



Chapitre 5

L'intégration continue en utilisant Jenkins (Continuous Integration CI)

Enseignante: Dr-Ing. Amina JARRAYA

Email : amina.jarraya@ensi-uma.tn

Niveau: II3- GL

Plan

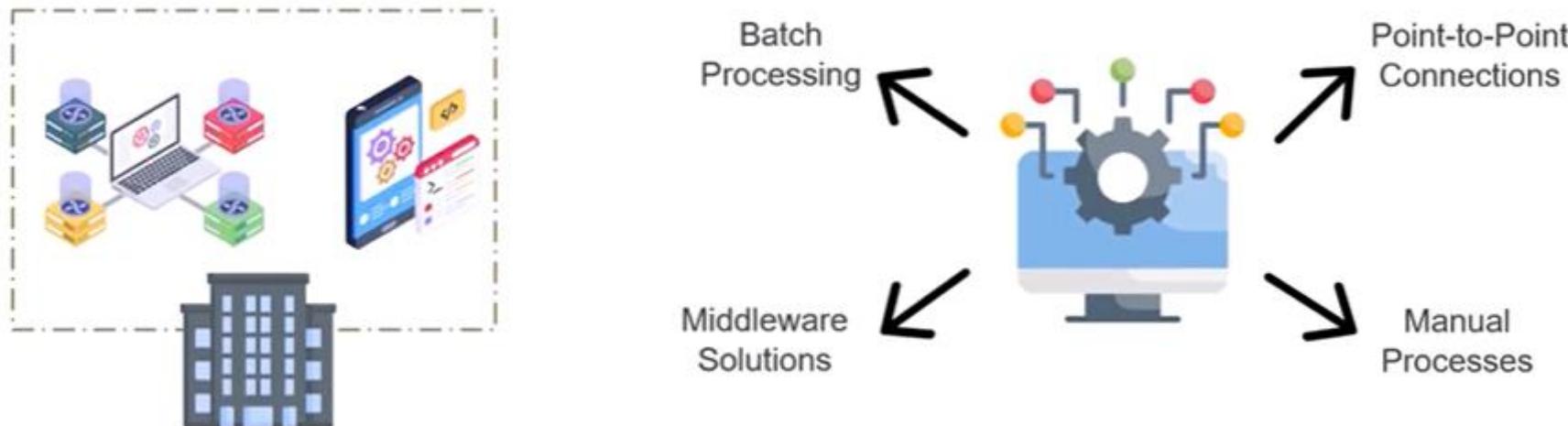
1. Introduction à l'intégration continue (CI)
2. Présentation de Jenkins
3. Installation de Jenkins
4. Gestion des jobs free-style sous Jenkins
5. Gestion de pipelines sous Jenkins
6. Création des pipelines déclaratives sous Jenkins
7. Github Webhook et Jenkins
8. Mise en place d'un pipeline CI/CD

1. Introduction à l'Intégration Continue (CI)

Limites de l'intégration traditionnelle

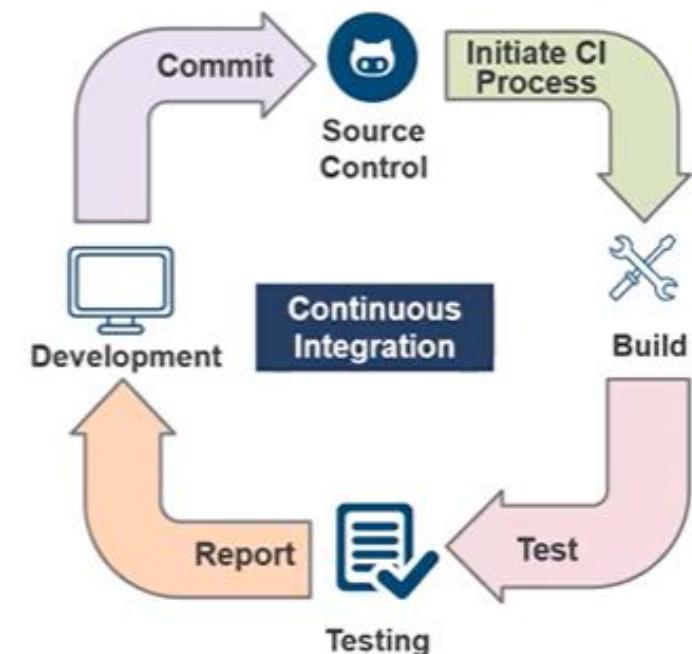
L'intégration traditionnelle se déroule généralement à la fin du développement d'un projet, une fois que les différents modules ont été développés et testés individuellement.

- Implique l'assemblage de tous les modules, suivi de tests, de corrections et de nouveaux tests en boucle jusqu'à ce que tout soit correct.
- Les corrections de bugs en fin de projet peuvent être coûteuses et complexes car elles impliquent souvent des modifications importantes du code.
- Manque souvent de tests exhaustifs, ce qui peut entraîner des versions logicielles avec des bugs persistants.



L'intégration continue (CI)

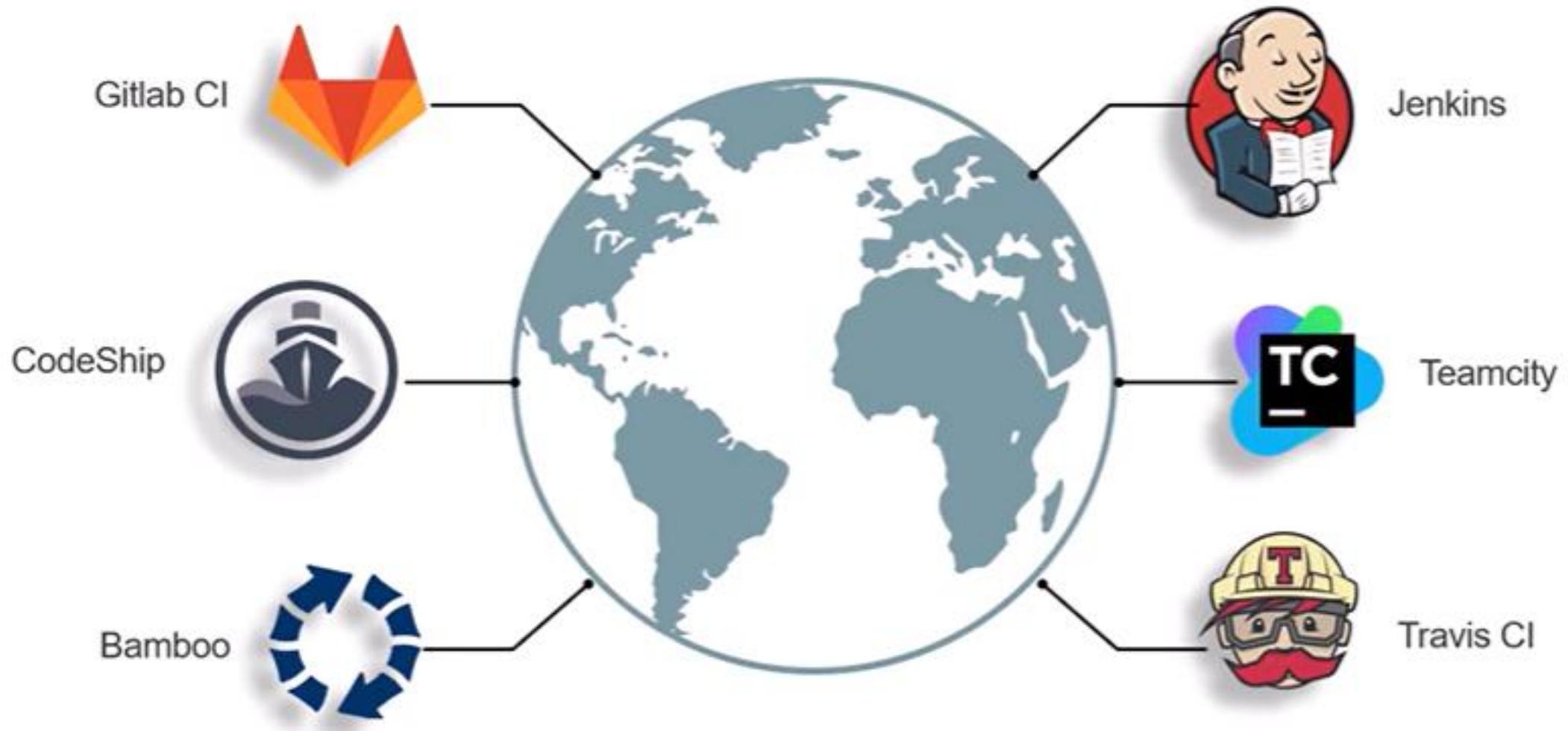
- L'intégration continue (CI) est un processus qui consiste à intégrer fréquemment les modifications de code dans un référentiel centralisé, suivi de la création et de l'exécution automatisée de tests.
- L'objectif est de détecter rapidement les erreurs et les conflits entre les différentes contributions, favorisant ainsi une collaboration plus efficace et une meilleure qualité du code.



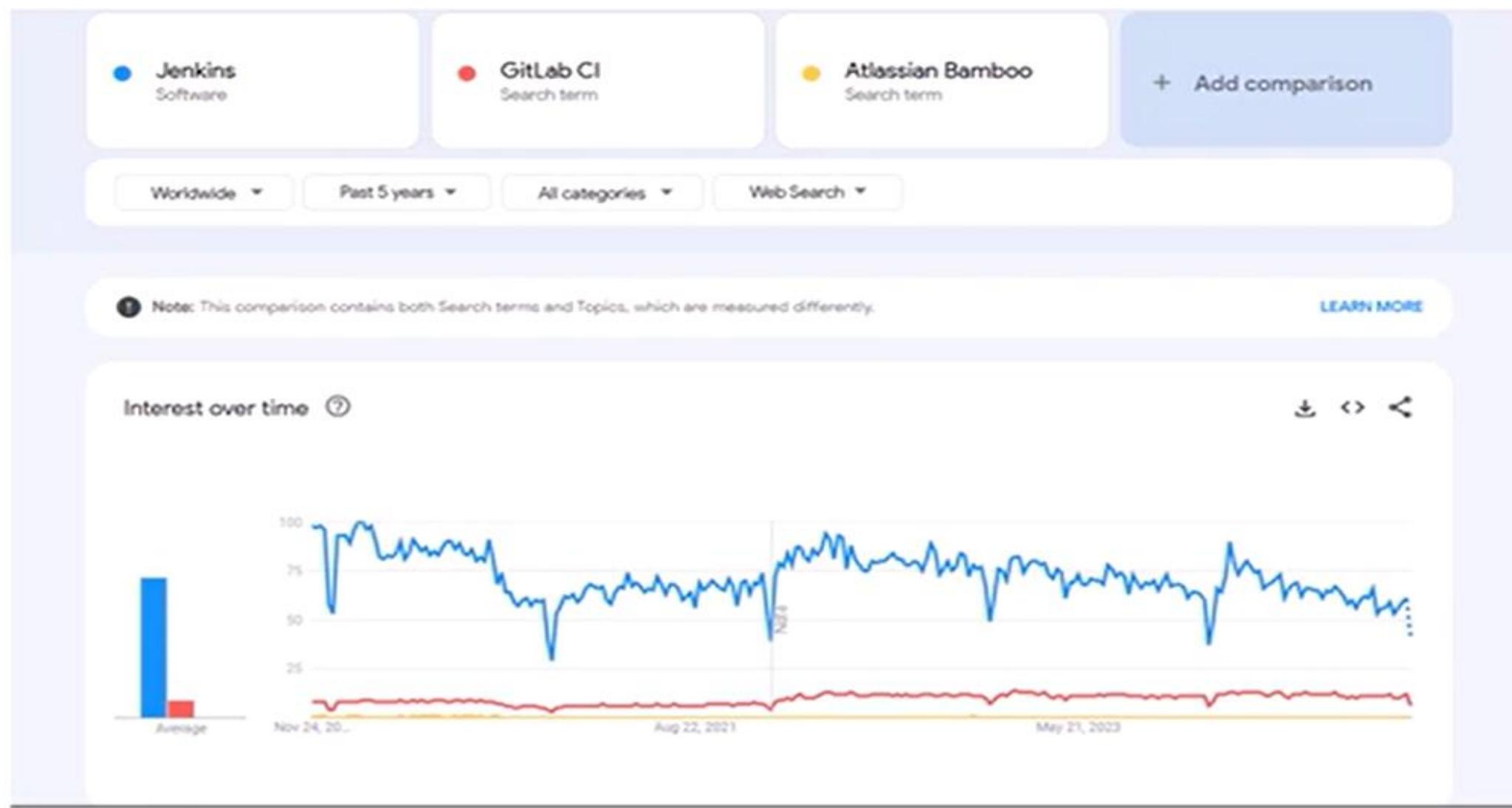
Avantages de l'intégration continue

- **Productivité accrue:** L'automatisation des tâches manuelles telles que la compilation, les tests et le déploiement libère les développeurs pour se concentrer sur des tâches plus complexes et créatives.
- **Détection précoce des erreurs:** Les tests automatisés intégrés à la CI permettent de détecter rapidement les problèmes de code, empêchant ainsi les erreurs de se propager et de devenir plus coûteuses à corriger plus tard.
- **Amélioration de la qualité du code:** Les tests fréquents et les retours d'information constants aident à identifier et à corriger les erreurs plus rapidement, ce qui se traduit par une meilleure qualité du code.
- **Livraisons plus rapides:** En automatisant le processus de construction et de test, la CI permet de livrer des mises à jour plus fréquemment et plus rapidement, ce qui permet aux entreprises de répondre plus efficacement aux besoins du marché.
- **Réduction des coûts:** L'automatisation de la CI permet de réduire le temps et les ressources nécessaires pour les tests et les déploiements, ce qui peut entraîner des économies significatives.

