



zenika  
<animés par la passion>

# Jenkins Configuration Details

Version 1.18-SNAPSHOT

2018-08-27

# Table of Contents

1. Server configuration .....	2
1.1. Plugins installation .....	2
1.2. Change Theme .....	3
1.3. Gerrit Trigger Configuration .....	3
1.4. Administration .....	5
1.4.1. Overall configuration .....	5
1.4.2. Tools configuration .....	5
1.5. Allow CSS on published HTML .....	6
1.6. SonarQube token .....	6
1.7. ssh key on remote server .....	6
1.8. Pipelines creations .....	7
1.8.1. The Review pipeline .....	7
1.8.2. The Deploy pipeline .....	9
1.8.3. The Deploy Int pipeline .....	12
1.8.4. The Release pipeline .....	12
1.9. Troobleshooting .....	15
1.9.1. Disk space usage > 90 % .....	15
2. Appendix .....	16
2.1. Revision marks .....	16

*Table 1. History*

Date	Author	Detail
2018-08-27	bcouetil	Asciidoc HTML look & feel changes
2018-08-23	bcouetil	Initial commit

# 1. Server configuration

Connect to Jenkins homepage.

## 1.1. Plugins installation



This has to be done only for a new Production Line

- Go to **Jenkins** → **Administration Jenkins** → **Gestion des plugins**
- Update all plugins which have an update available
- Select **Disponibles** (=available) and install :
  - Pipeline Maven Integration
  - Throttle Concurrent Builds Plug-in
    - To be able to force non concurrent builds
  - Xvnc
    - To have a virtual screen if needed in tests
  - Naginator
    - For retry on failure
  - Gerrit Trigger
    - To launch job on gerrit update
  - HTML Publisher plugin
    - To have the **Maven Reporting** link when "maven site" is launched
  - Monitoring
    - To see nice health data of Jenkins on <https://bpmfactory.s2-eu.nvx.com/jenkins/monitoring>
  - JUnit Attachments
    - for enhanced job reporting
  - Logstash
    - To send jenkins jobs output to logstash then elastic
  - diskcheck
    - Check filesystem space on slave before a build
  - disk-usage
    - Show disk usage per build, configuration in **Administrer Jenkins** → **[ Configurer le système ]** → **Utilisation du disque**
  - AnsiColor
    - To allow colors in build logs
  - Simple Theme Plugin

- to change Jenkins basic theme

## 1.2. Change Theme



This has to be done only for a new Production Line

- Have the Simple Theme Plugin installed
- Navigate **Administrer Jenkins** → **[ Configurer le système ]** → **Theme** section
  - URL of theme CSS = <https://cdn.rawgit.com/afonsof/jenkins-material-theme/gh-pages/dist/material-cyan.css>
    - see the author's page for other colors : <http://afonsof.com/jenkins-material-theme/>
  - Save

## 1.3. Gerrit Trigger Configuration



This has to be done only for a new Production Line

On Jenkins :

- Create the console-master job if not already existing
 

Create a new freestyle job.

Name it console-master

General

☒ **[ Restreindre où le projet peut être exécuté ]**

  - master

Put this **Build** → **[ Ajouter une étape au build ]** → **[ Exécuter un script shell ]** → paste this and save :

```
ssh-keygen -y -f /root/.ssh/id_rsa > /root/.ssh/id_rsa.pub
ls -lart /root/.ssh/
more /root/.ssh/id_rsa.pub
```

- Add 1 executor on the master node
  - **Home** → **[ État du lanceur de compilations ]** → **[ maître ]** → **Configurer**
- Execute the console-master
- Keep track of what the execution gave for later Gerrit configuration, example :

