

# DevOps

## CI / CD

# Jenkins

Jamel ESSOUSSI: Architecte logiciel

[jamel.essoussi@gmail.com](mailto:jamel.essoussi@gmail.com)

<https://github.com/jessoussi>

2025

# Plan du cours

I. Introduction

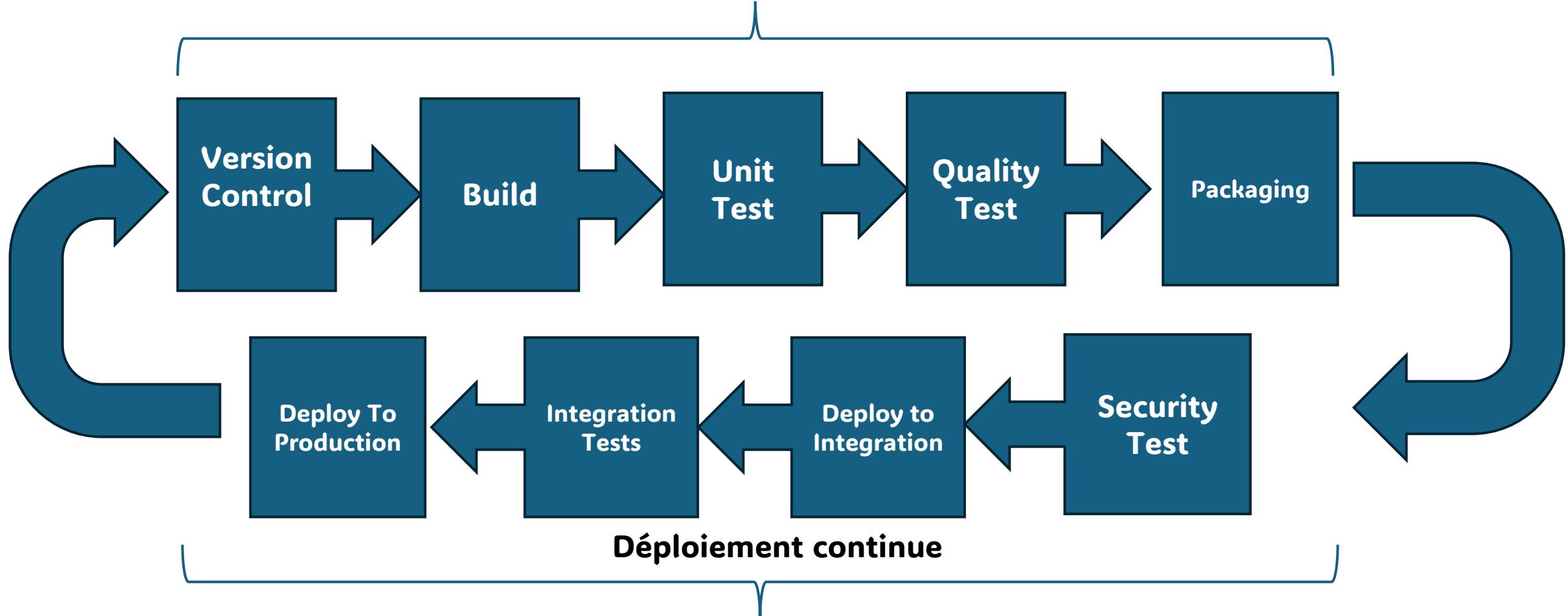
II. Job & Pipeline

III. Intégration avec les autres outils

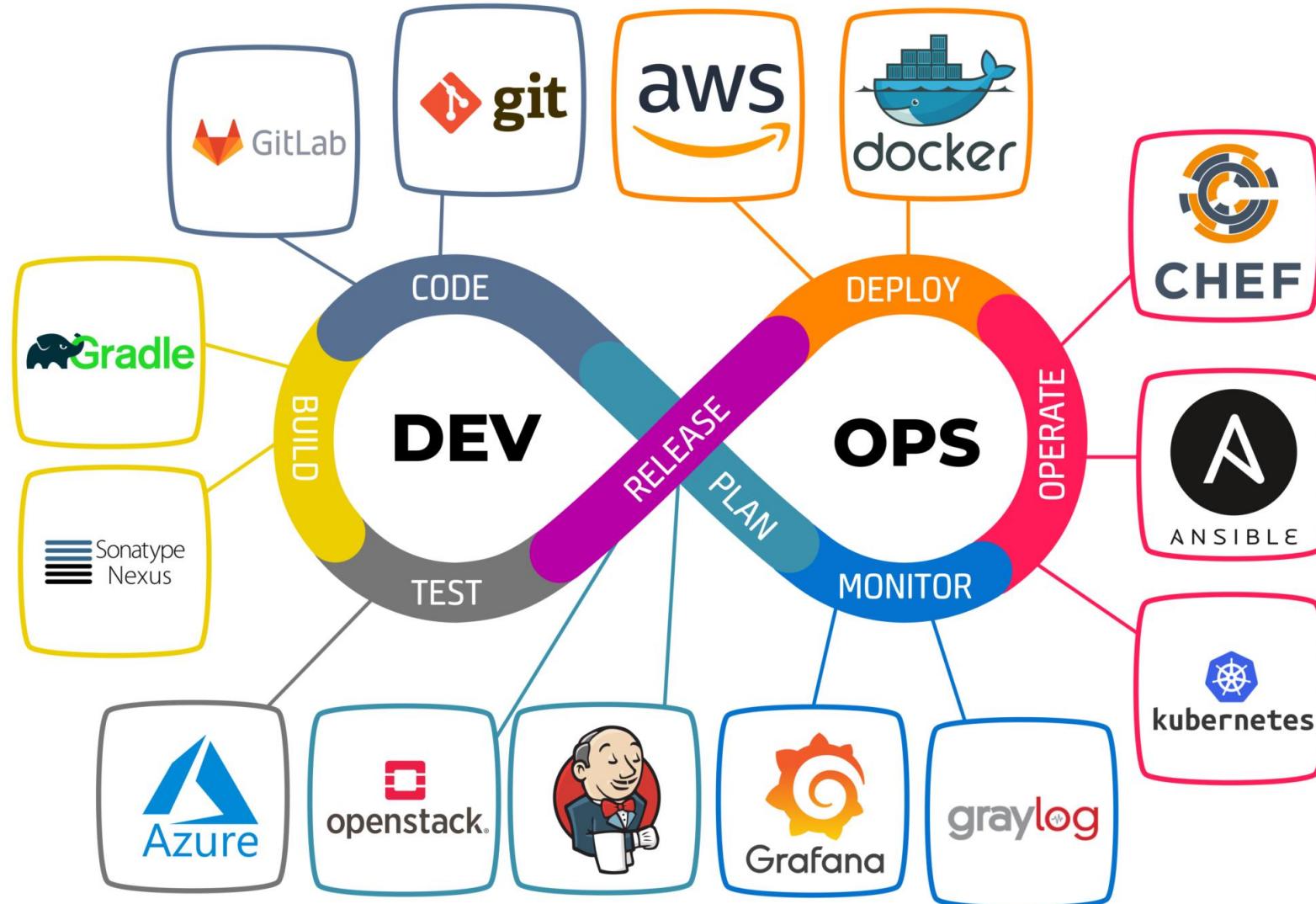
IV. Les bonnes pratiques

# Introduction

Intégration continue



# 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 chaîne 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.

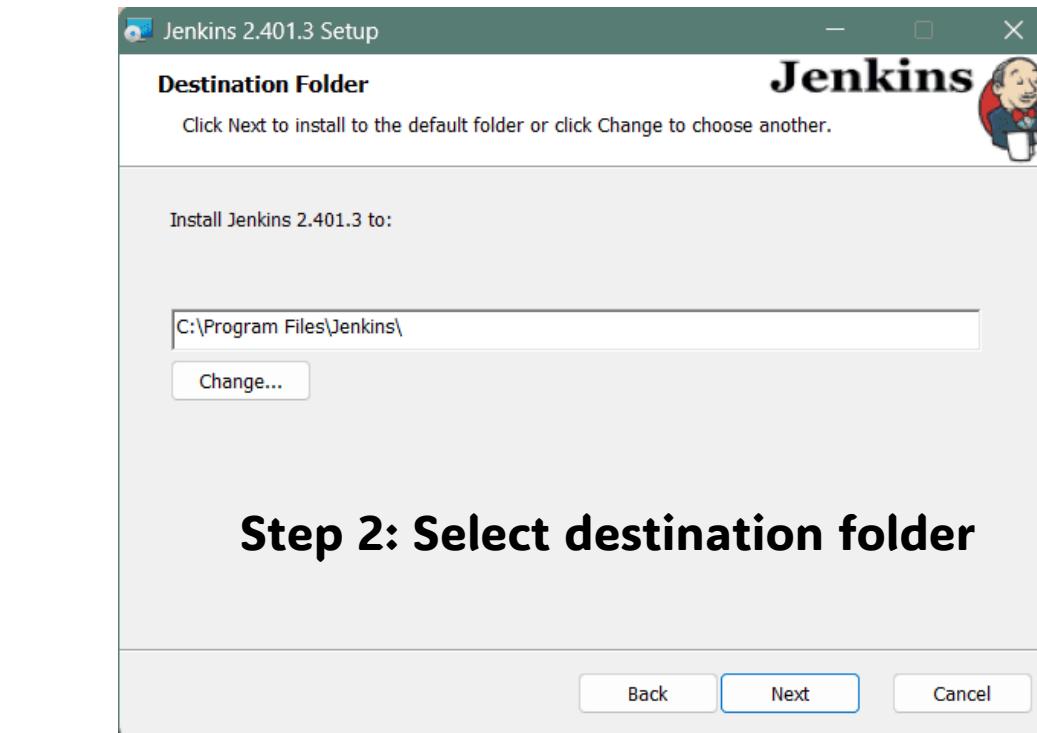
# Installation Jenkins

- Téléchargement

- <https://www.jenkins.io/download/#downloading-jenkins>



**Step 1: Setup wizard**



**Step 2: Select destination folder**

# Installation Jenkins

Jenkins 2.401.3 Setup

## Service Logon Credentials

Enter service credentials for the service.

Jenkins 2.401.3 installs and runs as an independent Windows service. To operate in this manner, you must supply the user account credentials for Jenkins 2.401.3 to run successfully.

**Logon Type:**

Run service as LocalSystem (not recommended)

Run service as local or domain user:

Account:

Password:

[Test Credentials](#) ⚠️ Credentials must be tested to continue

**Step 3: Service logon credentials**

[Back](#) [Next](#) [Cancel](#)

Jenkins 2.401.3 Setup

## Port Selection

Choose a port for the service.

Please choose a port.

