|  |  |  |
| --- | --- | --- |
|  | **Front end** | **Back end** |
| language | Java script (JS) | Java (core) |
| Code | From git repo to jenkins | From git repo to jenkins |
| By cloning through SCM checkout | By cloning through SCM checkout |
| Building  The  Code | package.json files | pom.xml files |
| by Node Package Manager(npm) | by Maven |
| * npm install * npm run build   we get a build folder in  JENKINS WORKSPACE | mvn clean install  we get a ARTIFACT inside the created TARGET folder in JENKINS WORKSPACE |
| * Contains project related   information  Dependencies for the application | * Get a File in war/jar format * project info * Dependencies * Artifact name, version, type   Ex:  Performace - 0.1 - war/jar file |
| NEXUS | Push the build folder to NEXUS Repo for deploying machine | Push the ARTIFACT to nexus repo for deploying machine. |
| Ansibile | Using playbook, we download the build folder.  Usually Service don’t stop. | Using playbook, we  Download the file.  Stop service  and start service |
| Ansible  Tower | 1.create projects   * + - * 1. Name         2. Description         3. Organization         4. Scm type         5. Credential of git   2.create an inventory template  (a)name  (b)description  (c)job type  (d)create scheduler  (e)inventory hosts  (f)email notification  3. Launch |  |

import groovy.json.\*   //importing some pieces of codes which need in the pipeline

def call(Map config)

{

        node('nodemachine')

    {

        tools

        {

            maven 'Maven\_Latest\_Linux'

            jdk 'JDK\_Latest\_Linux'

        }

            stage('scm checkout') {

                try{

                    echo "git cloning started"

                    git branch: 'development', credentialsId:<''>, url: <''>

                    echo "git clone done"

                } catch()

                    {

                        echo "git checkout failed"

                    }

            }

            stage('build') {

                def mvn\_home = tool "Maven\_Latest\_Linux"

                def jdk\_version = tool "JDK\_Latest\_Linux"

                def mvn\_exec = "export JAVA\_HOME=${jdk\_version};${mvn\_home}/bin/mvn"

                sh """

                    $mvn\_exec clean install

                or

                export JAVA\_HOME=${jdk\_version};${mvn\_home}/bin/mvn clean install

                """

            }

            stage('pushing to Nexus') {

                nexus3Url = "<url>"

                repositoryId = "<id>"

                folderName = "<>"

                Nexus\_Url   = $<>

                withCredentials([usernamePassword(credentialsId:<nexusid>, passwordVariable: 'password', usernameVariable: 'user')]){

                        def dataraw = """ '{

                            {

                                 groupId: "${folderName}"

                                        "artifactId": "${artifactID}"

                                        "version": "${artifactVersion}"

                            }

                        }

                            ' """

                        command = "curl -k -u ${USER}:${PASSWORD} -X POST ${Nexus\_URL} -d ${dataraw}"

                         def responce\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

                        def result = new JsonSluper().parseText(responce\_value)

                        echo "${result}"

                    if (0 != result) {

                    throw new exception("error- failed to upload artifact to nexus")

                     }

            stage('ansible tower'){

                    Ansible\_Tower\_URL =<url>

                    or

                    Ansible\_Tower\_URL =<url>/${ANsible\_playbook\_no}/launch/

                    withCredentials([usernamePassword(credentialsId: 'Ansible\_Tower\_ID', passwordVariable: 'password', usernameVariable: 'USER')]){

                        def dataraw = """ ' {

                            {

                                "extra\_vars": {

                                    "target\_hosts": "${Target\_Server}"

                                }

                            } ' """

                            command = "curl -k -u ${USER}:${PASSWORD} -X POST ${Ansible\_Tower\_URL} -d ${dataraw}"

                         def responce\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

                        def result = new JsonSluper().parseText(responce\_value)

                        echo "${result}"

                        jobID = result.id

                        echo "$jobID"

                        }

                    stage('Getting\_playbook\_output'){

                        Ansible\_Job\_URL = <id>/${jobID}/

                        withCredentials([usernamePassword(credentialsId: 'Ansible\_Tower\_ID', passwordVariable: 'password', usernameVariable: 'USER')]){

                        command = "curl -k -u ${USER}:${PASSWORD} -X GET ${Ansible\_Job\_URL} -d ${dataraw}"

                         def responce\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

                        def result = new JsonSluper().parseText(responce\_value)

                        echo "playbook output - ${result}"

