

CI-CD

Intégration continue et la livraison continue

Jenkins

Build → Test → Release → Deploy

C'est quoi

Installation avec docker-compose

Démarrage de Jenkins

Configuration

Pipeline

Pipeline workflow

Création simple Pipeline job

Jenkins File

Création Simple Pipeline Jenkins file

Arrêt de Jenkins

CI-CD Pipeline for a Spring Boot Application avec Docker et Gradle

Installer Jenkins en local

Créer le projet jenkins-pipeline-docker-demo

Tools

Comment installer le plugin Open JDK

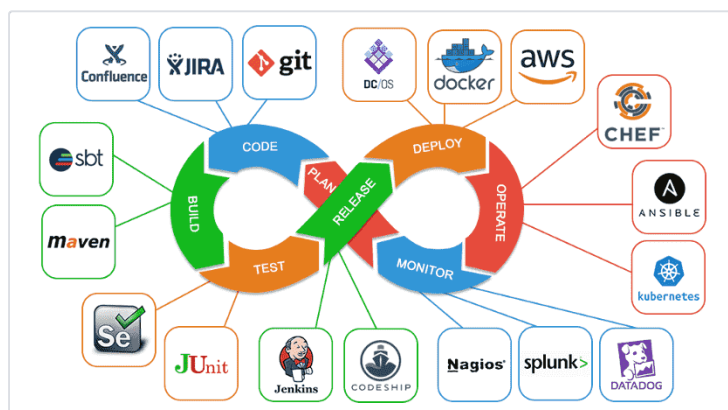
Comment installer le plugin Maven

Comment installer le plugin Gradle

Comment installer le plugin Docker

Comment créer un crédençiale pour Docker Hub

Intégration continue et la livraison continue [↗](#)



Jenkins [↗](#)

Build → Test → Release → Deploy [↗](#)

C'est quoi [↗](#)



- Un outil open source
- Automatise les parties du développement logiciel liées au build, aux tests et au déploiement
- Facilite l'intégration continue et la livraison continue CI-CD

Installation avec docker-compose [↗](#)

1. Créer le volume partagé : `docker volume create jenkins_home`
2. Liste les volumes existants : `docker volume ls`
3. Créer le répertoire de travail `jenkins-demo`
4. Editer le fichier `/jenkins-demo/docker-compose.yml`

```
1 version: '3.9'
2 services:
3   jenkins-service:
4     image: 'jenkins/jenkins'
5     container_name: "jenkins"
6     ports:
7       - 8080:8080
8     volumes:
9       - jenkins_home:/var/jenkins_home
```

```
10 volumes:
11     jenkins_home:
```

Démarrage de Jenkins

```
docker-compose up
```

```

jenkins@ajl0404[workspaces] jenkins-demo:docker-compose up
time="2024-05-20T10:28:12.10+02:00" level=warning msg="C:\Users\ajl0404[workspaces]\jenkins-demo\docker-compose.yml: 'version' is obsolete"
jenkins-service Pulled
c7f2de8a60 Pull complete
5d4d51c5d0 Pull complete
b1b1b7040c2d Pull complete
dfc6b0d26d Pull complete
7a22ae79e0 Pull complete
f2765221e1b Pull complete
2b2a037c63 Pull complete
8c86c355009 Pull complete
2258957c25 Pull complete
8015757646 Pull complete
92226-f6d834 Pull complete
f6d8019f6f Pull complete
Running Jenkins
Network jenkins-demo_default
Volume "jenkins-demo_jenkins_home"
Container jenkins
Attaching to jenkins
jenkins | Running from: /usr/share/docker/jenkins.war

```

[illegible]

La clé secret est générée dans le log.

Configuration

Url: <http://localhost:8080>

Démarrage

Débloquer Jenkins

Pour être sûr que Jenkins soit configuré de façon sécurisée par un administrateur, un mot de passe a été généré dans le fichier de logs ([où le trouver](#)) ainsi que dans ce fichier sur le serveur :

```
/var/jenkins_home/secrets/initialAdminPassword
```

Veuillez copier le mot de passe depuis un des 2 endroits et le coller ci-dessous.

Mot de passe administrateur

1- Copier le code secret depuis le log de Jenkins

Démarrage

Personnaliser Jenkins

Les plugins étendent Jenkins avec des fonctionnalités additionnelles pour satisfaire différents besoins.

Installer les plugins suggérés

Installer les plugins que la communauté Jenkins trouve les plus utiles.

