

# Pipelines

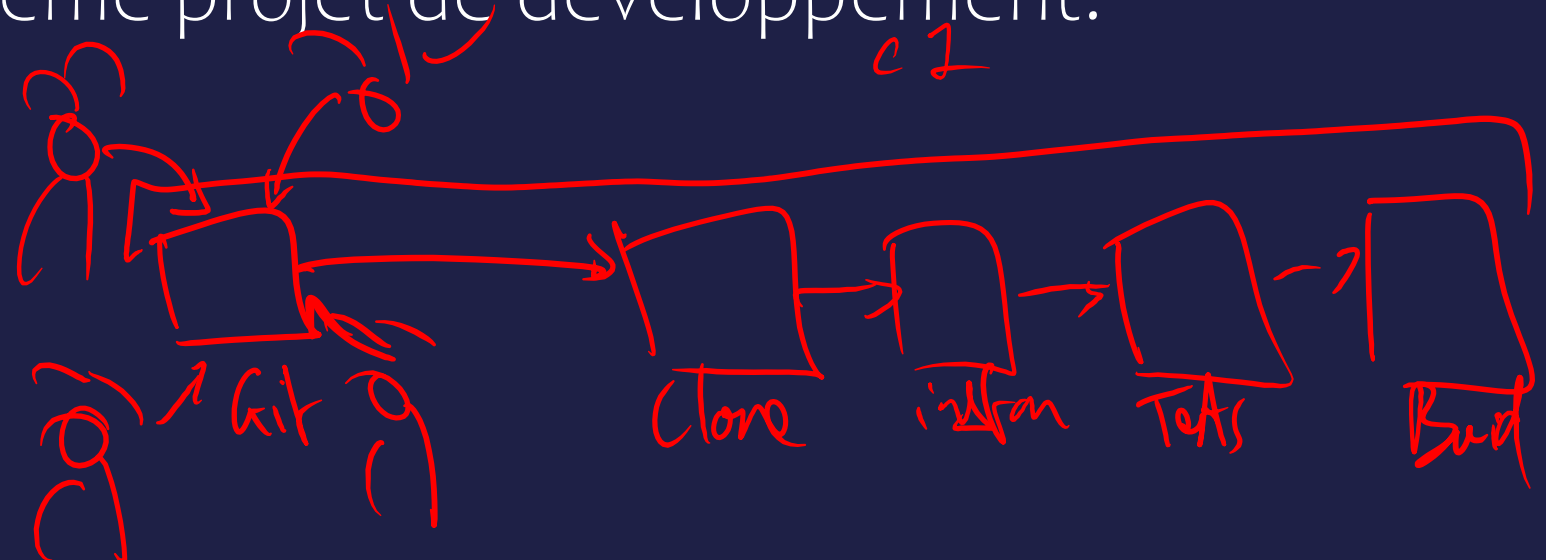
Comprendre le principe de pipelines



Compétence demandée :  
**Comprendre le principe de pipelines**

CI

L'intégration continue (CI) désigne la pratique qui consiste à automatiser l'intégration des changements de code réalisés par plusieurs contributeurs dans un seul et même projet de développement.



