

Industrialisation et Continuous Delivery

GÉRER SES SOURCES ET LES LIER À SON INTÉGRATION CONTINUE
AVEC L'UN DES OUTILS LES PLUS POPULAIRES



The world is how we shape it*

sopra  steria

* Le monde est tel que nous le façonnons.

Vos intervenants



Benoit POIRIER

Architecte Solution

Sopra Steria

benoit.poirier@soprasteria.com



Fabrice ROULAND

Expert Technique

Sopra Steria

fabrice.rouland@soprasteria.com

Ce que nous allons aborder

Les bases de Jenkins

L'administration

En mode manuel

Le pipeline

BlueOcean

01

Les bases de Jenkins

C'est quoi Jenkins

L'outil phare de la chaîne CI/CD

- Un serveur d'automatisation open-source
- Extensible grâce à des centaines de plugins
- Distribué : Il peut répartir les tâches sur plusieurs esclaves
- Il peut prendre en charge de nombreux types de tâches : build, test, packaging, déploiement
- Il est principalement utilisé pour l'automatisation de chaînes d'intégration continue, livraison continue, mais pas seulement



Jenkins

Utilisé par les plus grands

NETFLIX

ebay

GitHub



NOKIA

LinkedIn



Jenkins dans le Cloud

CloudBees, la version destinée aux entreprises

Une version destinée aux entreprises.

De nombreux plugins et connecteurs aux principales plateformes Cloud permettent d'élargir les possibilités de Jenkins.

D'ici 2020, CloudBees prévoit de fournir en mode SaaS sa plateforme de gestion de la distribution des logiciels, de même que Jenkins X, solution CI/CD open source pour les applications cloud sur Kubernetes.



A quoi ça sert

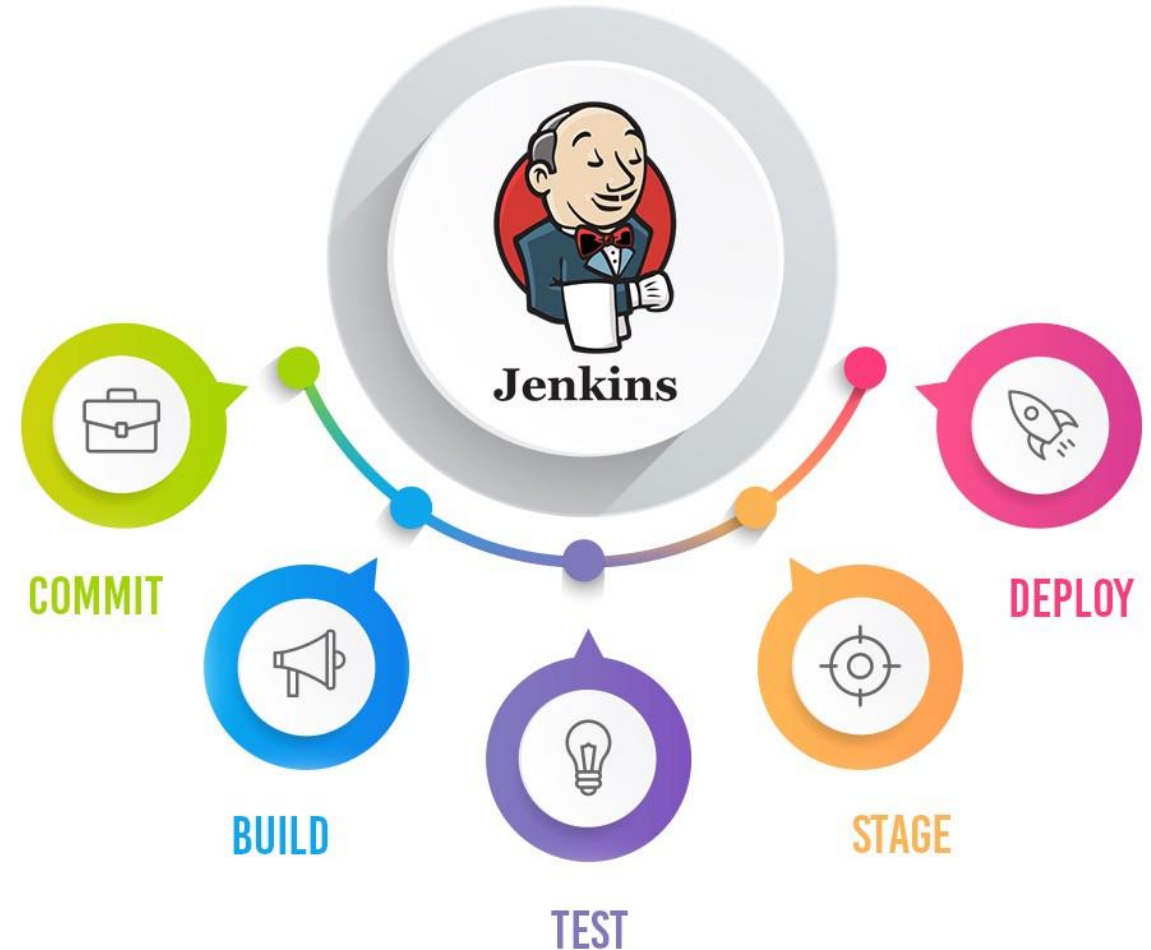
Jenkins au cœur de l'intégration continue

Automatisation des tâches de développement

- Lancement des tests
- Construction des releases
- Publication des tags, de la documentation, ...
- Déploiement de l'environnement de tests
- Déploiement en production
- Notification
- Reporting

