DevOps
CI / CD
Jenkins

Jamel ESSOUSSI: Architecte logiciel

jamel.essoussi@gmail.com

https://github.com/jessoussi

2025

Plan du cours

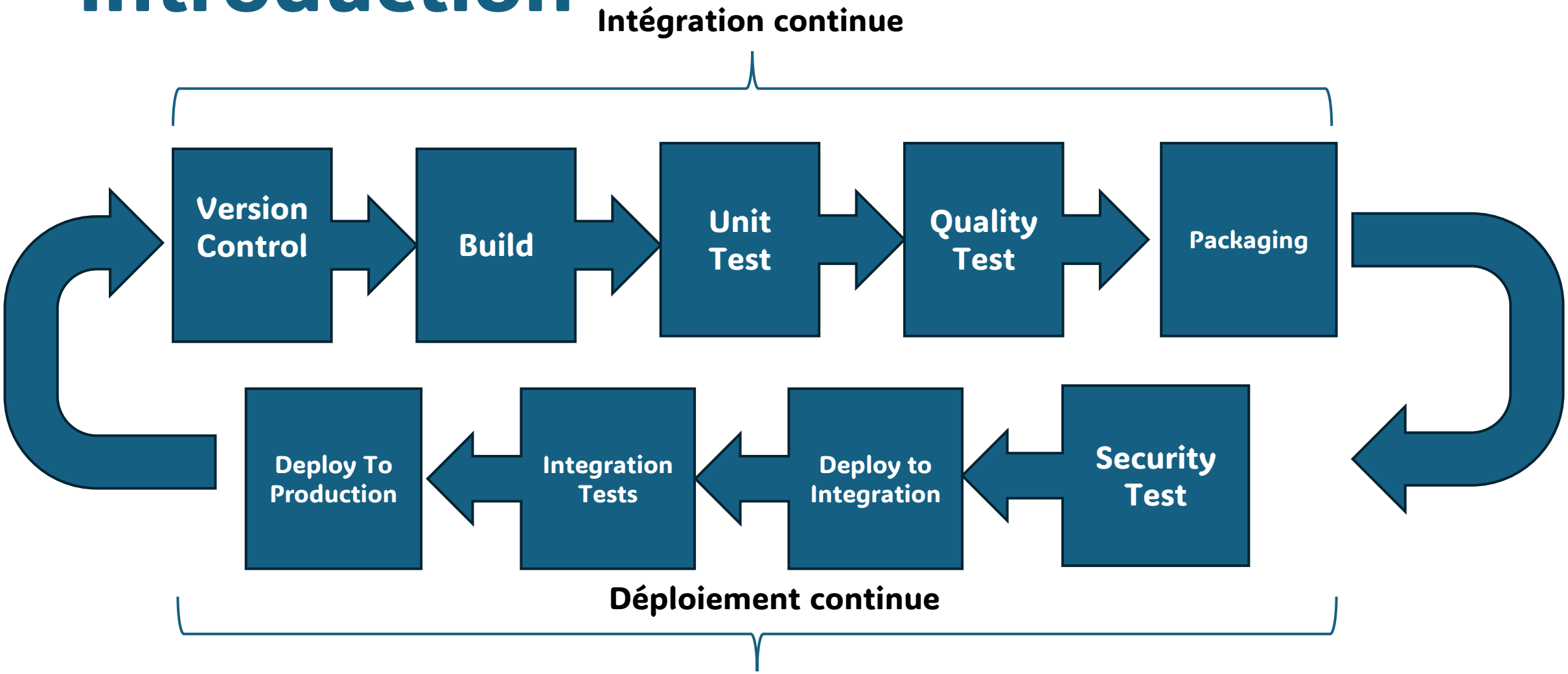
I. Introduction

II. Job & Pipeline

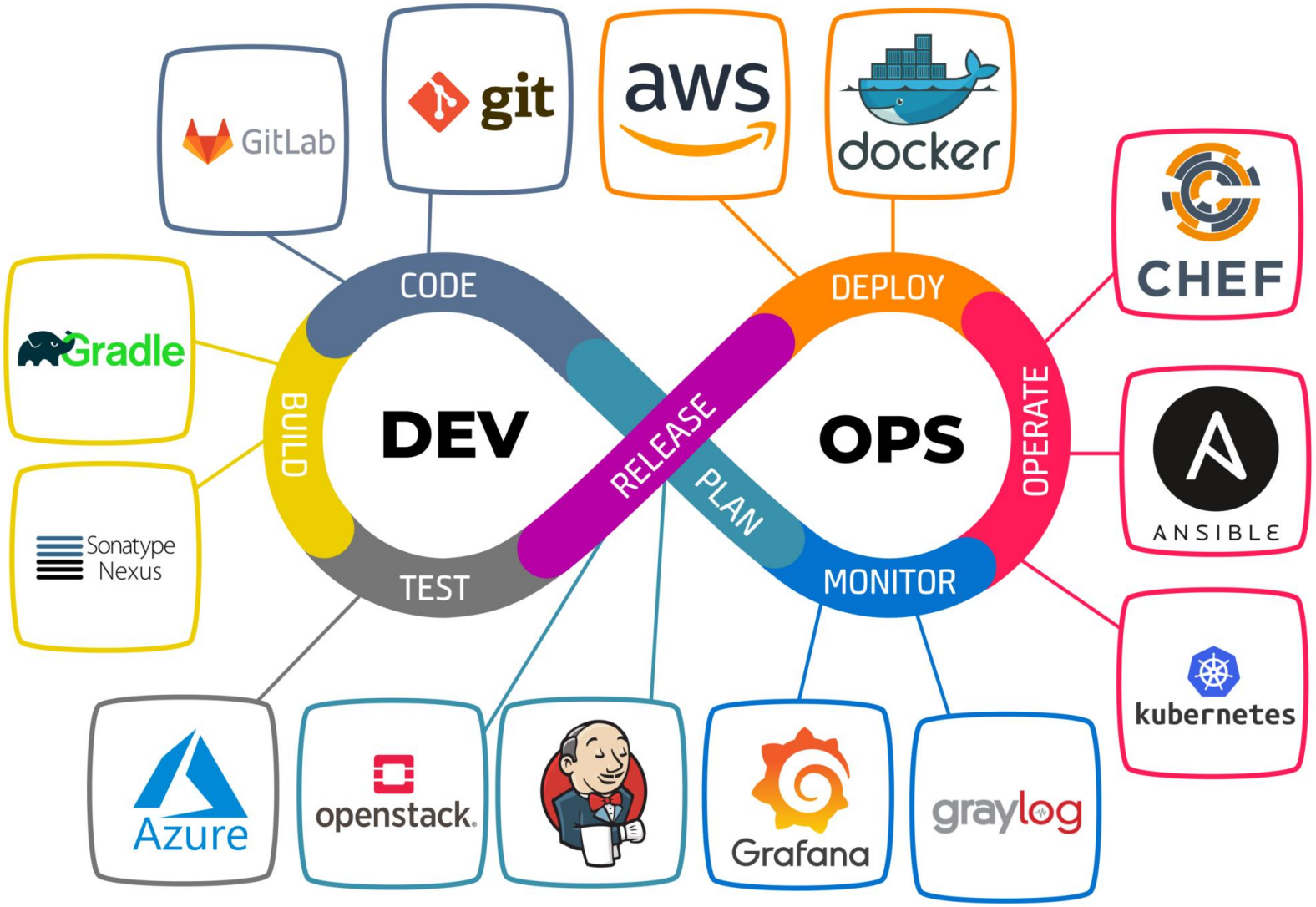
III. Intégration avec les autres outils

IV. Les bonnes pratiques

Introduction



Outils du DevOps



Introduction

- Jenkins est un serveur d'automatisation open source qui aide à automatiser la construction, le test et le déploiement d'applications logicielles.
- Jenkins est un outil central pour la chaine DevOps

Introduction

- Il est écrit en Java et a été initialement publié en 2011 en tant que clone du projet Hudson.
- Jenkins est largement utilisé pour les pratiques d'intégration continue (CI) et de livraison continue (CD) dans le développement de logiciels.

Introduction

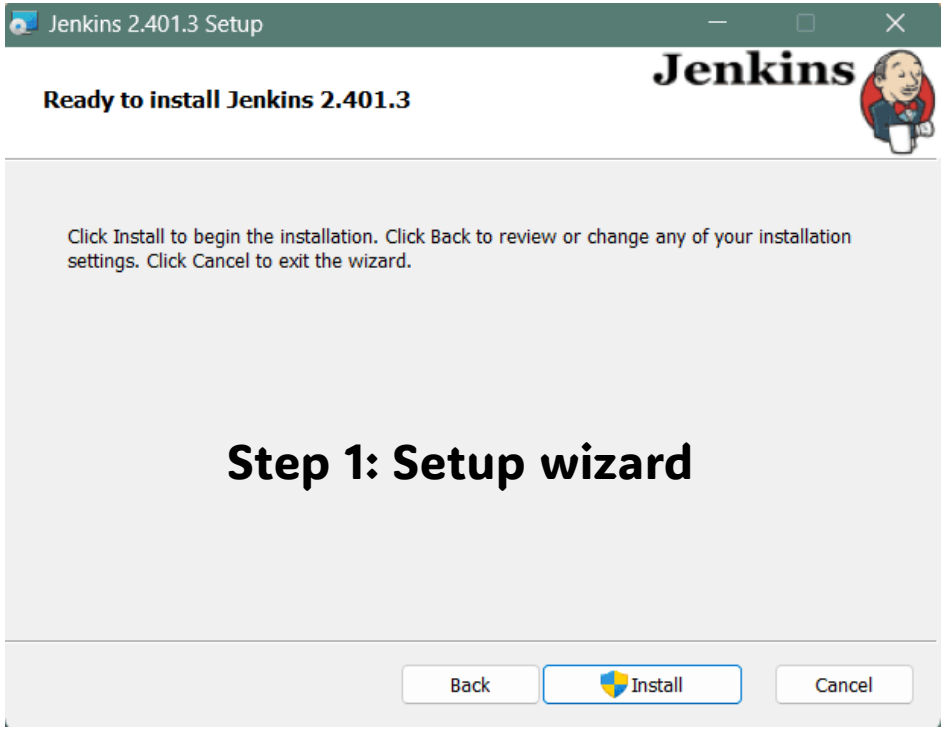
- En s'intégrant à divers outils de construction, systèmes de contrôle de version et cadres de test, Jenkins permet aux développeurs d'automatiser des tâches répétitives, telles que la construction de l'application, l'exécution de tests et le déploiement en production.

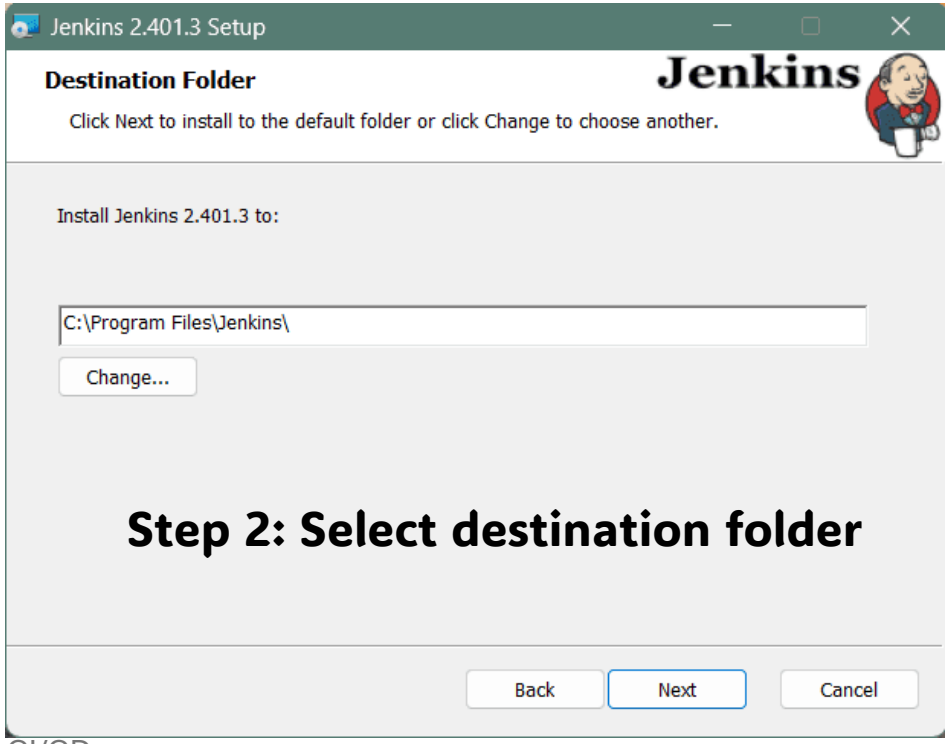
Introduction

- Cette automatisation permet d'augmenter la productivité, d'assurer la qualité du code et de permettre une livraison plus rapide des mises à jour logicielles.

