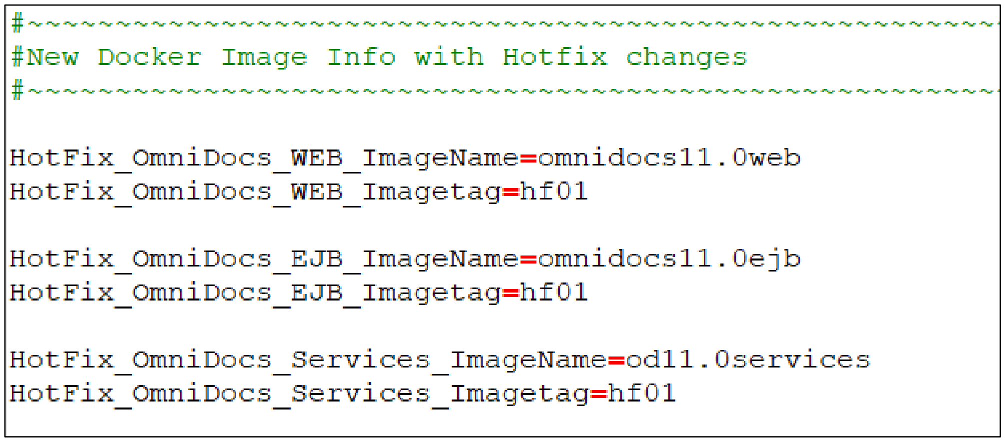
- **#Docker Image Info**
  This section contains the information about the source Docker images in which hotfix binaries is updated or deployed.
  For example,



```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#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

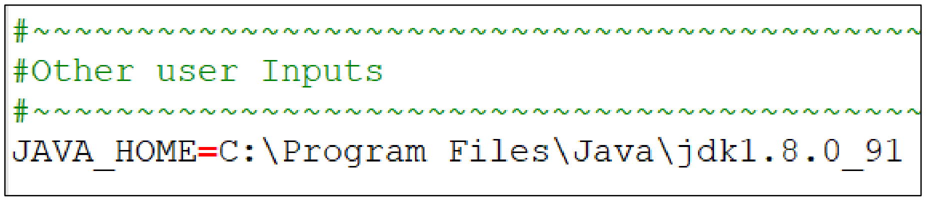
- **#New Docker Image Info with Hotfix changes**
  This section contains the information about new Docker images that are created after updating the hotfix binaries.
  For example,



```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#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

10. 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 the configuration of Jenkins for Build Pipeline.

3.2.1 Prerequisites

Prerequisites are as follows:

- **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.
- **Azure CLI** 2.28.0 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 have three jobs:

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, some server-level configurations in Jenkins are required.