Outils de l'intégration continue



Pourquoi Jenkins ?



Google Trends: Jenkins VS Travis VS Atlassian Bamboo

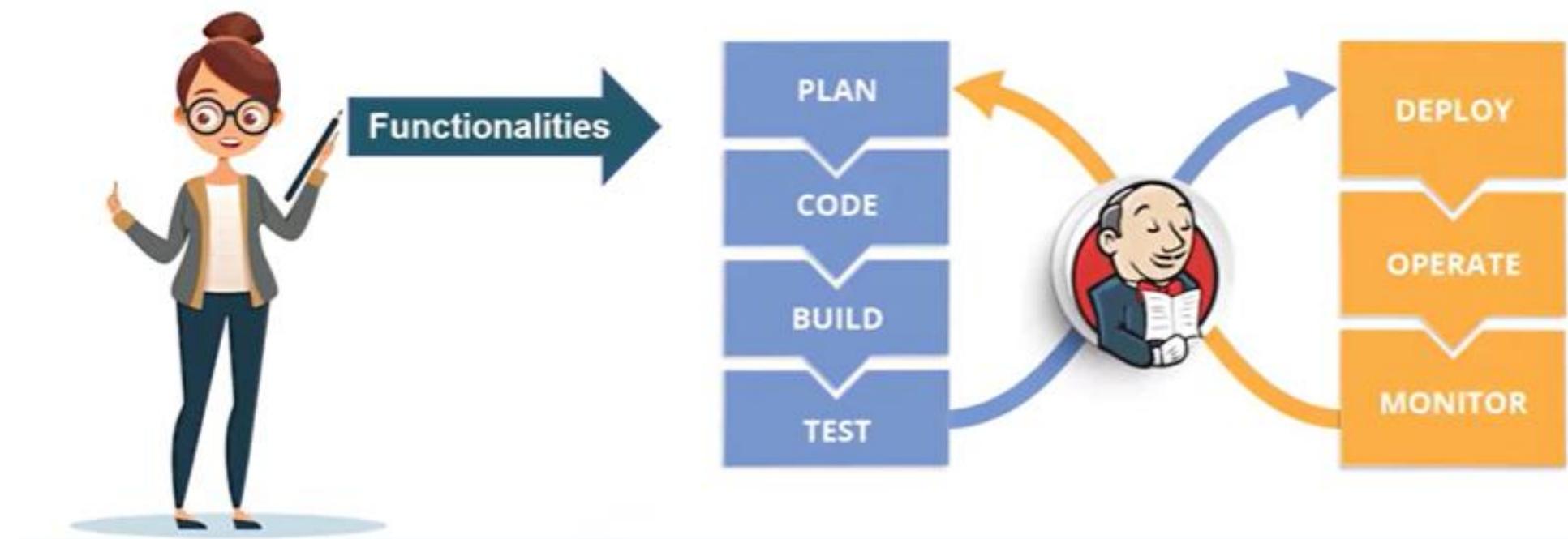
Les entreprises utilisant Jenkins



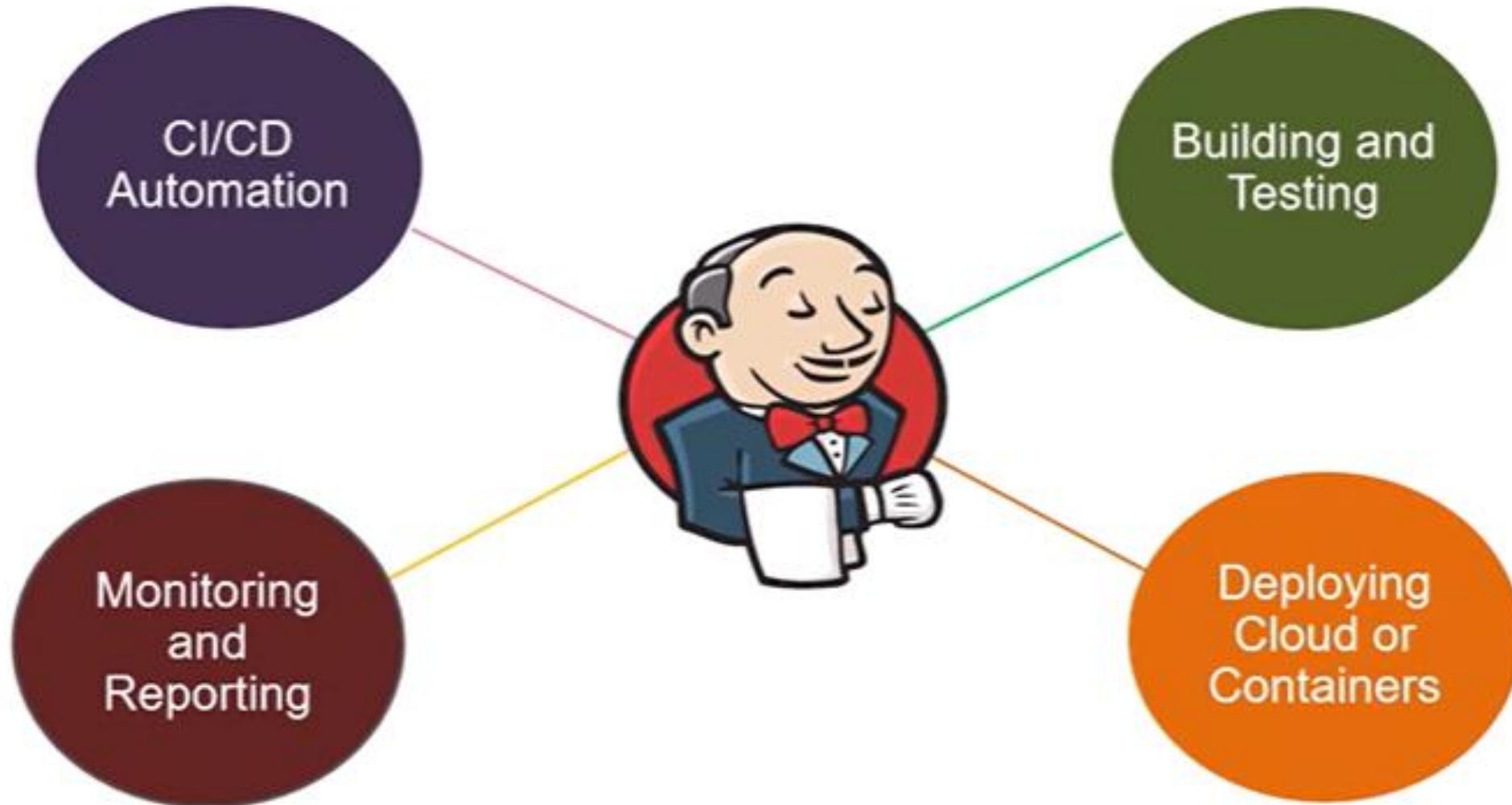
2. Présentation de Jenkins

Jenkins c'est quoi ?

- Jenkins est un serveur d'automatisation open source utilisé principalement pour l'intégration et la livraison continues (CI/CD) dans le développement logiciel.
- Il automatise les tâches de **build**, de **test** et de **déploiement**, facilitant ainsi l'intégration continue du code et la livraison rapide des changements.
- Écrit en Java, Jenkins peut être étendu via des plugins pour s'intégrer à divers outils et environnements.



Les cas d'utilisation de Jenkins



Les projets dans Jenkins

- En Jenkins, un "**job**" est une tâche automatisée que vous configurez pour construire, tester et déployer votre projet. C'est une unité fondamentale de travail qui permet d'exécuter une séquence d'actions (telles que la compilation du code, l'exécution de tests, ou le déploiement de l'application).



Building



Testing



Deploying



Delivering Softwares

Les différents types de projet dans Jenkins

Free Style Project

Central feature of Jenkins. Jenkins will build your project with any build system



Multi-configuration Project

Suitable for projects that need large number of different configurations



Organization

Scans a Github Organization for all matching repositories..



Pipeline

Suitable for building pipelines or organizing complex activities that do not easily fit in free style



Folder

Creates a container and stores nested items in it. Useful for grouping things together.

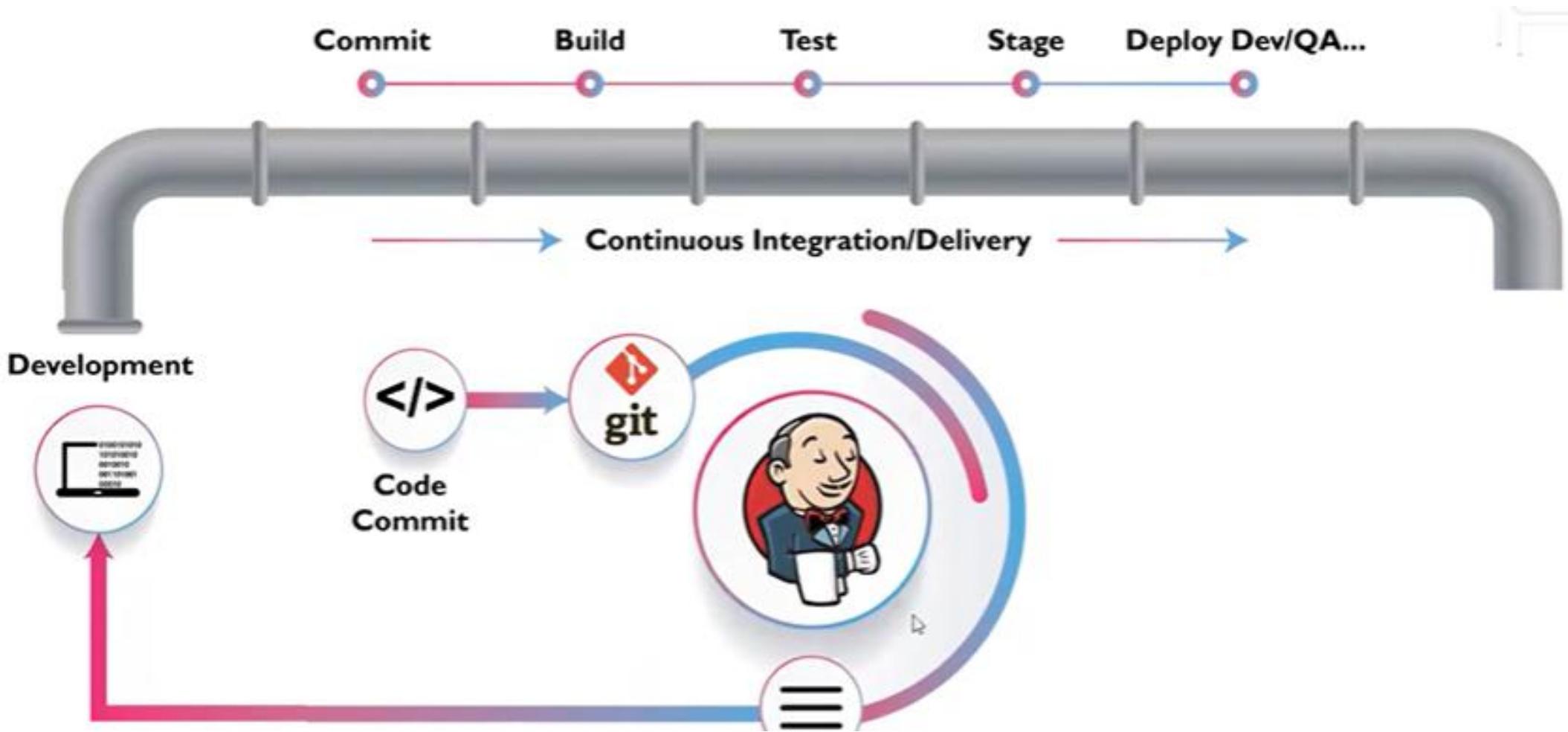


Multibranch Pipeline

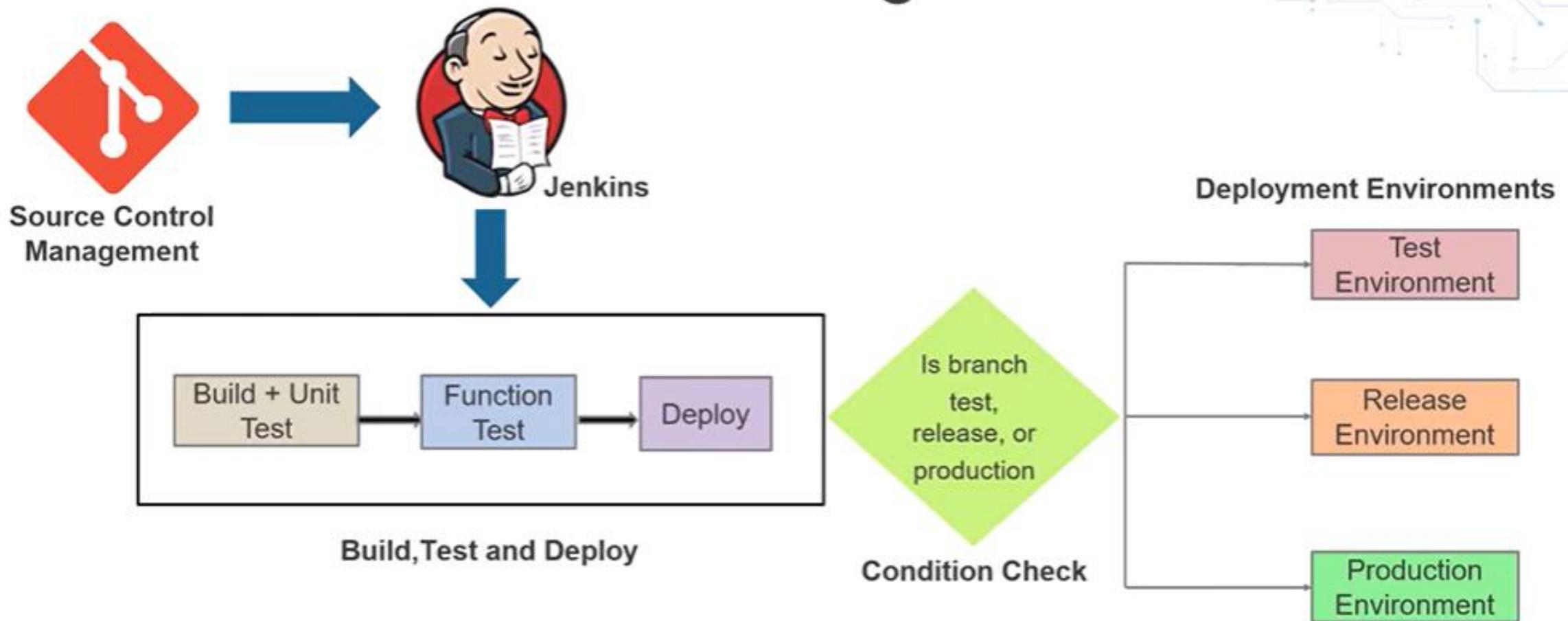
Creates a set of pipeline projects according to detected branches in one SCM repositories.



