



# Module Intégration Continue

Premiers pas Jenkins

# Historique

- En 2004, apparition d'Hudson, solution opensource d'intégration continue développée par Sun Microsystems
- Suite à des problèmes d'utilisation du nom Hudson par Oracle, en novembre 2010, proposition de renommer le projet en Jenkins
- En février 2011, Oracle décide de continuer à développer Hudson. Jenkins est considéré comme un fork d'Hudson
- Aujourd'hui, Jenkins est le seul survivant !

# Qu'est ce que Jenkins ?

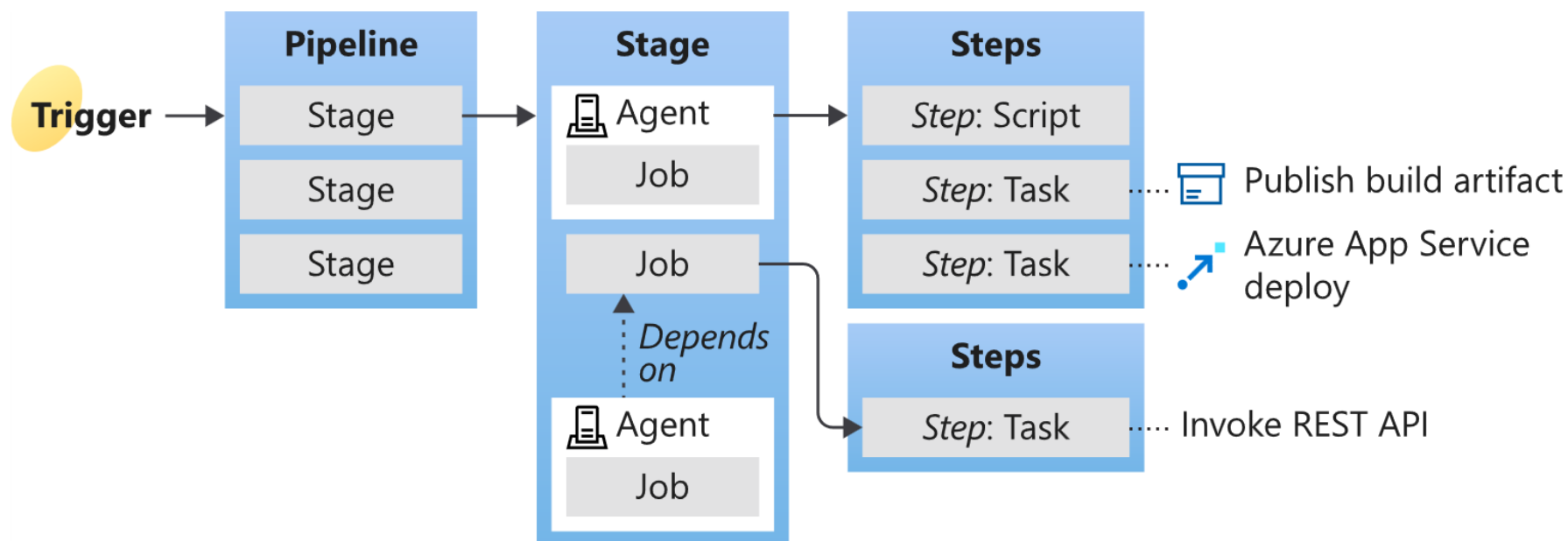
- Solution **OpenSource** CI/CD écrite en Java et permettant :
  - D'automatiser les build et les tests par configuration ou par script
  - De monitorer les phases de construction des projets
  - De construire dans des conteneurs docker ou sur des instances distantes (master / slave)
  - De pouvoir obtenir facilement les binaires des dernières versions stables
- Solution non spécifique à des projets Java. Possibilité d'utiliser Jenkins avec de nombreux langages (javascript, php, .net, c/c++, swift...)