                    }

                    }

            }

                }

                        {

                        {

                }

            }

    }

}

ChatGPT Deployment

The code you've provided appears to be a Jenkins Pipeline script in Groovy for building a Java application using Maven and deploying it using Nexus and Ansible. However, it seems to contain some errors and missing pieces of information, like credentials and URLs. Here's a corrected version:

```groovy

import groovy.json.\*

def call(Map config)

{

node('nodemachine')

{

tools

{

maven 'Maven\_Latest\_Linux'

jdk 'JDK\_Latest\_Linux'

}

stage('scm checkout') {

try{

echo "git cloning started"

git branch: 'development', credentialsId: 'your-git-credentials', url: 'your-git-url'

echo "git clone done"

} catch(Exception e)

{

echo "git checkout failed"

echo e.message

}

}

stage('build') {

def mvn\_home = tool "Maven\_Latest\_Linux"

def jdk\_version = tool "JDK\_Latest\_Linux"

def mvn\_exec = "export JAVA\_HOME=${jdk\_version};${mvn\_home}/bin/mvn"

sh """

$mvn\_exec clean install

"""

}

stage('pushing to Nexus') {

def nexus3Url = "your-nexus-url"

def repositoryId = "your-repo-id"

withCredentials([usernamePassword(credentialsId:'nexus-credentials', passwordVariable: 'PASSWORD', usernameVariable: 'USER')]){

def dataraw = """ '{

{

"groupId": "${repositoryId}"

"artifactId": "${artifactID}"

"version": "${artifactVersion}"

}

}

' """

def command = "curl -k -u ${USER}:${PASSWORD} -X POST ${nexus3Url} -d ${dataraw}"

def response\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

def result = new JsonSlurper().parseText(response\_value)

echo "${result}"

if (0 != result) {

throw new Exception("error- failed to upload artifact to nexus")

}

}

}

stage('ansible tower'){

def ansibleTowerURL = "your-ansible-tower-url"

withCredentials([usernamePassword(credentialsId: 'ansible-tower-credentials', passwordVariable: 'PASSWORD', usernameVariable: 'USER')]){

def dataraw = """ '{

{

"extra\_vars": {

"target\_hosts": "${Target\_Server}"

}

} ' """

def command = "curl -k -u ${USER}:${PASSWORD} -X POST ${ansibleTowerURL} -d ${dataraw}"

def response\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

def result = new JsonSlurper().parseText(response\_value)

echo "${result}"

def jobID = result.id

echo "$jobID"

}

}

stage('Getting\_playbook\_output'){

def ansibleJobURL = "your-ansible-job-url/${jobID}/"

withCredentials([usernamePassword(credentialsId: 'ansible-tower-credentials', passwordVariable: 'PASSWORD', usernameVariable: 'USER')]){

def command = "curl -k -u ${USER}:${PASSWORD} -X GET ${ansibleJobURL}"

def response\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

def result = new JsonSlurper().parseText(response\_value)

echo "playbook output - ${result}"

}

}

}

}

```

Please note that you'll need to replace placeholder values like 'your-git-credentials', 'your-git-url', 'your-nexus-url', 'your-repo-id', 'nexus-credentials', 'your-ansible-tower-url', and 'ansible-tower-credentials' with your actual credentials and URLs.

Also, note that the `try-catch` block now catches `Exception e` and the extra set of brackets is removed. I've also corrected some typographical errors, such as `responce\_value` to `response\_value` and `JsonSluper` to `JsonSlurper`.

Finally, remember to properly structure your JSON data and substitute variables like `${artifactID}`, `${artifactVersion}`, `${Target\_Server}`, and `${jobID}` with appropriate values according to your use case.

import groovy.json.\*

def call(Map config)

{

node('nodemachine')

{

tools

{

maven 'Maven\_Latest\_Linux'

jdk 'JDK\_Latest\_Linux'

}

stage('scm checkout') {

try{

echo "git cloning started"

git branch: 'development', credentialsId: 'your-git-credentials', url: 'your-git-url'

echo "git clone done"

} catch(Exception e)

{

echo "git checkout failed"

echo e.message

}

}

stage('build') {

def mvn\_home = tool "Maven\_Latest\_Linux"

def jdk\_version = tool "JDK\_Latest\_Linux"

def mvn\_exec = "export JAVA\_HOME=${jdk\_version};${mvn\_home}/bin/mvn"

sh """

$mvn\_exec clean install

"""

