**Building a Jenkins Pipeline**

Jenkins Pipeline allows you to treat all of your build logic as code. You can also integrate your build into a larger automated process that includes things like automated deployment. Pipelines are especially useful as you transition from doing CI to doing CI+CD.

In this hands-on lab, you will build a simple pipeline for a sample application. You will gain a basic familiarity with how a Jenkins Pipeline is created and how to implement simple build logic within a pipeline.

**The Scenario**

Your team is building a train scheduling app. Continuous integration has already been implemented in Jenkins, but they want to move toward doing automated deployments as well. You have been asked to build a Jenkins Pipeline project called train-schedule to build the train-schedule app. The pipeline project needs to build the code, run the automated tests, and archive the deployable zip artifact that is created by the build.

To do this, you will need to:

* Fork the source repo.
* Create a Jenkinsfile for the app.
* Create a stage to execute the gradle build.
* Archive the *dist/trainSchedule.zip* artifact.
* Create new multibranch pipeline project in Jenkins called train-schedule.
* Run the build successfully in Jenkins.

Feel free to look at the *example-solution* branch of the git repo for an example of the source code changes needed for the solution.

Some useful links:

* Jenkins Pipelines documentation: <https://jenkins.io/doc/book/pipeline/>

**BuilGraphical user interface, text

Description automatically generatedd the Master Branch of the train-schedule Pipeline Project**

Create a personal fork of the sample source code repository, which you can find [here](https://github.com/linuxacademy/cicd-pipeline-train-schedule-pipelines).

Click **Create new file**. Name the file **Jenkinsfile**.

Enter:

pipeline {

agent any

stages {

stage ('Build') {

steps {

echo 'Running build automation'

sh './gradlew build --no-daemon'

archiveArtifacts artifacts: 'dist/trainSchedule.zip'

}

}

}

}

Click **Commit new file** at the bottom.

Before we can test this in Jenkins, we need to get an API key from GitHub so Jenkins can pull down this source code from GitHub. To do this:

* Click your avatar in the top right corner
* Click **Settings**.
* Click **Developer settings**.
* Click **Personal access tokens**.
* Click **Generate new token**.
* *Token description*: **jenkins**
* Check the box next to **admin:repo\_hook**.
* **Generate token**

Copy the generated token to the clipboard.

**Create a New Multibranch Pipeline Project in Jenkins**

Navigate to the Jenkins server by entering the public IP address listed on the lab page, followed by :8080, in a browser.

In Jenkins, click **New Item**.

For the item name, type train-schedule, and select **Multibranch Pipeline**. Click **OK**.

Under *Branch Sources*, click **Add source**, and select **Git**.

Under *Credentials*, click **Add**, and select **Jenkins**.

* *Username*: Enter your GitHub username.
* *Password*: Paste in the API key you copied before.
* *ID*: **github\_key**
* *Description*: **GitHub key**
* Click **Add**.

In the new *Git* section under *Branch Sources*, click the *Credentials* dropdown and select the credential you just created. For *Project Repository*, copy and paste in the URL of your personal fork of the source code repository. Click **Save**.

The initial build will take several minutes to complete.

**Conclusion**

Congratulations on completing this lab!