ToiletPaper #110

Groovy Jenkins Pipeline, Baby!

Author: Christof Huber / Software Developer / Business Division Automotive Bavaria

✕ Problem

Jenkins is a widely used CICD tool. Working with many microservices can be very complex.
Small changes, like a new URL, can be a tedious task, because it has to be done for every job.
Due to the missing changelog, it's also hard to track which changes were made and by whom.

✓ Solution

With the plugin suite "Jenkins Pipeline" you can define different jobs as **Groovy code**. You can then check this into your preferred version control and maintain and further develop it in addition to the project code.
In connection with a multi-branch pipeline, all branches of a project are automatically built according to the instructions in the 'Jenkinsfile'.

➔ Example

A project with multiple services, all built with Maven. A merge on the branch develop is to build a new version.

Coffee-Service, Food-Service: Jenkinsfile

```
def pipeline
stage('Load pipeline') {
    // Lade die Pipeline vom gemeinsam genutzten Repository
    fileLoader.withGit(
    'https://url-to-pipeline-repo.git',
    'master',
    'in-jenkins-hinterlegte-credentials-id') {

        // Jeder Service kann pipeline.groovy benutzen
        pipeline = fileLoader.load('pipeline.groovy')
    }
}
pipeline.execute()
```

Pipeline Repo: pipeline.groovy

```
def execute() {
    stage('Checkout') {
        checkout scm
    }
    stage('Build service') {
        sh "mvn clean install"
    }

    // Die Variable env.BRANCH_NAME wird automatisch auf den aktuellen Branchname gesetzt.
    // Durch dieses Feature ist die Pipeline so dynamisch, dass sie mit jedem neuen Branch umgehen kann.

    if ("develop".equals(env.BRANCH_NAME) {
        stage('Release') {
            sh "mvn release:prepare release:perform"
        }
    }
}
```

✚ Further Aspects

- https://jenkins.io/doc/book/pipeline/