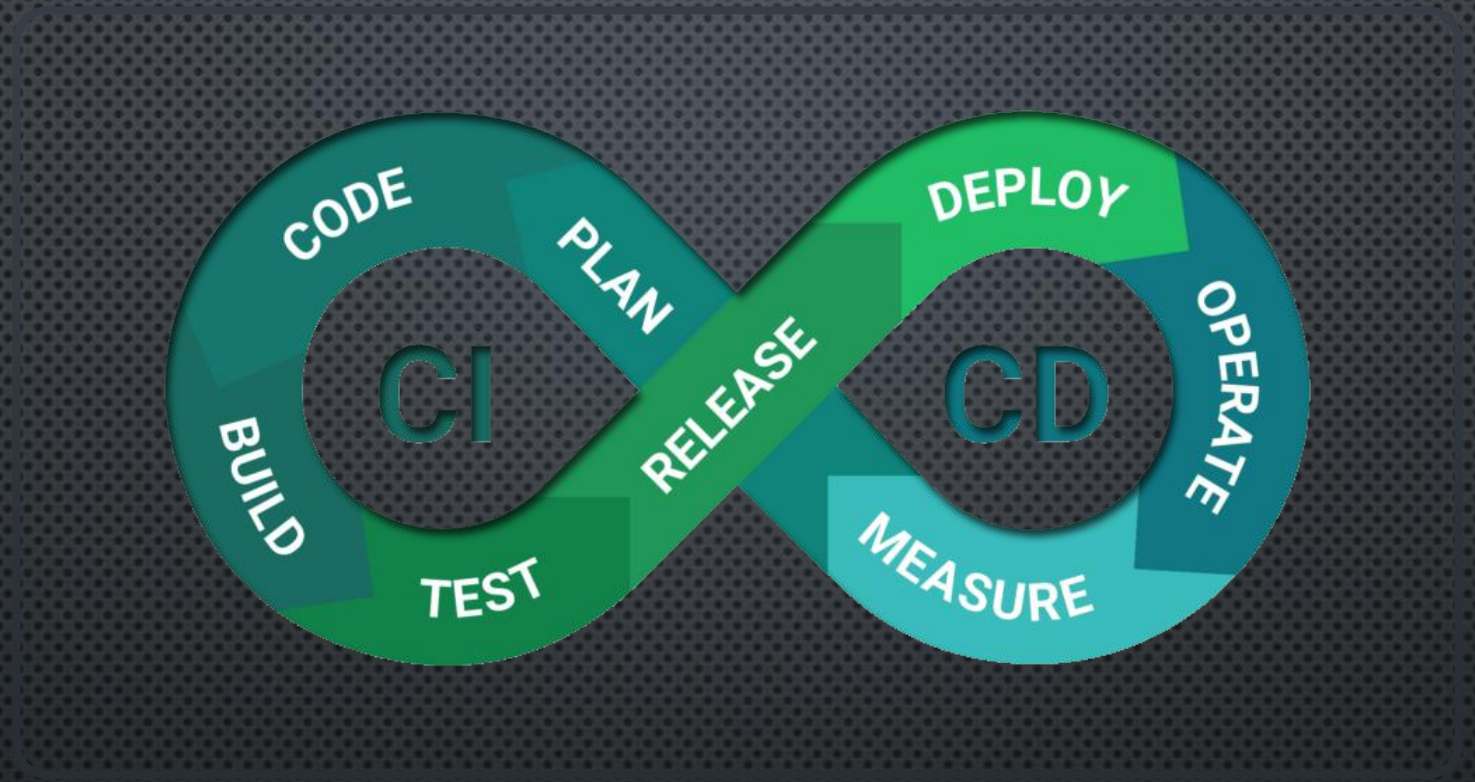
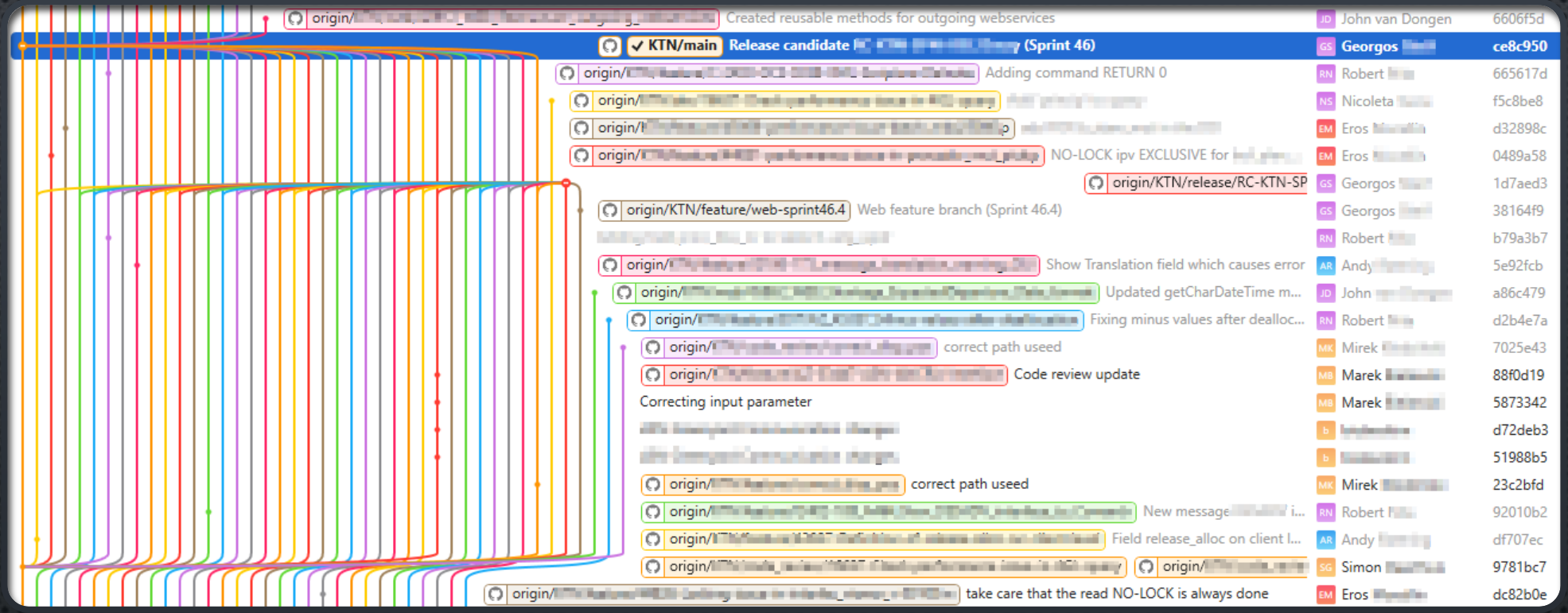
JENKINS PIPELINES