**Port Number (1-65535):**

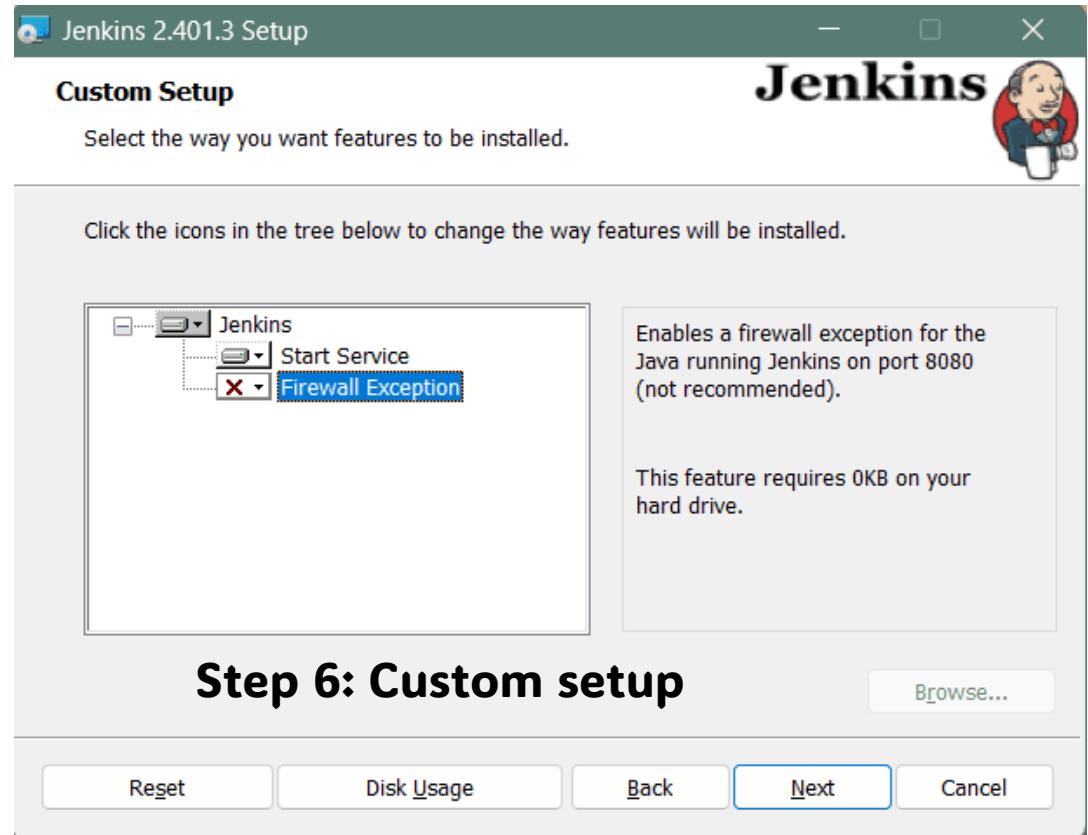
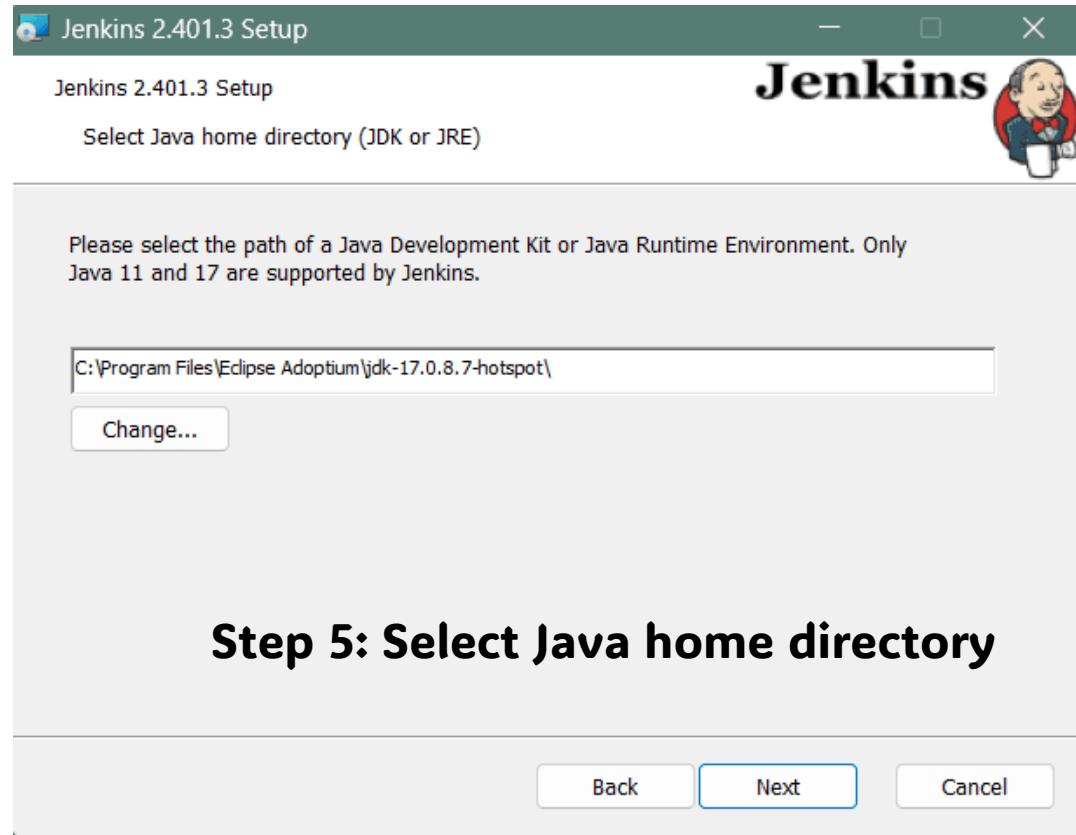
[Test Port](#) ⚠️ Click 'Test Port' button to proceed

It is recommended that you accept the selected default port.

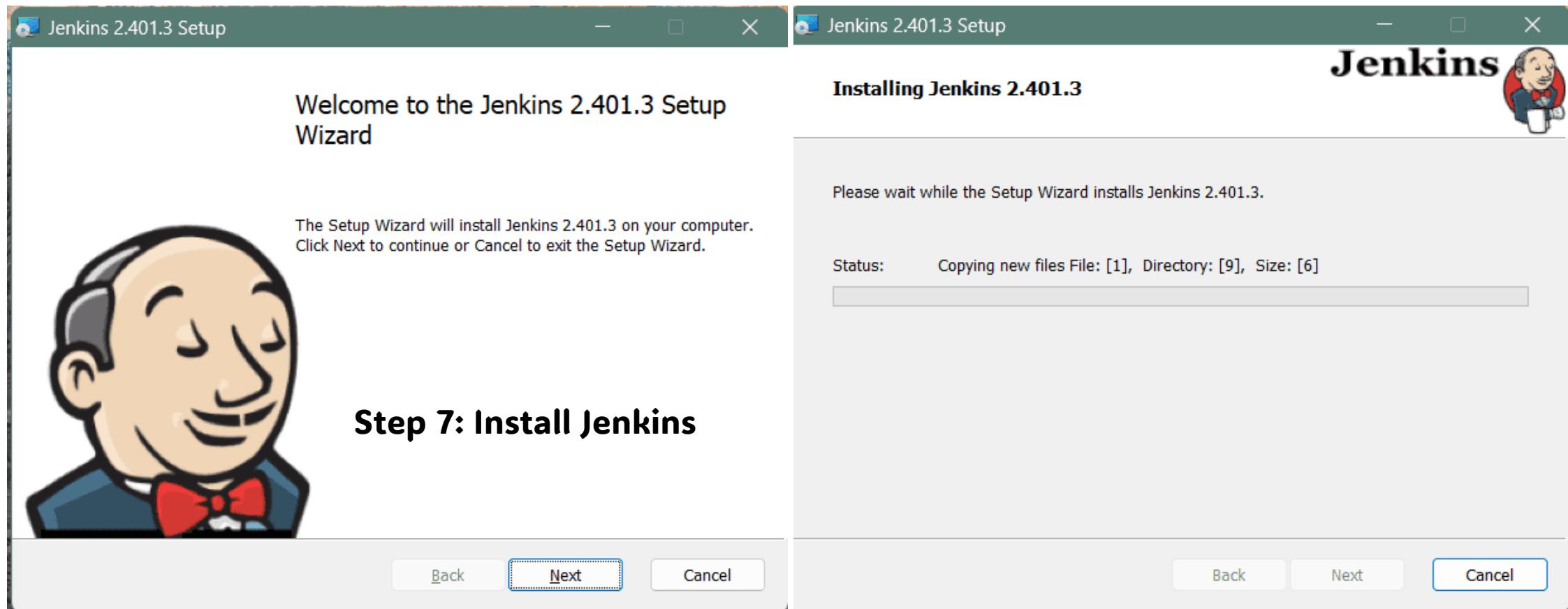
**Step 4: Port selection**

[Back](#) [Next](#) [Cancel](#)