Rôle de Jenkins dans devops

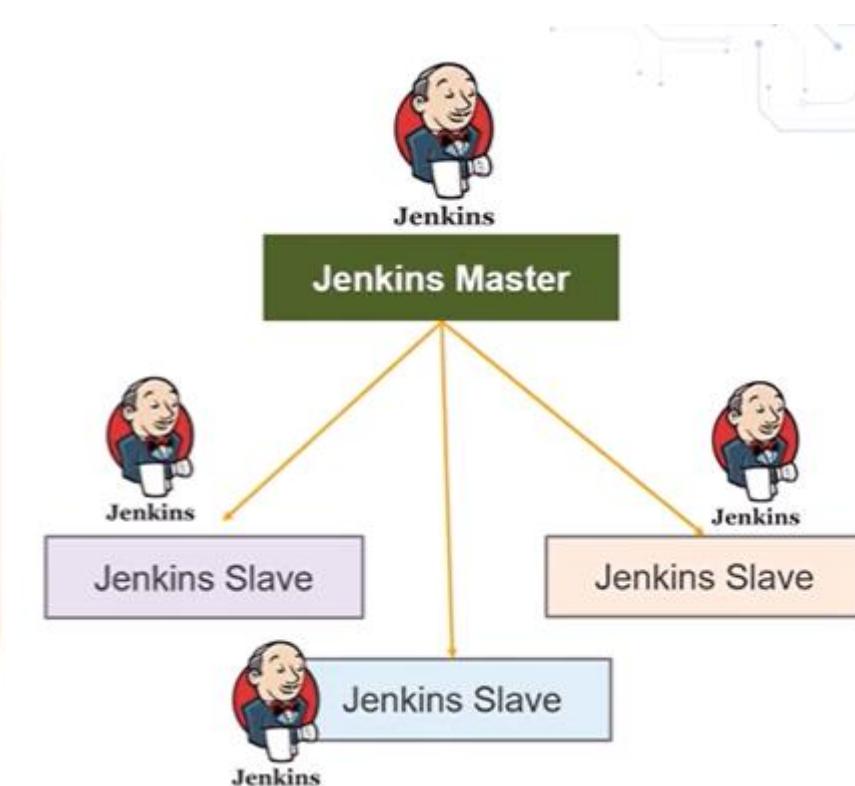
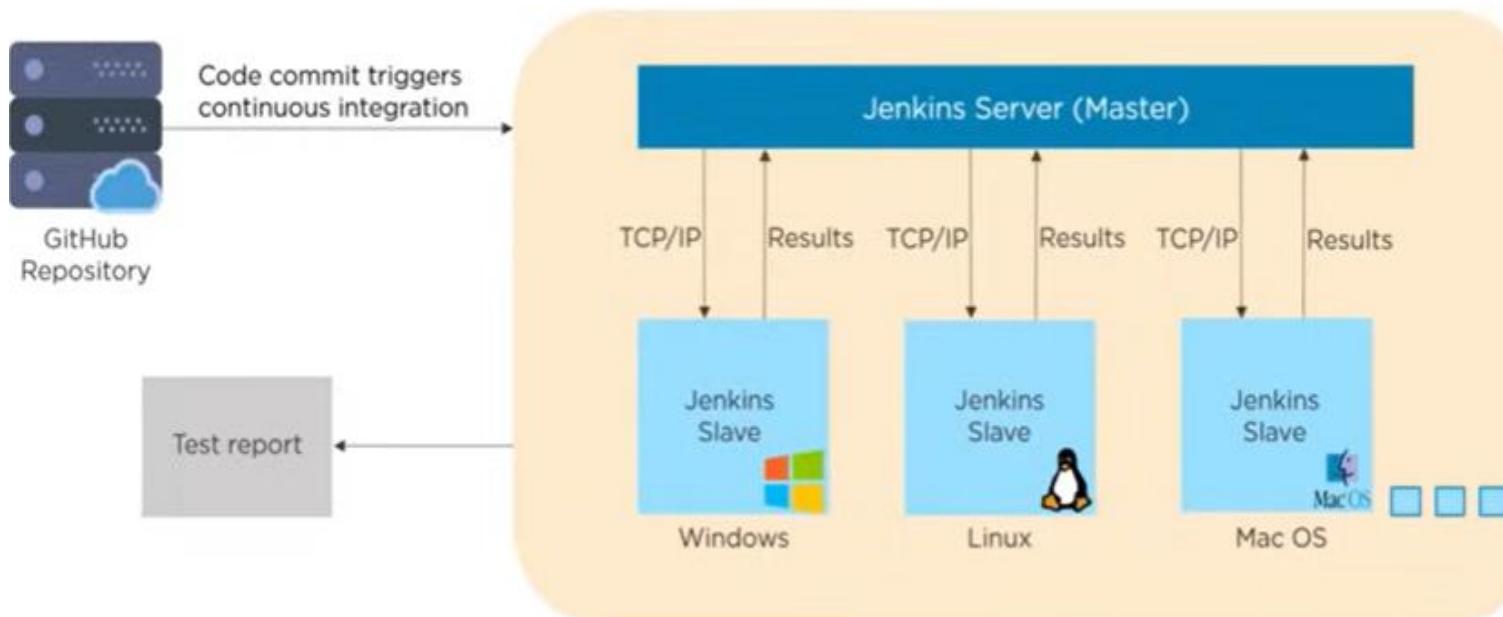


Processus de Jenkins



Architecture de Jenkins

- L'architecture de Jenkins est basée sur un modèle maître-esclave (master-slave).
- Le maître Jenkins est le composant central qui orchestre et coordonne les tâches, tandis que les agents Jenkins (esclaves) exécutent les tâches de construction, de test et de déploiement sur différentes machines.



Le maître et les esclaves en Jenkins

- **Jenkins Master:**
 - Le serveur principal qui gère la configuration, la planification et la surveillance des tâches. Il est responsable de la coordination de l'ensemble du processus CI/CD.
- **Jenkins Agents (Esclaves):**
 - Des machines distantes qui exécutent les tâches assignées par le maître. Ils peuvent être configurés pour exécuter des tâches spécifiques sur différents systèmes d'exploitation et environnements.

Le rôle des esclaves

- **Exécution des tâches :**

- Les agents sont les machines sur lesquelles les jobs Jenkins sont réellement exécutés, comme la compilation de code, l'exécution de tests automatisés et le déploiement de nouvelles versions.

- **Répartition de la charge :**

- Ils permettent de répartir la charge de travail, évitant ainsi que le noeud maître ne soit surchargé. C'est crucial pour les systèmes avec beaucoup d'automatisation et de tâches simultanées.

- **Support de configurations multiples :**

- Les agents peuvent être configurés sur différentes machines avec des systèmes d'exploitation variés (Windows, Linux, Docker, Kubernetes) pour répondre aux besoins spécifiques de chaque projet ou de chaque étape du processus CI/CD.

- **Utilisation optimisée :**

- Il est possible de configurer les agents pour qu'ils se lancent uniquement en cas de besoin et se mettent hors ligne lorsqu'ils ne sont pas utilisés, ce qui permet d'économiser les ressources système.

Les types des esclaves dans Jenkins

Esclaves statiques (permanents)

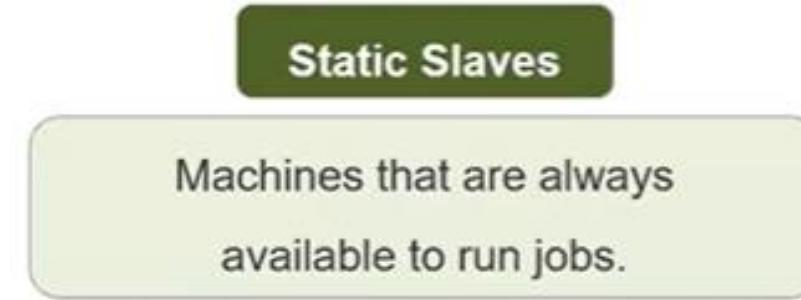
- **Configuration manuelle:**

- Ces agents sont configurés manuellement via des méthodes telles que SSH ou JNLP et sont connectés en permanence au nœud maître de Jenkins.

- **Utilisation courante:**

- Ils sont adaptés pour gérer une charge de travail constante et continue, où des ressources dédiées sont nécessaires.

- **Exemple:** Une machine virtuelle configurée spécifiquement pour exécuter les tâches de build de Jenkins et toujours accessible.



Les types des esclaves dans Jenkins

Esclaves dynamiques:

- **Provisionnement à la demande:**
- Ces agents sont créés et supprimés automatiquement par Jenkins, souvent en réponse à une demande de build ou à des besoins spécifiques de capacité.
- **Utilisation courante:**
- Ils sont idéaux pour gérer des charges de travail variables ou pour utiliser des ressources partagées dans des environnements tels que Kubernetes ou des conteneurs.
- **Exemple:** Un conteneur Docker créé et supprimé sur un cluster Kubernetes lorsqu'un build nécessite plus de ressources ou lorsqu'il est terminé.



Dynamic Slaves

Slaves that are created on-demand
and automatically provisioned.

3. Installation de Jenkins

Installer Jenkins sous WSL

- Ouvrez WSL et tapez les commandes suivantes :

```
root@DESKTOP-SK00CDK:~# apt-get update
```

```
root@DESKTOP-SK00CDK:~# apt-get install openjdk-21-jdk
```

```
root@DESKTOP-SK00CDK:~# java -version
openjdk version "21.0.8" 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
```

- Allez sur : <https://www.jenkins.io/doc/book/installing/linux/#debianubuntu>

Et exécutez les 4 commandes suivantes (ne pas oublier de supprimer les slacks « \ »)

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
/etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

Installer Jenkins sous WSL

- Version installée de Jenkins
- Démarrer le serveur de Jenkins
- Vérifier le statut du serveur

```
Setting up jenkins (2.516.1) ...
```

```
root@DESKTOP-SKOOCDK:~# systemctl start jenkins
```

```
root@DESKTOP-SKOOCDK:~# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
    Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
    Active: active (running) since Tue 2025-07-29 14:55:14 CEST; 2min 26s ago
      Main PID: 42040 (java)
        Tasks: 53 (limit: 9299)
       Memory: 535.2M (peak: 674.4M)
         CPU: 22.937s
        CGroup: /system.slice/jenkins.service
                └─42040 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cac>

Jul 29 14:55:11 DESKTOP-SKOOCDK jenkins[42040]: 5e2a2f500a4142698da28af2747dd213
Jul 29 14:55:11 DESKTOP-SKOOCDK jenkins[42040]: This may also be found at: /var/lib/jenkins/secrets/initialAdmi>
Jul 29 14:55:11 DESKTOP-SKOOCDK jenkins[42040]: ****
Jul 29 14:55:11 DESKTOP-SKOOCDK jenkins[42040]: ****
Jul 29 14:55:11 DESKTOP-SKOOCDK jenkins[42040]: ****
Jul 29 14:55:14 DESKTOP-SKOOCDK jenkins[42040]: 2025-07-29 12:55:14.471+0000 [id=50]           INFO      jenkins>
Jul 29 14:55:14 DESKTOP-SKOOCDK jenkins[42040]: 2025-07-29 12:55:14.511+0000 [id=40]           INFO      hudson.>
Jul 29 14:55:14 DESKTOP-SKOOCDK systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Jul 29 14:55:15 DESKTOP-SKOOCDK jenkins[42040]: 2025-07-29 12:55:15.572+0000 [id=88]           INFO      h.m.Dow>
Jul 29 14:55:15 DESKTOP-SKOOCDK jenkins[42040]: 2025-07-29 12:55:15.573+0000 [id=88]           INFO      hudson.>
lines 1-20/20 (END)
```