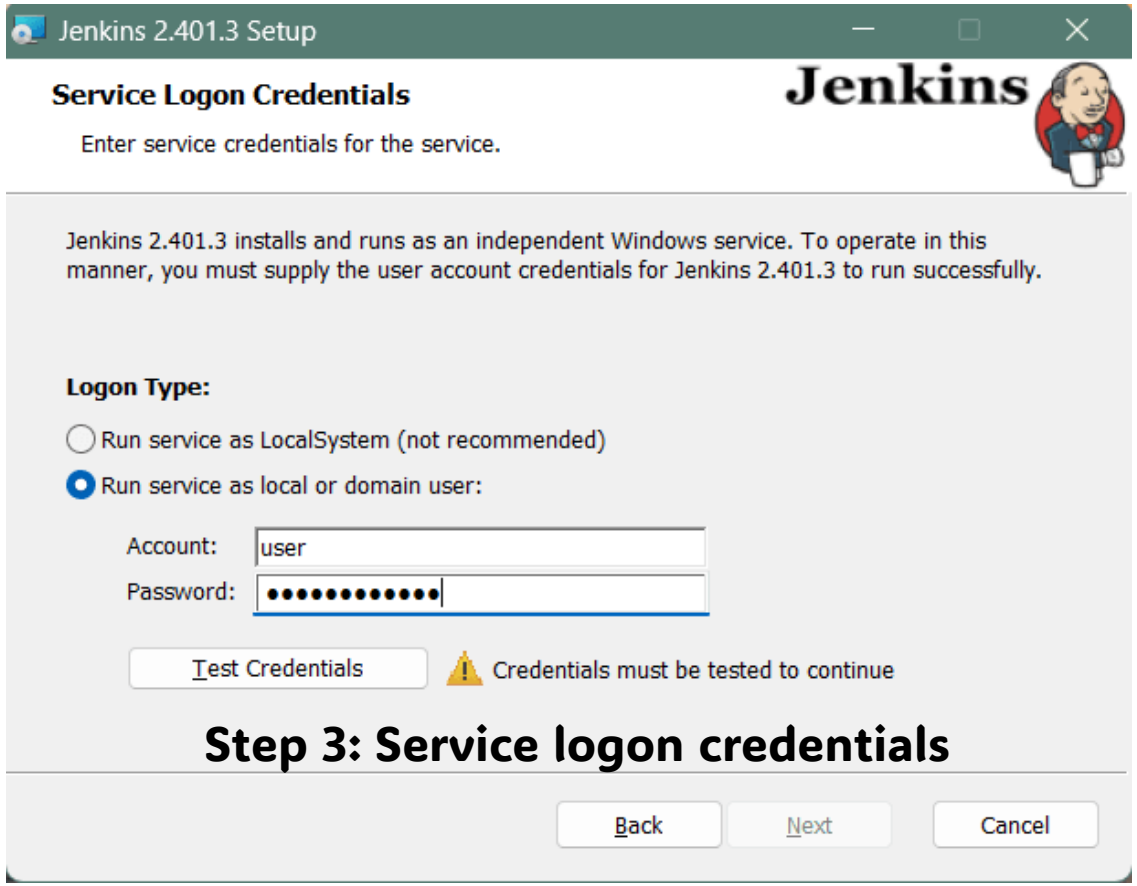
Installation Jenkins

- Téléchargement
  - https://www.jenkins.io/download/#downloading-jenkins

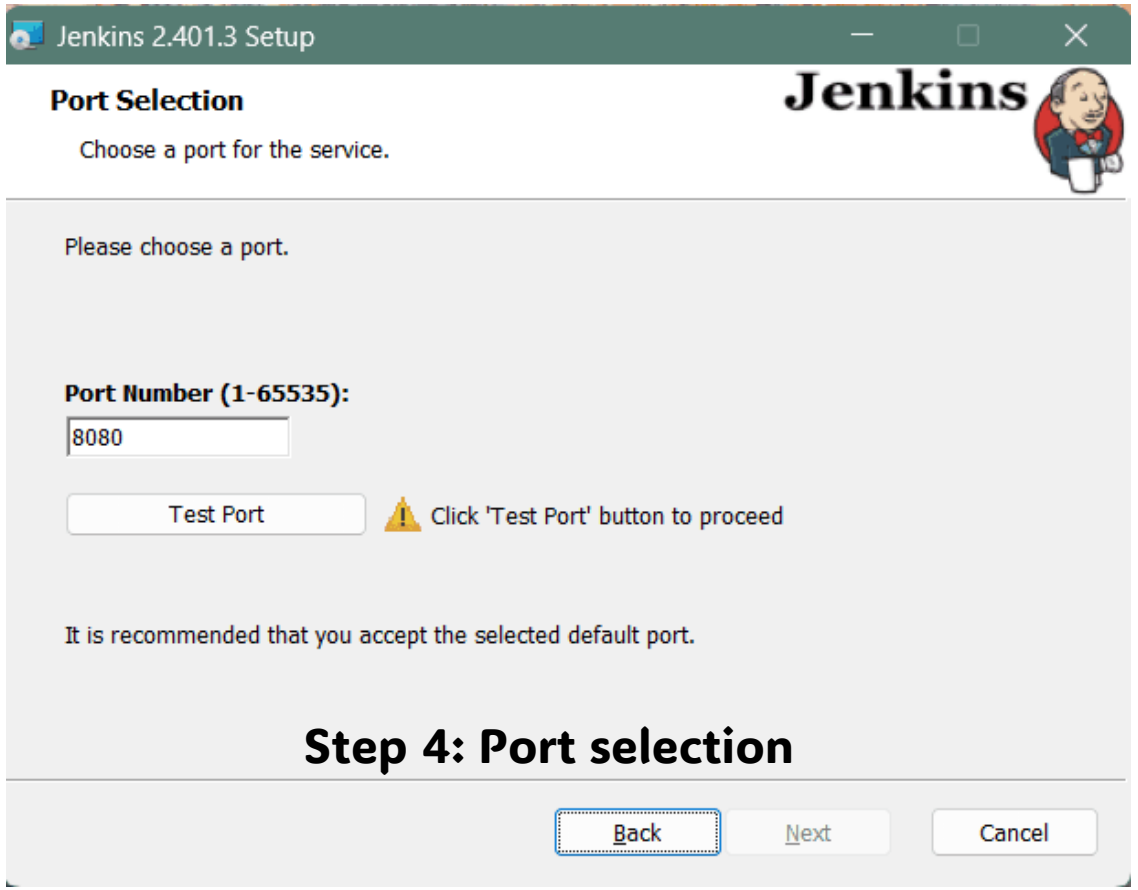




Installation Jenkins

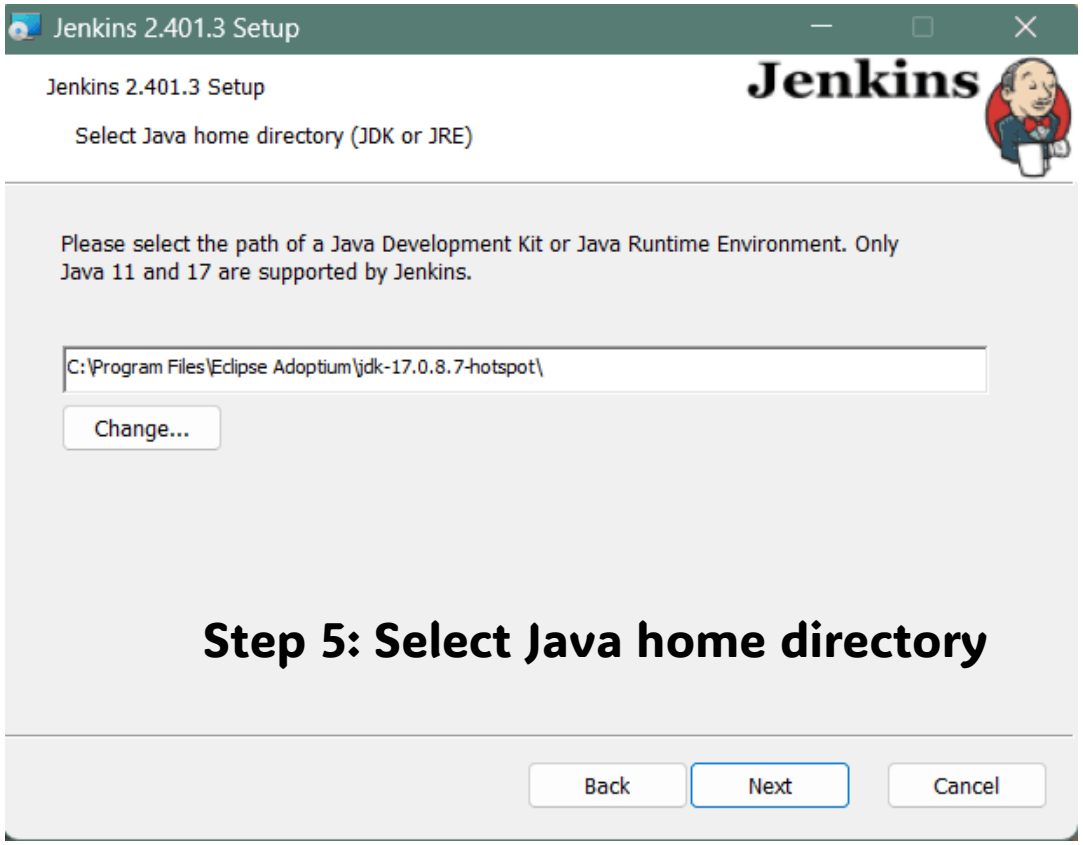


Step 3: Service logon credentials

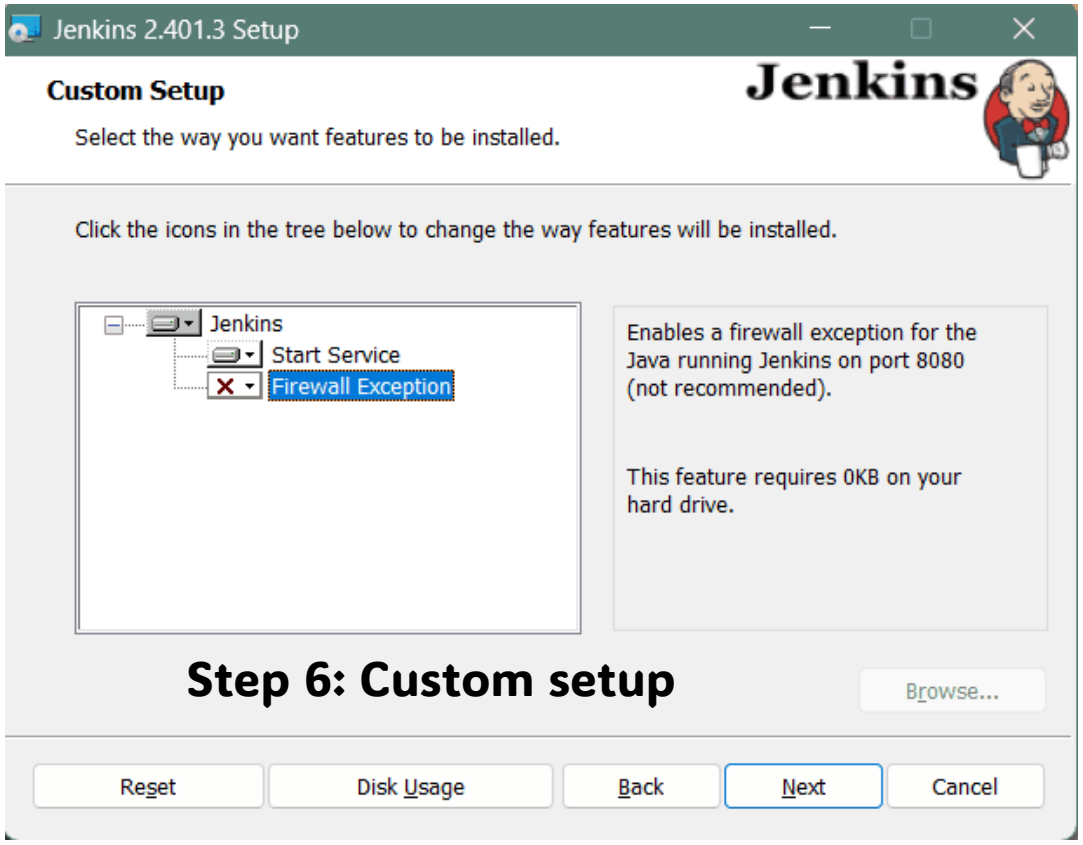


Step 4: Port selection

Installation Jenkins



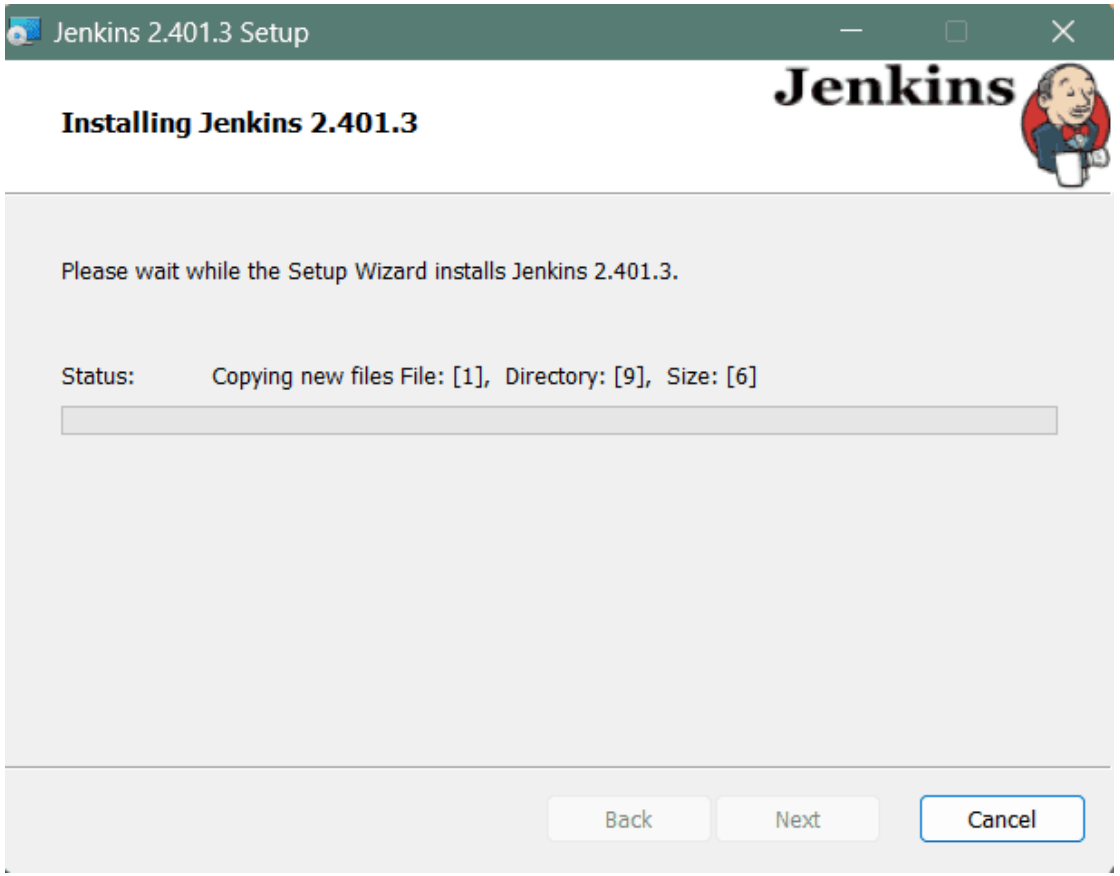
Step 5: Select Java home directory



Step 6: Custom setup

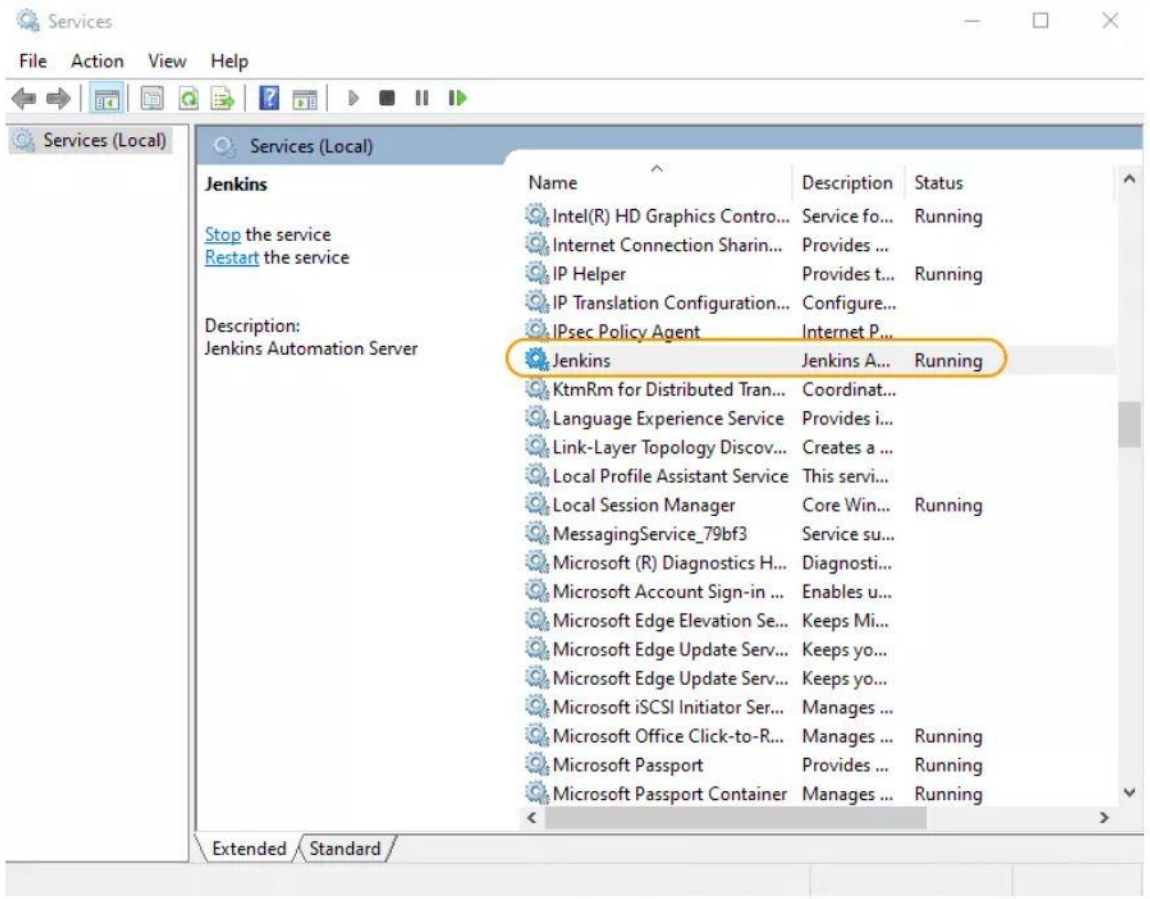
Installation Jenkins

Step 7: Install Jenkins

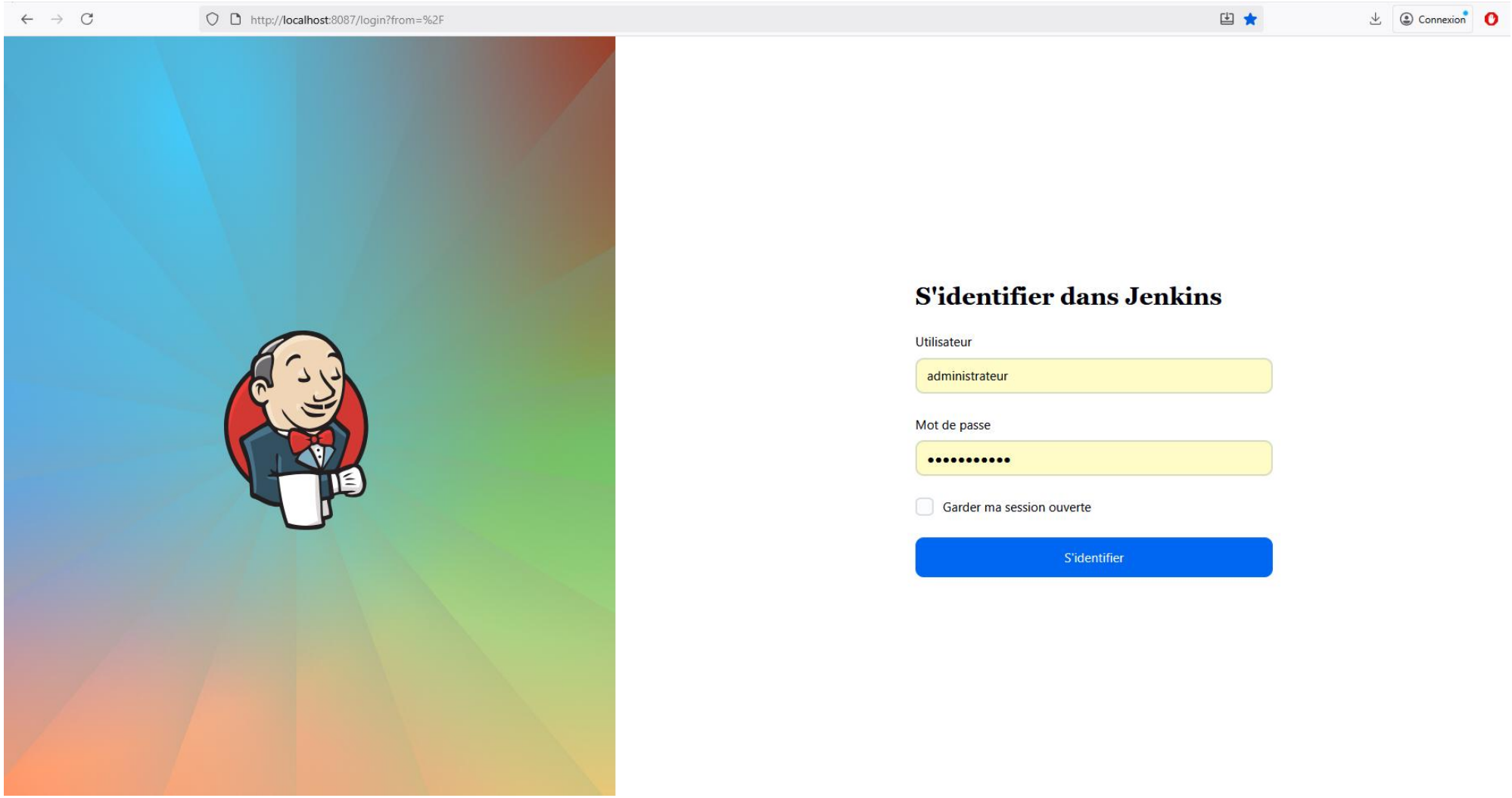


Installation Jenkins

Step 8: Finish Jenkins installation



Installation Jenkins



Plan du cours

I. Introduction

II. Job & Pipeline

III. Intégration avec les autres outils

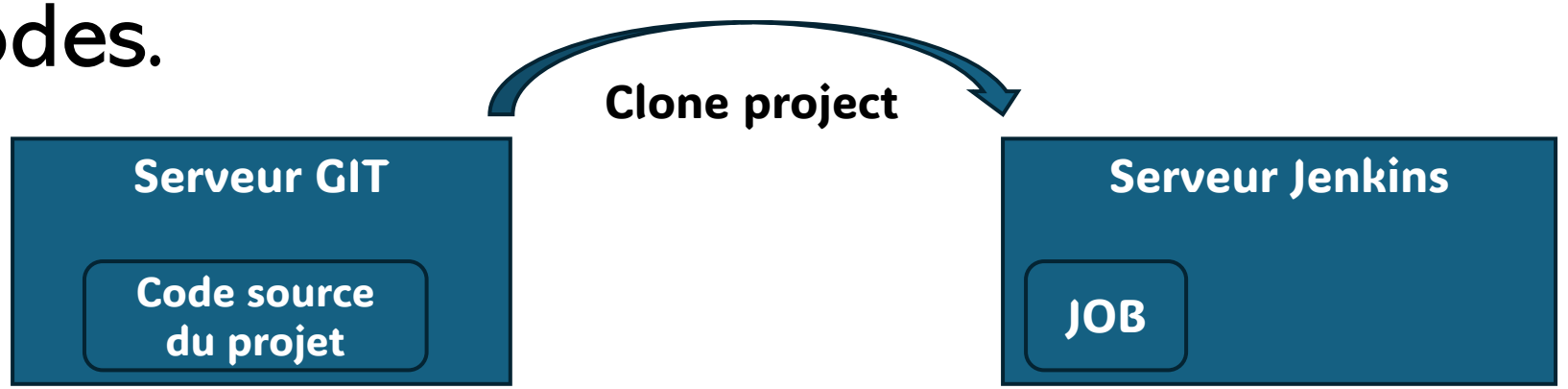
IV. Les bonnes pratiques

Job Jenkins

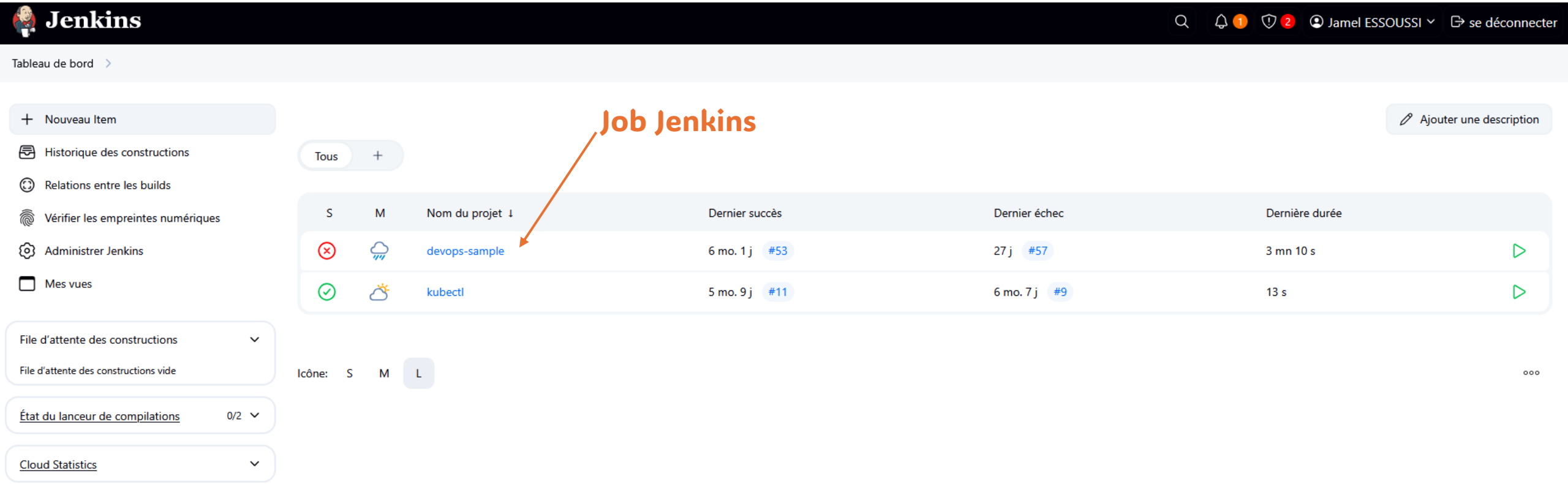
- Un Job dans Jenkins est une tâche qui peut être configurée et exécutée sur un serveur Jenkins, en fonction des besoins.

