

Jenkins

Mettre en place la CI/CD



Compétence demandée :
Savoir écrire un Jenkinsfile

1. Organisation de Jenkins
2. Declarative pipeline (vs scripted pipeline)
3. Codons le pipeline
4. Jenkinsfile

Organisation Jenkins

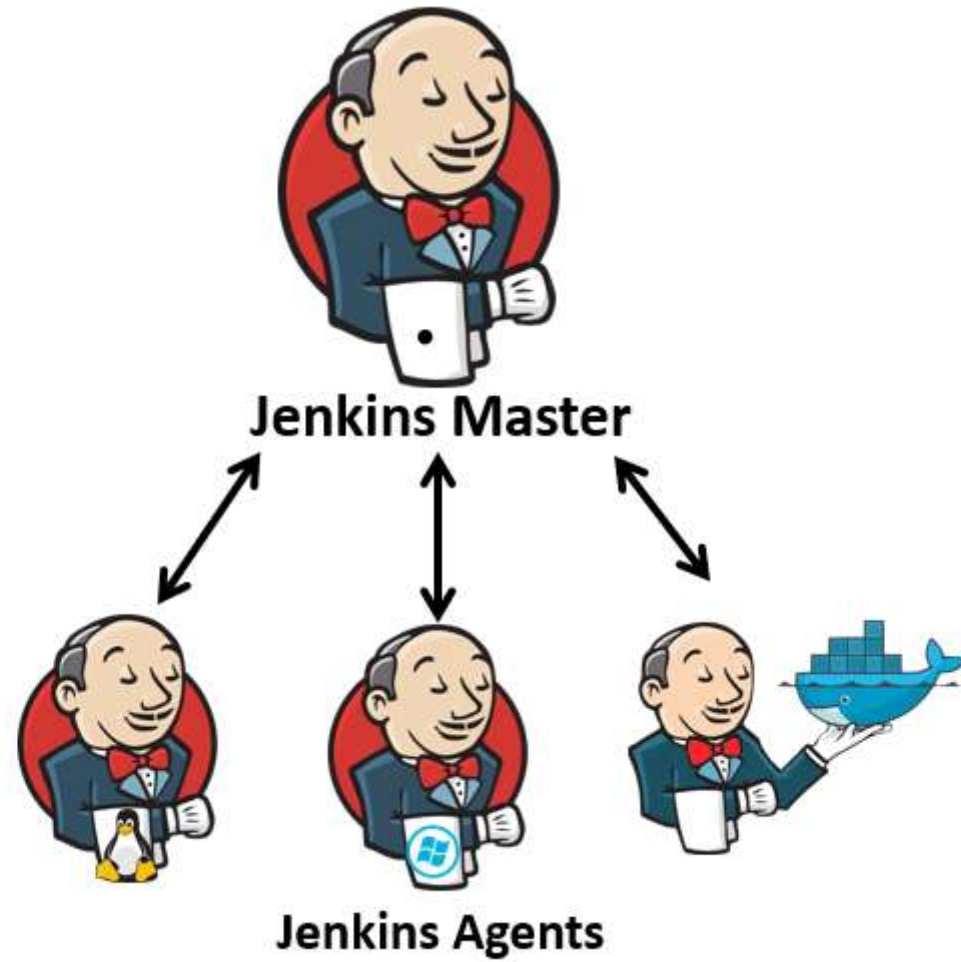
Qu'est-ce qu'un agent / node /
slave dans Jenkins ?



Un agent est un logiciel qui peut être utilisé par Jenkins pour faire tourner les jobs.

Ainsi Jenkins peut être le master/controller et peut utiliser 30 agents (sur linux, windows, mac ou même un conteneur) pour lancer les jobs.

C'est plutôt utile pour lancer des builds sur des plateformes spécifiques : utile pour les applications mobiles.



1. Organisation de Jenkins
2. Declarative pipeline (vs scripted pipeline)
3. Codons le pipeline
4. Jenkinsfile

Declarative pipeline

CREER UN PIPELINE

Saisissez un nom

erphrense-improved

» 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.



Pipeline

Organise des activités de longue durée qui peuvent s'étendre sur plusieurs agents de construction. Adapté pour la création des pipelines (anciennement connues comme workflows) et/ou pour organiser des activités complexes qui ne s'adaptent pas facilement à des tâches de type libre.



