



Annexe au CCTP Description PIC

MP 29-06

- **Acheteur :**

Agence de services et de paiement (ASP)
2, rue du Maupas
87040 LIMOGES cedex 1

Documentation utilisateurs de Jenkins ASP (PIC)

Sommaire

1. Présentation de la Plateforme d'Intégration Continue (PIC)
 - 1.1 Les outils
 - 1.2 Fonctionnalités mises à disposition des utilisateurs
 - 1.2.1 Mode de fonctionnement
 - 1.2.2 Librairie de Pipelines
 - 1.2.3 Le schéma d'architecture générale de la PIC
2. Présentation des différents pipelines
 - 2.1 GenericPipeline
 - 2.2 NrtPipeline
 - 2.3 BonitaPipeline
3. Usage du Jenkinsfile
 - 3.1 GenericPipeline
4. Organisation dans Jenkins
5. Les droits dans Jenkins

1. Présentation de la Plateforme d'Intégration Continue (PIC)

La plate-forme d'intégration Continue (PIC) est un ensemble d'applications dont le but est la construction automatique de livrables projets, à partir de leur code sources et d'autres dépendances.

Les missions principales de la PIC sont :

- La compilation de binaires depuis le gestionnaire de sources (**GitLab**)
- La création et le stockage de livrables (via **Nexus / Harbor**)
- Le suivi de la qualité (bonnes pratique de programmation, tests unitaires, test robot)
- L'automatisation de ces tâches (tâches planifiées, webhook)
- Le déploiement sur les environnements

Il s'agit d'une architecture modulaire, variable selon les besoins, qui se base sur plusieurs applications principales (**GitLab** pour la gestion du code source, **Sonarqube** pour l'analyse de qualité de code, etc.) dont le cœur est l'application **JENKINS**. Jenkins s'interface aux autres applications via des plugins internes. Le comportement d'un projet Jenkins est décrit de deux manières :

- Les projets manuels créés via interface web
- Le fichier **Jenkinsfile**, intégré au code d'un projet, qui décrit les étapes d'exécution en code Groovy

1.1 Les outils

- **Jenkins : orchestrateur principal**

Jenkins fait l'interface entre les différentes briques techniques de l'Usine Logicielle. Il est synchronisé à Git et détecte tout nouveau projet, commit et branche sur les groupes configurés.

Jenkins interprète et exécute tout projet contenant un fichier Jenkinsfile, qui sert à déclarer ses paramètres. Jenkins

met alors en œuvre l'exécution d'un pipeline via un nœud.

Jenkins fait appel à des nœuds temporaires via des images Docker pour toute exécution. Ces nœuds sont spécialisés en fonction du projet.

Chaque étape du pipeline exécuté par Jenkins peut faire appel à un autre composant selon la configuration du Jenkinsfile.

Le serveur Jenkins principal est Automate2 : <https://automate2.asp-public.fr/>

Autres: <https://automate3.asp-public.fr/> | <https://ceres.asp-public.fr/>

▪ **Git : gestionnaire de sources**

Git est un gestionnaire de sources. L'ASP utilise GitLab comme serveur Git principal.

Jenkins exécute un pipeline à partir des sources présentes, en se basant sur un fichier Jenkinsfile présent dans le projet Git.

Le serveur GitLab est Junon : <https://junon.asp-public.fr>

▪ **Nexus : stockage des artefacts générés et des dépendances**

Nexus permet de stocker les artefacts générés par Jenkins (archives Java, sources php, etc.). Il sert également aux projets Java ou Angular pour la construction via la récupération des dépendances nécessaires.

Le serveur Nexus est Artifact : <http://artifact.asp-public.fr>

*Dans le Jenkinsfile, le stockage dans Nexus est exécuté par le **stageNexus**.*

▪ **Maven : construction des applications Java**

Maven est l'outil de compilation, de test et de packaging des applications Java. La configuration de Maven se fait via des fichiers « pom.xml » présents dans l'arborescence d'un projet Java. Il récupère ses dépendances via Nexus et y stocke les artefacts générés.

Maven est présent dans les nœuds Jenkins qui permettent la construction de projet Java.