Installer Jenkins sous WSL

- Allez sur : <http://localhost:8080/>

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/lib/jenkins/secrets/initialAdminPassword
```

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

Mot de passe administrateur

- Extraire le mot de passe depuis le chemin sélectionné en rouge et le coller dans « Mot de passe administrateur » :

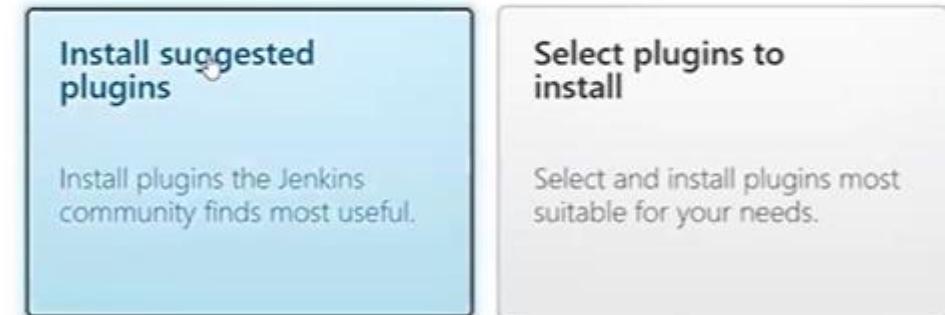
```
root@DESKTOP-SK00CDK:~# cat /var/lib/jenkins/secrets/initialAdminPassword
5e2a2f500a4142698da28af2747dd213
root@DESKTOP-SK00CDK:~# █
```

Installer Jenkins sous WSL

- Cliquer sur Install suggested plugins

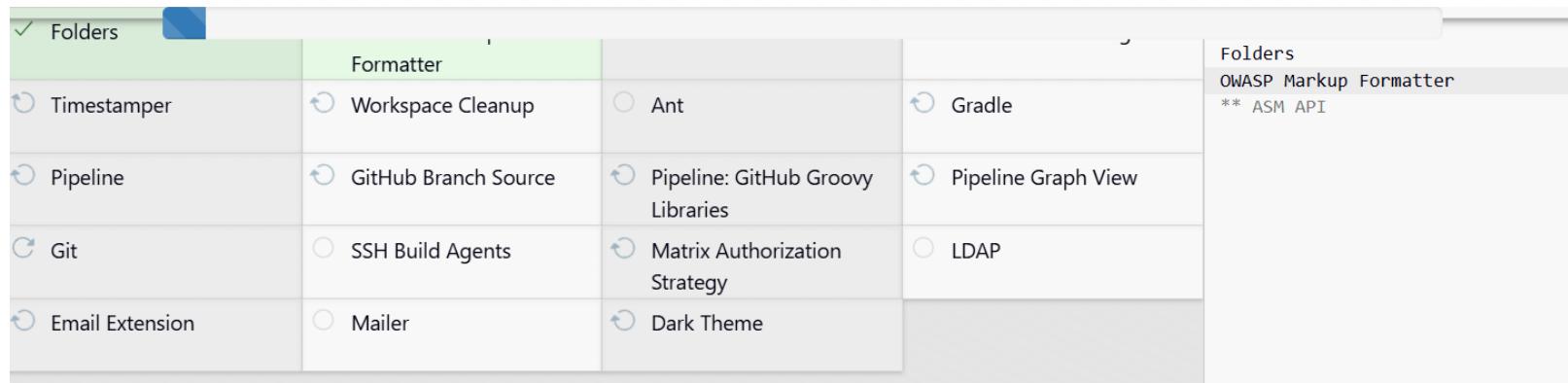
Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.



- L'installation des plugins proposée prend du temps....

Installation en cours...



Installer Jenkins sous WSL

- Compléter le formulaire
- Sauver et terminer l'URL

Créer le 1er utilisateur Administrateur

Nom d'utilisateur

Mot de passe

Confirmation du mot de passe

Configuration de l'instance

URL de Jenkins :

<http://localhost:8080/>

Bienvenue sur Jenkins



Jenkins

+ Nouveau Item

Ajouter

Historique des constructions

File d'attente des constructions



File d'attente des constructions vide

État du lanceur de compilations

0/2

Bienvenue sur Jenkins !

Vos jobs Jenkins seront affichés sur cette page. Pour commencer, vous pouvez mettre en place un build distribué ou commencer à créer un projet.

Commencer à créer votre projet

[Créer un job](#)



Configurer un build distribué

[Mettre en place un agent](#)



[Configurer un cloud](#)



Installer quelques plugins

- Afin de créer une chaîne d'intégration continue, nous allons installer les plugins suivants dans Jenkins (installez ces plugins sans redémarrer, puis redémarrez à la fin)
 - GIT plugin (normalement déjà installé mais vérifiez)
 - Maven integration
 - SonarGraph integration
 - SonarQube scanner

The screenshot shows the Jenkins plugin manager interface. At the top, there's a search bar labeled "Nom ↴". Below it, three plugins are listed, each with a checked checkbox:

- Maven Integration 3.26** (under "Outils de build")

This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.
- SonarQube Scanner 2.18** (under "Site externe/Intégration d'outils" and "Rapports de build")

This plugin allows an easy integration of [SonarQube](#), the open source platform for Continuous Inspection of code quality.
- Sonargraph Integration 5.0.2** (under "Site externe/Intégration d'outils", "Rapports de build", and "Autres actions après build")

This plugin integrates [Sonargraph](#) functionality into Jenkins, for Sonargraph versions 9 and 10.

Installer quelques plugins

- Après l'installation des plugins, il faut redémarrer le serveur Jenkins

```
root@DESKTOP-SKOOCDK:~# systemctl restart jenkins
```

Redémarrer Jenkins

...

En cours

→ [Revenir en haut de la page](#)

(vous pouvez commencer à utiliser les plugins installés dès maintenant)

→ Redémarrer Jenkins quand l'installation est terminée et qu'aucun job n'est en cours

Configuration de Jenkins - JDK

- Après installation de jdk sous WSL, il faut modifier la variable d'environnement JAVA_HOME comme suit :

```
root@DESKTOP-SK00CDK:/usr/lib/jvm# export JAVA_HOME="/usr/lib/jvm/java-21-openjdk-amd64/"  
root@DESKTOP-SK00CDK:/usr/lib/jvm# export PATH="$JAVA_HOME/bin:$PATH"  
root@DESKTOP-SK00CDK:/usr/lib/jvm# systemctl restart jenkins
```

- Puis sous Jenkins, il faut configurer Le JDK sous Tools :

The screenshot shows the Jenkins interface for managing tools. On the left, under the 'Tools' section, there are configuration options for Maven and global settings. On the right, a modal window titled 'Installations JDK' is open, showing the configuration for a new JDK named 'JDK21'. The 'JAVA_HOME' field is set to '/usr/lib/jvm/java-21-openjdk-amd64/'. A checkbox for 'Install automatically' is present.

Installations JDK ^ Edited

Ajouter JDK

JDK

Nom: JDK21

JAVA_HOME: /usr/lib/jvm/java-21-openjdk-amd64/

Install automatically ?

Jenkins / Administrer Jenkins / Tools

Tools

Configuration Maven

Fournisseur de réglages par défaut

Utiliser les réglages Maven par défaut

Fournisseur de réglages globaux par défaut

Utiliser les réglages globaux Maven par défaut

Configuration de Jenkins - Maven

- Rajoutez le chemin d'installation de Maven sous wsl

```
root@DESKTOP-SK00CDK:/# export M2_HOME="/usr/share/maven"  
root@DESKTOP-SK00CDK:/# export PATH="$M2_HOME/bin:$PATH"
```

```
root@DESKTOP-SK00CDK:/# systemctl restart jenkins
```

≡ **Maven**

Nom

M2_HOME

MAVEN_HOME

/usr/share/maven

Install automatically ?

Configuration de Jenkins - GIT

- Le GIT est déjà configuré sous Jenkins

Git installations

≡ **Git**

Name

Default

Path to Git executable ?

git

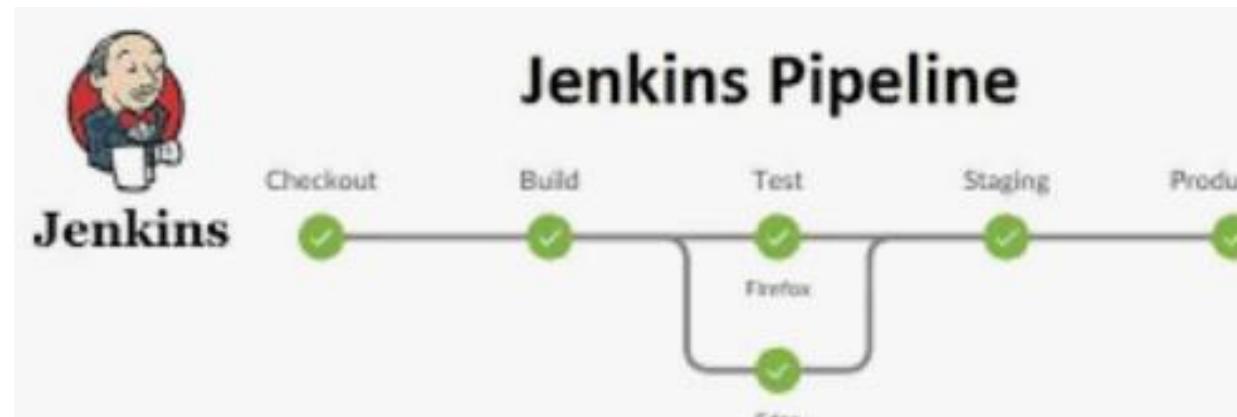
Install automatically ?

Add Git ▾

4. Gestion de jobs free-style sous Jenkins

Création de jobs sous Jenkins

- Il y a deux façons de configurer la compilation d'un projet :
 - **Méthode formulaire (freestyle)** : Vous pouvez configurer le projet en utilisant un formulaire à remplir
 - **Méthode pipeline** : l'alternative est d'utiliser un **pipeline**, qui configure le projet à l'aide de scripts (en utilisant **Groovy**). Cette approche permet également de paralléliser les étapes du projet et offre une interface plus conviviale pour lire les logs.



Créer notre premier job free-style

1. Créer un nouveau job de type free-style

Nouveau Item

Saisissez un nom

Git_Job

Select an item type



Construire un projet free-style

Job legacy polyvalent qui récupère l'état depuis un outil de gestion suivi d'étapes post-construction telles que l'archivage d'artefacts et

2. Crée un nouveau job de type free-style

Général

Description

This is a simple jenkins job

Créer notre premier job

3. Configurer les paramètres de GIT

Gestion de code source

Connect and manage your code repository to automatically pull t

Aucune

Git ?

Repositories ?

Repository URL ?

`https://github.com/Jamina-ENSI/Hello_example.git`

Credentials ?

Jamina-ENSI/*********

Branches to build ?

Branch Specifier (blank for 'any') ?

`*/main`

Créer notre premier job

- Le répository Hello_example contient un seul fichier Test.java :

The screenshot shows a GitHub repository page for 'Hello_example'. At the top, there's a navigation bar with icons for repository management, followed by the repository name 'Jamina-ENSI / Hello_example' and tabs for 'Code', 'Issues', 'Pull requests', and 'Actions'. Below this, the repository name 'Hello_example' is displayed with a 'Public' badge. A dropdown menu for the 'main' branch is open, showing 'main' and a 'Switch branch' option. It also shows '1 Branch' and '0 Tags'. There are two main sections at the bottom: one for 'Add files via upload' and another for the file 'Test.java'.

The screenshot shows the content of the 'Test.java' file within the repository. The file contains the following Java code:

```
1
2  public class Test {
3
4      public static void main(String[] args) {
5          // TODO Auto-generated method stub
6
7          System.out.println("Hello");
8
9      }
10 }
11 }
```

Créer notre premier job

4. Configurer les étapes du build et appliquer

Étapes du build

Automate your build process with ordered tasks like code com

≡ Exécuter un script shell ?

Commande

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

```
javac Test.java  
java Test
```

Exécuter notre premier job

Jenkins / Git_Job

- État
- </> Modifications
- Répertoire de travail
- Lancer un build
- Configurer
- Supprimer Projet
- Renommer

Builds

Filter

Today

#7 16:47

Supprimer Projet

Renommer

Builds

Filter

Today

#7 16:47

Modifications

- Sortie de la console
- Informations de la construction
- Supprimer le build "#7"
- Timings
- Git Build Data

Jenkins / Git_Job / #7 / Sortie de la console

État

</> Modifications

Sortie de la console

Informations de la construction

Started by user Amina Jarraya
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Git_Job
The recommended git tool is: NONE

Analyser la sortie du job

