

NewgenONE OmniDocs RMS

Docker Containers Custom Code Deployment Guide for Azure

Version: 4.0 SP1

Newgen Software Technologies Ltd.

This document contains propriety information of NSTL. No part of this document may be reproduced, stored, copied, or transmitted in any form or by any means of electronic, mechanical, photocopying, or otherwise, without the consent of NSTL.

Table of Contents

1	Pref	ace	2
	1.1	Revision history	2
	1.2	Intended audience	2
	1.3	Documentation feedback	2
2	CI/C	D pipeline	.3
	2.1	CI/CD Pipeline for Custom Code	3
3	Imp	lementing custom code deployment pipeline	.4
	3.1	Approach guide for build pipeline	4
	3.2	Configuring Jenkins for build pipeline	8
	3.2	.1 Prerequisites	8
	3.2	.2 Configuring Jenkins jobs	8
Ap	pend	ix	24

1 Preface

This guide describes the containerization approach for Newgen Enterprise products, which thus details out the container-based deployment architecture and container-based DevOps pipeline for NewgenONE flagship products OmniDocs Record Management System on Microsoft Azure.

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 OmniDocsRMS. 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 <u>docs.feedback@newgensoft.com</u>.

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 Azure DevOps 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 CI/CD Pipeline for Custom Code

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



Figure 2.1

- The Custom Code deployment pipeline can also be configured and maintained in Jenkins.
- For initiating custom code deployment, the Implementation team pushes the compiled custom code to the artifacts repository. An artifacts repository can be Azure FileShare, SVN, FTP, JFrog, and so on.

- After that Jenkins pulls the compiled custom code from the artifacts repository and pull the latest Docker image. Then, it creates a new Docker after merging the custom code changes and pushes the newly created Docker images to Image Registry.
- In this architecture, Cloud or Infra team have full access to initiate the build pipeline configured on the Jenkins server. The Implementation team has no have access to this Jenkins server. However, a common artifacts repository is shared for both teams.
- As soon as any Docker image is pushed to the Azure Container Registry, Azure DevOps 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 Implementing custom code deployment pipeline

The custom code deployment pipeline is separated into two parts: **Build Pipeline** and **Release Pipeline**. Build Pipeline is configured on the Jenkins Server and Release Pipeline is configured on the AzureDevOps.

For configuration on Release Pipeline, refer to the *Configuration of Azure DevOps Release Pipeline for AKS* document.

3.1 Approach guide for build pipeline

Perform the below steps to build pipeline:

- There's 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.
- 2. It covers 3 different types of custom codes in OmniDocs.

- OmniDocs_CustomCode
- omniDocs_Hook
- WebAPI_CustomCode
- 3. This folder structure is available in the artifacts repository. An artifacts repository can be like Azure FileShare, SVN, FTP, and so on. SVN is used in this implementation.

- 4. Only this folder structure defined in the artifacts repository is accessible from the Implementation team.
- 5. Jenkins Build Pipeline has the **5 jobs** that are as follows:
 - 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,

		Build Pipelin	e: Cust	tom Code Deplo	oyr	nent Pipeline			
😥 🔄 🌌 😁 🚫 💥 Run History Configure Add Step Delete Manage									
	Pull Custom Code Binaries	Pull Docker Image for Custom	iode 🔶 🔶	Merge Custom Code Changes	•	Build CustomCode Docker Image	•	Push CustomCode Docker Image	
Pipeline #4	#4 Pull Custom Code Binaries	#39 Pull Docker Image for CustomCode	٦	#42 Merge Custom Code Changes	l	#25 Build CustomCode Docker Image	l	#22 Push CustomCode Docker Image	
	Feb 18. 2021 7:20:26 PM 0.27 sec devops	Feb 18, 2021 7:20:36 PM	÷	🔲 Feb 18. 2021 7:20:51 PM 🔽 39 sec	*	 Feb 18, 2021 7:21:41 PM 1 min 1 sec 	۲	Eeb 18. 2021 7:22:51 PM	
				Figure 2.4					



- 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 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,





• #Custom code changes to be deployed

In this section, select the components to deploy the custom code. You can select or deselect the components by setting the component's value as Y or N respectively. For example,





#Custom code can be deployed to the following Docker 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, you can decide in which container you want to deploy OmniDocs custom code changes. For example,





• #Docker Image to be updated

In this section, select the Docker image(s) in which you want to deploy custom code changes. For example,



Figure 3.5

• #Docker Image Info

This section contains the information about the source Docker images in which custom code changes can be merged or deployed.