Job

- Avant de créer un Job sur Jenkins, il est essentiel d'avoir un serveur Jenkins installé et configuré, un projet prêt à être compilé, testé ou déployé, et un dépôt Git pour stocker les codes.



Job

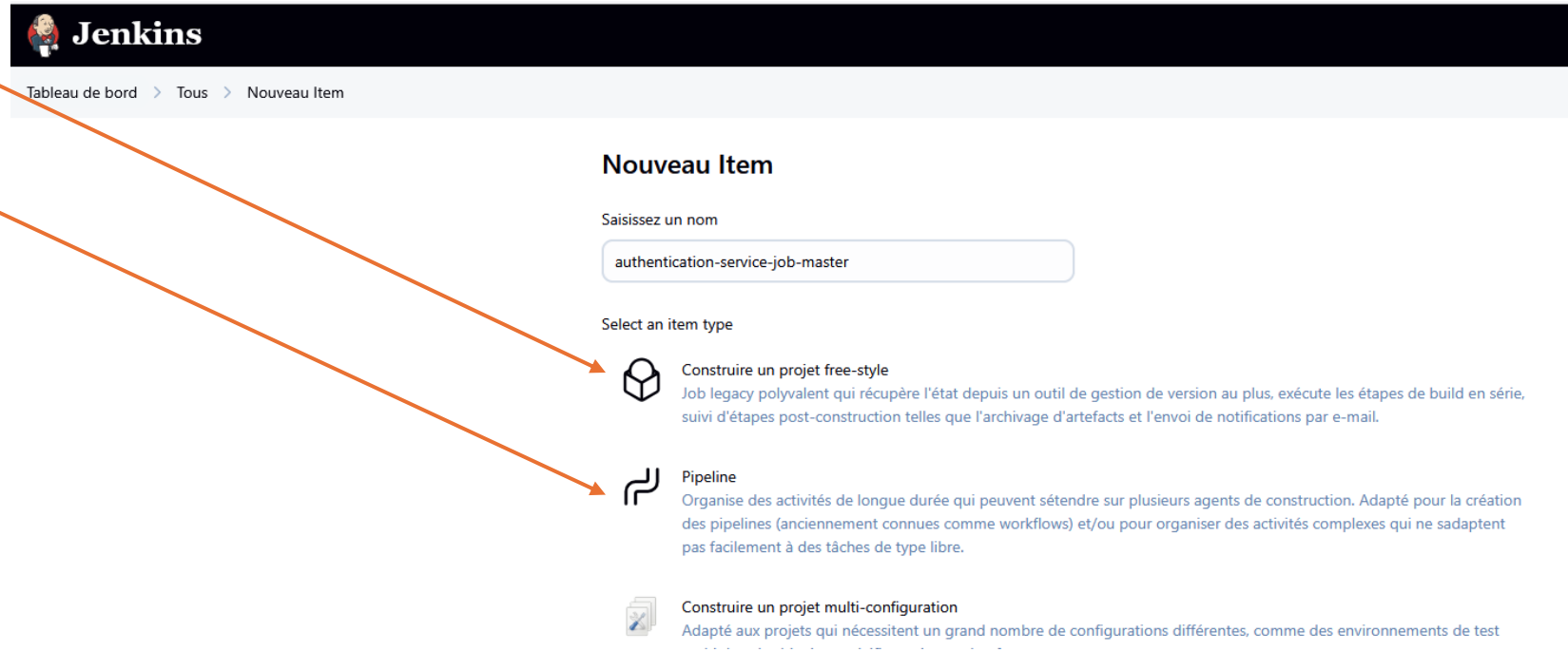

Jenkins
Jamel ESSOUSSI
se déconnecter
Tableau de bord
Nouveau Item
Historique des constructions
Relations entre les builds
Vérifier les empreintes numériques
Administrer Jenkins
Mes vues
Job Jenkins
Ajouter une description
Tous
S
M
Nom du projet
Dernier succès
Dernier échec
Dernière durée
devops-sample
6 mo. 1 j #53
27 j #57
3 mn 10 s
kubectl
5 mo. 9 j #11
6 mo. 7 j #9
13 s
File d'attente des constructions
File d'attente des constructions vide
Icône: S M L
État du lanceur de compilations 0/2
Cloud Statistics

Job


Jenkins
Jamel ESSOUSSI
se déconnecter
Tableau de bord
Nouveau Item
Historique des constructions
Relations entre les builds
Vérifier les empreintes numériques
Administrer Jenkins
Mes vues
Ajout d'un nouveau Job
Ajouter une description
Tous
S
M
Nom du projet
Dernier succès
Dernier échec
Dernière durée
devops-sample
6 mo. 1 j #53
27 j #57
3 mn 10 s
kubectl
5 mo. 9 j #11
6 mo. 7 j #9
13 s
File d'attente des constructions
File d'attente des constructions vide
Icône: S M L
État du lanceur de compilations 0/2
Cloud Statistics

Job

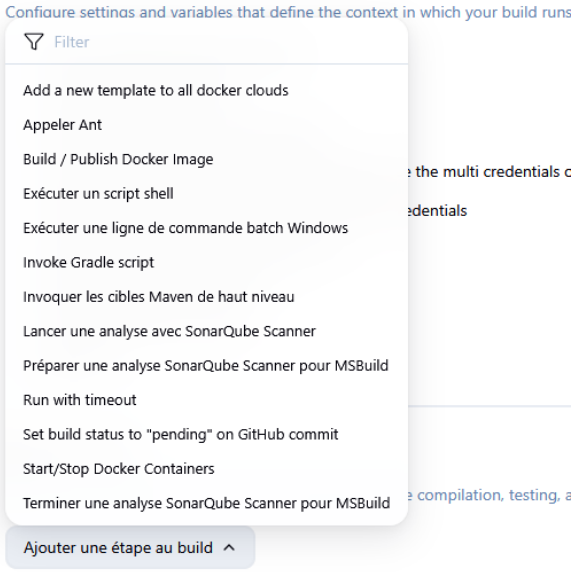
- Plusieurs types de Job peuvent être créer:
  - Freeestyle
  - Pipeline
  - …

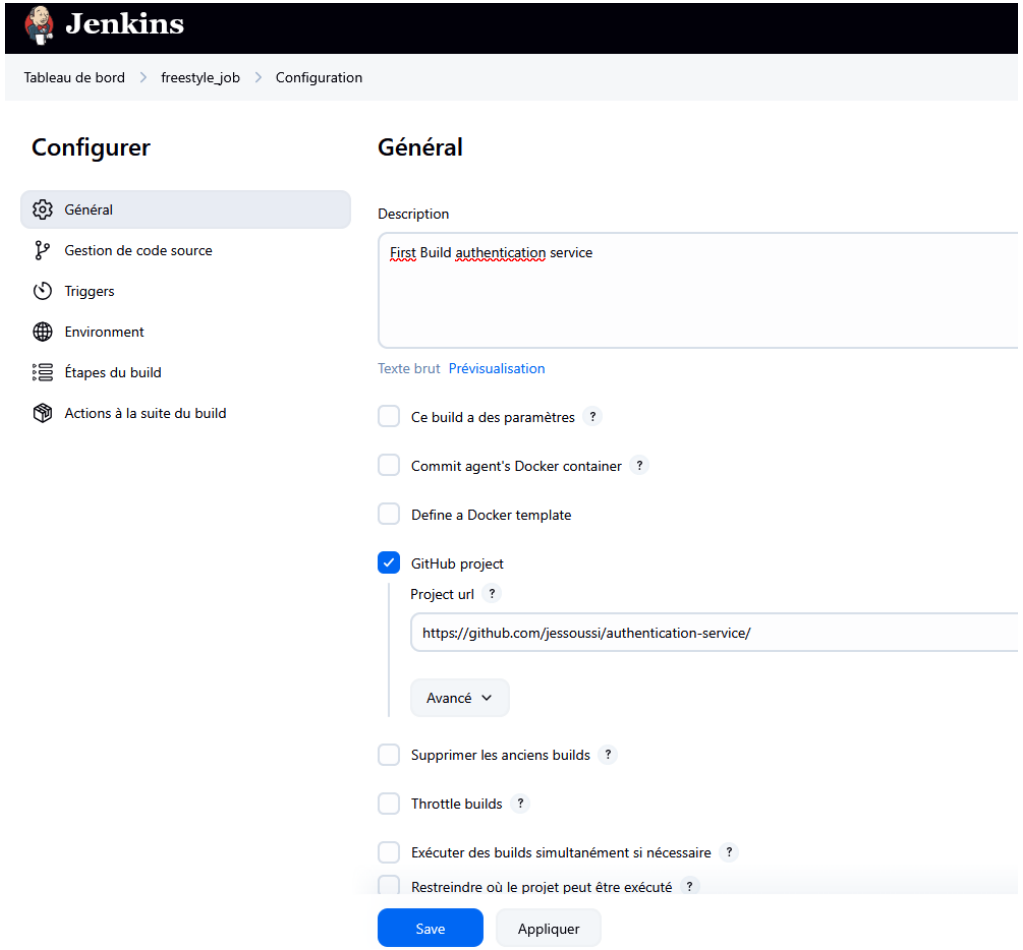


Job

- Freeestyle
  - Configuration manuelle du job

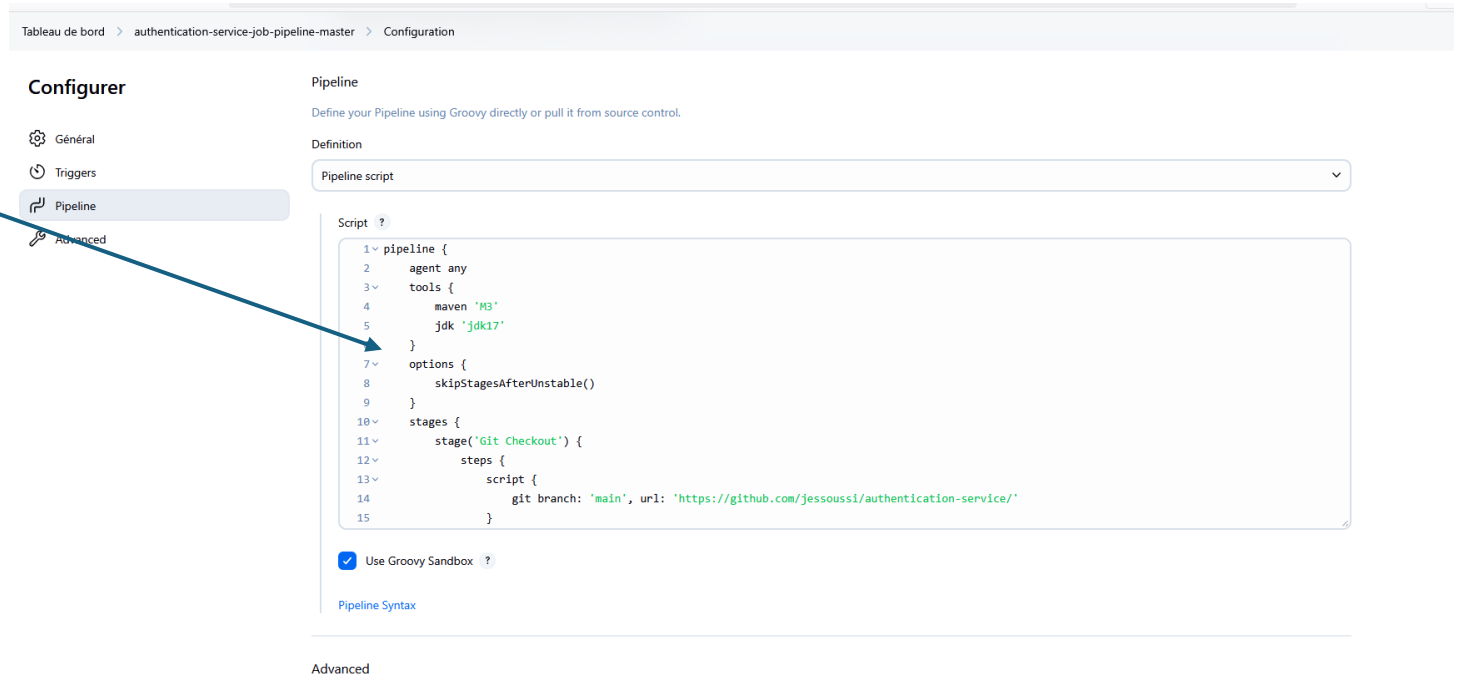
GIT, étapes, Script, …





Job

- Pipeline
  - Déclarative à travers des pipelines
  - Jenkinsfile



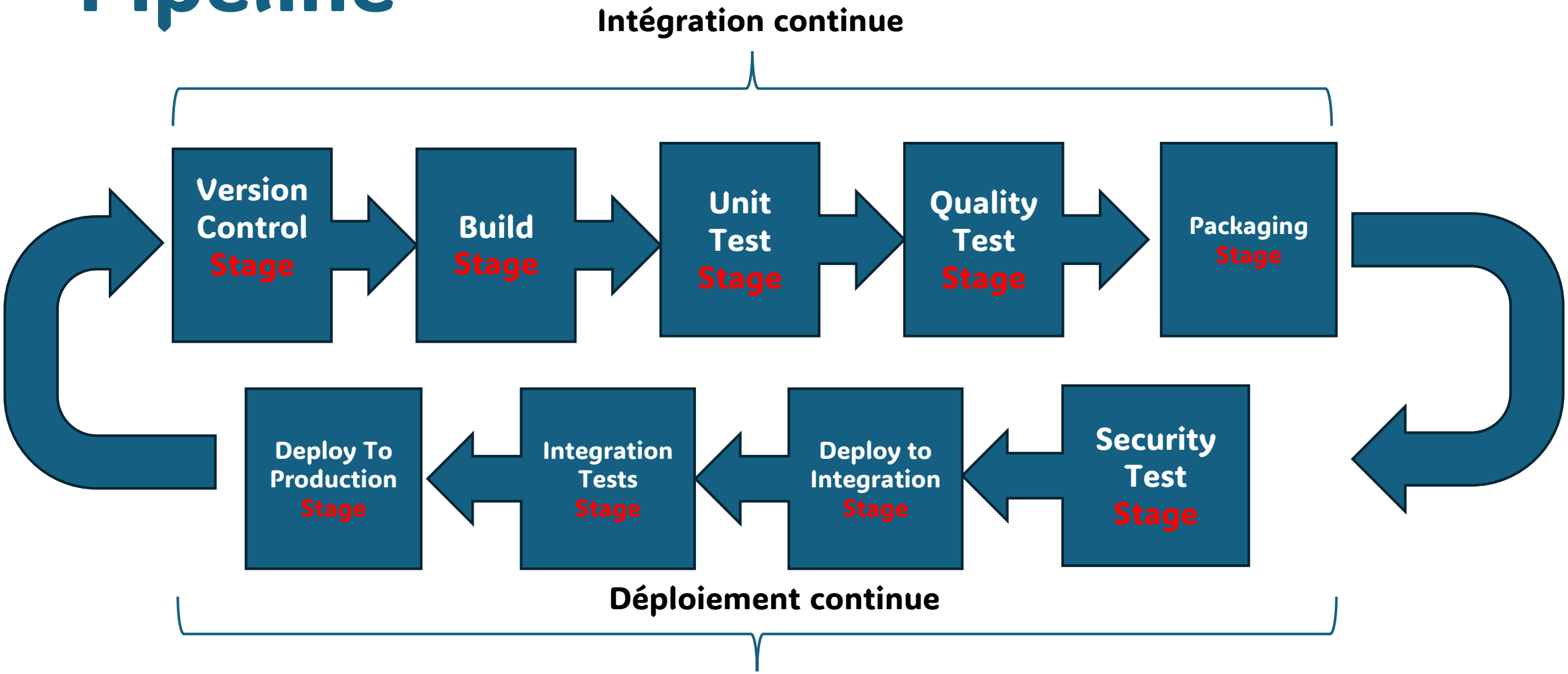
Pipeline

- Dans Jenkins, on peut définir des pipelines pour décrire les étapes d'intégration et de déploiement continue.

- Les pipelines reposent sur le langage de script Groovy. La documentation dédiée à Jenkins est disponible sur cette page https://www.jenkins.io/doc/book/pipeline/ .

Pipeline

Pipeline Jenkins



Pipeline Jenkinsfile

- Syntaxe (https://www.jenkins.io/doc/book/pipeline/syntax/)

```groovy
pipeline {
    /* insert Declarative Pipeline here */
}
```

```groovy
pipeline {
    agent any
    options {
        // Timeout counter starts AFTER agent is allocated
        timeout(time: 1, unit: 'SECONDS')
    }
    stages {
        stage('Example') {
            steps {
                echo 'Hello World'
            }
        }
    }
}
```

Pipeline Jenkinsfile

- Syntaxe (https://www.jenkins.io/doc/book/pipeline/syntax/)

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

Pipeline

- Les scripts Pipeline apportent les notions suivantes :
  - Pipeline: est l'ensemble du processus à exécuter ;
  - Node : représente un environnement pouvant exécuter un pipeline (une machine esclave) ;
  - Stage: représente un ensemble d'étapes du processus (par exemple, la récupération des sources, la compilation…) ;
  - Step: représente ce qu'il y a à faire à un moment donné (l'action à proprement parler, comme maven).