Sélectionner les plugins à installer

Sélectionner et installer les plugins les plus utiles à vos besoins.

2- Installer les plugins par défaut

Créer le 1er utilisateur Administrateur

Nom d'utilisateur

admin

Mot de passe

admin

Confirmation du mot de passe

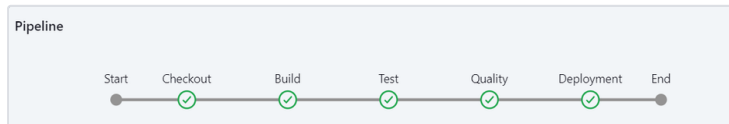
admin

Nom complet

admin@reel

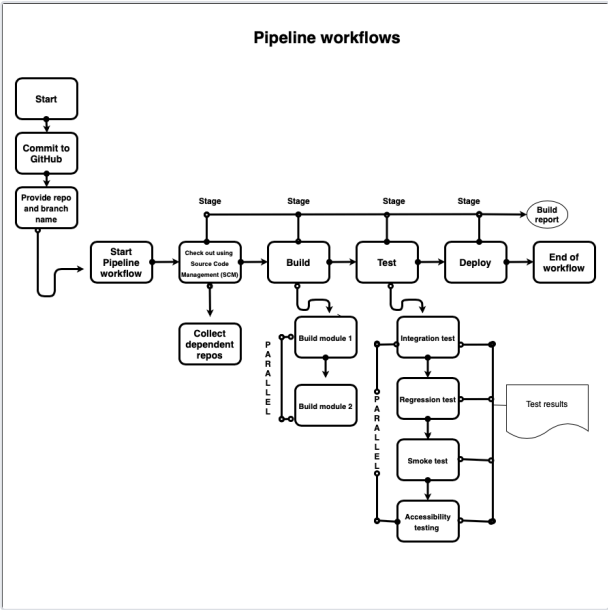
3- Créer l'administrateur de Jenkins

Pipeline

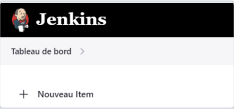


- Un pipeline est un ensemble de `steps` / `stages` exécutées dans le processus d'intégration et de déploiement continu (CI/CD).
- Un `pipeline` comprend en générale les `stages` suivants:
 - `Checkout`
 - `Build`
 - `Test`
 - `Quality`
 - `Deployment`
 - `nexus`
 - `docker`

Pipeline workflow



Création simple Pipeline job



1- Cliquer sur Nouveau Item

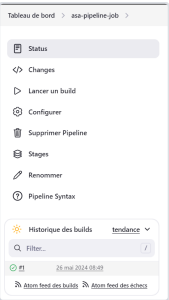


2- Saisir le nom du job simple-pipeline-job et cliquer sur pipeline



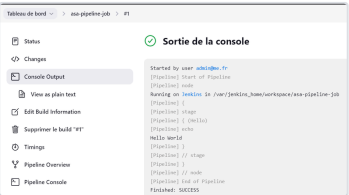
3- Dans définition sélectionner Pipeline script

4- Sélectionner le script Hello World

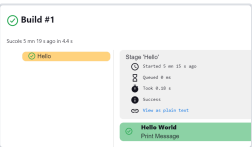


5- Lancer le build

6- Cliquer sur le build #1



7- Cliquer sur console output



8- Cliquer sur Pipeline console

Jenkins File [↗](#)

- Un fichier Jenkins est un fichier texte dans lequel nous définissons notre pipeline sous forme de code.
- Le fichier Jenkins existe dans le référentiel de projet.
- Il existe deux types de fichiers Jenkins :
 - Pipeline déclaratif
 - Pipeline scripté

Un exemple de pipeline déclaratif Jenkins File

```
1 pipeline {
2   agent any
3   stages {
4     stage('Checkout') {
5       steps {
6         echo 'Checkout...'
7       }
8     }
9     stage('Build') {
10      steps {
11        echo 'Building...'
12      }
13    }
14    stage('Test') {
15      steps {
16        echo 'Testing...'
17      }
18    }
19    stage('Quality') {
20      steps {
21        echo 'Quality...'
22      }
23    }
24    stage('Deployment') {
25      steps {
26        echo 'Deploying ...'
27      }
28    }
29  }
30 }
```

- Pipeline : mot-clé, par lequel chaque fichier Jenkins doit commencer.
- agent : Exemples d'agents peuvent être maven , docker , etc.

