CICD with Jenkins
Lab Notes Part 2

Introduction

This manual accompanies the course *CICD with Jenkins".*

Most of these labs have bee previously demonstrated during class. These lab notes are references to what was done in class.

Lab Nine: Simple Jenkins Pipeline

Create a pipeline project using the Pipeline plugin.

In the pipeline section of the configuration page, use the Hello World sample.

Cut and paste several copies of the “Hello” stage to form a pipeline and change the names “Build”, “Test”, “Package”. “Deploy” and “Report”

Alter the echo file to echo the name of the stage it’s building.

Run the pipeline several times and see how it tracks over the various builds.

The code for the pipeline is below

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Build'
            }
        }
    stage('Test') {
            steps {
                echo 'Test'
            }
        }
    stage('Package') {
            steps {
                echo 'Package'
            }
        }
    stage('Deploy') {
            steps {
                echo 'Deploy'
            }
        }
    stage('Report') {
            steps {
                echo 'Report'
            }
        }
    }

}
```

Failing Pipeline

Add a failed return code to the test stage to simulate a failed step. Code is below:

```
stage('Test') {
            steps {
                echo 'Test'
                sh 'exit 1'

            }
        }
```

Run the output and see how the pipeline responds.

Lab Ten: Post Build Stages

Add the post build stages listed below to the code for lab nine:

- Success
- Failure
- Cleanup
- Always
- Fixed

With each step printing out an informative message. The code is listed on the next page.

- Run a successful build and examine the console output
- Break the test stage by returning a failure error code like in the last lab.
- Run the broken build
- Fix the test stage and re-run

Examine the output at the console for each of the runs.

```
pipeline {
    agent any

    stages {
        << Snipped >>
        }
    }
    post {
         always {
            echo '++++++++ Always Post build step'
         }
         success {
            echo 'SUCCESS!!!'
         }
         failure {
            echo '***** Failure occured'
         }
         fixed {
            echo '----- Problem Fixed'
         }
         cleanup {
            echo 'Bye from Cleanup'
        }
    }

}
```

Lab Eleven: Using a SCM

1. Create a GitLab repo
2. Make sure it is a public repository so that you don't have to worry about credentials
3. Copy the code from the previous lab into a new file named "JenkinsFile" in the repo you just created
4. Create a new Pipeline project and use the SCM feature in the advanced options to configure the location of the repo
    a. Ensure that you set the branch to main and not master
    b. Ensure that the name of the JenkinsFile is correct.
    c. Use the URL from the "Clone" button with HTPPS URL

5. Run the build

In this example, the file is in the root directory. You could have different pipelines to run in different directories.

Lab Twelve: Environment Variables

From the list at this URL

https://www.jenkins.io/doc/book/pipeline/jenkinsfile/#using-environment-variables

Edit one of the stages to print out several of the variables using both the env. prefix and no prefix.

Notice that to print out the variable that it must be a string so it needs to be in double quotes

For example:

```
pipeline {
    agent any
    stages {
        stage('Example') {
            steps {
                echo "Running ${env.BUILD_ID} on ${env.JEN-
KINS_URL}"
            }
        }
    }
}
```

Lab Thirteen: Defining Variables

Following the demo, add and use the IN_PROD environment variable for both the whole pipeline and a given stage

Code is on the next page

```
pipeline {
    agent any
    environment {
        IN_PROD = 'false'
    }
    stages {
        stage('Build') {
            steps {
                echo "Build"
                echo "Value of IN_PROD is ${IN_PROD}"
            }
        }
    stage('Test') {
        environment {
            IN_PROD = 'true'

        }
            steps {
                echo 'Test'
                 echo "Value of IN_PROD is ${IN_PROD}"
            }
        }
    stage('Package') {
            steps {
                echo 'Package'
                 echo "Value of IN_PROD is ${IN_PROD}"
            }
        }
    stage('Deploy') {
            steps {
                echo 'Deploy'
            }
        }
    stage('Report') {
            steps {
                echo 'Report'
            }
        }
    }

}
```

Lab Fourteen: When

Replicate the demo. Code is below

```
pipeline {
    agent any
    environment {
        IN_PROD = 'false'
    }
    stages {
        stage('Build') {
            steps {
                echo "Build"
                echo "Value of IN_PROD is ${IN_PROD}"
            }
        }
    stage('Test') {

            steps {
                echo 'Test'

            }
        }
    stage('Package') {
        when {
            environment name: "IN_PROD", value: 'true'
        }
            steps {
                echo 'Package'

            }
        }
    stage('Deploy') {
            steps {
                echo 'Deploy'
            }
        }
    stage('Report') {
            steps {
                echo 'Report'
            }
        }
    }

}
```

Lab Fifteen: Parameters

Again, replicate the demo. Code follows.

```
pipeline {
    agent any

    parameters {
        booleanParam(name: "PRODUCTION", defaultValue: false, de-
scription: "For triggering Packahe")
        string(name: "AUTHOR", defaultValue: "Rod", trim: true,
description: "Code Author")
        choice(name: "TARGET", choices: ["production", "develop-
ment", "testing"], description: "Build Target")
    }
    stages {
        stage("Build") {
            steps {
                echo "Build stage."
                echo "This is $params.TARGET build authored by
$params.AUTHOR"
            }
        }
        stage("Test") {
            steps {
                echo "Test stage."
            }
        }
        stage("Release") {
            steps {
                echo "Release stage."
                echo "We have a ${params.TARGET} target"
            }
        }
    }
}
```

Lab Sixteen: Parallel Builds

```
pipeline {
    agent none
    stages {
        stage('Run Tests') {
            parallel {
                stage('Test On Any Node') {
                    agent {
                        label 'MainNode'
                    }
                    steps {
                        echo 'run-tests'
                    }
                }
                stage('Test On Remote Agent') {
                    agent {
                        label 'LinuxAgent'
                    }
                    steps {
                       echo 'Running remote tests'
                    }
                }
            }
        }
    }
}
```