```
Started by user Amina Jarraya
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Git_Job
The recommended git tool is: NONE
using credential 1fb93111-3a24-405e-915f-ffeeb1f80b68
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Git_Job/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Jamina-ENSI>Hello_example.git # timeout=10
Fetching upstream changes from https://github.com/Jamina-ENSI>Hello_example.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/Jamina-ENSI>Hello_example.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 8cac4f30783c4e47b58f3ffe730466f332f74226 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 8cac4f30783c4e47b58f3ffe730466f332f74226 # timeout=10
Commit message: "Add files via upload"
> git rev-list --no-walk 8cac4f30783c4e47b58f3ffe730466f332f74226 # timeout=10
[Git_Job] $ /bin/sh -xe /tmp/jenkins6571700839735225603.sh
+ javac Test.java
+ java Test
Hello
Finished: SUCCESS
```

Rajouter un autre script

- Vous pouvez rajouter d'autres scripts dans votre job :



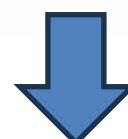
```
+ javac Test.java
+ java Test
Hello
[Git_Job] $ /bin/sh -xe /tmp/jenkins7850525868335514129.sh
+ mvn --version
[1mApache Maven 3.8.7[m
Maven home: /usr/share/maven
Java version: 21.0.8, vendor: Ubuntu, runtime: /usr/lib/jvm/java-21-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.6.87.2-microsoft-standard-ws12", arch: "amd64", family: "unix"
Finished: SUCCESS
```

☰ Exécuter un script shell ?

Commande

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

mvn --version



La construction périodique

- Sélectionnez l'action pour déclencher périodiquement le flux d'intégration continue.
- Cela peut être un processus qui s'exécute régulièrement, par exemple chaque minute, ou qui vérifie si une nouvelle version a été poussée sur GIT

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

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

Planning ?

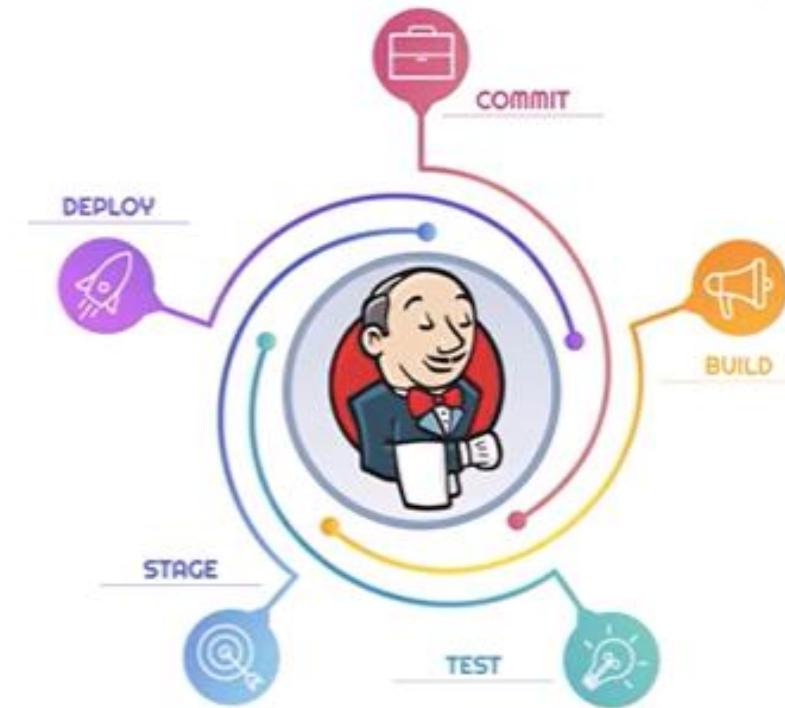


Builds		
<input type="text"/> Filter /		
Today		
▼		
<input checked="" type="checkbox"/>	#11	17:33
<input checked="" type="checkbox"/>	#10	17:32
<input checked="" type="checkbox"/>	#9	17:31

5. Gestion de pipelines sous Jenkins

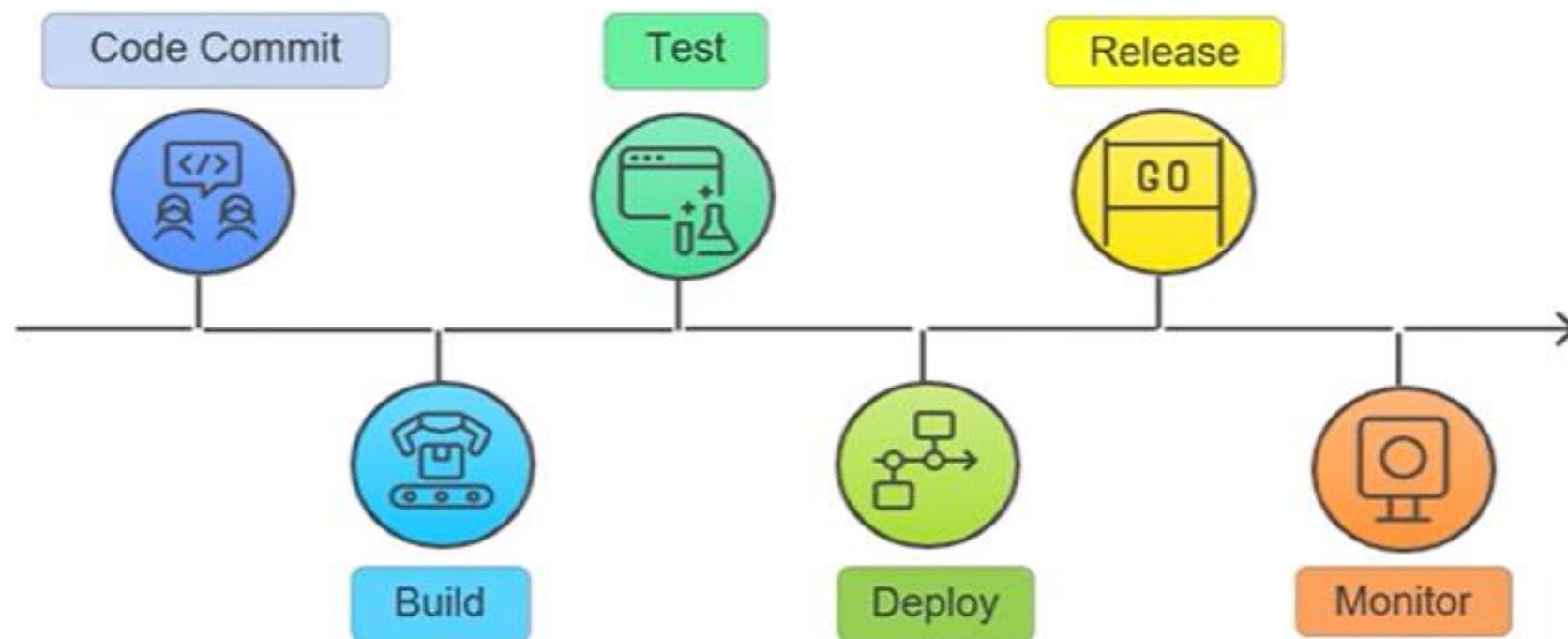
C'est quoi un delivery pipeline ?

- Un **pipeline Jenkins**, dans le contexte de l'intégration continue et de la livraison continue (CI/CD), est une série d'étapes automatisées qui permettent de construire, tester et déployer un logiciel de manière efficace et répétable.
- Il s'agit d'un ensemble de tâches séquentielles ou parallèles, définies dans un script (généralement **Groovy**) et exécutées par Jenkins pour automatiser le processus de développement logiciel.
- En d'autres termes, un pipeline Jenkins est une représentation de votre workflow de développement, de la création du code à sa mise en production, avec des étapes clairement définies et automatisées.

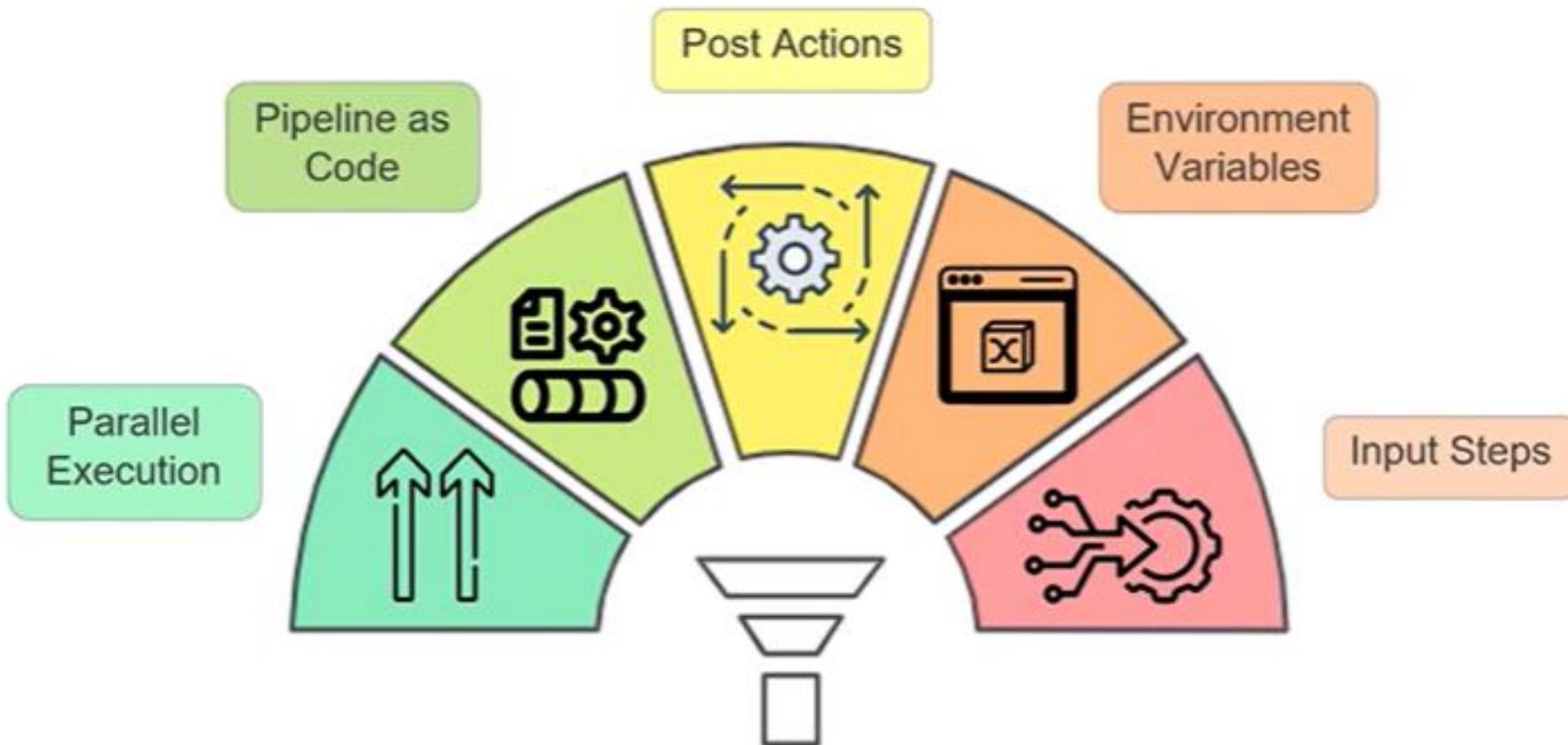


Le pipeline de la livraison continue

- Un ***pipeline de livraison continue*** est une série de processus automatisés destinés à livrer de nouveaux logiciels. Il s'agit d'une implémentation du paradigme « Continuous Everything », où les builds, les tests et les déploiements automatisés sont orchestrés pour former un workflow de livraison unique



Jenkins pipeline features



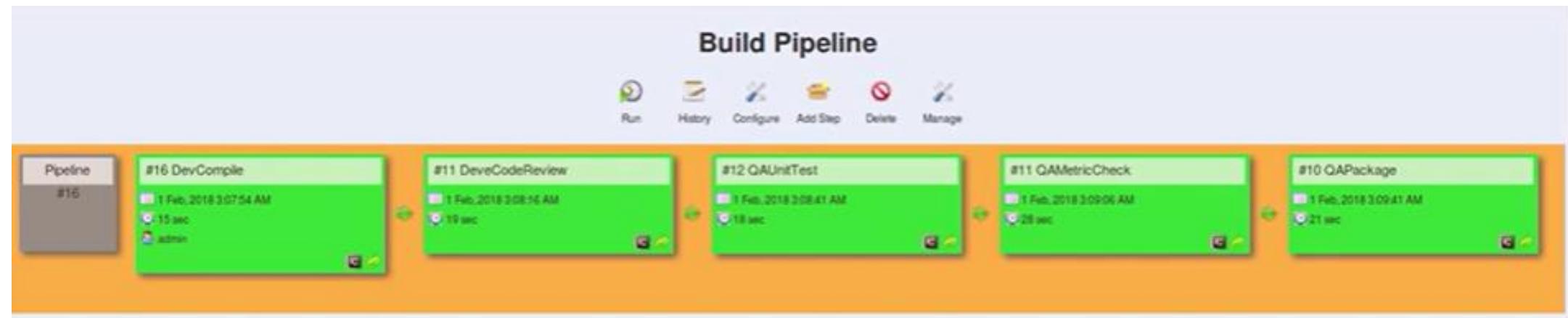
Une vue pipeline

Yellow blocks represent the currently executing job

Blue blocks represent the jobs waiting for execution

Green blocks represent the successful build of job

Red blocks represent failure while building a job



Installer plugin – build pipeline

- Installer le plugin build pipeline

The screenshot shows the Jenkins Plugins page. On the left, there's a sidebar with icons for Updates, Available plugins (which is selected and highlighted in grey), Installed plugins, Advanced settings, and Download progress. The main area has a search bar at the top with the text "build pip". Below the search bar, there are two tabs: "Install" and "Name ↓". A list of plugins is shown, with "Build Pipeline 2.0.2" selected and marked with a blue checkmark. Underneath the plugin name, there are three categories: User Interface, Build Tools, and Other Post-Build Actions. A descriptive text block states: "This plugin renders upstream and downstream connected jobs that typically form a build pipeline. In addition, it offers the ability to define manual triggers for jobs that require intervention prior to execution, e.g. an approval process outside of Jenkins." At the bottom of the plugin card, there's a red warning box containing the text: "Warning: This plugin version may not be safe to use. Please review the following security notices:" followed by a bullet point: "• [Stored XSS vulnerability](#)".

Connecter deux jobs sous Jenkins

- Créez deux jobs : « 1.Compile » et « 2. Execute ». Le premier exécute le shell « echo « compiler le code» et le 2^{ème} exécute le shell «echo « executer le code »

The screenshot shows the Jenkins dashboard with two jobs listed:

S	M	Nom du projet ↓	Dernier succès	Dernier échec	Dernière durée
...	...	1. Compile	s. o.	s. o.	ND
...	...	2. Execute	s. o.	s. o.	ND

Below the dashboard, the configuration for the '2. Execute' job is shown:

Jenkins / 2. Execute / Configuration

Configurer

- Général
- Gestion de code source
- Triggers** (selected)
- Environment
- Étapes du build
- Actions à la suite du build

Triggers

Set up automated actions that start your build based on specific events or schedule.

- Déclencher les builds à distance (Par exemple, à partir d'un autre job)
- Construire après le build sur d'autres projets ?

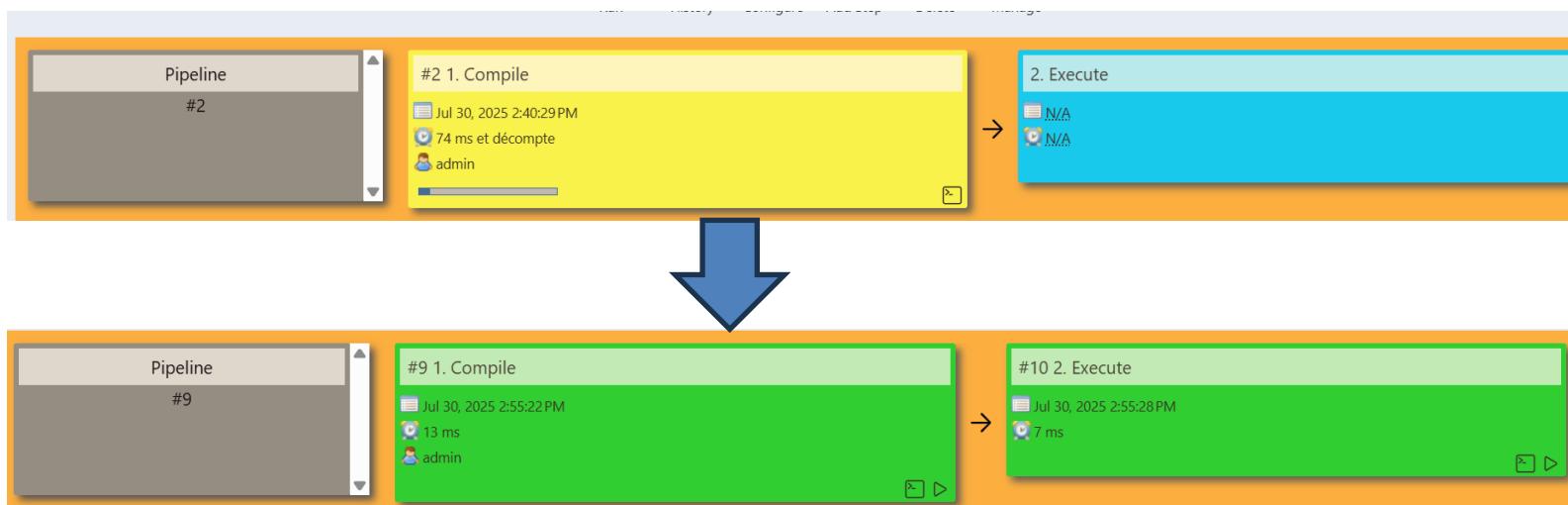
Projet à surveiller

1. Compile,

- Déclencher que si la construction est stable
- Déclencher même si la construction est instable
- Déclencher même si la construction échoue
- Always trigger, even if the build is aborted

Création d'une vue pipeline

- Créez une vue pipeline, sélectionner le job initial puis exécutez :



+ Nouveau Item

Historique des constructions

Relations entre les builds



Nouvelle Vue

Tous +

New view

Nom de la vue

BuildPipeline

Type

Build Pipeline View

Shows the jobs in a build pipeline v
through are shown as a row in the v

Ma vue

Cette vue affiche automatiquement

Vue liste

Montre les jobs dans une simple lis

Create



Build Pipeline View Title

Build pipeline project

Pipeline Flow

Layout

Based on upstream/downstream relationship

This layout mode derives the pipeline structure ba
only out-of-the-box supported layout mode, but

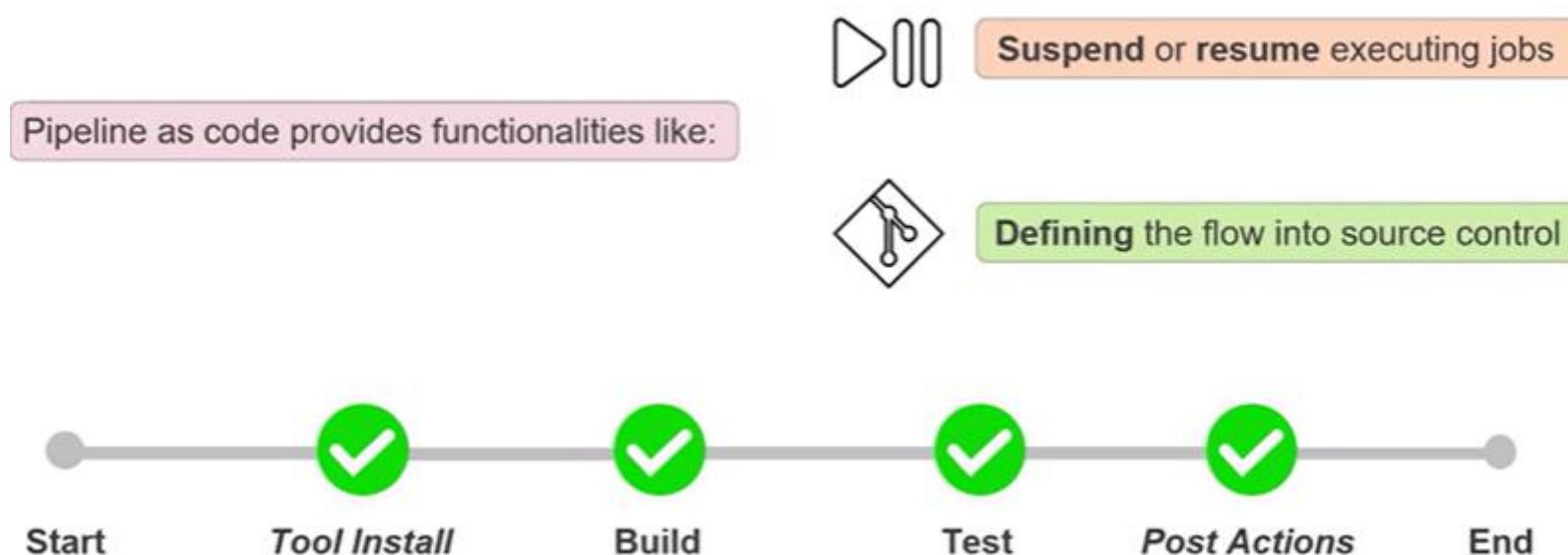
Upstream / downstream config

Select Initial Job ?

1. Compile

Pipeline as code sous Jenkins

- En Jenkins, "Pipeline as Code" (Pipeline en tant que code) signifie définir votre processus de livraison continue (CI/CD) dans un fichier texte, appelé **Jenkinsfile**, qui est ensuite stocké et versionné avec votre code source.
 - Cela permet de traiter le pipeline comme n'importe quel autre code, le rendant ainsi plus facile à gérer, partager et réviser.
 - **Jenkinsfile**: C'est un fichier texte qui contient la logique de votre pipeline, généralement écrit dans le langage **Groovy** avec une syntaxe spécifique à Jenkins.



Types de pipeline sous Jenkins

- Jenkins offre deux types de pipelines pour orchestrer des workflows d'intégration continue et de livraison continue (CI/CD) : les pipelines déclaratifs et les pipelines scriptés.
 - **Les pipelines déclaratifs** offrent une syntaxe plus simple et plus lisible
 - **Les pipelines scriptés** offrent plus de flexibilité grâce à l'utilisation du langage Groovy.

Two types of syntax are used for defining a JenkinsFile in Jenkins Pipeline.



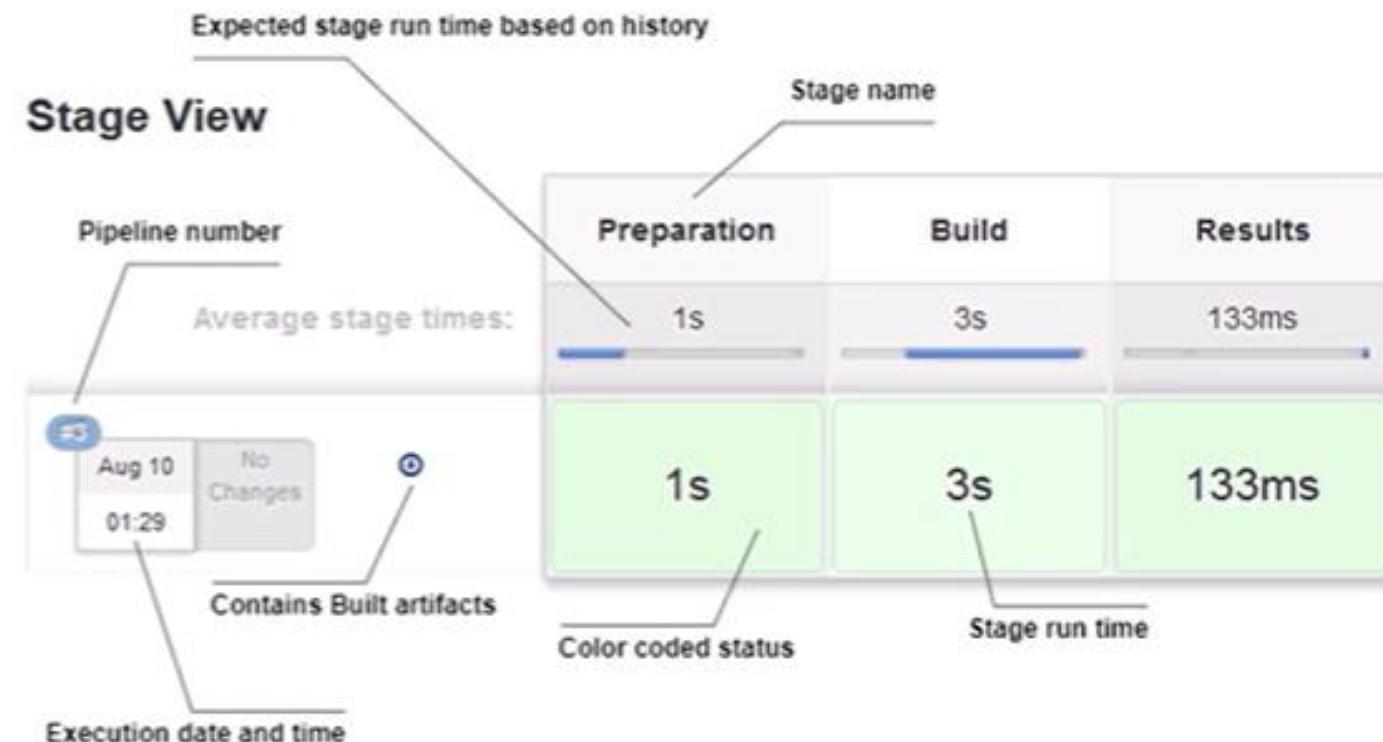
Declarative pipeline syntax provides the ability to control all aspects of a pipeline execution.

Scripted Jenkins pipeline syntax runs on the Jenkins master with the help of a lightweight executor.

Jenkins pipeline Stage view

Jenkins Stage View helps to visualize the **progress** of various stages of the Pipeline in **real time**.

The **Stage View** page will look something like the following screenshot:



Le pipeline déclarative

- **Syntaxe simplifiée:** Les pipelines déclaratifs utilisent une syntaxe plus concise et structurée, ce qui les rend plus faciles à lire et **à maintenir**.
- **Structure définie:** Ils suivent une structure prédéfinie avec des blocs tels que pipeline, agent, stages, steps.
- **Facilité d'utilisation:** Ils sont généralement plus conviviaux pour les utilisateurs qui ne sont pas familiers avec Groovy.
- **Limitations:** Moins flexibles pour des besoins de personnalisation complexes.