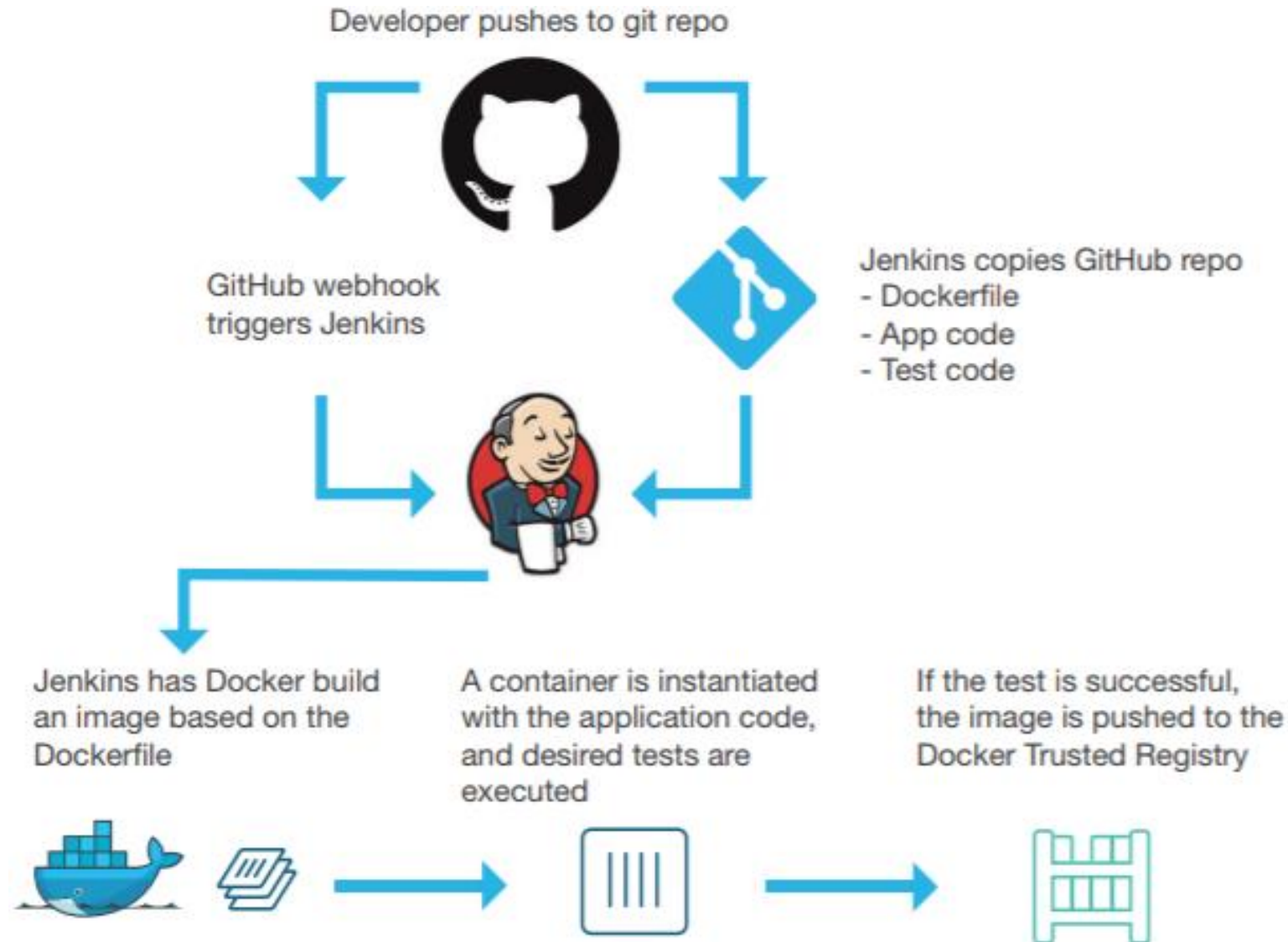
Automatisation des tâches de maintenance

- Back up
- Mise à jour
- Log analysis




A quoi ça sert

Jenkins au cœur de l'intégration continue et du déploiement via Docker




A quoi ça ressemble

Jenkins au cœur de l'intégration continue

 Job Config History

 Open Blue Ocean

 Job Import Plugin

 Identifiants

 New View

File d'attente des constructions

File d'attente des constructions vide

État du lanceur de compilations

 maître

1 Au repos

 Slave-1

1 Au repos












 Slave-10

1 Au repos

 Slave-2





























1 Au repos

 Ajouter une descripti

DEV		Docker		Tous	Experiments		
S	M	Nom du projet ↓	Dernier succès	Dernier échec	Dernière durée	Fav	Robot Results
		CI-status-monitor	5 h 53 mn - #858	5 j 18 h - #834	4.5 s	 	
		Experiments	s. o.	s. o.	ND		/ passed 
			1 mn 13 s - log	s. o.	4.9 s	 	/ passed 
		_cppcheck_results	5 mo. 19 j - #19	2 mo. 11 j - #21	2.4 s	 	
		Legacy Jobs	s. o.	s. o.	ND		/ passed 
			1 mn 13 s - log	s. o.	4.9 s	 	/ passed 
		_Docker-images-compilation-build	7 mo. 15 j - #10	7 mo. 15 j - #9	33 mn	 	
		_Docker-images-reference-build	1 mo. 12 j - #156	1 mo. 13 j - #154	1 mn 53 s	 	
		-INT	13 s - log	s. o.	0.91 s	 	/ passed 
		-INT-BasicTest	10 mo. - #6	11 mo. - #4	37 mn		17 / 17 passed 
			s. o.	s. o.	ND	 	
		-INT-launchCbamVnf-IPstatic	16 j - log	s. o.	2.6 s	 	/ passed 






Une interface de pilotage






Jenkins au cœur de l'intégration continue

 #20	19 sept. 2018 09:44	  
 #19	14 sept. 2018 18:04	  
 #18	13 sept. 2018 12:22	  
 #17	13 sept. 2018 10:25	  
 #16	12 sept. 2018 10:39	  
 #15	7 sept. 2018 12:35	  
 #14	7 sept. 2018 10:05	  

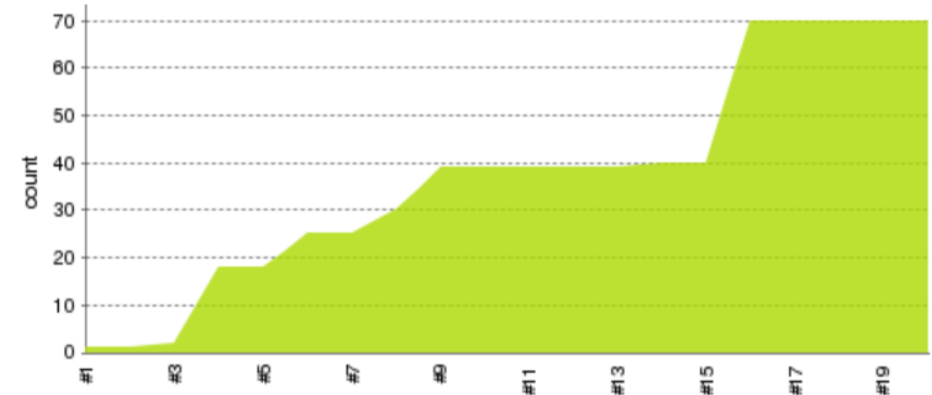


[Last Successful Artifacts](#)

-  [ves-agent](#)
-  [ves-agent-master-20.x86_64.rpm](#)
-  [ves-agent.exe](#)
-  [ves-simu](#)
-  [ves-simu.exe](#)

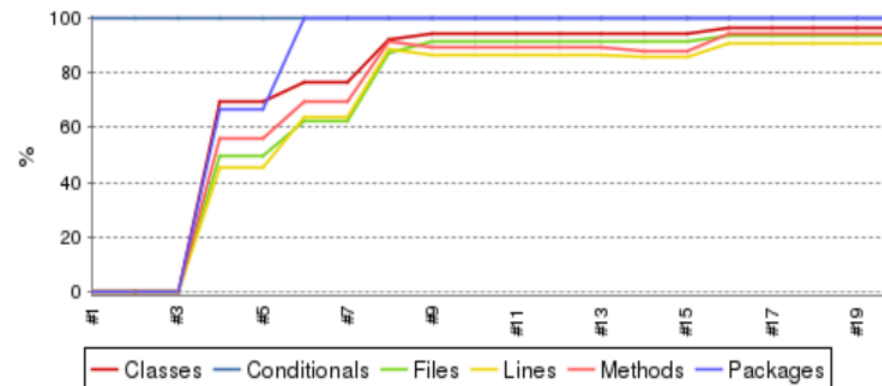
12.48  MB [view](#)
2.52  MB [view](#)
12.38  MB [view](#)
10.11  MB [view](#)
10.05  MB [view](#)