# Que serait le monde sans CI ?



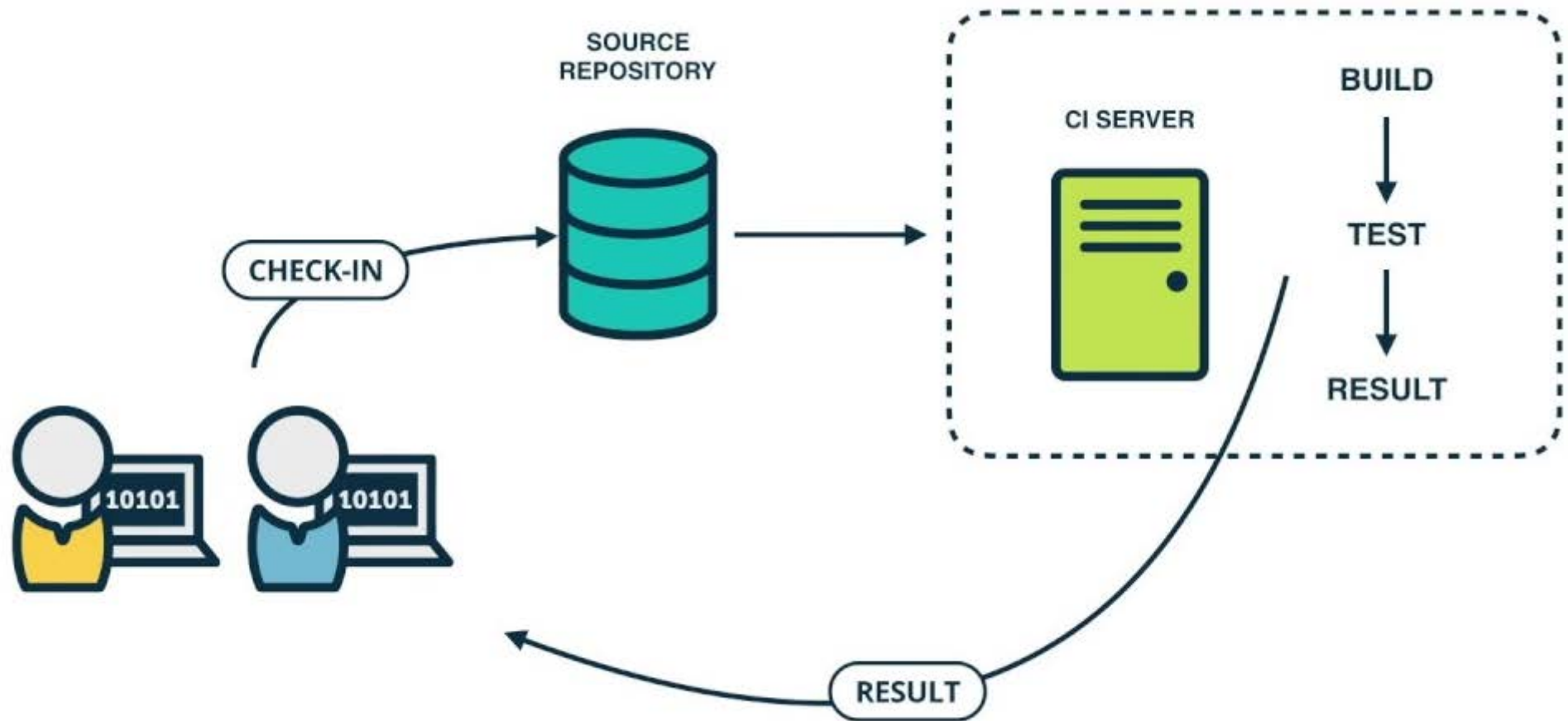
Sans elle, les développeurs doivent **se coordonner manuellement** et communiquer lorsqu'ils contribuent au code du produit final. Cette coordination s'étend au-delà des équipes de développement et **touche aussi les opérations et le reste de l'organisation.**

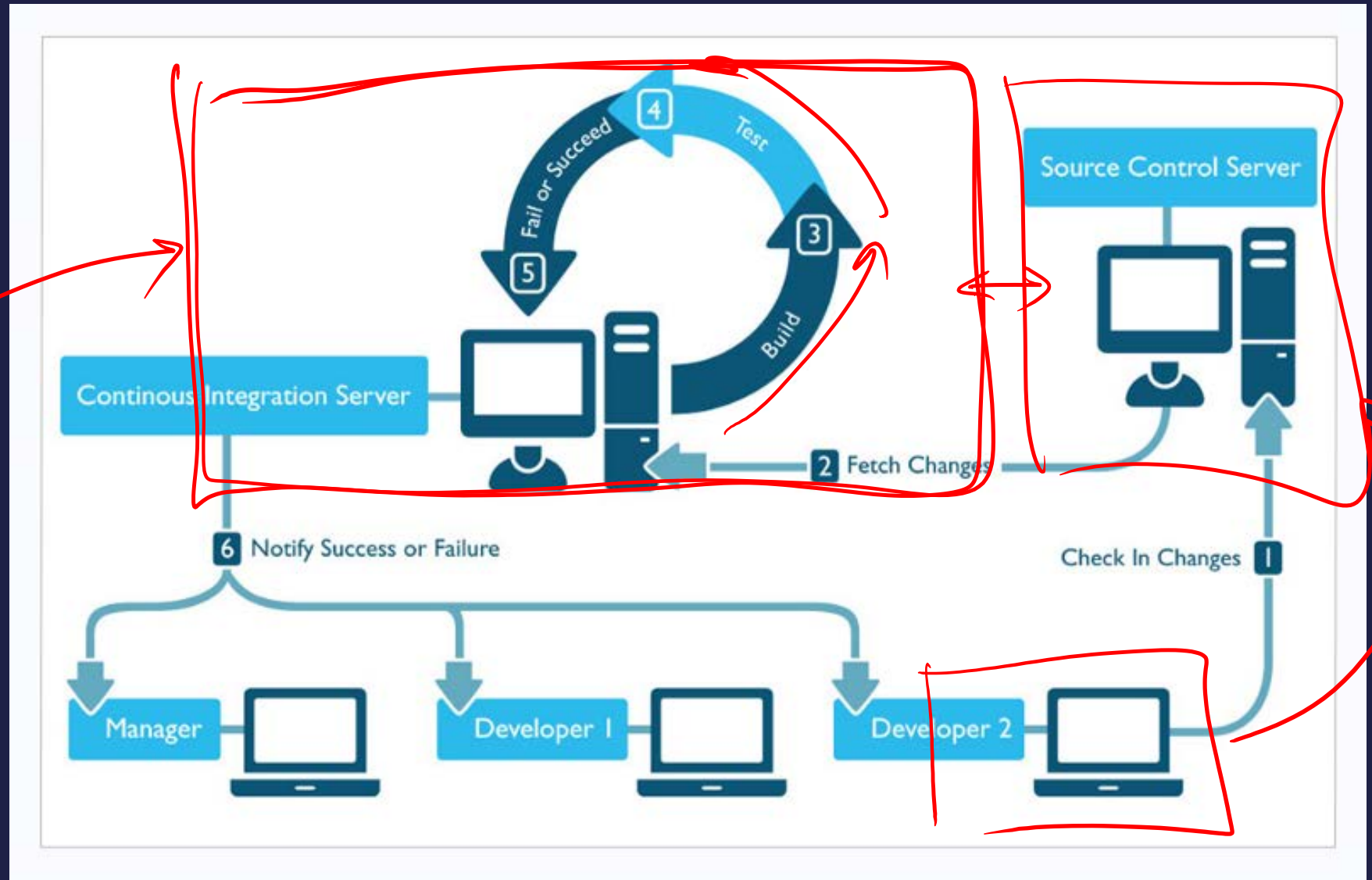
Dans un environnement sans CI, la communication peut devenir une tâche de synchronisation complexe et intriquée, ce qui ajoute des coûts d'administration inutiles aux projets.