# Qu'est ce que Jenkins ?

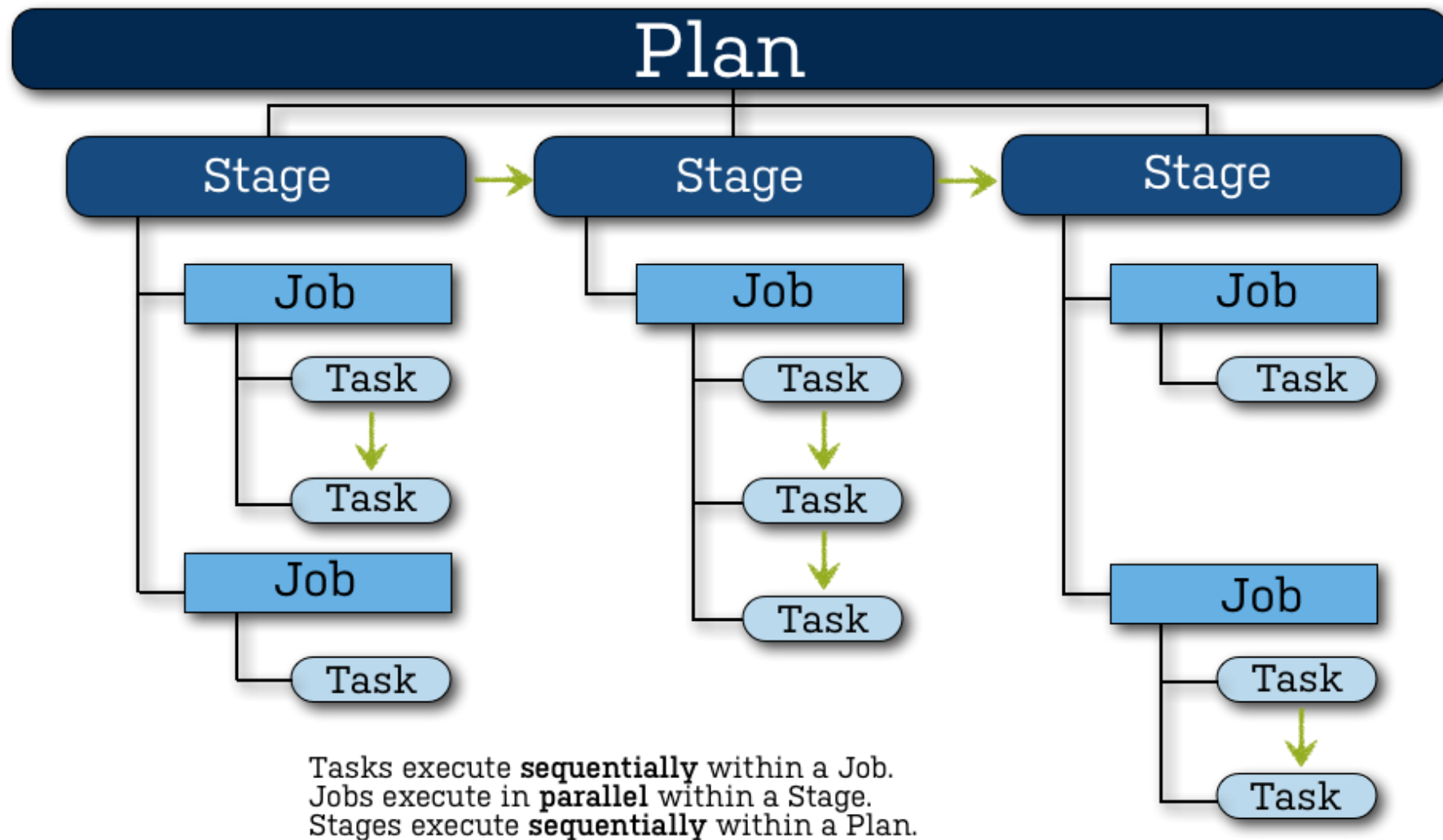
- Des centaines de plugins disponibles.
- Facilement extensible
- Basé sur RBAC. Possibilité de définir finement des droits d'accès par projet