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.



Dossier

Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.

OK

☒ Use Groovy Sandbox ?

Pipeline Syntax

Sauver

Apply

A votre avis, à quoi sert cette
page ?



La page Pipeline Syntax est très utile pour savoir comment retranscrire les étapes de votre build !

Au lieu d'utiliser l'interface graphique de Jenkins pour définir les jobs, il est possible de le coder ! C'est un langage particulier, mais utile.

En effet, comme habituellement l'intégration et le déploiement se fait par projet, on pourrait compléter le projet par son code source et le code du CI/CD !

Nous allons donc apprendre à coder le pipeline réalisé lors du cours précédent ! 😊

1. Organisation de Jenkins
2. Declarative pipeline (vs scripted pipeline)
3. Codons le pipeline
4. Jenkinsfile

Codons le pipeline

Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Comment traduire ces étapes en
lignes de code ?



Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Directives

Sample Directive

agent: Agent

?

See [the online documentation](#) for more information on the agent directive.

Agent ?

any: Run on any agent

Generate Declarative Directive

agent any

Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Directives

Sample Directive

triggers: Triggers

?

See [the online documentation](#) for more information on the triggers directive.

☰ **Scrutation de l'outil de gestion de version** ?

Planning ?

H/5 * * * * *

Aurait été lancé à jeudi 24 novembre 2022 à 04:48:43 heure normale d'Europe centrale

☐ Ignore post-commit hooks ?

Ajouter ▾

Generate Declarative Directive

```
triggers {
  pollSCM 'H/5 * * * * *'
}
```


Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Steps

Sample Step

git: Git

git ?

Repository URL ?

https://github.com/xonatis-academy/epsi-dev709-ci.git

Branch ?

main

Credentials ?

- aucun -

+ Ajouter

☒ Include in polling? ?

☒ Include in changelog? ?

Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Steps

Sample Step

dir: Change current directory

dir ?

Path ?

projects/erphrense

Generate Pipeline Script

```
dir("projects/erphrense") {  
  // some block  
}
```

Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Sample Step

bat: Windows Batch Script

bat

Batch Script ?

.\mvnw clean

Avancé...

Generate Pipeline Script

bat '.\mvnw clean'

Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Steps

Sample Step

bat: Windows Batch Script

bat

Batch Script ?

.\mvnw compile

Avancé...

Generate Pipeline Script

bat '.\mvnw compile'

Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Steps

Sample Step

bat: Windows Batch Script

bat

Batch Script ?

```
rename target\erphrense-0.0.1-SNAPSHOT.jar erphrense-%BUILD_NUMBER%.jar
```

Avancé...

Generate Pipeline Script

```
bat 'rename target\erphrense-0.0.1-SNAPSHOT.jar erphrense-%BUILD_NUMBER%.jar'
```

Algorithme CI/CD :

1. Utiliser n'importe quel agent disponible
2. Ajouter un trigger pour exécuter le polling toutes les 5 minutes
3. Clone la branche principale du remote
4. Changer le répertoire courant pour aller dans le projet
5. Faire un clean
6. Faire un compile
7. Lancer les tests
8. Packager l'appli
9. Renommer l'artefact
10. Archiver l'artefact

Steps

Sample Step

archiveArtifacts: Archiver des artefacts

archiveArtifacts ?

Fichiers à archiver ?

target\erphrense-*.jar

Avancé...

Generate Pipeline Script

```
archiveArtifacts artifacts: 'target\erphrense-*.jar', followSymlinks: false
```

Script ?

```

1 pipeline {
2     agent any
3
4     triggers {
5         pollSCM 'H/5 * * * *'
6     }
7
8     stages {
9         stage('checkout') {
10             steps {
11                 git branch: 'main', url: 'https://github.com/xonatis-academy/epsi-dev709-ci.git'
12             }
13         }
14
15         stage('run') {
16             steps {
17                 dir('projects/erphrense') {
18                     bat './mvnw clean'
19                     bat './mvnw compile'
20                     bat './mvnw test'
21                     bat './mvnw package'
22                     bat 'rename target\\erphrense-0.0.1-SNAPSHOT.jar erphrense-%BUILD_NUMBER%.jar'
23                     archiveArtifacts artifacts: 'target\\erphrense-*.jar', followSymlinks: false
24                 }
25             }
26         }
27     }
28 }
29 }
```

				checkout	run
Average stage times: (Average <u>full</u> run time: ~26s)				1s	23s
<div>#10</div> <div>Nov 24 03:56</div> <div>No Changes</div> <div>ⓘ</div>				1s	23s

Comme vu avant, le job s'est bien déroulé !