Les concepts principaux

- Un pipeline (workflow) est une procédure automatisée associée à votre dépôt.
- Un pipeline est composé de un ou plusieurs jobs
  - Les jobs peuvent s'exécuter en parallèle par défaut
  - Possibilité de définir des dépendances entre les jobs (Concept de stage)
- Un job est composé de steps (étapes)
  - Une étape est une action/commande

Les concepts principaux

- Un job est exécuté par un Agent
  - Un agent en charge d'exécuter un job sur un serveur
  - Jenkins fournit des agents (et les ressources pour les exécuter)
  - Les jobs d'un pipeline sont exécutés dans des contexte d'exécutions différents
  - Les jobs ne peuvent pas partager de données par défaut
- Des événements déclenchent le lancement d'un pipeline

Les concepts principaux

- Les jobs peuvent s'exécuter en parallèle.

```groovy
pipeline {
    agent any
    stages {
        stage('Parallel') {
            parallel {
                stage('Python') {
                    agent { docker 'python:latest' }
                    steps {
                        sh "python --version"
                    }
                }
                stage('Java') {
                    agent { docker 'openjdk:latest' }
                    when {
                        branch 'staging'
                    }
                    steps {
                        sh "java -version"
                    }
                }
            }
        }
    }
}
```

Les concepts principaux

- Utilisation de variables d'environnement

Jenkinsfile (Declarative Pipeline)

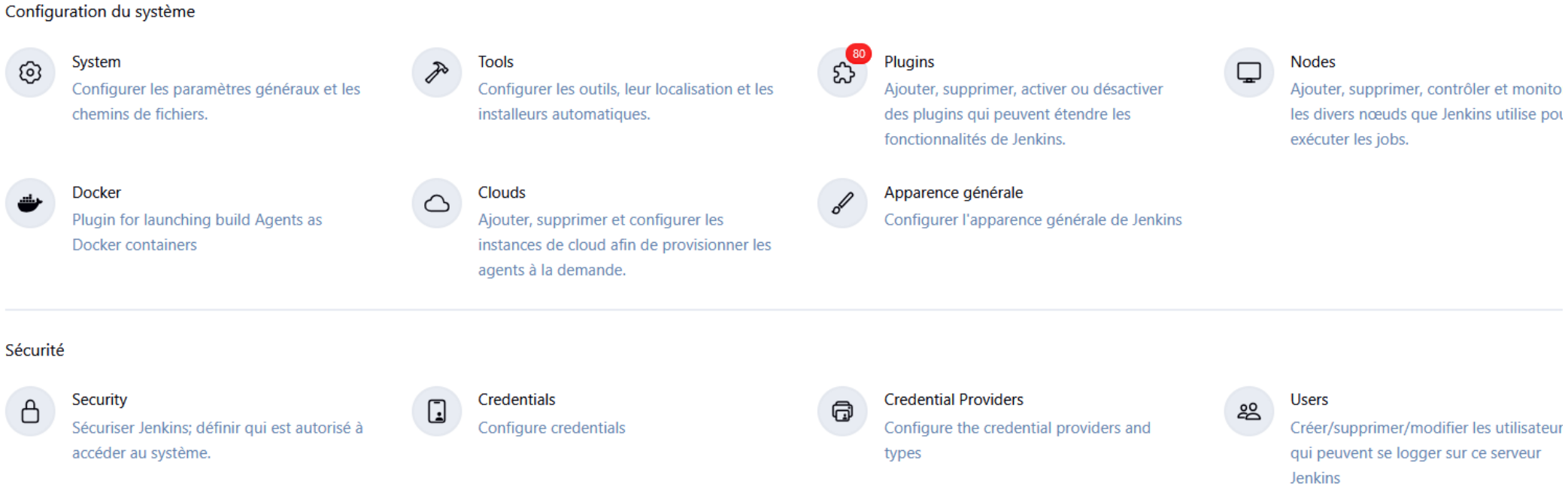
```
pipeline {
    agent {
        label '!windows'
    }

    environment {
        DISABLE_AUTH = 'true'
        DB_ENGINE    = 'sqlite'
    }

    stages {
        stage('Build') {
            steps {
                echo "Database engine is ${DB_ENGINE}"
                echo "DISABLE_AUTH is ${DISABLE_AUTH}"
                sh 'printenv'
            }
        }
    }
}
```

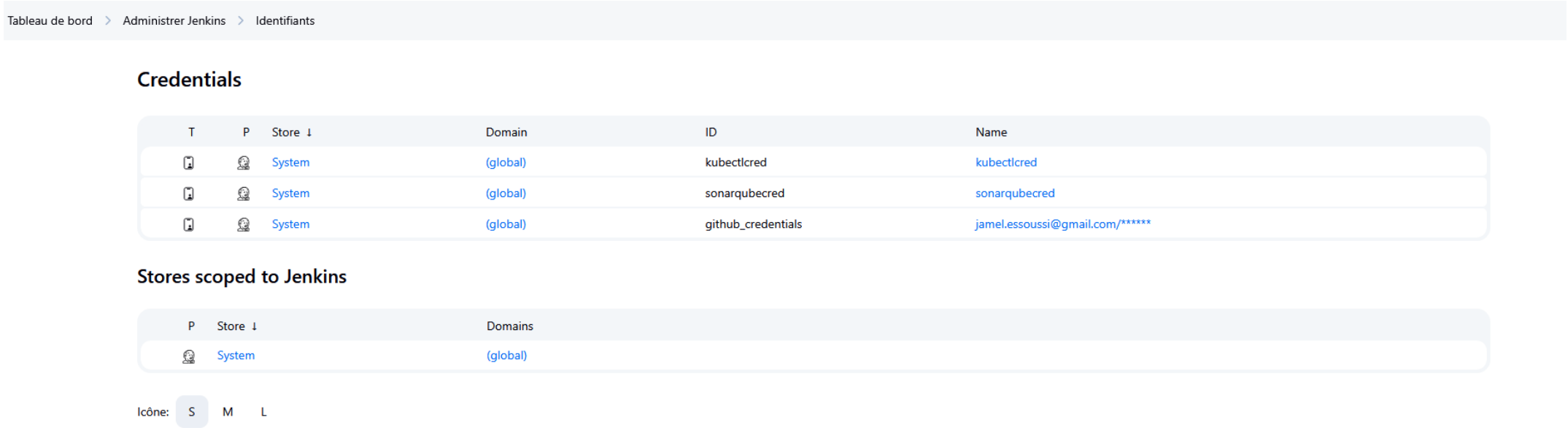
Les concepts principaux

- Utilisation des credentials
- Les credentials sont définit dans l'interface d'administration de Jenkins



Les concepts principaux

- Utilisation des credentials
- Liste des credentials



Les concepts principaux

- Utilisation des credentials

Jenkinsfile (Declarative Pipeline)

```
pipeline {
    agent {
        // Define agent details here
    }
    environment {
        AWS_ACCESS_KEY_ID     = credentials('jenkins-aws-secret-key-id')
        AWS_SECRET_ACCESS_KEY = credentials('jenkins-aws-secret-access-key')
    }
    stages {
        stage('Example stage 1') {
            steps {
                // ❶
            }
        }
        stage('Example stage 2') {
            steps {
                // ❷
            }
        }
    }
}
```

Pipeline

- Pour définir un pipeline, il faut avoir le plugin Pipeline installé sur le serveur Jenkins.
- Ce plugin est pré-installé avec le serveur Jenkins.
- Deux Types de pipeline sont possibles:
  - Pipeline Déclaratives,
  - Pipeline Scripté.

Pipeline

- Un pipeline peut être crée de l'une des manières suivantes:
  - Dans Jenkins: Gâce à l'interface utilisateur de Jenkins.
  - Dans SCM: écrire un Jenkins file manuellement qui sera versionner avec le code source du projet dans GIT.
- La syntaxe pour définir un pipeline avec l'une ou l'autre approche est la même, utilisation de la syntaxe Groovy

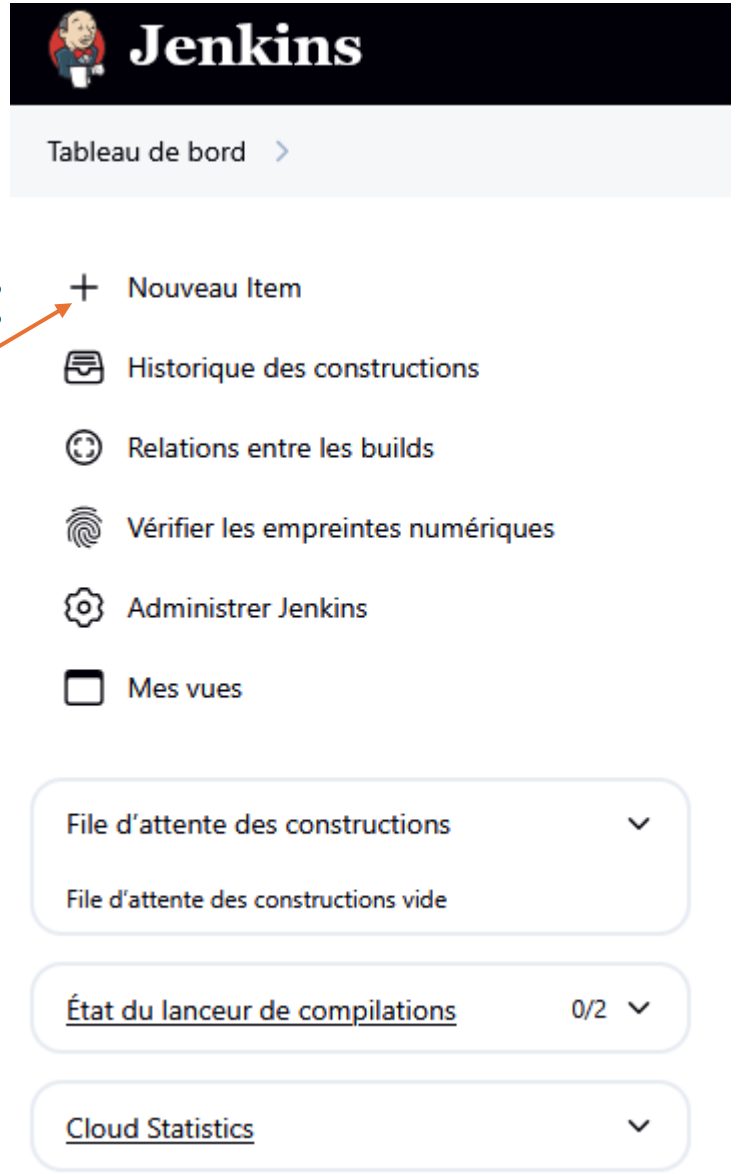
Pipeline

- Création à travers l'interface classique:
  - Un Jenkinsfile est créé à l'aide de l'interface utilisateur classique est stocké par Jenkins lui-même (à l'intérieur le répertoire d' accueil Jenkins).
  - Pour créer un pipeline de base à travers l'interface utilisateur classique de Jenkins:

Pipeline

- Création à travers l'interface classique:

1 - Dans le tableau de bord Jenkins, Selectionner nouvel élément:



Pipeline

- Création à travers l'interface classique:

2 – Renseigner le nom de votre nouveau pipeline.
3 – Sélectionner Pipeline
4 - Valider

- Jenkins utilise ce nom d'élément pour créer des répertoires sur disque.
- C'est recommandé d'éviter d'utiliser des espaces dans les noms d'articles

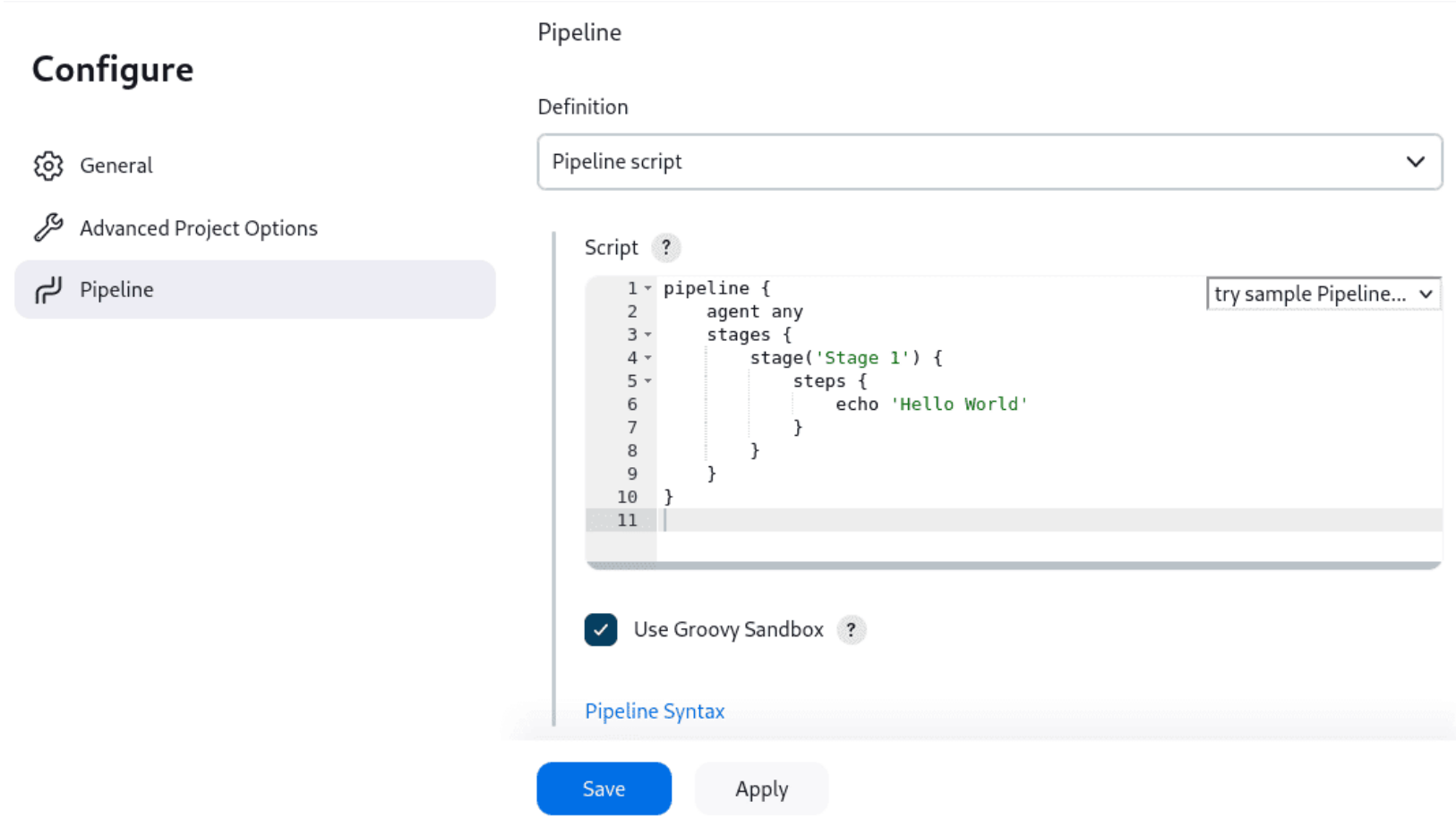
Pipeline

- Création à travers l'interface classique:

5- Cliquez sur Job créé.
6- Cliquez sur l'onglet Pipeline.
7- Sélectionnez le type du pipeline « Pipeline Script»
8- Entrez votre code Pipeline dans la zone de texte de script.

Pipeline

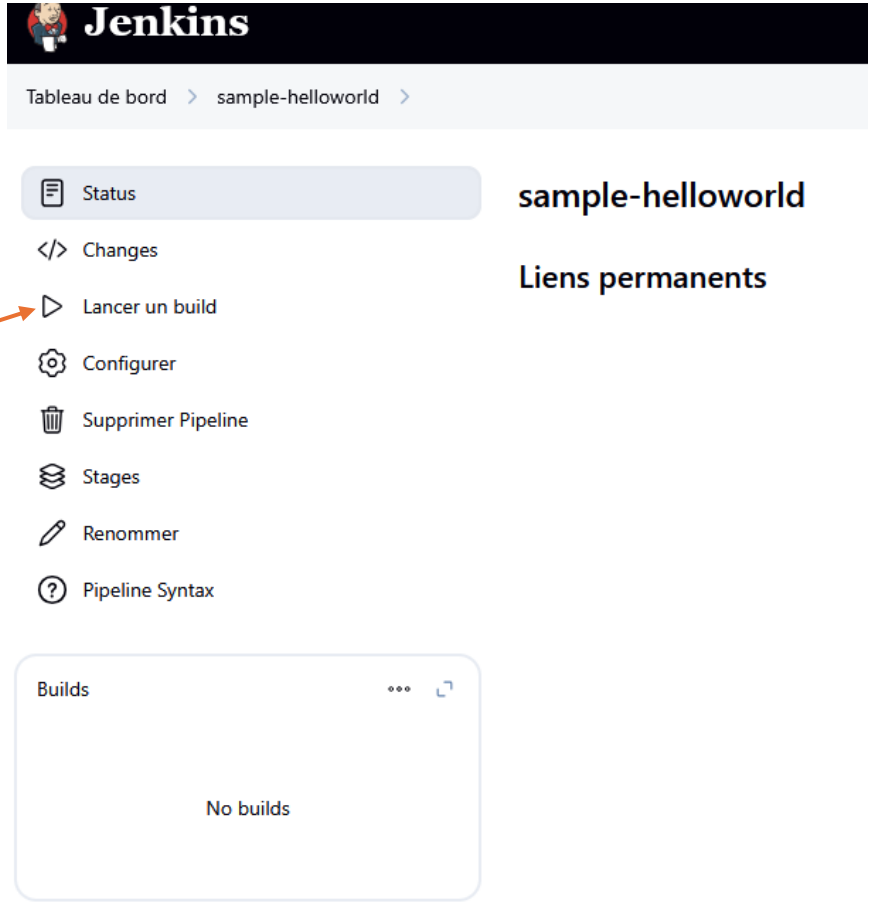
- Création à travers l'interface classique:



Pipeline

- Création à travers l'interface classique:

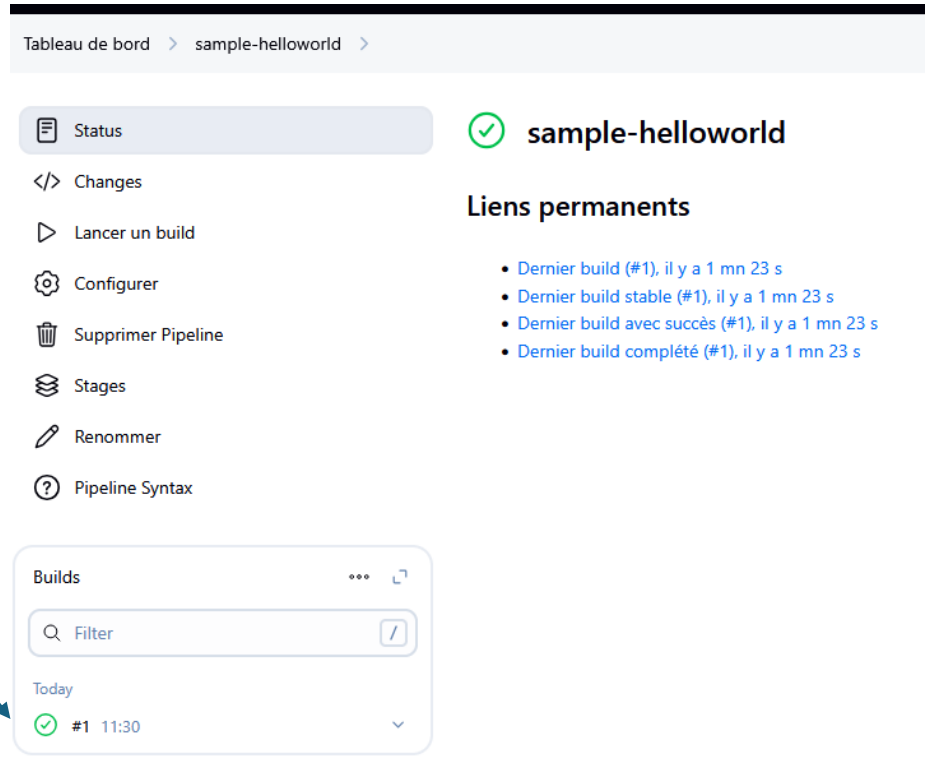
9 - Cliquez sur Enregistrer pour ouvrir la page de vue Projet/objet Pipeline.
10 - Sur cette page, cliquez sur Lancer un Build à gauche pour exécuter le pipeline.