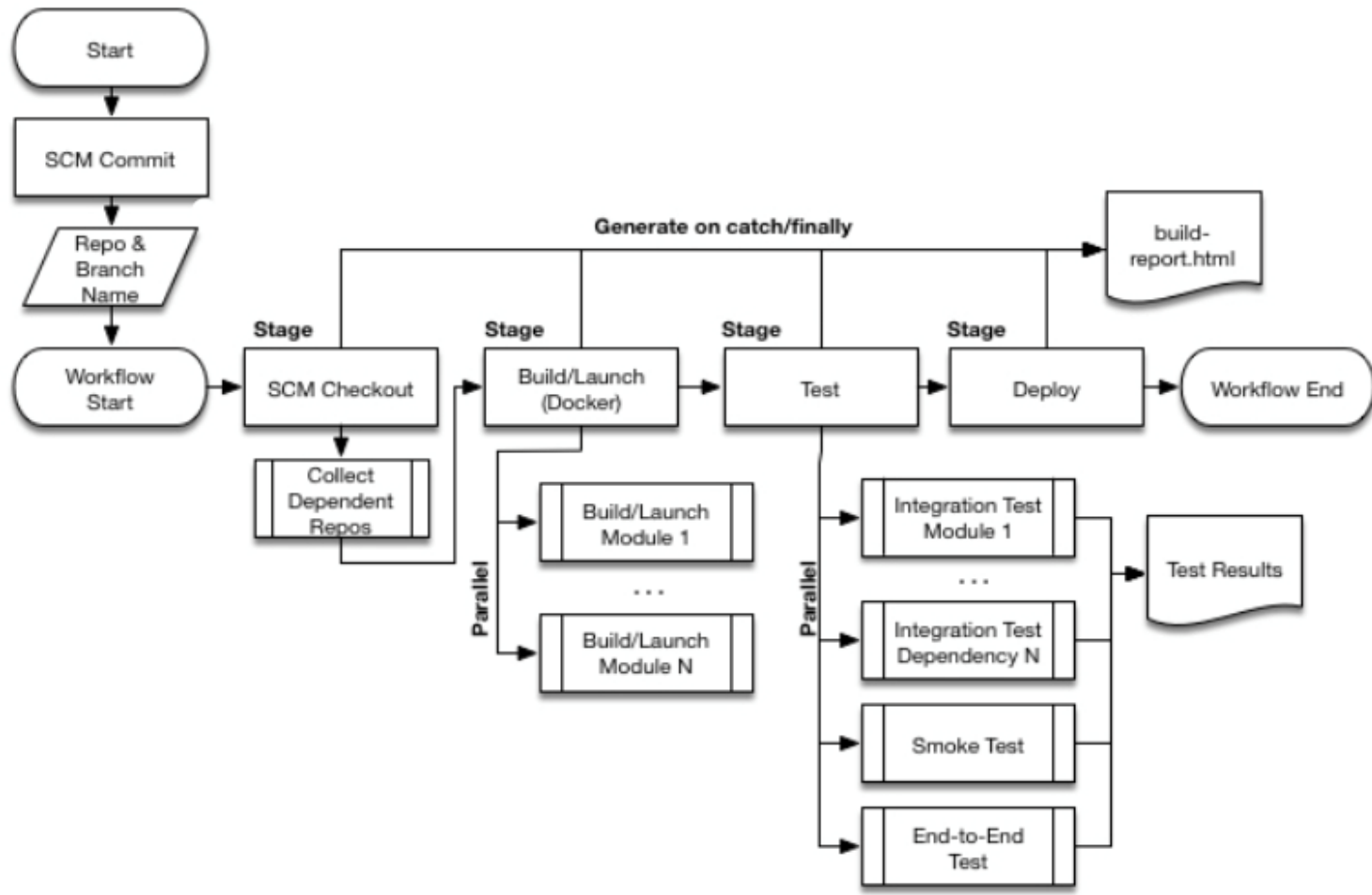
# Concepts Clé

- Un déclencheur (**Trigger**) indique à un pipeline de s'exécuter.
- Un **pipeline** est constitué d'une ou de plusieurs étapes. Un pipeline peut être déployé sur un ou plusieurs environnements.
- Une étape (**Stage**) est un moyen d'organiser les travaux dans un pipeline (ex : Build, Test, Deploy), et chaque étape peut avoir un ou plusieurs travaux.
- Chaque travail (**Job-Project**) s'exécute sur un agent. Un travail peut également s'exécuter sans agent.
- Chaque **agent** exécute un travail qui contient une ou plusieurs étapes.
- Une étape (**Step**) peut être une tâche ou un script et est le plus petit bloc de construction d'un pipeline.
- Une tâche (**Task**) est un script prédéfini qui effectue une action, telle que l'appel d'une API REST ou la publication d'un artefact de Build.
- Un **artefact** est une collection de fichiers ou de packages publiés par une exécution.

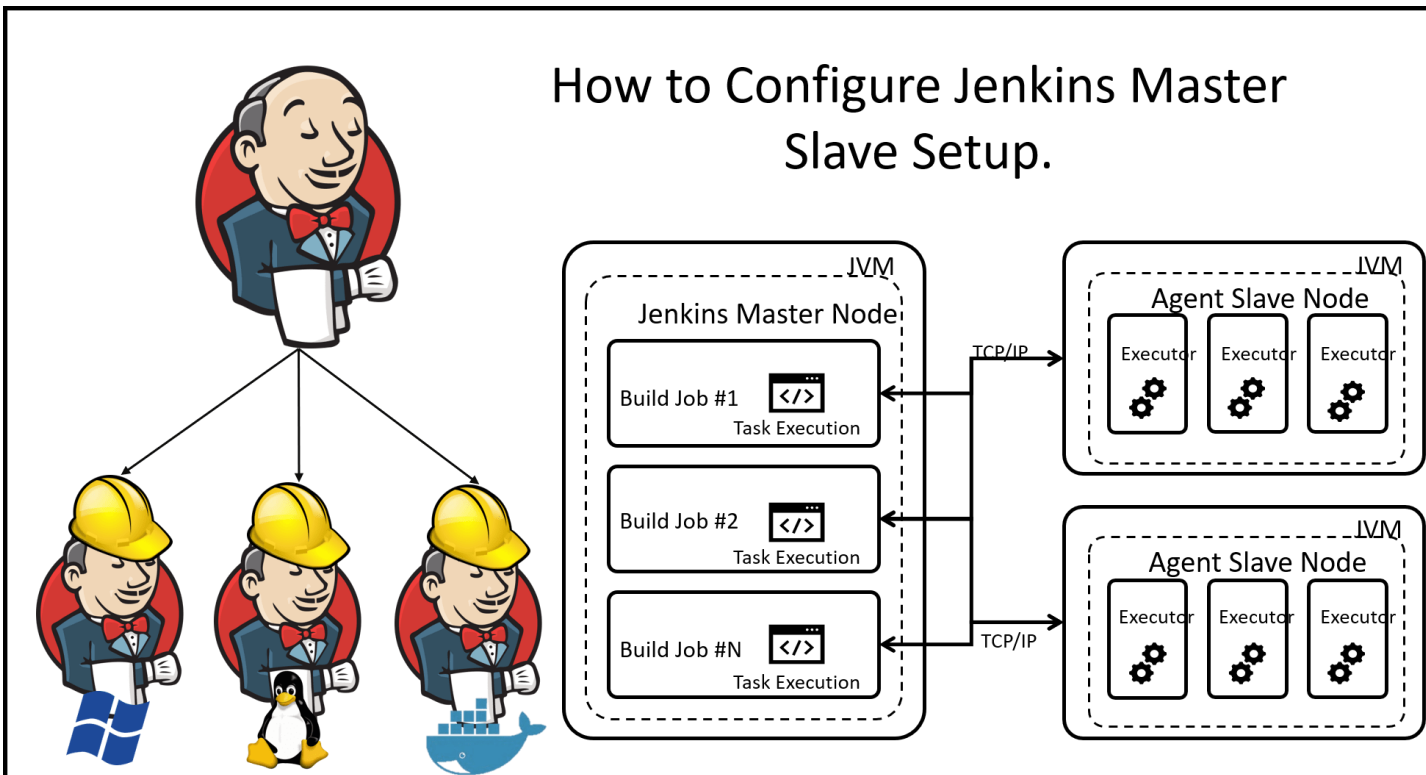


# Organisation de builds





# Jenkins Master Slave



- Master
  - Serveur principal de Jenkins coordonnant les processus tels que le stockage de la configuration, la gestion des plugins et l'affichage de l'interface Utilisateur
- Agent
  - Machine ou conteneur utilisé pour exécuter les étapes d'un « projet »
- Executor
  - Processus exécutant un projet ou un pipeline



# Configuration des diverses phases

# Installation

- Récupération du war <https://jenkins.io/download/>
- Déploiement sur un tomcat ou directement avec la commande java

```
java -jar jenkins.war
```

(PS : possibilité de spécifier le port --httpPort=8081)


# Nouveau projet


- Propose de nombreux types de projet
  - Free style
  - Construction d'un projet Maven
  - **Pipeline et multibranche pipeline (les plus utilisés pour des projets Git)**
- Possibilité de cloner un projet existant


Saisissez un nom


» Ce champ ne peut pas être vide. Veuillez saisir un nom valide et appuyer sur OK.


---

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

 **Tâche externe**  
Ce type de tâche permet de suivre l'exécution d'un process lancé en dehors de Jenkins, et ce, même sur une machine distante. Cela vous permet d'utiliser Jenkins comme tableau de bord de vos systèmes automatisés existant.