*Dans le Jenkinsfile d'un projet Java, la construction Maven est lancée par le **stageBuild**.*

▪ **Sonar : qualité du code**

Sonar analyse le code source de l'application afin de détecter la présence de mauvaises pratiques ou de vulnérabilités, et aider à améliorer la qualité du code d'un projet. Depuis l'interface Sonar, on peut ainsi accéder à son projet et voir les alertes et conseils de l'outil, ainsi que l'évolution du code du projet.

Le serveur Sonar est QualiteCode : <http://qualitecode.asp-public.fr/projects>

*Dans le Jenkinsfile, le lancement d'une analyse Sonar se fait par le **stageQuality**.*

- **Docker : création et gestion d'applications dans des containers**

Docker est un outil qui permet de créer et exécuter des conteneurs. Ceux-ci contiennent à la fois les applications et l'environnement qui permet de les exécuter. L'intérêt est de les rendre plus facilement exploitables et déployables sur différents environnements. Les conteneurs sont créés à partir d'images décrites dans des Dockerfile.

Docker est contenu dans les nœuds Jenkins exécutant les pipelines.

*Le **stageDocker** permet la création d'images Docker.*

- **Harbor : accès aux images Docker et stockage des images Docker**

Harbor est un outil qui permet de stocker les images Docker générés par un projet à l'issue du stageDocker.

Les images contenues dans Harbor sont également appelées lors du déploiement d'un projet conteneurisé par Docker.

Le serveur Harbor est Registry : <https://registry.asp-public.fr>

- **Epervier : outil de déploiement**

Epervier permet le déploiement des applications via des patches préalablement créés. Il permet également un suivi des actions de déploiement d'un projet.

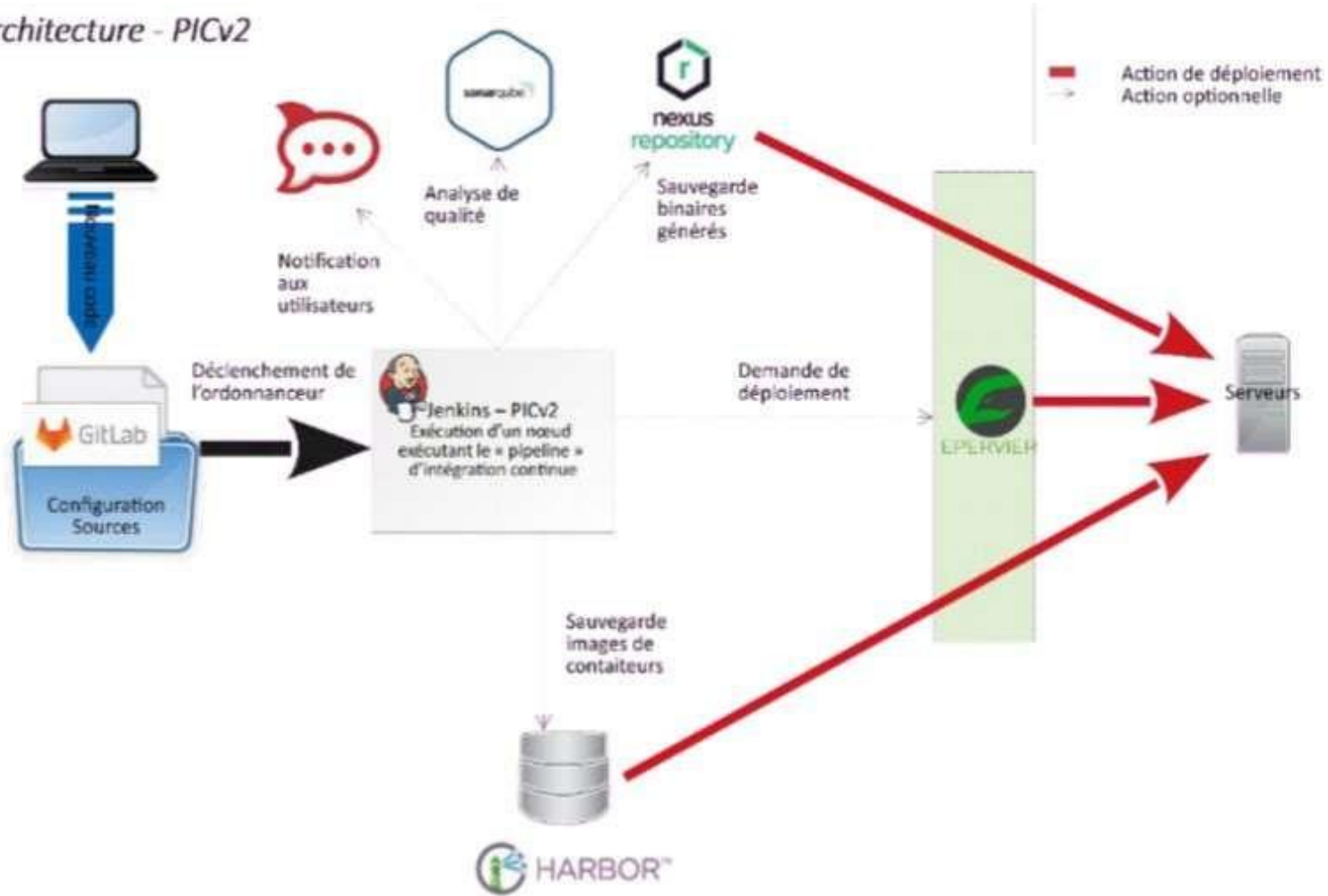
Jenkins peut via API créer des patches Epervier et déclencher leur déploiement.