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDKGER5oLwkNhcCYtTzmUQooA+1mdrjIGi84AVs0HyNpsMqFBhkpxfImvopvKLYiztXUA15dwwDsPWq1tUcy/4N
WqKnMTQA57xxxT2r8suF/DVlH6fNn8T73mGz9+kT77FXHuaMfmDTqrwPngUYQMm2Y9kTjGhIcH/jseq6jCUawITA0s/6EUbs7jtJ/S+jMb6Ed60S7S/n
R3IzQwVrXMiQjDdFsL8RWEBQ54T4cNia/HMI8MK7mEEF5K008g4Ru3BIdk+VSisPUYFPmNc/tE12RyAjjvkcwWxrYqFEB5h6R1S0yWXAjCUzjv8T0ov4W
us+ZqNgqUMYtBBf+zQvQC1ub
```

- When finished, remove the executor from master node
- Create a local trigger server
  - **Home → Adminstrer Jenkins → Gerrit Trigger → Add New Server**
    - Gerrit Connection Setting
      - Name = local\_server
      - Hostname = gerrit
      - Frontend URL = <http://gerrit/>
      - SSH Port = 29418
      - Username = svc-fr-bpmfact
      - SSH Keyfile = /root/.ssh/id\_rsa
    - Gerrit Reporting Values
      - Verify = <vide>, 1, -1, -1, -1
      - Code Review = <vide>, 1, -1, -1, -1
    - Gerrit Verified Commandes
      - Started = vide
      - Successful =

```
gerrit review <CHANGE>,<PATCHSET> --message 'Build Successful (      ) <BUILDS_STATS>' --verified <VERIFIED>
```

- Failed =

```
gerrit review <CHANGE>,<PATCHSET> --message 'Build Failed ( _ ) <BUILDS_STATS>' --verified <VERIFIED>
```

- Unstable =

```
gerrit review <CHANGE>,<PATCHSET> --message 'Build Unstable ( ° ° ) <BUILDS_STATS>' --verified <VERIFIED>
```

- Not Built =

```
gerrit review <CHANGE>,<PATCHSET> --message 'No Builds Executed (      ,) <BUILDS_STATS>' --verified <VERIFIED>
```

- Save

On Gerrit :

- Connect with the technical user (svc-fr-bpmfact / Bpm-fact0ry)
  - You may have to use a secondary browser, since authentication is very persistent on Gerrit
- Click on the user top right → **Settings** → **SSH Public Keys** → **[ Add Key... ]**

- Add the public key content from Jenkins server (the one asked to be kept track earlier), starting with **ssh-rsa**

On Jenkins :

- Test the earlier configured connection of the trigger with **Test Connection** while editing `local_server`
- Restart jenkins with : <https://bpmfactory.s2-eu.nvx.com/jenkins/safeRestart>
- The Gerrit trigger should be up and running

## 1.4. Administration



This has to be done only for a new Production Line

### 1.4.1. Overall configuration

Connect to Jenkins configuration page : <https://bpmfactory.s2-eu.nvx.com/jenkins/configure>

#### Propriétés globales

- `JAVA_HOME` = `/usr/`

#### Jenkins Location

- Adresse email de l'administrateur système = `xxxxxx@nvx.com`

#### Extended E-mail Notification

- SMTP server = `smtp.nvx.fr`
- Default user E-mail suffix = `@nvx.com`

#### Notification par email

- Serveur SMTP = `smtp.nvx.fr`
- Suffixe par défaut des emails des utilisateurs = `@nvx.com`

Save.

### 1.4.2. Tools configuration

Connect to Jenkins tools configuration page : <https://bpmfactory.s2-eu.nvx.com/jenkins/configureTools/>

#### Maven

- Nom = Maven 3.5
- Version = 3.5.2

#### Logstash Plugin

- Indexer type = ELASTICSEARCH
- Host name = <http://frpardge.corp.nvx.com>

- Port = 9200
- Key = /jenkins/builds

Save.

## 1.5. Allow CSS on published HTML



This has to be done only for a new Production Line

- Create a pipeline "css-support"
- Build Triggers
  - Construire périodiquement
    - Planning = 0 10,15,20 \* \* \*
- Pipeline

