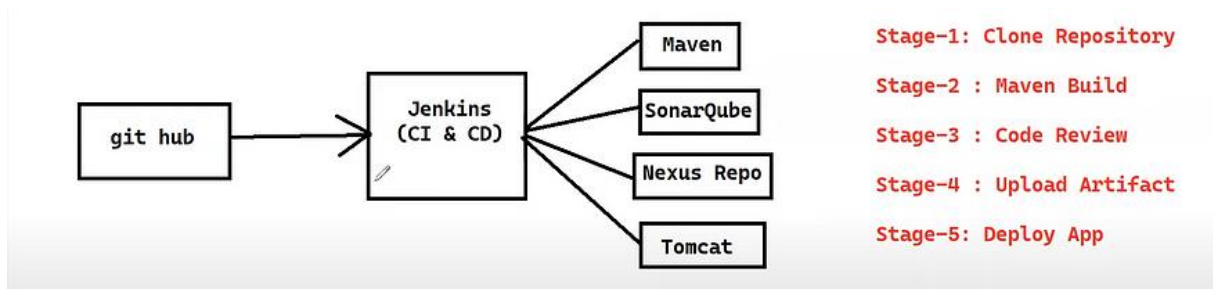


# Configuration du pipeline DevOps CICD via { Jenkins | Maven | Sonarqube | Nexus | Tomcat }

Donc dans le cadre de cette configuration, nous allons d'abord utiliser plusieurs outils, notre projet est disponible sur GitHub où le code source du projet sera disponible. Nous prendrons donc un projet de GitHub et le déploierons en différentes étapes à l'aide de la structure CICD.



Afin de créer et de déployer cette application, nous utilisons **Jenkins**. Chez Jenkins, nous utilisons le concept de pipeline CICD. Ce logiciel Jenkins va donc cloner le dépôt depuis GitHub et communiquera avec **Maven**.

**Maven** : Il s'agit d'un outil de build utilisé pour effectuer le processus de build des applications Java.

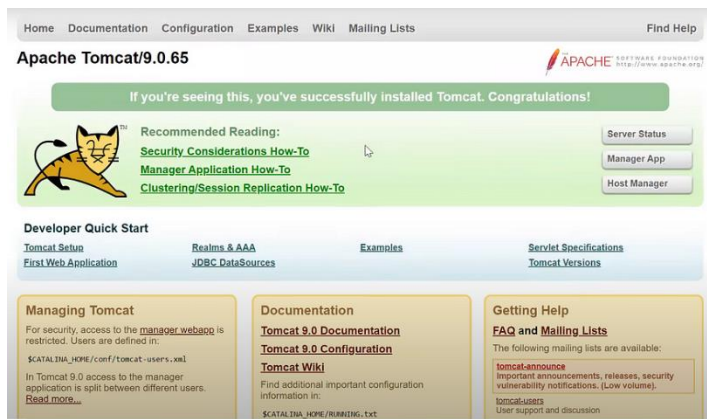
Maintenant, Jenkins va cloner le référentiel depuis le github et maven va compiler et empaqueter notre application Java. Une fois le processus de construction terminé, Jenkins devrait pouvoir communiquer avec le Sonarqube.

**SonarQube** : Pour effectuer la révision du code, Sonarqube est un logiciel de vérification de la qualité du code dans le pipeline.

Après avoir effectué la révision du code de notre application, nous souhaitons créer un artefact dans le référentiel Nexus.

**Nexus** : Nexus est le référentiel où sont placés tous les artefacts.

Après avoir téléchargé l'artefact dans le référentiel Nexus, nous souhaitons déployer le fichier war de l'application sur le serveur Apache Tomcat.