OK Folder

# Etapes de construction

## General

Gestion de code source

Ce qui déclenche le build

Environnements de Build

Build

Actions à la suite du build

Récupération des sources

Quel SCM ?

- Publication des rapports
- Envoi de notifications (Email..)
- Publication de la javadoc
- ...

Déclencheurs du build

- Déclencheur distant (hook github, gitlab)
- Suite à un autre build
- Périodiquement

Actions post-build

Environnement du build

- Variables d'environnement
- Configuration Sonar
- Règles de nettoyage du workspace

Scripts pour le build

- Scripts Maven, gradle, ant
- Scripts shell

# Gestionnaire de sources

**Gestion de code source**

☐ Aucune  
☒ Git

Repositories

Repository URL

Credentials

Branches to build

Branch Specifier (blank for 'any')

Navigateur de la base de code

Additional Behaviours

☐ Subversion

- Différents types des sources (Git, Subversion, CVS)

Possibilité d'ajouter des comportements

# Phase de construction

- Nombreux outils de build pour plusieurs langages
- Java
  - Maven, Gradle, Ant
- .Net
  - MSBuild
- iOS
- Scripts Shell

**Build**

Invoquer les cibles Maven de haut niveau

Version de Maven

Cibles Maven

Avancé...

Ajouter une étape au build ▼

# Phase Post-Build

- Nombreux mécanismes de notification

- Slack
- SMS
- Email

**Actions à la suite du build**

**Notifier par email** X ?

Destinataires

Liste des destinataires, séparés par un espace. Un email sera envoyé lors d'un échec d'un build.

☒ Envoyer un email pour chaque build instable

☐ Envoyer des emails séparés aux personnes qui ont cassé le build ?

**SMS Notification** X ?

Recipients

! You must fill recipients' numbers!

**Slack Notifications** X

# Phase Post Build

- Nombreux types de rapports publiables
  - Rapports sur l'analyse statistique du code (Checkstyle, PMD, Findbug, ...)
  - Rapport d'exécution et de couverture des tests unitaires (JUnit, Cobertura, TestNG, JaCoCo...)
  - Publication de la JavaDoc

The screenshot shows the 'Actions à la suite du build' (Post-build Actions) configuration page in Jenkins. It contains three main sections:

- Consolider les résultats des tests en aval**: Includes a checked checkbox for 'Assembler automatiquement tous les tests en aval' and an unchecked checkbox for 'Include failed builds in results'.
- Publier les Javadocs**: Includes a text input for 'Répertoire des javadocs', a description of the relative path, and an unchecked checkbox for 'Conserve les javadocs à chaque build qui passe avec succès'.
- Publish Cobertura Coverage Report**: Includes a text input for 'Cobertura xml report pattern' and detailed instructions about file name patterns and concurrent builds.