1. Log in to the Jenkins Server.

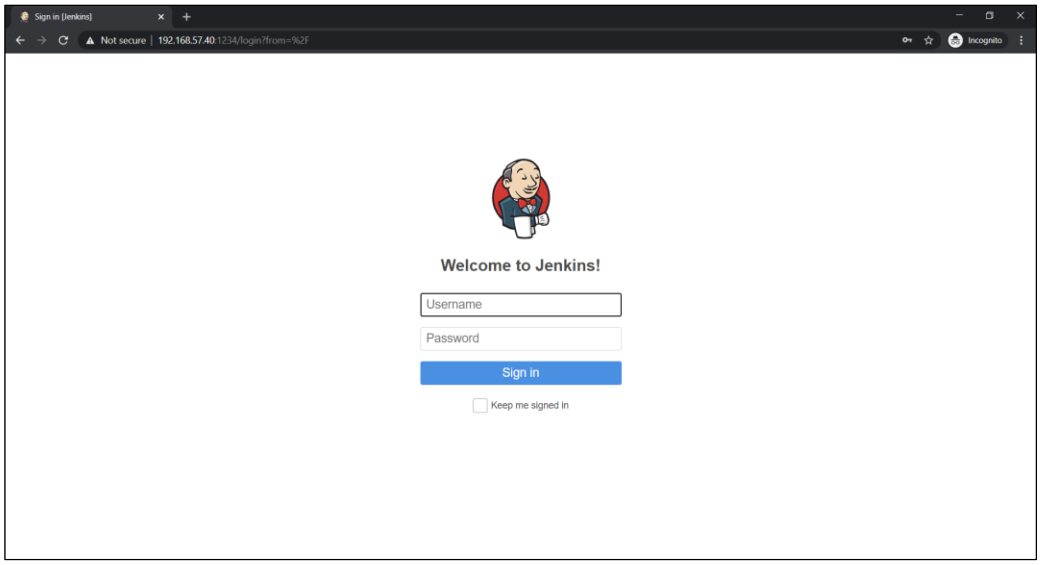


Figure 3.9

2. After successful login, click **Manage Jenkins** link.

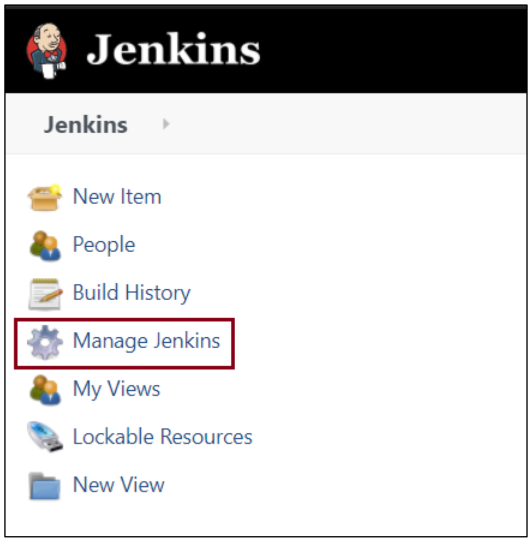
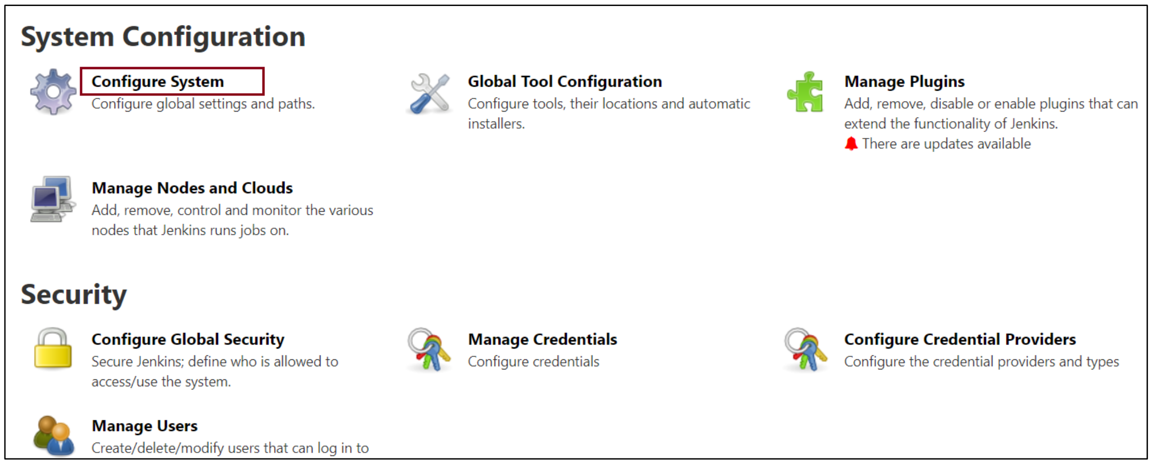


Figure 3.10

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


System Configuration

Configure System
Configure global settings and paths.

Global Tool Configuration
Configure tools, their locations and automatic installers.

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
There are updates available

Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Manage Credentials
Configure credentials

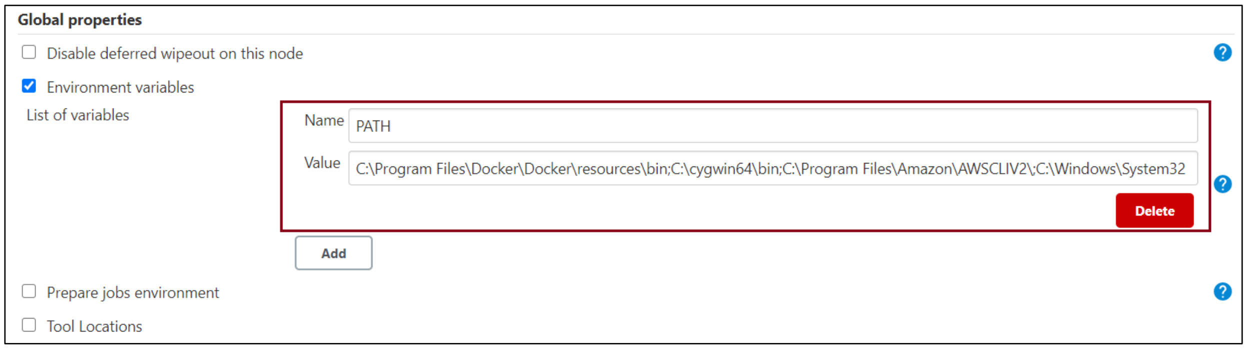
Configure Credential Providers
Configure the credential providers and types

Manage Users
Create/delete/modify users that can log in to

Figure 3.11

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]
   - Azure CLI installation path [C:\Program Files (x86)\Microsoft SDKs\Azure\CLI2\wbin]
   - Windows System32 path [C:\Windows\System32]
5. For example,

```
PATH= C:\Program Files\Docker\Docker\resources\bin;C:\cygwin64\bin;C:\Program
Files (x86)\Microsoft SDKs\Azure\CLI2\wbin;C:\Windows\System32
```