For example,



Figure 3.6

• #New Docker Image Info with Custom Code changes

This section contains the information about new Docker images that gets created after merging the custom code changes.

For example,

Figure 3.7

• #Other user Inputs

This section contains other information that can be used in the Jenkins pipeline. For example,



- 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, deploy .JAR file to the EJB components Docker containers like OmniDocs EJB container and this container is running on the underlying AppServer JBoss EAP. In the case of JBoss EAP, deploy any dependent library then 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 be deploy it as a dependent library. Rest Build pipeline manages the updates of the *module.xml* file inside the containers. For example:



3.2 Configuring Jenkins for build pipeline

This section describes the configuration of Jenkins for Build Pipeline.

3.2.1 Prerequisites

Following are the prerequisites:

- **Operating System**: Windows Server 2019 (Edition: Standard/Datacenter)
- 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 the 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.
 - o Conditional Build Step
 - o Credentials Binding
 - o Subversion
 - Environment Injector

3.2.2 Configuring Jenkins jobs

Jenkins have the following jobs for the custom code deployment pipeline:

- 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.

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

1. Sign in to 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.



- 4. Under the Global properties, define an environment variable **PATH** with the following values separated with a semi-colon:
 - 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]

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
```

Environment variables			
List of variables	Name PATH		
	Value C:\Progr	am Files (x86)\Microsoft SDKs\Azure\CLI2\wbin;C:\Program Files\Docker\Docker\resource	2
		Delete	
	Add		

Figure 3.13

5. Save the changes.

3.2.2.1 Pull Custom Code Binaries

To pull custom code binaries, 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**.

🤵 New Item [Jenkins] 🛛 🗙					
← → C ▲ Not secure 19	2.168.57.40.1234/view/all/newJob			\$	😸 Incognito 🗄
		Q search	2	💄 DevOps Admin	→ log out
	Enter an item name				
	Pull Custom Code Binaries. » Required field				
	Freestyle project This is the central feature of Jenkins. Jenkins will build your project, comb used for something other than software build.	bining any SCM with any build system, a	nd this can be even		
	Maven project Build a maven project. Jenkins takes advantage of your POM files and dra	astically reduces the configuration.			
	Pipeline Orchestrates long-running activities that can span multiple build agents. and/or organizing complex activities that do not easily fit in free-style job	Suitable for building pipelines (formerly b type.	known as workflows)		
	Multi-configuration project Suitable for projects that need a large number of different configurations builds, etc.	s, such as testing on multiple environme	nts, platform-specific		
	Folder Creates a continer that stores nested items in it. Useful for grouping this or properties or you can have multiple things of the same name to utility such Bieating	ngs together. Unlike view, which is just a as long as they are in different folders.	filter, a folder creates a		

Figure 3.15

- 3. Specify the project description.
- 4. Under the **General**, click **Advanced** and specify the **Use custom workspace**. For example,

Source Code	Management	Build Triggers	Build Environment	Build	Post-build Actions	
e concurrent bail	us in necessary					U
t where this proje	ct can be run					()
period						?
Count						0
ouild when upstre	am project is bu	ilding				0
ouild when downs	stream project is	building				()
stom workspace						0
ory	D:\CustomCod	eDeployment				
/ Name						0
	Source Code e concurrent ound it where this proje period Count build when upstre build when downs istom workspace pry y Name	Source Code Management e concurrent bunds in necessary it where this project can be run period Count build when upstream project is bu build when downstream project is stom workspace pry D:\CustomCode y Name	Source Code Management Build Triggers e concurrent builds in necessary e it where this project can be run e period e Count e build when upstream project is building build when downstream project is building istom workspace pry D:\CustomCodeDeployment y Name	Source Code Management Build Triggers Build Environment e concurrent builds in necessary	Source Code Management Build Triggers Build Environment Build e concurrent builds in necessary a b b t where this project can be run b b b period b b b b Count b b b b build when upstream project is building b b b build when downstream project is building b b b bory D:\CustomCodeDeployment b b y Name b b b b	Source Code Management Build Triggers Build Environment Build Post-build Actions e concurrent builds in necessary t where this project can be run where this project can be run where this project can be run period

Figure 3.1

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.
- 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.
 For example,

General	Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions		
NoneSubversion	ion						
Modules	s					•	
	Reposito	ry URL	http://genesissvn.ne	ewgen.co.ir	n/svn/omnidocs/Scrutiny/Cl_CD_Pipie	•	
	Credentia	als	vivek kumar/******	(ok) 🗡	€Add ▼	•	
	Local mo	dule directory	CustomCode_Deplo	yment_Arc	chitecture	•	
	Reposito	ry depth	infinity		~	0	
	Ignore ex	kternals				0	
	Cancer p	Incess on externals fair					



9. Save the changes.

3.2.2.2 Pull Docker Image for Custom Code

To pull Docker images for custom code, 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. Select the checkbox **Inject passwords to the build as environment variables** given in the **Build Environment** section.
- 5. Specify 1 Job password: **ContainerRegistryPassword** and specify the Azure Container Registry password. For example,

✓ Inject passwords to the buil	ld as environment	variables		
Global passwords				0
Job passwords	Name	X ContainerRegistryPassword	0	0
	Password	Concealed Change Password	?	
	Add Passwords list			
Mask password parameters				0
U With Ant				?

Figure 3.19

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

Build		
Inject environment	variables	× ?
Properties File Path	D:\CustomCodeDeployment\Master\UserInput.properties	0
Properties Content		0
		11

Figure 3.20

- 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
docker login %ContainerRegistryPath% -u %ContainerRegistryUser% -p
%ContainerRegistryPassword%
docker pull
%ContainerRegistryPath%/%OmniDocs WEB ImageName%:%OmniDocs WEB Imagetag%
```