```
println(System.getProperty("hudson.model.DirectoryBrowserSupport.CSP"))
System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")
println(System.getProperty("hudson.model.DirectoryBrowserSupport.CSP"))
```

- Uncheck **Use Groovy Sandbox** and save

## 1.6. SonarQube token

To be able to upload quality results to SonarQube, you have to create a token.

Go to SonarQube application on the PL → **YourName** → **My Account** → **Security** → Name = Jenkins → **Generate**

Now maven can upload results to SonarQube with something like :

```
mvn sonar:sonar -Dsonar.login=ab7451586619e21d0e2bb50389899ce3595e3 -Dsonar.host.url=http://sonarqube:9000/sonarqube
```

## 1.7. ssh key on remote server



This has to be done only for a new remote server

If you have a remote server where you deploy your artifacts for further developments or tests : \* note the result of the slavePrep.sh script under **Here is this server's ssh public key**. Here is an example

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDZRLfTsI+cTRjbhYhndvIOI3LsexMiJpwcBmeuJrISnEdh1LRPlviQjtI1h7NCihejVIPgvzyMVn3tMLsvABBXLtbV
FIetOudpJn+8isnYAWWaaqX2fce/BqjLC26ygR4n25sqTO/GE9AhV5uBPbYTr4HCrH9Wzd8nU13DXm8C0hxUKh1+Uwm47KB11fVH/boIUygocIRu1FXS9TJy
MU0qFf3GGmDXs56VTe4ZQtPBHJ1k1RXQQc6UIhTbdLpedo4Khvzr7TpdVZg13qXZt35/t7Gu41bImHS1N64TKhaxAYgCPjYKgl9tAWJpEkk3WzXghohLivIQ
PInu5h3uvckH jenkins@b43496a2520e
```



- Connect on the remote server via SSH
- add the key to ~/.ssh/authorized\_keys file

## 1.8. Pipelines creations

### 1.8.1. The Review pipeline

This will be the review pipeline with steps from checkout to quality check. This pipeline is a "pipeline as code".

Go to Jenkins home page :

<https://bpmfactory.s2-eu.nvx.com/jenkins>

- Click **New Item**
- Choose a name : CG-WM\_P1\_Review
- Choose Pipeline type

#### General

- Description = This is the review pipeline fired by Gerrit on non yet validated push
- Check **Supprimer les anciens builds**
  - Strategy = Log Rotation
  - Nombre de builds à conserver = 10

#### Build Triggers



In the field **Choose a Server**, **Any Server** won't work

- Choose **Gerrit event**

Gerrit Trigger

- Choose a Server = local\_server
- Trigger on = Patchset Created
- Gerrit Project
  - Type = Plain
  - Pattern = cg-wm
  - Branches
    - Type = Plain
    - Pattern = master

## Advanced Project Options

None.