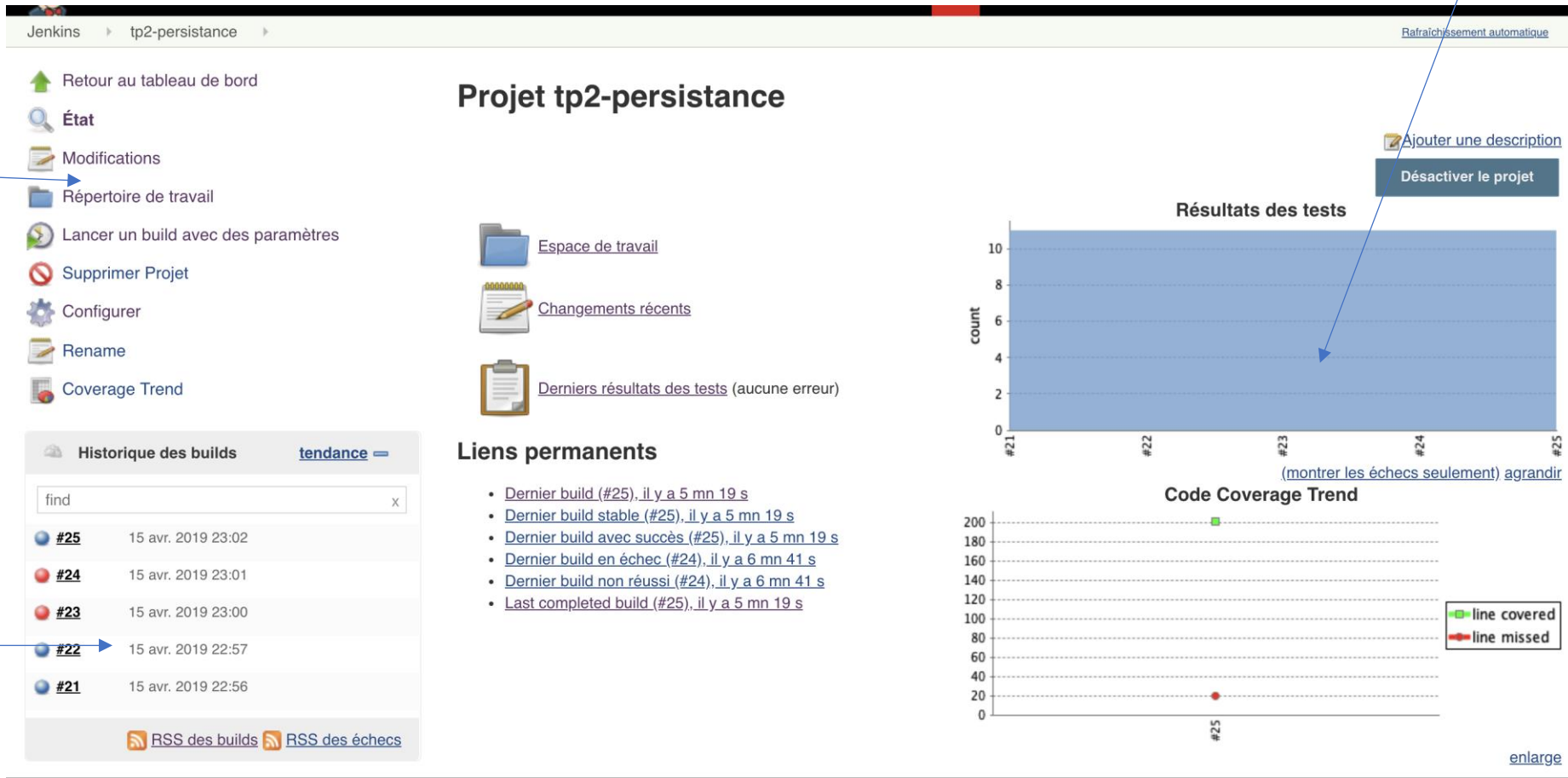


# Résultat du build

# Dashboard

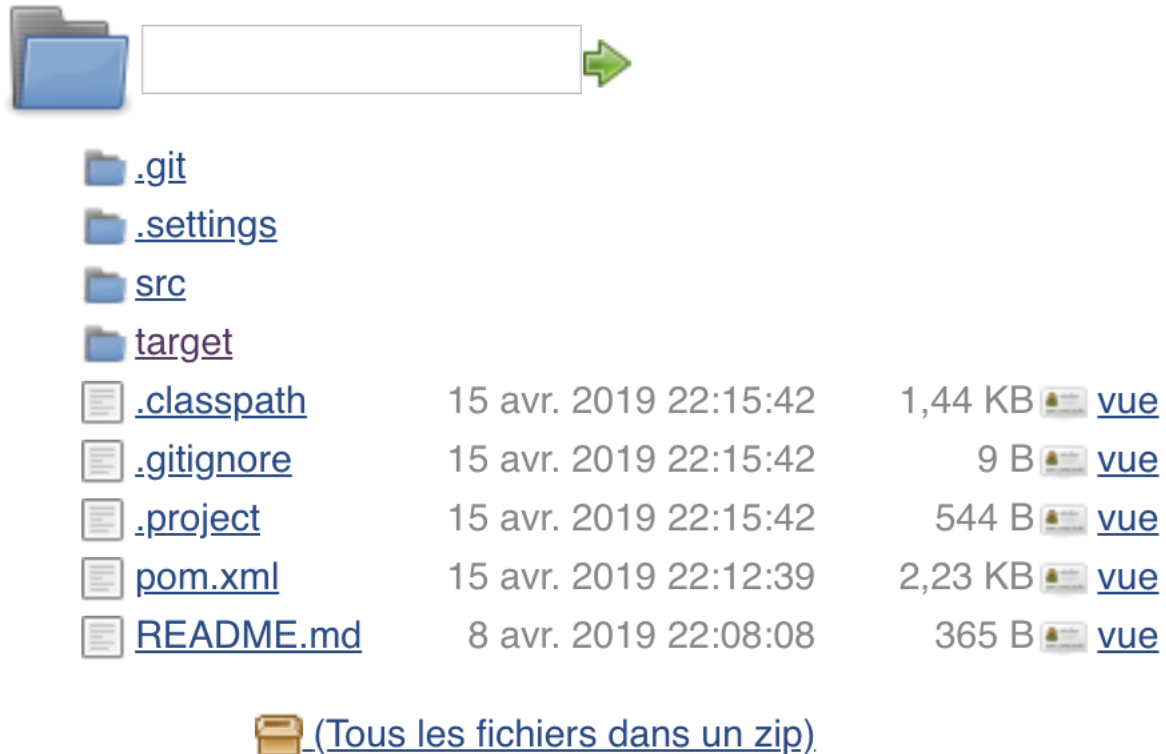
Accès  
Configuration /  
Modification

Historique  
Des builds



# Vision de l'espace de travail

## Workspace of tp2-persistence on maître



- Possibilité de voir l'espace de travail

Conseil : il est préférable de le supprimer en phase Post Build

=> Risque d'occuper beaucoup d'espace  
(Exemple : node\_modules > 400 Mo par build)

# Rapport des Tests Unitaires

## Projet tp2-persistence



[Espace de travail](#)



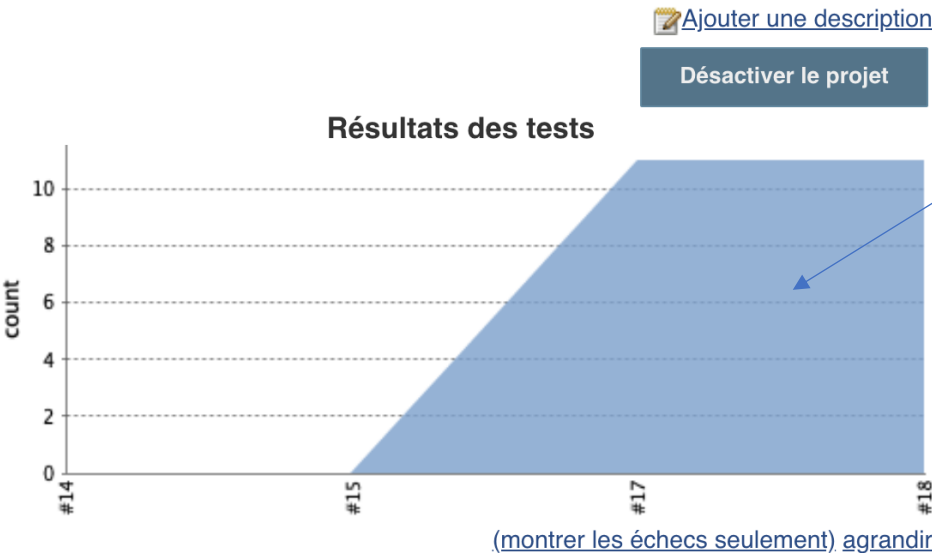
[Changements récents](#)