Pipeline

- Création à travers l'interface classique:

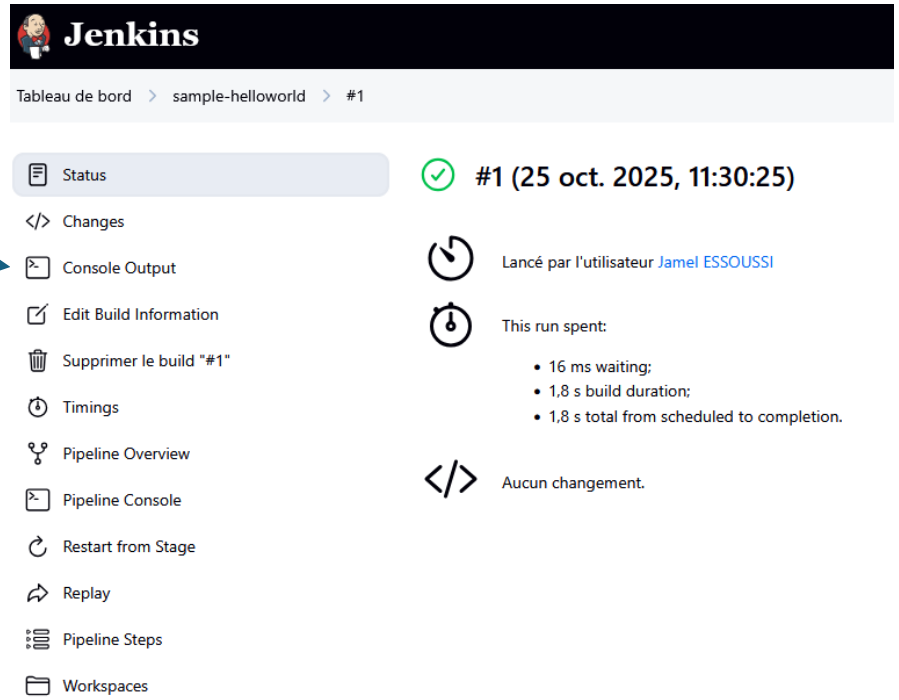
10 – Une fois le build lancer, cliquer sur le dernier numéro de build



Pipeline

- Création à travers l'interface classique:

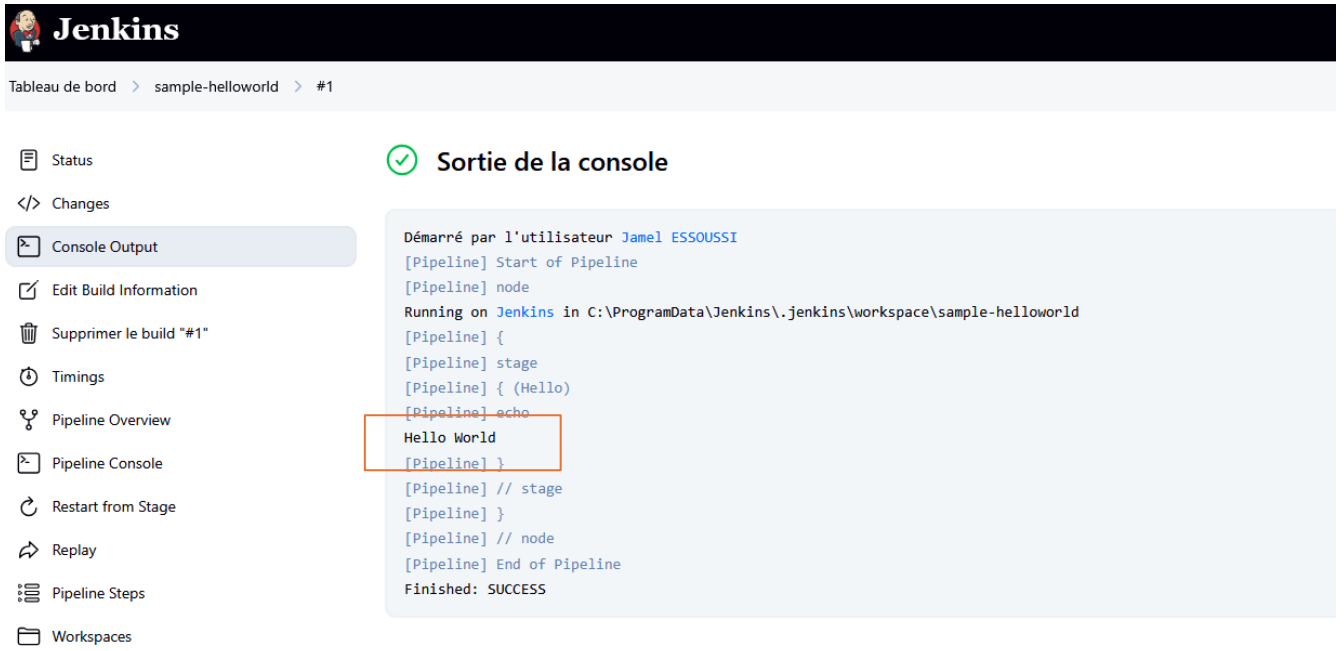
11 – Ensuite sur Console output pour voir l'historique



Pipeline

- Création à travers l'interface classique:

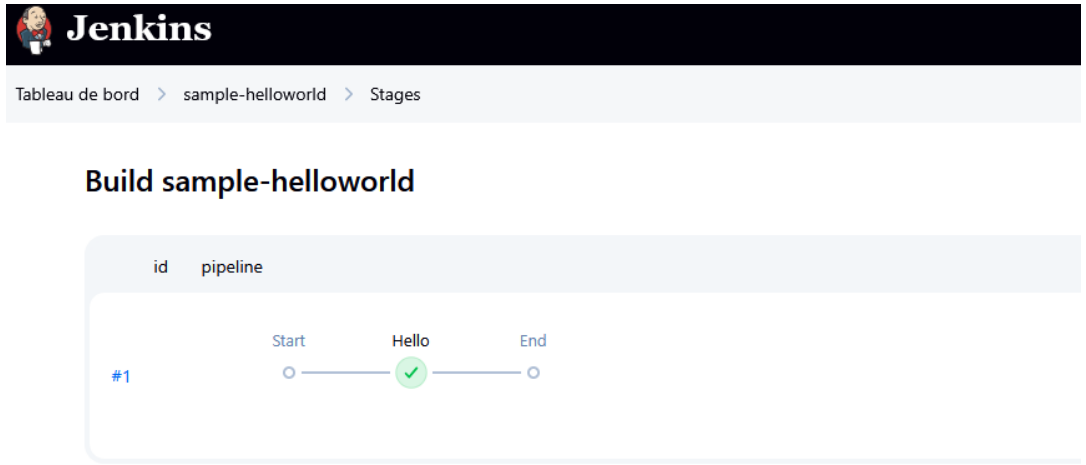
11 – Consulter l'historique

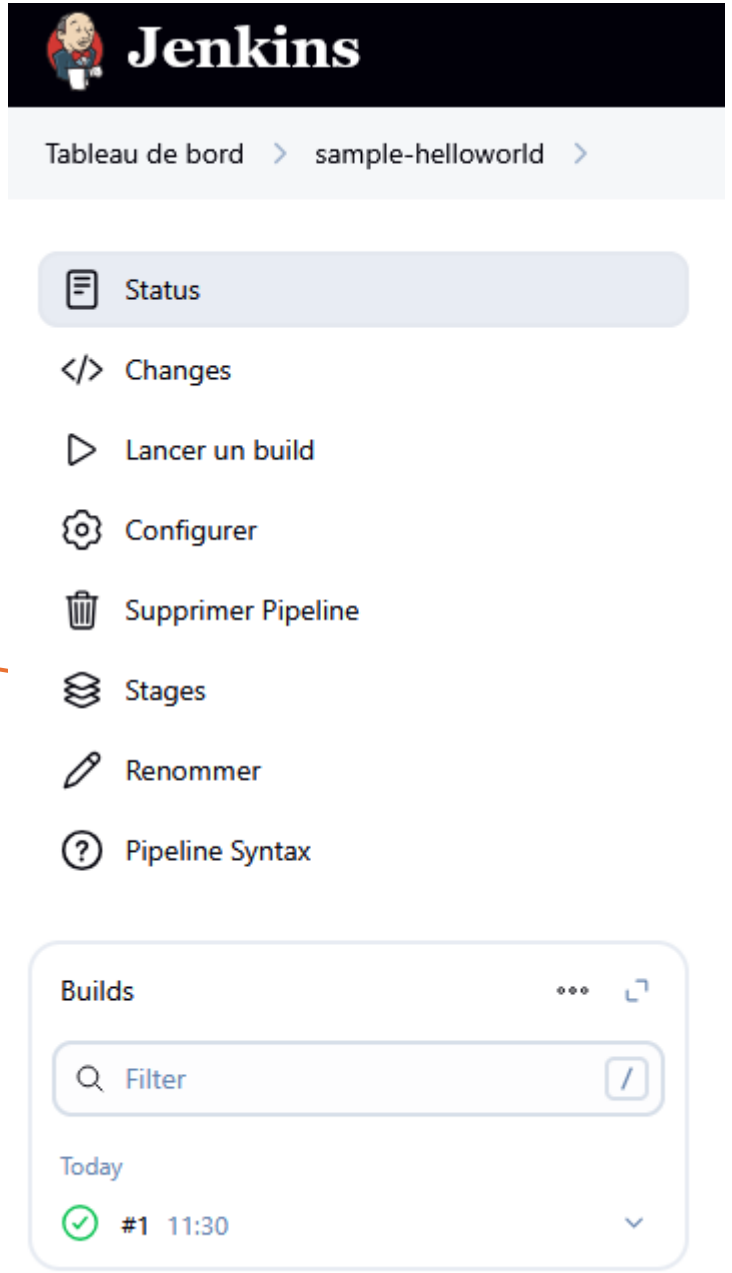


Pipeline

- Création à travers l'interface classique:

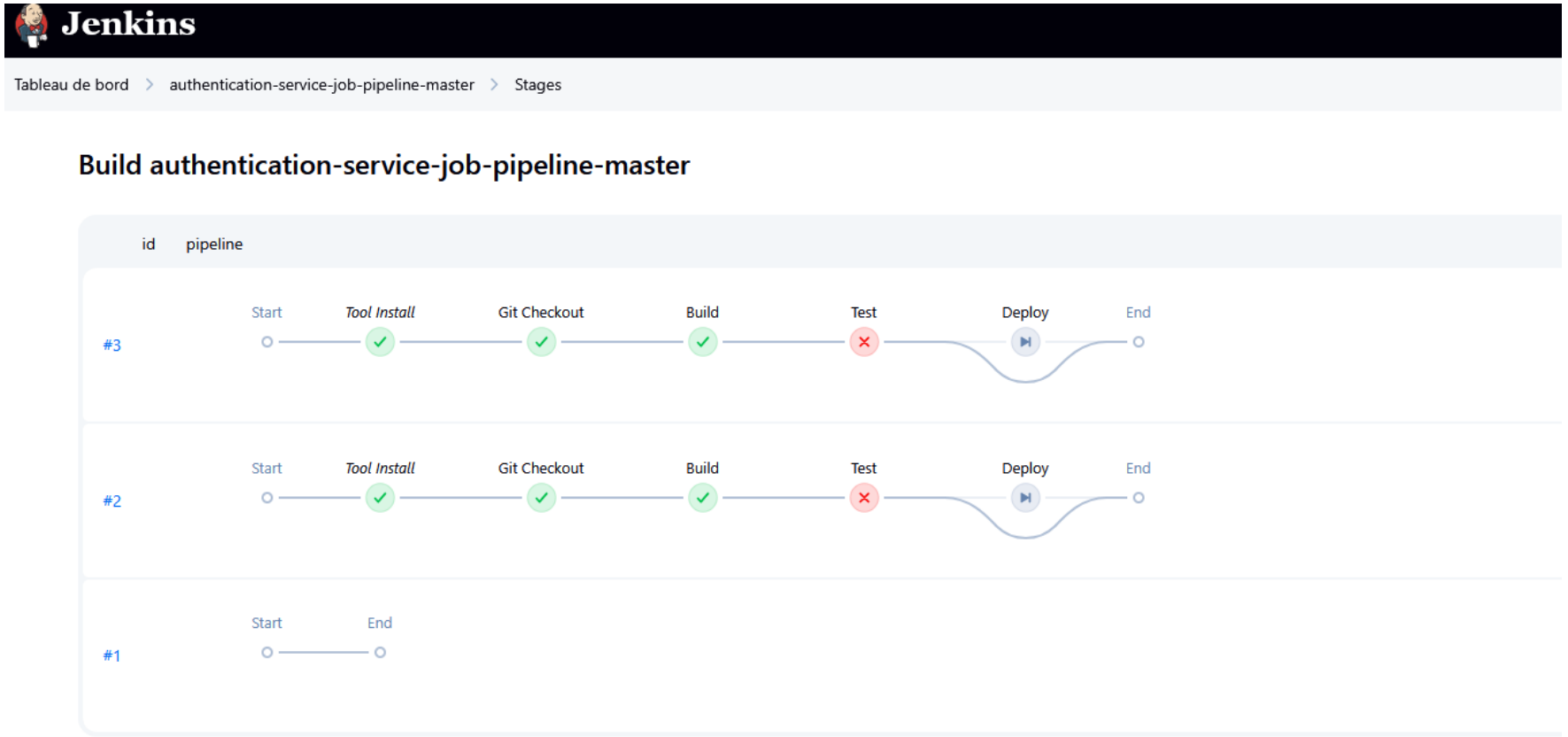
12 – Visualiser les stages





Pipeline

- Notre exemple de TD:



Pipeline

- Notre exemple de TD:

```
pipeline {
    agent any
    tools {
        maven 'M3'
        jdk 'jdk17'
    }
    options {
        skipStagesAfterUnstable()
    }
    stages {
        stage('Git Checkout') {
            steps {
                script {
                    git branch: 'main', url: 'https://github.com/jessoussi/authentication-service/'
                }
            }
        }
        stage ('Build') {
            steps {
                bat 'mvn clean compile'
            }
        }
        stage ('Scan SonarQube') {
            steps {
                withSonarQubeEnv(installationName:'SonarQube'){
                    bat 'mvn clean install sonar:sonar'
                }
            }
        }
        stage ('Test') {
            steps {
                bat 'mvn -Dmaven.test.failure.ignore=true clean install'
            }

        }
        stage('Deploy') {
            steps {
                echo "Deployment application"
            }
        }
    }
}
```

Plan du cours

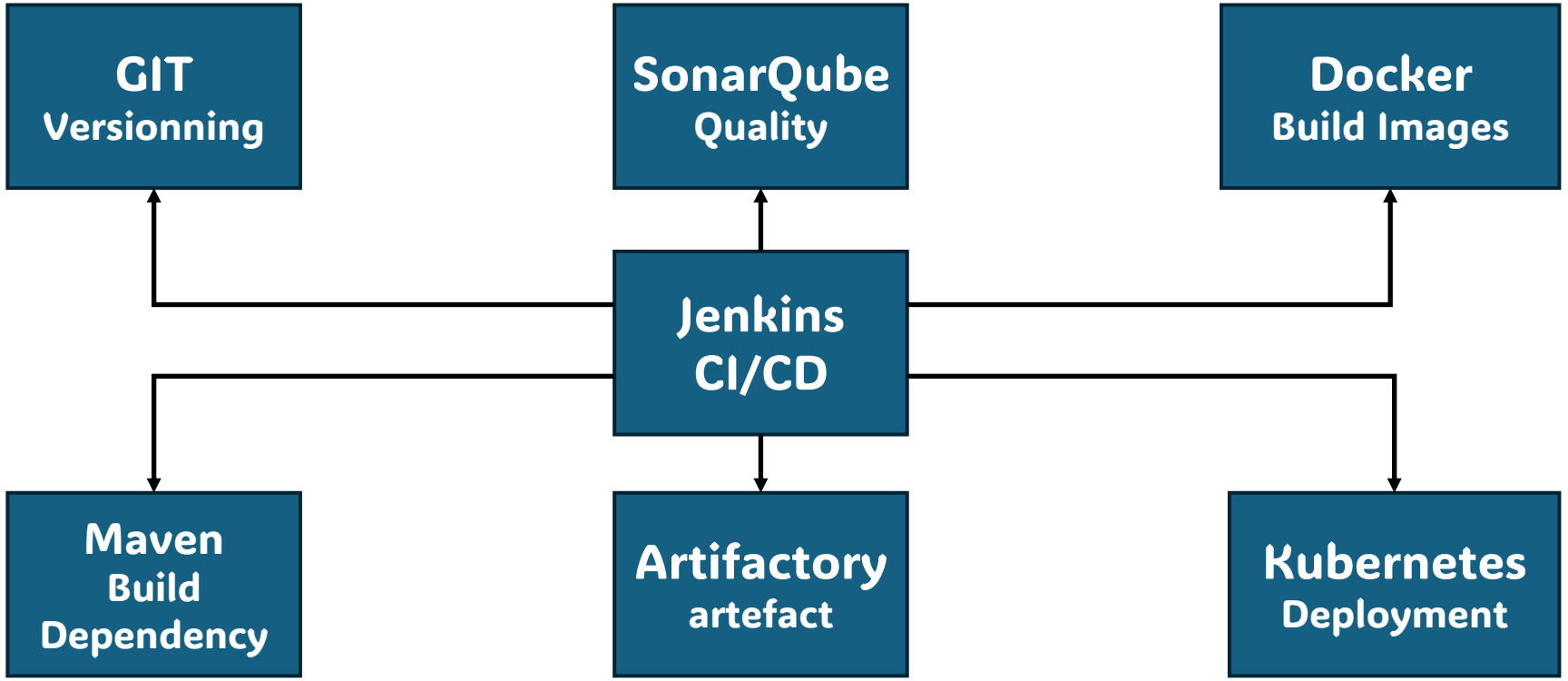
I. Introduction

II. Job & Pipeline

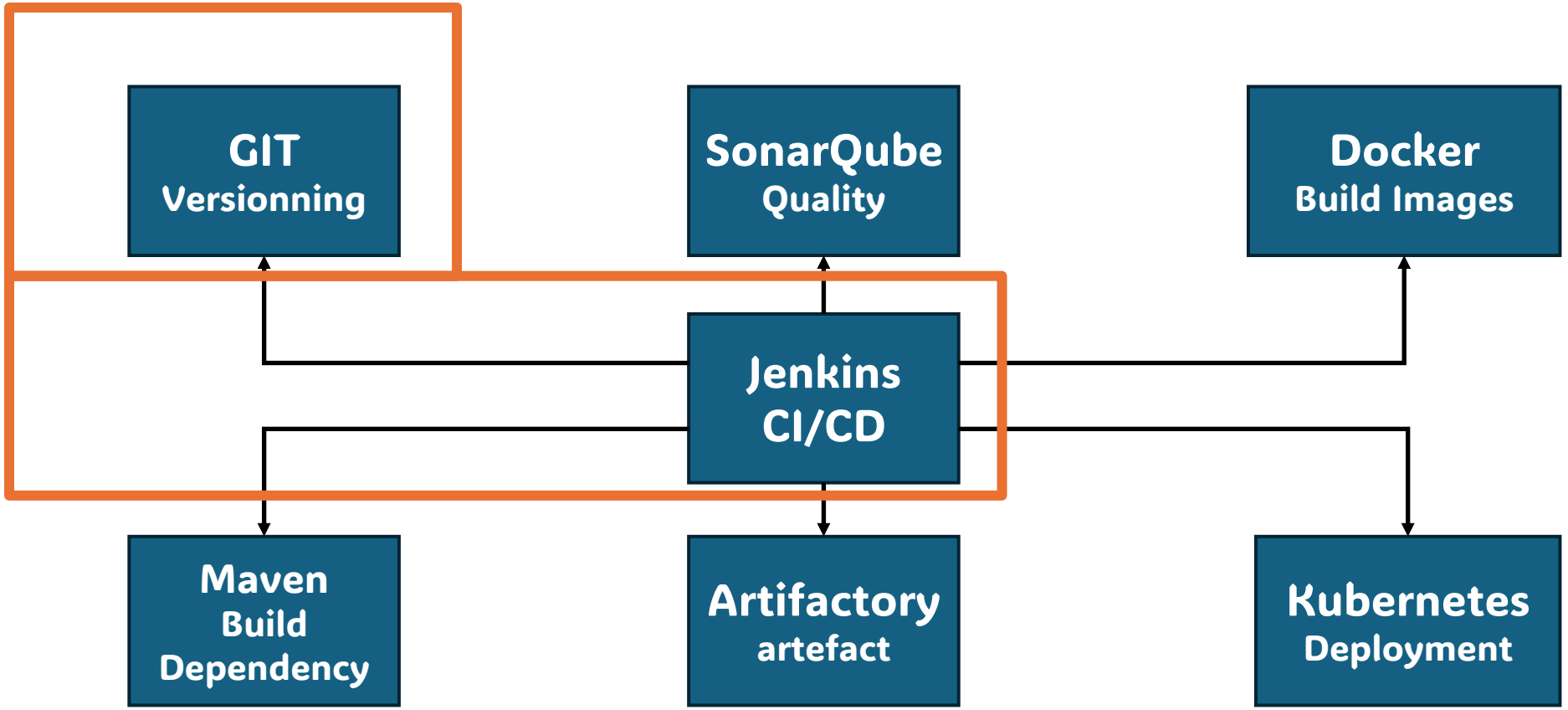
III. Intégration avec les autres outils

IV. Les bonnes pratiques

Intégration avec les autres outils

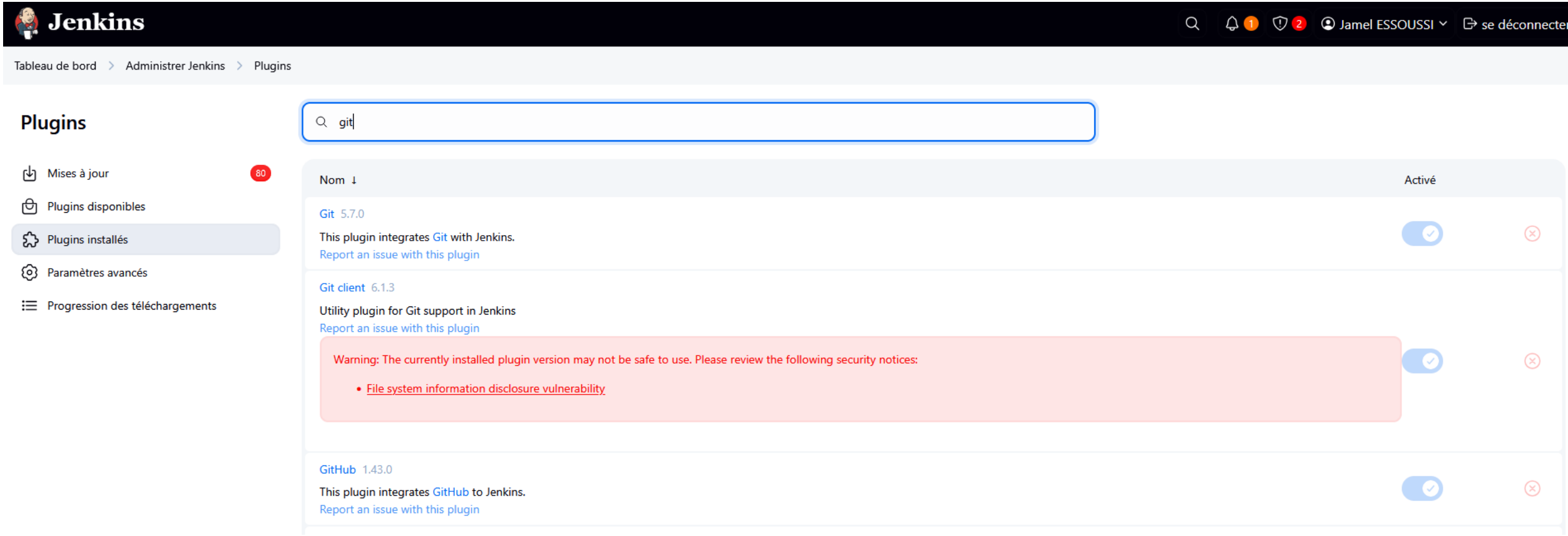


GIT


GIT
Versionning
SonarQube
Quality
Docker
Build Images
Jenkins
CI/CD
Maven
Build
Dependency
Artifactory
artefact
Kubernetes
Deployment