# Installation Jenkins



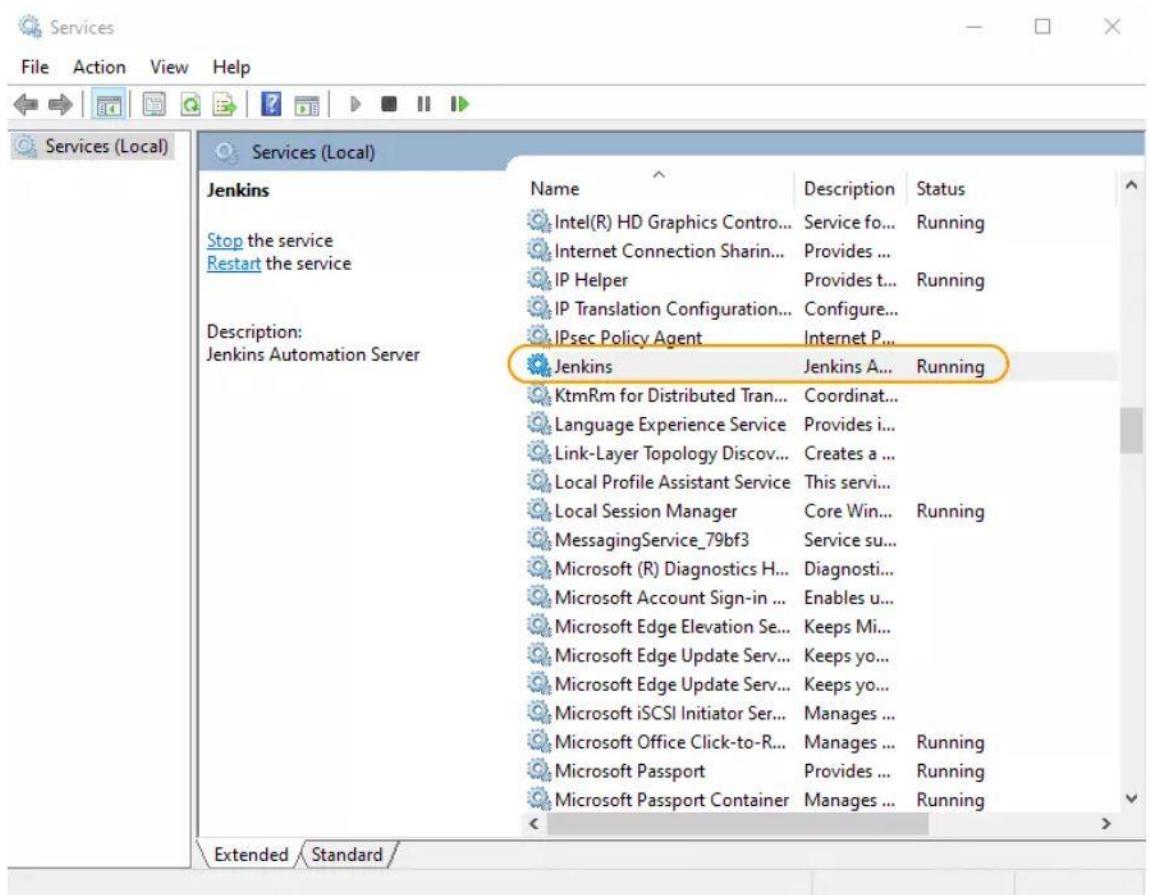
# Installation Jenkins



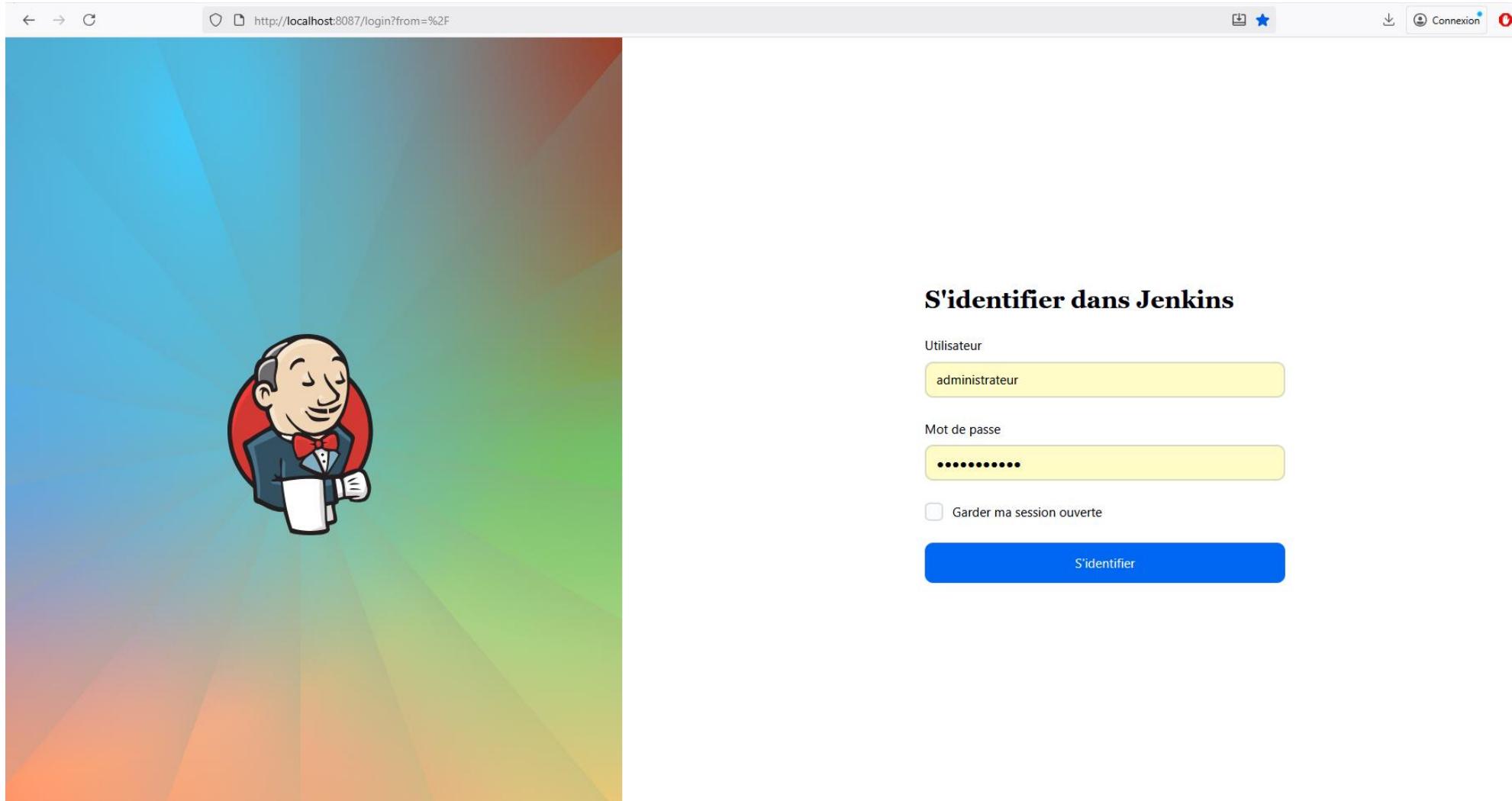
# Installation Jenkins



**Step 8: Finish Jenkins installation**



# Installation Jenkins



# Plan du cours

I. Introduction

**II. Job & Pipeline**

III. Intégration avec les autres outils

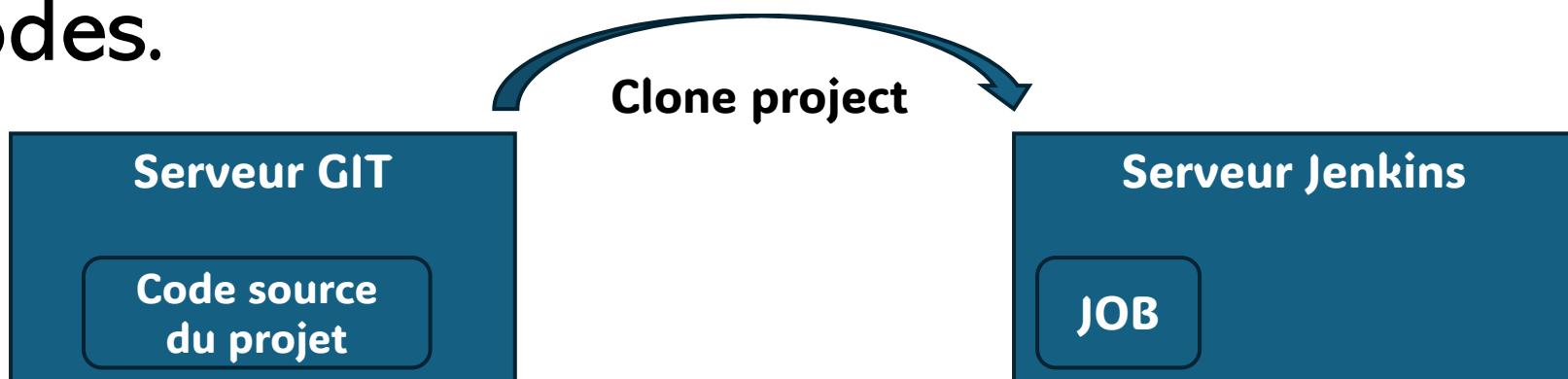
IV. Les bonnes pratiques

# 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

Tableau de bord >

+ Nouveau Item

Historique des constructions

Relations entre les builds

Vérifier les empreintes numériques

Administrer Jenkins

Mes vues

File d'attente des constructions

File d'attente des constructions vide

État du lanceur de compilations 0/2

Cloud Statistics

Tous +

## Job Jenkins

S	M	Nom du projet ↓	Dernier succès	Dernier échec	Dernière durée
✗	rainy	devops-sample	6 mo. 1 j #53	27 j #57	3 mn 10 s
✓	sunny	kubectl	5 mo. 9 j #11	6 mo. 7 j #9	13 s

Icône: S M L

Job Jenkins



# Job

Jenkins

Tableau de bord >

+ Nouveau Item ← Ajout d'un nouveau Job Ajouter une description

Historique des constructions Tous +

Relations entre les builds

Vérifier les empreintes numériques

Administrer Jenkins

Mes vues

File d'attente des constructions ▾ File d'attente des constructions vide

État du lanceur de compilations 0/2 ▾

Cloud Statistics ▾

S M Nom du projet ↓ Dernier succès Dernier échec Dernière durée

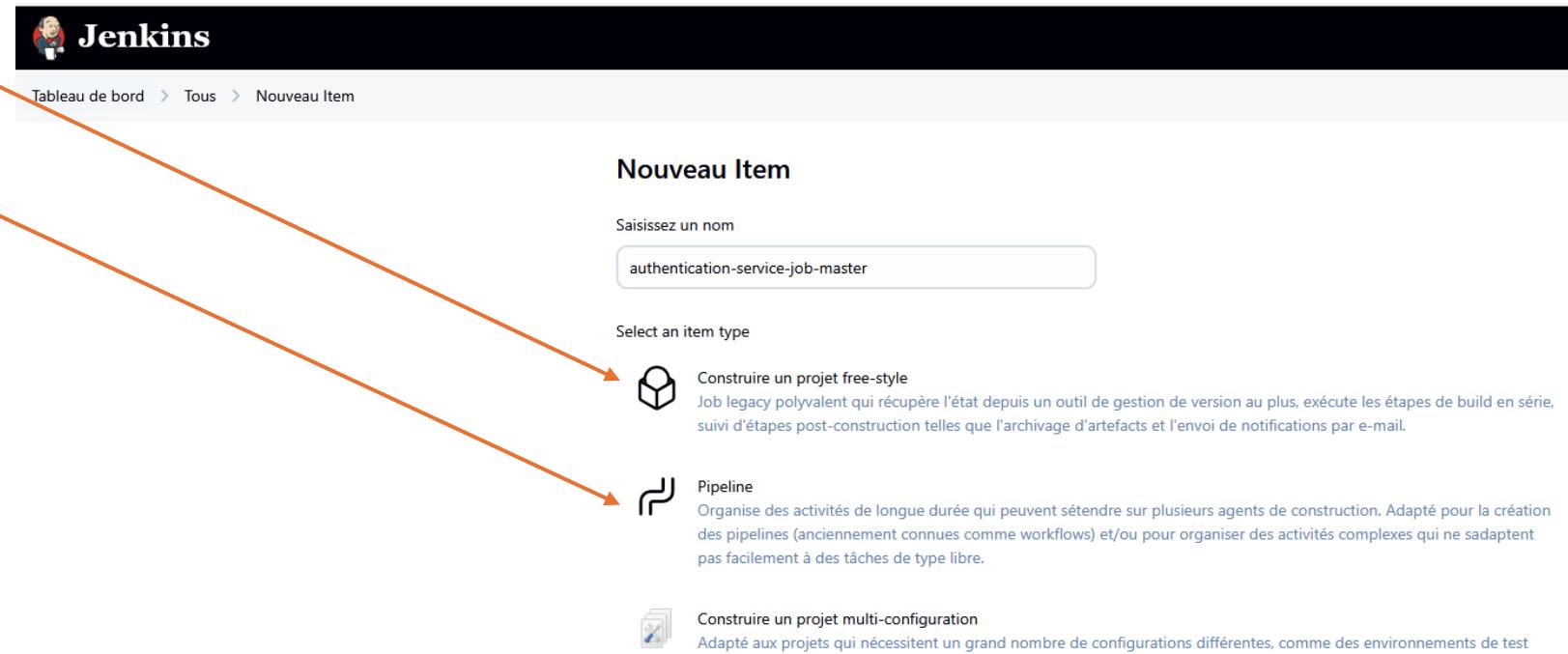
✗	rainy	devops-sample	6 mo. 1 j #53	27 j #57	3 mn 10 s
✓	sunny	kubectl	5 mo. 9 j #11	6 mo. 7 j #9	13 s

Icône: S M L

ooo

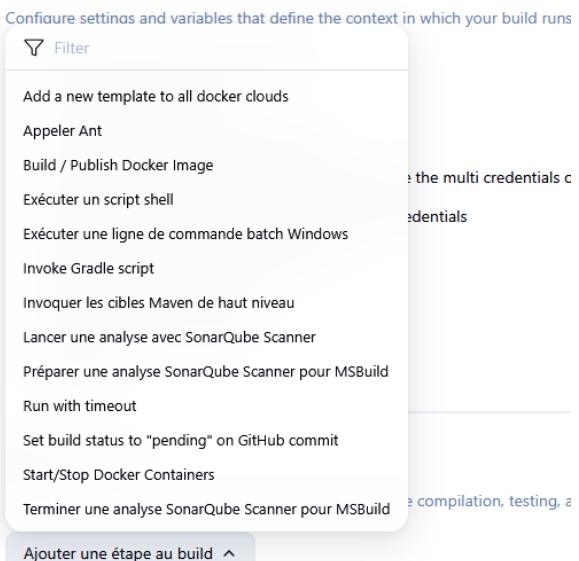
# Job

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



# Job

- Freeestyle
  - Configuration manuelle du job  
GIT, étapes, Script, ...



Jenkins

Tableau de bord > freestyle\_job > Configuration

Configurer Général

Général

Description

First Build authentication service

Texte brut Prévisualisation

Ce build a des paramètres ?

Commit agent's Docker container ?

Define a Docker template

GitHub project

Project url ?

https://github.com/jessoussi/authentication-service/

Avancé ▾

Supprimer les anciens builds ?

Throttle builds ?

Exécuter des builds simultanément si nécessaire ?

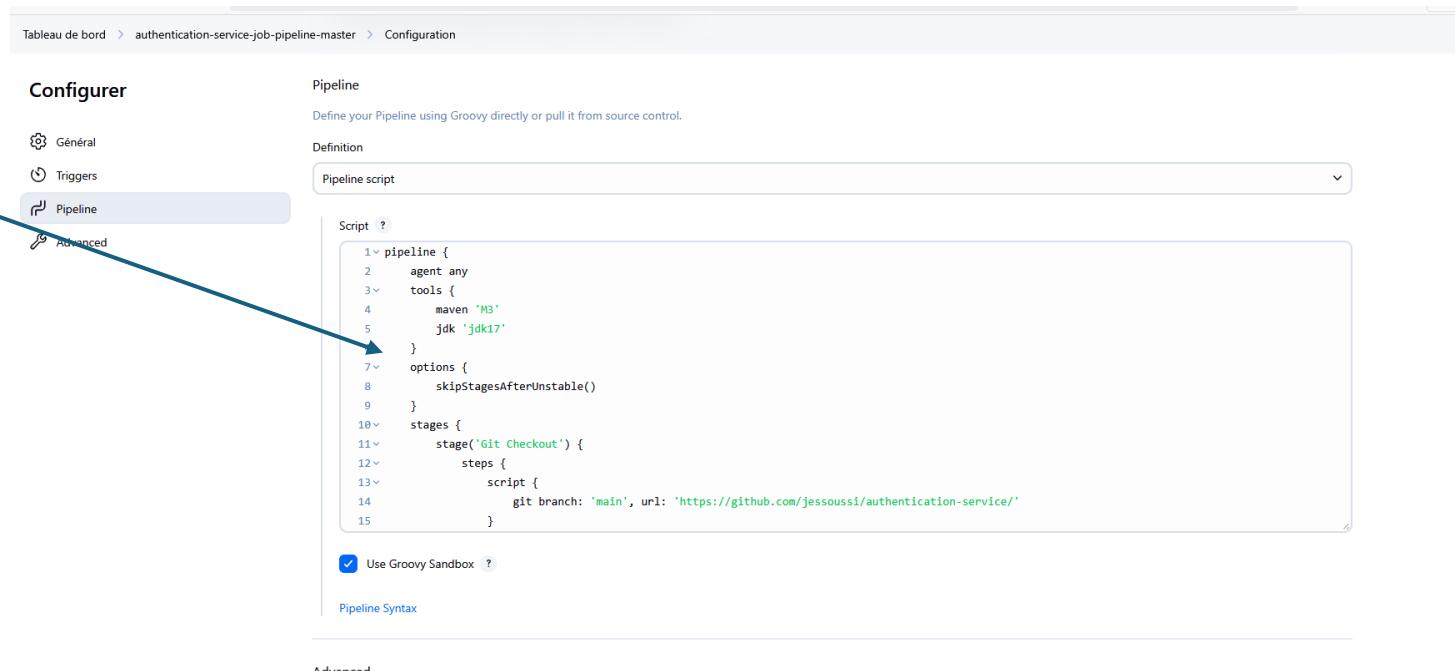
Restreindre où le projet peut être exécuté ?