- stages : Un ensemble de stage
- stage : une tâche du flux de travail.
- steps : Un ensemble d'instructions à exécuter cadre de la tâche.

Création Simple Pipeline Jenkins file [↗](#)

Nouveau Item

Saisissez un nom

new-pipeline-with-jenkinsfile

Select an item type

 Construire un projet free-style

Job legacy polyvalent qui récupère l'état depuis un outil de gestion de version au plus, exécute les étapes de build en série, suit d'étapes post-construction telles que l'archivage d'artefacts et l'envoi de notifications par e-mail.

 Pipeline

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

☐ Do not allow the pipeline to access if the controller exists

 GitHub project

Project ID

https://github.com/jean-101/jean-101-test

Cancel

Save

☒ GitHub hook trigger for GITScm polling ?

☐ Scrutation de l'outil de gestion de version ?

Pipeline

Definition

Pipeline script from SCM

- 1- Cliquer sur nouveau item
- 2- Saisir le nom du job simple-pipeline-jenkinsfile et sélectionner Pipeline

- 3- Cocher Github project
- 4- Copier l'url du projet dans Github

- 5- Sélectionner Github hook trigger for GITScm poling

- 6- Sélectionner Pipeline script from SCM

- 7- Sélectionner Git
- 8- Saisir l'url du projet dans Github

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/asa-7670/asa-jenkins-demo

Credentials ?

- aucun -

+ Ajouter +

Avancé ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/main

Add Branch

Navigateur de la base de code ?

(Auto) ▾

Additional Behaviours

Ajouter ▾

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

✔ < **Build #2**

Succès 12 s ago in 3.1 s

- ✔ Checkout SCM
- ✔ Build
- ✔ Test
- ✔ Quality
- ✔ Deployment
- ✔ Docker build image
- ✔ Docker push image

Stage 'Docker push image'

Started 9.7 s ago

Queued 0 ms

Took 64 ms

Success

View as plain text

✔ Docker push image ...

Print Message

0 Docker push image ...

9- Corriger la branche `master` par la branche `main`

10- Cliquer sur le bouton `Sauvegarder`

11- Lancer le `build`

12- Cliquer sur le `Build #1`

13- Cliquer sur le lien `pipeline-console`

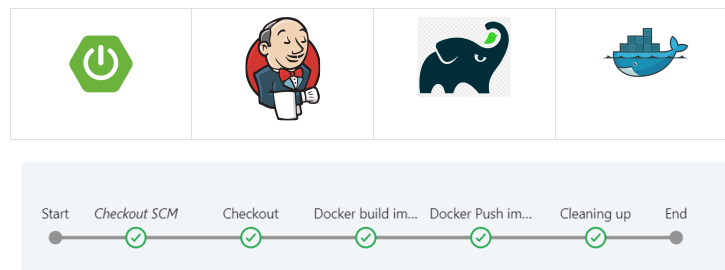
Les `stages` du pipeline sont indiqués.

Arrêt de Jenkins

`docker-compose down`

```
C:\Users\la894884\cd workspaces\asa-dev-workspaces\jenkins-demo
C:\Users\la894884\workspaces\asa-dev-workspaces\jenkins-demo>docker-compose down
time="2024-05-28T11:01:34+02:00" level=warning msg="C:\Users\la894884\workspaces\asa-dev-workspaces\jenkins-demo\docker-compose.yml: 'version' is obsolete"
v1.29.0-rc.2
Container jenkins Removed 0.6s
Network jenkins-demo_default Removed 0.3s
C:\Users\la894884\workspaces\asa-dev-workspaces\jenkins-demo>
```

CI-CD Pipeline for a Spring Boot Application avec Docker et Gradle



Installer Jenkins en local

- Télécharger le fichier `jenkins.war` depuis <https://get.jenkins.io/war-stable/2.452.1/jenkins.war>
- Copier le fichier téléchargé dans `/jenkins/jenkins.war`
- Créer le fichier `/jenkins/run.sh` pour linux ou `/jenkins/run.bat` pour Windows

```
1 java -jar ./jenkins.war
```

- Installer `docker plugin` comme indiqué dans **Tools**.
- Créer le crédentiale `cred_docker_hub_id` comme indiqué dans **Tools**.