Résultats des tests



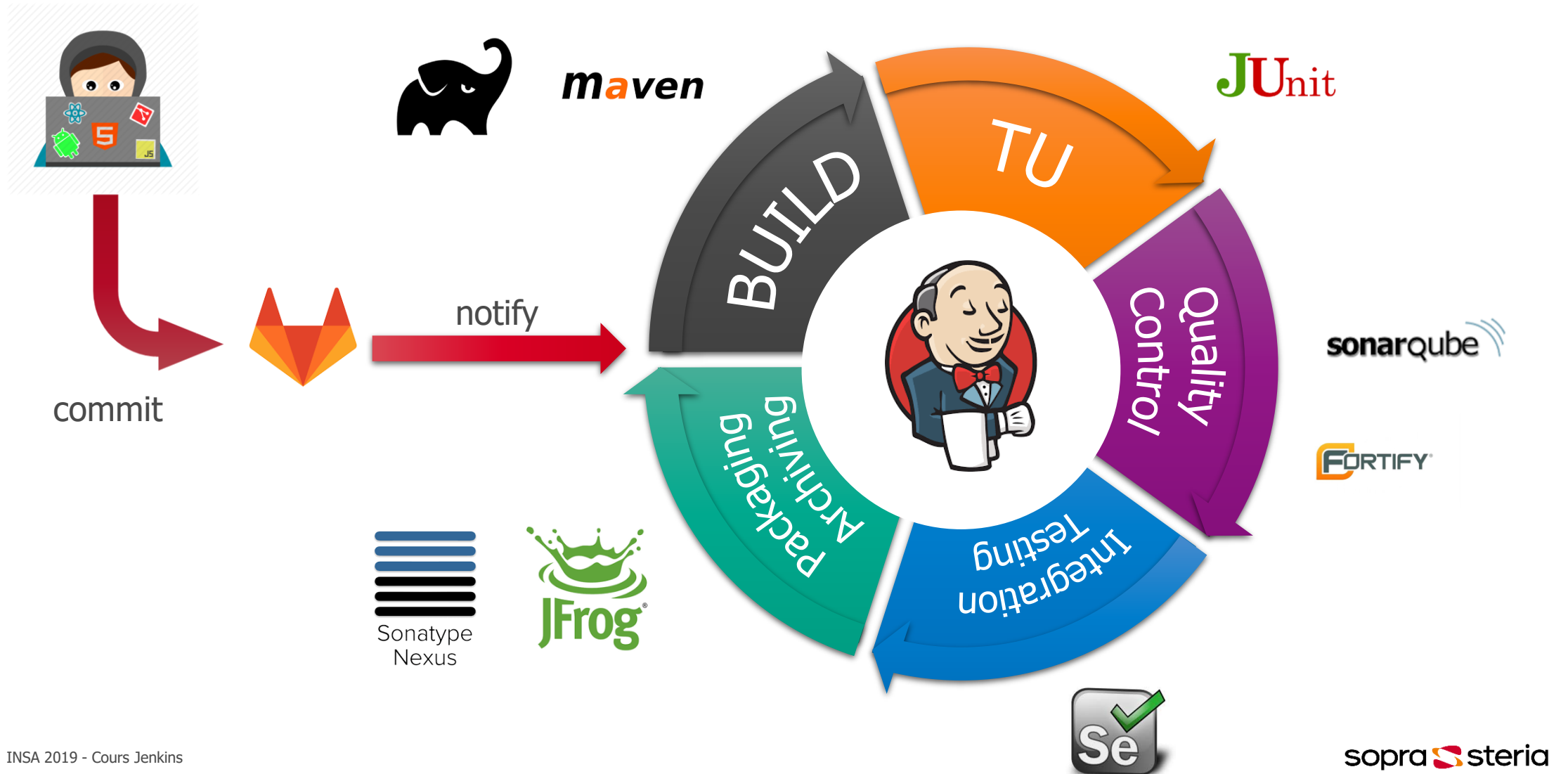
Code Coverage

Packages 100% Files 94% Classes 96% Methods 95% Lines 91%
Conditionals 100%



Les bases de Jenkins

Jenkins au cœur de l'intégration continue



02

L'administration

L'administration de Jenkins

Accueil

Avant de commencer l'utilisation de Jenkins, quelques tâches d'administration sont nécessaires :

- Configuration des paramètres système (emplacement du JDK, ...)
- Configuration de la sécurité (accès aux fonctionnalités)
- Configuration des identités (accès à des repository externe)
- Configuration des outils (en général des plugins)
- Gestion des plugins (ajout, suppression des plugins parmi un catalogue très riche)
- Information sur le système
- Affichage des logs
- Affichage de statistiques

Administrer Jenkins



Configurer le système

Configurer les paramètres généraux et les chemins de fichiers.



Configurer la sécurité globale

Sécuriser Jenkins; définir qui est autorisé à accéder au système.



Configure Credentials

Configure the credential providers and types



Configuration globale des outils

Configurer les outils, leur localisation et les installeurs automatiques.



Recharger la configuration à partir du disque

Supprimer toutes les données en mémoire et recharger tout à partir du système de fichiers. Utile quand vous mo



Gestion des plugins

Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins.

▲ mises à jour disponibles



Informations sur le système

Affiche diverses informations relatives au système pour aider à la résolution de problèmes.



Logs systèmes

Le log système capture la sortie `java.util.logging` relative à Jenkins.



Statistiques d'utilisation

Vérifiez l'utilisation des ressources et décidez si vous avez besoin d'ordinateurs supplémentaires pour vos builds

L'administration de Jenkins

La configuration système

Maven

Maven installations

name

☒ Install automatically

Install from Apache

Version

List of Maven installations on this system

JDK

JDK installations

name

☒ Install automatically

Install from java.sun.com

Version

☒ I agree to the Java SE Development Kit License Agreement

Home directory

System Message

of executors

Quiet period

SCM checkout retry count

☐ Enable security

☐ Prevent Cross Site Request Forgery exploits

☒ Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project.