## Pipeline

- Definition = Pipeline script

*Pipeline content to copy/paste*

```
#!/groovy

properties([
    buildDiscarder(logRotator(artifactDaysToKeepStr: '', artifactNumToKeepStr: '', daysToKeepStr: '', numToKeepStr: '7')),
    [$class: 'ThrottleJobProperty',
     categories: [],
     limitOneJobWithMatchingParams: false,
     maxConcurrentPerNode: 0,
     maxConcurrentTotal: 0,
     paramsToUseForLimit: '',
     throttleEnabled: false,
     throttleOption: 'project'],
    pipelineTriggers([
        Gerrit(customUrl: '',
        GerritProjects: [[branches: [[compareType: 'PLAIN', pattern: 'master']],
        compareType: 'PLAIN', disableStrictForbiddenFileVerification: false, pattern: 'cg-wm']],
        serverName: 'local_server',
        triggerOnEvents: [patchsetCreated(excludeDrafts: false, excludeNoCodeChange: false, excludeTrivialRebase: false)]
        )
    ])
])

node {
    timeout(30) {
        try {
            stage('Checkout') {
                cleanWs() // requires workspace cleanup plugin to be installed
                echo "***** Starting checkout of patchset ${GERRIT_PATCHSET_NUMBER} on change number ${GERRIT_CHANGE_NUMBER}"
                git username: 'svc-fr-cric', password: '*****', url: 'https://cric.pl.s2-eu.nvx.com/gerrit/cg-wm.git'
                def changeBranch = "change-${GERRIT_CHANGE_NUMBER}-${GERRIT_PATCHSET_NUMBER}"
                sh "git fetch origin ${GERRIT_REFSPEC}:${changeBranch}"
                sh "git checkout ${changeBranch}"

                def v = version(readFile('pom.xml'))
                echo "Building version ${v}"
            }
            stage('Compilation') {
                //slaves are wiped out randomly, so we prepare them on each execution
                sh '$WORKSPACE/src/scripts/slavePrep.sh'

                withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
                    //clean to deploy libs to local maven repository
                    sh "mvn clean dependency:purge-local-repository"
                    //The assembly is postponed : it needs some further generated PDF
                    sh "mvn install verify -DskipTests -Dassembly.skipAssembly=true"
                }
            }
            stage('Verification'){
                parallel (
                    "Unit Tests" : {
                        wrap([$class: 'Xvnc', takeScreenshot: false, useXauthority: true]) {
                            withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
                                sh "mvn test -s cg-settings.xml -Dcheckstyle.skip=true"
                                //Maven auto reports JUnit surefire results
                            }
                        }
                    }
                ),
            }
        }
    }
}
```

```

"Documentation" : {
  sh '$WORKSPACE/src/scripts/asciidocOnlyModified.sh'
  //get history from git to asciidoc documentation
  sh '$WORKSPACE/src/scripts/asciidocHistory.sh $WORKSPACE'
  withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
    //validate produces the date for PDF
    sh "mvn validate asciidoctor:process-asciidoc antrun:run@pdfsAddVersion -s cg-settings.xml
-Dcheckstyle.skip=true"
  }
  archiveArtifacts artifacts: '**/*.pdf', excludes: '**/test*.pdf', allowEmptyArchive: true
}
)
}
stage('Integration Tests'){
  //integration tests have to be after documentation for the tracker zip to include the user manual
  wrap([$class: 'Xvnc', takeScreenshot: false, useXauthority: true]) {
    withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
      try{
        //we do not install, since these suspicious jars could be misused by other projects
        sh "mvn verify failsafe:verify -Dcg.ut.skip=true -Dcheckstyle.skip=true"
      } finally {
        //Maven does not auto report JUnit failsafe results
        junit '**/target/failsafe-reports/*.xml'
      }
    }
  }
}
stage('Quality Gate') {
  withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
    sh "mvn sonar:sonar -Dsonar.login=0d1356516289799b179c6c7f851c9d4464ab04e2
-Dsonar.host.url=http://sonarqube:9000/sonarqube"
  }
  sh '$WORKSPACE/src/scripts/sonarStatus.sh'
}
stage('Assembly') {
  withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
    sh "mvn install -DskipTests -Dcheckstyle.skip=true"
    sh "mvn dependency:purge-local-repository"
  }
  archiveArtifacts artifacts: '**/target/*.zip'
}
} catch (any) {
  step([
    $class: 'Mailer', notifyEveryUnstableBuild: true,
    recipients: emailxtreipients([[ $class: 'CulpritsRecipientProvider'],
    [ $class: 'RequesterRecipientProvider']]])
  ])
  currentBuild.result = 'FAILURE'
}
} //timeout
logstashSend failBuild: false, maxLines: 1000
} //node

@NonCPS
def version(text) {
  def matcher = text =~ '<version>(.)</version>'
  matcher ? matcher[0][1] : null
}

```

## 1.8.2. The Deploy pipeline

This will be the main pipeline with everything from checkout to deployment. This pipeline is a “pipeline as code”.