General	Source Code Management Build Triggers Build Environment Build Post-build Actions	
	Command ?	
	See the list of available environment variables	
	<pre>@echo off findstr /I "OmniDocs_WEB=Y" D:\CustomCodeDeployment\Master\UserInput.properties</pre>	
Bui	Advanced	
E	recute Windows batch command	
	Command See the list of available environment variables	
	<pre>@echo off docker login %ContainerRegistryPath% -u %ContainerRegistryUser% -p %ContainerRegistryPassword% docker pull %ContainerRegistryPath%/%OmniDocs WEB ImageName%:%OmniDocs WEB Imagetag%</pre>	
	Save Apply	

Figure 3.21

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

Here, the condition and builder for the **OmniDocs_WEB** Docker 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:

- 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 under the Build section.
- 5. Specify the **UserInput.properties** file path. For example,

Build			
Inject environment	variables	X	0
Properties File Path	D:\CustomCodeDeployment\Master\UserInput.properties		?
Properties Content			?
		/i	

Figure 3.22

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

Condit	ional steps (multiple)			X
Run?	Execute Wi	ndows Batch commands			~ 😢
	Commar	ıd			0
				_	
		See the list of available	environment variables		
					Advanced
S	teps to r	un if condition is me	et		
	Add	step to condition 🔺			
	Execu	ite Windows batch command			
	Execu	ite shell			
Add build	step Inject	environment variables			
L	Invok	e Ant			
	Invok	e top-level Maven targets			
Post-bui	ld 🖌 🛛 Run v	vith timeout			
	Trigg	er/call builds on other projects			

Figure 3.23

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
```

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
qoto 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 %destDirl%\%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
```

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
```

3.2.2.4 Build Custom Code Docker Image

To build the custom code docker 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.
- Specify the UserInput.properties file path. For example,

Build		
Inject environment	variables	× ?
Properties File Path	D:\CustomCodeDeployment\Master\UserInput.properties	2
Properties Content		0

Figure 3.24

- 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:\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 (
    sed -i s+ContainerRegistryPath+%ContainerRegistryPath%+g Dockerfile
    sed -i s+IMAGE_NAME+%OmniDocs_WEB_Imagetag%+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%
```