Global properties

☐ Environment variables

JDK

JDK installations

List of JDK installations on this system

Ant

Ant installations

List of Ant installations on this system

Maven

Maven installations

List of Maven installations on this system

Maven Project Configuration

Global MAVEN_OPTS

L'administration de Jenkins

Plus de 1500 plugins

Ansible Installs: 16839 Jenkins 1.596.1++ Build tools, Deployment plugins, Invoke Ansible Ad-Hoc commands and playbooks. Jean-Christophe Sirot An	CVS Installs: 47624 Jenkins 1.625.3++ SCM connections, Integrates Jenkins with CVS version control system using a modified version of the Netbeans cvsclient. Kohsuke Kawaguchi Michael Clarke CV	Shared Workspace Installs: 1819 Jenkins 1.576++ SCM connections, This plugin shares workspaces between projects. Sergey Mylnikov Sergey Vidyuk	Template Project Installs: 2892 Jenkins 1.580.1++ Build notifiers, Build tools, SCM connections, This plugin lets you use builders, publishers and SCM settings from another project.	REPO Installs: 3420 Jenkins 2.60.3++ SCM connections, This plugin allows use of repo as an SCM tool. A repo binary is required.	Mercurial Installs: 77273 Jenkins 2.138.4++ SCM connections, This plugin integrates Mercurial SCM with Jenkins. It includes repository browsing support for hg serve/hgweb. Jesse Glick Mirko Friedenhagen (3 other contributors) Me	Google Deployment Manager Installs: 253 Jenkins 1.596.1++ Deployment plugins, This plugin provides a create and delete Google Cloud Platform resources using Google Cloud Deployment Manager. Mohsen Vakilian Matt Moore (1 other contributors) GD	SourceGear Vault Installs: 63 Jenkins 1.480++ SCM connections, This plugin integrates SourceGear Vault/ Fortress SCM to Jenkins. (Only limited functionality is implemented) Stuart Whelan SV	DeployHub Installs: 87 Jenkins 1.625.3++ Clean-up actions, Deployment plugins, Allows DeployHub to track builds against Components and to optionally deploy an Application Phil Gibbs Steve Taylor De	Dimensions Installs: 173 Jenkins 1.609.3++ Artifact uploaders, Build notifiers, Clean-up actions, SCM connections, This plugin integrates the Dimensions CM SCM with Jenkins. David Conneely Di
URL SCM Installs: 731 Jenkins 1.392++ SCM connections, A URL SCM plugin for Hudson. This plugin supports polling and checkout. Polling is done by checking the last-modified date of the file. Michael Donohue Jesse Farinacci US	Config Rotator Installs: 82 Jenkins 2.154++ SCM connections, Clean-up actions, Automatic try-out of possible configurations. Monitors the SCM for newer versions of components and tests if they are compatible. Praqma Jorja CR	CMVC Installs: 15 Jenkins 1.398++ SCM connections, This plugin provides integration to Hg. Fábio Franco Ueda	Parasoft Environment Manager Installs: 46 Jenkins 1.625.3++ Deployment plugins, Parasoft environments with Parasoft Continuous Testing Platform Jesse Glick	File System SCM Installs: 2009 Jenkins 1.642.3++ SCM connections, File System SCM Jesse Glick	Seapine Surround SCM Installs: 121 Jenkins 1.580.3++ SCM connections, This plugin integrates the source control tool Surround SCM to Jenkins. Jesse Glick	Team Concert Installs: 2405 Jenkins 1.625.1++ SCM connections, Integrates Rational Team Concert Sridevi Sangalah Raghu NS (1 other contributors) TC	XebiaLabs XL Deploy Installs: 557 Jenkins 1.642.3++ Artifact uploaders, Clean-up actions, Deployment plugins, Package and deploy your applications from Jenkins with XebiaLabs XL Deploy. XebiaLabs XX	Git Installs: 248924 Jenkins 2.138.4++ SCM connections, This plugin integrates Git with Jenkins. Mark Waite Robert Sandell (1 other contributors) Gi	Subversion Installs: 220679 Jenkins 2.107.3++ SCM connections, This plugin integrates Subversion with Jenkins. Kohsuke Kawaguchi Stephen Connolly (9 other contributors) Su
deployment-notification Installs: 787 Jenkins 1.580.1++ Deployment plugins, This plugin provides abstract common parts for integration with configuration management tools like Chef Puppet Ansible etc. Felix Belzunce Arcos De	StarTeam Installs: 148 Jenkins 1.345++ SCM connections, This plugin is a SCM interface to StarTeam. Ilkka Laukkanen John McNair (5 other contributors) St	Harvest SCM Installs: 66 Jenkins 1.609.2++ SCM connections, This plugin makes it possible to retrieve files from a Harvest SCM. Gábor Lipták HS	jenkins-testswarm-plugin Installs: 50 Jenkins 1.424++ SCM connections, Supported browsersAll available browsersCurrent release of all major browsersBrowsers currently supported Kevin Nilson Je	Synergy Installs: 116 Jenkins 1.424++ SCM connections, This plugin integrates Telelogic Synergy SCM to Hudson. Jean-Noël RIBETTE Keith Mendoza Sy	Docker Pipeline Installs: 222174 Jenkins 2.60.3++ Deployment plugins, Build and use Docker containers from pipelines. Oleg DP	Blame Subversion Installs: 942 Jenkins 1.355++ SCM connections, This plug-in provides utilities for getting svn info and build number synchronize from upstream job to downstream Developer Guy BS	IFTTT Build Notifier Installs: 80 Jenkins 1.642.3++ SCM connections, Plugin for notifying build results via IFTTT Maker Channel Trigger Unmesh Gundecha IB	Parameter Pool Installs: 323 Jenkins 1.580.1++ SCM connections, Allows user to define a range of parameters, an unused value will be picked per build Damien Biggs PP	AWS CodePipeline Installs: 1658 Jenkins 1.479++ SCM connections, Clean-up actions, AWS CodePipeline Integration Felipe Almeida AC

Categories

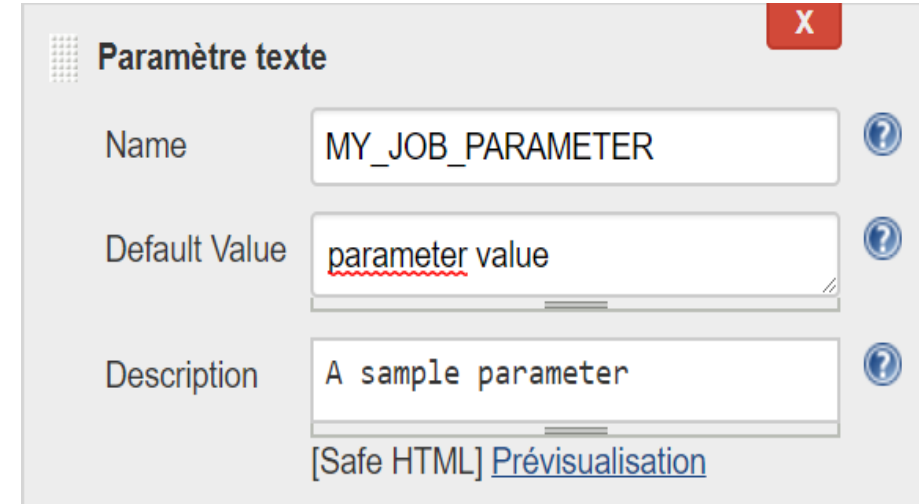
- ☐ Platforms
 - ☐ iOS development
 - ☐ .NET
 - ☐ Android development
 - ☐ Ruby development
- ☐ User interface
 - ☐ User Interface
 - ☐ List view column plugins
- ☐ Administration
 - ☐ Agent controllers
 - ☐ Page decorators
 - ☐ Users and security
 - ☐ Cluster management
 - ☐ CLI extensions
- ☐ Source code management
 - ☐ SCM connections
 - ☐ SCM related
- ☐ Build management
 - ☐ Build triggers
 - ☐ Build wrappers
 - ☐ Build notifiers
 - ☐ Deployment plugins
 - ☐ Build parameters
 - ☐ Clean-up actions
 - ☐ Build tools
 - ☐ Build reports
 - ☐ Artifact uploaders