Global properties

☐ Disable deferred wipeout on this node

☑ Environment variables

List of variables

Name: PATH

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

☐ Prepare jobs environment

☐ Tool Locations

Figure 3.12

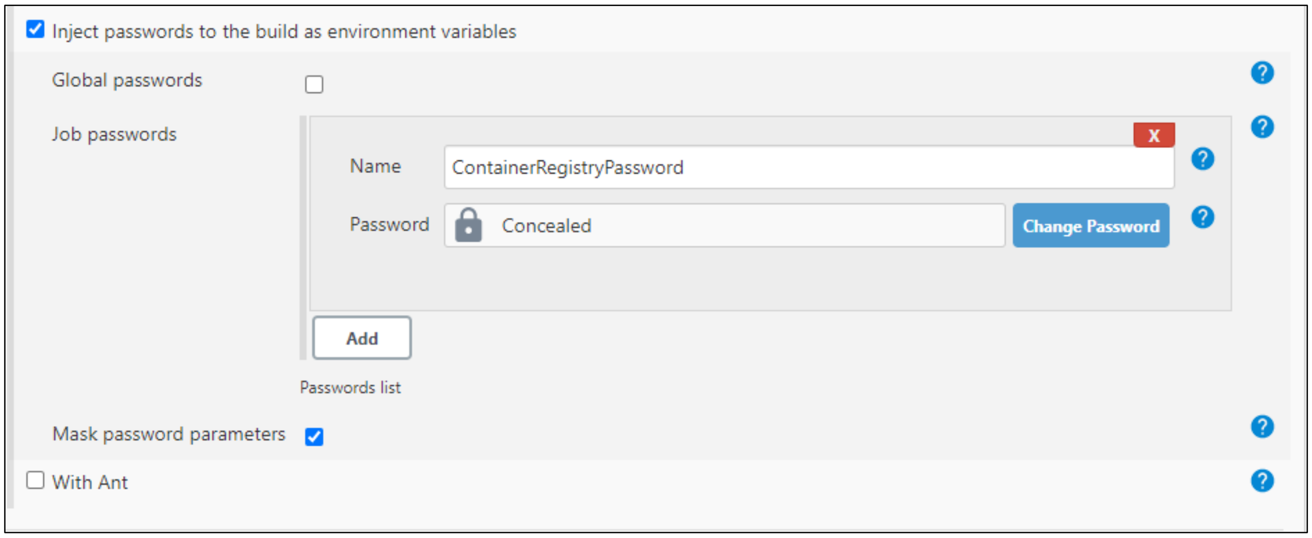
6. **Save** the changes.

3.2.2.1 Pull Docker Image for HotFix

To pull the Docker image for the hotfix, follow the below steps:

1. Click **New Item** link.
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 1 Job password: **ContainerRegistryPassword** and specify the Azure Container Registry password.
   For example,



☑ Inject passwords to the build as environment variables

Global passwords ☐

Job passwords

Name: ContainerRegistryPassword

Password: Concealed

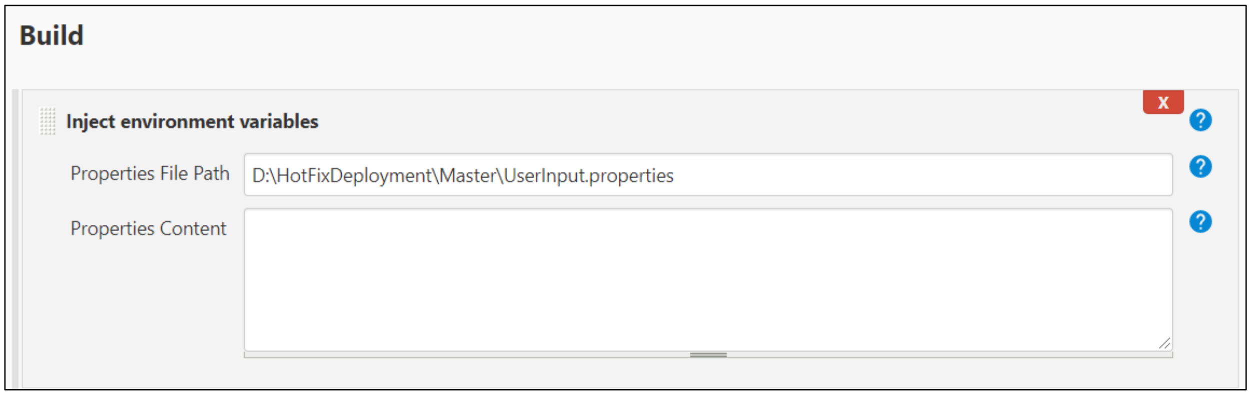
Passwords list

Mask password parameters ☑

☐ With Ant

Figure 3.13

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



Build

Inject environment variables

Properties File Path: D:\HotFixDeployment\Master\UserInput.properties

Properties Content

Figure 3.14

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 must run or not].