Build #10 (24 nov. 2022, 03:56:54)



Build Artifacts

 **erphrense-10.jar** 11,51 KB [view](#)



Lancé par l'utilisateur [admin](#)



Revision: f50df222005182daef1258810bf9a1fa6d8f90fc

Repository: <https://github.com/xonatis-academy/epsi-dev709-ci.git>

- [refs/remotes/origin/main](#)

Comme vu avant, on a construit un bon artefact !

Dernier Log du dernier accès à Git

```
Started on 24 nov. 2022, 04:08:00
Using strategy: Default
[poll] Last Built Revision: Revision f50df222005182daef1258810bf9a1fa6d8f90fc (refs/remotes/origin/main)
The recommended git tool is: NONE
No credentials specified
> git.exe --version # timeout=10
> git --version # 'git version 2.31.0.windows.1'
> git.exe ls-remote -h -- https://github.com/xonatis-academy/epsi-dev709-ci.git # timeout=10
Found 1 remote heads on https://github.com/xonatis-academy/epsi-dev709-ci.git
[poll] Latest remote head revision on refs/heads/main is: f50df222005182daef1258810bf9a1fa6d8f90fc - already built by 10
Done. Took 0,6 s
No changes
```

Build #12 (24 nov. 2022, 04:23:10)



Build Artifacts

 [erphrense-12.jar](#) 11,51 KB [view](#)



Changes

1. [Updating readme.md \(details / githubweb\)](#)



[Lancé par un changement dans la base de code](#)





Revision: 3fffe0cf8d54160d74daedd328ba5541621e5e1f

Repository: <https://github.com/xonatis-academy/epsi-dev709-ci.git>

- [refs/remotes/origin/main](#)

Comme vu avant, on a toujours le polling pour la détection des changements !

				checkout	run
Average stage times: (Average <u>full</u> run time: ~26s)				1s	23s
					
#10	Nov 24 03:56	No Changes	🕒	1s	23s

Voyez-vous des choses à
améliorer ?

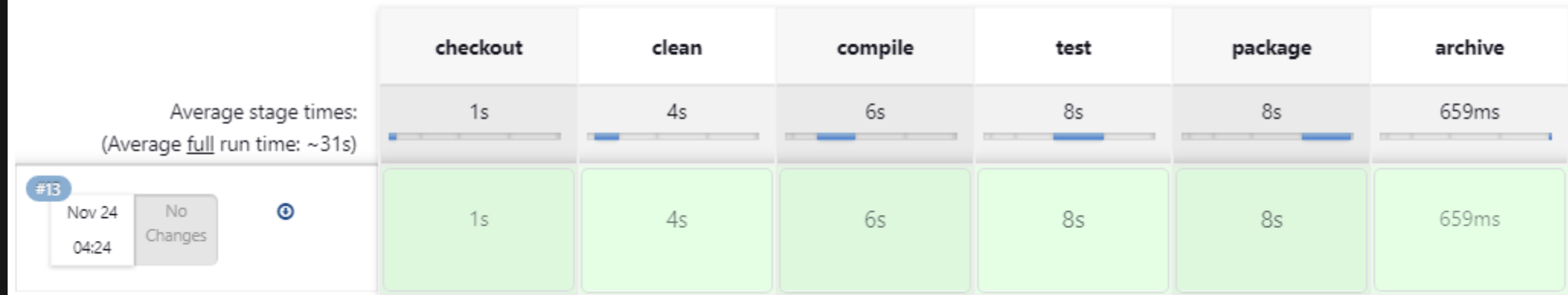


Cela dit, un pipeline comme cela est trop monolithique, il serait préférable de séparer les différentes étapes en des stages différents