GILLES QUERRET • RIVERSIDE SOFTWARE

US PUG CHALLENGE 2024

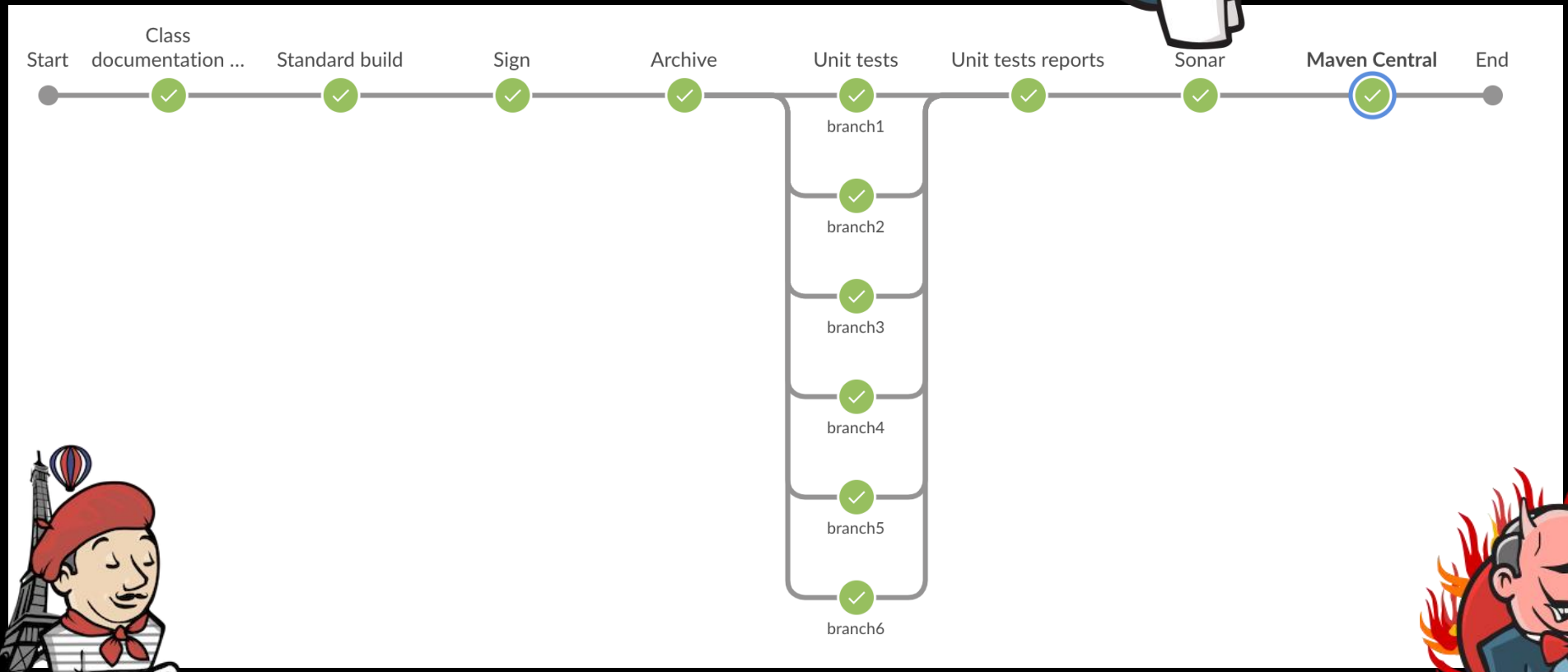


CI & CD



GIT BRANCHES

JENKINS


Start
Class documentation ...
Standard build
Sign
Archive
Unit tests
Unit tests reports
Sonar
Maven Central
End
branch1
branch2
branch3
branch4
branch5
branch6

Jenkinsfile

```
pipeline {
  agent any

  stages {
    stage ('Build') {
      steps {

      }
    }

    stage ('Code Analysis') {
      steps {

      }
    }

    stage ('Test') {
      steps {

      }
    }

    stage ('Deploy') {
      steps {

      }
    }

  }
}

