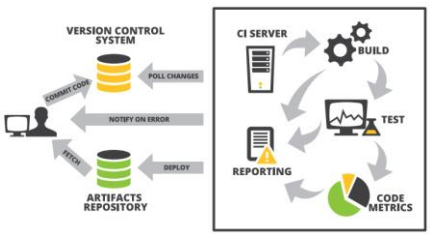
Continuous integration and delivery with Jenkins 2

Gilles QUERRET
Riverside Software

About The Speaker

- Started Riverside Software 10 years ago
- Based in Lyon, France
- Continuous integration and technical expertise around Java / OpenEdge
- Code analysis for OpenEdge

define:continuous integration

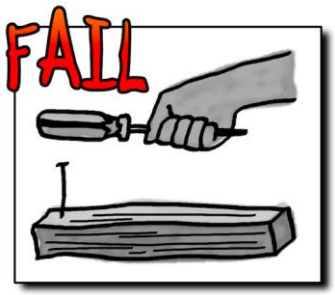


define:continuous delivery

define:continuous deployment

- Continuous delivery PLUS…

Use The Right Tools



Hudson History

- Hudson first release in 2005
  - Single location for configuration files
  - Master / Slave
  - SCM integration
  - History
  - Very good replacement for cron / scheduled tasks


Version 0.2 (07/15/2003)

- Documentation included in tarball & mirror changes
- PCTCreateBase : added destDir attribute
- PCTCompile : logs how many files were compiled or not
- PCTCompile : automatically creates subdirectories during the build process
- PCTCompile : uses XANT to compile only modified files
- PCTDumpSchema, PCTLoadSchema, PCTSchemaDoc : changed the way it handles PCTConnection
- PCTLibrary : added the noCompress attribute (false by default)

Version 0.1 (06/05/2003)

- First alpha release

Hudson / Jenkins

- Oracle took over Sun Microsystems
- Kohsuke Kawaguchi left Oracle, but Oracle had trademark over Hudson
- Renamed to Jenkins in 2011
- Both parties consider the other one is a fork…

Jenkins 2

- Released in 2016
- Pipelines were introduced in 1.x
- First-class citizens in 2.x

Why Pipelines? The Problem

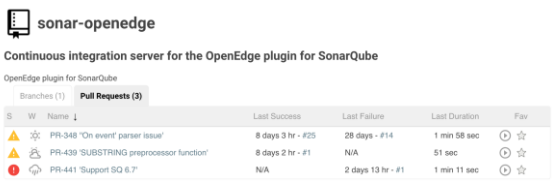
- Versioning of jobs
- Job dependencies
- Jobs split on multiple nodes
- Job management with large number of branches

The Solution

- DSL for the job description
  - Based on Groovy
- Stored in SCM
  - Keep history of the job
- Each branch execute the same job

Multi-Branch Pipeline Configuration

- Simplified configuration over free-style
  - SCM location
  - Script location inside SCM



Scripted vs Declarative

- Scripted pipelines let you use the full power of Groovy
- Declarative pipelines require a structured syntax, allowing validation and visual editor

```
pipeline {
  agent { label 'windows' }
  options {
    buildDiscarder(logRotator(numToKeepStr:'5'))
    timeout(time: 30, unit: 'MINUTES')
  }
```

```
stages {
  stage ('Build') {
    steps {
      withEnv(["PATH+ANT=${tool name: 'Ant 1.9', type: 'hudson.tasks.Ant$AntInstallation'}/bin",
               "DLC=${tool name: 'OpenEdge-11.7', type: 'jenkinsci.plugin.openedge.OpenEdgeInstallation'}"]) {
        bat "ant -DDLC=%DLC% -lib PCT.jar build dist"
      }
      archiveArtifacts artifacts: 'target/DataDigger.zip'
    }
  }
```

```
16      post {
17        always {
18          mail to:" gquerret@riverside-software.fr",
19               subject: "${currentBuild.fullDisplayName} DD build executed",
20               body: "Empty..."
21        }
22        failure {
23          mail to: "gquerret@riverside-software.fr",
24               subject: "${currentBuild.fullDisplayName} build failure!",
25               body: "Empty..."
26        }
27      }
```

```
37      stage ('Test install') {
38        agent {
39          node {
40            label 'windows'
41            customWorkspace "Z:\\TestDeployment\\DD-${BRANCH_NAME}"
42          }
43        }
```

```
44          steps {
45            unstash name: 'windows-build'
46            unzip zipFile: 'target/DataDigger.zip'
47          }
```

Demo

Environment Variables

- Multiple variables available:
  - BRANCH_NAME
  - CHANGE_ID
  - NODE_NAME
  - JOB_NAME and JOB_BASE_NAME
- Documentation available in the Pipeline Syntax help

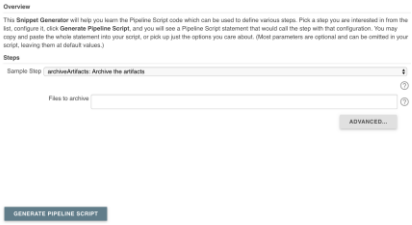
Environment Variables - Security

```
stages {
    stage('Example') {
        environment { 2
            AN_ACCESS_KEY = credentials('my-prefined-secret-text') 3
        }
        steps {
            sh 'printenv'
        }
    }
}
```

- For login / password, environment variables are AN_ACCESS_KEY_USR and AN_ACCESS_KEY_PSW

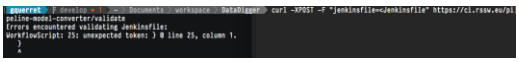
More tasks…

- Pipeline syntax button available from the main page of a Pipeline job



Script validation

- Tired of waiting for job to fail with a syntax error ?
- curl -XPOST -F "jenkinsfile=<Jenkinsfile" https://jenkins-server/pipeline-model-converter/validate



Blue Ocean

- Did you know that Jenkins could have a good look and feel ?
- Project started a long time ago, and still in development

Blue Ocean

Migration

- To quote multiple Cloudbees engineers, it's time to upgrade to Jenkins 2
- And to move away from freestyle jobs

OpenEdge

- Build steps
  - Ant is supported in pipelines jobs (and PCT)
  - Use the OpenEdge Jenkins plugin
  - Docker images for OpenEdge ?
- Unit tests
  - Just make sure you create a JUnit-compatible file

OpenEdge

- Deployment
  - Create full packages
  - Prepare clean-up script
  - Execute DB update script
  - Prepare startup script
- Code analysis
  - Use SonarQube ☺

OpenEdge - Caveats

- Resource allocation for databases can be difficult
  - Port ranges have to be allocated in order to allow TCP/IP connection (one port for the broker, and one port per server)
- Legacy AppServer has the same problem
- PASOE is easier to deal with (only one port)

Pipelines

- Declarative pipelines can also include the scripted pipeline syntax in the *script* node
  - In case you need to execute logic on variables for example (for loop, if conditions, …)
- Or use libraries

Questions ?

- Jenkins : http://jenkins-ci.org
- https://github.com/jenkinsci/pipeline-model-definition-plugin/wiki/getting-started
- https://jenkins.io/doc/book/pipeline/syntax/

- And contact me directly (g.querret@riverside-software.fr) if you want additional informations