Le serveur Epervier : <https://epervier.asp-public.fr/epervier/>

*La création de patch se fait via le **stageEpervier**.*

Le déploiement d'un patch Epervier se fait via le **stageDeploy**.

Architecture - PICv2



1.2 Fonctionnalités mises à disposition des utilisateurs

1.2.1 Mode de fonctionnement

Jenkins déclenche des « pipelines » par l'intermédiaire de « nœuds » (ou « slaves »).

Les nœuds sont des environnements d'exécution fournis par Jenkins. Ils peuvent être un autre serveur (Linux ou Windows) et dans le cas de l'ASP ils sont des images Docker lancées au besoin lors de l'exécution d'un pipeline.

L'Usine Logicielle fournit plusieurs nœuds en fonction des besoins d'un projet :

- Nœuds Java (slave_jdk8, slave_jdk11, slave_jdk17, slave_jdk21) : ces nœuds permettent l'exécution d'un pipeline pour un projet Java selon la version demandée. Ils permettent une construction Maven.
- Nœud générique (slave_generic) : nœud généraliste. Il permet la construction de projets Angular ou PHP.
- Nœud PHP (slave_php) : nœud spécifique pour les projets PHP8.

Les pipelines sont des algorithmes exécutés par Jenkins et paramétrés dans les Jenkinsfiles présents dans les sources d'un projet. Ils sont découpés en « stages », qui sont les différentes étapes d'un pipeline (par exemple : construction, analyse de qualité, déploiement).

Les pipelines ASP sont standards et fournis par la librairie de l'Usine Logicielle. Le Jenkinsfile d'un projet contient leur déclaration et leur paramétrage sur ce modèle :

- Appel de la librairie
- Déclaration du pipeline
- Paramètres des différents stages

