CI/CD with Jenkins Pipeline

Pierre-Luc Maheu

What is Jenkins Pipeline?

- Pipeline defined as a Groovy DSL

- Successor of the UI driven configuration

- By convention, the build script a file called Jenkinsfile

Declarative vs scripted

DSL based

Opinionated

Declarative

Since 2017

Groovy script

DIY style

Imperative

Since 2014

```
pipeline {
    agent { docker { image 'node:6.3' } }
    stages {
        stage('build') {
            steps {
                sh 'npm --version'
            }
        }
    }
}
```

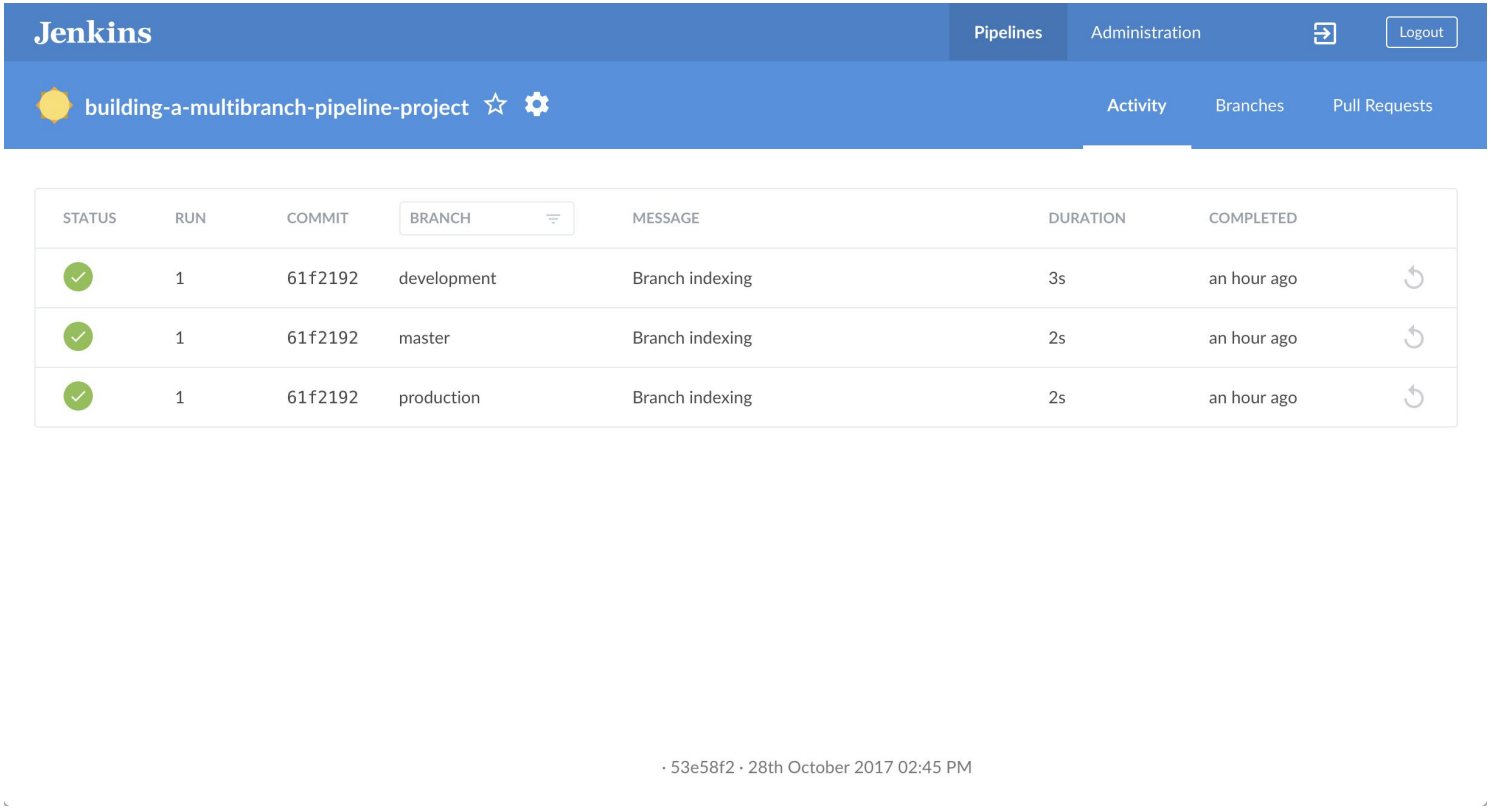
Multibranch pipeline

First class support for source control

Builds PRs out of the box.

