



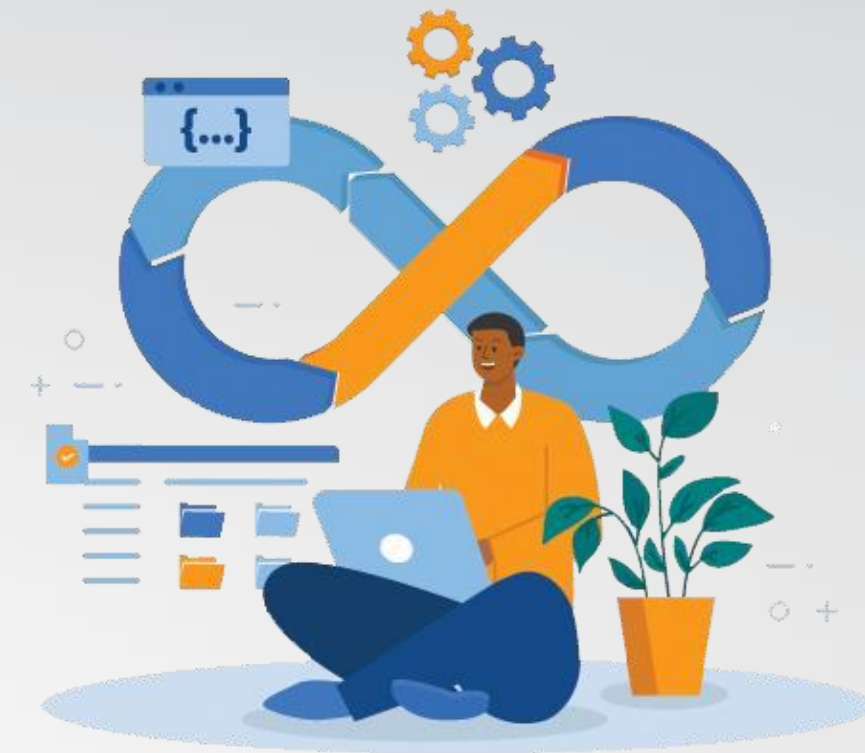
DevOps

Chapitre 2 : Intégration continue Jenkins

ESPRIT –UP ASI (Architecture des Systèmes d'Information)
BureauE204

► Plan du cours

- La problématique du développement logiciel traditionnel
- Intégration continue - Définition
- Intégration continue – Avantages
- Jenkins – Outil d'automatisation CI/CD
- Jenkins – Plugins
- Jenkins – Installation et configuration
- Jenkins – Job
- Travail à faire



▶ La problématique du développement logiciel traditionnel



🚨 Le cauchemar de l'intégration manuelle

Avant Jenkins et l'automatisation, les équipes de développement vivaient ce scénario typique :

```
# JOUR DU "MERGE HELL" - Processus manuel
# Développeur 1 :
git pull origin main
mvn clean compile
# → "Ça compile sur ma machine..."

# Développeur 2 :
git pull origin main
mvn clean compile
# → ERREUR: ClassNotFoundException - dépendance manquante

# Développeur 3 :
git pull origin main
npm install
npm run build
# → ERREUR: Version de Node.js incompatible

# Lead Developer :
# Passera les 4 heures suivantes à résoudre les conflits,
# configurations d'environnement et problèmes de dépendances
```



▶ La problématique du développement logiciel traditionnel



 Les problèmes concrets rencontrés :

1. "Ça marche sur ma machine" syndrome

- Environnements de développement disparates
- Versions de logiciels différentes entre développeurs
- Configurations système non standardisées

2. Intégration tardive et douloureuse

- Merges manuels complexes et error-prone
- Détection très tardive des conflits d'intégration
- "Merge days" pouvant prendre plusieurs jours

3. Feedback lent et inefficace

- Découverte des bugs seulement en staging/production
- Temps de correction exponentiel
- Développeurs bloqués en attendant les retours

4. Processus de build non reproductible

- Builds successifs donnant des résultats différents
- Artefacts non reproductibles
- Déploiements unpredictables

5. Manque de visibilité totale

- Impossible de savoir l'état exact du code
 - Historique des builds non centralisé
 - Métriques de qualité absentes



La problématique du développement logiciel traditionnel

Impact business réel :



Time-to-market :
Semaines → Mois



Qualité :
Bugs récurrents en production



Morale : Équipes frustrées par les
processus manuels



Coût : Développeurs passant plus de temps à
intégrer qu'à développer



Intégration continue - Définition



 Le paradigme CI : "Intégrer tôt, intégrer souvent"

L'Intégration Continue est une pratique de développement logiciel où les membres d'une équipe intègrent leur travail fréquemment, généralement chaque personne intègre au moins quotidiennement.

1. Repository de code unique
2. Automatisation du build
3. Build auto-testant et rapide
4. Environnements reproductibles



Intégration continue - Avantages



Pour les développeurs :

- Feedback immédiat sur le code
- Réduction du temps de debugging
- Confiance dans les merges
- Environnements standardisés

Pour le business :

- Réduction des risques
- Livraisons plus fréquentes
- Coûts de développement réduits
- Qualité améliorée

Pour les ops :

- Processus reproductibles
- Artefacts fiables
- Déploiements prévisibles



▶ Jenkins – Outil d'automatisation CI/CD

 Jenkins : Le robot qui fait le travail à votre place

- Jenkins est l'outil qui matérialise les principes du CI en les automatisant complètement. C'est un serveur d'automatisation open source qui implémente le CI/CD.
- Il automatise les étapes de construction, de tests et de déploiement, facilitant ainsi l'intégration continue et la livraison continue.
- Il permet aux développeurs d'intégrer rapidement des modifications au projet, accélérant ainsi l'amélioration continue du produit.



Jenkins



Jenkins – Outil d'automatisation CI/CD

 Pourquoi Jenkins spécifiquement ?

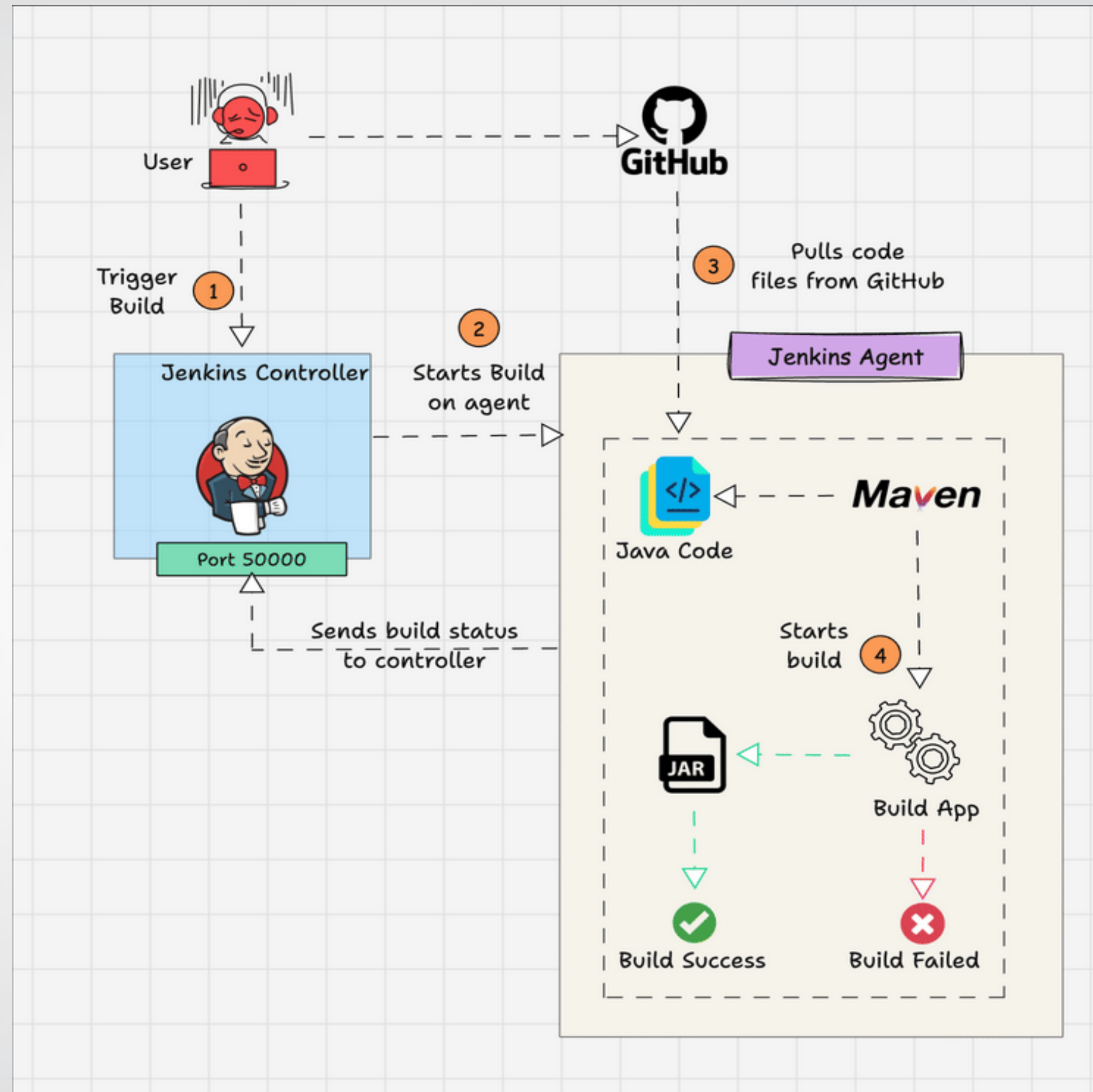
- Open source : Gratuit et communauté active
- Extensible : Plus de 1500 plugins
- Flexible : Supporte tous les langages et outils
- Scalable : Architecture maître/agents
- Enterprise-ready : Utilisé par les plus grandes companies

Jenkins n'est pas juste un outil - c'est l'automatisation de votre processus de développement entier.