03

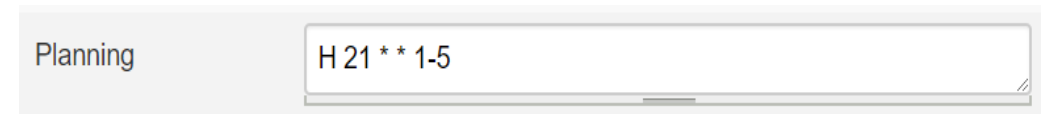
Création de job en mode manuel

Définition des tâches

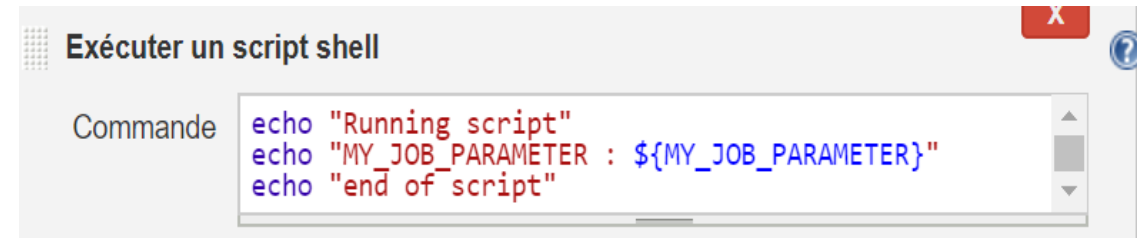
- On peut définir chaque tâche via l'UI de Jenkins
 - └ Les paramètres d'entrée
 - └ Les déclencheurs / La planification
 - └ Les scripts
 - └ Les actions à posteriori (archivage, publication, résultats / graphiques, mailing, ...)



The screenshot shows the 'Paramètre texte' (Text Parameter) configuration dialog in Jenkins. It has a title bar with a grid icon, the title 'Paramètre texte', and a red close button with an 'X'. The dialog contains three input fields: 'Name' with the value 'MY_JOB_PARAMETER', 'Default Value' with the value 'parameter value' (underlined with a red squiggly line), and 'Description' with the value 'A sample parameter'. Each field has a blue help icon to its right. At the bottom, there is a label '[Safe HTML]' and a blue link 'Prévisualisation'.



The screenshot shows a configuration field for 'Planning' in Jenkins. The field is a text input box containing the value 'H 21 ** 1-5'.



The screenshot shows the 'Exécuter un script shell' (Run Shell Script) configuration dialog in Jenkins. It has a title bar with a grid icon, the title 'Exécuter un script shell', and a red close button with an 'X'. The dialog contains a 'Commande' (Command) field with a text area containing the following shell script:

```
echo "Running script"
echo "MY_JOB_PARAMETER : ${MY_JOB_PARAMETER}"
echo "end of script"
```


Les problèmes

- **Plus le projet grossit, plus il devient :**
 - └ Difficile à maintenir
 - └ Difficile à comprendre
 - └ Difficile à débbuguer
- **Il devient difficile de suivre les changements**
- **On ne peut pas vérifier les changements d'une taches sans la valider**
- **Que se passe t'il si plusieurs personnes procèdent à des modifications simultanément ?**
- **On ne peut pas rejouer une ancienne tache qui a été modifiée**

04

Le pipeline

VOUS AVEZ DIT DES BRANCHES ?

C'est quoi un pipeline

Les Pipelines Jenkins ont été introduits dans la version 2 de Jenkins.

Ces Pipelines permettent de décrire au sein d'un fichier Groovy l'ensemble des étapes qui seront utiles à la fabrication de votre logiciel.

L'avantage avec les Pipelines déployés via un Jenkinsfile est de pouvoir :

- **configurer des jobs complexes facilement sans passer par les formulaires Jenkins**
- **placer cette configuration dans un repository partagé avec toute l'équipe souvent au niveau du code source**

Des exemples de Pipelines sont mis à disposition sur l'espace GitHub Jenkins. Dans le répertoire « pipeline-examples » .

C'est quoi un pipeline

Il se caractérise par un fichier Jenkinsfile

Documentation

<https://jenkins.io/doc/book/pipeline/syntax/>

```
pipeline {  
  agent {  
    label "windows"  
  }  
  tools {  
    maven 'Maven3.1.1'  
    jdk 'java8'  
  }  
  stages {  
    stage ('Initialize') {  
      steps {  
        bat '''  
          echo "PATH = %PATH%"  
          echo "M2_HOME = %M2_HOME%"  
          '''  
      }  
    }  
    stage ('Build') {  
      steps {  
        bat 'cd NumberGenerator & mvn install'  
      }  
      post {  
        success {  
          junit 'NumberGenerator/target/surefire-reports/*.xml'  
        }  
      }  
    }  
  }  
}
```

pipeline : definition du pipeline

agent : Label Jenkins et agents à lancer

tools : Outils nécessaires au pipeline et configurables dans l'administration des outils

stages : Processus du pipeline

steps : Etapes du processus

post : Action à réaliser en cas de réussite / echec de l'étape

Anatomie d'un pipeline



Structure du Jenkinsfile

Agent

Permet de définir où exécuter le pipeline

- **Via un esclave ou directement dans le host ?**
- **Via quel esclave (identifié par son label) ?**
- **Dans un conteneur Docker ?**
 - Directement sous forme d'image
 - Avec la compilation d'un Dockerfile

```
agent any

agent {
    label "slave-with-
python2.7"
}

agent {
    docker {
        image: "python:2.7"
    }
}

agent {dockerfile true}
```

Structure du Jenkinsfile

Post

Permet de définir les actions de fin de pipeline

- **Archivage**
- **Publication des résultats**
- **Envoi de mail**
- ...