[Derniers résultats des tests](#) (aucune erreur)

### Liens permanents

- [Dernier build \(#18\), il y a 37 s](#)
- [Dernier build stable \(#18\), il y a 37 s](#)
- [Dernier build avec succès \(#18\), il y a 37 s](#)
- [Dernier build en échec \(#16\), il y a 12 mn](#)
- [Dernier build non réussi \(#16\), il y a 12 mn](#)
- [Last completed build \(#18\), il y a 37 s](#)



Progression du passage des TU

Etat des TU par packages / par TU

### Résultats des tests



### Tous les tests

Package	Durée	Échec (diff)	Sauté (diff)	Pass (diff)	Total (diff)
<a href="#">fr.epsi.tp.persistence.dao</a>	4 s	0	0	11 +11	11 +11

# Derniers changements

Jenkins ▶ tp2-persistence ▶

 Retour au tableau de bord

 État

 **Modifications**

 Répertoire de travail

 Lancer un build

 Supprimer Projet

 Configurer

 Rename

## Modifications

### [#15 \(15 avr. 2019 22:15:41\)](#)

1. Modification gestion exception suite chgt version java — [nicolas.rousseau1](#) / [githubweb](#)

### [#14 \(15 avr. 2019 22:12:38\)](#)

1. Modification version java — [nicolas.rousseau1](#) / [githubweb](#)



Historique des builds

[tendance](#) —

find

x



**#17**

15 avr. 2019 22:24

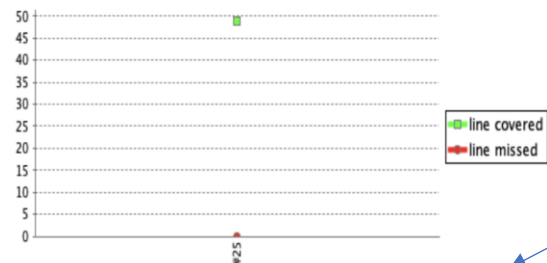
# Journal d'exécution

```
Démarré par l'utilisateur nicolas
Running as SYSTEM
Building in workspace /Users/nrousseau1/.jenkins/workspace/tp2-persistence
using credential Github
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/nicolas59/epsi-correction.git # timeout=10
Fetching upstream changes from https://github.com/nicolas59/epsi-correction.git
> git --version # timeout=10
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress https://github.com/nicolas59/epsi-correction.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/tp2-persistence^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/tp2-persistence^{commit} # timeout=10
Checking out Revision elfef17f33e11e8a5a33d6778f8ba13729e07ae2 (refs/remotes/origin/tp2-persistence)
> git config core.sparsecheckout # timeout=10
> git checkout -f elfef17f33e11e8a5a33d6778f8ba13729e07ae2
Commit message: "Modification niveau de log"
> git rev-list --no-walk 113200c34c8cc73271dfb29ea163435b926882cb # timeout=10
[tp2-persistence] $ /Users/nrousseau1/Documents/01.Logiciels/01.Dev/01-JAVA/apache-maven-3.5.4/bin/mvn clean install test
[INFO] Scanning for projects...
[INFO]
[INFO] -----< fr.epsi.tp:epsi-jee-persistence-jdbc >-----
[INFO] Building epsi-jee-persistence-jdbc 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ epsi-jee-persistence-jdbc ---
[INFO] Deleting /Users/nrousseau1/.jenkins/workspace/tp2-persistence/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ epsi-jee-persistence-jdbc ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ epsi-jee-persistence-jdbc ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 10 source files to /Users/nrousseau1/.jenkins/workspace/tp2-persistence/target/classes
[WARNING] /Users/nrousseau1/.jenkins/workspace/tp2-persistence/src/main/java/fr/epsi/tp/persistence/dao/IJdbcCrud.java: /Users/nrousseau1/.jenkins/workspace/tp2-persistence/src/main/java/fr/epsi/tp/persistence/dao/IJdbcCrud.java uses unchecked or unsafe operations.
[WARNING] /Users/nrousseau1/.jenkins/workspace/tp2-persistence/src/main/java/fr/epsi/tp/persistence/dao/IJdbcCrud.java: Recompile with -Xlint:unchecked for details.
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ epsi-jee-persistence-jdbc ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 2 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ epsi-jee-persistence-jdbc ---
```

Permet de facilement déterminer les problèmes lors de la compilation et de corriger

# Couverture de code avec JaCoCo

Package: CommandeDAO



Statistiques par classes

CommandeDAO

name	instruction	branch	complexity	lin
<a href="#">CommandeDAO()</a>	M: 0 C: 8 100% <div></div>	M: 0 C: 0 100%	M: 0 C: 1 100% <div></div>	M: 0 C: 2 100% <div></div>
<a href="#">create(Commande)</a>	M: 0 C: 91 100% <div></div>	M: 4 C: 6 60% <div></div>	M: 4 C: 2 33% <div></div>	M: 0 C: 24 100% <div></div>
<a href="#">findById(Long)</a>	M: 0 C: 88 100% <div></div>	M: 2 C: 4 67% <div></div>	M: 2 C: 2 50% <div></div>	M: 0 C: 23 100% <div></div>

Coverage

Couverture par ligne

```
15: import fr.epsi.tp.persistance.bean.CommandeLigne;
16:
17: public class CommandeDAO implements IJdbcCrud<Commande, Long> {
18:
19:     private ProduitDAO produitDAO = new ProduitDAO();
20:
21:     public Commande findById(Long identifier) throws SQLException {
22:         ResultSet rs = null;
23:         try (Connection conn = ConnectionFactory.getInstance()
24:             .getConnection();
25:             PreparedStatement ps = conn.prepareStatement("select id, date_creation, produit_
26:                 + "inner join comm_produit on comm_produit.commande_id=commande.id where id=?";
27:             ps.setLong(1, identifier);
28:             rs = ps.executeQuery();
29:             Commande commande = null;
30:             if (rs.next()) {
31:                 commande = new Commande();
32:                 commande.setIdentifier(rs.getLong(1));
33:                 Timestamp ts = rs.getTimestamp("date_creation");
34:                 commande.setDateCreation(ts.toInstant().atZone(ZoneId.of("UTC"))
35:                     .toLocalDate());
36:
37:
38:         List<CommandeLigne> comLines = new ArrayList<>();
```