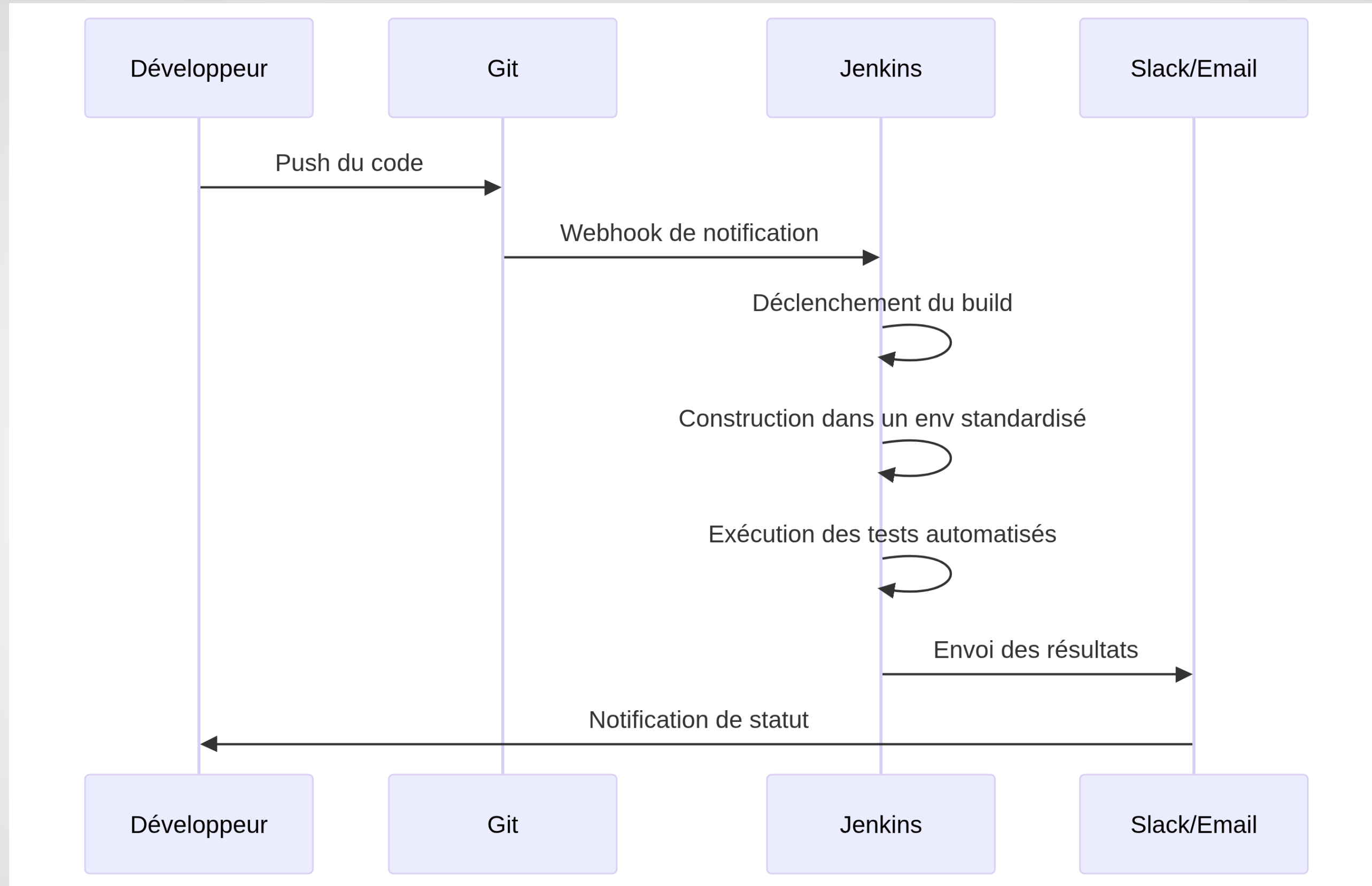


► Jenkins – Outil d'automatisation CI/CD

- 70%+ de réduction du temps de livraison
- 85%+ de réduction des erreurs de déploiement
- 50%+ de gain de temps pour les développeurs



► Jenkins – Outil d'automatisation CI/CD



► Jenkins – Outil d'automatisation CI/CD

🤖 Comment Jenkins résout-il chaque problème ?

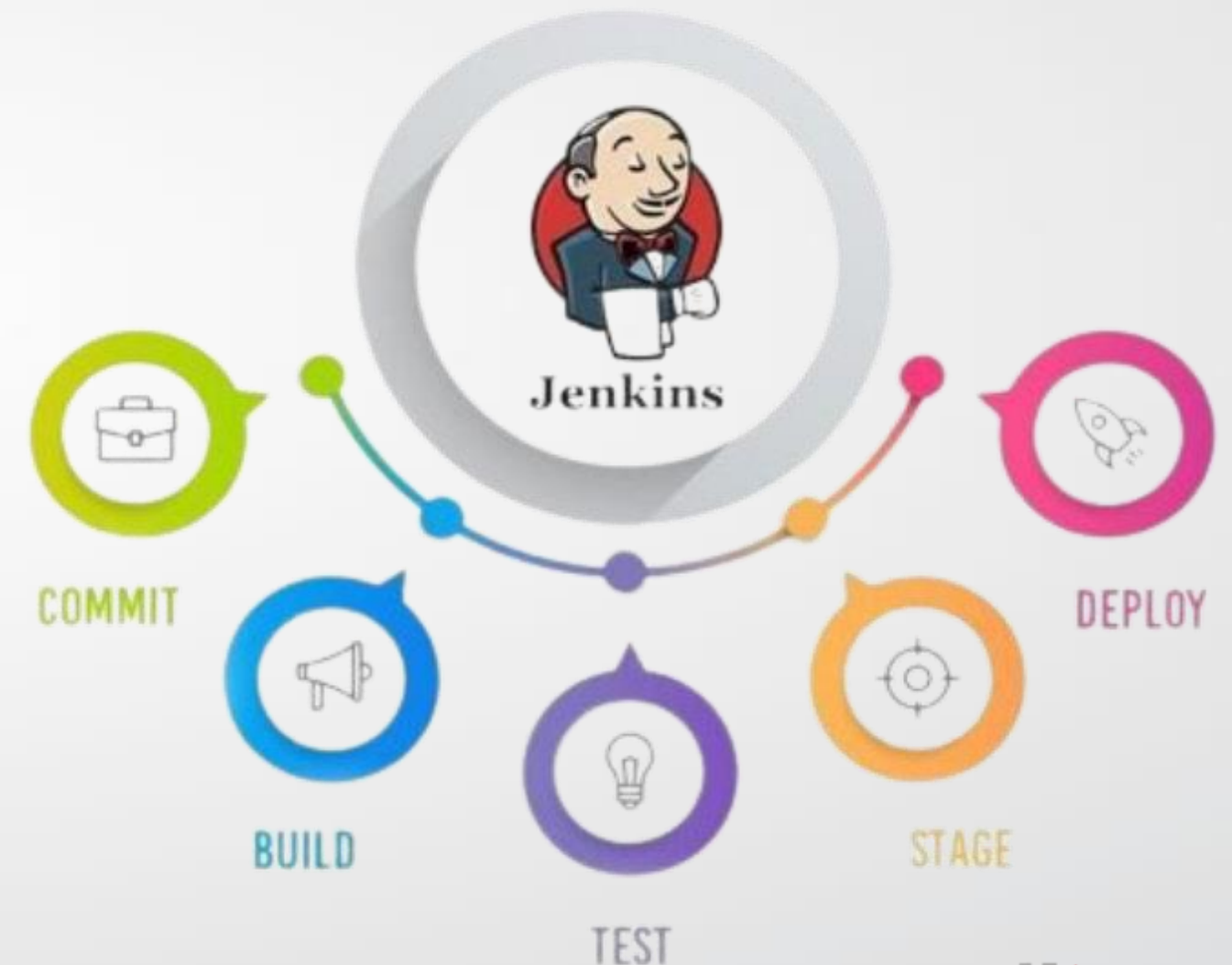
Problème 1 : Ça marche sur ma machine --> Environnements standardisés

Problème 2 : Intégration tardive --> Intégration à chaque changement

Problème 3 : Feedback lent --> Feedback immédiat

Problème 4 : Processus non reproductible --> Pipeline as Code

Problème 5 : Manque de visibilité --> Dashboard centralisé



Jenkins – Outil d'automatisation CI/CD



Concepts fondamentaux

- Job/Pipeline : C'est la définition d'une tâche automatisée, décrite par du code ou une configuration. Il permet d'orchestrer les différentes étapes du processus d'intégration et de déploiement.
- Build : C'est l'exécution concrète d'un pipeline ou d'un job, qui produit un résultat (succès ou échec). Chaque build est historisé et peut être analysé à travers ses logs.
- Agent/Node : C'est la machine (physique ou virtuelle) sur laquelle Jenkins exécute les builds. Elle peut être locale (maître) ou distante (esclave) selon l'architecture.
- Plugin : C'est une extension qui ajoute de nouvelles fonctionnalités ou intégrations à Jenkins (par ex. Git, Docker, Kubernetes). Grâce aux plugins, Jenkins devient hautement personnalisable et adaptable aux besoins du projet.



Jenkins – Plugins



Une variété de plugins en tout genre :

- Plugins de constructions (Maven Integration, Pipeline Integration),
- Plugins d'analyse de code (Sonargraph Integration, SonarQube Scanner),
- Plugins de gestion de sources (Git plugin),
- Plugins de gestion de configurations (Prometheus metrics), de VM, de
- conteneurs (Docker Pipeline),
- ...

Base de plugins très importante (>1500 plugins)



Jenkins – Installation et configuration



Installation des outils pour le build d'un projet Maven

JDK, MAVEN et GIT

Installation et configuration de Jenkins

JENKINS

(différents types d'installation: .war, distribution linux, conteneurs, ...)

Installation de Plugins + Configuration d'outils dans Jenkins

PLUGINS



► Jenkins – Installation et configuration

- 1 ère étape - Installation JDK 17
 - Vérifie si le JDK est déjà installé à partir l'exécution de la commande `java -version`.

```
vagrant@vagrant:~$ java -version
```

- Dans la machine `-bash: java: command not found` exécuter les commandes suivantes: `sudo apt update` (Pour la mise à jour)

```
vagrant@vagrant:~$ sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,012 kB]
19% [5 Packages 124 kB/1,012 kB 12%]
```



► Jenkins – Installation et configuration



- `sudo apt install openjdk-17-jdk -y`

```
vagrant@vagrant:~$ sudo apt install openjdk-11-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

- Définir la variable d'environnement JAVA_HOME:

Ouvrir le fichier de configuration système `/etc/environment` et créer la variable

d'environnement: `vagrant@vagrant:~$ sudo nano /etc/environment`

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:$JAVA_HOME/bin"
JAVA_HOME="/usr/lib/jvm/java-11-openjdk-amd64"
```

Appliquer les modifications apportées avec la commande

```
vagrant@vagrant:~$ source /etc/environment
```



► Jenkins – Installation et configuration

- 2 ème étape - Installation Maven
 - Exécuter les commandes nécessaires pour installer une version stable de Maven
 - Configurer la variable d'environnement M2_HOME (Les memes étapes du JDK)
 - Pour vérifier que le Maven est bien installé, exécuter la commandes suivantes:

```
vagrant@vagrant:~$ echo $M2_HOME
opt/apache-maven-3.6.3
vagrant@vagrant:~$ mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.20.1, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.15.0-67-generic", arch: "amd64", family: "unix"
```



► Jenkins – Installation et configuration



- 3 ème étape – Installation Jenkins
 - Vous devez installer une version Stable de jenkins (LTS :Long Term Support)

1 - Mettre à jour ton système

```
sudo apt update
```

```
sudo apt upgrade -y
```

2 - Importer la clé GPG et le dépôt Jenkins

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \  
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \  

```

```
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null
```



Jenkins – Installation et configuration



3 - Installer Jenkins

```
sudo apt update
```

```
sudo apt install jenkins -y
```

4 - Vérifier l'installation

Pour lancer Jenkins, exécuter la commande suivante:

```
sudo systemctl start jenkins
```

Optionnel: Pour lancer jenkins comme service au démarrage

```
sudo systemctl enable jenkins
```

Vérifier l'état :

```
sudo systemctl status jenkins
```



► Jenkins – Installation et configuration



