**Pipeline Overview**

The Jenkins pipeline is designed to automate the process of deploying code through various environments, including DEV, TEST-DR, PROD, and DR. The pipeline leverages Kubernetes for container orchestration and Informatica Cloud (IICS) for data integration tasks. The pipeline also integrates with Slack for notifications.

**Stages**

**1. Initialize**

**Purpose:**

* Checkout the source code
* Load properties
* Set global variables
* Initialize Slack channels

**Steps:**

* Delete the workspace directory.
* Checkout the source code from the repository and get the commit ID.
* Load properties and set global variables.
* Initialize Slack channels.

**2. Get Commit Hash from Dev Environment**

**Purpose:**

* Fetch the commit hash from the development environment.

**Steps:**

* Fetch the commit hash from the development environment and store it in an environment variable.

**3. Promote Commit Hash to Prod Environment**

**Purpose:**

* Promote the commit hash to the production environment.

**Steps:**

* Checkout the code from the DEV and PRD branches.
* Configure git and perform a cherry-pick operation.
* Push the changes to the PRD branch.
* Run the infa\_update\_and\_test.py script to update and test the deployment.

**4. Promote Commit Hash to Test Environment**

**Purpose:**

* Promote the commit hash to the Test environment.

**Steps:**

* Checkout the code from the DEV and TEST branches.
* Configure git and perform a cherry-pick operation.
* Push the changes to the TEST branch.

**5. Promote Commit Hash to Test-DR Environment**

**Purpose:**

* Promote the commit hash to the Test-DR environment.

**Steps:**

* Checkout the code from the DEV and TEST-DR branches.
* Configure git and perform a cherry-pick operation.
* Push the changes to the TEST-DR branch.

6**. Promote Commit Hash to DR Environment**

**Purpose:**

* Promote the commit hash to the DR environment.

**Steps:**

* Checkout the code from the DEV and DR branches.
* Configure git and perform a cherry-pick operation.
* Push the changes to the DR branch.

**7. Test Deployment Code/Commit Hash**

**Purpose:**

* Test the deployment code and the commit hash.

**Steps:**

* Login to the IICS DEV environment and test the mapping tasks.

**Post Actions**

The pipeline includes post actions to handle different build outcomes:

* **Always:** Complete the build process and archive artifacts if necessary.
* **Success:** Indicate that the build process was successful.
* **Unstable:** Indicate that the build is unstable.
* **Aborted:** Indicate that the pipeline was aborted.
* **Failure:** Indicate that the build encountered failures.

**Functions**

**initSlackChannels**

**Purpose:**

* Initialize Slack channels and tokens.

**Code:**

def initSlackChannels() {

SLACK\_CHANNELS.each { key, value ->

withCredentials([usernamePassword(credentialsId: value.channel, passwordVariable: 'channel\_token', usernameVariable: 'channel\_name')]) {

value.channel = "${channel\_name}"

value.token = "${channel\_token}"

echo "channel token is '${channel\_token}'"

echo "channel name is '${channel\_name}'"

}

}

}

**slackNotification**

**Purpose:**

* Send notifications to Slack channels.

**Code:**

def slackNotification(channel, msg, color, nc) {

def lt, lc

lt = SLACK\_CHANNELS[channel].token

lc = SLACK\_CHANNELS[channel].channel

slackSend color: color, channel: lc, message: msg, baseUrl: SLACK\_URL, token: lt, notifyCommitters: nc

}