```
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any ①
    stages {
        stage('Build') { ②
            steps {
                // ③
            }
        }
        stage('Test') { ④
            steps {
                // ⑤
            }
        }
        stage('Deploy') { ⑥
            steps {
                // ⑦
            }
        }
    }
}
```

Le pipeline scripté

- **Flexibilité**: Ils sont écrits en Groovy, offrant une plus grande flexibilité pour des tâches complexes et des personnalisations.
- **Contrôle total**: Permettent un contrôle précis sur chaque étape du pipeline.
- **Complexité**: Nécessitent une connaissance de Groovy et peuvent être plus difficiles à maintenir pour les utilisateurs moins expérimentés.

Jenkinsfile (Scripted Pipeline)

```
node { ①
    stage('Build') { ②
        // ③
    }
    stage('Test') { ④
        // ⑤
    }
    stage('Deploy') { ⑥
        // ⑦
    }
}
```

6. Création des pipelines déclaratives sous Jenkins

Exemple de création d'un pipeline déclarative

Jenkins / Tous / Nouveau Item

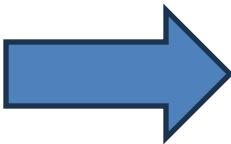
Nouveau Item

Saissez un nom

Select an item type

-  Construire un projet free-style
Job legacy polyvalent qui récupère l'état suivi d'étapes post-construction telles que les tests et les notifications.
-  Construire un projet maven
Construit un projet avec maven. Jenkins gère la configuration. Cette fonctionnalité est recommandée pour les projets utilisant Maven.
-  Pipeline
Organise des activités de longue durée en utilisant des pipelines (anciennement connues comme les workflows). Cela facilite l'organisation de tâches de type librairie.

OK



Configurer

Definition

Pipeline script

Script ?

```
1 v pipeline {  
2     agent any
```

Général

Triggers

Pipeline

Le script déclarative

Script ?