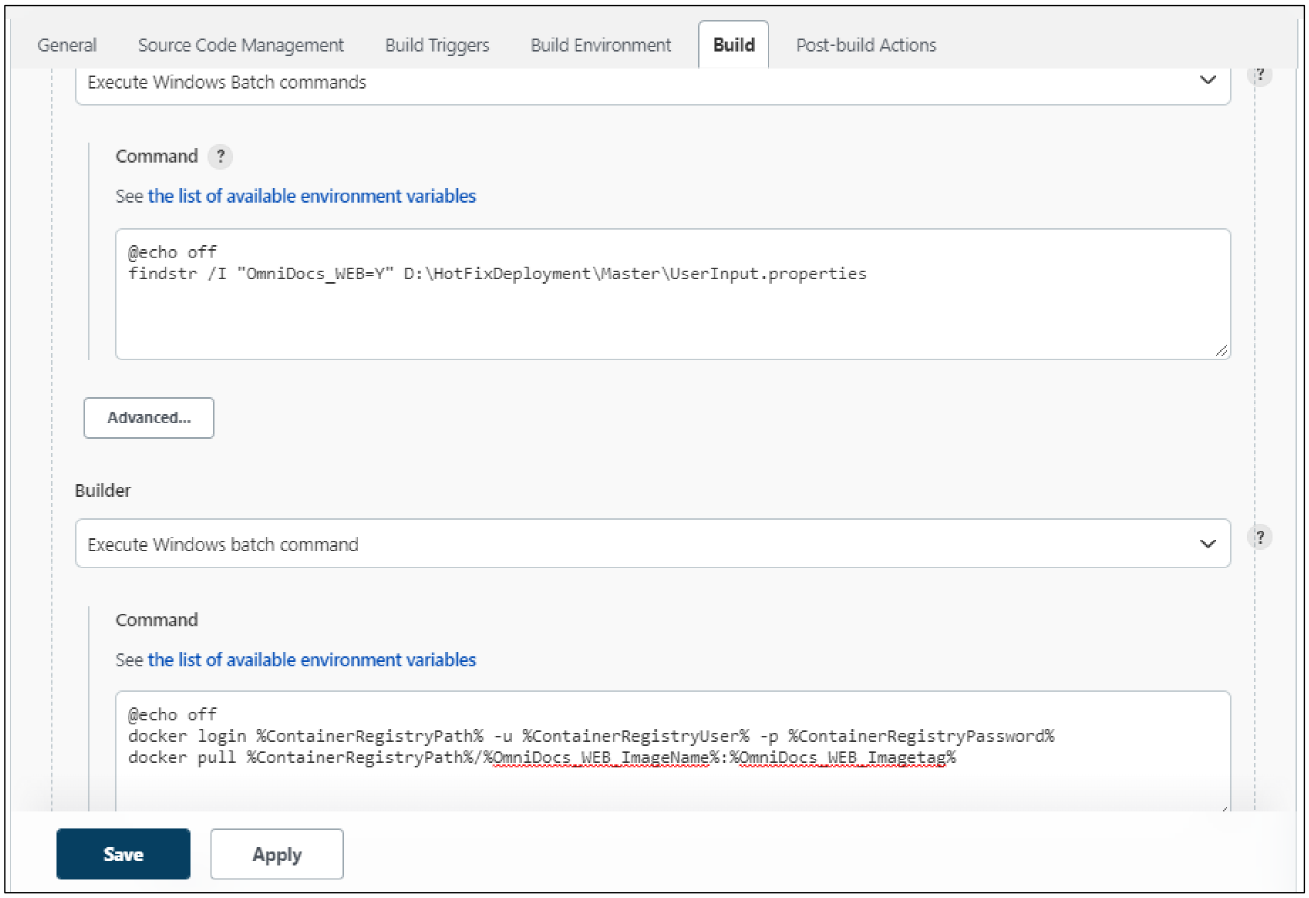
10. Specify the following command for the condition:

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

11. Specify the following commands for the builder:

```
@echo off
docker login %ContainerRegistryPath% -u %ContainerRegistryUser% -p
%ContainerRegistryPassword%
docker pull
%ContainerRegistryPath%/%OmniDocs_WEB_ImageName%:%OmniDocs_WEB_Imagetag%
```

For example,


General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Execute Windows Batch commands
Command
See the list of available environment variables
Advanced...
Builder
Execute Windows batch command
Command
See the list of available environment variables
Save
Apply

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

```
@echo off
docker login %ContainerRegistryPath% -u %ContainerRegistryUser% -p %ContainerRegistryPassword%
docker pull %ContainerRegistryPath%/%OmniDocs_WEB_ImageName%:%OmniDocs_WEB_Imagetag%
```

