IBM BPM (BAW) Application Deployment Pipeline Documentation (sh and wsadmin commands approach)

Purpose

This document provides comprehensive guidance for the automated deployment pipeline designed for IBM BPM (BAW) applications. The pipeline facilitates the deployment process from the process server to the process center.

Scope

The pipeline is tailored for automating deployments in IBM BPM environments, streamlining tasks through various stages from building snapshots to post-deployment.

I optionally have parameterized all the variables required to run the script so it can be easily passed during the deployment time

Components

- **CI/CD Tools**: Jenkins or equivalent for continuous integration and delivery.
- **IBM BPM Process Server and Process Center**: Target environments for deployment.
- **Deployment Scripts**: Custom scripts for each stage of the pipeline.

Workflow Summary

The pipeline encompasses four primary stages: Build Snapshot, Export Snapshot, Deploy Snapshot, and Post Deployment, each automated through specific scripts.

Pre-requisites

- Software Requirements
  - IBM BPM (specific version)
  - Jenkins (specific version)
  - Relevant script execution environment
  - Access and Permissions

    Users must have appropriate access rights to the IBM BPM Process Server and Process Center, along with credentials for script execution.

Pipeline Configuration Parameters

The pipeline is configured with several parameters that define its behavior:

containerAcronym: A short, unique identifier for the container being deployed.

containerSnapshotAcronym: A unique identifier for the snapshot of the container.

offlineServerName: Name of the offline server involved in the deployment process.

containerTrackAcronym: An acronym for tracking the container deployment.

sourceSnapshot: The snapshot of the source environment from which the deployment is initiated.

syncEnvironmentVariables: A Boolean parameter to decide whether to sync environment variables.

syncEVPs: A flag to synchronize environment variable profiles.

deploymentPackagePath: The file path where the deployment package is located.

buildPackageScriptPath: The path to the script used for building the deployment package.

exportPackageScriptPath: The path to the script used for exporting the deployment package.

deployPackageScriptPath: The path to the script for deploying the package.

postDeploymentScriptPath: The path to the script for post-deployment activities.

wsAdminCommandPath: The file path to the wsadmin command tool.

processCenterSOAPConnectorPort: The SOAP connector port for the IBM BPM Process Center.

processServerSOAPConnectorPort: The SOAP connector port for the IBM BPM Process Server.

processCenterHost: The hostname of the IBM BPM Process Center.

processServerHost: The hostname of the IBM BPM Process Server.

Jenkins Script:

```
def containerAcronym = params.containerAcronym
def containerSnapshotAcronym = params.containerSnapshotAcronym
def offlineServerName = params.offlineServerName
def containerTrackAcronym = params.containerTrackAcronym
def sourceSnapshot = params.sourceSnapshot
def syncEnvironmentVariables = params.syncEnvironmentVariables
def syncEVPs = params.syncEVPs
def deploymentPackagePath = params.deploymentPackagePath
def buildPackageScriptPath = params.buildPackageScriptPath
def exportPackageScriptPath = params.exportPackageScriptPath
def deployPackageScriptPath = params.deployPackageScriptPath
def postDeploymentScriptPath = params.postDeploymentScriptPath
def wsAdminCommandPath  = params.wsAdminCommandPath
def processCenterSOAPConnectorPort = params.processCenterSOAPConnectorPort
def processServerSOAPConnectorPort = params.processServerSOAPConnectorPort
def processCenterHost = params.processCenterHost
def processServerHost = params.processServerHost
```

Pipeline Stages

The pipeline comprises several stages:

- Build Snapshot

Objective: To build a snapshot of the deployment package.

Process: This stage uses the buildPackageScriptPath script to create a snapshot of the container specified by containerAcronym and containerSnapshotAcronym.

Error Handling: In case of an error, a message is displayed, and the user is prompted to decide whether to continue the build.