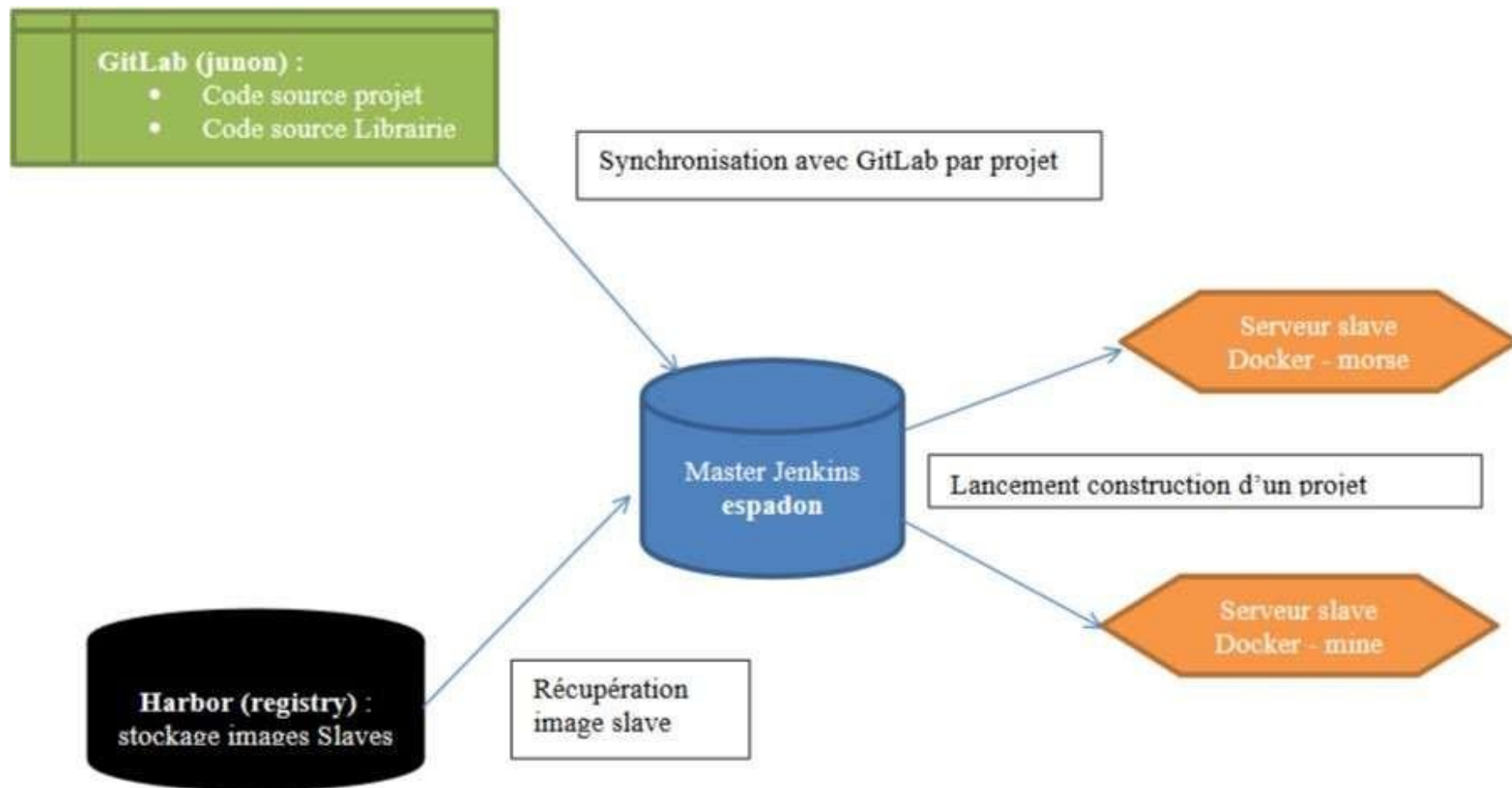
1.2.2 Librairie de Pipelines

Via la librairie **PIC_Libraries** mise à disposition via sur Gitlab (https://junon.asp-public.fr/sul/pic/pic_libraries) écrites en groovy et développé en interne à ASP, la PIC permet aux utilisateurs de réaliser les actions suivantes :

- de construire, de déployer et de faire d'analyse de code d'application ou des API pour les langages PHP, Java et Typescript via le pipeline générique **GenericPipeline**
- de construire et de livrer des projets Bonita en interface avec un slave Jenkins BCD (Bonita Continuous Delivery) via le pipeline spécifique **BonitaPipeline**
- de livrer des fichiers de configuration pour les environnements de développement et d'Intégration via un pipeline spécifique **ConfigPipeline**
- d'exécuter des tests de Non Regression en gherkin via le pipeline spécifique **NrtPipeline**

Chacun de ces pipelines est découpé en **stage** pour exécuter les fonctionnalités fournis pour chacun.

1.2.3 Le schéma d'architecture générale de la PIC



2. Présentation des différents pipelines

2.1 GenericPipeline

Le GenericPipeline est le pipeline recommandé pour la plupart des projets qui utilisent les langages PHP, Java ou Typescript.

- dans MyASP : https://my.asp-public.fr/myasp/jcms/113373782_ASPDBDocument/genericpipeline?details=true
- dans Junon : https://junon.asp-public.fr/sul/pic/pic_libraries/-/blob/stable/doc/GenericPipeline.md

2.2 NrtPipeline

Le NrtPipeline est l'unique pipeline pour exécuter des tests de non regressions via Jenkins sur la plateforme de test en continu Protest.