Les livraisons de code sont ralenties et les taux de défaillance augmentent, car les développeurs doivent gérer les intégrations de façon sensible et réfléchie.

Il s'agit d'une **bonne pratique DevOps** principale, permettant aux développeurs de logiciels de **merger fréquemment des changements de code** dans un dépôt central où les builds et les tests s'exécutent ensuite.







Des **outils automatisés** sont utilisés pour affirmer l'exactitude du nouveau code **avant son intégration**.

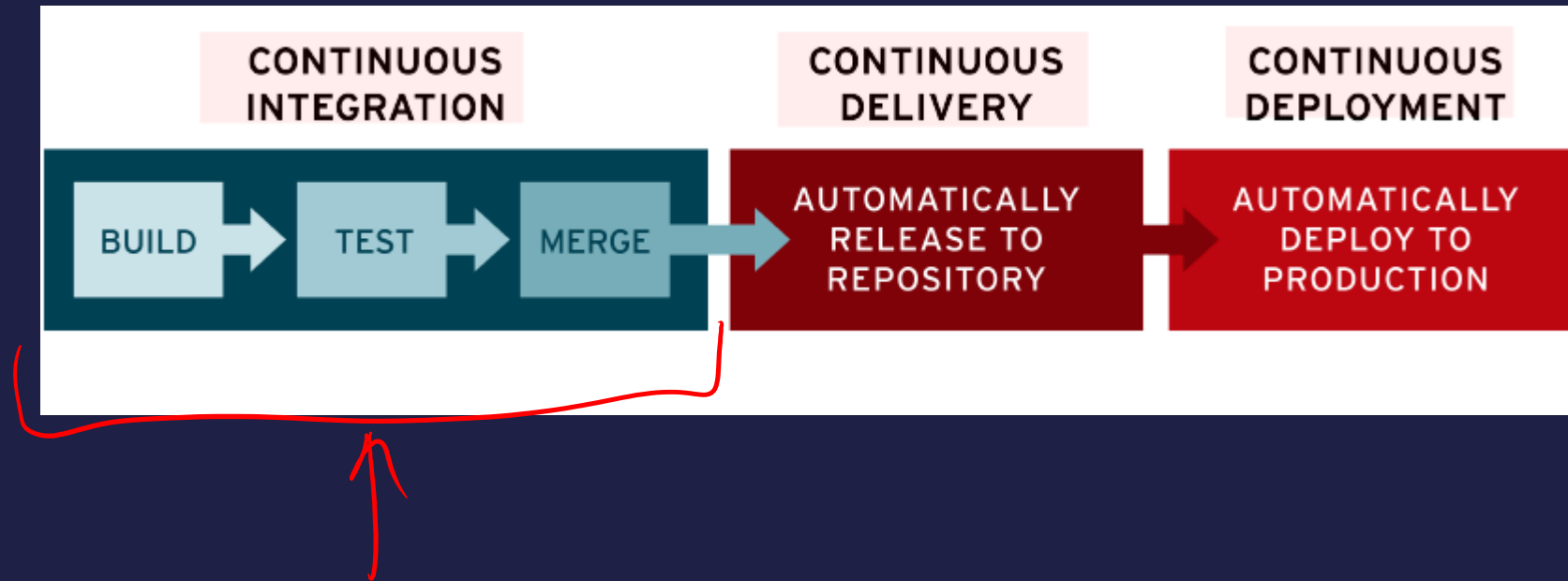
Les outils de CI se trouvent sur un autre serveur qui acte comme un agent de vérification et d'intégration.