Jenkins Script:

```
stage("Build Snapshot") {
        steps {
            script {
                  try {
                                                                            withCredentials([[$class:
'UsernamePasswordMultiBinding', credentialsId:'pcCred', usernameVariable: 'USERNAME',
passwordVariable: 'PASSWORD']]) {
                                                                                                        sh "sudo $wsAdminCommandPath -
conntype SOAP -port $processCenterSOAPConnectorPort -host $processCenterHost -user $USERNAME -
password $PASSWORD -lang jython -f $buildPackageScriptPath $containerAcronym
$containerSnapshotAcronym $offlineServerName  $containerTrackAcronym
$deploymentPackagePath$containerSnapshotAcronym"+".zip"
                                                                                      }

                  } catch (Throwable e) {
                      echo "Caught ${e.toString()}"
                      input message: 'Do you want to Continue Build?', ok: 'Yes'
                      currentBuild.result = "SUCCESS"
                  }

            }

          }
        }
```

- Export Snapshot

Objective: To export the built snapshot for deployment.

Process: The **exportPackageScriptPath** script is executed to export the package, preparing it for deployment.

Error Handling: Similar to the Build Snapshot stage, errors are caught, and user input is requested for further action.

Jenkins Script:

```
stage("Export Snapshot") {
        steps {
              script {
                 try {
                   withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId:'pcCred',
usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {
                                                            sh "sudo $wsAdminCommandPath -conntype SOAP -port
$processCenterSOAPConnectorPort -host $processCenterHost -user $USERNAME -password $PASSWORD
-lang jython -f $exportPackageScriptPath $containerAcronym $containerSnapshotAcronym
$offlineServerName  $containerTrackAcronym
$deploymentPackagePath$containerSnapshotAcronym"+".zip"

                    }
                  } catch (Throwable e) {
                    echo "Caught ${e.toString()}"
                    input message: 'Do you want to Continue Build?', ok: 'Yes'
                    currentBuild.result = "SUCCESS"
                 }

           }

              }
        }
```

- ## Deploy Snapshot

Objective: To deploy the exported snapshot to the process server.

Process: This stage uses deployPackageScriptPath to deploy the package to the specified process server.

Error Handling: Errors are logged, and the user is asked whether to continue.

Jenkins Script:

```
stage("Deploy Snapshot") {
       steps {
          script {
               try {
                           withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId:'psCred',
usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {

                    sh "sudo $wsAdminCommandPath -conntype SOAP -port
$processServerSOAPConnectorPort -host $processServerHost -user $USERNAME -password $PASSWORD -
lang jython -f $deployPackageScriptPath
$deploymentPackagePath$containerSnapshotAcronym"+".zip"
                           }
               } catch (Throwable e) {
                  echo "Caught ${e.toString()}"
                  input message: 'Do you want to Continue Build?', ok: 'Yes'
                  currentBuild.result = "SUCCESS"
               }

          }
        }
      }
```

- Post Deployment

Objective: To perform post-deployment activities like:

- Sync environment variables with previous snapshots
- Sync EVPs
- Set Default Snapshot
- Migrate the BPM instances

Process: Executes the postDeploymentScriptPath script, finalizing the deployment process.

Error Handling: Any issues encountered are logged, and user input is sought on whether to proceed.

Jenkins Script

```
stage("Post Deployment") {
        steps {

            script {
                try {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId:'psCred',
usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {
                                                            sh "sudo $wsAdminCommandPath -conntype SOAP -port
$processServerSOAPConnectorPort -host $processServerHost -user $USERNAME -password $PASSWORD -
lang jython -f $postDeploymentScriptPath $containerAcronym $containerSnapshotAcronym
$containerTrackAcronym $sourceSnapshot"
                    }
                } catch (Throwable e) {
                    echo "Caught ${e.toString()}"
                    input message: 'Do you want to Continue Build?', ok: 'Yes'
                    currentBuild.result = "SUCCESS"
                    }

              }
          }
      }
   }
```