- dans MyASP : https://my.asp-public.fr/myasp/jcms/113373836_ASPDBDocument/nrtpipeline?details=true
- dans Junon : https://junon.asp-public.fr/sul/pic/pic_libraries/-/blob/stable/doc/NrtPipeline.md

2.3 BonitaPipeline

Le BonitaPipeline est l'unique pipeline dédié au projet Bonita. il permet de construire et de livrer des projets Bonita en interface avec un slave Jenkins BCD (Bonita Continuous Delivery).

- dans MyASP : https://my.asp-public.fr/myasp/jcms/113373755_ASPDBDocument/bonitapipeline?details=true
- dans Junon : https://junon.asp-public.fr/sul/pic/pic_libraries/-/blob/stable/doc/BonitaPipeline.md

3. Usage du Jenkinsfile

Un Jenkinsfile (qui s'écrit en un seul mot) est un fichier texte au format Groovy qui décrit un pipeline Jenkins. Il détaille ses différentes phase du processus d'intégration continue : package Maven, git pull, notification Rocketchat, etc.

NOTE : Le document GettingStarted mis à disposition dans l'espace GitLab de la librairie indique les templates de Jenkinsfiles et les différents noeuds et options disponibles : https://junon.asp-public.fr/pic/pic_libraries/-/blob/stable/doc/GettingStarted.md?ref_type=heads

Voici un exemple de Jenkinsfile et du pipeline GenericPipeline couramment utilisé à l'ASP :

```
@Library('PIC_libraries')_

GenericPipeline("Typescript",[
  node: "slave_generic",
  rocketChatChannel: "SUL_outils",
  stageBuild: [
    enabled: true,
    buildArgs: "",
    executeTests: true,
    executeTestsCi: false
  ]
])
```

La première ligne du **Jenkinsfile** importe la librairie **PIC_libraries** et utilise la branche **stable** sur les PICs de production
Vous pouvez spécifier une branche particulière en ajoutant **@nom-de-la-branche** avant le quote (') de fermeture.

```
@Library('PIC_libraries@master')_
```

Cela indique à Jenkins d'utiliser la branche 'master' de la PIC_Libraries

3.1 GenericPipeline

Le *GenericPipeline* est le pipeline recommandé pour la plupart des projets qui utilisent les langages PHP, Java ou Typescript.
Ce pipeline permet de tester, construire et déployer des applications en *Java*, *Typescript*, *PHP*, *Docker*, *API*.

Les stages disponibles, qui sont accessibles selon le langage déclaré sont :

- Build
- Shell
- Qualité
- Release
- Nexus
- Docker
- Epervier
- Déploiement
- Déploiement d'API
- Workflow

Chacun de ces stages sont activables en utilisant l'option *enable* comme suit :

```
GenericPipeline("Typescript",[
  node: "slave_generic",
  rocketChatChannel: "SUL_outils",
  stageBuild: [ //
    enabled: true, //Activation du stage
    buildArgs: "",
    executeTests: true,
    executeTestsCi: false
  ]
])
```

4. Organisation dans Jenkins

Dans Jenkins, les pipelines des projets sont organisés en **Organization Folder**. Un **Organization Folder** permet à Jenkins de surveiller l'intégralité d'un groupe GitLab et de créer automatiquement de nouveaux pipelines multibranches pour les dépôts dont des branches et / ou des Merge Request contiennent un **Jenkinsfile**.

Voici une exemple d'**Organization Folder** pour le projet **Osyris** :

The screenshot shows the Jenkins 'Configuration' page for an 'Organization Folder' named 'GitLab Group'. The left sidebar contains a 'Configuration' section with sub-items: General, Projects (selected), Scan GitLab Group Triggers, Orphaned Item Strategy, Appearance, Health metrics, and Properties. The main content area is titled 'GitLab Group' and includes the following fields:

- Servers**: A dropdown menu showing 'Gitlab Junon (https://junon.asp-public.fr)'.
- Checkout Credentials**: A dropdown menu showing '- aucun -' and an 'Ajouter' button.
- Owner**: A text input field containing 'OSIRIS'.
- A confirmation message: 'OSIRIS is a valid Group'.
- Behaviours**: A section for defining branch patterns.

At the bottom, there are 'Save' and 'Appliquer' buttons.