Figure 3.15

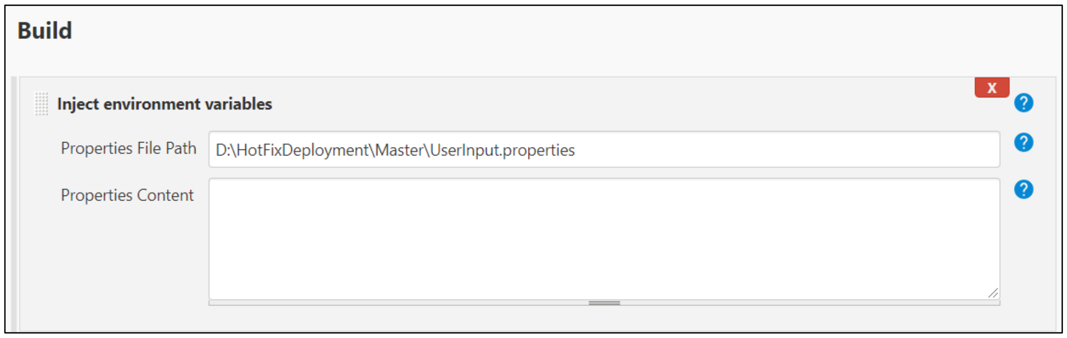
12. Click **Save** to save the changes. Through above steps, you have performed Hotfix for **OmniDocs_WEB** Docker image.

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 Create Docker Image for HotFix

To create the Docker image for Hotfix, follow the below steps:

1. Click **New Item** link showing.
2. Specify the item name or job name and select the project type as **Freestyle project**.
3. 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,



Build

Inject environment variables

Properties File Path: D:\HotFixDeployment\Master\UserInput.properties

Properties Content:

Figure 3.16

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 must run or not].
8. Specify the following command for the condition:

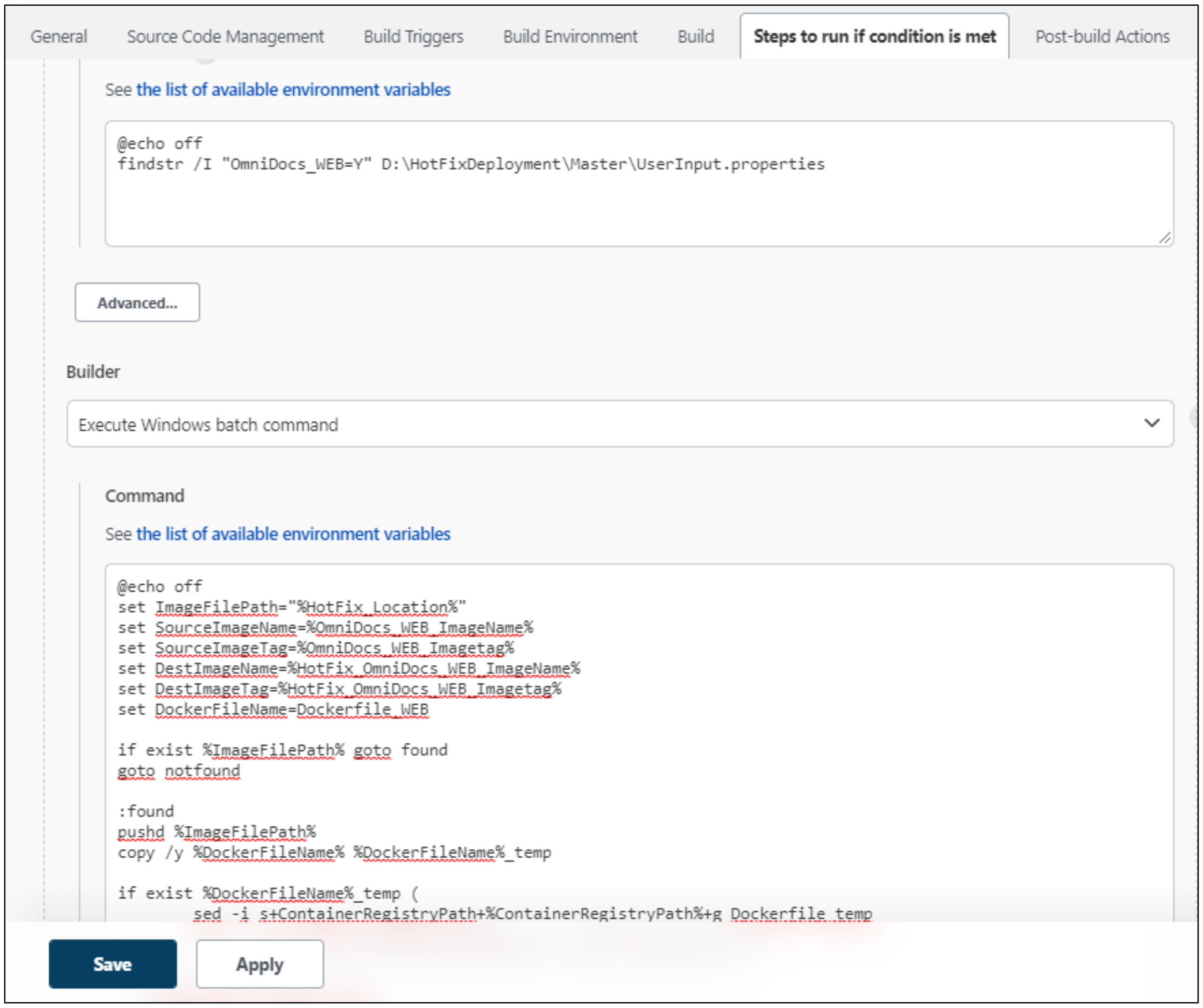
```
@echo off
findstr /I "OmniDocs_WEB=Y" D:\HotFixDeployment\Master\UserInput.properties
```