}

stage('pushing to Nexus') {

def nexus3Url = "your-nexus-url"

def repositoryId = "your-repo-id"

withCredentials([usernamePassword(credentialsId:'nexus-credentials', passwordVariable: 'PASSWORD', usernameVariable: 'USER')]){

def dataraw = """ '{

{

"groupId": "${repositoryId}"

"artifactId": "${artifactID}"

"version": "${artifactVersion}"

}

}

' """

def command = "curl -k -u ${USER}:${PASSWORD} -X POST ${nexus3Url} -d ${dataraw}"

def response\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

def result = new JsonSlurper().parseText(response\_value)

echo "${result}"

if (0 != result) {

throw new Exception("error- failed to upload artifact to nexus")

}

}

}

stage('ansible tower'){

def ansibleTowerURL = "your-ansible-tower-url"

withCredentials([usernamePassword(credentialsId: 'ansible-tower-credentials', passwordVariable: 'PASSWORD', usernameVariable: 'USER')]){

def dataraw = """ '{

{

"extra\_vars": {

"target\_hosts": "${Target\_Server}"

}

} ' """

def command = "curl -k -u ${USER}:${PASSWORD} -X POST ${ansibleTowerURL} -d ${dataraw}"

def response\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

def result = new JsonSlurper().parseText(response\_value)

echo "${result}"

def jobID = result.id

echo "$jobID"

}

}

stage('Getting\_playbook\_output'){

def ansibleJobURL = "your-ansible-job-url/${jobID}/"

withCredentials([usernamePassword(credentialsId: 'ansible-tower-credentials', passwordVariable: 'PASSWORD', usernameVariable: 'USER')]){

def command = "curl -k -u ${USER}:${PASSWORD} -X GET ${ansibleJobURL}"

def response\_value = sh(script:""" set +x; ${command} """,returnStdout: true)

def result = new JsonSlurper().parseText(response\_value)

echo "playbook output - ${result}"

}

}

}

}

This Jenkins Pipeline script is designed to build a Java application using Maven and JDK on a designated node ('nodemachine'). It consists of two stages: 'scm checkout' and 'build'.

1. `def call(Map config)`: This defines a Jenkins Pipeline function named `call`, which takes a `Map` as a parameter. The function is the entry point of the pipeline script and will be automatically executed when the pipeline is run.

2. `node('nodemachine')`: This specifies that the pipeline should run on a node with the label 'nodemachine'. Nodes in Jenkins represent the machines on which jobs and pipelines can be executed.

3. `tools { maven 'Maven\_Latest\_Linux'; jdk 'JDK\_Latest\_Linux' }`: This defines the required tools for the pipeline. It ensures that the specified versions of Maven and JDK are available on the node where the pipeline is executed.

4. `stage('scm checkout')`: This defines the first stage of the pipeline, named 'scm checkout', which represents checking out the source code from the version control system (Git) to the workspace.

5. `try { ... } catch (Exception e) { ... }`: This is a try-catch block that attempts to perform the 'git clone' operation. If an exception occurs during the checkout process, the catch block will execute, and the error message will be printed.

6. `git branch: 'development', credentialsId: 'your-git-credentials', url: 'your-git-url'`: This is the actual Git checkout command that clones the 'development' branch from the specified Git repository ('your-git-url'). It uses the provided 'your-git-credentials' to access the repository.

7. `stage('build')`: This defines the second stage of the pipeline, named 'build'. In this stage, the Java application is built using Maven.

8. `def mvn\_home = tool "Maven\_Latest\_Linux"`: This retrieves the path to the installed Maven tool on the Jenkins agent and stores it in the `mvn\_home` variable.

9. `def jdk\_version = tool "JDK\_Latest\_Linux"`: This retrieves the path to the installed JDK on the Jenkins agent and stores it in the `jdk\_version` variable.

10. `def mvn\_exec = "export JAVA\_HOME=${jdk\_version};${mvn\_home}/bin/mvn"`: This sets the `mvn\_exec` variable to a command that exports the JDK version to the `JAVA\_HOME` environment variable and then runs Maven from the specified `mvn\_home` path.

11. `sh """ ... """`: This step runs the shell command specified within triple-double quotes. In this case, it runs the Maven build command using the `mvn\_exec` variable to build the Java application.

Please note that the pipeline script provided here is a basic outline and doesn't include the full pipeline logic, such as deploying to Nexus or Ansible Tower. Those parts were not included in the provided script. Additionally, you need to replace `'your-git-credentials'` and `'your-git-url'` with the appropriate credentials and Git repository URL specific to your setup.