Go to Jenkins home page :

<https://bpmfactory.s2-eu.nvx.com/jenkins>

- Click **New Item**
- Choose a name : CG-WM\_P2\_Deploy
- Choose **Pipeline** type

## General

- Check **Supprimer les anciens builds**
  - Strategy = Log Rotation
  - Nombre de builds à conserver = 10

## Build Triggers

- Choose « Scrutation de l'outil de gestion de version »
- Planning = H \* \* \* \*

## Advanced Project Options

None.

## Pipeline

Definition = Pipeline script from SCM

SCM = Git

- Repositories
  - Repository URL = <http://bpmfactory.s2-eu.nvx.com/gerrit/p/cg-wm.git>
  - Credentials = svc-fr-bpmfact / Bpm-fact0ry
- Branches to build : \*/master

Script Path = Jenkinsfile-2-deploy-to-dev

☒ Lightweight checkout

*Pipeline content (for information)*

```
#!/groovy
node {
    timeout(60) {
        try {
            stage('Checkout') {
                cleanWs() // requires workspace cleanup plugin to be installed
                retry(3) {
                    checkout scm
                }
                def v = version(readFile('pom.xml'))
                echo "Building version ${v}"
            }
            stage('Compilation') {
                //slaves are wiped out randomly, so we prepare them on each execution
                sh '$WORKSPACE/src/scripts/slavePrep.sh'
                withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
                    //used to deploy libs to local maven repository
                }
            }
        }
    }
}
```

```

sh "mvn clean"
//The assembly is postponed : it needs some further generated PDF
sh "mvn install -DskipTests -Dassembly.skipAssembly=true"
}
}
stage('Unit Tests') {
  wrap([$class: 'Xvnc', takeScreenshot: false, useXauthority: true]) {
    withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
      sh "mvn test -Dcheckstyle.skip=true"
      //Maven auto reports JUnit surefire results
    }
  }
}
stage('Documentation') {
  //get history from git to asciidoc documentation
  sh '$WORKSPACE/src/scripts/asciidocHistory.sh $WORKSPACE'
  withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
    //validate produces the date for PDF
    //javadoc:aggregate is CPU intensive, we don't parallelize for now
    sh "mvn validate asciidoctor:process-asciidoc antrun:run@pdfsAddVersion javadoc:aggregate
-Dcheckstyle.skip=true"
    sh "mvn javadoc:jar -pl cg-utils -Dcheckstyle.skip=true"
  }
  step([$class: 'JavadocArchiver', javadocDir: 'target/site/javadoc', keepAll: true])
  archiveArtifacts artifacts: '**/*.pdf,**/*-javadoc.jar', excludes: '**/test*.pdf'
}
stage('Integration Tests') {
  wrap([$class: 'Xvnc', takeScreenshot: false, useXauthority: true]) {
    withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
      try {
        sh "mvn verify failsafe:verify -Dcg.ut.skip=true -Dcheckstyle.skip=true"
      } finally {
        //Maven does not auto report JUnit failsafe results
        junit '**/target/failsafe-reports/*.xml'
      }
    }
  }
}
stage('Quality Check') {
  withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
    sh "mvn sonar:sonar -Dsonar.login=0d1356516289799b179c6c7f851c9d4464ab04e2
-Dsonar.host.url=http://sonarqube:9000/sonarqube"
  }
  sh '$WORKSPACE/src/scripts/sonarStatus.sh'
}
stage('Assembly') {
  withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: false)]) {
    sh "mvn install -DskipTests -Dcheckstyle.skip=true"
  }
  //archiveArtifacts is now in "Deployment" phase since we download packages
}
stage('Publication'){
  parallel (
    "Deployment to Nexus and IS": {
      withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
        //sh 'mvn wagon:update-maven-3'
        sh "mvn deploy -DskipTests -Dassembly.skipAssembly=true -Dcheckstyle.skip=true -s cg-settings.xml"
      }
      sh "ssh devops@frpardge.corp.nvx.com 'cd /opt/sagis/profiles/IS_default/bin;./restart.sh'"
      sh '$WORKSPACE/src/scripts/deployJavadoc.sh'
      sh '$WORKSPACE/src/scripts/getPackages.sh'

      //SchemaSpy must not fail the deployment so we put it after deployment
      sh '$WORKSPACE/src/scripts/schemaspy.sh'
    }
    publishHTML([
      allowMissing      : false,
      alwaysLinkToLastBuild: false,
      keepAll           : true,
      reportDir          : 'target/schemaspy',
      reportFiles        : 'index.html',
    ])
  )
}