```
vagrant@vagrant:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /lib/sy
stemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable jenkins
vagrant@vagrant:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor p>
   Active: active (running) since Mon 2023-09-25 21:55:38 UTC; 2min 0s ago
 Main PID: 2165 (java)
    Tasks: 43 (limit: 14111)
   Memory: 2.1G
      CPU: 38.862s
   CGroup: /system.slice/jenkins.service
           └─2165 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/>
```



► Jenkins – Installation et configuration

Les interfaces web de l'outil Jenkins sont accessibles sur le numéro du port 8080.

S'identifier - Jenkins x +

localhost:8080/login?from=%2F

ToDo Facebook LinkedIn Outlook Calendrier WhatsApp OneDrive 3A MLOps DevOps-Travai... gassen77/ci-m... RO-2425 24-25 UP Teams To Do Enonc

Démarrage

Débloquer Jenkins

Pour être sûr que Jenkins soit configuré de façon sécurisée par un administrateur, un mot de passe a été généré dans le fichier de logs ([où le trouver](#)) ainsi que dans ce fichier sur le serveur :

`/var/jenkins_home/secrets/initialAdminPassword`

Veuillez copier le mot de passe depuis un des 2 endroits et le coller ci-dessous.

Mot de passe administrateur

Continuer



► Jenkins – Installation et configuration



Pour la première fois, il faut débloquent Jenkins en tapant le mot de passe qui est stocké dans le fichier de log mentionné dans la fenêtre