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+ContainerRegistryPath+%ContainerRegistryPath%+g Dockerfile_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,



See the list of available environment variables

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

Builder

Execute Windows batch command

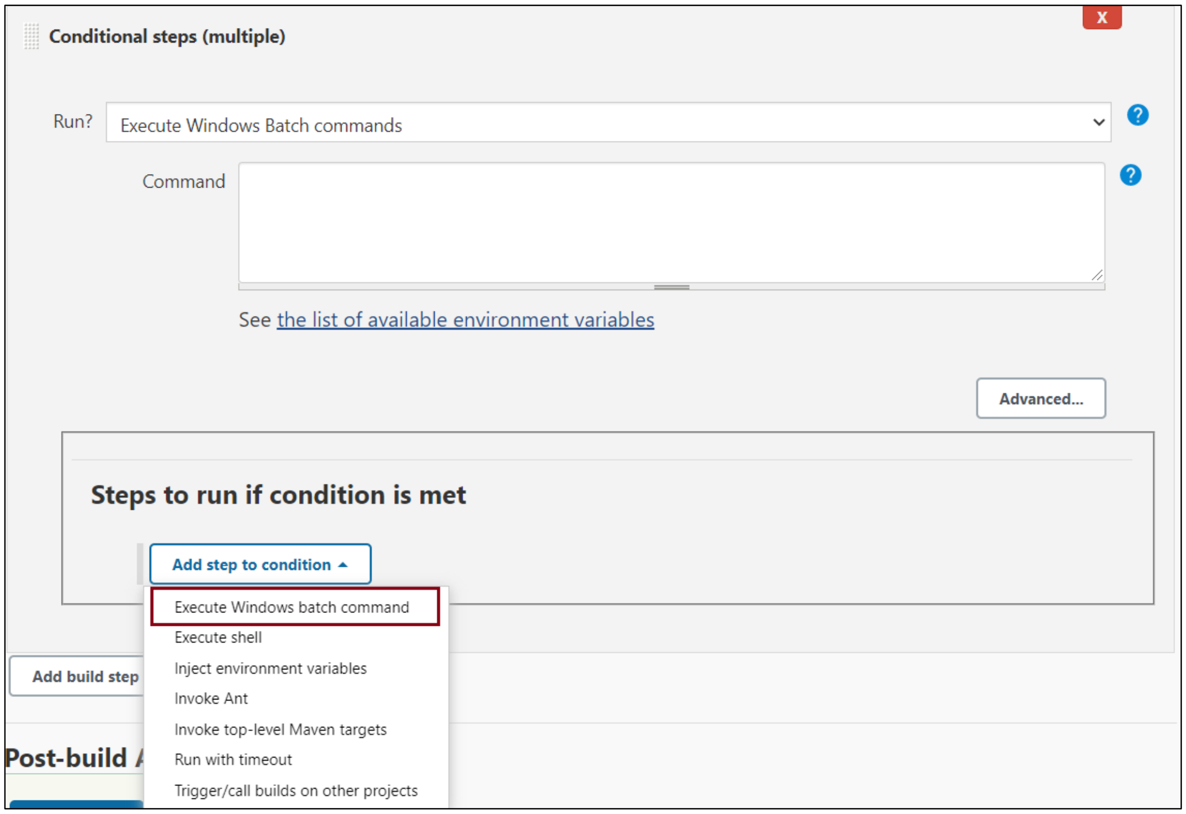
Command

See the list of available environment variables

```
@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+ContainerRegistryPath+%ContainerRegistryPath%+g Dockerfile_temp
```

Figure 3.17

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) as 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 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. Select **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,


Conditional steps (multiple)