CD

La livraison continue (en anglais : **continuous delivery**, CD) est une approche d'ingénierie logicielle dans laquelle les équipes produisent des logiciels dans des cycles courts, ce qui permet de le mettre à disposition à n'importe quel moment. Le but est de construire, tester et diffuser un logiciel plus rapidement.

Attention Continuous Delivery et Continuous Deployment n'est pas la même chose !

Continuous Deployment = Continuous Delivery + Automatic Deployment



# OUTILLAGE



JENKINS



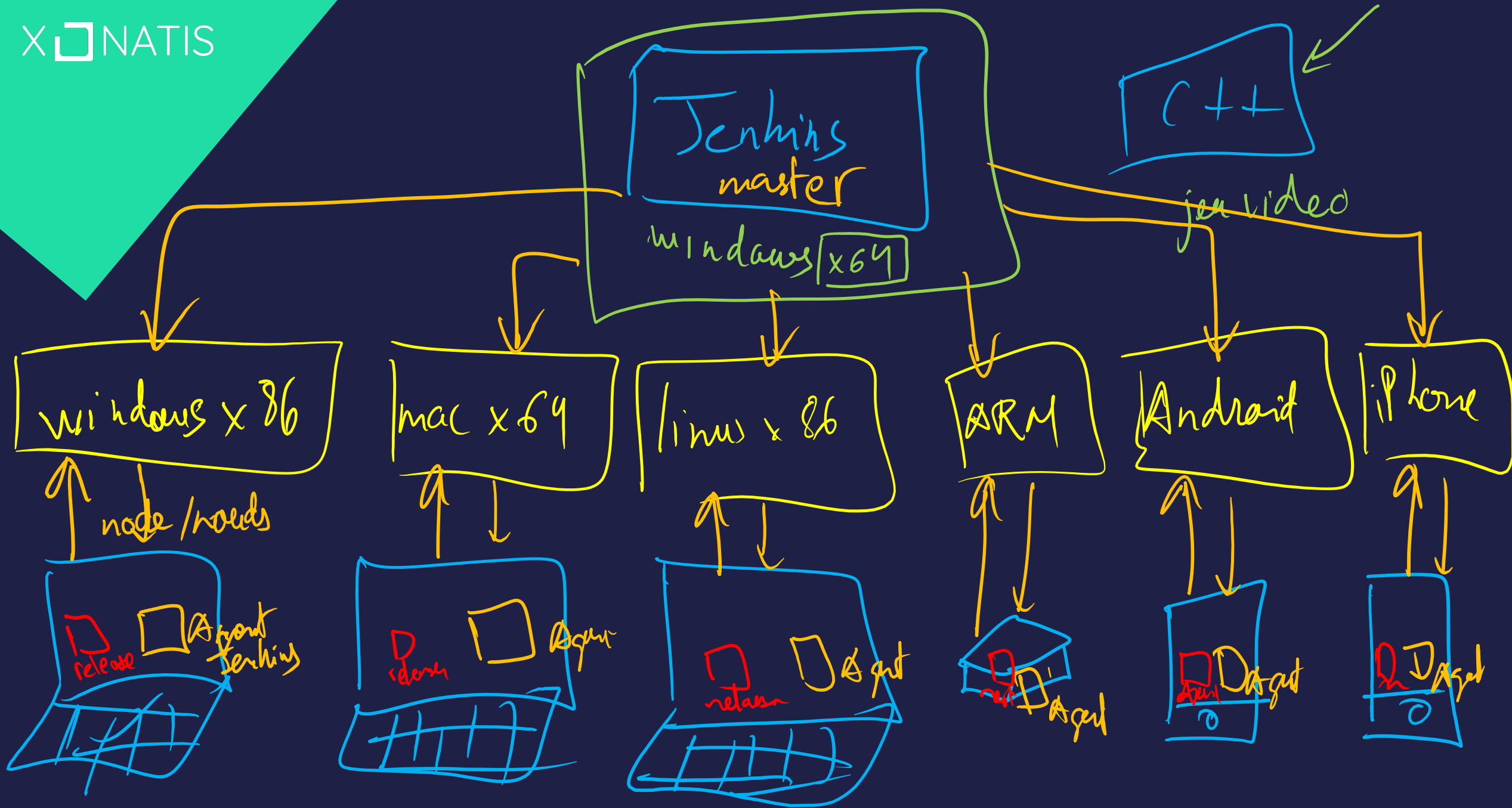
1. INSTALLATION & INTERFACE
2. BUILD JOBS
3. LES PLUGINS
4. ARCHETYPES DE PIPELINE
5. CI BEST PRACTICES