```
/var/jenkins_home/secrets/initialAdminPassword
```

Veuillez copier le mot de passe depuis un des 2 endroits et le coller ci-dessous.

Mot de passe administrateur

.....|



► Jenkins – Installation et configuration



Installer les plugins suggérés

Démarrage

Personnaliser Jenkins

Les plugins étendent Jenkins avec des fonctionnalités additionnelles pour satisfaire différents besoins.

Installer les plugins suggérés

Installer les plugins que la communauté Jenkins trouve les plus utiles.

Sélectionner les plugins à installer

Sélectionner et installer les plugins les plus utiles à vos besoins.

Jenkins 2.526

© 2025-2026 –Module DevOps –CI

esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

24

► Jenkins – Installation et configuration



Installation en cours...

Installation en cours...

✓ Folders	✓ OWASP Markup Formatter	⌚ Build Timeout	⌚ Credentials Binding	** SSH server
⌚ Timestampers	⌚ Workspace Cleanup	⌚ Ant	⌚ Gradle	Folders
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View	OWASP Markup Formatter
⌚ Git	⌚ SSH Build Agents	⌚ Matrix Authorization Strategy	⌚ PAM Authentication	** Structs
⌚ LDAP	⌚ Email Extension	⌚ Mailer		** Trilead API
				** Pipeline: Step API
				** - dépendance requise

Jenkins 2.303.2



► Jenkins – Installation et configuration



Démarrage

Créer le 1er utilisateur Administrateur

Nom d'utilisateur:

Mot de passe:

Confirmation du mot de passe:

Nom complet:

Adresse courriel:

Jenkins 2.303.2

[Continuer en tant qu'Administrateur](#) [Sauver et continuer](#)



► Jenkins – Installation et configuration

Pour personnaliser le numéro de port de Jenkins, il suffit de le modifier à travers cette fenêtre.

Démarrage

Configuration de l'instance

URL de Jenkins :

L'URL de Jenkins est utilisée pour fournir l'URL de base pour les liens absolus vers les diverses ressources Jenkins. Cela signifie que cette valeur est nécessaire pour le bon fonctionnement de nombreuses fonctionnalités de Jenkins, notamment les notifications par mail, les mises à jour des statuts des pull requests, et la variable d'environnement BUILD_URL fournie pour les étapes de build.

La valeur par défaut affichée **n'est pas encore sauvegardée** et est générée à partir de la requête actuelle, lorsque c'est possible. Il est fortement recommandé d'utiliser comme valeur l'URL qui est censée être utilisée par les utilisateurs. Cela évitera des confusions lors du partage ou de la visualisation de liens.

Jenkins 2.303.2

[Passer cette étape et terminer](#)

Sauver et terminer



► Jenkins – Installation et configuration



Démarrage

Jenkins est prêt !

Vous avez passé la **création d'un utilisateur administrateur**.

Pour vous connecter, utilisez le login : "admin" et le mot de passe administrateur que vous avez utilisé pour accéder au wizard d'installation.

Vous avez passé la **configuration de l'instance**.

Vous pouvez toujours vous rendre sur les pages de configuration pour remplir les informations qui pourraient manquer.

L'installation de votre Jenkins est terminée.

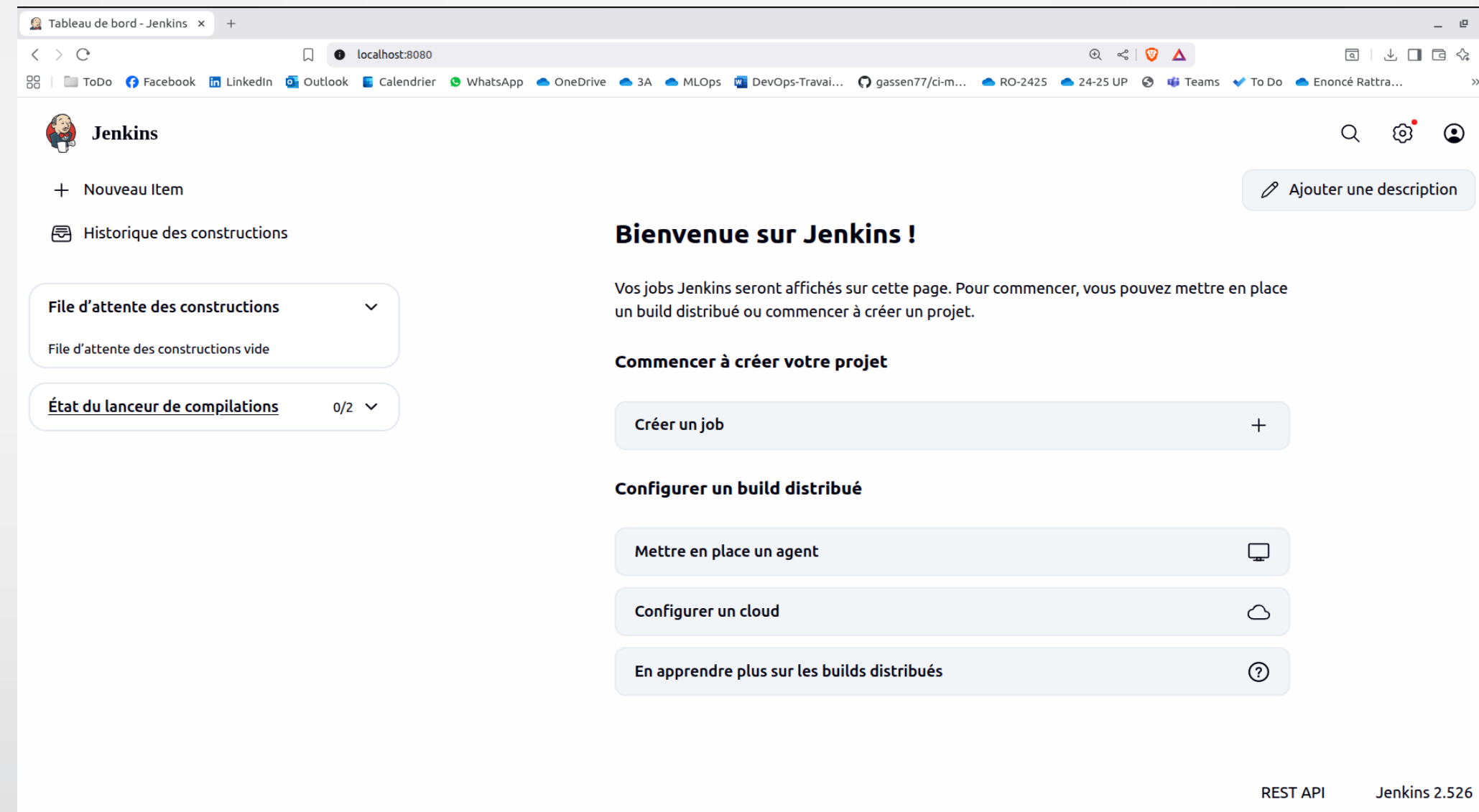
Commencer à utiliser Jenkins

Jenkins 2.526



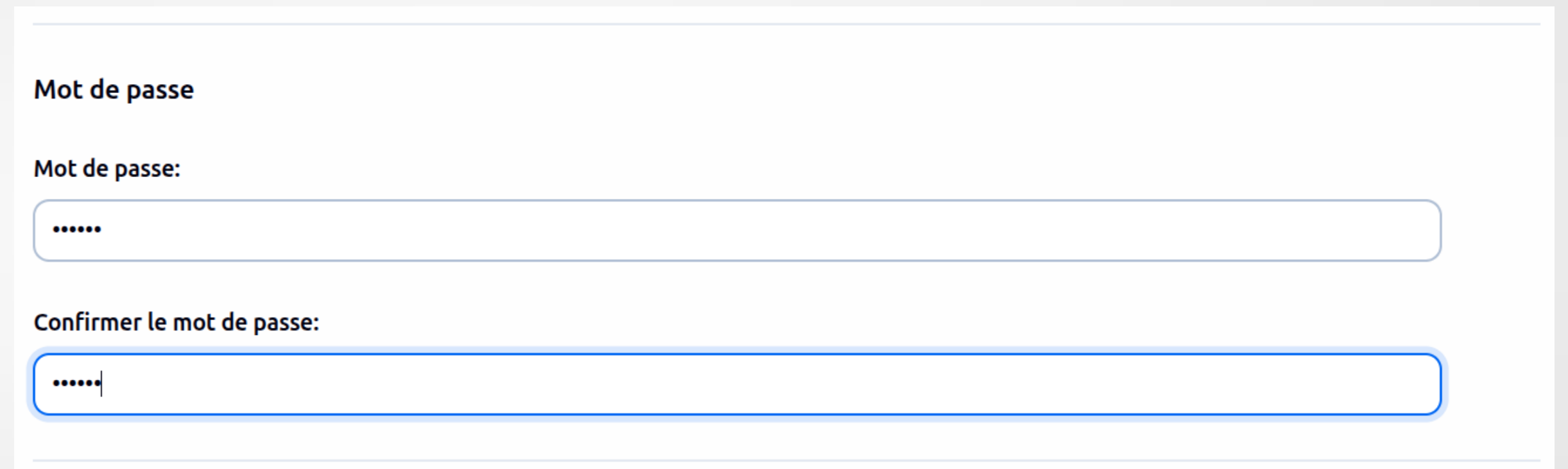
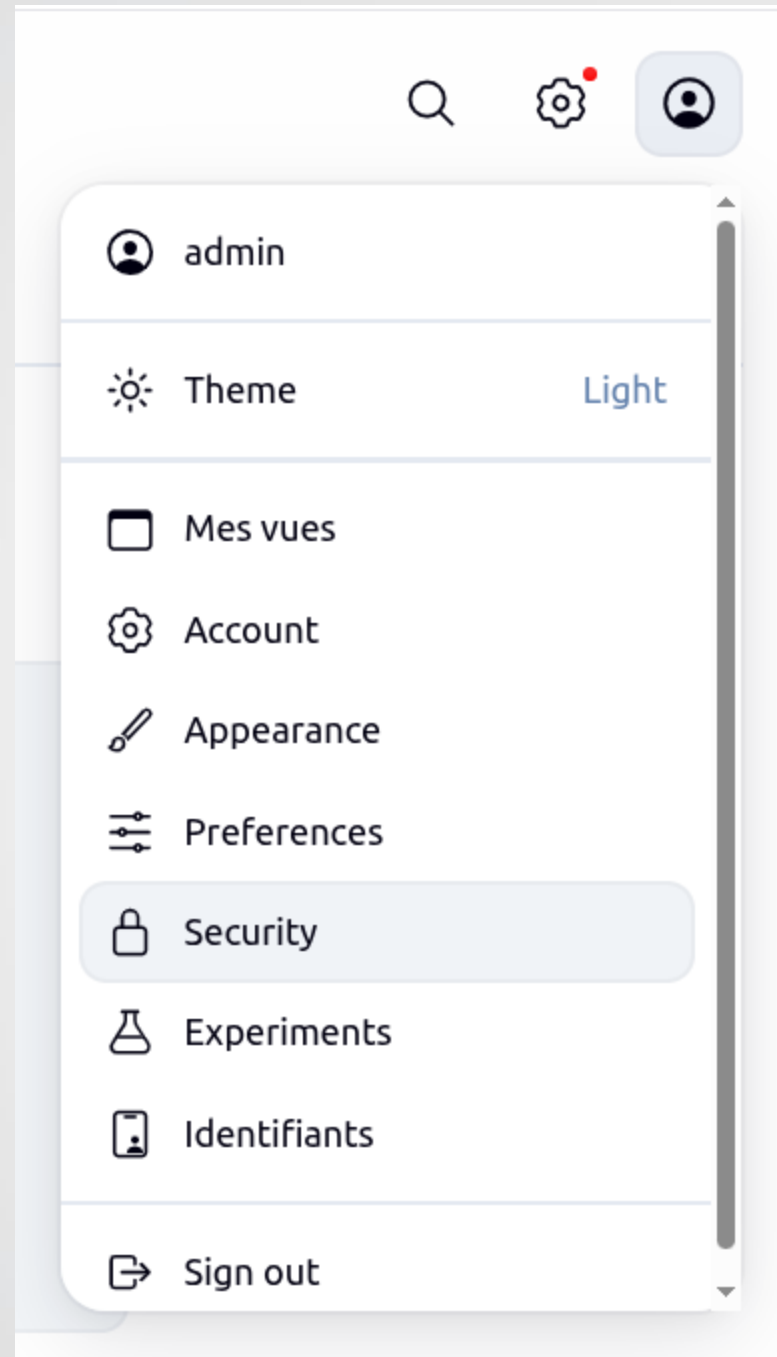
► Jenkins – Installation et configuration

Jenkins dispose d'une interface web très simple et pratique. On peut accéder directement vers tous les configurations possibles. On trouve toutes les informations à propos de tous les jobs.



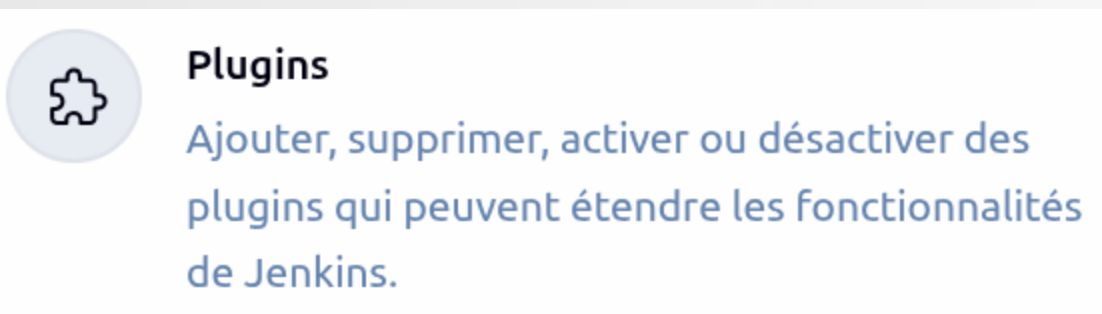
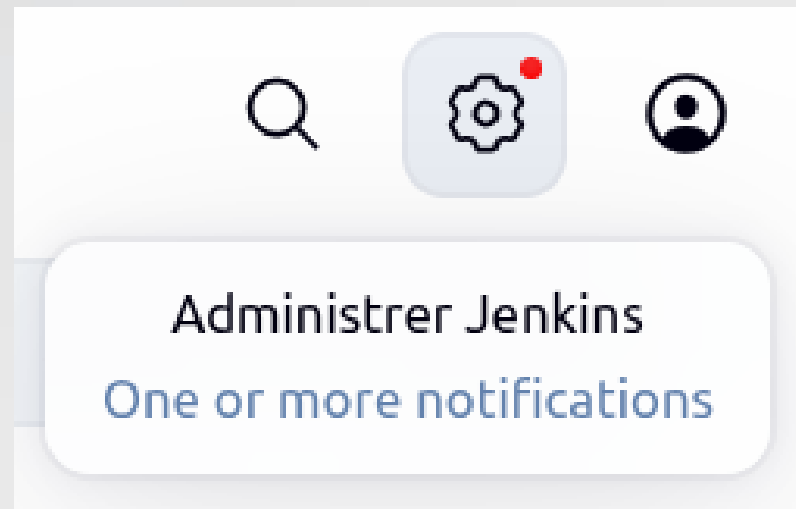
► Jenkins – Installation et configuration

Pour modifier le mot de passe, consulter l'interface « Configurer »

A screenshot of the Jenkins 'Configure' page for password management. The page has a title 'Mot de passe'. Below it is a label 'Mot de passe:' followed by a text input field containing six dots. Below that is a label 'Confirmer le mot de passe:' followed by another text input field containing six dots and a cursor. The page has a clean, modern design with a light gray background.

► Jenkins – Installation et configuration

Pour installer des plugins, il suffit d'accéder à la fenêtre « Gestion des plugins »