# Possibilité de paramétrer le build

The screenshot shows the Jenkins web interface. At the top, there's a black header with the Jenkins logo and name. Below it, a breadcrumb trail shows 'Jenkins' > 'tp2-persistence'. On the left sidebar, there are several menu items: 'Retour au tableau de bord' (with a green arrow icon), 'État' (with a magnifying glass icon), 'Modifications' (with a notepad icon), 'Répertoire de travail' (with a folder icon), 'Lancer un build avec des paramètres' (with a play icon), 'Supprimer Projet' (with a red 'X' icon), 'Configurer' (with a gear icon), and 'Rename' (with a notepad icon). The main content area is titled 'Projet tp2-persistence'. Below the title, it says 'Ce build nécessite des paramètres :'. There is a text input field labeled 'BRANCH' containing the text 'origin/tp2-persistence'. Below this, there is a 'Build' button. To the right of the 'Build' button, there is a 'Deploiement' button with a dropdown menu showing 'OUI' (checked) and 'NON'. At the bottom, there is a 'Historique des builds' section with a search bar containing the text 'find' and a 'tendance' link.

Jenkins

Jenkins > tp2-persistence

Retour au tableau de bord

État

Modifications

Répertoire de travail

Lancer un build avec des paramètres

Supprimer Projet

Configurer

Rename

## Projet tp2-persistence

Ce build nécessite des paramètres :

BRANCH

Build

Deploiement ☒ OUI ☐ NON

Historique des builds [tendance](#)

find x

- Nombreuses extensions  
Ex : Récupération des branches / tags Git / SVN
- Définition de liste de valeur, checkbox, saisie manuelle
- Paramètres exploitable dans les scripts de build



# Pipeline

jenkinsfile

# Job Pipeline

- Pipeline est un module de plus en plus populaire permettant de configurer les tâches au travers d'un code.
- Ce dernier peut être sauvegardé avec le reste du code source de votre projet, sur votre serveur de contrôle de version.

# Script Pipeline

- Les pipelines reposent sur le langage de script Groovy.
- Les scripts Pipeline apportent les grandes notions suivantes :
  - un pipeline est l'ensemble du processus à exécuter ;
  - un nœud (node) représente un environnement pouvant exécuter un pipeline (une machine esclave) ;
  - un niveau (stage) représente un ensemble d'étapes de votre processus (par exemple, la récupération des sources, la compilation...) ;
  - une étape (step) représente ce qu'il y a à faire à un moment donné (l'action à proprement parler, comme make).

# Exemple

- Les scripts peuvent s'écrire de deux manières différentes, soit en utilisant une syntaxe de script, soit une syntaxe déclarative.
- Voici l'architecture d'un Pipeline scripté :

```
node {  
  stage('Build') {  
    //  
  }  
  stage('Test') {  
    //  
  }  
  stage('Deploy') {  
    //  
  }  
}
```

# Syntaxe déclarative

- Et la même chose, avec la syntaxe déclarative :

```
pipeline {  
  agent any  
  stages {  
    stage('Build') {  
      steps {  
        //  
      }  
    }  
    stage('Test') {  
      steps {  
        //  
      }  
    }  
    stage('Deploy') {  
      steps {  
        //  
      }  
    }  
  }  
}
```

# Example

```
node('testing') {  
    stage('Initialize') {  
        echo 'Initializing...'  
        def node = tool name: 'Node-7.4.0', type:  
            'jenkins.plugins.nodejs.tools.NodeJSInstallation'  
        env.PATH = "${node}/bin:${env.PATH}"  
    }  
  
    stage('Checkout') {  
        echo 'Getting source code...'  
        checkout scm  
    }  
}
```

```
stage('Build') {  
    echo 'Building dependencies...'  
    sh 'npm i'  
}  
  
stage('Test') {  
    echo 'Testing...'  
    sh 'npm test'  
}  
  
stage('Publish') {  
    echo 'Publishing report...'  
}  
}
```

# Workflow-Stage

- GitHub
- Unit Test Report
- Cucumber Report
- Embeddable Build Status

## Stage View

**Build History** [trend](#)

find

- #10255  
Aug 15, 2017 11:12 PM
- #10254  
Aug 15, 2017 7:52 PM
- #10253  
Aug 15, 2017 7:01 PM
- #10252  
Aug 15, 2017 6:46 PM
- #10251  
Aug 15, 2017 5:22 PM
- #10250  
Aug 15, 2017 4:27 PM
- #10249  
Aug 15, 2017 3:58 PM
- #10248  
Aug 15, 2017 3:56 PM
- #10247  
Aug 15, 2017 3:53 PM
- #10246  
Aug 15, 2017 3:51 PM
- #10245  
Aug 15, 2017 3:50 PM
- #10244  
Aug 15, 2017 3:48 PM
- #10243  
Aug 15, 2017 3:45 PM
- #10242  
Aug 15, 2017 3:34 PM