Run? Execute Windows Batch commands

Command

See the list of available environment variables

Steps to run if condition is met

X
Advanced...
Add step to condition
Execute Windows batch command
Execute shell
Inject environment variables
Invoke Ant
Invoke top-level Maven targets
Run with timeout
Trigger/call builds on other projects
Add build step
Post-build A

Figure 2.18

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+ContainerRegistryPath+%ContainerRegistryPath%+g Dockerfile_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
```

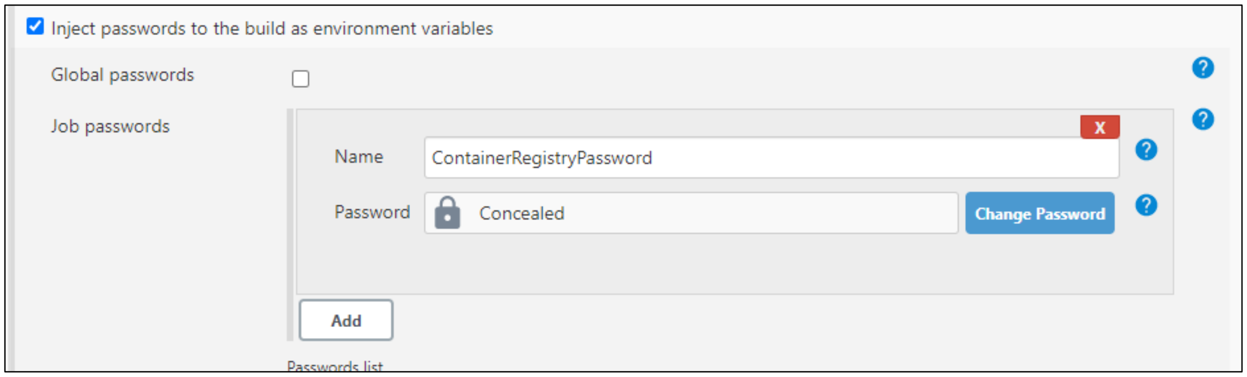
```
echo "%DockerFileName%_temp does not exist."
goto finish

: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 Dicker image:

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** under the **Build Environment** section.
5. Specify 1 Job password: **ContainerRegistryPassword** and specify the Azure Container Registry password.
   For example,