Script ?

```
1 pipeline {  
2   agent any  
3  
4   triggers {  
5     pollSCM 'H/5 * * * *'  
6   }  
7  
8   stages {  
9     stage('checkout') {  
10      steps {  
11        git branch: 'main', url: 'https://github.com/xonatis-academy/epsi-dev709-ci.git'  
12      }  
13    }  
14  
15    stage('clean') {  
16      steps {  
17        dir('projects/erphrense') {  
18          bat './mvnw clean'  
19        }  
20      }  
21    }  
22  
23    stage('compile') {  
24      steps {  
25        dir('projects/erphrense') {  
26          bat './mvnw compile'  
27        }  
28      }  
29    }  
30  
31    stage('test') {  
32      steps {  
33        dir('projects/erphrense') {  
34          bat './mvnw test'  
35        }  
36      }  
37    }  
38  
39  }  
40 }
```

Stage View



<https://plugins.jenkins.io/git/>

Checkout with defaults

Checkout from the git plugin source repository using https protocol, no credentials, and the master branch.

```
checkout([$class: 'GitSCM',  
  branches: [[name: '*/master']],  
  userRemoteConfigs: [[url: 'https://github.com/jenkinsci/git-plugin.git']]])
```

Checkout with a specific branch

Checkout from the git client plugin source repository using https protocol, no credentials, and a non-default branch.

```
checkout([$class: 'GitSCM',  
  branches: [[name: 'stable-2.x']],  
  userRemoteConfigs: [[url: 'https://github.com/jenkinsci/git-client-plugin.git']]])
```

Checkout with ssh and a private key credential

Checkout from the git client plugin source repository using ssh protocol, ssh private credentials, and a non-default branch. The git plugin supports private key credentials provided by the [Jenkins credentials plugin](#).

```
checkout([$class: 'GitSCM',  
  branches: [[name: 'stable-2.x']],  
  userRemoteConfigs: [[credentialsId: 'my-ssh-private-key-id',  
    url: 'ssh://github.com/jenkinsci/git-plugin.git']]])
```

Checkout with https and a username/password credential

Checkout from the git client plugin source repository using https protocol, username/password credentials, and a non-default branch. The git plugin supports username/password credentials provided by the [Jenkins credentials plugin](#).

1. Organisation de Jenkins
2. Declarative pipeline (vs scripted pipeline)
3. Codons le pipeline
4. Jenkinsfile

Le Jenkinsfile

Il est possible de déporter le code du pipeline directement dans le SCM (git par exemple) afin de centraliser le code du projet :

- Code source de l'application
- Code du CI/CD du pipeline

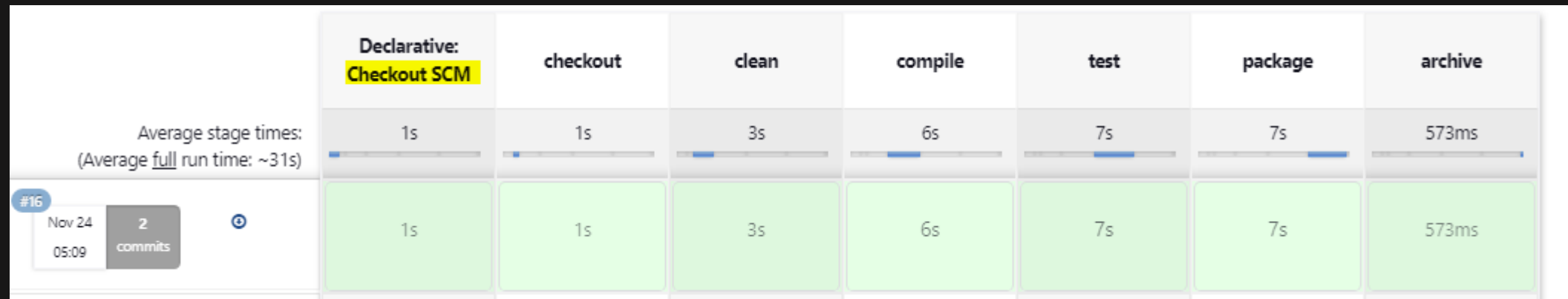
Pour ce faire, il faut créer un fichier nommé
« Jenkinsfile » (par convention)