```

JENKINS PIPELINE

JENKINS PIPELINE

Jenkinsfile

```
pipeline {
  agent any

  stages {
    stage ('Build') {
      steps {
        echo "Create DB, execute Ant / PCT, generate ZIP file"
      }
    }

    stage ('Code Analysis') {
      steps {
        echo "Execute SonarQube"
      }
    }

    stage ('Test') {
      steps {
        echo "Execute unit tests, regression tests, performance tests"
      }
    }

    stage ('Deploy') {
      steps {
        echo "Deployment to test environment"
      }
    }

  }
}
```

```
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Create DB, execute Ant / PCT, generate ZIP file
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Code Analysis)
[Pipeline] echo
Execute SonarQube
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Execute unit tests, regression tests, performance tests
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deployment to test environment
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline

GitHub has been notified of this commit’s build result
```

JENKINS PIPELINE

```
[Pipeline] node
Running on Windows-Office in C:\Jenkins-Public\workspace\SonarOETest-Private_Test
[Pipeline] {
```

```
Running on Linux-Office in /srv/jenkins-slave/workspace/SonarOETest-Private_Test
[Pipeline] {
```

JENKINS PIPELINE

PIPELINE: BASIC STEPS

- Pipeline: Basic Steps
  - `catchError`: Catch error and set build result to failure
  - `deleteDir`: Recursively delete the current directory from the workspace
  - `echo`: Print Message
  - `error`: Error signal
  - `fileExists`: Verify if file exists in workspace
  - `isUnix`: Checks if running on a Unix-like node
  - `mail`: Mail
  - `pwd`: Determine current directory
  - `readFile`: Read file from workspace
  - `retry`: Retry the body up to N times
  - `sleep`: Sleep
  - `stash`: Stash some files to be used later in the build
  - `step`: General Build Step
  - `timeout`: Enforce time limit
  - `tool`: Use a tool from a predefined Tool Installation
  - `unstable`: Set stage result to unstable
  - `unstash`: Restore files previously stashed
  - `waitUntil`: Wait for condition
  - `warnError`: Catch error and set build and stage result to unstable
  - `withEnv`: Set environment variables
  - `wrap`: General Build Wrapper
  - `writeFile`: Write file to workspace
  - `archive`: Archive artifacts
  - `getContext`: Get contextual object from internal APIs
  - `unarchive`: Copy archived artifacts into the workspace
  - `withContext`: Use contextual object from internal APIs within a block

- Pipeline: Nodes and Processes
  - `bat`: Windows Batch Script
  - `dir`: Change current directory
  - `node`: Allocate node
  - `powershell`: Windows PowerShell Script
  - `pwsh`: PowerShell Core Script
  - `sh`: Shell Script
  - `ws`: Allocate workspace

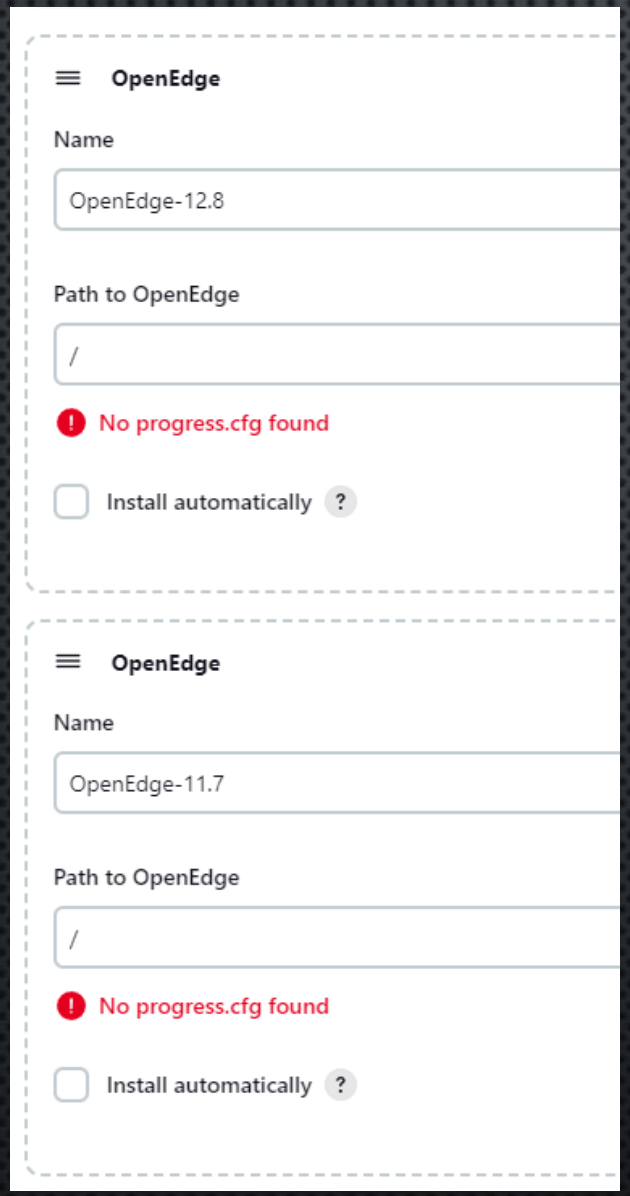
```
stage ('Build') {
  steps {
    script {
      echo "Create DB, execute Ant / PCT, generate ZIP file"
      withAnt(installation: 'Ant 1.10') {
        withEnv(["DLC=${tool name: 'OpenEdge-12.8', type: 'openedge'}"]) {
          if (isUnix()) {
            sh 'ant -lib xmltask.jar -lib $DLC/pct/PCT.jar -DDLC=$DLC build'
          } else {
            bat 'ant -lib xmltask.jar -lib %DLC%\\pct\\PCT.jar -DDLC=%DLC% build'
          }
        }
      }
    }
  }
}
```