☑ Inject passwords to the build as environment variables

Global passwords ☐

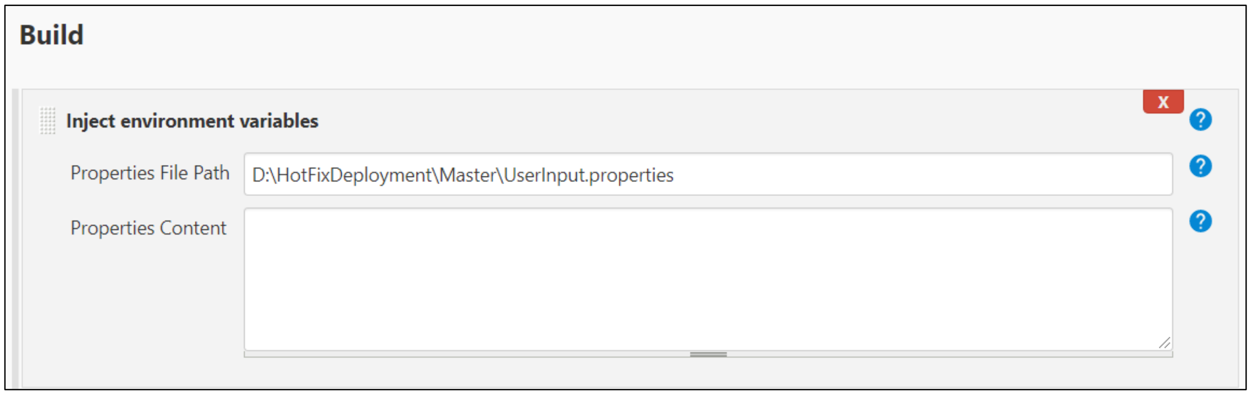
Job passwords

Name: ContainerRegistryPassword

Password: Concealed

Figure 2.19

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


Build

Inject environment variables

Properties File Path: D:\HotFixDeployment\Master\UserInput.properties

Properties Content

Figure 2.20

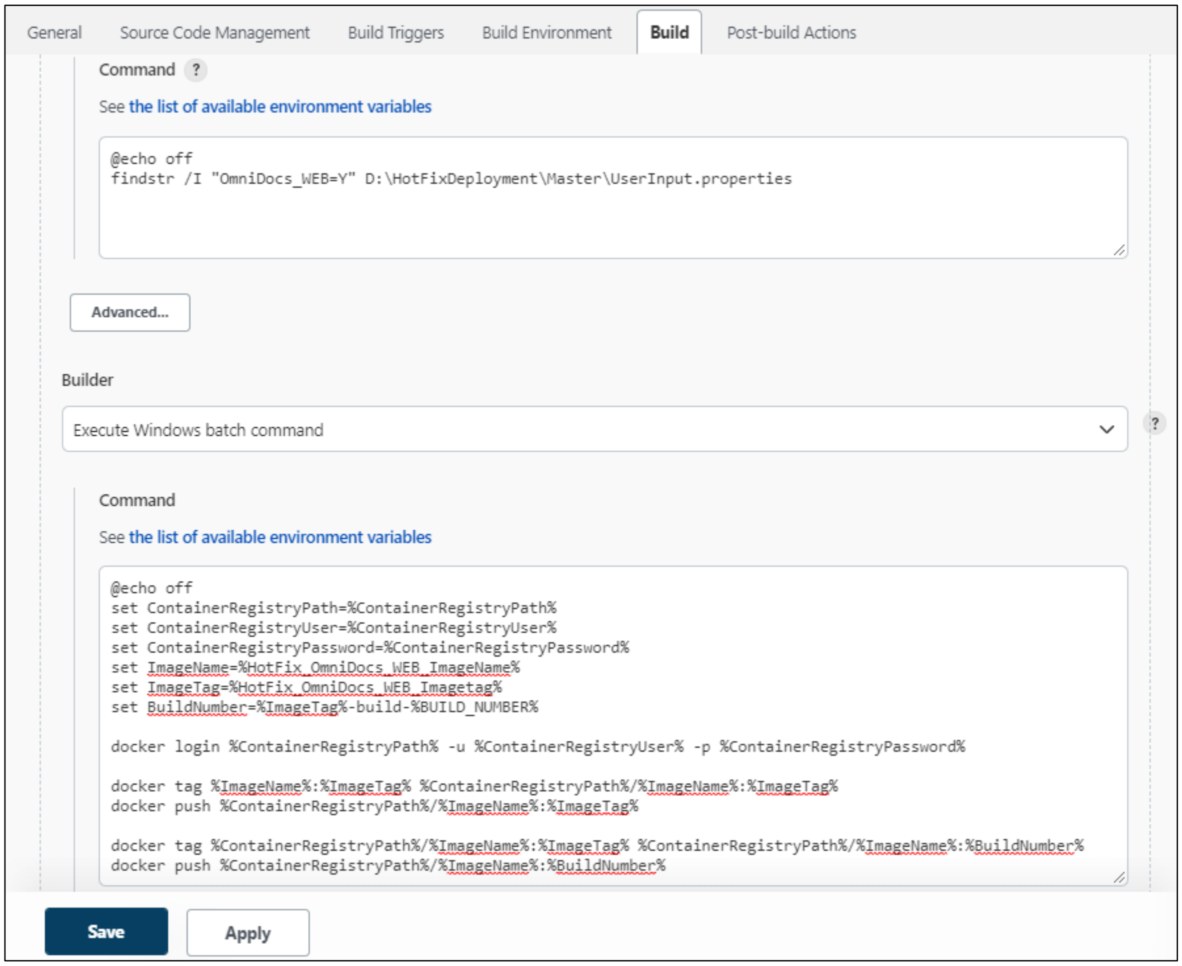
8. Add **Conditional step (single)** as a build step task under the **Build** section.
9. Select **Execute Windows batch command** as **Run?** and **Builder**. [‘Run?’ is a condition to decide whether a ‘builder’ command must run or not].
10. Specify the following command for the condition:

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

11. Specify the following commands for the builder:

```
@echo off
set ContainerRegistryPath=%ContainerRegistryPath%
set ContainerRegistryUser=%ContainerRegistryUser%
set ContainerRegistryPassword=%ContainerRegistryPassword%
set ImageName=%HotFix_OmniDocs_WEB_ImageName%
set ImageTag=%HotFix_OmniDocs_WEB_Imagetag%
set BuildNumber=%ImageTag%-build-%BUILD_NUMBER%

docker login %ContainerRegistryPath% -u %ContainerRegistryUser% -p
%ContainerRegistryPassword%

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

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

For example,



Command ?

See the list of available environment variables

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

Builder

Execute Windows batch command

Command

See the list of available environment variables

```
@echo off
set ContainerRegistryPath=%ContainerRegistryPath%
set ContainerRegistryUser=%ContainerRegistryUser%
set ContainerRegistryPassword=%ContainerRegistryPassword%
set ImageName=%HotFix_OmniDocs_WEB_ImageName%
set ImageTag=%HotFix_OmniDocs_WEB_Imagetag%
set BuildNumber=%ImageTag%-build-%BUILD_NUMBER%

docker login %ContainerRegistryPath% -u %ContainerRegistryUser% -p %ContainerRegistryPassword%

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

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

Figure 2.21

12. **Save** the changes.
13. Here, set 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