GIT

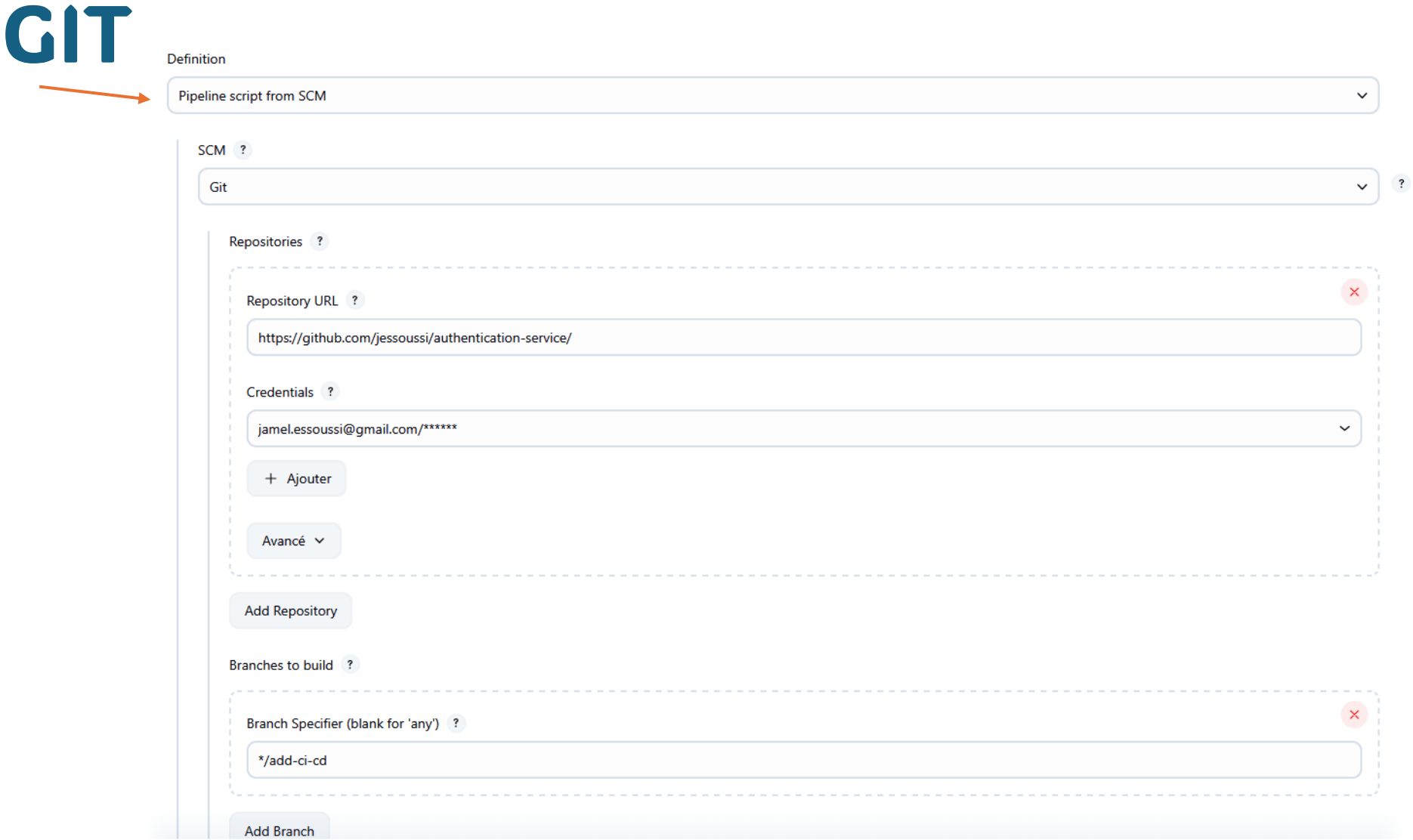
- A travers le plugin GIT de Jenkins



GIT

- A travers le plugin GIT de Jenkins

```
stages {
    stage('Git Checkout') {
        steps {
            script {
                git branch: 'main', url: 'https://github.com/jessoussi/authentication-servic
            }
        }
    }
    stage ('Build') {
```


GIT
Definition
Pipeline script from SCM
SCM
Git
Repositories
Repository URL
https://github.com/jessoussi/authentication-service/
Credentials
jamel.essoussi@gmail.com/******
Ajouter
Avancé
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/add-ci-cd
Add Branch

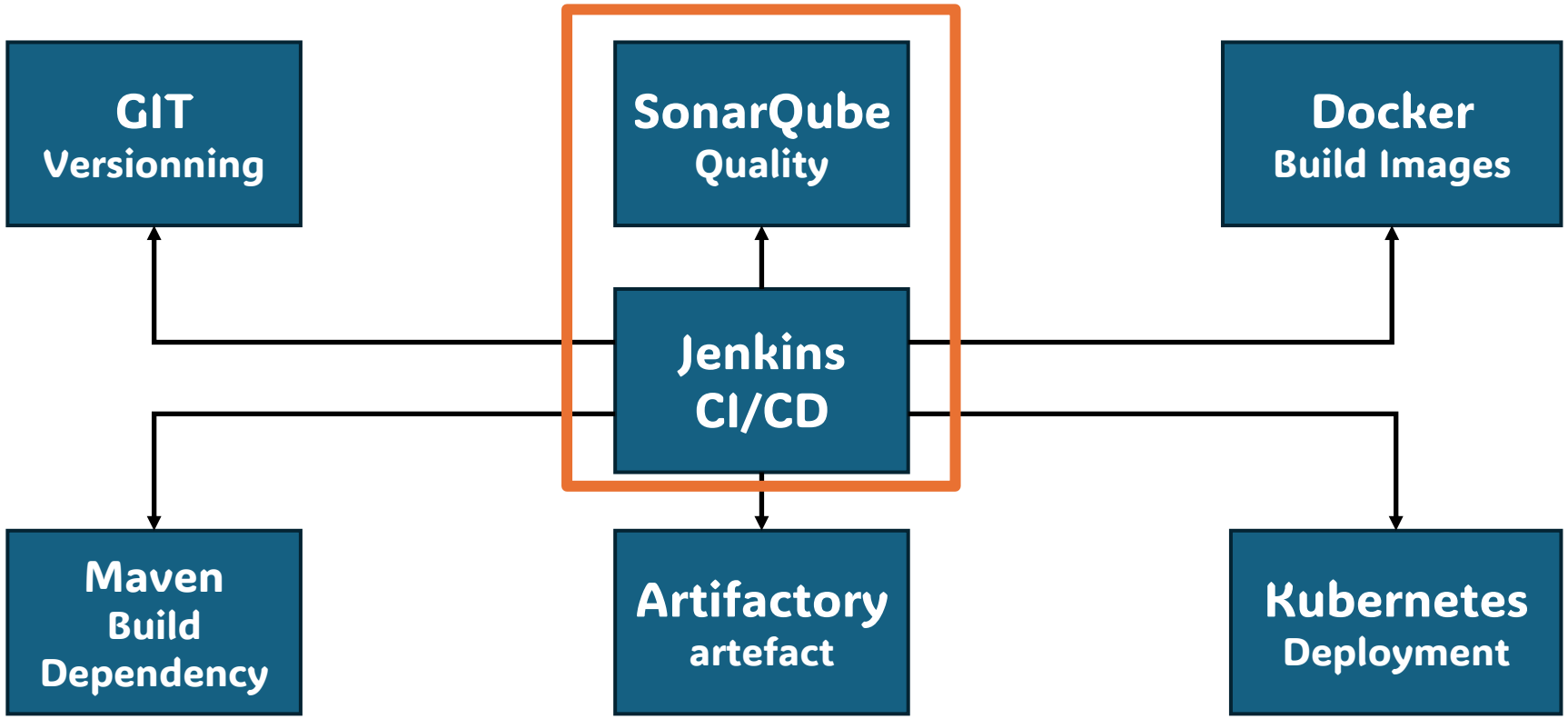
SonarQube

- SonarQube (précédemment Sonar) est un logiciel libre de qualimétrie en continu de code.
- Il aide à la détection, la classification et la résolution de défaut dans le code source, permet d'identifier les duplications de code, de mesurer le niveau de documentation et connaître la couverture de test déployée.

SonarQube

- SonarQube permet une surveillance continue de la qualité du code grâce à son interface web permettant de voir les défauts de l'ensemble du code et ceux ajoutés par la nouvelle version.
- Le logiciel peut être interfacé avec un système d'automatisation comme Jenkins pour inclure l'analyse comme une extension du développement.

SonarQube

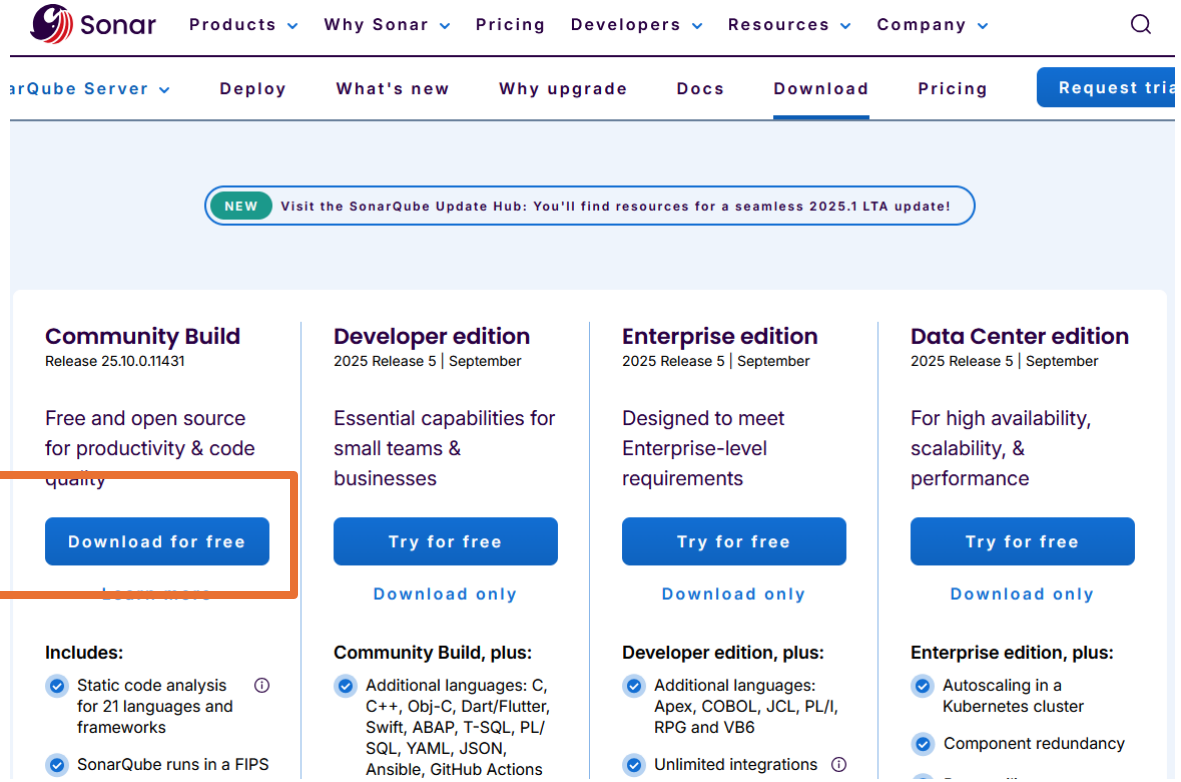

GIT
Versionning
SonarQube
Quality
Docker
Build Images
Jenkins
CI/CD
Maven
Build
Dependency
Artifactory
artefact
Kubernetes
Deployment

SonarQube

- Installation de SonarQube

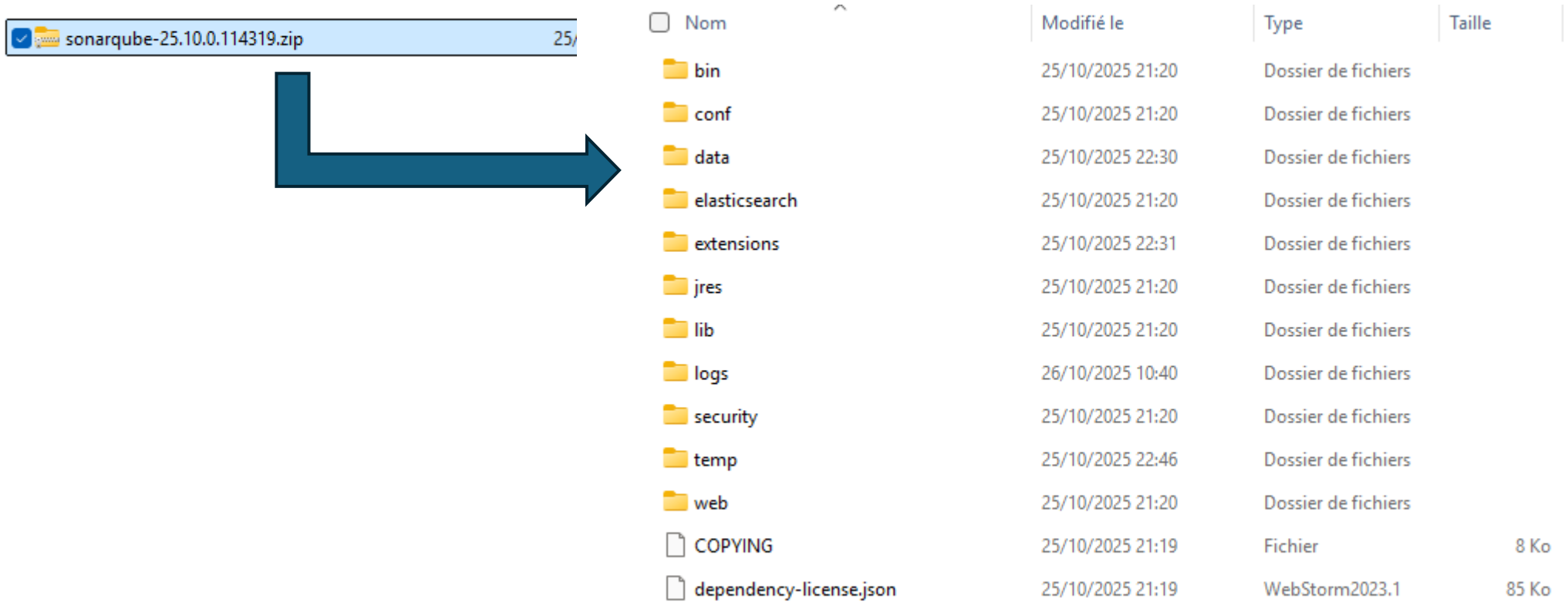
https://www.sonarsource.com/products/sonarqube/downloads/

Téléchargement



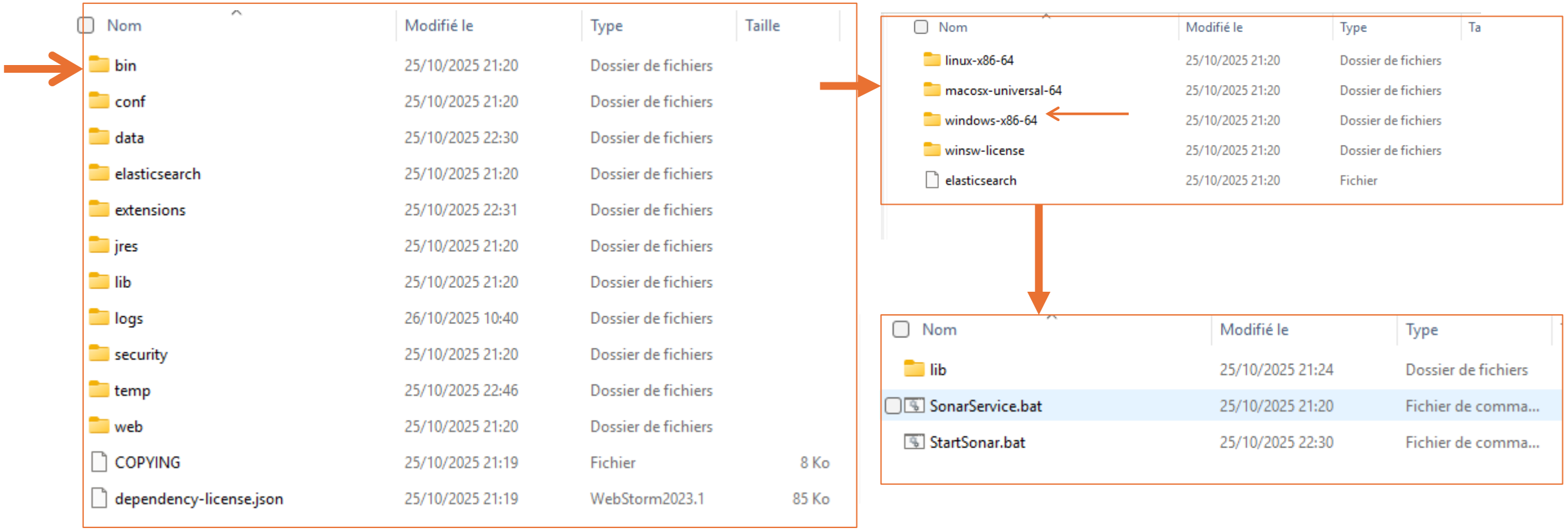
SonarQube

- Dézipper le package téléchargé:



SonarQube

- Lancer le script de démarrage:



Nom	Modifié le	Type	Taille
bin	25/10/2025 21:20	Dossier de fichiers	
conf	25/10/2025 21:20	Dossier de fichiers	
data	25/10/2025 22:30	Dossier de fichiers	
elasticsearch	25/10/2025 21:20	Dossier de fichiers	
extensions	25/10/2025 22:31	Dossier de fichiers	
jres	25/10/2025 21:20	Dossier de fichiers	
lib	25/10/2025 21:20	Dossier de fichiers	
logs	26/10/2025 10:40	Dossier de fichiers	
security	25/10/2025 21:20	Dossier de fichiers	
temp	25/10/2025 22:46	Dossier de fichiers	
web	25/10/2025 21:20	Dossier de fichiers	
COPYING	25/10/2025 21:19	Fichier	8 Ko
dependency-license.json	25/10/2025 21:19	WebStorm2023.1	85 Ko

Nom	Modifié le	Type	Ta
linux-x86-64	25/10/2025 21:20	Dossier de fichiers	
macosx-universal-64	25/10/2025 21:20	Dossier de fichiers	
windows-x86-64	25/10/2025 21:20	Dossier de fichiers	
winsw-license	25/10/2025 21:20	Dossier de fichiers	
elasticsearch	25/10/2025 21:20	Fichier	

Nom	Modifié le	Type
lib	25/10/2025 21:24	Dossier de fichiers
SonarService.bat	25/10/2025 21:20	Fichier de comma...
StartSonar.bat	25/10/2025 22:30	Fichier de comma...