Deployment Scripts

build-package.py

Purpose: Builds the deployment package.

```
import sys
from com.ibm.ws.scripting import ScriptingException
import time
containerAcronym=sys.argv[0]
containerSnapshotAcronym=sys.argv[1]
```

```
offlineServerName=sys.argv[2]
containerTrackAcronym=sys.argv[3]
path=sys.argv[4]
AdminTask.BPMCreateOfflinePackage('[-containerAcronym ' +containerAcronym+ ' -containerSnapshotAcronym '
+containerSnapshotAcronym+ ' -containerTrackAcronym ' +containerTrackAcronym+ ' -serverName '
+offlineServerName+ ' -skipGovernance false]')
raise SystemExit
```

export-package.py

Purpose: Builds the deployment package.

```
import sys
from com.ibm.ws.scripting import ScriptingException
import time
containerAcronym=sys.argv[0]
containerSnapshotAcronym=sys.argv[1]
offlineServerName=sys.argv[2]
containerTrackAcronym=sys.argv[3]
path=sys.argv[4]
time.sleep(30)
AdminTask.BPMExtractOfflinePackage('[-containerAcronym ' +containerAcronym+ ' -
containerSnapshotAcronym ' +containerSnapshotAcronym+ ' -containerTrackAcronym '
+containerTrackAcronym+ ' -serverName ' +offlineServerName+ ' -outputFile ' +path+ ']')
raise SystemExit
```

deploy-package.py

Purpose: Builds the deployment package.

```
import sys
from com.ibm.ws.scripting import ScriptingException
import time
path=sys.argv[0]
AdminTask.BPMInstallOfflinePackage('[-inputFile ' +path+ ']')
raise SystemExit
```

post-deployment.py

Purpose: Builds the deployment package.

```
import sys
from com.ibm.ws.scripting import ScriptingException

containerAcronym=sys.argv[0]
targetSnapshot=sys.argv[1]
containerTrackAcronym = sys.argv[2]
sourceSnapshot=sys.argv[3]
```

```
# Sync Environment Variables with previous snapshot (sourceSnapshot)
AdminTask.BPMSyncEnvironmentVariables('[-containerAcronym ' +containerAcronym+ ' -
sourceContainerSnapshotAcronym ' +sourceSnapshot+ ' -targetContainerSnapshotAcronym ' +targetSnapshot+ ']')

# Sync EPVs Variables with previous snapshot (SourceSnapshot)
AdminTask.BPMSyncEPVValues('[-containerAcronym ' +containerAcronym+ ' -sourceContainerSnapshotAcronym '
+sourceSnapshot+ ' -targetContainerSnapshotAcronym ' +targetSnapshot+ ']')

# set default snapshot
AdminTask.BPMSetDefaultSnapshot('[-containerAcronym ' +containerAcronym+ ' -containerSnapshotAcronym '
+targetSnapshot+ ']')

#Migrate Instances Based on Flag

AdminTask.BPMMigrateInstances('[-containerAcronym ' +containerAcronym+ ' -sourceContainerSnapshotAcronym
' +sourceSnapshot+ ' -targetContainerSnapshotAcronym ' +targetSnapshot+ ']')

# Deactivate Previous Snapshot Based on Flag (TODO)

#AdminTask.BPMDeactivate('[-containerAcronym ' +containerAcronym+ ' -containerSnapshotAcronym '
+targetSnapshot+ ' -containerTrackAcronym '  +containerTrackAcronym+ ']')
```

This document serves as a guide for efficiently deploying IBM BPM applications using the defined pipeline, ensuring streamlined automated deployment process.