PIPELINE: BASIC STEPS

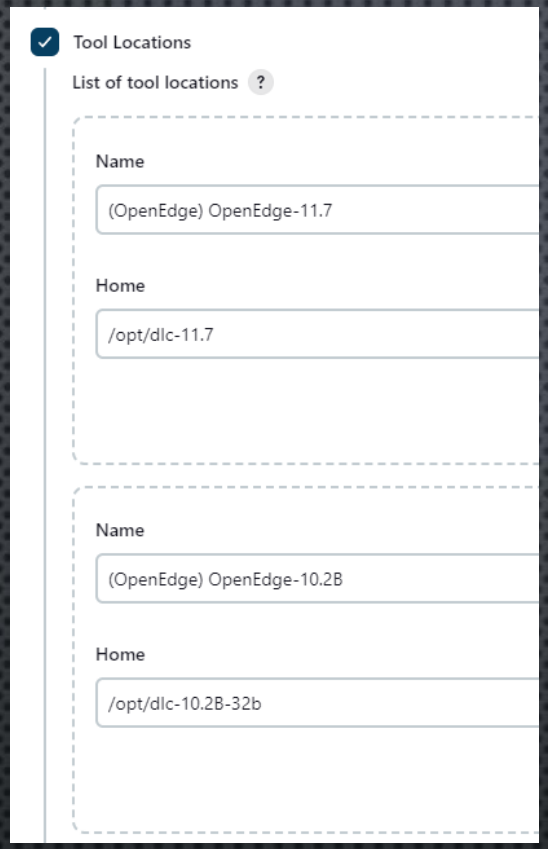
```
 > C:\Program Files\Git\bin\git.exe rev-list --no-walk 4d69aeb64d4d9b6c6a8aec7b5f85a166e371cda1 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] script
[Pipeline] {
[Pipeline] echo
Create DB, execute Ant / PCT, generate ZIP file
[Pipeline] withAnt
[Pipeline] {
[Pipeline] tool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] isUnix
[Pipeline] bat

C:\Jenkins-Public\workspace\SonarOETest-Private_Test>ant -lib xmltask.jar -lib C:\Progress\OpenEdge-12.8\pct\PCT.jar -DDLC=C:\Progress\OpenEdge-12.8 build
Buildfile: C:\Jenkins-Public\workspace\SonarOETest-Private_Test\build.xml

init:
    [mkdir] Created dir: C:\Jenkins-Public\workspace\SonarOETest-Private_Test\db
    [mkdir] Created dir: C:\Jenkins-Public\workspace\SonarOETest-Private_Test\build
    [mkdir] Created dir: C:\Jenkins-Public\workspace\SonarOETest-Private_Test\profiler
[PCTVersion] PCT Version : pct-228-main-0308a4d5
     [echo] OpenEdge Release 12.8.3 as of Thu May 30 13:49:27 EDT 2024
```

PIPELINE: BASIC STEPS

PIPELINE: TOOLS

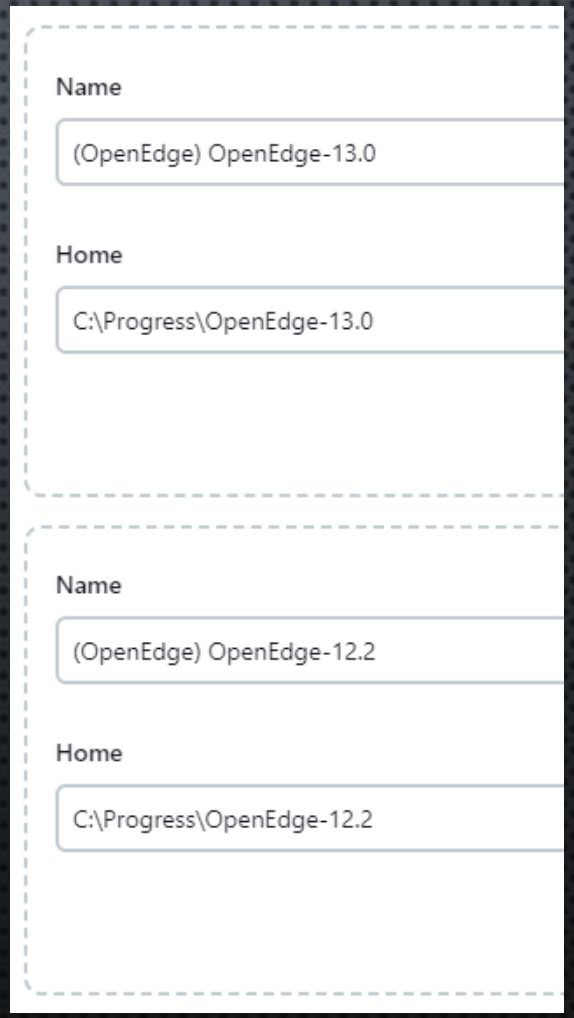


Master



Linux agent

Windows agent



new.Jenkinsfile

```
1  pipeline {
2    agent any
3
4    tools {
5      ant 'Ant 1.10'
6      openedge 'OpenEdge-12.8'
7    }
8
```