Jenkins – Installation et configuration



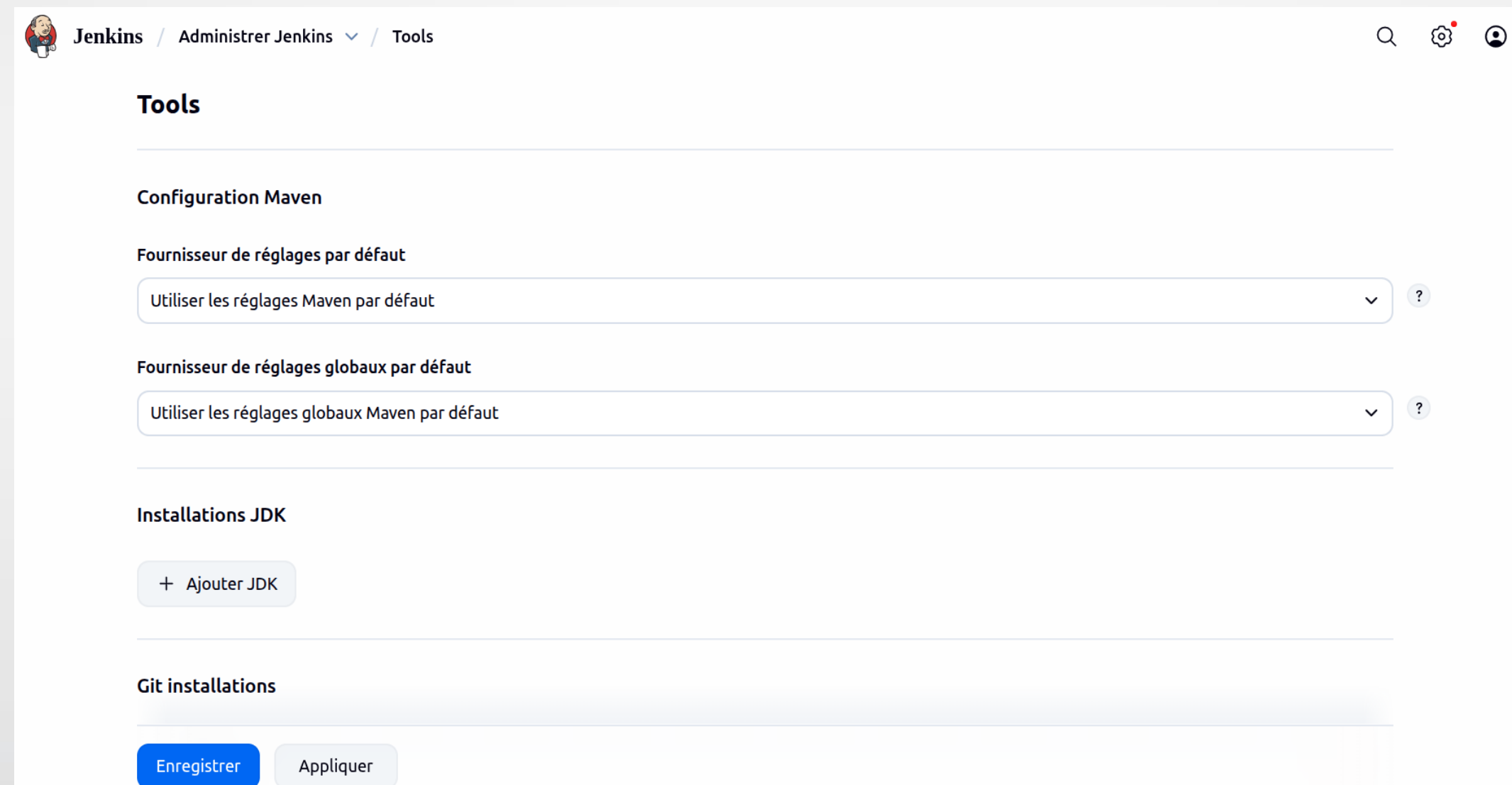
Pour créer notre chaine d'intégration continue, on va installé les plugin suivant dans Jenkins:

- Git plugin
- Maven Integration
- Sonargraph Integration
- SonarQube Scanner



► Jenkins – Installation et configuration

Pour configurer les outils, il faut accéder à la fenêtre « Configuration global des outils »

A screenshot of the Jenkins 'Tools' configuration page. The breadcrumb trail at the top shows 'Jenkins / Administrer Jenkins / Tools'. The page title is 'Tools'. Under the 'Configuration Maven' section, there are two dropdown menus: 'Fournisseur de réglages par défaut' (set to 'Utiliser les réglages Maven par défaut') and 'Fournisseur de réglages globaux par défaut' (set to 'Utiliser les réglages globaux Maven par défaut'). Below this is the 'Installations JDK' section with a '+ Ajouter JDK' button. At the bottom is the 'Git installations' section. The page concludes with 'Enregistrer' and 'Appliquer' buttons.

► Jenkins – Installation et configuration



Configurer la partie "JDK" et "maven" en associant Jenkins avec les versions installées localement

Ajouter JDK

≡ **JDK**

Nom

JAVA_HOME

JAVA_HOME

/usr/lib/jvm/java-11-openjdk-amd64/

☐ Install automatically ?

Installations Maven

Ajouter Maven

≡ **Maven**

Nom

M2_HOME

MAVEN_HOME

opt/apache-maven-3.6.3



Jenkins – Job



Les jobs ou tâches représentent le cœur du processus de « build » dans Jenkins.

Le projet dans Jenkins est représenté par un job qui composé du plusieurs processus du build.

Tout projet sous Jenkins passe par les 3 étapes suivantes:

- ✓ Création du job
- ✓ Configuration du job (configuration des étape du build)
- ✓ Lancement du build

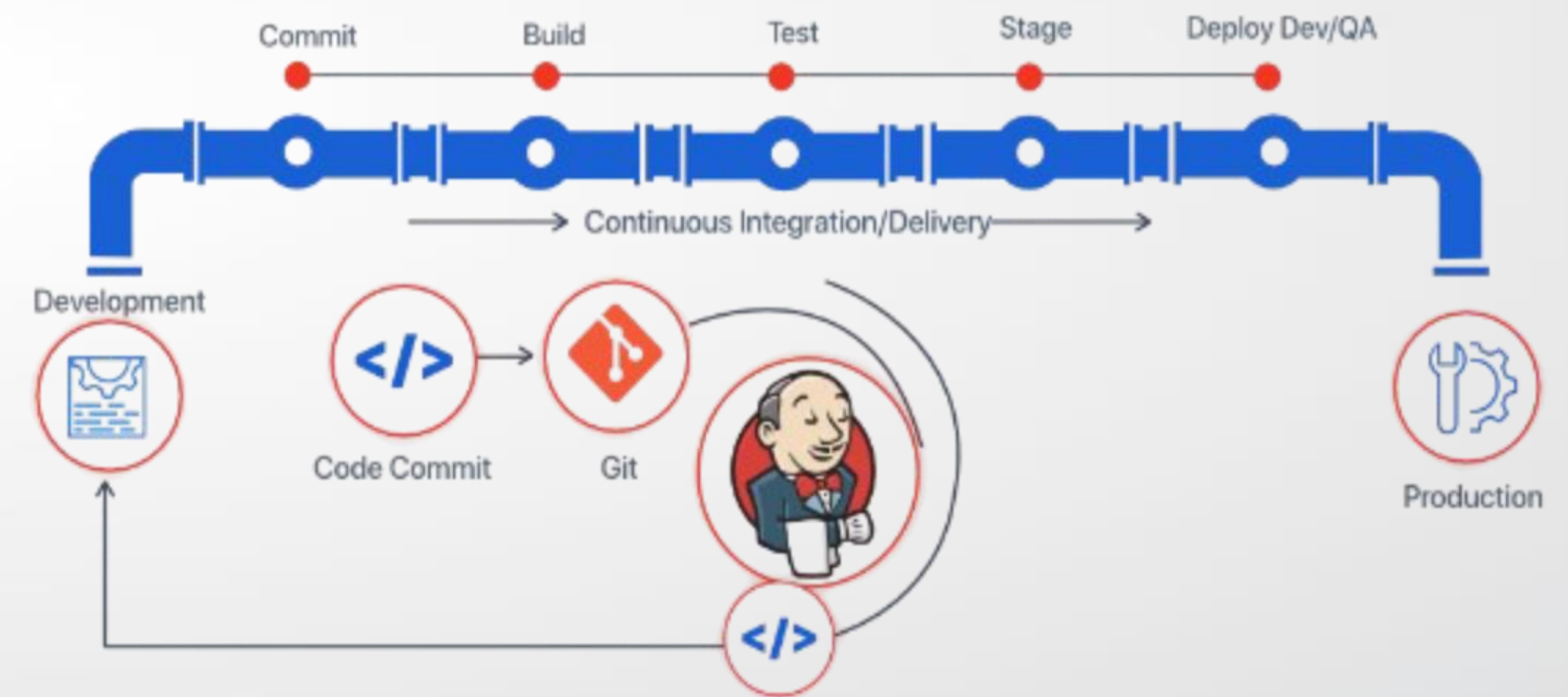
Tous processus (ou étapes) du build dans les jobs sont manipuler directement à travers des plugins.



► Jenkins – Job

Les Jobs qui permettent de compiler (build) un projet est les suivantes:

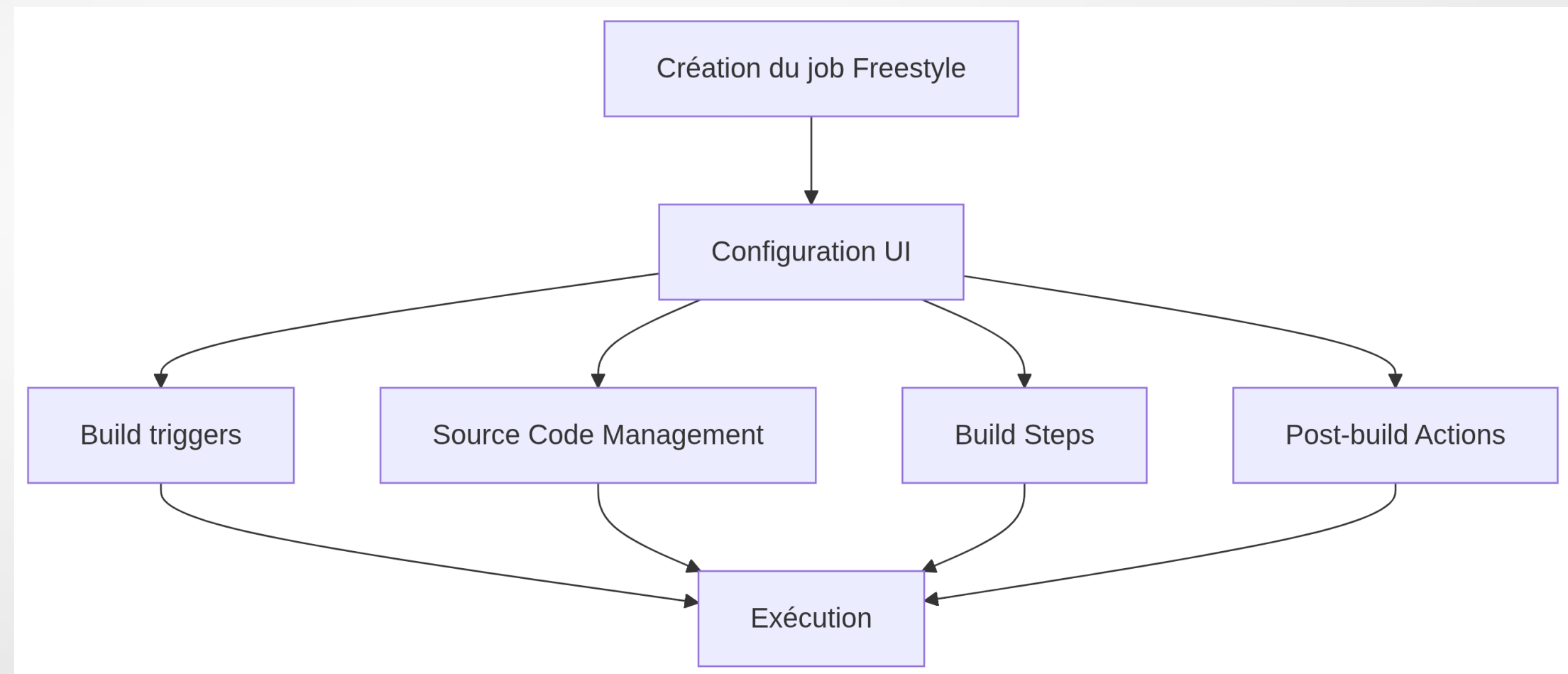
- 1/ Récupération du projet
- 2/ Compilation
- 3/ Lancement des tests unitaires automatiques (JUnit).
- 4/ Lancement des tests de qualité (Sonar)
- 4/ Préparation de la version à distribuer.
- 5/ Mise en place de la version à distribuer.
- 6/ Déployer avec Docker
- 7/ Ochestrer avec Kubernetes



► Jenkins – Job

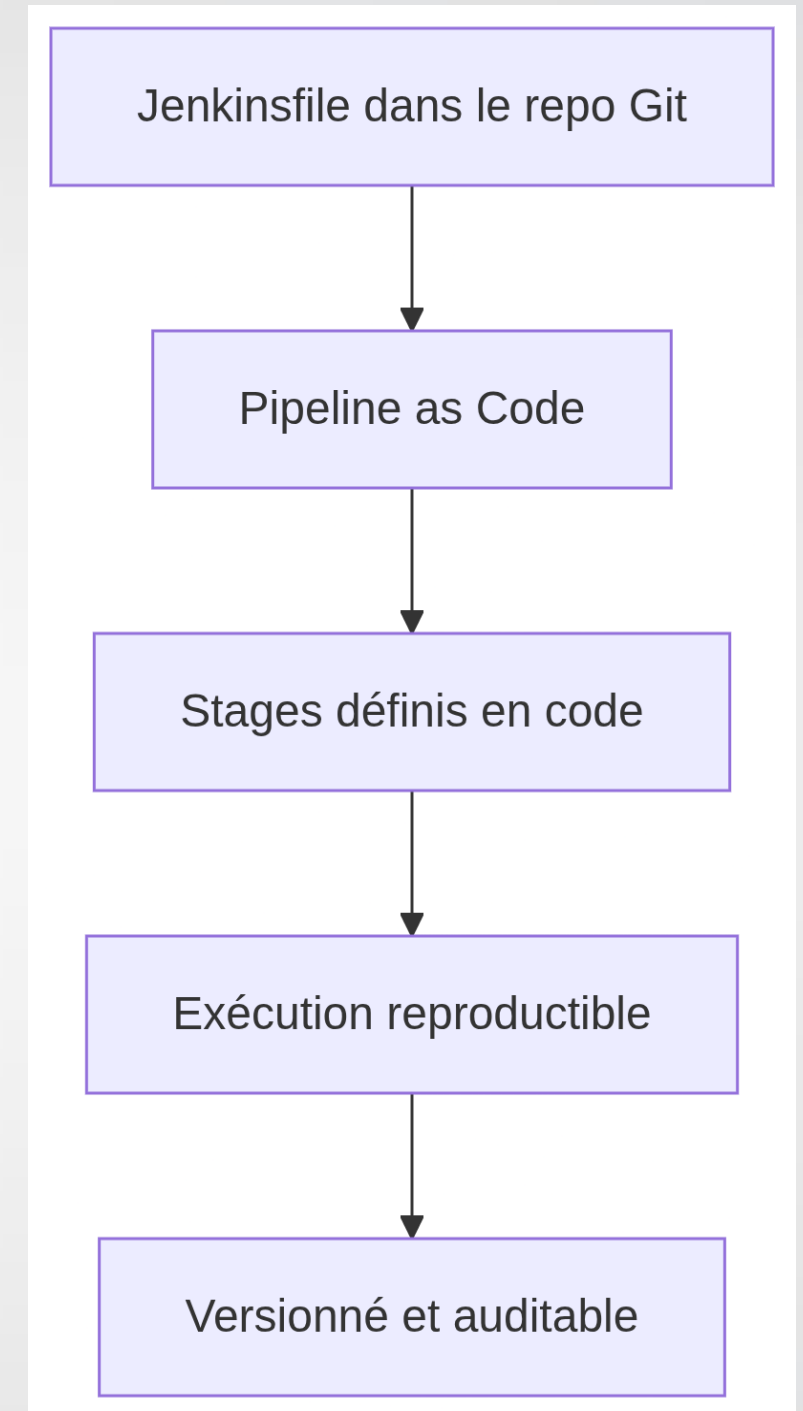
Deux méthodes de configuration sont possibles pour mettre en place un processus de compilation d'un projet :

- La méthode freestyle où la configuration du projet se réalise à travers un formulaire à remplir.



► Jenkins – Job

- L'implémentation d'un pipeline, qui définit la configuration d'un projet grâce à des scripts (basé sur Groovy). Cette méthode offre aussi les avantages de permettre de paralléliser les étapes du projet et offre une meilleure interface pour lire les logs.



► Jenkins – Job



⚡ Pourquoi Pipeline est objectivement meilleur

Critère	Freestyle	Pipeline
Définition	Configuration UI	Code (Jenkinsfile)
Versionning	Stocké dans Jenkins	Versionné avec le code
Reproductibilité	Lié à l'instance	Portable partout
Gestion des erreurs	Basique	Avancée (retry, timeout)
Visualisation	Console log basique	Stage View avancée
Complexité	Limitée	Illimitée

Pipeline n'est pas juste "mieux" - c'est le standard moderne pour une raison très simple : ça transforme votre CI/CD from magic to code.









▶ Jenkins – Créer un Job



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, suivi 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.
-  **Dossier**
Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.
-  **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.
-  **Pipeline Multibranches**
Crée un ensemble de projets Pipeline en se basant sur les branches détectées dans le dépôt d'un gestionnaire de code source.

OK









Jenkins – Créer un Job FreeStyle



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, suivi 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.
-  **Dossier**
Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.