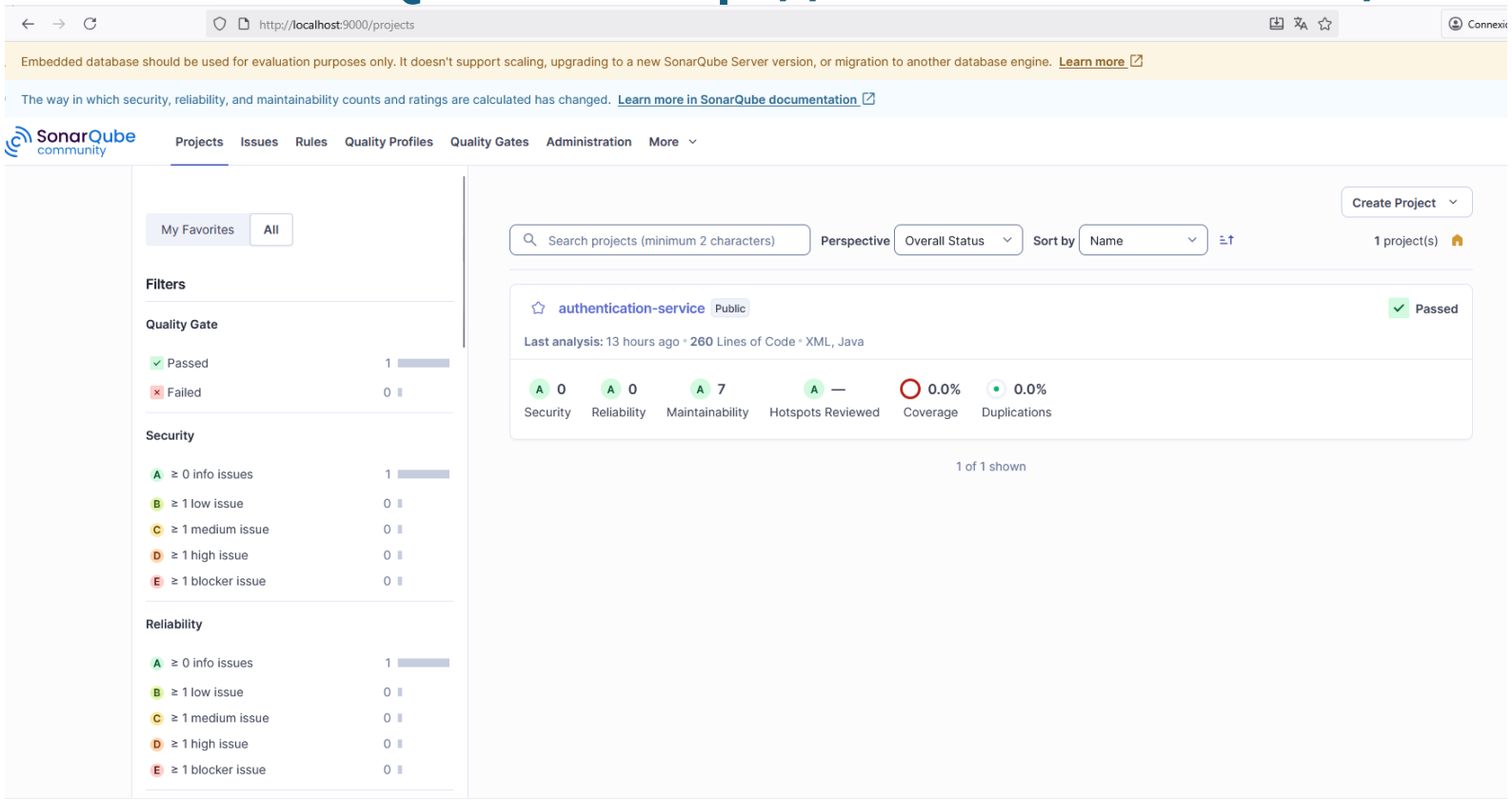
SonarQube

- Lancer le script de démarrage:

```
$ ./StartSonar.bat
'jps' n’est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.
Starting SonarQube...
2025.10.25 22:44:12 INFO  app[][o.s.a.AppFileSystem] Cleaning or creating temp directory C:\dev_tools\sonarqube-25.10.0.114319\sonar
qube-25.10.0.114319\temp
2025.10.25 22:44:12 INFO  app[][o.s.a.es.EsSettings] Elasticsearch listening on [HTTP: 127.0.0.1:9011, TCP: 127.0.0.1:{}]
2025.10.25 22:44:12 INFO  app[][o.s.a.ProcessLauncherImpl] Launch process[ELASTICSEARCH] from [C:\dev_tools\sonarqube-25.10.0.114319
\sonarqube-25.10.0.114319\elasticsearch]: C:\Program Files\Java\jdk-17.0.2\bin\java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=serve
r -Dcli.script=./bin/elasticsearch -Dcli.libs=lib/tools/server-cli -Des.path.home=C:\dev_tools\sonarqube-25.10.0.114319\sonarqube-25
.10.0.114319\elasticsearch -Des.path.conf=C:\dev_tools\sonarqube-25.10.0.114319\sonarqube-25.10.0.114319\temp\conf\es -Des.distribut
ion.type=tar -cp C:\dev_tools\sonarqube-25.10.0.114319\sonarqube-25.10.0.114319\elasticsearch\lib\*;C:\dev_tools\sonarqube-25.10.0.1
14319\sonarqube-25.10.0.114319\elasticsearch\lib\cli-launcher\* org.elasticsearch.launcher.CliToolLauncher
2025.10.25 22:44:12 INFO  app[][o.s.a.SchedulerImpl] Waiting for Elasticsearch to be up and running
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoid pot
ential conflicts
2025.10.25 22:44:34 INFO  app[][o.s.a.SchedulerImpl] Process[es] is up
2025.10.25 22:44:34 INFO  app[][o.s.a.ProcessLauncherImpl] Launch process[WEB_SERVER] from [C:\dev_tools\sonarqube-25.10.0.1143
19\sonarqube-25.10.0.114319]: C:\Program Files\Java\jdk-17.0.2\bin\java -Djava.awt.headless=true -Dfile.encoding=UTF-8 -Djava.i
o.tmpdir=C:\dev_tools\sonarqube-25.10.0.114319\sonarqube-25.10.0.114319\temp -XX:-OmitStackTraceInFastThrow --add-opens=java.ba
se/java.util=ALL-UNNAMED --add-opens=java.base/java.lang=ALL-UNNAMED --add-opens=java.base/java.io=ALL-UNNAMED --add-opens=java
.rmi/sun.rmi.transport=ALL-UNNAMED --add-exports=java.base/jdk.internal.ref=ALL-UNNAMED --add-opens=java.base/java.nio=ALL-UNNA
MED --add-opens=java.base/sun.nio.ch=ALL-UNNAMED --add-opens=java.management/sun.management=ALL-UNNAMED --add-opens=jdk.managem
ent/com.sun.management.internal=ALL-UNNAMED -Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError -Dhttp.nonProxyHosts=localhost|12
7.*|[::1] -cp ./lib/sonar-application-25.10.0.114319.jar;C:\dev_tools\sonarqube-25.10.0.114319\sonarqube-25.10.0.114319\lib\jdb
c\h2\h2-2.3.232.jar org.sonar.server.app.WebServer C:\dev_tools\sonarqube-25.10.0.114319\sonarqube-25.10.0.114319\temp\sq-proce
ss12833236382398619013properties
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoi
d potential conflicts
2025.10.25 22:44:56 INFO  app[][o.s.a.SchedulerImpl] Process[web] is up
```

SonarQube

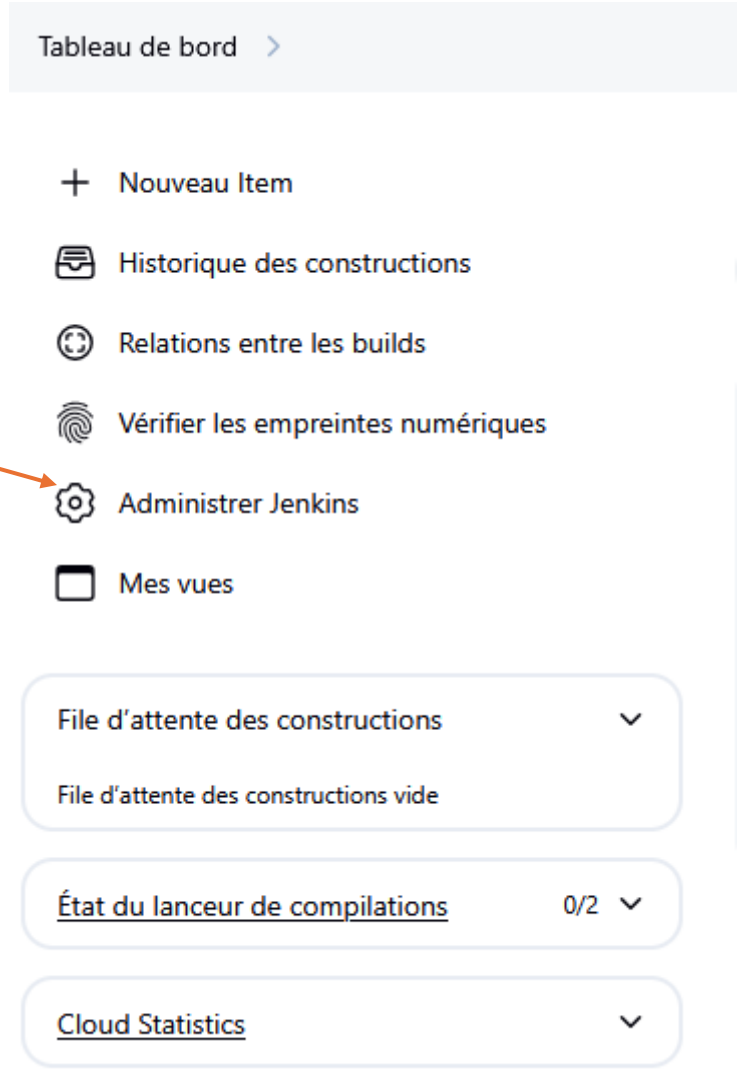
- Interface SonarQube : http://localhost:9000/



SonarQube

- Branchement avec Jenkins

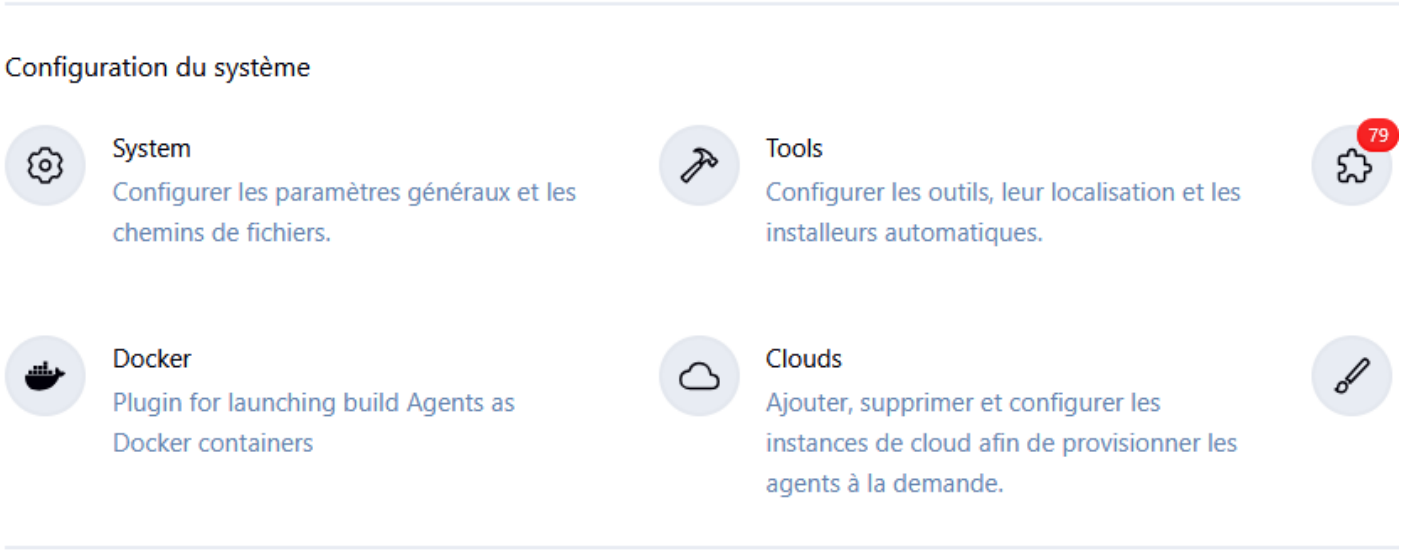
Aller sur la page d'administration de Jenkins