```

```

        reportName      : 'DB Schema'])

    archiveArtifacts artifacts: '**/target/*.zip'
},
"Reporting" : {

    //Git Inspector
    sh 'mkdir target/gitinspector'
    sh 'export PYTHONIOENCODING=utf-8 ; gitinspector --format=html -rTw > target/gitinspector/index.html'
    publishHTML([
        allowMissing      : false,
        alwaysLinkToLastBuild: false,
        keepAll           : true,
        reportDir          : 'target/gitinspector',
        reportFiles        : 'index.html',
        reportName         : 'Git Inspector'])

    //Maven Site
    withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
        sh 'mvn site site:stage -DskipTests -Dcheckstyle.skip=true -s cg-settings.xml'
    }
    publishHTML([
        allowMissing      : false,
        alwaysLinkToLastBuild: false,
        keepAll           : true,
        reportDir          : 'target/staging',
        reportFiles        : 'index.html',
        reportName         : 'Maven Reporting'])
    }
    )
}
} catch (any) {
    step([
        $class: 'Mailer', notifyEveryUnstableBuild: true,
        recipients: emailxrecipients([[ $class: 'CulpritsRecipientProvider'],
        [ $class: 'RequesterRecipientProvider']])
    ])
    currentBuild.result = 'FAILURE'
}
} //timeout
logstashSend failBuild: false, maxLines: 1000
} //node

@NonCPS
def version(text) {
    def matcher = text =~ '<version>(.)</version>'
    matcher ? matcher[0][1] : null
}

```

### 1.8.3. The Deploy Int pipeline



Describe this pipeline when stable

### 1.8.4. The Release pipeline

This is the release pipeline launched manually at will when an external release is needed. This pipeline is a "pipeline as code".

Go to Jenkins home page :

<https://bpmfactory.s2-eu.nvx.com/jenkins>

- Click **New Item**

- Choose a name : CG-WM\_P3\_Release
- Choose **Pipeline** type

## General

- Check **Ce build a des paramètres**
  - Paramètre texte
    - RELEASE\_VERSION
    - the release version, with pattern 1.YY.MM[increment] (ex : 1.17.5.9)
- Check **Supprimer les anciens builds**
  - Strategy = Log Rotation
  - Nombre de builds à conserver = 10

## Build Triggers

No trigger (manual launch).

## Advanced Project Options

None.

## Pipeline

Definition = Pipeline script from SCM

SCM = Git

- Repositories
  - Repository URL = <http://bpmfactory.s2-eu.nvx.com/gerrit/p/cg-wm.git>
  - Credentials = svc-fr-bpmfact / Bpm-fact0ry
- Branches to build : \*/master