Cette configuration indique à Jenkins de surveiller les dépôts du groupe **OSIRIS** dans **Gitlab (Junon)** dont voici l'arborescence :



Search GitLab



1



9



OSIRIS

Group information

Issues1

Merge requests1

CI/CD

Packages and registries

Settings

« Collapse sidebar

OSIRIS

OSIRIS

Group ID: 303



New subgroup

New project

Subgroups and projects

Shared projects

Archived projects

Search

Name

 B	batch-tpl-pb-edtpaiement	★ 0	3 days ago
 B	bom-socle-technique	★ 0	2 weeks ago
 C	certificationfeamp.notificationAppelDeFonds	★ 0	2 weeks ago
 C	certificationfeamp.notificationAppelDeFonds-batch	★ 0	2 weeks ago
 C	certificationfeamp.notificationComptesAnnuels	★ 0	2 weeks ago
 C	certificationfeamp.recuperationCRSifa	★ 0	2 weeks ago

Et voici la liste des dépôts qui contiennent un **Jenkinsfile** :

Tableau de bord Osiris Applications du programme RDR3 Osiris

Folder name: OSIRIS

Disable Organization Folder

Repositories (6)

S	M	Name	Description
		osiris/API	Support des API Osiris pour OCTAV
		osiris/osirisfwk	
		osiris/osirisgenerique	
		osiris/osirisguh	
		osiris/osirisservice-dynamicui-parent	
		osiris/sorgho	Outil de gestion des habilitations pour Osiris et Valosiris

Utilisateurs Historique des constructions Relations entre les builds Vérifier les empreintes numériques Déplacer Job Config History Open Blue Ocean

Group

Icône: S M L Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

Ce fonctionnement permet de mutualiser la configuration du pipeline pour l'ensemble des dépôts du groupe Gitlab (Junon).

Sur un **Organization Folder**, Il est possible de configurer les éléments suivants :

- les sources à surveiller (un ou plusieurs groupes Gitlab, un projet Gitlab, etc) ainsi que les éléments à surveiller (branches, tags, MR, etc) via la section **Repository Sources**
- la nom du ou des Jenkinsfile dans le projet Gitlab via la section **Project Recognizers**

Project Recognizers

≡ Pipeline Jenkinsfile

Script Path ?

Jenkinsfile

≡ Pipeline Jenkinsfile

Script Path ?

JenkinsfileASP

Save

Appliquer

- la stratégie de build automatique via la section **Build strategies**
 - par branche
 - par tag
 - par MRs

Build strategies ?

≡

Tags ?

×

Ignore tags newer than ?

Ignore tags older than ?

7

- La périodicité de scan du groupe Gitlab via la section **Scan Gitlab Group Trigger**

Scan GitLab Group Triggers

☐ Build whenever a SNAPSHOT dependency is built ?

☒ Periodically if not otherwise run ?

Interval ?

12 hours

- La stratégie de nettoyage des pipelines dont le dépôt n'existe plus via la section **Orphans Item Strategy** : par défaut, les projets seront supprimés dès que Jenkins déterminera que leur référentiel associé n'existe plus.

Orphaned Item Strategy

Multibranch projects can be removed immediately when Jenkins detects that their associated repository no longer exists, or kept based on a desired retention strategy. By default, projects will be removed as soon as Jenkins determines their associated repository no longer exists.

☐ Abort builds ?

☒ Discard old items

Days to keep old items

if not empty, old items are only kept up to this number of days

7

Max # of old items to keep

if not empty, only up to this number of old items are kept

10

5. Les droits dans Jenkins

Par défaut, un utilisateur a uniquement les droits pour lancer et voir les résultats d'un build de Job Jenkins.

Via la matrice des droits, il est possible d'avoir des droits supplémentaires comme avoir les droits de modifier la configuration d'un job.

Properties

☒ Enable project-based security

Inheritance Strategy

Inherit permissions from parent ACL

This item will inherit its parent item's permissions (in addition to any permissions granted here). If this item is at the top level in Jenkins, it will inherit the [global security security settings](#).

Utilisateur/groupe	Identifiants			Job							Historique des builds			Vues			Job Config History	Gestion de version					
	Create	Delete	Manage Domains	Update	View	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Configure	Create	Delete	Read	Delete Entry	Log
 Anonyme	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒ 
 Authenticated Users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒ 
 eric.tricone	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐	☒	☐	☒ 
 frederic.seraphine	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒	☒ 
 henrique.ferreira	☐	☐	☐	☐	☐	☒	☒	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐	☒	☐	☒ 
 henrique.ferreira2	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐	☐	☐	☐	☒	☐	☒ 

Add user...

Add group...

?