Les actions peuvent dépendre de l'état du pipeline

- | | |
|--------------|------------|
| • Always | • Aborted |
| • Changed | • Failure |
| • Fixed | • Success |
| • Regression | • Unstable |
| | • Cleanup |

```
post {
    always {
        archive "build/*.exe"
        deleteDir()
    }
    failure {
        echo "Failure"
    }
    success {
        echo "Success"
    }
    unstable {
        echo "Unstable"
    }
}
```

Structure du Jenkinsfile

Stages

Chaque Stage permet de définir :

- un groupe d'étapes (step) séquentiel
- un groupe d'étapes à exécuter en parallèle

Chaque Stage peut avoir :

- des condition d'exécution
- ses propres variables d'environnement
- ses propres agents

```
stages {  
    stage("Stage-1") {  
        steps {  
            echo "Welcome in stage 1"  
            sh "python script.py"  
            sh "./script.sh"  
        }  
    }  
    stage('Stage-2') {  
        when {  
            branch "master"  
        }  
        environment {  
            MY_VARIABLE = "My-Value"  
        }  
        parallel {  
            stage("Sub-stage-1") {  
                steps {  
                    echo "sub stage 1"  
                }  
            }  
            stage("Sub-stage-2") {  
                steps {  
                    echo "sub stage 2"  
                }  
            }  
        }  
    }  
}
```

Structure du Jenkinsfile

Steps

Une étape (step) permet de caractériser une action simple.

Chaque plugin Jenkins peut embarquer ses propres étapes.

Les étapes se déroulent séquentiellement au sein d'un stage.

Chaque étape peut avoir sa propre sortie de log.

Une liste complète des étapes est disponible ici :

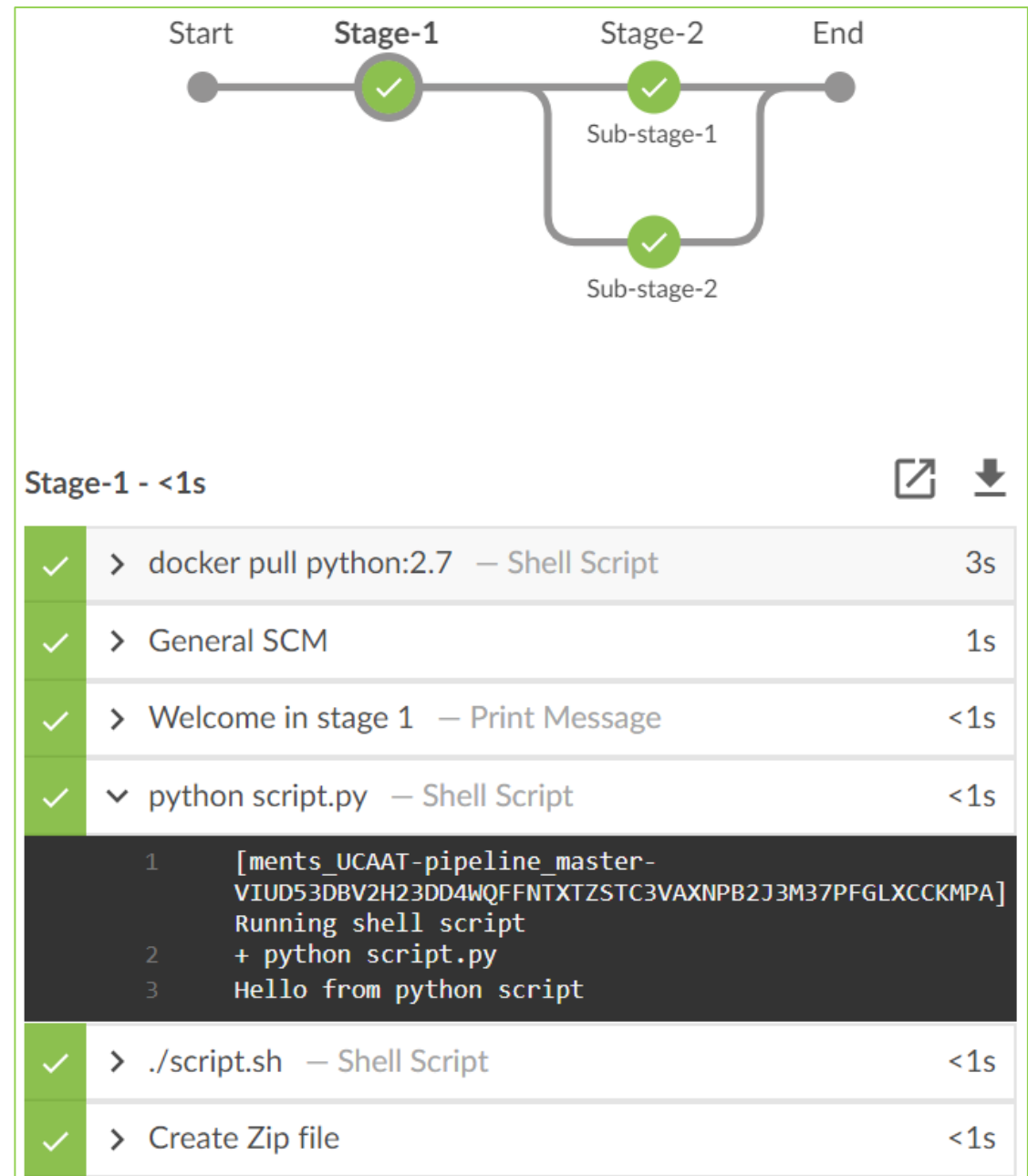
<https://jenkins.io/doc/pipeline/steps/>

```
steps {  
    addBadge icon: 'computer.png', text: env.NODE_NAME  
    echo "Welcome in stage 1"  
    sh "python script.py"  
    sh "./script.sh"  
    zip zipFile: "compressed.zip", dir: "."  
}
```

```
post {  
    always {  
        junit "build/testresults.xml"  
        checkstyle pattern: 'build/checkstyle.xml'  
        cobertura coberturaReportFile: 'build/coverage.xml'  
        sloccountPublish pattern: 'build/sloccount.scc'  
        archive "build/*.exe,build/*.rpm"  
        deleteDir() // Delete workspace  
    }  
}
```

Exemple de Jenkinsfile

```
stages {
  stage("Stage-1") {
    steps {
      echo "Welcome in stage 1"
      sh "python script.py"
      sh "./script.sh"
      zip zipFile: "compressed.zip", dir: "."
    }
  }
  stage('Stage-2') {
    when {
      branch "master"
    }
    environment {
      MY_VARIABLE = "My-Value"
    }
    parallel {
      stage("Sub-stage-1") {
        steps {
          echo "sub stage 1"
        }
      }
      stage("Sub-stage-2") {
        steps {
          echo "sub stage 2"
        }
      }
    }
  }
}
```



Comment ça marche !