PIPELINE: TOOLS (ALT)

The following variables are available to shell and batch build steps:

BRANCH_NAME
For a multibranch project, this will be set to the name of the branch b
corresponding to some kind of change request, the name is generally
BRANCH_IS_PRIMARY
For a multibranch project, if the SCM source reports that the branch
than one branch as a primary branch while others may not supply thi
CHANGE_ID
For a multibranch project corresponding to some kind of change requ
CHANGE_URL
For a multibranch project corresponding to some kind of change requ
CHANGE_TITLE
For a multibranch project corresponding to some kind of change requ
CHANGE_AUTHOR
For a multibranch project corresponding to some kind of change requ
CHANGE_AUTHOR_DISPLAY_NAME
For a multibranch project corresponding to some kind of change requ
CHANGE_AUTHOR_EMAIL
For a multibranch project corresponding to some kind of change requ
CHANGE_TARGET
For a multibranch project corresponding to some kind of change requ
unset.

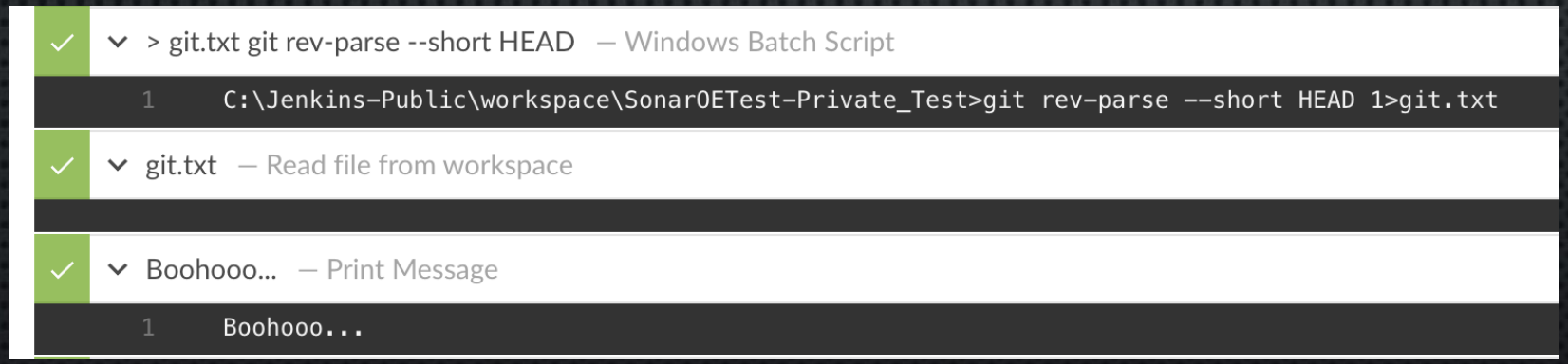
PIPELINE: ENVIRONMENT VARIABLES

```
stage ('Build') {
  when {
    expression {
      BRANCH_NAME == 'Test'
    }
  }
  steps {
    script {
      echo "Create DB, execute Apt /
```

PIPELINE: VARIABLES, CONDITIONS, …

```
bat "> git.txt git rev-parse --short HEAD"
def commit = readFile('git.txt').trim()
if (commit.startsWith("1")) {
  println "Commit ID starts with 1"
} else {
  println "Boohooo..."
}
```

PIPELINE: GROOVY IS JUST JAVA



```groovy
def suffix = env.BRANCH_NAME.contains("foobar") ? "XX" : "YY"
long days = Long.parseLong(java.time.LocalDateTime.now().format(
  java.time.format.DateTimeFormatter.ofPattern('d')))
long seconds = Long.parseLong(java.time.LocalDateTime.now().format(
  java.time.format.DateTimeFormatter.ofPattern('A'))) / 1000
def artifactVersionNumber = java.time.LocalDateTime.now().format(
  java.time.format.DateTimeFormatter.ofPattern('YYYY.M.')) + ((days * 86400) + seconds) + suffix;
println "Version number: ${artifactVersionNumber}"
```

✓ > git.txt git rev-parse --short HEAD — Windows Batch Script

✓ git.txt — Read file from workspace

✗ Boohooo... — Print Message

```
1  Boohooo...
2  Scripts not permitted to use staticMethod java.lang.Long parseLong java.lang.String
```

PIPELINE: GROOVY IS JUST JAVA

PIPELINE: GROOVY IS JUST JAVA

No pending script approvals.

