Lab: Configure a global pipeline library

In this exercise you will:

- Configure a global pipeline library.
- Create and run a simple Jenkinsfile to verify that the library is set up correctly.

Task: Configure a global pipeline library

1. From the Jenkins Dashboard, select **Manage Jenkins** in the left navigation bar.
2. Under **System Configuration**, select **Configure System**.
3. Scroll down to **Global Pipeline Libraries**.
4. Select **Add** and set the following values:
    - **Name**: `shared-library`
    - **Default version**: `master`
    - **Load implicitly**: unchecked
    - **Allow default version to be overridden**: checked
    - **Include @Library changes in job recent changes**: checked
5. Under **Retrieval method**, select **Modern SCM**.
6. Select **Git** and set the following values:
    - **Project Repository**: `http://localhost:5000/gitserver/butler/shared-library`
    - **Credentials**: **butler**
7. Select **Save**.

Task: Verify that the library is set up correctly

To verify that the library is set up correctly, create and run a simple `Jenkinsfile`:

1. From the Jenkins Dashboard, select **New Item**.
2. Enter the item name `test-shared-library`.
3. Select **Pipeline**.
4. Select **OK**.
5. Scroll down to the Pipeline text section and paste the following into the **Script** text box:

```
@Library('shared-library') _
pipeline {
    agent { label 'java' }
    stages {
        stage('verify') {
            steps {
                helloWorld(name: elies)
            }
        }
    }
}
```

6. Select **Save**.
7. Select **Build Now** in the left navigation bar.
8. Under **Build History**, select the blue ball to the left of #1 to open the Console Output.
9. Scroll down and verify that you see

```
Hello world, elies
```

Now you will create a custom step.

Task: Create a custom step

Use the Gitea editor to create the custom step:

1. From your CloudBees Lab Instance home page, select the **Gitserver (A local Git service)** link.
2. Sign in with your lab environment username *and* password: `butler`.
3. Navigate to the `shared-library` Gitea repository.
4. Confirm that `Branch` is set to `master`.
5. Select the `vars` directory.
6. Select **New File** to the right of the directory name to create a new file.
7. Name the file `postBuildSuccess.groovy`.
8. Paste in the following content:

   vars/postBuildSuccess.groovy

```
def call(Map config = [:]) {
    archiveArtifacts 'target/*.jar'
    stash(name: "${config.stashName}", includes: 'target/**')
}
```

9. Add a commit message and select **Commit Changes** to commit directly to the `master` branch.

Commit changes to the `master` branch in all the exercises in the lab.

Task: Use a custom step

In this exercise you will modify your existing Pipeline to use the custom step created in the previous step.

Task: Modify the existing Pipeline to use the custom step

Use the Gitea editor to modify the existing Pipeline:

1. Navigate to the `pipeline-lab` Gitea repository.
2. Confirm that `Branch` is set to `master`.
3. Select `Jenkinsfile`.
4. Select the pencil icon in the tool bar to enter edit mode.
5. Enter the following line at the top of the file:

   ```
   @Library('shared-library') _
   ```

6. Scroll down to the `post { success … } }` section of the Build Java 7 stage.
7. Replace
8.
   ```
   archiveArtifacts 'target/*.jar'
   stash(name: 'Java 7', includes: 'target/**')
   ```

   with

   ```
   postBuildSuccess(stashName: "Java 7")
   ```

9. Scroll down to the bottom and select **Commit Changes**. The job should start automatically once the commit has completed.
10. Verify that the pipeline ran successfully and waits for input. You can select **Abort** as your input for this exercise.

    Cancel (select **Abort**) or confirm (select **Let's Do It!**) deploying in Blue Ocean if the Console Output does not respond to your input.

Solution

Jenkinsfile

```
@Library('shared-library') _
pipeline {
  agent none
  stages {
    stage('Fluffy Build') {
      parallel {
        stage('Build Java 8') {
          agent {
            node {
              label 'java8'
            }
          }
          steps {
            sh './jenkins/build.sh'
          }
          post {
            success {
              stash(name: 'Java 8', includes: 'target/**')
            }
          }
        }
        stage('Build Java 7') {
          agent {
            node {
              label 'java7'
            }
          }
          steps {
            sh './jenkins/build.sh'
          }
          post {
            success {
              postBuildSuccess(stashName: "Java 7")
            }
          }
        }
      }
    }
    stage('Fluffy Test') {
      parallel {
        stage('Backend Java 8') {
          agent {
            node {
              label 'java8'
            }
          }
          steps {
            unstash 'Java 8'
            sh './jenkins/test-backend.sh'
          }
          post {
            always {
              junit 'target/surefire-reports/**/TEST*.xml'
            }
          }
        }
        stage('Frontend Java 8') {
```

```
    agent {
      node {
        label 'java8'
      }
    }
    steps {
      unstash 'Java 8'
      sh './jenkins/test-frontend.sh'
    }
    post {
      always {
        junit 'target/test-results/**/TEST*.xml'
      }
    }
  }
  stage('Performance Java 8') {
    agent {
      node {
        label 'java8'
      }
    }
    steps {
      unstash 'Java 8'
      sh './jenkins/test-performance.sh'
    }
  }
  stage('Static Java 8') {
    agent {
      node {
        label 'java8'
      }
    }
    steps {
      unstash 'Java 8'
      sh './jenkins/test-static.sh'
    }
  }
  stage('Backend Java 7') {
    agent {
      node {
        label 'java7'
      }
    }
    steps {
      unstash 'Java 7'
      sh './jenkins/test-backend.sh'
    }
    post {
      always {
        junit 'target/surefire-reports/**/TEST*.xml'
      }
    }
  }
  stage('Frontend Java 7') {
    agent {
      node {
        label 'java7'
      }
    }
    steps {
      unstash 'Java 7'
```

```
            sh './jenkins/test-frontend.sh'
          }
          post {
            always {
              junit 'target/test-results/**/TEST*.xml'
            }
          }
        }
        stage('Performance Java 7') {
          agent {
            node {
              label 'java7'
            }
          }
          steps {
            unstash 'Java 7'
            sh './jenkins/test-performance.sh'
          }
        }
        stage('Static Java 7') {
          agent {
            node {
              label 'java7'
            }
          }
          steps {
            unstash 'Java 7'
            sh './jenkins/test-static.sh'
          }
        }
      }
    }
    stage('Confirm Deploy') {
      when {
        branch 'master'
      }
      steps {
        input(message: 'Okay to Deploy to Staging?', ok: 'Let\'s Do it!')
      }
    }
    stage('Fluffy Deploy') {
      when {
        branch 'master'
      }
      agent {
        node {
          label 'java7'
        }
      }
      steps {
        unstash 'Java 7'
        sh "./jenkins/deploy.sh ${params.DEPLOY_TO}"
      }
    }
  }
  parameters {
    string(name: 'DEPLOY_TO', defaultValue: 'dev', description: '')
  }
}
```