Save Appliquer

# Job

## ○ Pipeline

- Déclarative à travers des pipelines
- Jenkinsfile



The screenshot shows the Jenkins Pipeline configuration page for a job named "authentication-service-job-pipeline-master". The "Pipeline" tab is selected in the navigation bar. The main area contains a "Pipeline script" input field with the following Groovy code:

```
1 v pipeline {
2   agent any
3   tools {
4     maven 'M3'
5     jdk 'jdk17'
6   }
7   options {
8     skipStagesAfterUnstable()
9   }
10  stages {
11    stage('Git Checkout') {
12      steps {
13        script {
14          git branch: 'main', url: 'https://github.com/jessoussi/authentication-service/'
15        }
16      }
17    }
18  }
19}
```

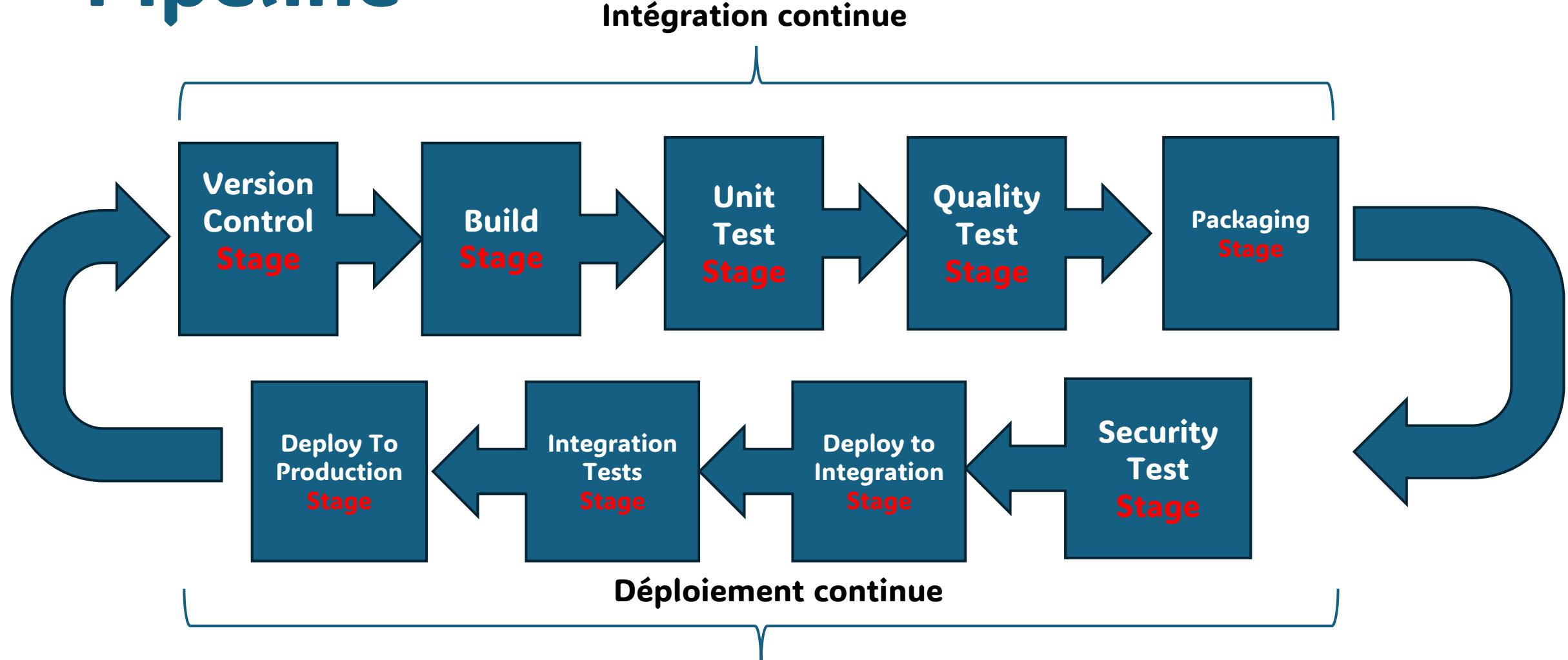
Below the script, there is a checkbox labeled "Use Groovy Sandbox" and a link to "Pipeline Syntax". At the bottom, there is an "Advanced" section.

# 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/>)

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

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

GROOVY | 

# 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 définis dans l'interface d'administration de Jenkins

## Configuration du système

 System Configurer les paramètres généraux et les chemins de fichiers.	 Tools Configurer les outils, leur localisation et les installeurs automatiques.	 Plugins Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins. <small>80</small>	 Nodes Ajouter, supprimer, contrôler et moniter les divers noeuds que Jenkins utilise pour exécuter les jobs.
--	--	---	---

## Sécurité

 Security Sécuriser Jenkins; définir qui est autorisé à accéder au système.	 Credentials Configure credentials	 Credential Providers Configure the credential providers and types	 Users Créer/supprimer/modifier les utilisateurs qui peuvent se logger sur ce serveur Jenkins
---	--	--	---

# Les concepts principaux

- Utilisation des credentials
- Liste des credentials

The screenshot shows two sections of the Jenkins 'Identifiants' (Credentials) page.

**Credentials**

T	P	Store ↓	Domain	ID	Name
git	user	System	(global)	kubectlcred	kubectlcred
git	user	System	(global)	sonarqube cred	sonarqube cred
git	user	System	(global)	github_credentials	jamel.essoussi@gmail.com/*****

**Stores scoped to Jenkins**

P	Store ↓	Domains
user	System	(global)

Icône: S M L

# 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 créé 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 et 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:



File d'attente des constructions

File d'attente des constructions vide

État du lanceur de compilations 0/2

Cloud Statistics

# 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

Nouveau Item

Saisissez un nom

Select an item type

 Construire un projet free-style  
Job legacy polyvalent qui récupère l'état depuis un outil de gestion de version au plus, exécute les étapes de build en série, suivie d'étapes post-construction telles que l'archivage d'artefacts et l'envoi de notifications par e-mail.

 Pipeline  
Organise des activités de longue durée qui peuvent s'étendre sur plusieurs agents de construction. Adapté pour la création des pipelines (anciennement connues comme workflows) et/ou pour organiser des activités complexes qui ne s'adaptent pas facilement à des tâches de type libre.

 Construire un projet multi-configuration  
Adapté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, etc.

# 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:

Configure

Pipeline

Definition

Pipeline script

Script ?

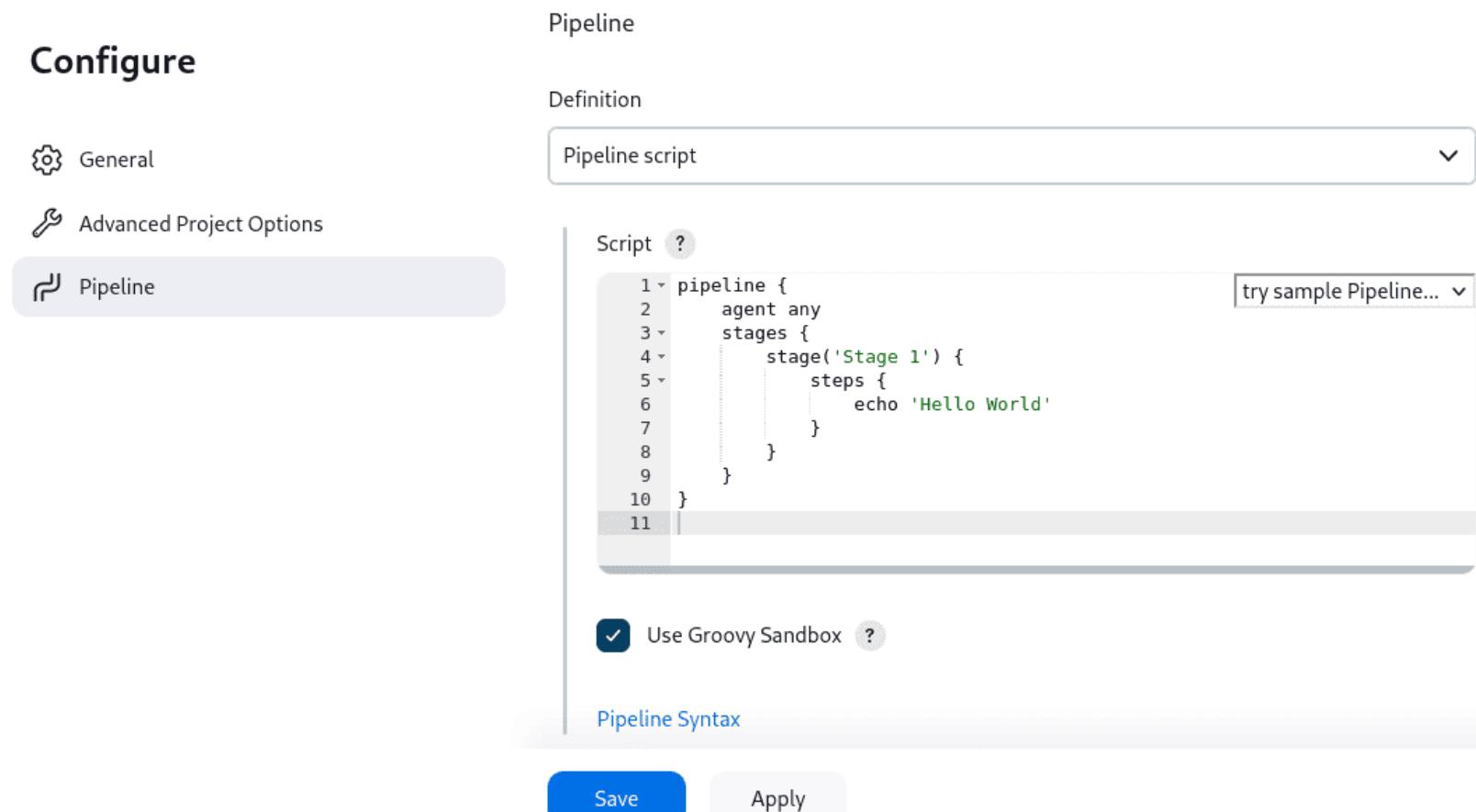
```
1 pipeline {  
2     agent any  
3     stages {  
4         stage('Stage 1') {  
5             steps {  
6                 echo 'Hello World'  
7             }  
8         }  
9     }  
10 }  
11
```

try sample Pipeline... ▾

Use Groovy Sandbox ?

Pipeline Syntax

Save Apply



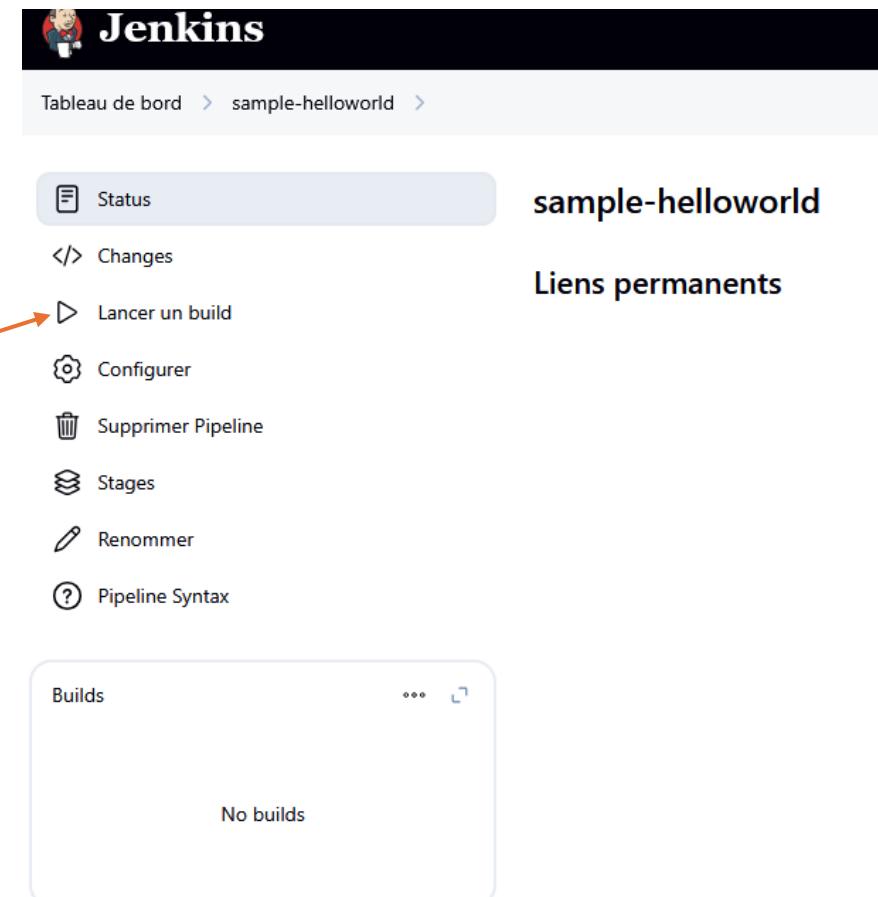
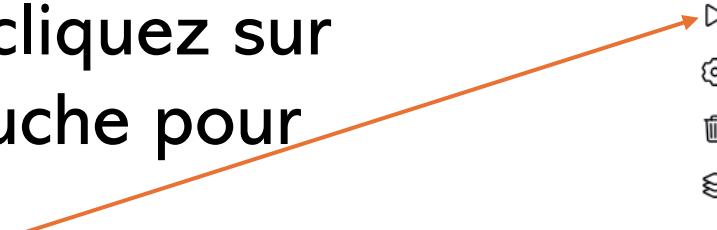
The screenshot shows the Jenkins Pipeline configuration screen. On the left, there's a sidebar with 'Configure' and three tabs: 'General', 'Advanced Project Options', and 'Pipeline'. The 'Pipeline' tab is selected and highlighted with a grey background. The main area is titled 'Pipeline' and contains a 'Definition' section. Under 'Definition', it says 'Pipeline script' and shows a code editor with a Groovy script. The script defines a pipeline with one stage named 'Stage 1' that echoes 'Hello World'. Below the code editor is a checkbox labeled 'Use Groovy Sandbox' which is checked. At the bottom of the screen are two buttons: 'Save' and 'Apply'.