Script Path = Jenkinsfile-4-release

☒ Lightweight checkout

*Pipeline content (for information)*

```
#!/groovy

//Release is a manual firing (and should always be)
//No need to do the whole process, trunk is always trustworthy with our setup
//Just check that the merge pipeline (DeployToDev) is successful

node {
    timeout(30) {
        try {
            stage('Checkout') {
                cleanWs() // requires workspace cleanup plugin to be installed
                retry(3) {
                    checkout scm
                }
            }
            echo "Releasing version $RELEASE_VERSION"
        }
    }
}
```

```

}
stage('Documentation') {
    //get history from git to asciidoc documentation
    sh '$WORKSPACE/src/scripts/asciidocHistory.sh $WORKSPACE'

    withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
        //to put jars in local maven repository if needed
        sh "mvn clean"
        sh "mvn versions:set -DnewVersion=$RELEASE_VERSION"
        //without this local installation, modules are searched on internet on mvn validate
        sh "mvn install -DskipTests -Dassembly.skipAssembly=true"
        //we launch some (quick) tests that contains the generation of service list for the cg-utils doc
        sh "mvn test -pl cg-utils"
        //time to launch the actual doc generation
        //validate produces the date for PDF
        sh "mvn validate asciidoctor:process-asciidoc antrun:run@pdfsAddVersion javadoc:aggregate
-Dcheckstyle.skip=true"
        sh "mvn javadoc:jar -pl cg-utils -Dcheckstyle.skip=true"
    }

    step([$class: 'JavadocArchiver', javadocDir: 'target/site/javadoc', keepAll: true])
    archiveArtifacts artifacts: '**/*.pdf,**/*-javadoc.jar', excludes: '**/test*.pdf'
}
stage('Deployment') {
    //Deployment is after documentation because a pdf must be in the zip

    //Delete tag if this is a replayed-on-error build...

    //...locally
    sh "git tag -d cg-wm-$RELEASE_VERSION || true"

    //...remotely
    //Special characters have to be URL encoded : https://stackoverflow.com/questions/6172719/escape-character-in-
git-proxy-password
    sh "git push --force --delete https://svc-fr-cric:****@cric.pl.s2-eu.nvx.com/gerrit/p/cg-wm.git cg-wm-
$RELEASE_VERSION || true"

    withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
        sh "mvn deploy scm:tag -s cg-settings.xml -DskipTests -Dcheckstyle.skip=true"
    }
    sh "$WORKSPACE/src/scripts/deployJavadoc.sh"
    sh "$WORKSPACE/src/scripts/getPackages.sh"
    archiveArtifacts artifacts: '**/target/*.zip'
}
stage('Reporting') {
    withMaven(maven: 'Maven 3.5', mavenOpts: '-Xmx1024M', options: [artifactsPublisher(disabled: true)]) {
        sh "mvn site site:stage -DskipTests -Dcheckstyle.skip=true -s cg-settings.xml"
    }
    publishHTML([
        allowMissing      : false,
        alwaysLinkToLastBuild: false,
        keepAll           : true,
        reportDir          : 'target/staging',
        reportFiles         : 'index.html',
        reportName         : 'Maven Reporting'])
}
} catch (any) {
    step([
        $class: 'Mailer', notifyEveryUnstableBuild: true,
        recipients: emailExtrecipients([$class: 'CulpritsRecipientProvider'],
[$class: 'RequesterRecipientProvider'])
    ])
    currentBuild.result = 'FAILURE'
}
} //timeout
logstashSend failBuild: false, maxLines: 1000
} //node

@NonCPS
def version(text) {

```



```
def matcher = text =~ '<version>(.)</version>'  
matcher ? matcher[0][1] : null  
}
```

## 1.9. Troobleshooting

### 1.9.1. Disk space usage > 90 %

If the disk space usage is too high and your build fails at the start for this reason, you can purge some folders with the below actions.

- Edit the **console** job.
- Put these lines and save :

```
du --max-depth=1 /home/jenkins/workspace/ | sort -n -r | head -n 30  
find /home/jenkins/workspace/ -maxdepth 1 -mtime +90 -type d -depth -print
```

- Launch the job
- Following the results, do the necessary deletions
- If there are some ws-cleanup directory, you can delete them safely :

```
rm -rf /home/jenkins/workspace/*ws-cleanup*/ ???
```

## 2. Appendix

### 2.1. Revision marks

*Differences since last tag*