La création


- On crée un Jenkinsfile comme vu précédemment
- On commit ce Jenkinsfile dans un Git/SVN
- On ajoute les plugins et outils configurables via la configuration globale des outils


Création d'un traitement


A) Ajout d'un nouveau traitement ("job") de type Pipeline


Enter an item name


» Required field


 **Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, compile something other than software build.

 **Maven project**
Build a maven project. Jenkins takes advantage of your POM files and dependencies.

 **Pipeline**
Orchestrates long-running activities that can span multiple build slaves. Suitable for organizing complex activities that do not easily fit in free-style job type.

 **External Job**
This type of job allows you to record the execution of a process run outside Jenkins. Use Jenkins as a dashboard of your existing automation system.

 **Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on different operating systems or hardware.

 **Folder**
Creates a container that stores nested items in it. Useful for grouping similar items.

OK

Comment ça marche

La création (suite)

B) On renseigne un nom et une description

C) Dans la section Pipeline : On sélectionne “pipeline script from scm” et on renseigne les informations de connexion au repository contenant le Jenkinsfile.

Jenkins pourra ainsi détecter tout changement sur les étapes du pipeline en scrutant les “Checkout” automatiquement.

D) On sélectionne le compte d'accès, la branche de travail où est stocké le Jenkinsfile, le chemin du Jenkinsfile.

pipelinejobtuto1

The screenshot shows the 'General' tab of the Jenkins Pipeline configuration for 'pipelinejobtuto1'. The 'Pipeline name' field is filled with 'pipelinejobtuto1'. The 'Description' field contains 'run pipeline job'. Below the description, there is a '[Plain text] Preview' link. At the bottom, there is a checkbox labeled 'Discard old builds' which is currently unchecked.

pipelinejobtuto1

The screenshot shows the 'Pipeline' tab of the Jenkins Pipeline configuration for 'pipelinejobtuto1'. The 'Definition' dropdown is set to 'Pipeline script from SCM'. The 'SCM' dropdown is set to 'Git'. Under 'Repositories', the 'Repository URL' is 'https://github.com/amritsql/maven_java_tutoria' and the 'Credentials' are 'amritsql/***** (my github account)'. There is an 'Add Repository' button. Under 'Branches to build', the 'Branch Specifier (blank for \'any\')' is set to '*/master' and there is an 'Add Branch' button. The 'Repository browser' is set to '(Auto)'. Under 'Additional Behaviours', there is an 'Add' button. At the bottom, the 'Script Path' is 'Jenkinsfile' and the ' lightweight checkout' checkbox is checked. There are 'Save' and 'Apply' buttons at the bottom left.

Comment ça marche !

Le build

On lance le build via "Build now" et on voit la creation du build en temps reel.

 [Back to Dashboard](#)

 [Status](#)

 [Changes](#)

 [Build Now](#)



 [Delete Pipeline](#)


 [Configure](#)


 [Full Stage View](#)


 [Job Config History](#)


 [Pipeline Syntax](#)

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



 **#32**

 May 23, 2017 9:11 PM 

 **#31**

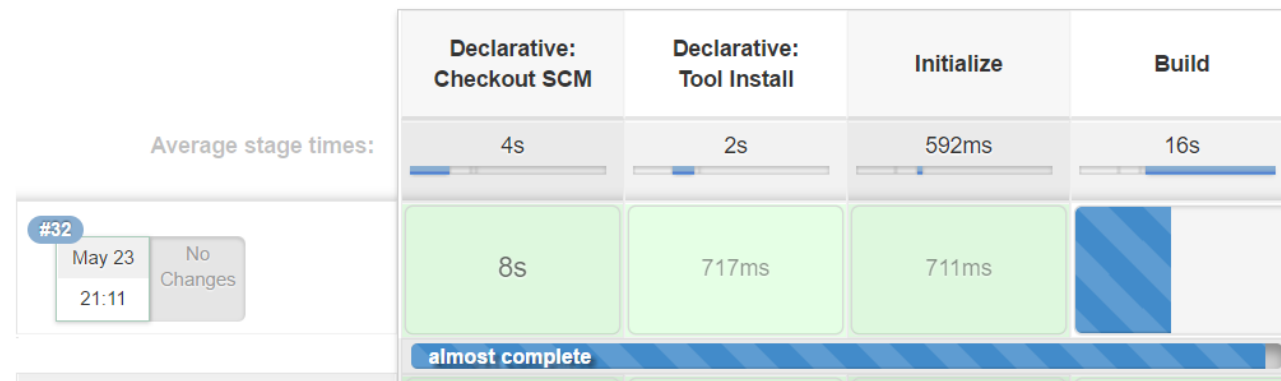
 May 21, 2017 1:36 PM

Pipeline pipelinejobtuto1

run pipeline job



Stage View



Exemple de vue Jenkins

Les différentes build



14.0.0 - Stage View



On peut accéder directement aux logs par un simple click

Build réussie

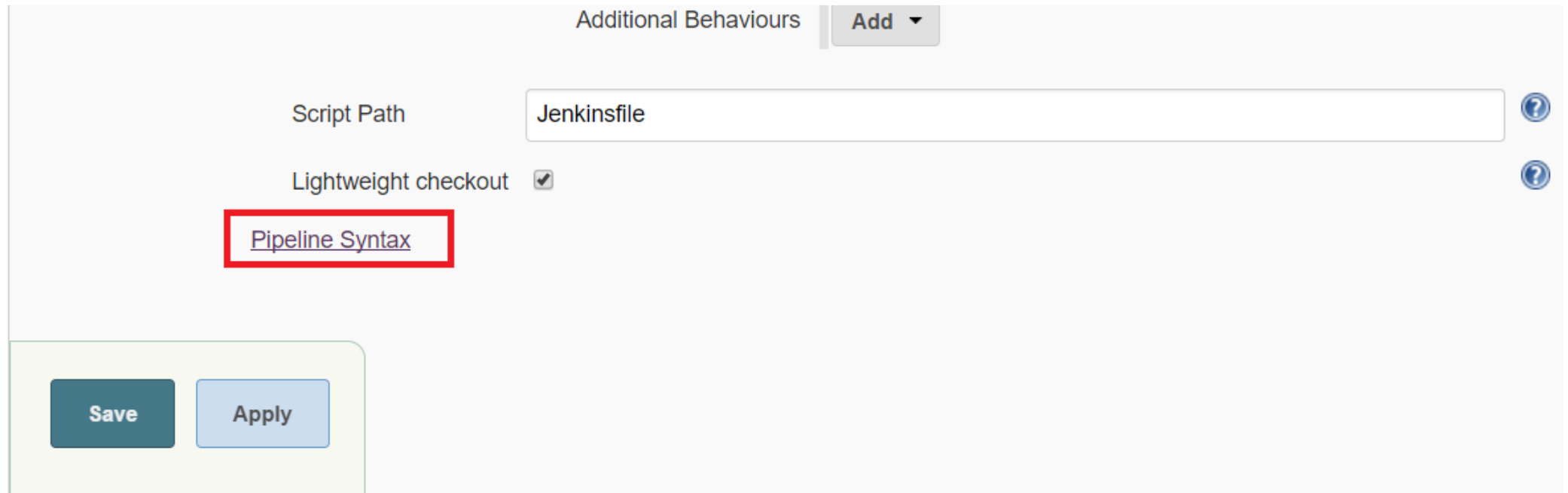
Build non déployée

Build non stable

Comment ça marche !