# 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.



The screenshot shows the Jenkins interface for a Pipeline project named "sample-helloworld". The left sidebar contains the following items:

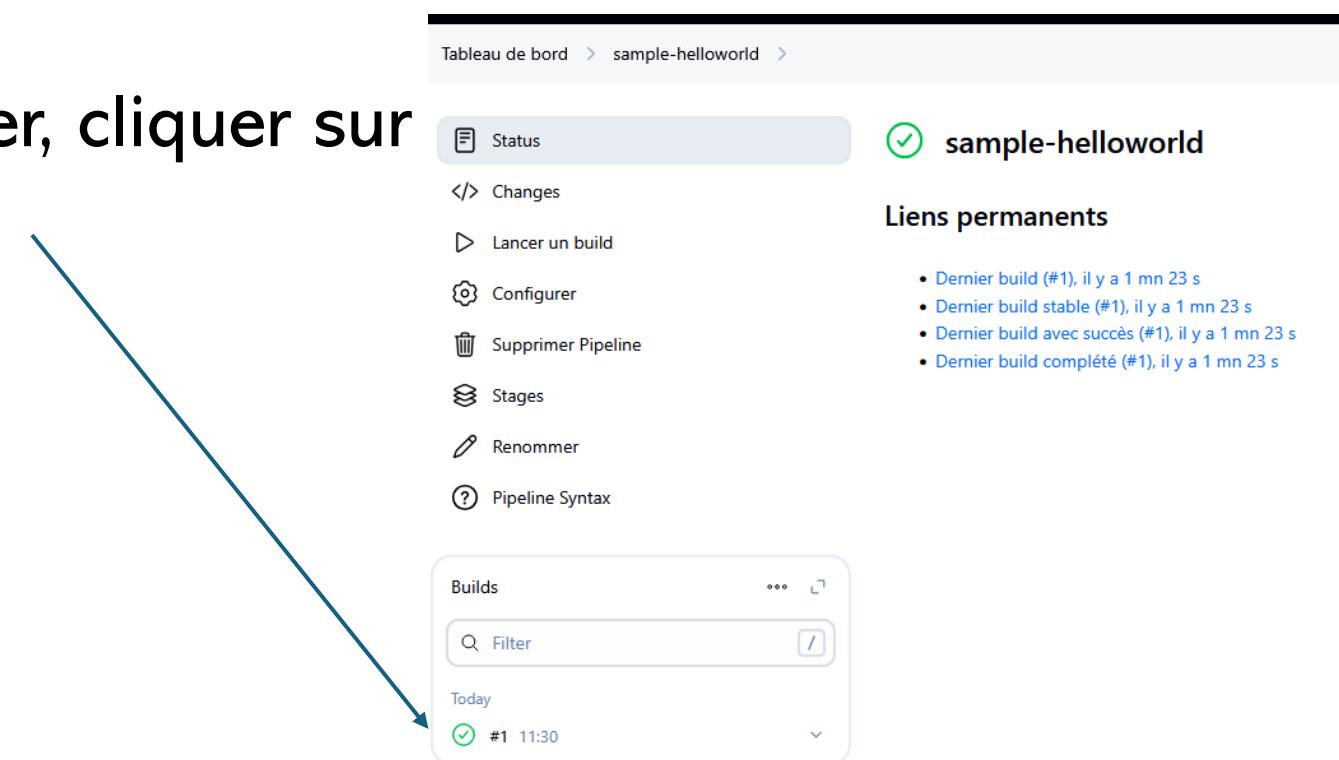
- Status (highlighted with a blue bar)
- </> Changes
- ▷ Lancer un build (with a red arrow pointing to it)
- ⚙️ Configurer
- trash Supprimer Pipeline
- 📦 Stages
- ✍️ Renommer
- ⓘ Pipeline Syntax

The main content area has a header "sample-helloworld" and "Liens permanents". Below that, there is a section titled "Builds" with the message "No builds".

# Pipeline

- Création à travers l'interface classique:

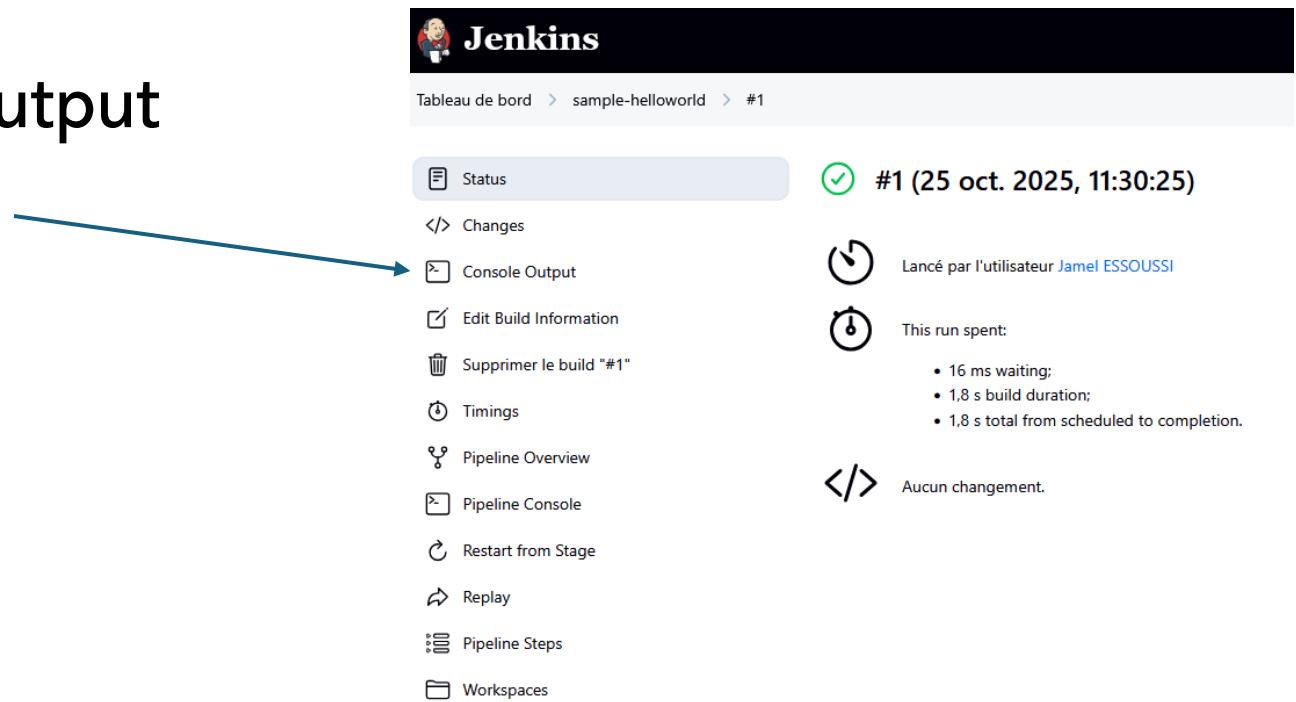
10 – Une fois le build lancé, cliquer sur le dernier numéro de build



# 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



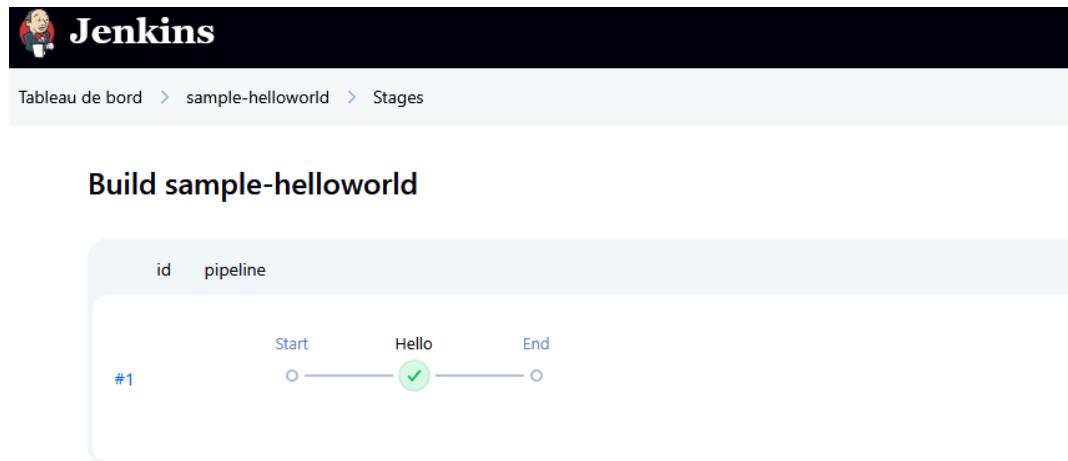
The screenshot shows the Jenkins interface for a pipeline named "sample-helloworld". The build number is "#1". The left sidebar lists various options: Status, Changes, Console Output (which is selected and highlighted in grey), Edit Build Information, Supprimer le build "#1", Timings, Pipeline Overview, Pipeline Console, Restart from Stage, Replay, Pipeline Steps, and Workspaces. The right panel is titled "Sortie de la console" (Console Output) and displays the build log. The log starts with "Démarré par l'utilisateur Jamel ESSOUSSI" and ends with "Finished: SUCCESS". A red box highlights the line "[Pipeline] echo Hello World".

```
Démarré par l'utilisateur Jamel ESSOUSSI
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\sample-helloworld
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] echo
Hello World
[Pipeline]
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

# Pipeline

- Crédit à travers l'interface classique:

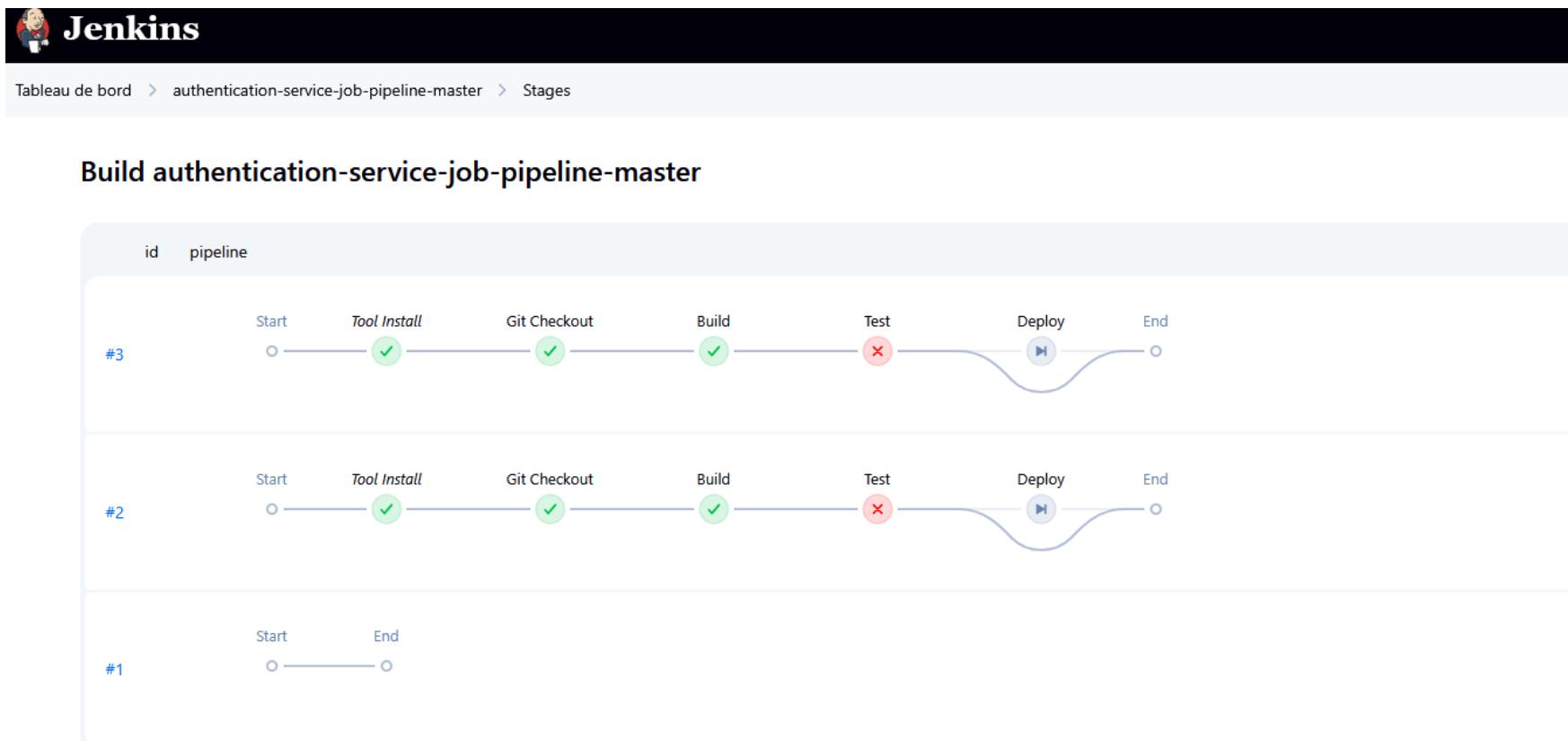
## 12 – Visualiser les stages



The screenshot shows the Jenkins Builds interface. At the top, there's a "Builds" header with a "Filter" input field. Below it, the status "Today" is shown, with a green checkmark next to "#1 11:30", indicating a successful build. A large blue arrow points from the Jenkins Pipeline interface above to this "Builds" section.

# Pipeline

- Notre exemple de TD:



# Pipeline

- Notre exemple de TD:

```
1 pipeline {  
2     agent any  
3     tools {  
4         maven 'M3'  
5         jdk 'jdk17'  
6     }  
7     options {  
8         skipStagesAfterUnstable()  
9     }  
10    stages {  
11        stage('Git Checkout') {  
12            steps {  
13                script {  
14                    git branch: 'main', url: 'https://github.com/jessoussi/authentication-service/'  
15                }  
16            }  
17        }  
18        stage ('Build') {  
19            steps {  
20                bat 'mvn clean compile'  
21            }  
22        }  
23        stage ('Scan SonarQube') {  
24            steps {  
25                withSonarQubeEnv(installationName:'SonarQube'){  
26                    bat 'mvn clean install sonar:sonar'  
27                }  
28            }  
29        }  
30        stage ('Test') [  
31            steps {  
32                bat 'mvn -Dmaven.test.failure.ignore=true clean install'  
33            }  
34        ]  
35    }  
36    stage('Deploy') {  
37        steps {  
38            echo "Deployment application"  
39        }  
40    }  
41}
```

# Plan du cours

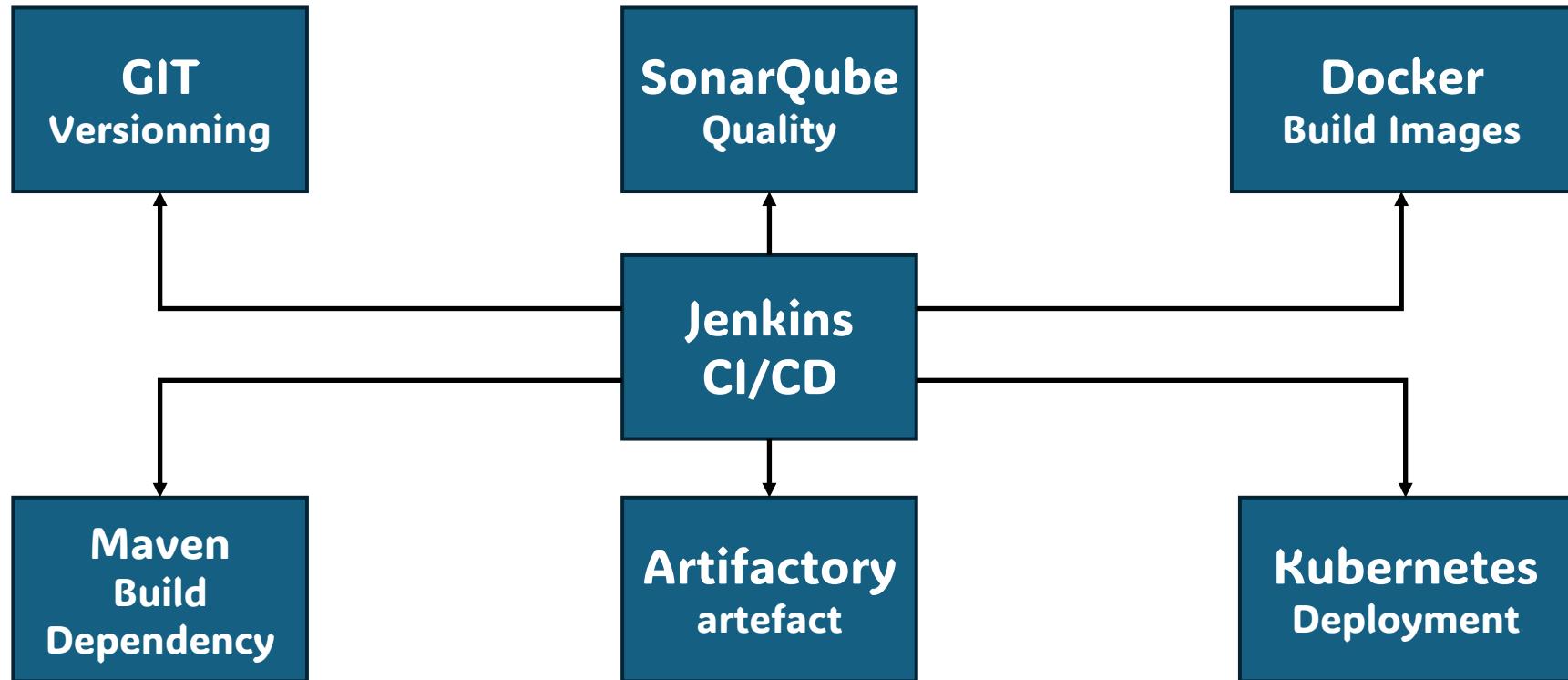
I. Introduction

II. Job & Pipeline

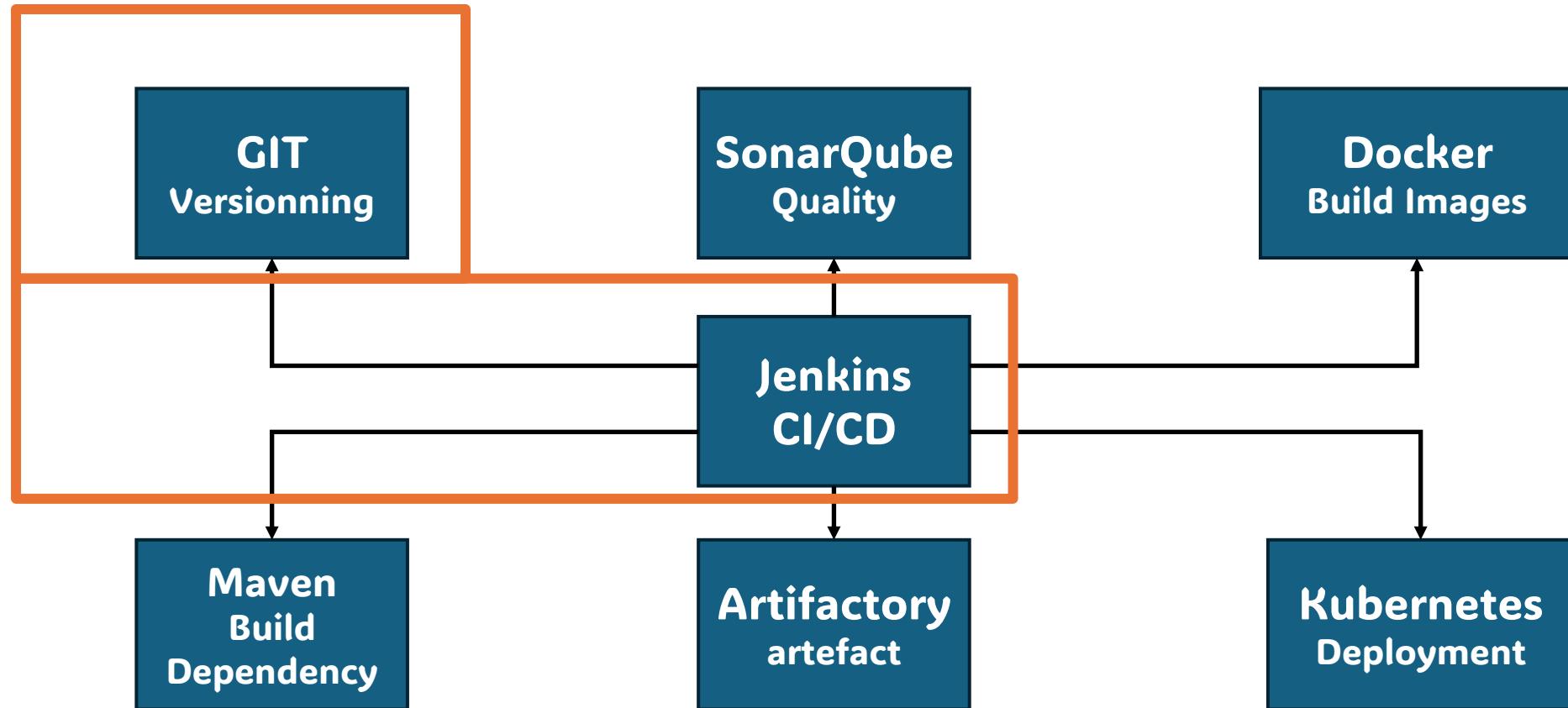
**III. Intégration avec les autres outils**

IV. Les bonnes pratiques

# Intégration avec les autres outils



# GIT



# GIT

## ○ A travers le plugin GIT de Jenkins

The screenshot shows the Jenkins Plugins management interface. On the left, a sidebar lists navigation options: Mises à jour, Plugins disponibles, Plugins installés (selected), Paramètres avancés, and Progression des téléchargements. The number 80 is displayed above the Plugins disponibles and Paramètres avancés buttons. The main area has a search bar at the top containing the text "git". Below it, a table lists installed plugins:

Nom	Version	Description	Activé
Git	5.7.0	This plugin integrates Git with Jenkins. Report an issue with this plugin	<input checked="" type="checkbox"/> <span>(green)</span>
Git client	6.1.3	Utility plugin for Git support in Jenkins Report an issue with this plugin	<input checked="" type="checkbox"/> <span>(green)</span> Warning: The currently installed plugin version may not be safe to use. Please review the following security notices: • <a href="#">File system information disclosure vulnerability</a>
GitHub	1.43.0	This plugin integrates GitHub to Jenkins. Report an issue with this plugin	<input checked="" type="checkbox"/> <span>(green)</span>

# GIT

- A travers le plugin GIT de Jenkins

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

# 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

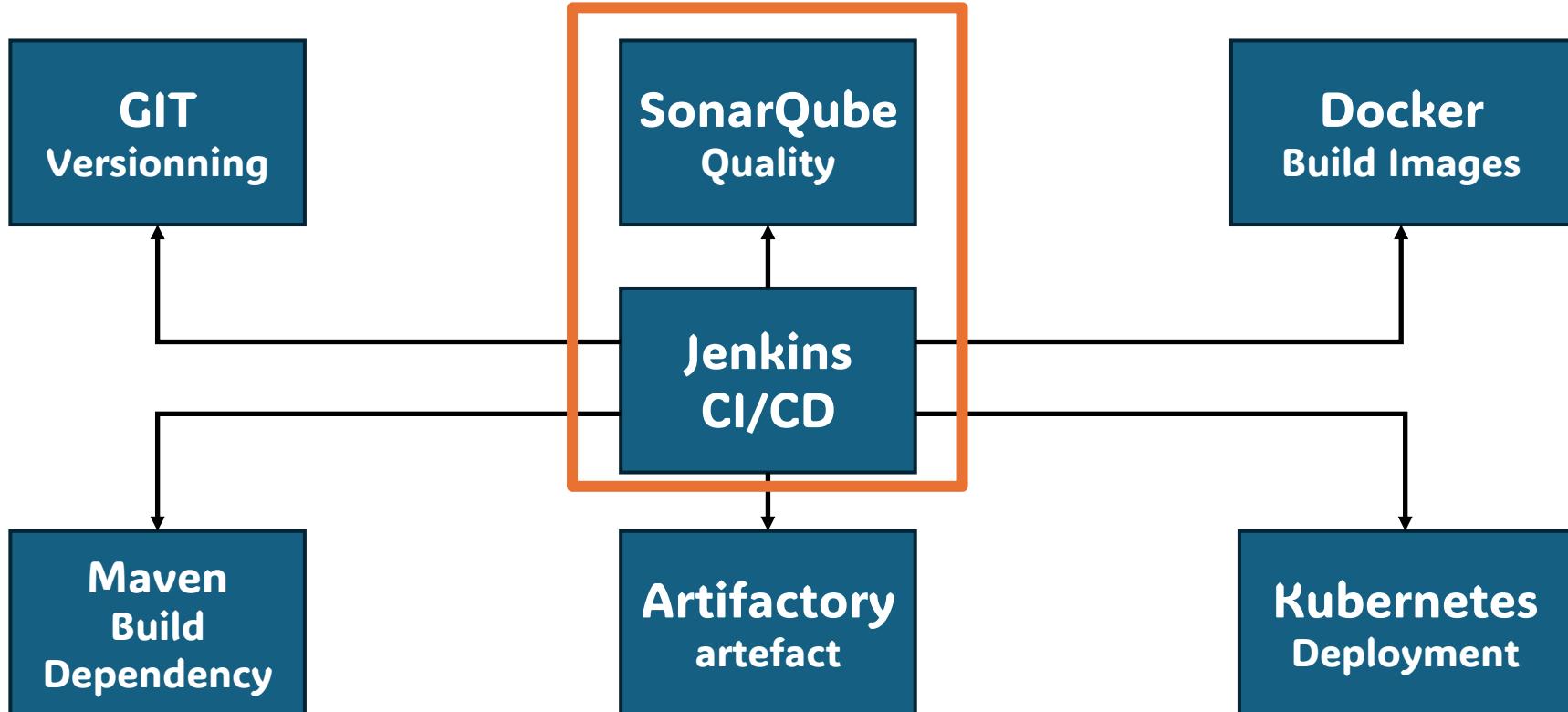
# SonarQube

- 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



# SonarQube

## ○ Installation de SonarQube

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

**Téléchargement**

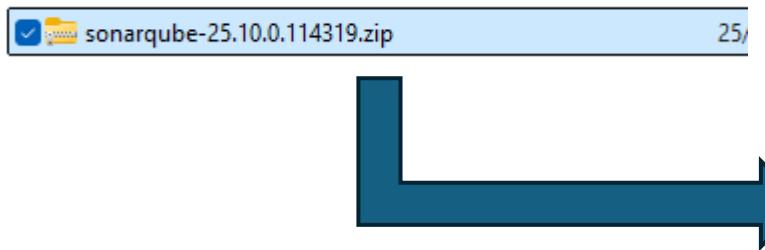
Visit the SonarQube Update Hub: You'll find resources for a seamless 2025.1 LTA update!

Community Build	Developer edition	Enterprise edition	Data Center edition