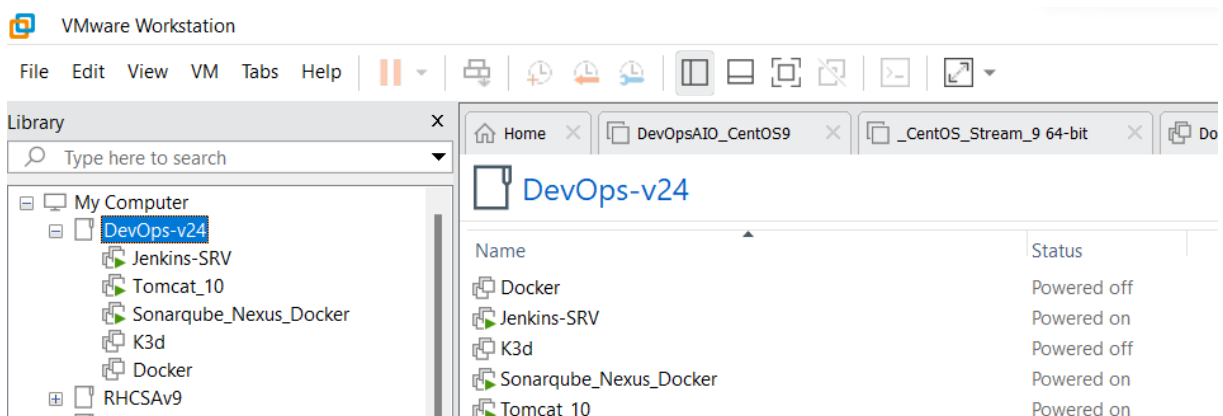


**Serveur Apache Tomcat :** est un conteneur utilisé pour exécuter nos applications Web.

Nous utiliserons le serveur Tomcat pour exécuter notre application Java. Jenkins est chargé de communiquer avec tous les outils pour automatiser le processus de création et de déploiement d'applications. Afin d'effectuer toutes ces étapes, nous allons créer un pipeline dans Jenkins.

Après cela, nous créerons AWS Linux Machine, vous pourrez utiliser n'importe quelle machine dans votre zone de confort.

Nous allons créer 4 serveurs comme décrit ci-dessous :



Pour ce projet, nous utiliserons Linux Centos pour l'installation du serveur Tomcat.

## Pour l'installation du serveur Tomcat

### a) Installer OpenJDK

```
[root@dlp ~]# dnf -y install java-17-openjdk java-17-openjdk-devel
[root@dlp ~]# cat > /etc/profile.d/java.sh <<'EOF'
export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which
java))))))
export PATH=$PATH:$JAVA_HOME/bin
EOF
[root@dlp ~]# source /etc/profile.d/java.sh
[root@dlp ~]# java --version

openjdk 17.0.3 2022-04-19 LTS
OpenJDK Runtime Environment 21.9 (build 17.0.3+7-LTS)
OpenJDK 64-Bit Server VM 21.9 (build 17.0.3+7-LTS, mixed mode, sharing)
```

# verify to create test program

```
[root@dlp ~]# cat > java_test.java <<'EOF'
class java_test {
    public static void main(String[] args) {
        System.out.println("Hello Java World !");
    }
}
EOF
```

```
[root@dlp ~]# javac java_test.java

[root@dlp ~]# java java_test
```

Hello Java World !

### b) Install Tomcat 10.

```
# yum install tomcat tomcat-admin-webapps
```

Modifier le fichier comme /usr/share/tomcat/webapps/manager/META-INF/context.xml suit :

```
# nano /usr/share/tomcat/webapps/manager/META-INF/context.xml
...

<!--
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="127\.\d+\.\d+\.\d+|:::1|0:0:0:0:0:0:0:1" />
-->
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
    allow="^.*$" />
  <Manager
sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|N
umber|String)|org\.apache\.catalina\.filters\.CsrfPreventionFilter\$LruCach
e(?:\$1)?|java\.util\.(?:Linke>
</Context>
```

Modifier le fichier comme /usr/share/tomcat/conf/tomcat-users.xml suit :

```
# nano /usr/share/tomcat/conf/tomcat-users.xml

<role rolename="admin"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<role rolename="manager"/>
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user name="admin" password="password" roles="admin,manager,admin-
gui,admin-script,manager-gui,manager-script,manager-jmx,manager-status" />

# systemctl enable -now tomcat.service
```