▼ ERPHRENS
> .mvn
> src
> target
◆ .gitignore
🔥 Jenkinsfile
🔥 mvnw
🖥 mvnw.cmd
🔥 pom.xml
📄 readme.md

Jenkinsfile U X

Jenkinsfile

```
1 pipeline {
2     agent any
3
4     triggers {
5         pollSCM 'H/5 * * * *'
6     }
7
8     stages {
9
10        stage('checkout') {
11            steps {
12                checkout
13            }
14        }
15
16        stage('clean') {
17            steps {
18                dir('projects/erphrense') {
19                    bat './mvnw clean'
20                }
21            }
22        }
23
24        stage('compile') {
25            steps {
26                dir('projects/erphrense') {
27                    bat './mvnw compile'
28                }
29            }
30        }
31
32        stage('test') {
33            steps {
34                dir('projects/erphrense') {
35                    bat './mvnw test'
36                }
37            }
38        }
39    }
```



Gestion des identifiants

Afin de gérer des identifiants dans Jenkins (n'importe lesquels), l'idée est la chose suivante :

1. Enregistrer les identifiants à l'intérieur de Jenkins
2. Lier ces identifiants à des variables dans votre job
3. Utiliser ces variables dans votre job

Afin de gérer des identifiants dans Jenkins (n'importe lesquels), l'idée est la chose suivante :

1. Enregistrer les identifiants à l'intérieur de Jenkins
2. Lier ces identifiants à des variables dans votre job
3. Utiliser ces variables dans votre job

Security



Configurer la sécurité globale

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



Gérer les utilisateurs

Créer/supprimer/modifier les utilisateurs qui peuvent se logger sur ce serveur Jenkins



Manage Credentials

Configure credentials



In-process Script Approval

Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions. **1 scripts pending approval.**



Configure Credential Providers

Configure the credential providers and types

Tableau de bord > Administrer Jenkins > Identifiants

Credentials

T P Store ↓

Stores scoped to Jenkins

P Store ↓



System

Icône:

S

M

L

Tableau de bord > Administrer Jenkins > Identifiants > System >

System

Domaine ↓



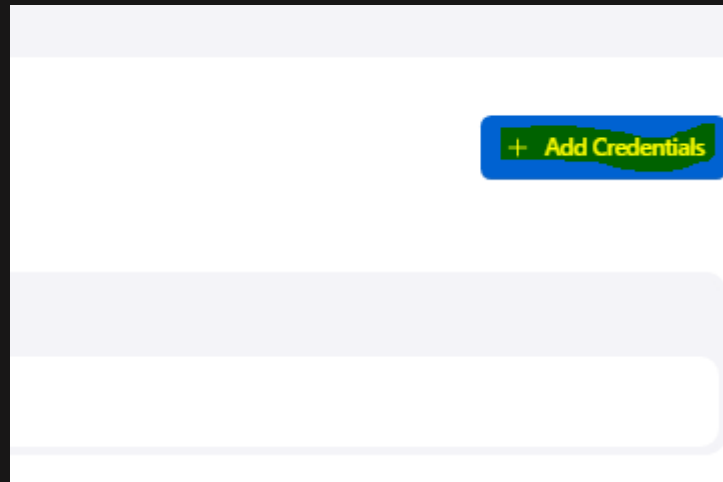
Identifiants globaux (illimité)

Icône:

S

M

L



New credentials

Type

Nom d'utilisateur et mot de passe



Nom d'utilisateur et mot de passe

GitHub App

SSH Username with private key

Secret file

Secret text

Certificat



Afin de gérer des identifiants dans Jenkins (n'importe lesquels), l'idée est la chose suivante :

1. Enregistrer les identifiants à l'intérieur de Jenkins
2. Lier ces identifiants à des variables dans votre job
3. Utiliser ces variables dans votre job

☒ Use secret text(s) or file(s) ?

Bindings

Ajouter ▴

☒ Filter

- ☐ Certificate
- ☐ Git Username and Password
- ☐ SSH User Private Key
- ☐ Secret ZIP file
- ☐ Secret file
- ☐ Secret text
- ☐ Username and password (conjoined)
- ☐ Username and password (separated)

Build