-  **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.
-  **Pipeline Multibranches**
Crée un ensemble de projets Pipeline en se basant sur les branches détectées dans le dépôt d'un gestionnaire de code source.

OK



► Jenkins – Créer un Job FreeStyle



Après la création du projet, il faut enchaîner avec les Jobs pour compiler le projet.
La première partie de la page de configuration indique les informations générales du projet.

The screenshot shows the Jenkins configuration interface. At the top, the breadcrumb navigation reads 'Jenkins / TutoFreeStyle / Configuration'. On the left, a sidebar titled 'Configurer' contains a list of configuration sections: 'Général' (selected), 'Gestion de code source', 'Triggers', 'Environment', 'Étapes du build', and 'Actions à la suite du build'. The main area is titled 'Général' and features an 'Enabled' toggle switch in the top right corner, which is currently turned on. Below the title, there is a 'Description' field, which is currently empty. Underneath the description field, there are two links: 'Texte brut' and 'Prévisualisation'. Below these links, there are five unchecked checkboxes with corresponding labels and help icons: 'Ce build a des paramètres', 'GitHub project', 'Supprimer les anciens builds', 'Throttle builds', and 'Exécuter des builds simultanément si nécessaire'. At the bottom of the configuration section, there is a button labeled 'Avancé' with a dropdown arrow.

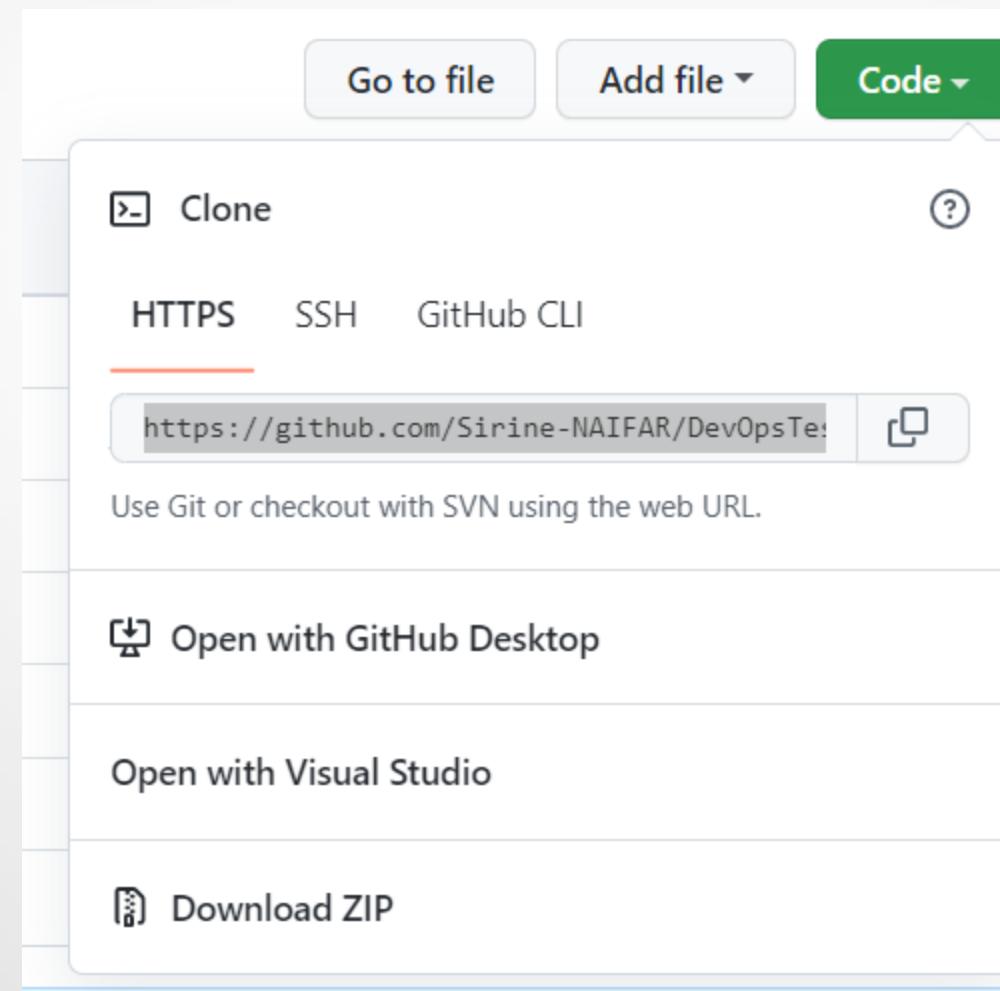


► Jenkins – Créer un Job FreeStyle



1/ Récupération du code

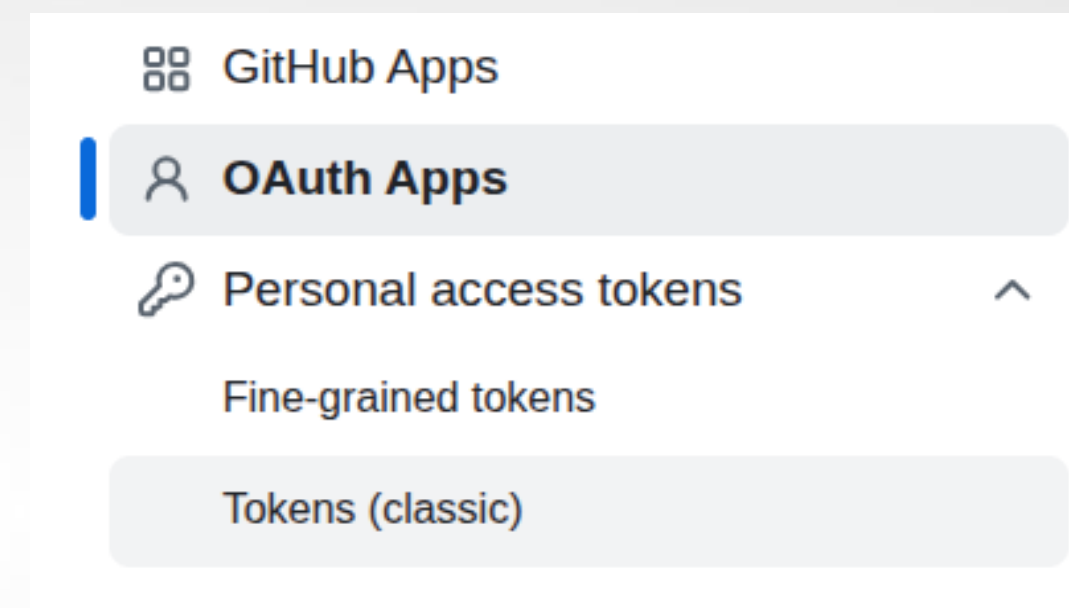
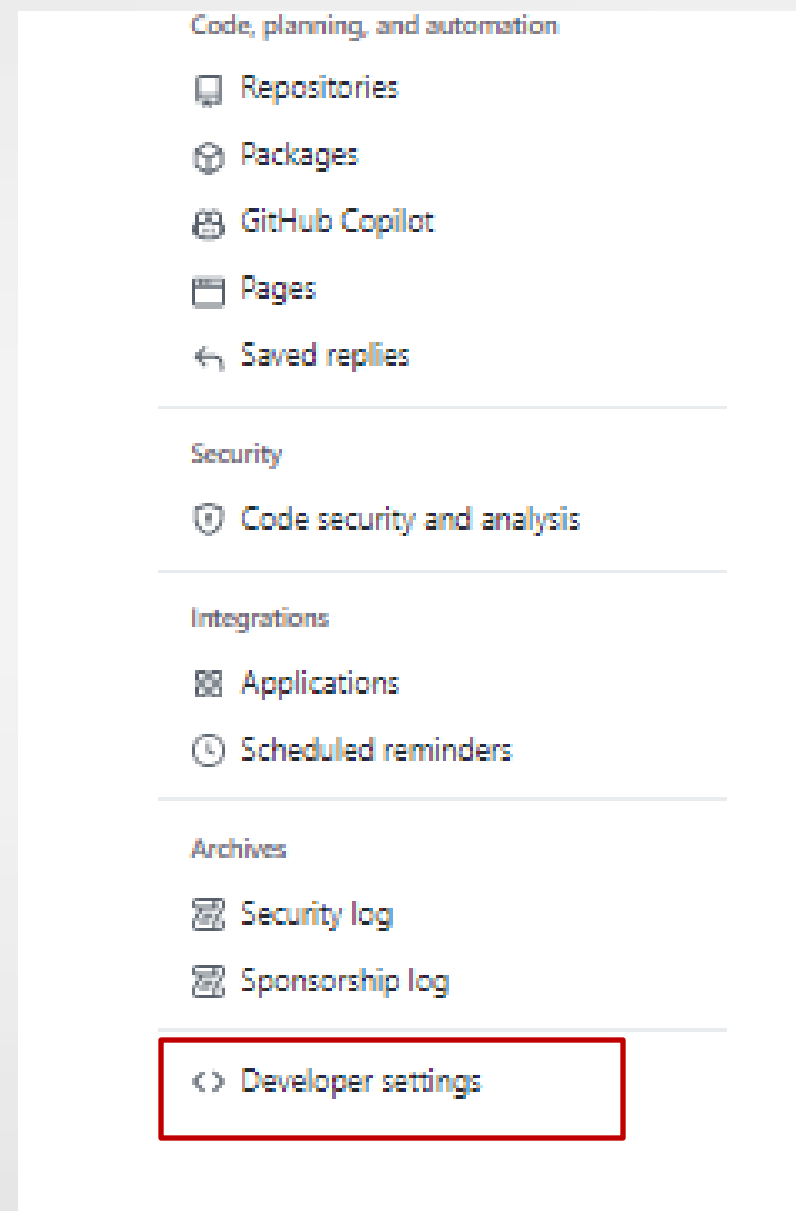
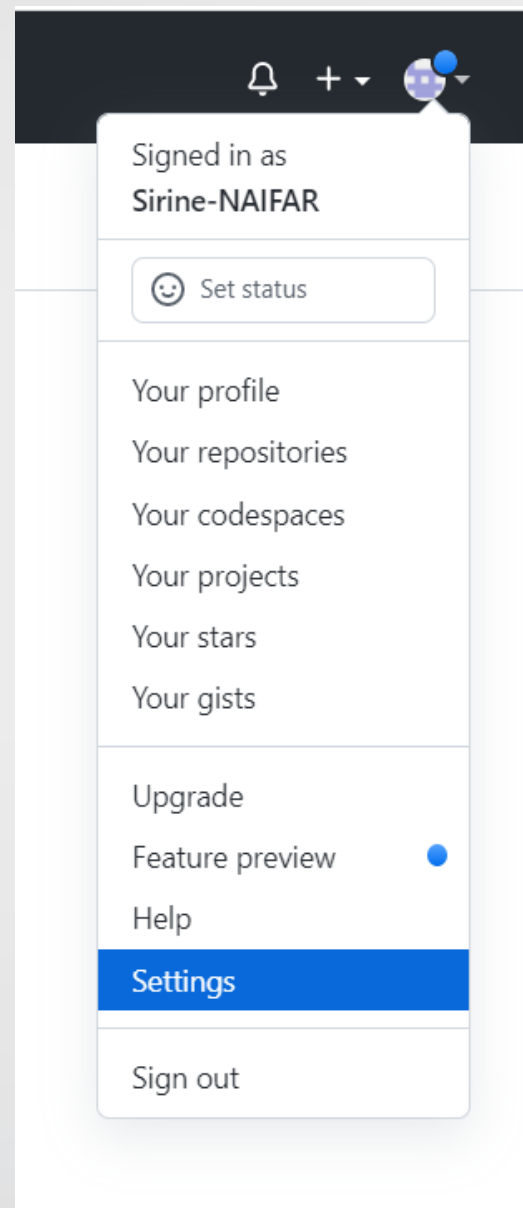
Il faut ajouter des clés SSH ou de définir des noms d'utilisateur et des mots de passe dans la partie « Identifiants » pour que le Jenkins soit capable de récupérer le code.



► Jenkins – Créer un Job FreeStyle

1/ Récupération du code

Il faut générer un token à partir du Git.



- Remplir le formulaire avec les informations nécessaires pour générer un token



► Jenkins – Créer un Job FreeStyle



Il faut choisir l'action qui va lancer la chaine de l'intégration continue périodiquement (process qui se lance chaque 15 minutes).

Ce qui déclenche le build

- ☐ Déclencher les builds à distance (Par exemple, à partir de scripts) ?
- ☐ Construire après le build sur d'autres projets ?
- ☒ Construire périodiquement ?

Cliquer ici pour plus d'informations



Planning

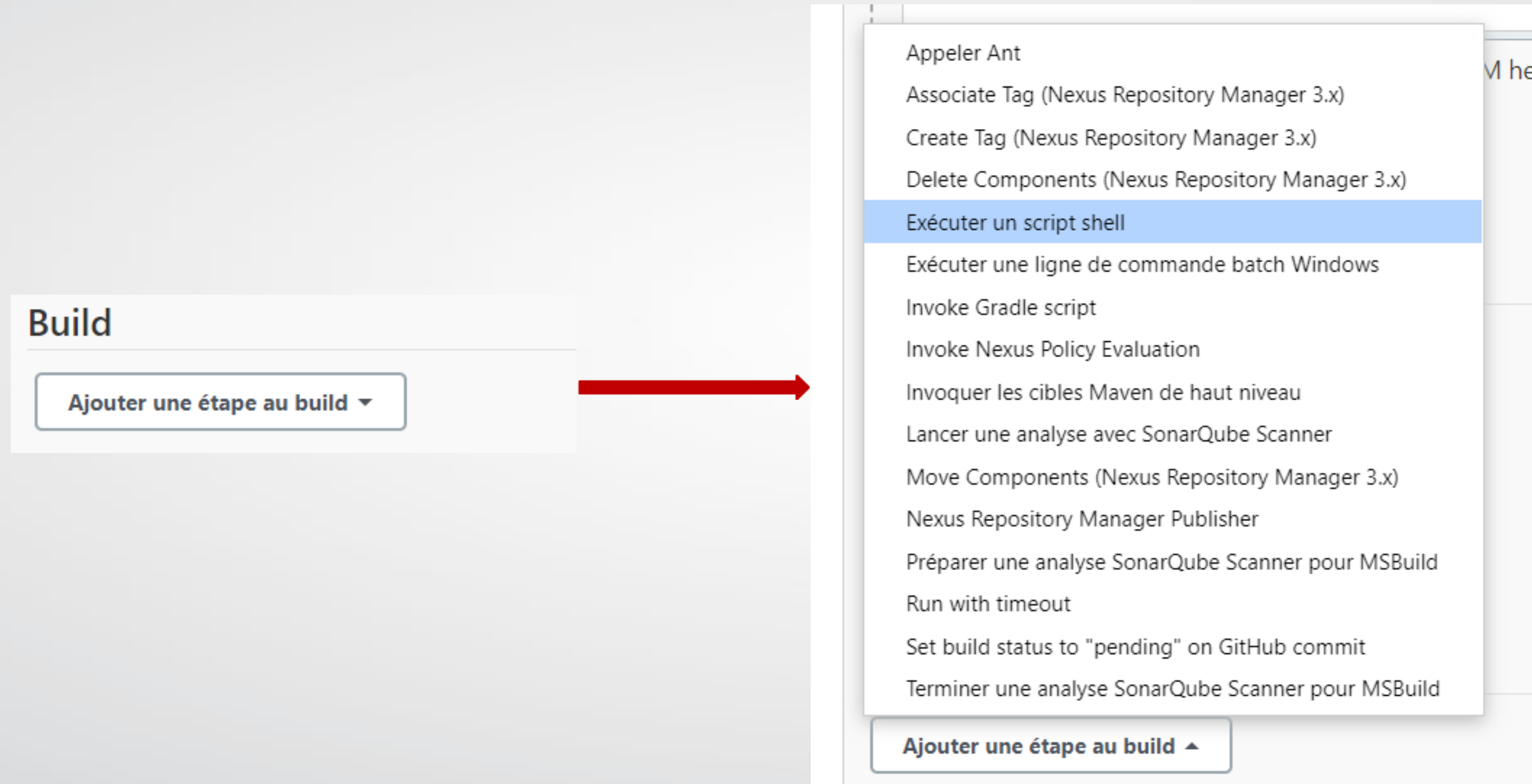
H/15 * * * *

Aurait été lancé à lundi 25 octobre 2021 14 h 23 WAT; prochaine exécution à lundi 25 octobre 2021 14 h 38 WAT.



► Jenkins – Créer un Job FreeStyle

Pour les autres tâches (2, 3, 4, ...), on peut les configurer à travers la partie Build.



The image shows a screenshot of the Jenkins 'Build' section. On the left, there is a button labeled 'Ajouter une étape au build' with a downward arrow. A red arrow points from this button to a dropdown menu on the right. The dropdown menu lists various build steps, with 'Exécuter un script shell' highlighted in blue. Below the dropdown menu, there is another button labeled 'Ajouter une étape au build' with an upward arrow.

Build

Ajouter une étape au build ▼

Exécuter un script shell

Ajouter une étape au build ▲

- Appeler Ant
- Associate Tag (Nexus Repository Manager 3.x)
- Create Tag (Nexus Repository Manager 3.x)
- Delete Components (Nexus Repository Manager 3.x)
- Exécuter un script shell
- Exécuter une ligne de commande batch Windows
- Invoke Gradle script
- Invoke Nexus Policy Evaluation
- Invoquer les cibles Maven de haut niveau
- Lancer une analyse avec SonarQube Scanner
- Move Components (Nexus Repository Manager 3.x)
- Nexus Repository Manager Publisher
- Préparer une analyse SonarQube Scanner pour MSBuild
- Run with timeout
- Set build status to "pending" on GitHub commit
- Terminer une analyse SonarQube Scanner pour MSBuild



► Jenkins – Créer un Job FreeStyle



Exemple Simple 01: Ecrire un message simple en affichant la date système.

☰ Exécuter un script shell ?

Commande

Voir [la liste des variables d'environnement disponibles](#)