You can also remove all previous script approvals: Clear Approvals

Approve / Approve assuming permission check / Deny signature : `staticMethod java.lang.Long parseLong java.lang.String`

Signatures already approved:

```
method hudson.model.AbstractItem setDisplayName java.lang.String
method hudson.plugins.git.GitSCM getBranches
method hudson.plugins.git.GitSCM getUserRemoteConfigs
method hudson.plugins.git.GitSCMBackwardCompatibility getExtensions
method java.lang.Runtime availableProcessors
method jenkins.model.Jenkins getItemByFullName java.lang.String
new groovy.util.AntBuilder
```

Version number: 2024.9.1756696YY — Print Message

```
1  Version number: 2024.9.1756696YY
```

PIPELINE: CHECK SYNTAX

```
curl -k -X POST -F "jenkinsfile=<Jenkinsfile" \
https://jenkins.yourcompany.com/pipeline-model-converter/validate
```

```
  def testBranch(nodeName, jdkVersion, antVersion, dlcVersion, stashCoverage, label, dockerImg) {
    node(nodeName) {
      ws {
        deleteDir()
        def dlc = tool name: dlcVersion, type: 'openedge’
        …
```

PIPELINE: FUNCTIONS

```
stage('Unit tests') {
  steps {
    parallel
      branch1: { testBranch('Windows-Office', 'JDK8', 'Ant 1.10', 'OpenEdge-11.7', true, '11.7-Win', '') },
      branch2: { testBranch('Windows-Office', 'Corretto 11', 'Ant 1.10', 'OpenEdge-12.2', true, '12.2-Win', '')
      branch3: { testBranch('Windows-Office', 'JDK17', 'Ant 1.10', 'OpenEdge-12.8', true, '12.8-Win', '') },
      branch4: { testBranch('Linux-Office03', 'JDK8', 'Ant 1.10', 'OpenEdge-11.7', false, '11.7-Linux', 'docker.
      branch5: { testBranch('Linux-Office03', 'Corretto 11', 'Ant 1.10', 'OpenEdge-12.2', false, '12.2-Linux', '
      branch6: { testBranch('Linux-Office03', 'Corretto 11', 'Ant 1.10', 'OpenEdge-12.8', true, '12.8-Linux', 'd
      failFast: false
  }
}
```

PIPELINE: EXTERNAL GROOVY SCRIPTS

script.groovy U

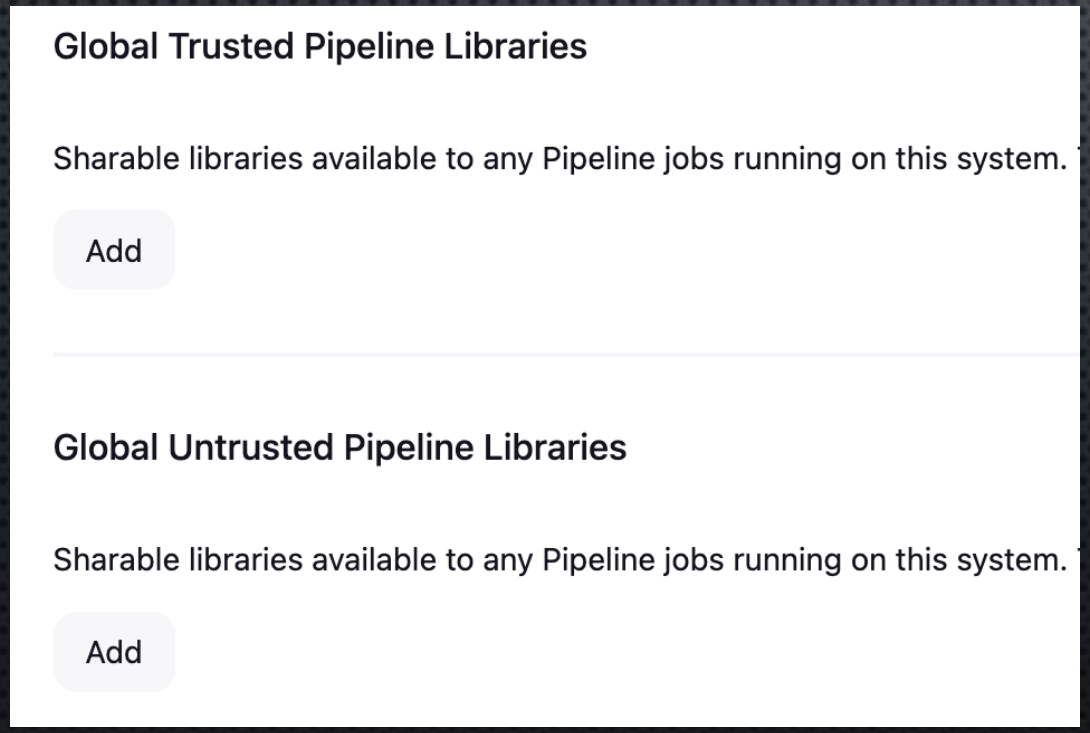
script.groovy

```groovy
def prettyMessage(msg) {
    echo "(((((((((  ${msg}  )))))))))"
    echo "---------"
}

return this // Don't remove this line !!

```

```groovy
def ext = load("script.groovy")
ext.prettyMessage(artifactVersionNumber)
```

```
> Version number: 2024.9.1758167YY — Print Message
> ((((((((( 2024.9.1758167YY ))))))))) — Print Message
> --------- — Print Message
```

```
@Library('my-shared-library')
@Library('my-shared-library@1.0.1')
@Library('my-shared-library@1.0.1’,
         'slib2')
```



PIPELINE: SHARED LIBRARIES

```
parameters {
  string(name: 'name', defaultValue: 'Jenkins',
         description: 'Target name')
  choice(name: 'environment', choices: ['UAT1', 'UAT2', 'QA1', 'QA2'],
         description: 'Target environment')
}
```

```
stage ('Deploy') {
  when {
    expression {
      BRANCH_NAME == 'Test'
    }
  }
  steps {
    echo "We'll deploy to environment ${params.environment}"
    echo "   with parameter ${params.name}"
  }
}
```