		Checkout	Build	Install Client and Server	Install wfa-ts-server	Install wfa-ts	Test Client and Server	Commit Msg Test	Lint Test	Unit Test Client	Unit Test Server	Build Server	Build Client	Publish	Deploy	SauceLabs
Average stage times:		18s	24ms	127ms	38s	1min 23s	36ms	11s	1min 40s	5min 37s	24s	13s	8min 9s	44s	13min 58s	6min 49s
#10255	Aug 16 02:12 66 commits	12s	6ms	18ms	39s	1min 34s	27ms	11s	1min 47s	6min 22s	23s	14s	8min 13s	1min 11s	10min 23s	6min 53s
#10254	Aug 15 22:52 16 commits	10s	6ms	21ms	39s	1min 13s	38ms	8s	1min 48s	6min 20s	21s	13s	7min 58s	34s	10min 24s	6min 32s
#10253	Aug 15 22:01 1 commits	10s	29ms	23ms	35s	1min 21s	42ms	8s	1min 45s	6min 17s	25s	13s	7min 58s	41s	10min 18s	6min 27s
#10252	Aug 15 21:46 6 commits	1min 23s	9ms	20ms	41s	1min 46s	28ms	14s	1min 51s	6min 13s	25s	13s	7min 58s	39s	9min 31s	7min 21s failed
#10251	Aug 15 20:22 No Changes	11s	8ms	19ms	37s	1min 21s	33ms	13s	1min 50s	6min 12s	19s	13s	8min 12s	46s	6min 27s	6min 54s
#10250	Aug 15 19:27 No Changes	10s	28ms	55ms	33s	1min 20s	35ms	16s	1min 51s	6min 6s	30s	13s	8min 13s	18s	21min 29s failed	
#10249	Aug 15 18:58 2 commits	9s	7ms	42ms	35s	1min 16s	29ms	9s	1min 47s	6min 30s	25s	13s	8min 13s	54s	21min 34s failed	
#10248	Aug 15 18:55 2 commits	7s	7ms	19ms	39s	2min 3s	38ms	10s	32s failed	32s failed	25s					

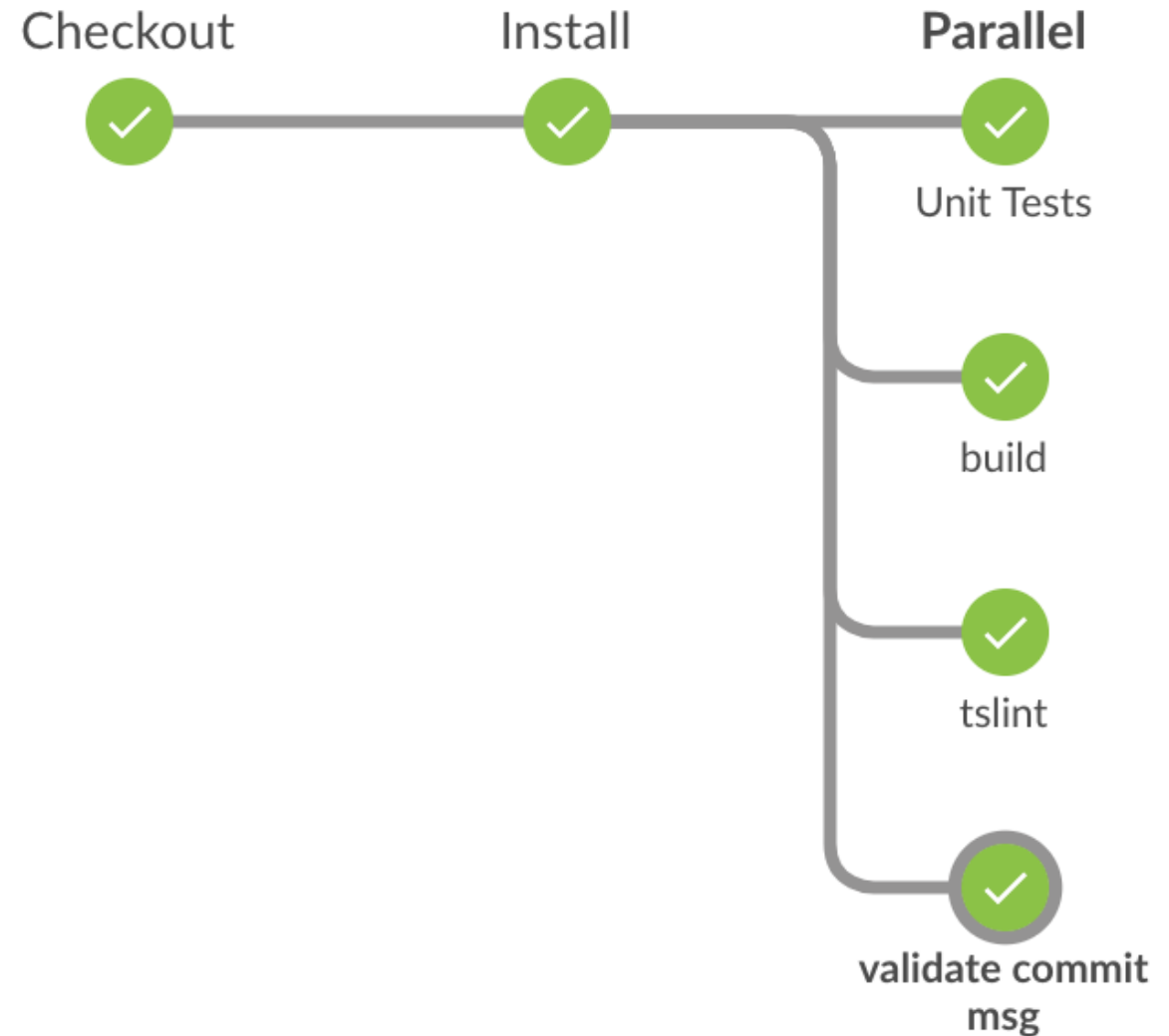
# Steps en parallèle

```
parallel(  
  //Deux deployments exécutés en parallèle  
  deployToEnv1: {  
    node('environment-1') {  
      deploy(...)  
    }  
  },  
  deployToEnv2: {  
    node('environment-2') {  
      deploy(...)  
    }  
  }  
)  
//Execution continue à la fin du step finishes
```



# Blue Ocean Plugin

## New UI –Blue Ocean



A vous de jouer !