# INSTALLATION & INTERFACE

JRE 8 on 11

<https://www.jenkins.io/>

Jenkins est un **outil open source** de serveur d'automatisation. Il aide à automatiser les parties du développement logiciel liées au build, aux tests et au déploiement, et facilite l'intégration continue et la livraison continue.





 **Jenkins**


rechercher ?


Admin se déconnecter


Tableau de bord


 Nouveau Item


 Utilisateurs

 Historique des constructions

 Administrer Jenkins

 Mes vues

 Ressources Verrouillables

 Créer une Vue

File d'attente des constructions ^

File d'attente des constructions vide

État du lanceur de compilations ^

1 Au repos

2 Au repos

Ajouter une description

Tous +

S	M	Nom du projet ↓	Dernier succès	Dernier échec	Dernière durée
		Phonebook	S. O.	S. O.	ND

Icône: S M L

Légende

Atom feed pour tout

Atom feed de tous les échecs

Atom feed juste pour les dernières compilations

The screenshot shows the Jenkins dashboard with the following elements:

- Header:** Jenkins logo, search bar (rechercher), user (Admin), and logout (se déconnecter).
- Left Sidebar:**
  - Tableau de bord
  - Nouveau Item
  - Utilisateurs
  - Historique des constructions
  - Administrer Jenkins
  - Mes vues
  - Ressources Verrouillables
  - Créer une Vue
  - File d'attente des constructions
  - État du lanceur de compilations
- Main Content Area:**
  - Buttons: Tous, +
  - Table with columns: S, M, Nom du projet, Dernier succès, Dernier échec, Dernière durée.
  - Table row: Phonebook, S. O., S. O., ND.
  - Footer: Icône: S M L, Légende, Atom feed pour tout, Atom feed de tous les échecs, Atom feed juste pour les dernières compilations.

**Handwritten Annotations:**

- Folders, jobs etc...* (points to 'Nouveau Item')
- Security, Plugins* (points to 'Administrer Jenkins')
- Téléphone mobile, clés usb, imprimante* (points to 'Ressources Verrouillables')
- Projets* (points to the table header 'Nom du projet')




BUILD JOBS

Un **build job** est une terminologie utilisé dans Jenkins pour **dénommer une étape dans le pipeline** ci-dessus.

Un **pipeline** est donc une **suite de build jobs**.

Chaque build job peut être déclenché par un événement (trigger), habituellement par des webhooks.

 **Jenkins**

rechercher ?


Admin se déconnecter

Tableau de bord > Tous >

### Saisissez un nom


Nom du job

» Champ obligatoire




**Construire un projet free-style**

Ceci est la fonction principale de Jenkins qui sert à builder (construire) votre projet. Vous pouvez intégrer tous les outils de gestion de version avec tous les systèmes de build. Il est même possible d'utiliser Jenkins pour tout autre chose qu'un build logiciel.




**Construire un projet maven**

Construit un projet avec maven. Jenkins utilise directement vos fichiers POM et diminue radicalement l'effort de configuration. Cette fonctionnalité est encore en bêta mais elle est disponible afin d'obtenir vos retours.