Release 25.10.0.11431 Free and open source for productivity & code quality <a href="#">Download for free</a>	2025 Release 5   September Essential capabilities for small teams & businesses <a href="#">Try for free</a> <a href="#">Download only</a>	2025 Release 5   September Designed to meet Enterprise-level requirements <a href="#">Try for free</a> <a href="#">Download only</a>	2025 Release 5   September For high availability, scalability, & performance <a href="#">Try for free</a> <a href="#">Download only</a>
<b>Includes:</b> <ul style="list-style-type: none"><li>Static code analysis for 21 languages and frameworks</li><li>SonarQube runs in a FIPS</li></ul>	<b>Community Build, plus:</b> <ul style="list-style-type: none"><li>Additional languages: C, C++, Obj-C, Dart/Flutter, Swift, ABAP, T-SQL, PL/SQL, YAML, JSON, Ansible, GitHub Actions</li></ul>	<b>Developer edition, plus:</b> <ul style="list-style-type: none"><li>Additional languages: Apex, COBOL, JCL, PL/I, RPG and VB6</li><li>Unlimited integrations</li></ul>	<b>Enterprise edition, plus:</b> <ul style="list-style-type: none"><li>Autoscaling in a Kubernetes cluster</li><li>Component redundancy</li></ul>

Jenkins CI/CD

# SonarQube

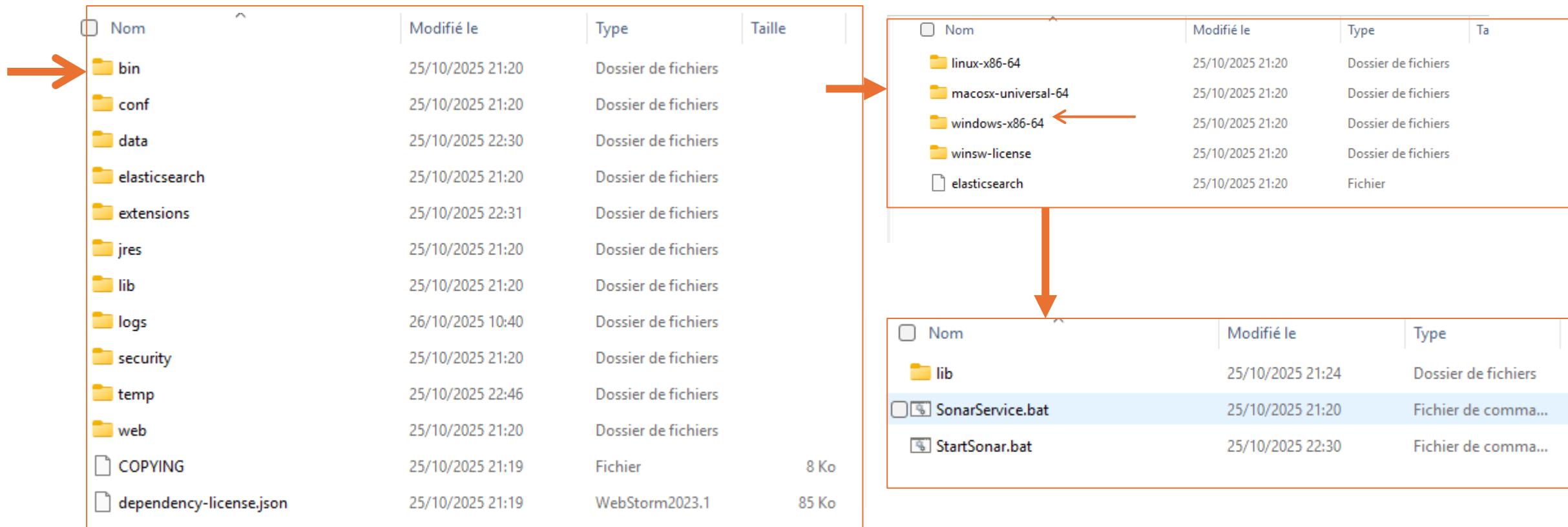
- Dézipper le package téléchargé:



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

# SonarQube

## ○ Lancer le script de démarrage:



# 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/>

The screenshot shows the SonarQube web interface at <http://localhost:9000/projects>. The main navigation bar includes links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. A sidebar on the left contains sections for My Favorites (All), Filters, Quality Gate, Security, Reliability, and Reliability. The main content area displays the analysis results for the project 'authentication-service' (Public). The analysis was performed 13 hours ago on 260 Lines of Code in XML and Java. The quality gate status is 'Passed'. Key metrics shown include 0 info issues, 0 low issues, 7 medium issues, 0 high issues, and 0 blocker issues. Coverage is at 0.0% and Duplications are at 0.0%. The overall status is 'Passed'.

# SonarQube

## ○ Branchement avec Jenkins

Aller sur la page d'administration de Jenkins

The screenshot shows the SonarQube dashboard with a sidebar on the right containing various links:

- + Nouveau Item
- Historique des constructions
- Relations entre les builds
- Vérifier les empreintes numériques
- Administrer Jenkins** (highlighted with an orange arrow)
- Mes vues

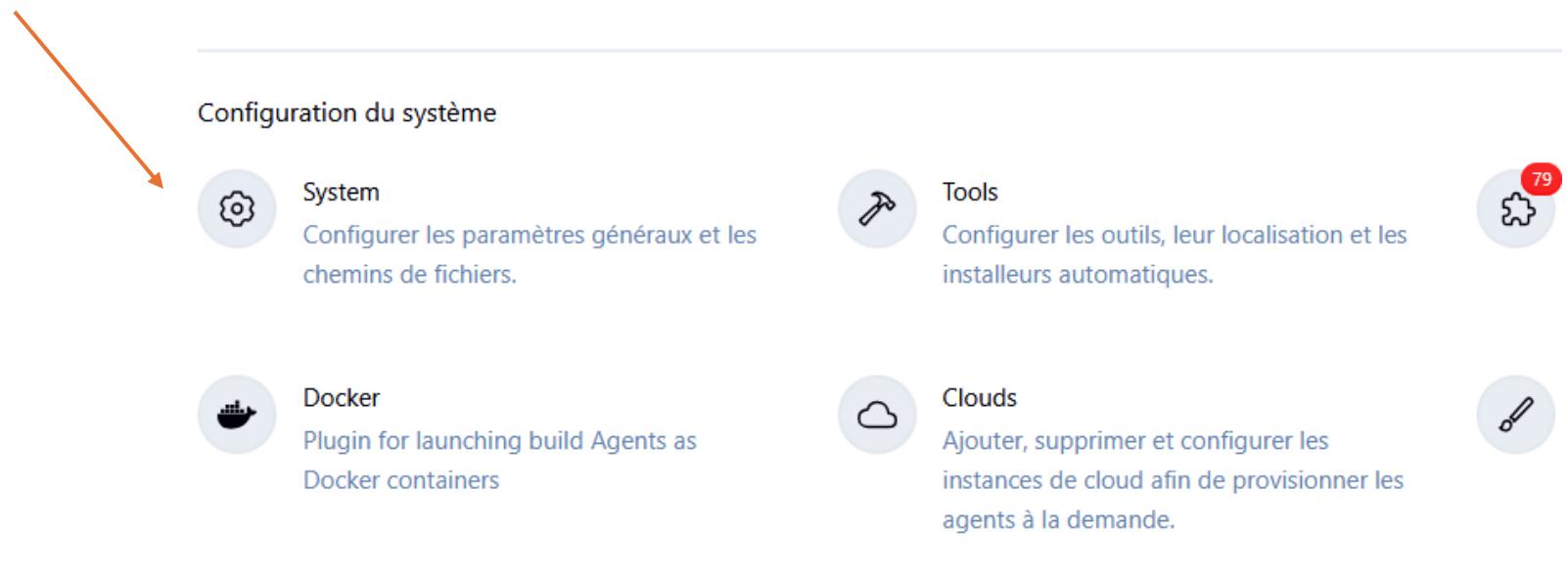
Below the sidebar, there are three collapsed sections:

- File d'attente des constructions** (collapsed)
  - File d'attente des constructions vide
- État du lanceur de compilations** (collapsed)
  - 0/2
- Cloud Statistics** (collapsed)

# SonarQube

## ○ Branchement avec Jenkins

Aller sur la page système



The screenshot shows the Jenkins System Configuration page. At the top, it says "Configuration du système". Below that are four main sections: "System", "Tools", "Docker", and "Clouds". Each section has a description and a corresponding icon. A red arrow points from the text "Aller sur la page système" to the "System" section.

Section	Description	Icon
System	Configurer les paramètres généraux et les chemins de fichiers.	gear icon