Webhook trigger

Blue Ocean


Jenkins
Pipelines
Administration
Logout
building-a-multibranch-pipeline-project
Activity
Branches
Pull Requests
STATUS
RUN
COMMIT
BRANCH
MESSAGE
DURATION
COMPLETED
1
61f2192
development
Branch indexing
3s
an hour ago
1
61f2192
master
Branch indexing
2s
an hour ago
1
61f2192
production
Branch indexing
2s
an hour ago
· 53e58f2 · 28th October 2017 02:45 PM

Plugins

Plugins add features to Jenkins

Examples: AWS EC2, GitHub, Slack

Pipelines themselves are a plugin.

```
stage('API tests (PRs or dev)') {
    when {
        anyOf {
            branch 'dev'
            changeRequest target: 'dev'
            changeRequest target: 'release-*', comparator: 'GLOB'
        }
    }
     steps { ...
```

Executors

Basic allocation slot for Jenkins agents

Can be addressed with labels

Lightweight executors are threads and do not consume an agent slot

Docker-backed Executor

Any container can be used as an executor

Doesn’t require plugin configuration for each tool beside Docker itself

Examples: Database, NodeJS, Selenium grid, AWS CLI, Python

```
pipeline {
    agent {
        docker { image 'java11' }
    }
    stages {
        stage('Test') {
            steps {
                sh 'node --version'
            }
        }
    }
}
```

Reusable libraries

Libraries can be shared between pipelines

Handy to change many pipelines or branches at once

```
@Library('somelib')
import com.mycorp.pipeline.somelib.Helper

int useSomeLib(Helper helper) {
    helper.prepare()
    return helper.count()
}

echo useSomeLib(new Helper('some text'))
```

Parallel Execution

Stages can be executed in parallel

Can use same executor or split to other agents

```
stage('Run Tests') {
    parallel {
        stage('Test On Windows') {
            agent {
                label "windows"
            }
            steps {
                bat "run-tests.bat"
            }
            post {
                always {
                    junit "**/TEST-*.xml"
                }
            }
        }
        stage('Test On Linux) {
...
```

```
def testList = ["a", "b", "c", "d"]
def branches = [:]

for (int i = 0; i < 4 ; i++) {
       int index=i, branch = i+1
       stage ("branch_${branch}"){
            branches["branch_${branch}"] = {
                node ('label_example'){
                    sh "testListVal: " + testList[index] + "'"
                }
            }
      }
}

parallel branches
```

```
def sendSlackNotification() {
   if (env.CHANGE_AUTHOR != null) {
       // Github/Slack username mappings.
       def githubToSlack = [:]
       githubToSlack."plmaheu-github" = "plmaheu-slack"
       def slackUsername = githubToSlack.getOrDefault(env.CHANGE_AUTHOR, env.CHANGE_AUTHOR)

       slackSend(color: 'red', channel: "@${slackUsername}", message: ":warning: Build
${BUILD_NUMBER} failed for ${BRANCH_NAME}. See ${BUILD_URL}.")
   } else {
       slackSend(color: 'red', channel: "#failedagain", message: ":warning: Build
${BUILD_NUMBER} failed for ${BRANCH_NAME}. See ${BUILD_URL}.")
   }
}
```

```
stage('Analysis') {
    when { not { changeRequest() } }
    steps {
        withSonarQubeEnv('SonarCloud') {
            sh 'mvn org.sonarsource.scanner.maven:sonar-maven-plugin:3.6.0.1398:sonar' +
                    ' -Dsonar.projectKey=someproject' +
                    ' -Dsonar.organization=someorg' +
                    ' -Dsonar.branch.name=${BRANCH_NAME}'
        }
    }
}
```

```
withCredentials([string(credentialsId: 'mytoken', variable: 'TOKEN')]) {
  sh '''
    curl -H "Token: $TOKEN" https://some.api/
  '''
}
```

```
stage ('push artifact') {
    steps {
       archiveArtifacts artifacts: '/output/*.jar', fingerprint: true
    }
}
```

```
 stage('pull artifact') {
     steps {
         copyArtifacts filter: '*.jar', fingerprintArtifacts: true,
projectName: 'projectname/master'
       }
 }
```

Resources

https://jenkins.io/doc/

https://jenkins.io/doc/pipeline/examples

Thank you!