```
echo "test from spring AOP"
date
```

Avancé...



► Jenkins – Créer un Job FreeStyle



Exemple Simple 02: Vérifie l'installation de maven.

 Exécuter un script shell ? 

Commande

Voir [la liste des variables d'environnement disponibles](#)

```
mvn -version
```


Avancé...





► Jenkins – Créer un Job FreeStyle





Sauvegarder la configuration et lancer le build.


 **Jenkins** / TutoFreeStyle


 **État**


 Modifications


 Répertoire de travail

 Lancer un build

 Configurer

 Supprimer Projet

 Renommer


 Identifiants

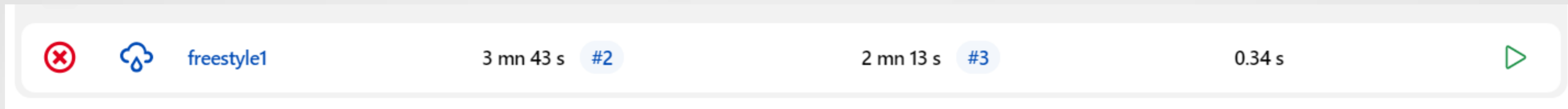
TutoFreeStyle


Liens permanents



► Jenkins – Créer un Job FreeStyle

Si le build a échoué, l'icone  s'affiche avant le nom du projet au niveau du dashboard.




Si le build a réussi, l'icone  s'affiche.

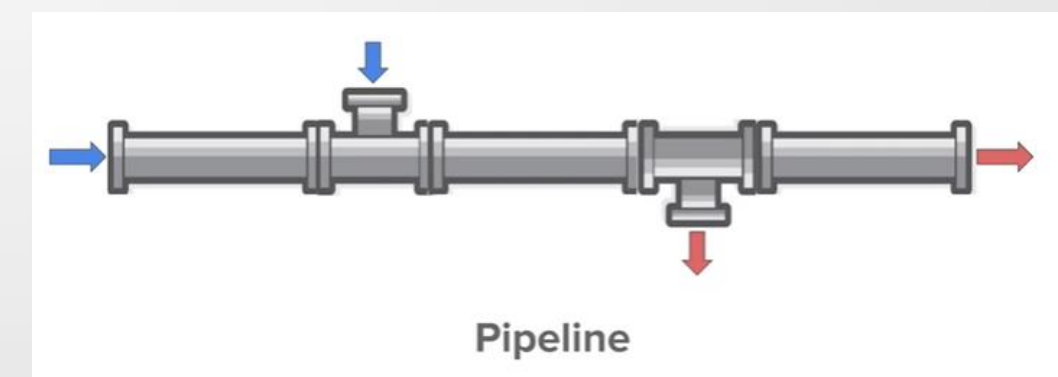


► Jenkins – Créer un Job Pipeline



- Le pipeline Jenkins est implémenté sous forme de code qui permet à plusieurs utilisateurs de modifier et d'exécuter le processus de pipeline.
- Pipeline donne une solution à cette problématique en offrant un nouveau langage pour l'écriture de la configuration des tâches grâce notamment au langage groovy. 
- Le pipeline peut ainsi être sauvegardé dans un fichier (versionning) et peut gérer les différents cas possibles.

- <https://www.jenkins.io/doc/book/pipeline/getting-started/>









Jenkins – Créer un Job Pipeline



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, suivi 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.
-  **Dossier**
Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.
-  **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.
-  **Pipeline Multibranches**
Crée un ensemble de projets Pipeline en se basant sur les branches détectées dans le dépôt d'un gestionnaire de code source.

Si vous voulez créer un nouvel élément à partir d'un autre, vous pouvez utiliser cette option :



► Jenkins – Créer un Job Pipeline



Il faut choisir l'action qui va lancer la chaine de l'intégration continue si Jenkins détecte un push dans le Git (process qui se lance chaque 5 minutes).

Build Triggers

☐ Construire après le build sur d'autres projets ?
☐ Construire périodiquement ?
☐ GitHub hook trigger for GITScm polling ?
☒ Scrutation de l'outil de gestion de version ?

Planning

H/5 * * * *

Aurait été lancé à lundi 25 octobre 2021 18 h 00 WAT; prochaine exécution à lundi 25 octobre 2021 18 h 05 WAT.

☐ Ignore post-commit hooks ?

Cliquer ici pour plus d'informations



► Jenkins – Créer un Job Pipeline



Exemple Simple : Afficher « Hello world » avec « Groovy »

Pipeline

Definition

Pipeline script ▼

Script ?

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8       }
9     }
10  }
11 }
12
```