Tools	Configurer les outils, leur localisation et les installeurs automatiques.	wrench icon
Docker	Plugin for launching build Agents as Docker containers	Docker icon
Clouds	Ajouter, supprimer et configurer les instances de cloud afin de provisionner les agents à la demande.	cloud icon

# SonarQube

## ○ Branchement avec Jenkins

### Credential: Token Sonar



Il suffit que l'administrateur sera à même d'injecter une configuration de serveur SonarQube en tant que variables d'environnement dans la construction.

Environnement variables

Installations de SonarQube

Liste des installations de SonarQube

Nom  
SonarQube

URL du serveur  
Par défaut à <http://localhost:9000>

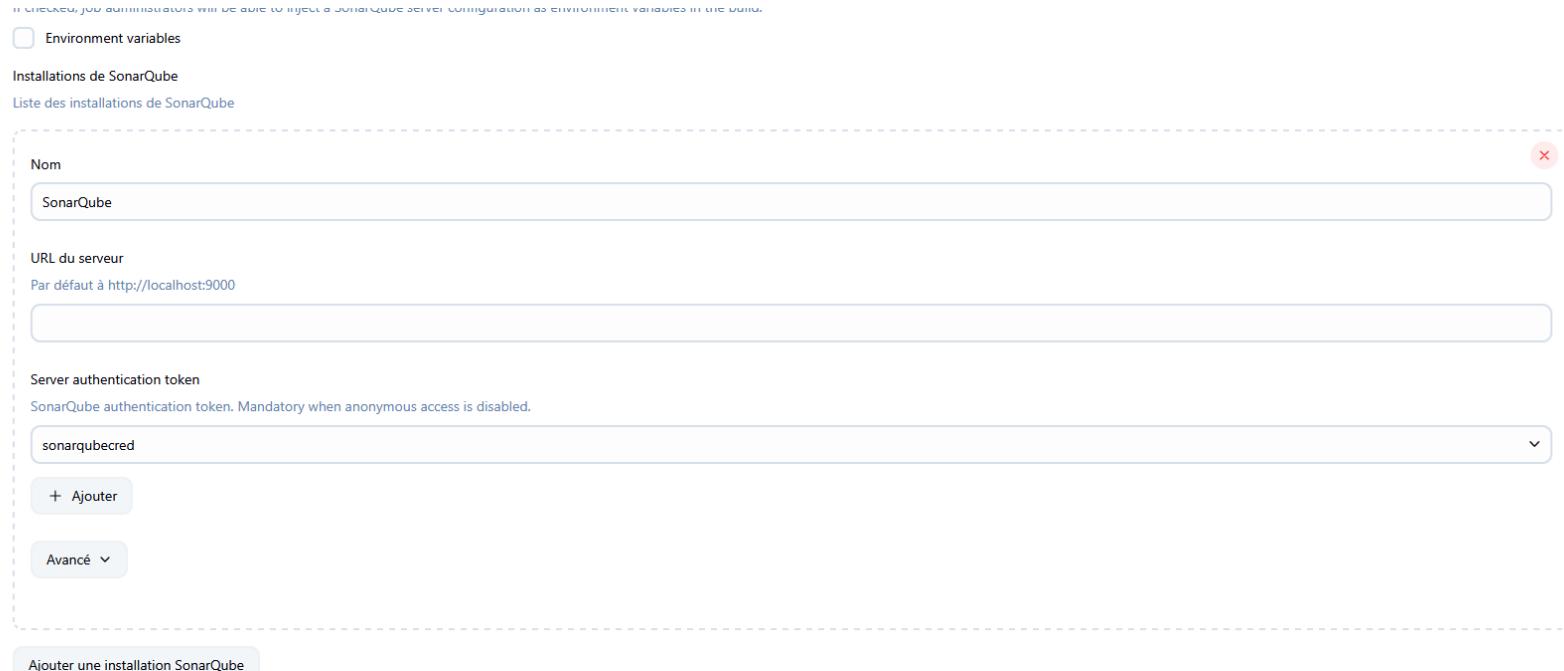
Server authentication token  
SonarQube authentication token. Obligatoire lorsque l'accès anonyme est désactivé.

sonarqubecred

+ Ajouter

Avancé ▾

Ajouter une installation SonarQube



# SonarQube

## ○ Générer un token sur SonarQube

### Credential: Token Sonar

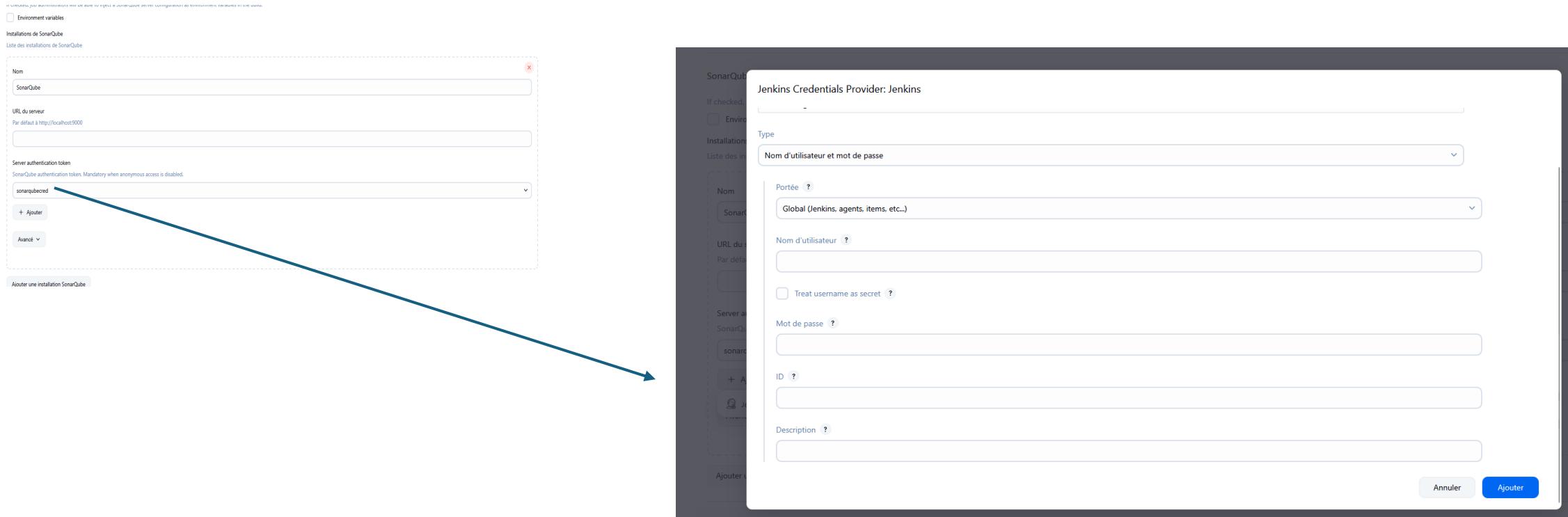


The screenshot shows the SonarQube account security page at <http://localhost:9000/account/security>. The page includes a warning about using an embedded database and navigation links for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The 'Administrator' role is selected. The 'Security' tab is active, showing a 'Generate Tokens' section with fields for Name, Type, and Expires in (set to 30 days), and a 'Generate' button. Below this is a table of existing tokens, including one for 'jenkins'. At the bottom, there's a 'Enter a new password' section with fields for Old Password\*, Password\*, and Confirm Password\*.

Name	Type	Project	Last use	Created	Expiration
jenkins	User		14 hours ago	October 25, 2025	November 24, 2025

# SonarQube

## ○ Renseigner le token dans Jenkins



# SonarQube

## ○ Branchement avec Jenkins

Aller sur la page d'administration de Jenkins

Tableau de bord >

- + Nouveau Item
- 🕒 Historique des constructions
- 🕒 Relations entre les builds
- 🔍 Vérifier les empreintes numériques
- ⚙️ Administrer Jenkins
- 👁 Mes vues

File d'attente des constructions

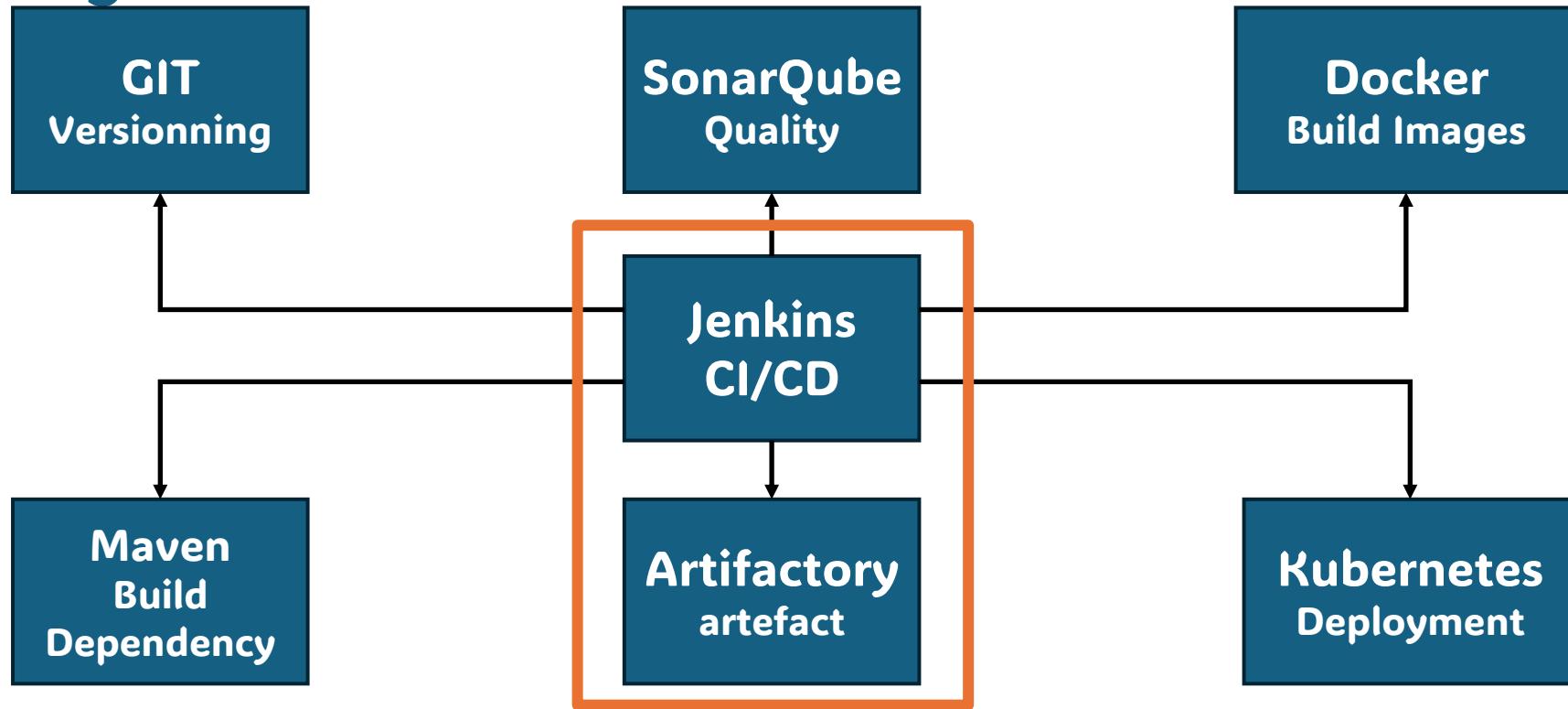
File d'attente des constructions vide

État du lanceur de compilations 0/2

Cloud Statistics

# 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

Tableau de bord > Administrer Jenkins > Plugins

## Plugins

Search bar: jfrog

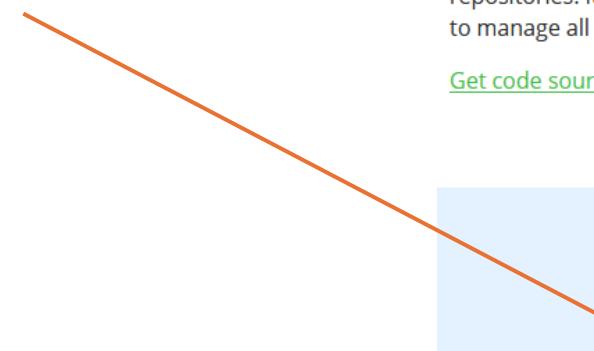
Mises à jour	80	Installer	Nom ↓	Publié
<input checked="" type="checkbox"/>		JFrog 1.5.10	Développement .NET Maven npm Déploiement docker	Il y a 1 mo. 26 j
		Artifactory Client API 2.19.0-116.vc6e7efff0e6b_	This plugin provides the <a href="#">Artifactory Client</a> (v2.19.0) for other plugins.	Il y a 3 mo. 2 j
		Artifact Manager Artifactory 252.v944a_69ea_4873	A Jenkins plugin to keep artifacts and Pipeline stashes in JFrog Artifactory	Il y a 3 mo. 0 j
		Jobcacher Artifactory Storage Extension 198.vd6fa_ec5e1028	Extension of jobcacher plugin that add JFrog artifactory for item storage.	Il y a 3 mo. 2 j

# Artifactory

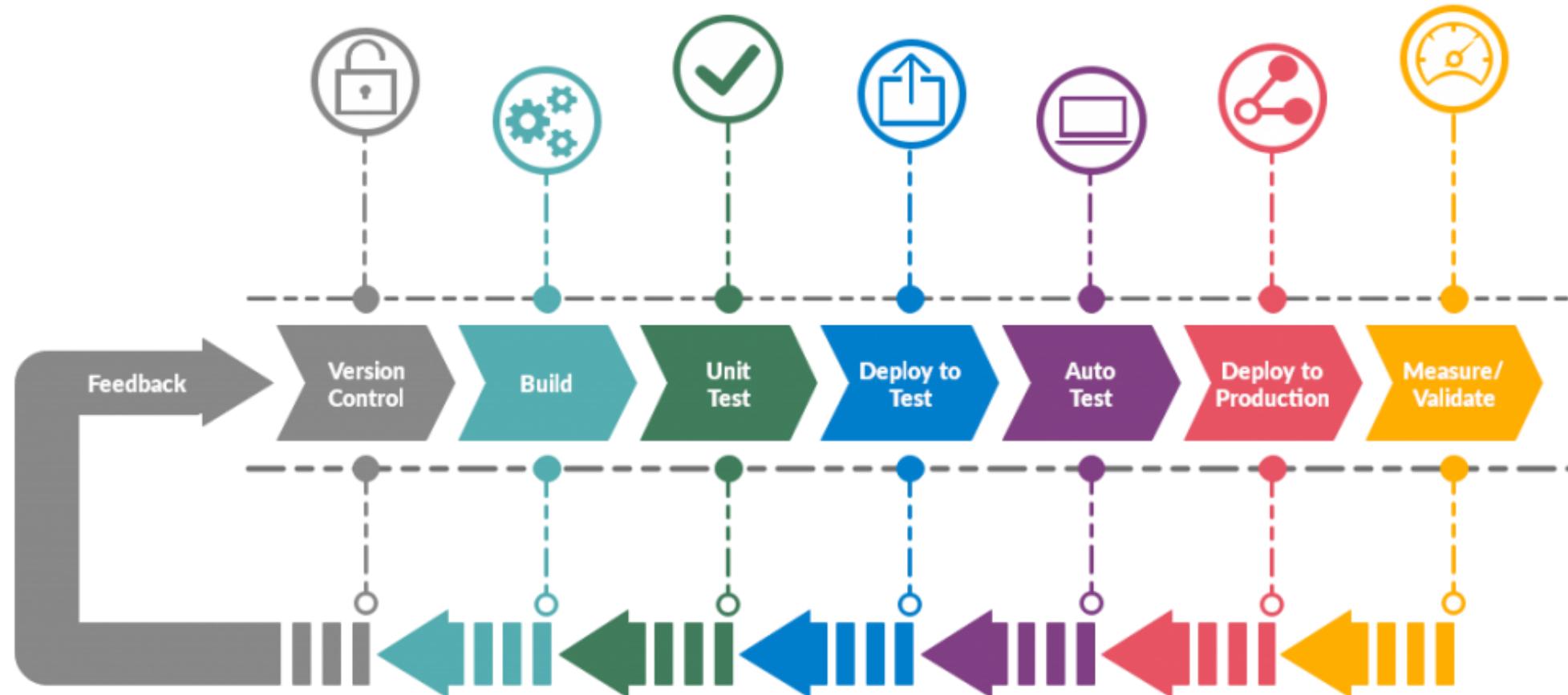
## ○ Installation



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