```
1 ` pipeline {
2     agent any
3     tools {
4         jdk 'JDK21'
5     }
6     stages {
7         stage('Checkout code')
8         {
9             steps {
10                 git branch: 'main', url: 'https://github.com/Jamina-ENSI/Hello_example.git'
11             }
12         }
13         stage('Compile code')
14         {
15             steps {
16                 sh 'javac Test.java'
17             }
18         }
19         stage('Execute code')
20         {
21             steps {
22                 sh 'java Test '
23             }
24         }
25     }
}
```

Exécution du pipeline

Jenkins / CI-build / Stages

Stages

30 juillet 2025

#4
15:38 - 4.2s



Jenkins / CI-build / #5 / Pipeline Overview

✓ < #5

Manually run by Amina Jarraya Démarrée il y a 25 s. Queued 2 ms Took 4.5 s

Graph



Start	Tool Install	Checkout code	Compile code	Execute code	End
○	✓	✓	✓	✓	○

Utilisation de jenkinsfile

- Créer jenkins file sous votre projet

```
root@DESKTOP-SKOOCDK:/mnt/c/mydevops>Hello# git init
```

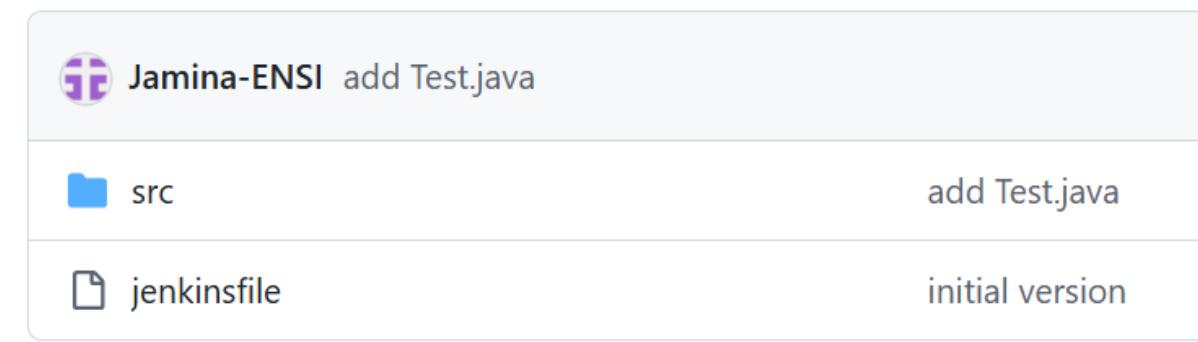
```
root@DESKTOP-SKOOCDK:/mnt/c/mydevops>Hello# vi jenkinsfile
```

- Tapez vi mon_fichier.txt.
- Pour entrer en mode édition, appuyez sur la touche i.
- Tapez votre texte.
- Pour enregistrer et quitter, appuyez sur la touche Échap, puis tapez :wq et appuyez sur Entrée.

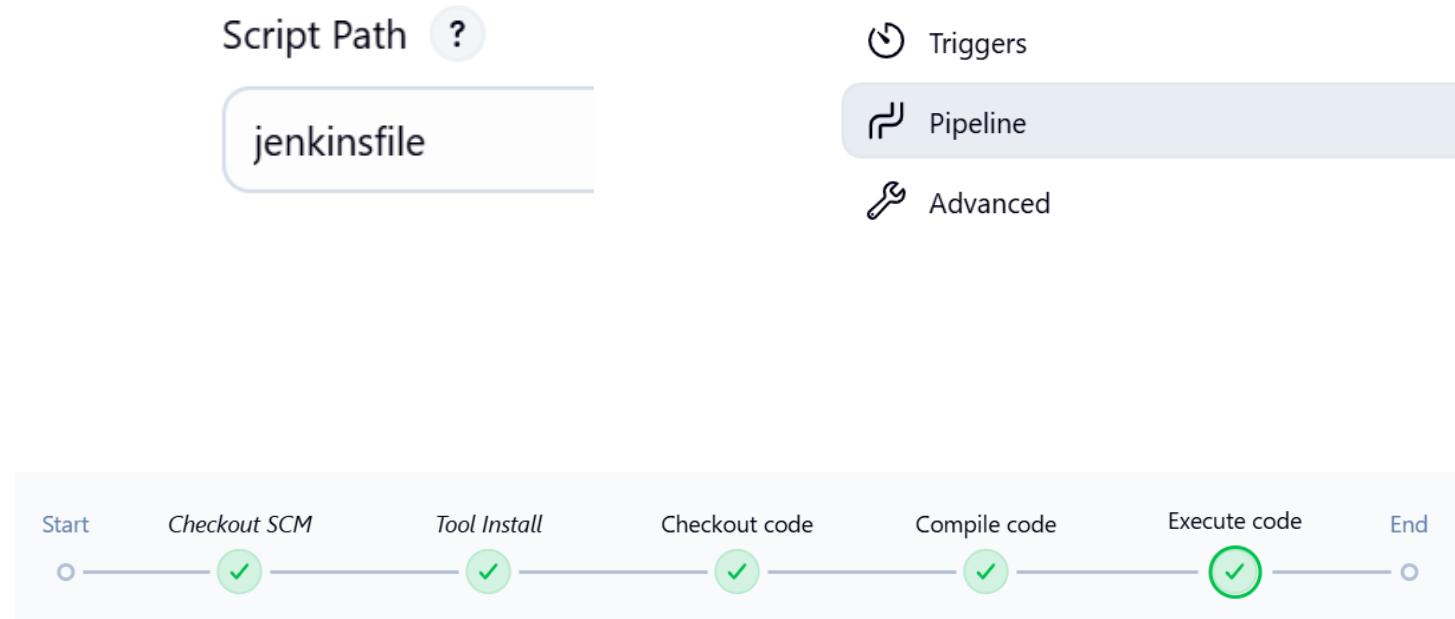
```
pipeline {
    agent any
    tools {
        jdk 'JDK21'
    }
    stages {
        stage('Checkout code')
        {
            steps {
                git branch: 'main', url: 'https://github.com/Jamina-ENSI/Hello_example.git'
            }
        }
        stage('Compile code')
        {
            steps {
                sh 'javac Test.java'
            }
        }
        stage('Execute code')
        {
            steps {
                sh 'java Test'
            }
        }
    }
}
```

Utilisation de jenkinsfile

- Push le fichier Jenkinsfile dans le repository



- Créer un nouveau pipeline en spécifiant que le pipeline est à partir de GIT



The Jenkins Pipeline configuration screen. The 'Definition' section is set to 'Pipeline script from SCM'. The 'SCM' section is set to 'Git'. The 'Repositories' section shows a single repository with the URL 'https://github.com/Jamina-ENSI>Hello_example.git'. The 'Credentials' section shows a credential entry for 'Jamina-ENSI/********'. The 'Definition' section also shows a note: 'Pipeline script from SCM'.

7. Github Webhook et Jenkins

Github Webhook

- Un **webhook GitHub** est un mécanisme qui permet à GitHub de notifier un serveur externe (Jenkins) lorsqu'un événement spécifique se produit sur un dépôt ou sur un compte.
- Lorsque vous créez un nouveau pipeline « GithubWebhook », cochez l'option suivante dans la rubrique Triggers

Triggers

Set up automated actions that start your build based on specific events, like

- Construire après le build sur d'autres projets ?
- Construire périodiquement ?
- GitHub hook trigger for GITScm polling ?
- Scrutation de l'outil de gestion de version ?
- Déclencher les builds à distance (Par exemple, à partir de scripts) ?

Definition

Pipeline script from SCM

Script Path ?

jenkinsfile

SCM ?

Git

Repositories ?

Repository URL ?

`https://github.com/Jamina-ENSI>Hello_example.git`

Credentials ?

Jamina-ENSI/*****

The screenshot shows the Jenkins Pipeline configuration page. On the left, under 'Triggers', the 'GitHub hook trigger for GITScm polling' option is checked. On the right, the 'Definition' section shows 'Pipeline script from SCM' selected, with 'SCM' set to 'Git' and 'Repository URL' set to 'https://github.com/Jamina-ENSI>Hello_example.git'. Under 'Credentials', it shows 'Jamina-ENSI/*****'. The 'Script Path' is set to 'jenkinsfile'.

Configurer Webhook sous Github

- Dans votre repository, allez sur **Settings → Webhooks → add webhook**

Webhooks / Manage webhook

The screenshot shows the GitHub 'Webhooks / Manage webhook' interface. At the top, there are two tabs: 'Settings' (selected) and 'Recent Deliveries'. Below the tabs, a note says: 'We'll send a POST request to the URL below with details of any subset you'd like to receive (JSON, x-www-form-urlencoded, etc). M [documentation](#)'. The 'Payload URL *' field contains 'https://a38f4db380b5.ngrok-free.app/github-webhook/'. The 'Content type *' field is set to 'application/json'. On the right side, under 'SSL verification', 'Enable SSL verification' is selected. Under 'Which events would you like to trigger this webhook?', 'Just the push event.' is selected. A checked checkbox labeled 'Active' is present. A green 'Add webhook' button is at the bottom. At the very bottom, there is a summary bar with 'Settings' and 'Recent Deliveries' tabs, a green checkmark icon, a blue cube icon, the ID 'dadb9ce2-6d56-11f0-95c3-95976eb63ce8', and buttons for 'ping' and 'redelivery'.

SSL verification

By default, we verify SSL certificates when delivering payloads.

Enable SSL verification Disable (not recommended)

Which events would you like to trigger this webhook?

Just the push event.

Send me everything.

Let me select individual events.

Active

We will deliver event details when this hook is triggered.

Add webhook

Settings Recent Deliveries

✓ dadb9ce2-6d56-11f0-95c3-95976eb63ce8 ping redelivery

Utiliser ngrok pour créer une adresse publique

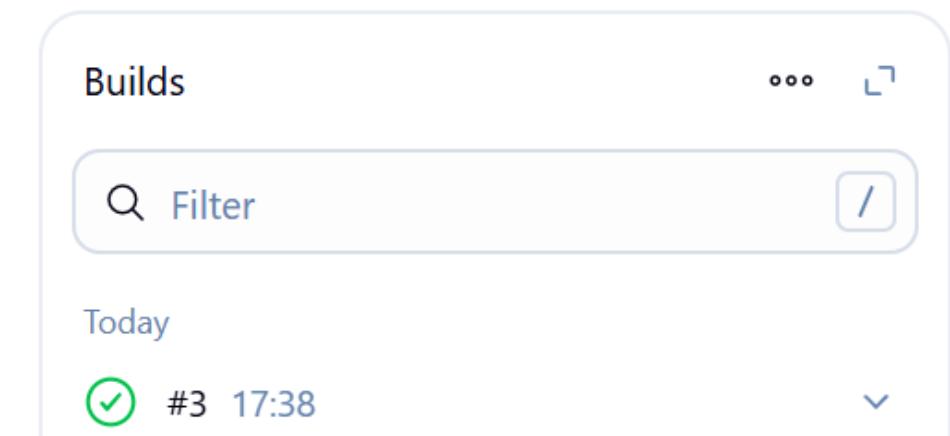
- Install ngrok sur windows (il se trouve dans microsoft app store)
- Ensuite tapez ngrok http 8080

Session Status	online
Account	Jamina-ENSI (Plan: Free)
Version	3.24.0-msix
Region	Europe (eu)
Latency	63ms
Web Interface	http://127.0.0.1:4040
Forwarding	https://a38f4db380b5.ngrok-free.app → http://loc
Connections	ttl opn rt1 rt5 p50 p90 14 0 0.00 0.02 0.41 38.88

Notifier Jenkins par GitHub

- En créant avec succès le webhook sur github et si vous essayez de modifier le fichier jenkins sur github, un build se crée automatiquement dans le pipeline GithubWebhook :
- Rajoutez ce stage dans jenkinsfile sous github et sauvegarder == ça va créer automatiquement un build dans votre pipeline sous Jenkins.

```
25      stage('display message')  
26      {  
27          steps {  
28              echo 'Hello from github '  
29          }  
30      }
```

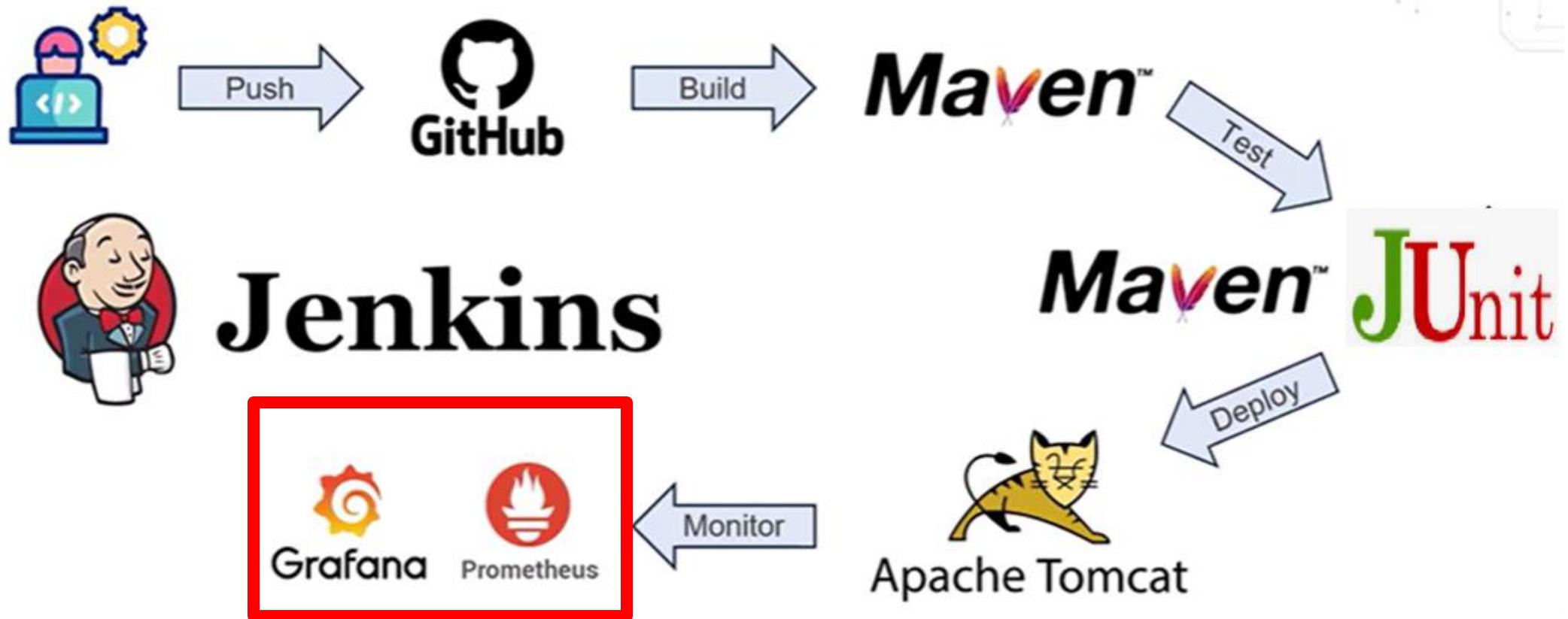


8. Mise en place d'un pipeline CI/CD (Github, maven, Junit, Apache Tomcat)

Pipeline CI/CD pour un micro-service

- Réalisez le pipeline **CI/CD (Continuous Integration/Continuous Delivery)** pour le micro-service country-service comme suit :
 1. Faites le **push** du micro-service dans votre **github**
 2. Utilisez un **github webhook** pour lancer le pipeline à chaque modification au niveau du github
 3. Créez un **Jenkinsfile** dans votre github. Ce pipeline décrit les étapes de compilation, de tests et de déploiement (Livraison continue) avec le serveur Apache Tomcat.
 4. Créez un job de type pipeline sous Jenkins qui automatise l'intégration des différents outils

Pipeline CI/CD pour un micro-service



On va le voir dans le dernier chapitre

Jenkins file



Jenkins

/ CI_Country / Pipeline Syntax

① Online Documentation

① Examples Reference

① IntelliJ IDEA GDSL

Sample Step

junit: Archive JUnit-formatted test results

junit ?

XML des rapports de test

Une configuration du type Fileset 'includ
'myproject/target/test-reports/*.xml'. Le r

target/surfice-reports/*.xml

Generate Pipeline Script

junit 'target/surfice-reports/*.xml'