Hello World ▼



► Jenkins – Créer un Job Pipeline



Exemple 01 : Récupération du code du git

Pipeline

Definition

Pipeline script

Script

```
1 pipeline {
2   agent any
3   stages{
4     stage('Checkout GIT'){
5       steps {
6
7         echo 'Pulling... ';
8         git branch: 'main',
9           url : 'https://github.com/sirineDevOps/SpringIOC',
10          credentialsId: '5847fd35-c2c3-45fc-a0d5-656108235cb2';
11       }
12     }
13   }
14 }
15
16
17
```

Si le repository est privé

☒ Use Groovy Sandbox



► Jenkins – Créer un Job Pipeline



Exemple 02 : Exécuter une commande Maven

Pipeline

Definition

Pipeline script ▼

Script ?





```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Testing maven') {
6       steps {
7         sh """mvn -version"""
8       }
9     }
10  }
11 }
12
```



► Jenkins – Créer un Job Pipeline



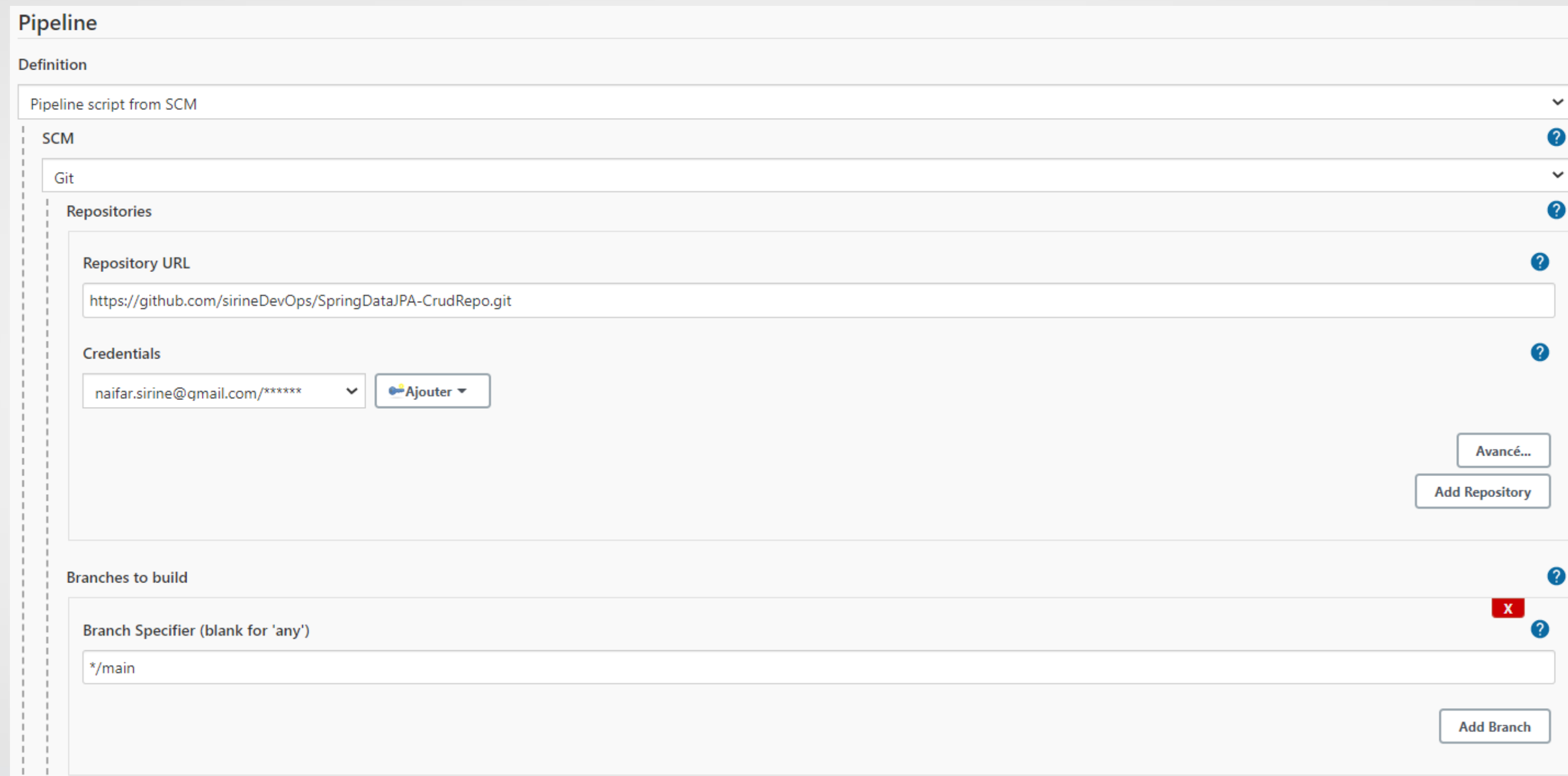
- Si on applique les bonnes pratiques, il est préférable d'utiliser un fichier stocké dans le référentiel du code (git) qui contient le pipeline
- Ce fichier est nommé Jenkinsfile.

	sirineDevOps Create Jenkinsfile	1ef97a6 30 minutes ago	🕒 14 commits
	src/main/java/tn/esprit/esponline	commit from user3	19 days ago
	Jenkinsfile	Create Jenkinsfile	30 minutes ago
	pom.xml	Update pom.xml	2 hours ago



► Jenkins – Créer un Job Pipeline

Pour pointer sur ce fichier, il suffit juste de configurer le pipeline jenkins.



The screenshot shows the Jenkins Pipeline configuration interface. The 'Definition' section is set to 'Pipeline script from SCM'. Under the 'SCM' section, 'Git' is selected. The 'Repositories' section contains a 'Repository URL' field with the value 'https://github.com/sirineDevOps/SpringDataJPA-CrudRepo.git' and a 'Credentials' dropdown menu showing 'naifar.sirine@gmail.com/*****'. There is an 'Ajouter' button next to the credentials. At the bottom right of the repositories section are 'Avancé...' and 'Add Repository' buttons. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with the value '*/main' and an 'Add Branch' button. There are also several help icons (question marks) and a red 'X' icon in the branches section.

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/sirineDevOps/SpringDataJPA-CrudRepo.git

Credentials

naifar.sirine@gmail.com/*****

Ajouter

Avancé...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Add Branch



▶ Travail à faire



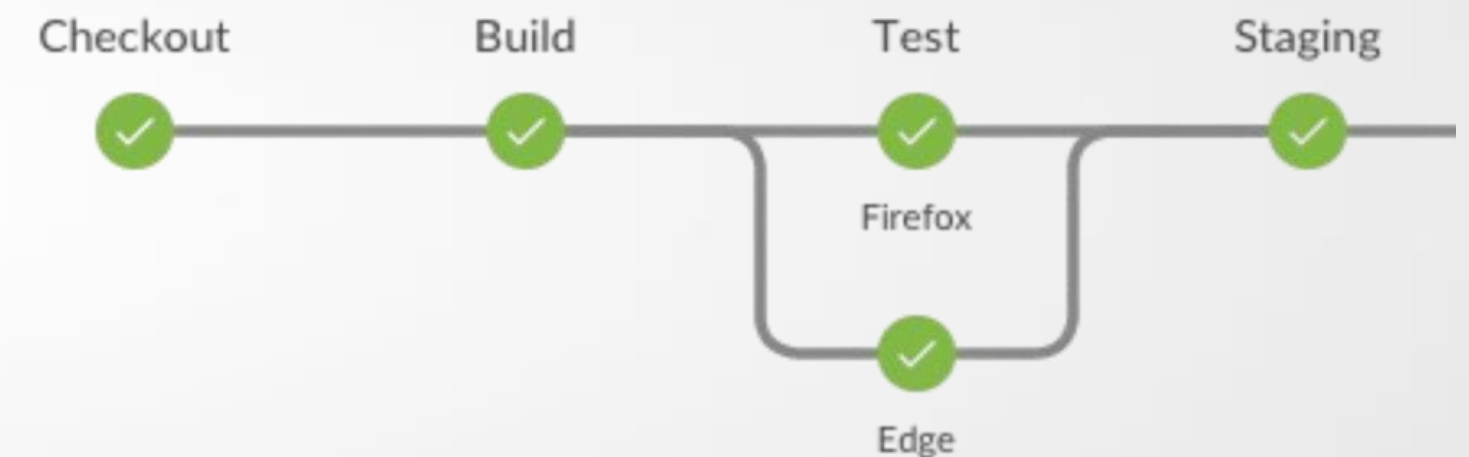
- En utilisant « Groovy », créer une chaine d'intégration continue qui se lance lors du détection d'un push dans le git (Jenkinsfile).
- Ce pipeline contient:

La récupération du code source.

La suppression du contenu du dossier target

La compilation

La création du livrable





► " Apprendre par le projet, c'est transformer la théorie en pratique, renforcer l'esprit d'équipe et développer des solutions durables. Avec Jenkins, nous posons les bases d'une meilleure gestion des ressources et de l'efficacité "