SonarQube

- Branchement avec Jenkins

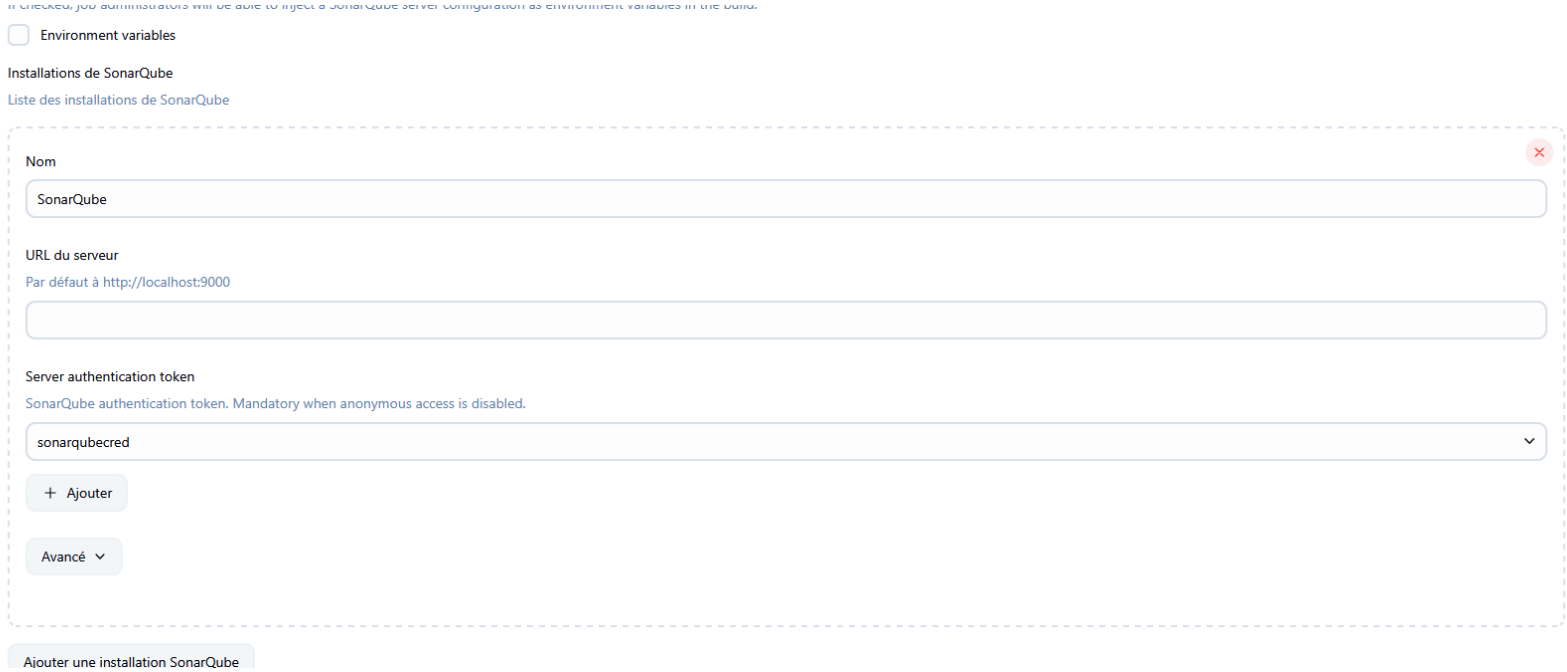
Aller sur la page système



SonarQube

- Branchement avec Jenkins

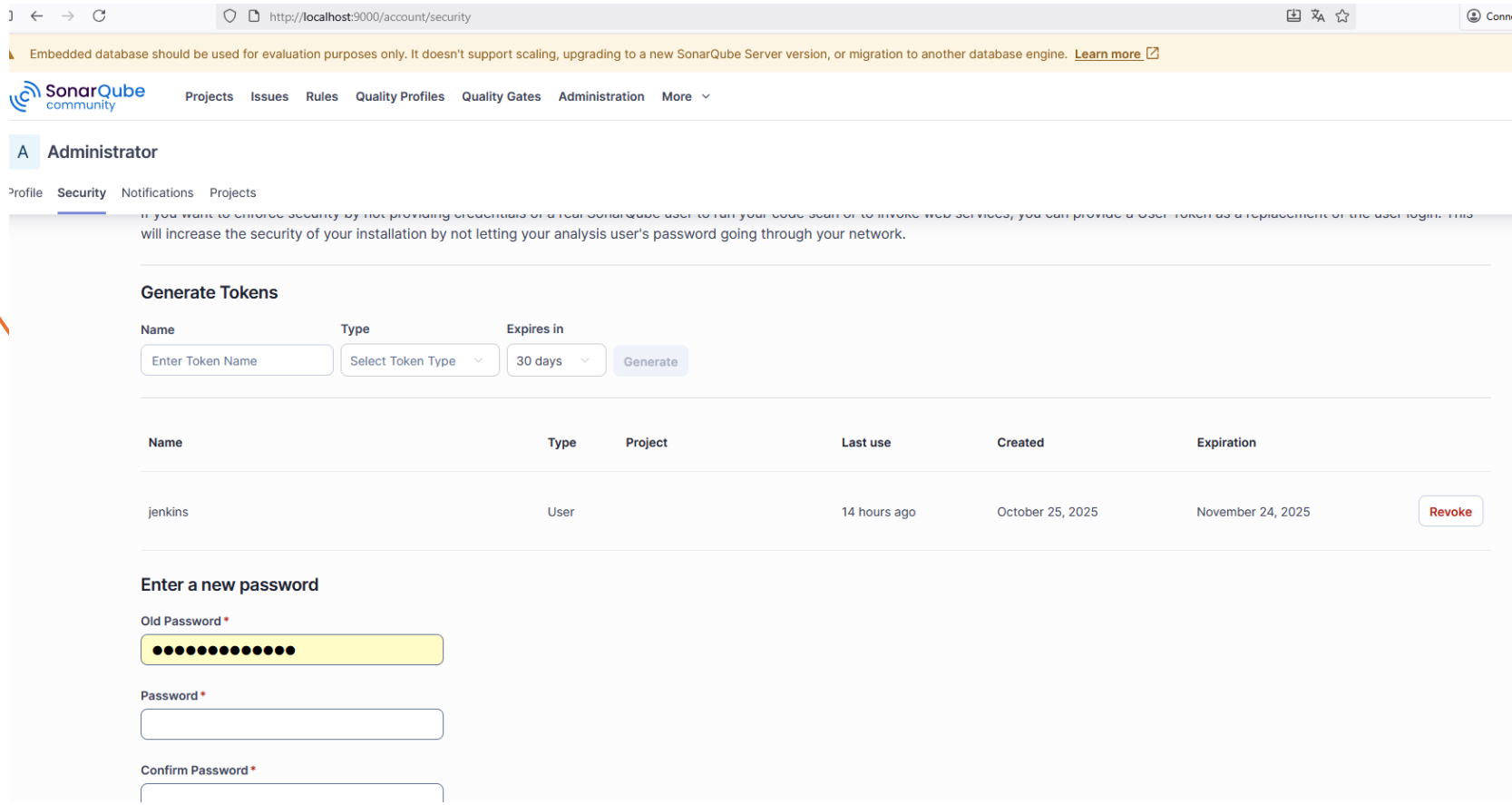
Credential: Token Sonar



SonarQube

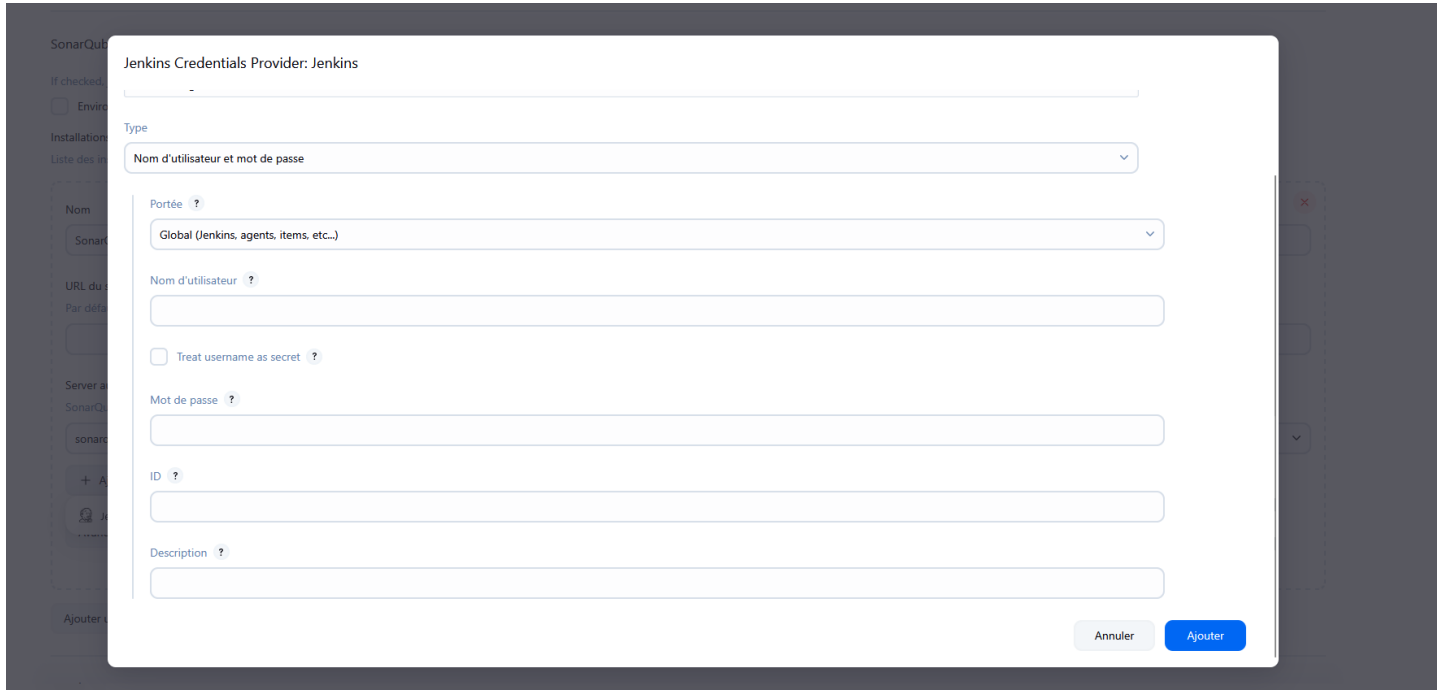
- Génée̊r un token sur SonrQube

Credential: Token Sonar



SonarQube

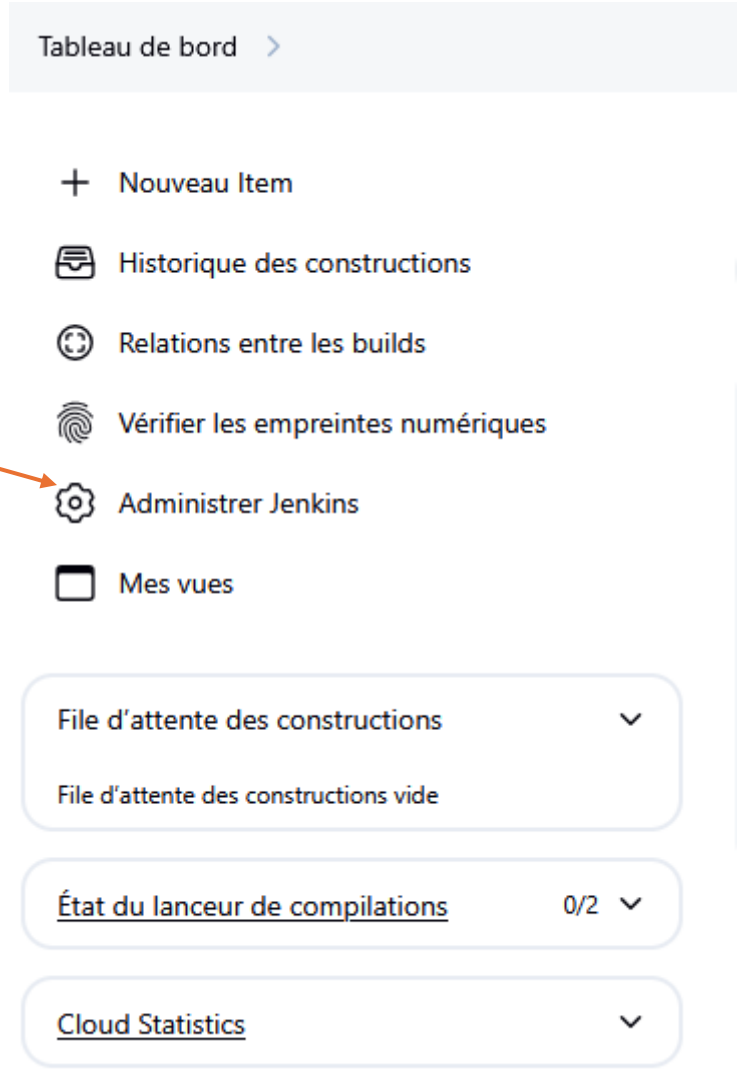
- Renseigner le token dans Jenkins



SonarQube

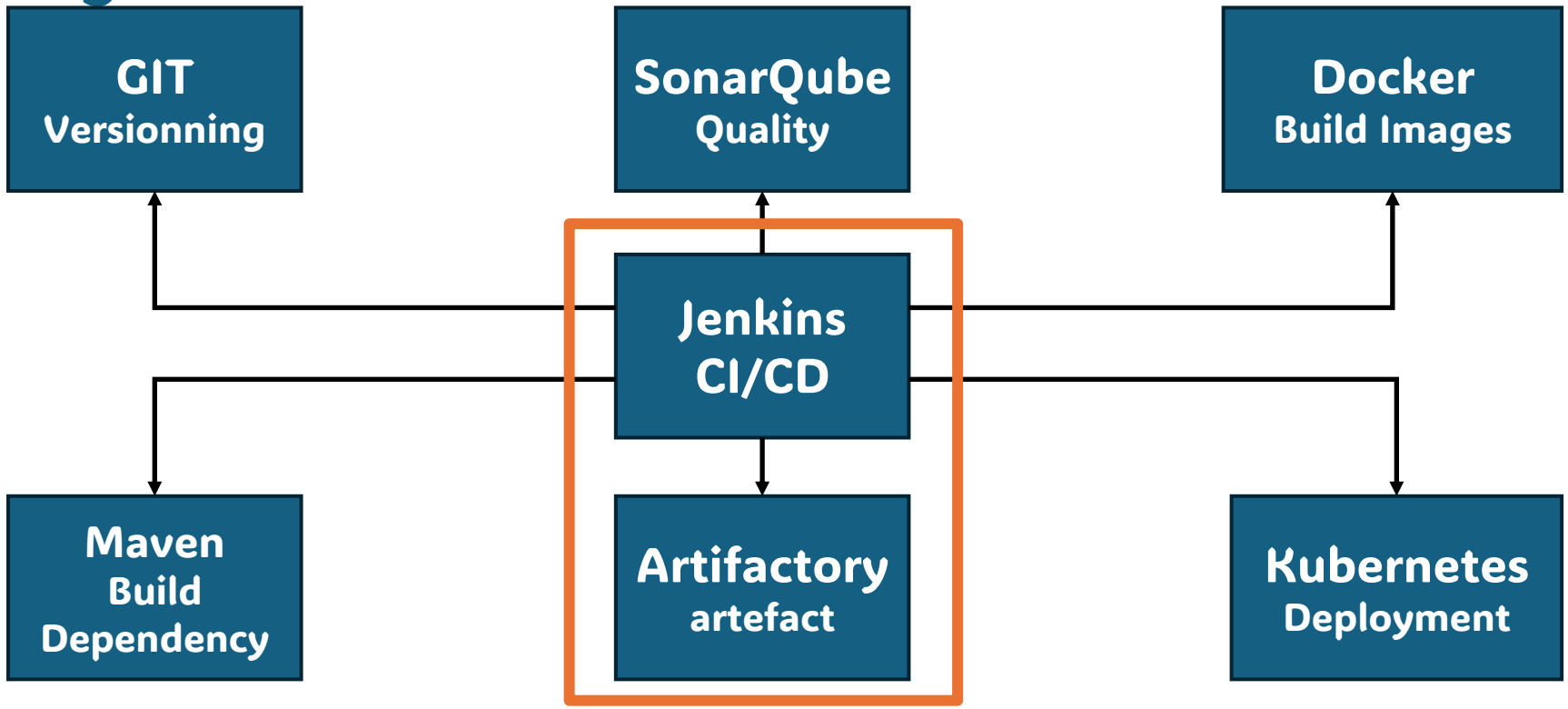
- Branchement avec Jenkins

Aller sur la page d'administration de Jenkins



Artifactory

- Dépôt Centrale dans l'organisation
- Partage des livrables



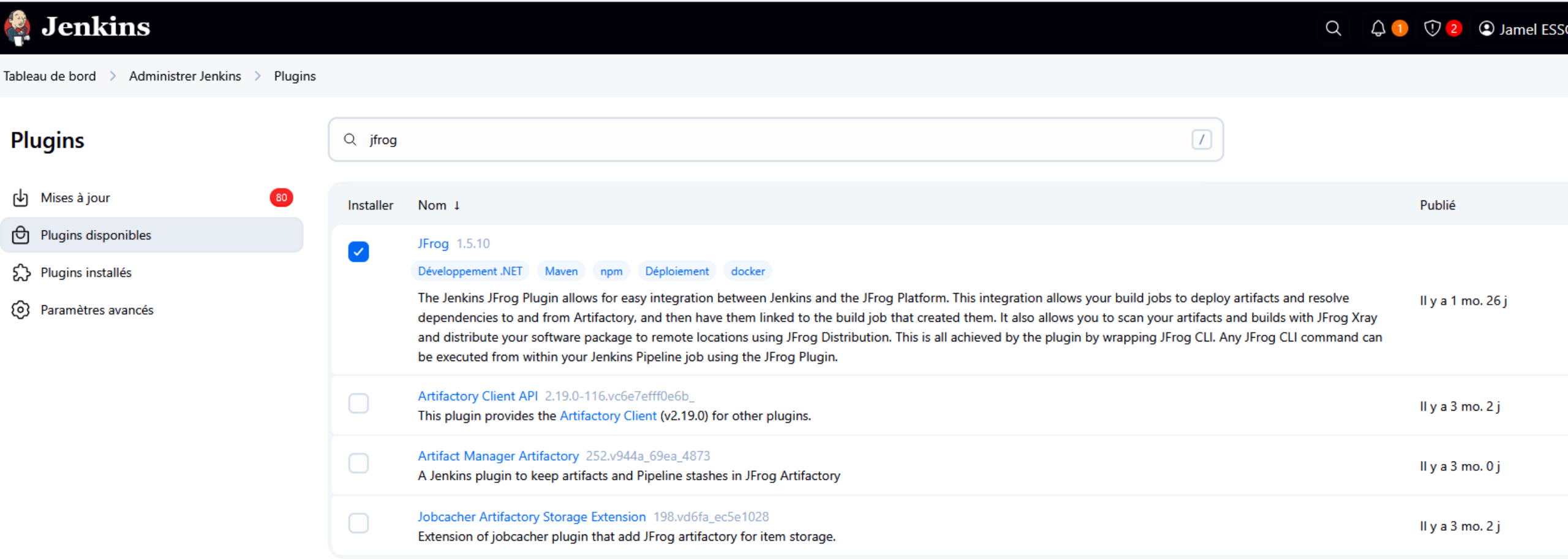
Artifactory

- JFrog Artifactory est une solution pour héberger et gérer tous les artefacts logiciels, les modèles d'IA/ML, les binaires, les packages, les fichiers, les conteneurs, les composants et les versions utilisés et générés dans la chaîne d'approvisionnement logicielle de l'organisation.

Artifactory

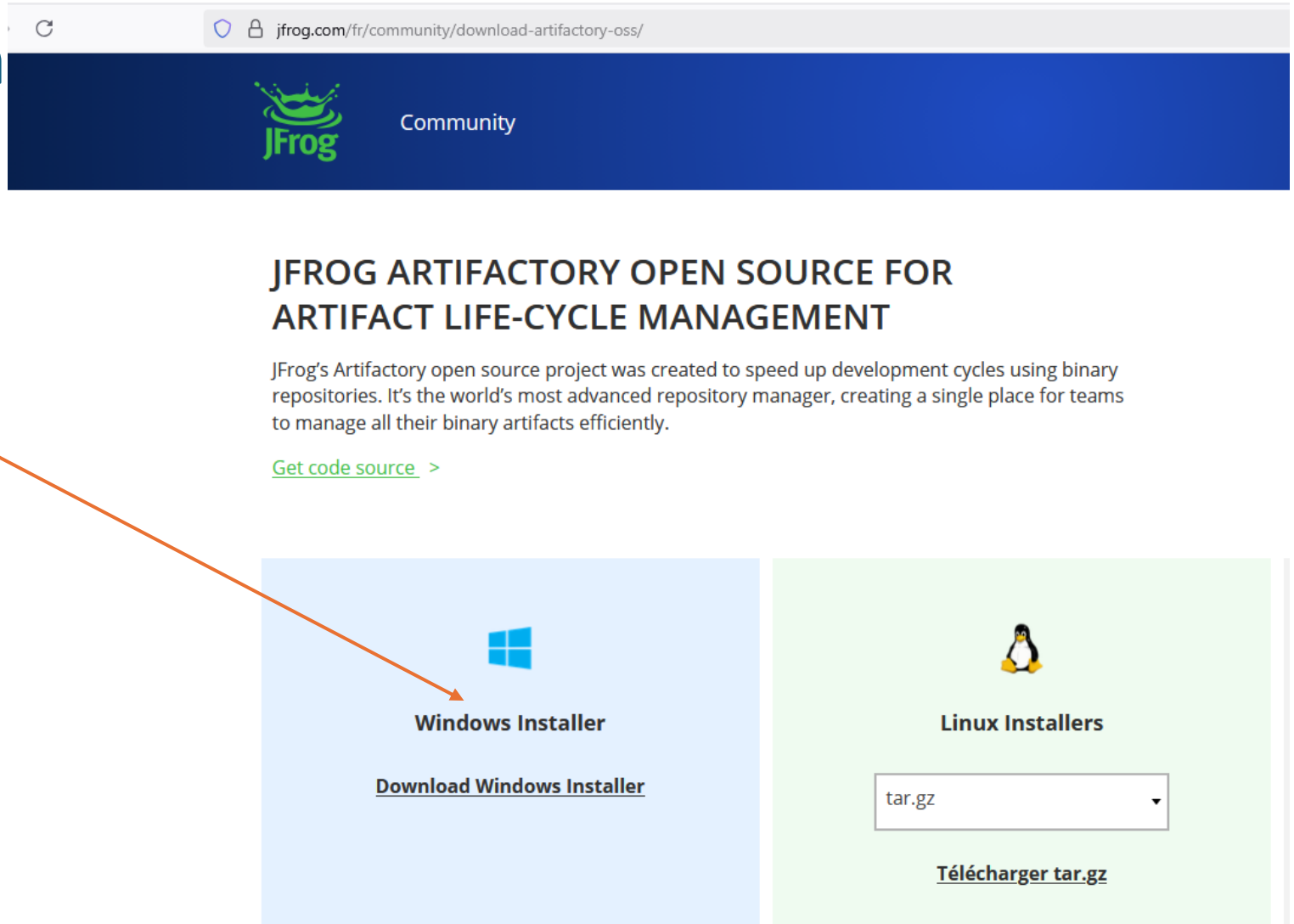
- Artifactory sert de plaque tournante centrale pour le DevOps et les développeurs, en s'intégrant aux outils et processus pour améliorer l'automatisation, capturer les attestations, assurer l'intégrité des livraisons et fournir une visibilité inégalée sur vos processus de développement.

Artifactory Jenkins Plugin



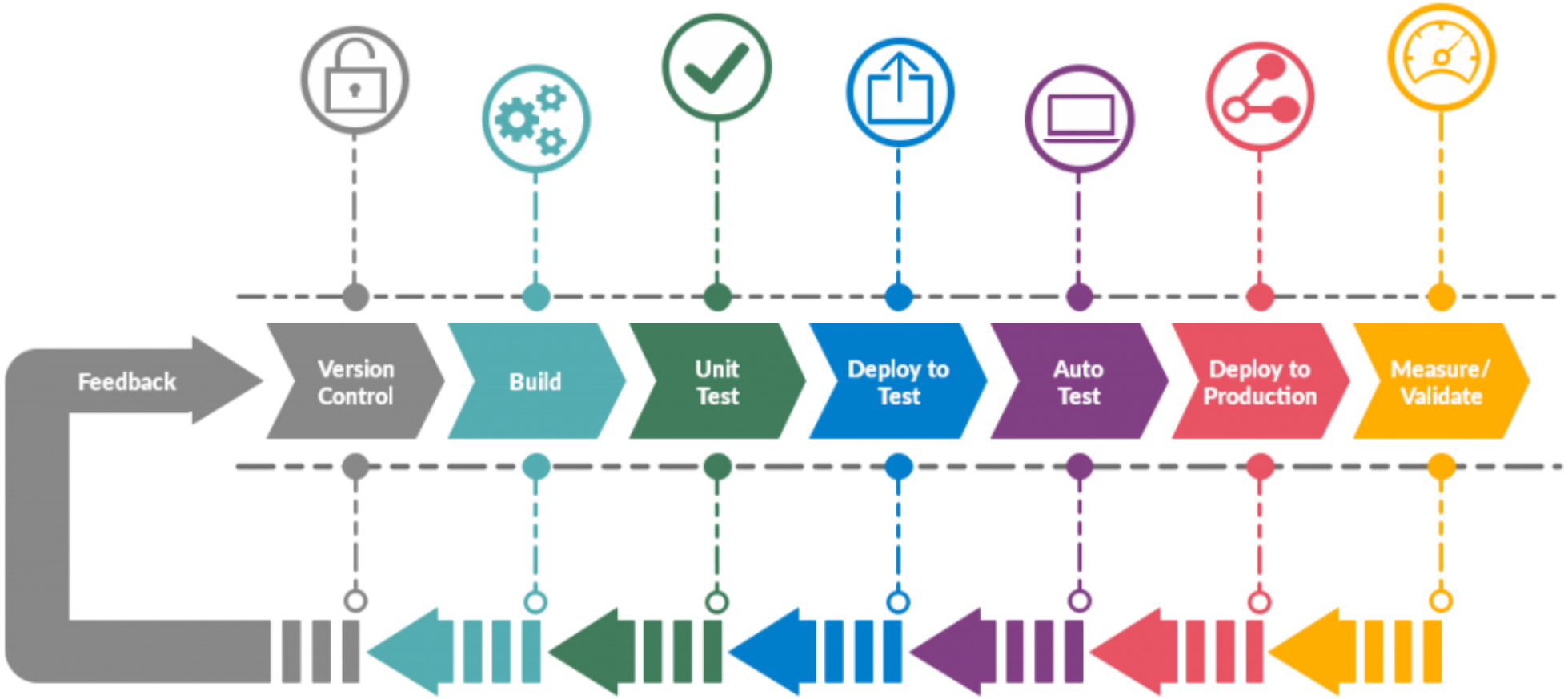
Artifactory

- Installation



Téléchargement

Introduction



Plan du cours

I. Introduction

II. Job & Pipeline

III. Intégration avec les autres outils

IV. Les bonnes pratiques

Les bonnes pratiques

- **Créer un Job Jenkins par Branche**

Merci