Pipeline Python Script Files:

build-package.py export-package.py post-deployment.py

deploy-package.py

Pipeline Jenkins Script:

```
def containerAcronym = params.containerAcronym
def containerSnapshotAcronym = params.containerSnapshotAcronym
def offlineServerName = params.offlineServerName
def containerTrackAcronym = params.containerTrackAcronym
def sourceSnapshot = params.sourceSnapshot
def syncEnvironmentVariables = params.syncEnvironmentVariables
def syncEVPs = params.syncEVPs
def deploymentPackagePath = params.deploymentPackagePath
def buildPackageScriptPath = params.buildPackageScriptPath
def exportPackageScriptPath = params.exportPackageScriptPath
def deployPackageScriptPath = params.deployPackageScriptPath
def postDeploymentScriptPath = params.postDeploymentScriptPath
def wsAdminCommandPath  = params.wsAdminCommandPath
def processCenterSOAPConnectorPort = params.processCenterSOAPConnectorPort
def processServerSOAPConnectorPort = params.processServerSOAPConnectorPort
def processCenterHost = params.processCenterHost
def processServerHost = params.processServerHost

pipeline {
    agent any
    stages {
        stage("Build Snapshot") {
         steps {
             script {
                    try {
                                                                        withCredentials([[$class: 'UsernamePasswordMultiBinding',
credentialsId:'pcCred', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {
                                                                                            sh "sudo $wsAdminCommandPath -conntype
SOAP -port $processCenterSOAPConnectorPort -host $processCenterHost -user $USERNAME -
password $PASSWORD -lang jython -f $buildPackageScriptPath $containerAcronym
$containerSnapshotAcronym $offlineServerName  $containerTrackAcronym
$deploymentPackagePath$containerSnapshotAcronym"+".zip"
                                                                             }

                    } catch (Throwable e) {
                        echo "Caught ${e.toString()}"
                        input message: 'Do you want to Continue Build?', ok: 'Yes'
                        currentBuild.result = "SUCCESS"
                    }

             }

                                     }
         }
 stage("Export Snapshot") {
         steps {
                script {
                    try {
                       withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId:'pcCred',
usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {
                                                                   sh "sudo $wsAdminCommandPath -conntype SOAP -port
$processCenterSOAPConnectorPort -host $processCenterHost -user $USERNAME -password
$PASSWORD -lang jython -f $exportPackageScriptPath $containerAcronym
$containerSnapshotAcronym $offlineServerName  $containerTrackAcronym
$deploymentPackagePath$containerSnapshotAcronym"+".zip"

                        }
                      } catch (Throwable e) {
                        echo "Caught ${e.toString()}"
                        input message: 'Do you want to Continue Build?', ok: 'Yes'
                        currentBuild.result = "SUCCESS"
                    }

             }

                                     }
         }
                          stage("Deploy Snapshot") {
```

```
        steps {
           script {
                 try {
                                 withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId:'psCred',
usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {

                      sh "sudo $wsAdminCommandPath -conntype SOAP -port
$processServerSOAPConnectorPort -host $processServerHost -user $USERNAME -password
$PASSWORD -lang jython -f $deployPackageScriptPath
$deploymentPackagePath$containerSnapshotAcronym"+".zip"
                                 }
                 } catch (Throwable e) {
                     echo "Caught ${e.toString()}"
                     input message: 'Do you want to Continue Build?', ok: 'Yes'
                     currentBuild.result = "SUCCESS"
                 }

           }
         }
       }
                       stage("Post Deployment") {
        steps {

           script {
                 try {
                     withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId:'psCred',
usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD']]) {
                                                     sh "sudo $wsAdminCommandPath -conntype SOAP -port
$processServerSOAPConnectorPort -host $processServerHost -user $USERNAME -password
$PASSWORD -lang jython -f $postDeploymentScriptPath $containerAcronym
$containerSnapshotAcronym $containerTrackAcronym $sourceSnapshot"
                     }
                  } catch (Throwable e) {
                        echo "Caught ${e.toString()}"
                        input message: 'Do you want to Continue Build?', ok: 'Yes'
                        currentBuild.result = "SUCCESS"
                     }

              }
          }
       }
    }
}
```