**Pipeline**

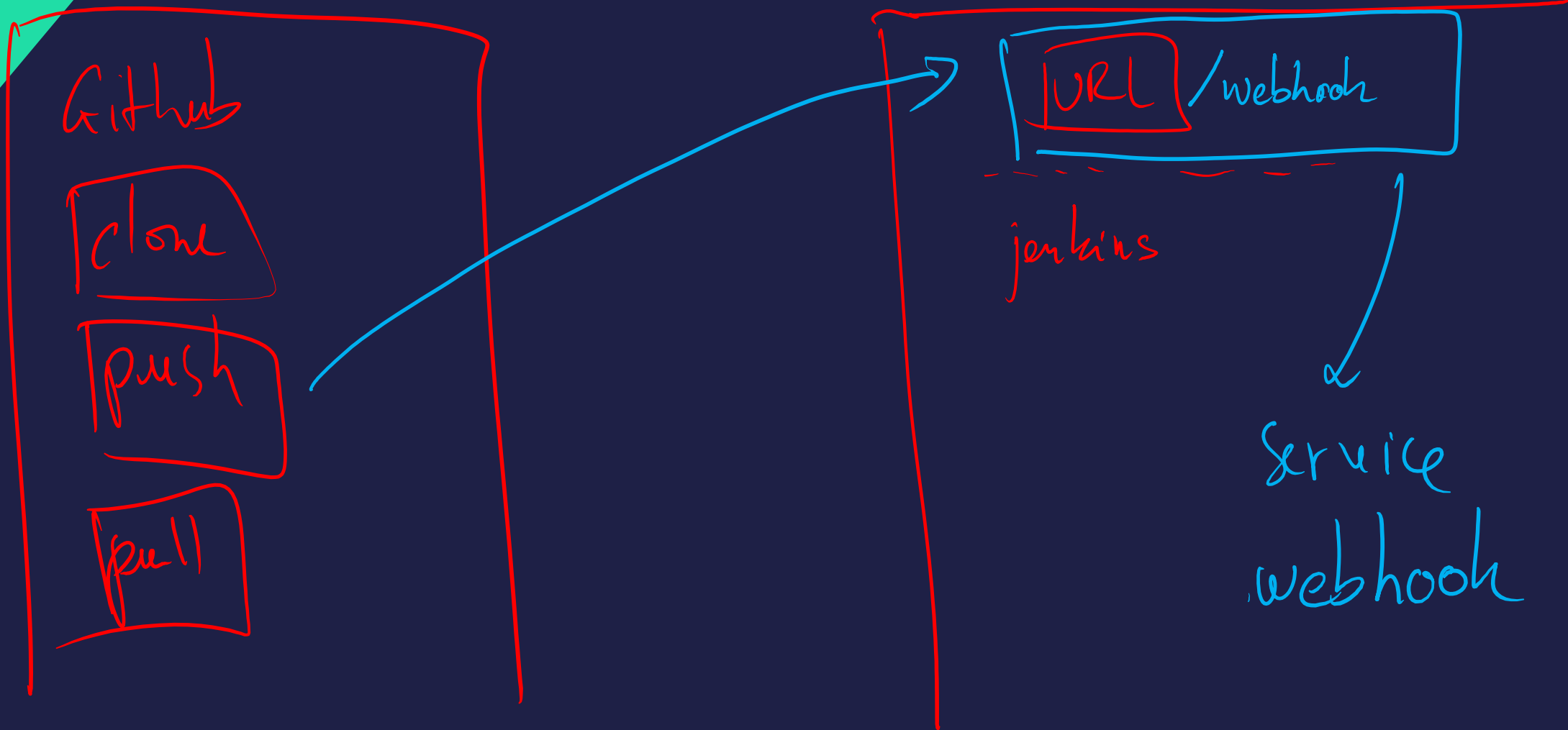
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



**Construire un projet multi-configuration**

OK apté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, etc.

## WEBHOOKS



Un artifact est le résultat produit par un job. Par exemple, pour un build job, l'artifact récupéré peut être la version packagée et distribuable de l'application.

Habituellement, l'artifact (parlé) est le résultat du processus de build.

# LES PLUGINS

Des **plugins** ont été publiés pour Jenkins qui **étendent son utilisation** et sont disponibles pour intégrer Jenkins avec la plupart des systèmes populaires.

De nombreux outils de construction sont pris en charge via leurs plugins respectifs. Les plugins peuvent également **modifier l'apparence de Jenkins** ou ajouter de **nouvelles fonctionnalités**.

Vous pouvez gérer vos plugins sur : « Administrer Jenkins » puis « Gestion des plugins »

Mises à jour

Disponibles

Installés

Avancé

Installer	Nom ↓	Version	Released	Installé
<input type="checkbox"/>	<b>Bootstrap 5 API</b> Provides Bootstrap 5 for Jenkins plugins.	5.1.3-3	1 j 23 h ago	5.1.3-2
<div><div>Credentials</div><div>Unavailable</div><div>api-plugin</div><div>This plugin allows you to store credentials in Jenkins.</div><div>This version of the plugin exists but it is not being offered for installation, so the latest bug fixes or features are not available to you. This is typically the case when plugin requirements, e.g. a recent version of Jenkins, are not satisfied. If you are using the latest version of Jenkins offered to you, newer plugin releases may not be available to your release line yet. See the <a href="#">plugin documentation</a> for information about its requirements.</div></div>				
<b>Font Awesome API</b>				