```
pipeline {  
    agent any  
    tools {  
        maven 'mymaven'  
    }  
    stages {  
        stage('Checkout code')  
        {  
            steps {  
                git branch: 'master', url: 'https://github.com/Jamina-ENSI/Country-service.git'  
            }  
        }  
        stage('Compile code')  
        {  
            steps {  
                sh 'mvn compile'  
            }  
        }  
        stage('Test code')  
        {  
            steps {  
                sh 'mvn test'  
            }  
            post{  
                success {  
                    junit allowEmptyResults: true, testResults: '**/target/surfice-reports/*.xml'  
                }  
            }  
        }  
        stage('Package code')  
        {  
            steps {  
                sh 'mvn package'  
            }  
        }  
    }  
}
```

3. Mise en place d'un pipeline CI/CD
(Github, maven, Junit,SonarQube,
Nexus Repository, Apache Tomcat)

Installer le plugin SonarQube scanner

Jenkins / Administrer Jenkins / Plugins

Plugins

Mises à jour 48

Plugins disponibles

Plugins installés

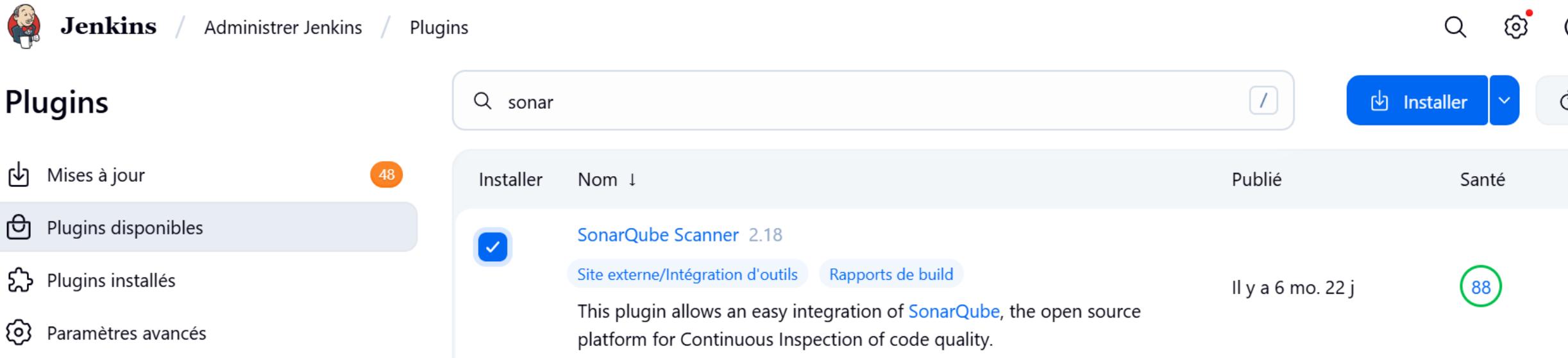
Paramètres avancés

sonar

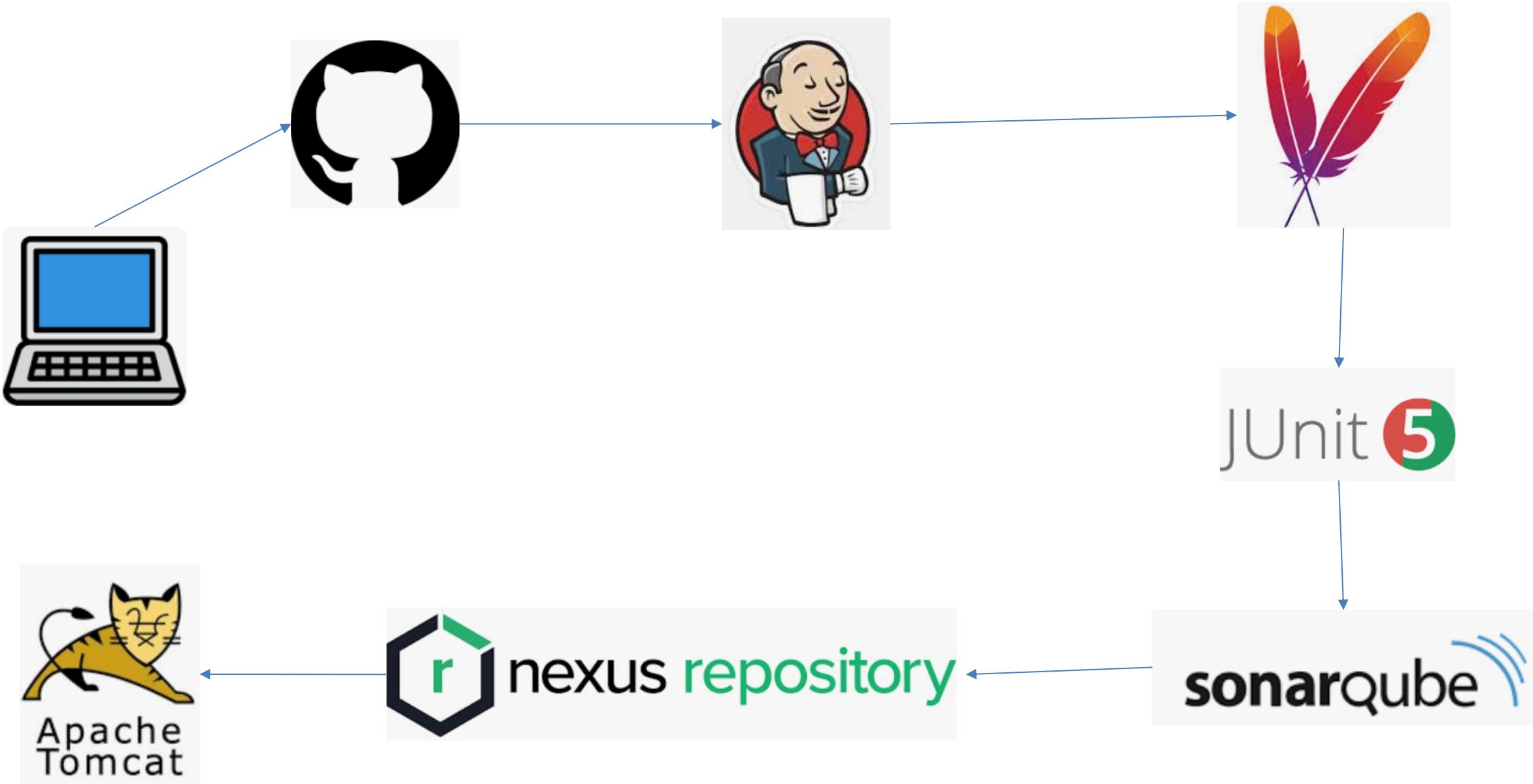
Installer Nom ↓ Publié Santé

SonarQube Scanner 2.18 Site externe/Intégration d'outils Rapports de build Il y a 6 mo. 22 j 88

This plugin allows an easy integration of [SonarQube](#), the open source platform for Continuous Inspection of code quality.



Pipeline CI/CD à mettre en place



Configurer Sonarqube Server dans Jenkins



Jenkins

/ Administrer Jenkins

/ System

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server

Environment variables

Installations de SonarQube

[Liste des installations de SonarQube](#)

Nom

MySonarQubeServer

URL du serveur

Par défaut à <http://localhost:9000>

http://localhost:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

sonarqubePWD

Configurer Sonarqube Server dans Jenkins



localhost:9000/account/security



⚠️ Embedded database should be used for evaluation purposes only. It doesn't support scaling.



SonarQube
community

Projects

Issues

Rules

Quality Profiles

Quality Gates

- Générer un token utilisateur

A

Administrator

Profile

Security

Notifications

Projects

Generate Tokens

Name

Type

Expires in

Jenkins

User Token

30 days

Generate

Configurer Sonarqube Server dans Jenkins

- Générer un identifiant sonarqube

Jenkins Credentials Provider: Jenkins

Identifiants globaux (illimité) ▾

Type

Secret text ▾

Portée ?

Global (Jenkins, agents, items, etc...) ▾

Secret

.....

ID ?

sonarqubePWD

Configurer SonarqubeScanner



Jenkins

Administrer Jenkins

Tools

≡ SonarQube Scanner

Name

MySonarQubeScanner



Install automatically



≡ Installer depuis Maven Central

Version

SonarQube Scanner 7.2.0.5079

Ajouter un installateur ▾

Le script Groovy

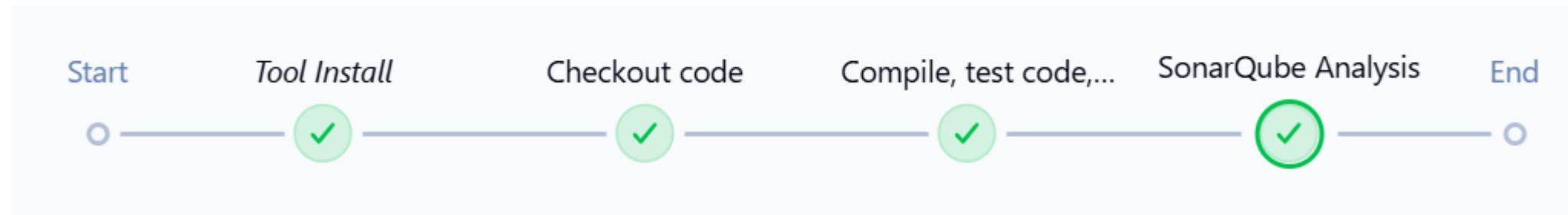
Script ?

```
1< pipeline {
2     agent any
3     tools {
4         maven 'mymaven'
5     }
6     stages {
7         stage('Checkout code')
8     {
9         steps {
10            git branch: 'master', url: 'https://github.com/Jamina-ENSI/Country-service.'
11        }
12    }
13    stage('Compile, test code, package in war file and store it in maven repo')
14    {
15        steps {
16            sh 'mvn clean install'
17        }
18    post{
19        success {
20            junit allowEmptyResults: true, testResults: '**/target/surfice-reports
21        }
22    }
23 }
```

Le script Groovy

```
24          stage('SonarQube Analysis')
25 {
26     steps {
27         withSonarQubeEnv(installationName: 'MySonarQubeServer', credentialsId: 'sonar')
28             sh "mvn sonar:sonar -Dsonar.projectKey=country-service -Dsonar.projectName=country-service"
29     }
30 }
31 }
32 }
33 }
```

Résultat



Jenkins

/ CI_Sonarqube



Status



CI_Sonarqube

</> Changes

▷ Lancer un build

⚙ Configurer

trash Supprimer Pipeline

Wi-Fi SonarQube

Stages

Renommer

Pipeline Syntax

This is an example of a CI pipeline using Sonarqube.

SonarQube Quality Gate

country-service Passed

server-side processing: Success

Liens permanents

- Dernier build (#4), il y a 3 mn 56 s
- Dernier build stable (#4), il y a 3 mn 56 s
- Dernier build avec succès (#4), il y a 3 mn 56 s
- Dernier build en échec (#2), il y a 11 mn

Travail à faire

- Vous allez compléter le pipeline pour :
 - Stocker les fichiers .war ou .jar générés par Maven dans un dépôt **nexus**. Vous pouvez même stocker les rapports générés par Sonarqube (cnes report).
 - Déployer le fichier .war dans le serveur **Tomcat**.
- **Nexus Repository** : une plateforme qui permet de stocker, gérer et distribuer des artefacts logiciels, tels que des fichiers JAR, WAR, des images Docker, des paquets NPM, etc. <https://help.sonatype.com/en/download.html>

➤ "Apprendre par le projet, c'est découvrir par l'action, créer par la compréhension, et réussir par la persévérance."



amina.jarraya@ensi-uma.tn

ENSI Manouba