General	Source Code Management Build Triggers Build Environment Build Post-build Actions	
	See the list of available environment variables	
	<pre>@echo off findstr /I "OmniDocs_WEB=Y" D:\CustomCodeDeployment\Master\UserInput.properties</pre>	
Buil	Advanced	
Ex	cute Windows batch command	
	Command See the list of available environment variables	
	<pre>@echo off set ImageFilePath="D:\CustomCodeDeployment\Build Docker Images\OmniDocs WEB" set ImageName=%Custom OmniDocs WEB ImageName% set ImageTag=%Custom OmniDocs WEB Imagetag%</pre>	
	pushd %ImageFilePath% copy /y Dockerfile Original Dockerfile	
	<pre>if exist Dockerfile (sed -i s+ContainerRegistryPath+%ContainerRegistryPath%+g Dockerfile sed -i s+IMAGE NAME+%OmniDocs WEB ImageName%+g Dockerfile sed -i s+IMAGE TAG+%OmniDocs WEB Imagetag%+g Dockerfile</pre>	
	ave Apply	



10. Click **Save** to Save the changes.

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.

3.2.2.5 Push Custom Code Docker Image

To push custom code Docker 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.

 Specify 1 Job password: ContainerRegistryPassword and specify the Azure Container Registry password.
 For example,

Inject passwords to the b	ouild as environment	variables		
Global passwords			X 2 Change Password 2	
Job passwords	Name Password Add Passwords list	ContainerRegistryPassword Concealed Change Password	0 0	

Figure 3.26

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

Build			
Inject environment	variables	x	?
Properties File Path	D:\CustomCodeDeployment\Master\UserInput.properties		?
Properties Content			2
		11	



- 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 ContainerRegistryPath=%ContainerRegistryPath%
set ContainerRegistryUser=%ContainerRegistryUser%
```

```
set ContainerRegistryPassword=%ContainerRegistryPassword%
set ImageName=%Custom_OmniDocs_WEB_ImageName%
set ImageTag=%Custom_OmniDocs_WEB_Imagetag%
set BuildNumber=%ImageTag%-build-%BUILD_NUMBER%
docker login %ContainerRegistryPath% -u %ContainerRegistryUser% -p
%ContainerRegistryPassword%
docker tag %ImageName%:%ImageTag% %ContainerRegistryPath%/%ImageName%:%ImageTag%
```

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

eral	Source Code Management	Build Triggers	Build Environment	Build	Post-build Actions	
	Command ? See the list of available environm	nent variables				
	<pre>@echo off findstr /I "OmniDocs_WEB=Y" D:\CustomCodeDeployment\Master\UserInput.properties</pre>					
A	dvanced					
Exec	cute Windows batch command				~	?
	Command					
3	See the list of available environn	nent variables				
	<pre>@echo off set ContainerRegistryPath set ContainerRegistryUse</pre>	h=%ContainerReg r=%ContainerReg	istryPath% zistryUser%			
	set ContainerRegistryPass set ImageName=%Custom Om	sword=%Containe niDocs WEB Imag	rRegistryPassword% ZeName%			
	<pre>set ImageTag=%Custom Omn set BuildNumber=%ImageTag</pre>	iDocs WEB Image g%-build-%BUIL[tag%) NUMBER%			
	docker login %ContainerRe	egistryPath% -ı	– u %ContainerRegistr	yUser% -p	p %ContainerRegistryPassword%	
	docker tag % <u>ImageName</u> %:% docker push %ContainerRe	ImageTag% %Cont gistryPath%/%Ir	ainerRegistryPath% :ageName%:%ImageTag	/%ImageNa %	ame%:% <u>ImageTag</u> %	
	docker tag %ContainerReg docker push %ContainerReg	istryPath%/%Ima gistryPath%/%Im	ageName%:%ImageTag% nageName%:%BuildNum	%Contain ber%	nerRegistryPath%/% <u>ImageName</u> %:% <u>BuildNumber</u> %	

Figure 3.28

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

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.

Appendix

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.