Télécharger maintenant et installer après redémarrage

Update information obtained: 13 h ago

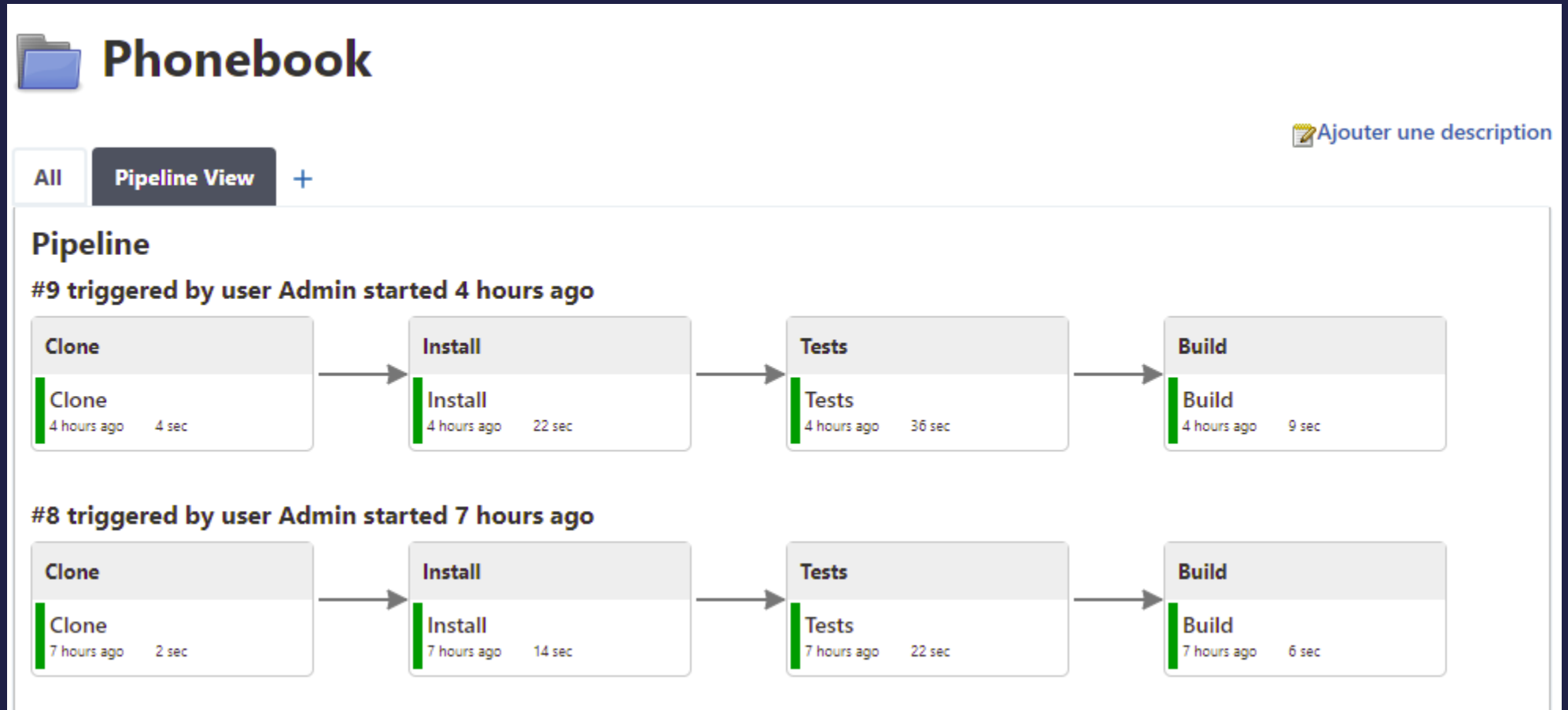
Vérifier maintenant



Delivery pipeline plugin : permet de visualiser les dépendances entre les jobs

Role-based Authorization Strategy plugin : permet de définir facilement des roles à des utilisateurs

# Delivery pipeline plugin



# Role-based Authorization Strategy plugin

## Autorisations

- ☐ Les utilisateurs connectés peuvent tout faire
- ☐ Mode legacy
- ☒ Stratégie basée sur les rôles



Permet la gestion des autorisations par une stratégie basée sur les rôles.