Aide à la création du script

Il est possible de récupérer le script par défaut en allant dans la configuration du pipeline et en cliquant sur le lien présent en bas du formulaire.



The screenshot shows the 'Additional Behaviours' section of the Jenkins Pipeline Configuration page. At the top, there is a tab labeled 'Additional Behaviours' and a button labeled 'Add' with a dropdown arrow. Below this, there is a text input field for 'Script Path' containing the text 'Jenkinsfile'. To the right of this field is a help icon (a blue circle with a question mark). Below the 'Script Path' field, there is a checkbox labeled 'Lightweight checkout' which is checked. To the right of this checkbox is another help icon. Below the 'Lightweight checkout' checkbox, there is a link labeled 'Pipeline Syntax' which is highlighted with a red rectangular box. At the bottom left of the form, there are two buttons: 'Save' (dark blue) and 'Apply' (light blue).

Comment ça marche !

Aide à la création du script (suite)

Vous avez alors accès à un outils d'aide à la création de script Jenkins à partir de différentes sources (script batch Windows, script Linux, ...)

Pipeline Syntax

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)

Steps

Sample Step bat: Windows Batch Script ▼

Batch Script echo "amrit"

Advanced...

Generate Pipeline Script

bat 'echo "amrit"'

Pipeline ou gestion des tâches manuellement ?

Le pipeline permet de :

- Gérer plus facilement les gros projets
- De suivre les changements via l'archivage des Jenkinsfile
- De versionner les Jenkinsfiles
- De créer des Jenkinsfile par sous projet
- De rejouer d'anciennes versions de Jenkinsfile (pour de la maintenance par exemple)
- De travailler à plusieurs sur les mêmes Jenkinsfile

Le pipeline n'est pas parfait :

- La conception de Jenkinsfile peut être chronophage. Il faut comprendre les subtilités de certaines syntaxes.
- Il est plus complexe à déboguer en cas de problème
- Il ne permet pas toutes les actions possibles en mode manuel
- L'accumulation de Jenkinsfile au sein d'un projet, peut le rendre plus difficile à maintenir

05

Blue Ocean

Blue Ocean c'est quoi

LE plugins

- **Un plugin Jenkins**
- **Apparu en 2016, la communauté est encore très active**
- **Il permet d'enrichir l'expérience utilisateur via :**
 - Une interface graphique repensée
 - Une nouvelle syntaxe, toutes les tâches peuvent être scriptées
 - Une édition graphique des pipeline
- **Construit autour de la modélisation des pipelines**
- **Précision améliorée des étapes affichée sur l'UI afin de trouver rapidement l'origine des anomalies**
- **Intégration forte avec Docker**

-INT 4948 Pipeline Modifications Tests Artefacts Déconnexion

Branche: master 35m 31s Modifications par
Commit: 4418d95 12 days ago Branch indexing

Description

Start Prepare Build Test Package Publish Deploy Basic Test End

Prepare - 40s

Step	Duration
> Record trace of a Docker image used in FROM	<1s
> docker build -t 7101a100c6baff3e502ede074a8d6e7f8cf3a73e -f "Dockerfile" "" -- Shell Script	<1s
> Dockerfile -- Read file from workspace	<1s
> Checks if running on a Unix-like node	<1s
> General SCM	1s
> Set job properties	<1s
> Update the commit status in GitLab	<1s
> make tools -- Shell Script	40s

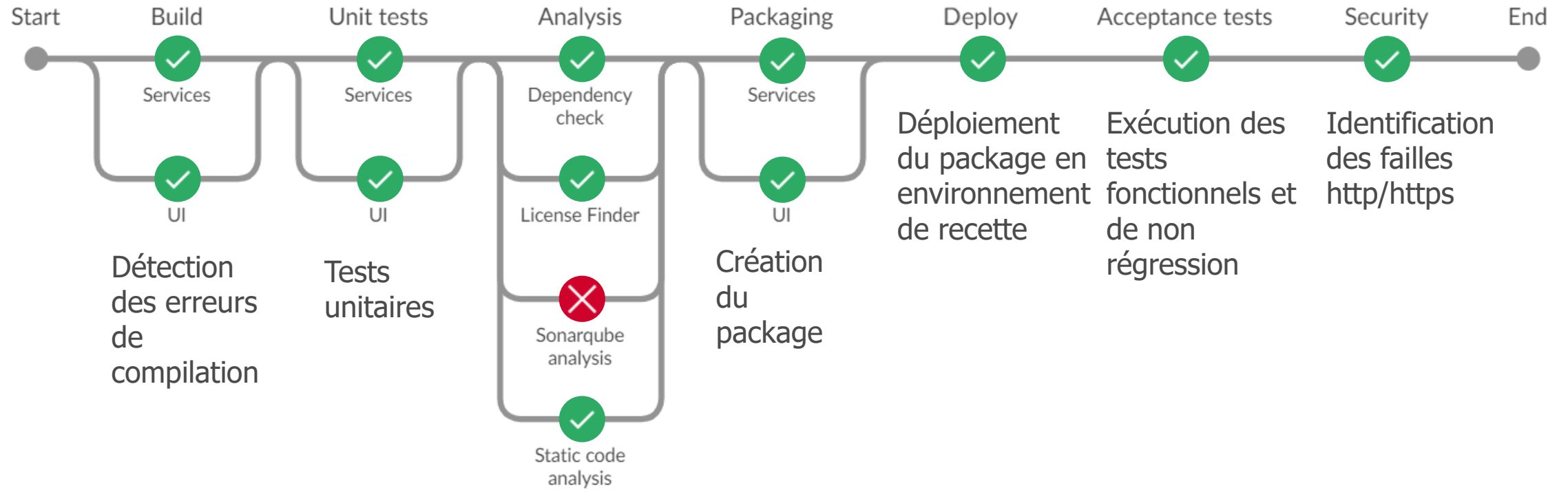
```
1 [VES-Agent_master-2PLERRN3XX75F6UWH5FEJ6RMJND4TWS4ZF7H2PMWBTQ8BT3RA] Running shell script
2 + make tools
3 GOPATH: /go:/var/lib/jenkins/workspace/VES-Agent_master-2PLERRN3XX75F6UWH5FEJ6RMJND4TWS4ZF7H2PMWBTQ8BT3RA
4 PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/go/bin
5 go get -u github.com/alecthomas/gometalinter
6 go: disabling cache (/home/jenkins/.cache/go-build) due to initialization failure: mkdir /home/jenkins: permission denied
```

L'editeur de pipeline



Blue Ocean

Exemple de pipeline étendu



Blue Ocean

Une interface Jenkins standard améliorée