Créer le projet `jenkins-pipeline-docker-demo` [↗](#)

- Créer le fichier `jenkins-pipeline-docker-demo/index.html`

```

1 <!doctype html>
2 <html lang="en-US">
3   <head>
4     <meta charset="utf-8" />
5     <meta name="viewport" content="width=device-width" />
6     <title>My test page</title>
7   </head>
8   <body>
9     <h1>Hello form Jenkins pipeline docker !</h1>
10  </body>
11 </html>

```

- Créer le fichier `jenkins-docker-demo/Dockerfile`

```

1 FROM nginx
2 COPY ./index.html /usr/share/nginx/html
3 EXPOSE 80

```

- Créer le fichier `jenkins-pipeline-docker-demo/Jenkinsfile`

```

1 pipeline{
2   environment {
3     registry = "<YOUR_ACCOUNT_DOCKER_HUB>/jenkins-pipeline-doker-demo"
4     registryCredential = 'cred_docker_hub_id'
5     dockerImage = ''
6   }
7   agent any
8   stages{
9
10    stage('Checkout'){
11      steps{
12        git branch: 'main', url: 'https://github.com/asa-7670/jenkins-pipeline-docker-demo'
13      }
14    }
15    stage('Docker build image'){
16      steps{
17        script {
18          dockerImage = docker.build registry + ":$BUILD_NUMBER"
19        }
20      }
21    }
22    stage('Docker Push image'){
23      steps{
24        script {
25          docker.withRegistry( '', registryCredential ) {
26            dockerImage.push()
27          }
28        }
29      }
30    }
31    stage('Cleaning up') {
32      steps {
33        sh 'docker rmi $registry:$BUILD_NUMBER'
34      }
35    }
36  }
37 }

```

- Initier le projet avec `Git`

```

1 cd jenkins-pipeline-docker-demo
2 git init

```

- Partager le projet avec `GitHub`

```

1 git add .
2 git commit -m "first commit"
3 git push origine jenkins-docker-demo

```

- Créer et configurer le job `jenkins-pipeline-docker-demo-job`



Installations IDX

Agente IDX

IDX

Form

openjdk-17

+

Install automatically 3

OpenJDK installer

OpenJDK package

java-17-openjdk

▼

Agente on installation ▼

Installations Maven

Installations Maven < Edit

Apache Maven

id: Maven

name

Maven


☒ Install automatically

Install From Apache

version

3.87

Apache on installation



Install Docker

Agree to Docker

1/1 Docker

Name

Docker

Install automatically

2/1 Download from docker.com

Docker version

latest

Agree to installation

- 1- Cliquer sur `Tableau de bord` → `Administration Jenkins` → `Plugins`
- 2- Rechercher `openJdk`
- 3- Cliquer sur bouton `installer`

- 4- Cliquer sur **Tableau de bord** → **Administration Jenkins** → **Tools**
- 5- définir le nom du JDK `opexjdk-21` pour la version 21
- 6- Cocher la case **Install automatically**
- 7- Cliquer sur le bouton **Enregistrer**

- 4- Cliquer sur **Tableau de bord** → **Administration Jenkins** → **Tools**
- 5- Cliquer sur le bouton **Ajouter Maven**
- 6- Définir le nom du Maven **Maven-3**
- 7- Cocher la case **Install automatically**
- 8- Sélectionner la version à installer
- 9- Cliquer sur le bouton **Enregistrer**

- 1- Cliquer sur **Tableau de bord** → **Administration Jenkins** → **Tools**
- 2- Cliquer sur le bouton **Ajouter Gradle**
- 3- Définir le nom du Docker **Docker-8**
- 4- Cocher la case **Install automatically**
- 5- Sélectionner la version à installer
- 6- Cliquer sur le bouton **Enregistrer**

- 1- Cliquer sur `Tableau de bord` → `Administration Jenkins` → `Plugins`
- 2- Rechercher `openJdk`
- 3- Cliquer sur bouton `installer`

- 1- Cliquer sur **Tabl eau de bord** → **Administration Jenkins** → **Tools**
- 2- Cliquer sur le bouton **Ajouter Docker**
- 3- Définir le nom du Docker **Docker**
- 4- Cocher la case **Install automatically**
- 5- Sélectionner la version à installer **latest**
- 6- Cliquer sur le bouton **Enregistrer**

Talhou de bord > Administrar credenciais > Identificantes > System > Identificantes globais (limitado)

New credentials

Type

Nome d'utilizador e mot de passe

Partes

Global (permissões, agentes, items, etc.)

Nome d'utilizador

USER.ACCOUNT.DOCKER.HUB

☐ Tratar como nome de segredo

Mot de passe

ID

CRED.DOCKER.HUB.ID

Descrição

Docker hub credentials user/password

Gravar