(from [Role-based Authorization Strategy](#))

- ☐ Stratégie d'autorisation matricielle basée sur les projets
- ☐ Sécurité basée sur une matrice
- ☐ Tout le monde a accès à toutes les fonctionnalités



# Role-based Authorization Strategy plugin

## Rôles globaux

Rôle	Global	Identifiants				Agent								Job								Histor		
	Administer	Read	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Provision	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete
 admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
 developer	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

# ARCHETYPES DE PIPELINE

Les pipelines suivent habituellement les schémas suivants :

- Fetch
- Build
- Tests
- (Build de release)
- Code quality

[illegible]

	Angular	React
Fetch	SCM	SCM
Build	ng build	react-scripts build
Tests	karma	jest
Code quality	SonarQube	SonarQube



# CI BEST PRACTICES

Dans les **guidelines de la CI/CD**, vous retrouverez les meilleures pratiques suivantes :

- Garder les jobs en **vert**
- **Versionner** votre code fréquemment
- Ne **surcharger pas vos tests** si cela vous empêche d'avoir des résultats rapides
- **Enlever les autres canaux** d'intégration et de livraison

# TEST DE CHARGE

Test d'acceptation, une technique de test effectuée pour déterminer si le système logiciel a satisfait ou non aux spécifications des exigences.

L'objectif principal de cette phase de test est d'évaluer la conformité du système avec les exigences de l'entreprise et l'état de préparation de l'application à la mise en production

Il y a 2 types de tests d'acceptation :

- Ceux menés par les utilisateurs (UAT)
- Ceux menés automatiquement pour les tests opérationnels, e.g. performance (OAT)

Nous allons voir l'automatisation de la phase OAT

JMETER



# INSTALLATION & INTERFACE



<https://jmeter.apache.org/>

# PLAN DE TEST

Un **plan de test** est un document détaillant les objectifs, les ressources et les processus d'un test spécifique pour un produit logiciel ou matériel. Le plan contient généralement une compréhension détaillée du **flux de travail** éventuel.

# THREAD GROUP

Les éléments de groupe de threads sont les points de départ de tout plan de test.

Tous les contrôleurs et sampler doivent être sous un groupe de threads.

D'autres éléments, par ex. les listeners peuvent être placés directement sous le plan de test, auquel cas ils s'appliqueront à tous les groupes de threads.

## Thread Group

Name: Thread Group

Comments:

Action to be taken after a Sampler error

☒ Continue ☐ Start Next Thread Loop ☐ Stop Thread ☐ Stop Test ☐ Stop Test Now

Thread Properties

Number of Threads (users):

1

Ramp-up period (seconds):

1

Loop Count: ☐ Infinite

1

☒ Same user on each iteration☐ Delay Thread creation until needed☐ Specify Thread lifetime

Duration (seconds):

Startup delay (seconds):

Chaque thread exécutera le plan de test dans son intégralité et de manière totalement indépendante des autres threads de test. Plusieurs threads sont utilisés pour simuler des connexions simultanées à votre application serveur.

La **période de montée en puissance** indique à JMeter combien de temps il faut pour "monter en puissance" jusqu'au nombre total de threads choisis.

Si 10 threads sont utilisés et que la période de montée en puissance est de 100 secondes, JMeter prendra 100 secondes pour que les 10 threads soient opérationnels.



Chaque thread démarrera 10 (100/10) secondes après le début du thread précédent. S'il y a 30 threads et une période de montée en puissance de 120 secondes, alors chaque thread successif sera retardé de 4 secondes

# SAMPLER

Les **sampler** indiquent à JMeter d'envoyer des requêtes à un serveur et d'attendre une réponse. Ils sont traités dans l'ordre où ils apparaissent dans l'arborescence.

Les contrôleurs peuvent être utilisés pour **modifier le nombre de répétitions** d'un sampler.

Sampler	Description
FTP Request	Emet une requête FTP pour le transfert de fichiers
HTTP Request	Emet une requête HTTP
JDBC Request	Emet une requête vers les bases de données
Java object request	Manipule une classe Java
JMS request	Emet un message suivant les spécifications des JMS de Java
JUnit Test request	Exécute des tests unitaires en Java
LDAP Request	Emet des requêtes vers des annuaires LDAP
Mail request	Emet des requêtes de manipulation d'emails
OS Process request	Exécute des commandes locales sur la machine
TCP request	Ouvre des sockets en TCP

LISTENER

Les **listeners** donnent accès aux informations que JMeter recueille sur les scénarios de test **pendant l'exécution de JMeter**.

Le listener **Graph Results** trace les temps de réponse sur un graphique. Le listener **View Results Tree** affiche les détails des demandes et des réponses de l'échantillonneur et peut afficher des représentations HTML et XML de base de la réponse.

D'autres auditeurs fournissent des informations de synthèse ou d'agrégation.