PIPELINE: PARAMETERS

PIPELINE: PARAMETERS

Branch Test

This build requires parameters:

name

Target name

Gilles

environment

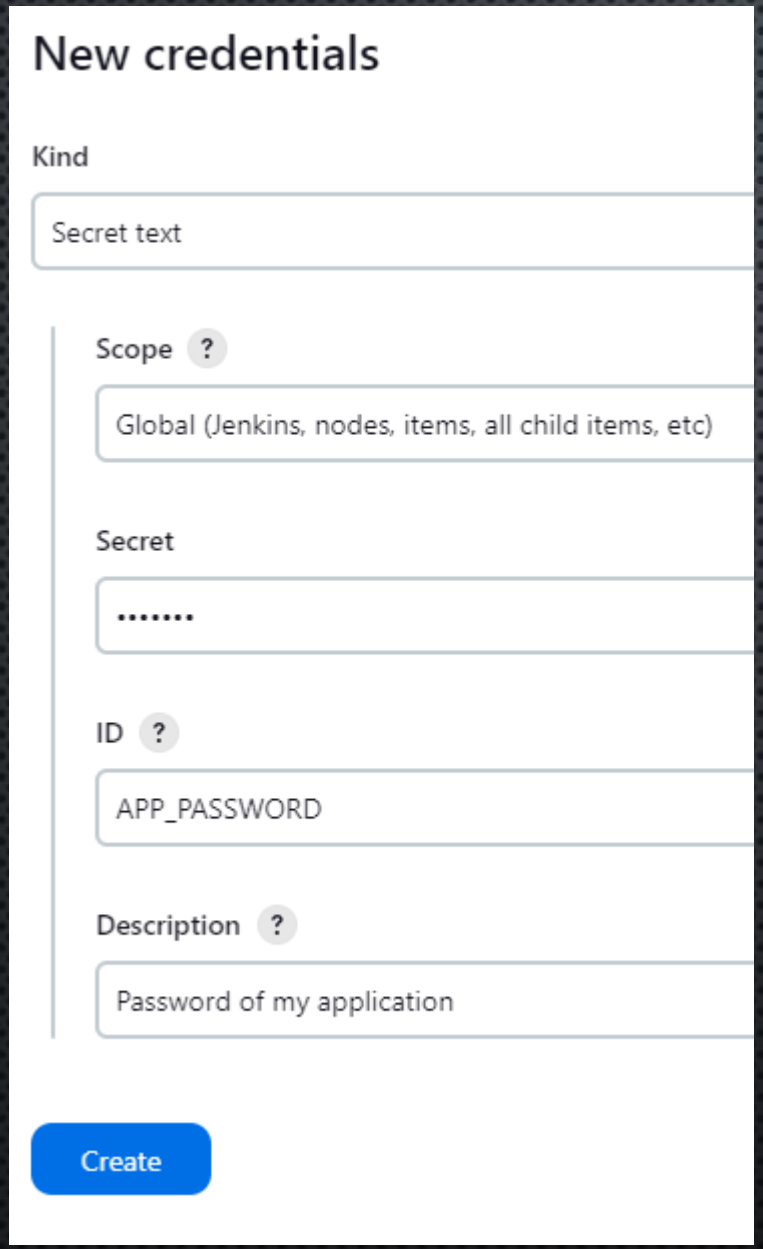
Target environment

QA1

Build | Cancel

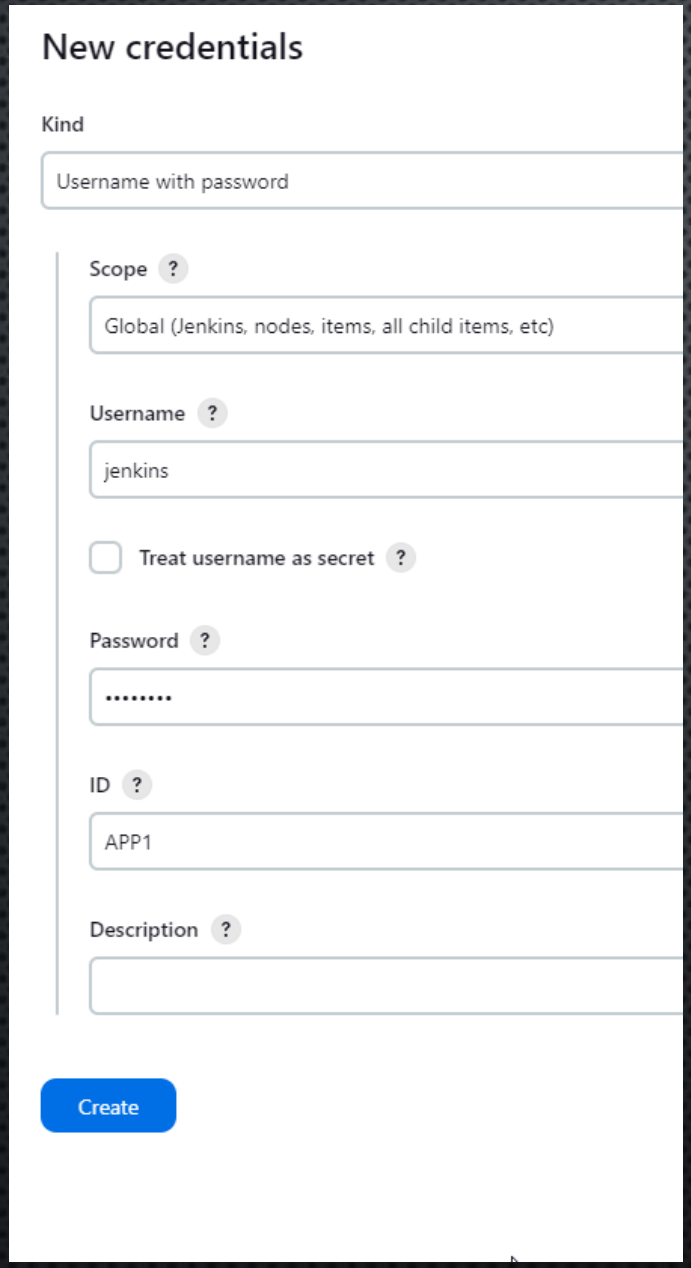
Deploy - <1s

- We'll deploy to environment QA1 — Print Message
- with parameter Gilles — Print Message




```
environment {
  CRED1 = credentials('ID')
}
```

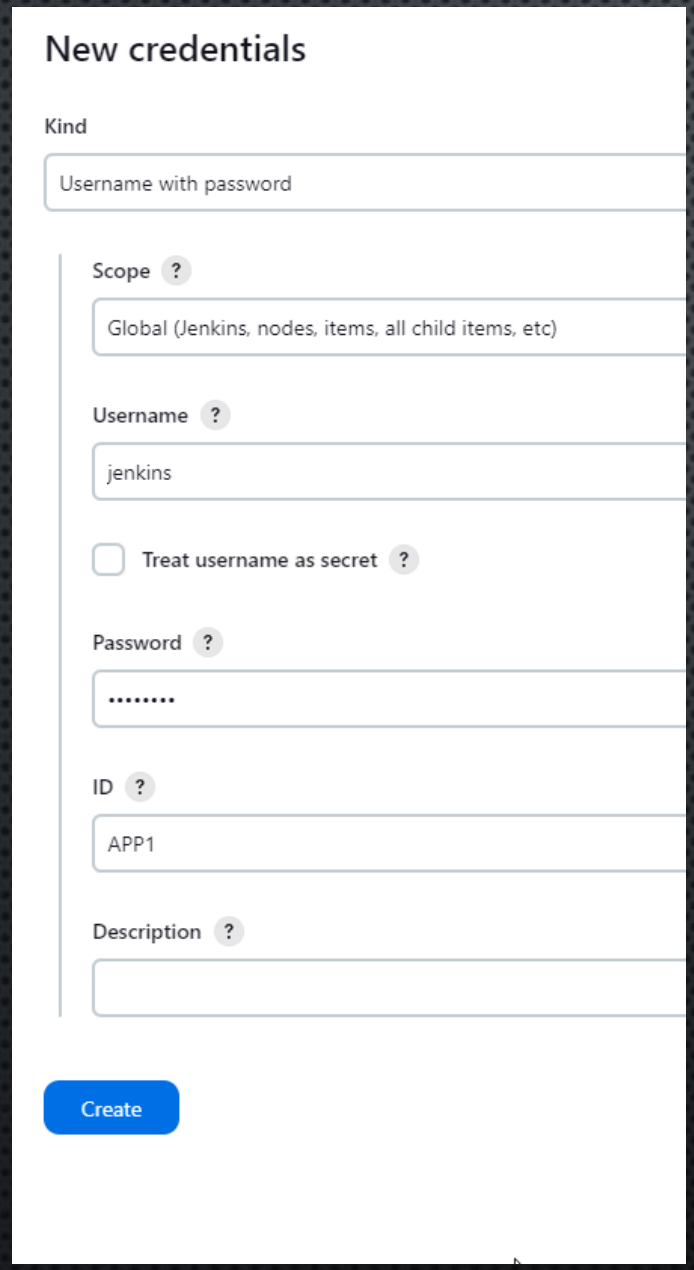
PIPELINE: CREDENTIALS




```groovy
withCredentials([usernamePassword(credentialsId: 'APP1',
                 usernameVariable: 'USERNAME',
                 passwordVariable: 'PASSWORD')]) {
  echo "app login param -u $USERNAME -p $PASSWORD"
}
```

PIPELINE: CREDENTIALS

```
[Pipeline] withCredentials
Masking supported pattern matches of %PASSWORD%
[Pipeline] {
[Pipeline] echo
Warning: A secret was passed to "echo" using Groovy String interpolation, which is insecure.
                 Affected argument(s) used the following variable(s): [PASSWORD]
                 See https://jenkins.io/redirect/groovy-string-interpolation for details.
app login param -u jenkins -p ****
[Pipeline] }
[Pipeline] // withCredentials
```




```groovy
withCredentials([usernamePassword(credentialsId: 'APP1',
                 usernameVariable: 'USERNAME',
                 passwordVariable: 'PASSWORD')]) {
  echo 'app login param -u $USERNAME -p $PASSWORD'
}
```

```
[Pipeline] withCredentials
Masking supported pattern matches of %PASSWORD%
[Pipeline] {
[Pipeline] echo
app login param -u $USERNAME -p $PASSWORD
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] withCredentials
```

PIPELINE: CREDENTIALS

THANK YOU !

- Riverside Software: https://riverside-software.fr
- SonarQube: https://sonarsource.com
- CABL on Github: https://github.com/riverside-software/sonar-openedge
- VS Code: https://github.com/vscode-abl/vscode-abl
- contact@riverside-software.fr