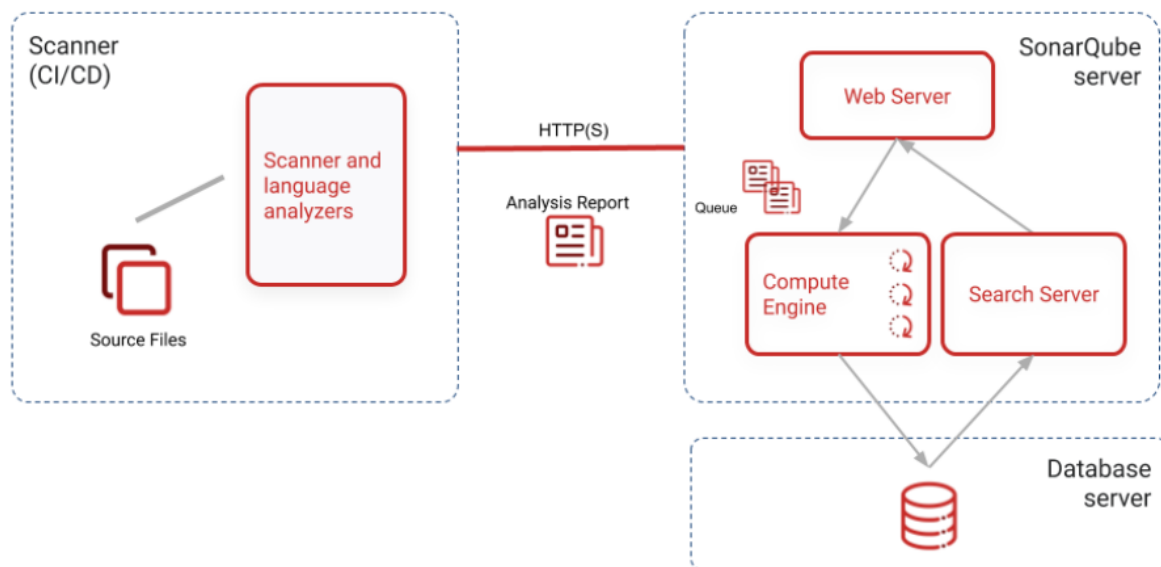
## Pour l'installation de SonarQube

SonarQube est un outil d'inspection continue qui peut être utilisé pour tester la qualité du code. Il analyse le code source et nous envoie le rapport analytique pour vérifier la qualité finale.

Il est disponible en tant que plate-forme open source et prend en charge plusieurs langages de programmation tels que Java, Python, Javascript, TypeScript, COBOL, HTML, XML, C#, C/C++ Apex, Object-C, Swift, Kotlin, Ruby, Scala, CSS, ABAP, etc. Cependant, certaines langues nécessitent une licence commerciale pour fonctionner.

Vous n'avez pas besoin de modifier votre flux de travail ou d'apprendre de nouveaux outils pour installer SonarQube, car il peut facilement être intégré à des outils de construction courants tels que Ant, Maven, Make, Gradle, MS Build, etc. Tout ce que vous avez à faire est d'ajouter les plugins appropriés pour une analyse plus fluide dans votre processus de construction.

Une instance unique de SonarQube comprend trois composants :



1. Le serveur SonarQube exécute les processus suivants :
  - Un serveur Web qui dessert l'interface utilisateur de SonarQube.
  - Un serveur de recherche basé sur Elasticsearch.
  - Le moteur de calcul chargé de traiter les rapports d'analyse de code et de les enregistrer dans la base de données SonarQube.
2. La base de données pour stocker les éléments suivants :
  - Métriques et problèmes de qualité et de sécurité du code générés lors des analyses de code.
  - La configuration de l'instance SonarQube.
3. Un ou plusieurs scanners exécutés sur vos serveurs de build ou d'intégration continue pour analyser les projets.

Suivez ces étapes pour l'installer via docker compose :

a) Créer le fichier docker-compose.yml suivant dans un répertoire dédié

```
version: "3"

services:
  sonarqube:
    image: sonarqube:community
    depends_on:
      - db
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://db:5432/sonar
      SONAR_JDBC_USERNAME: sonar
      SONAR_JDBC_PASSWORD: password
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
    ports:
      - "9000:9000"
    restart: always
  db:
    image: postgres:12
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: password
    volumes:
      - postgresql:/var/lib/postgresql
      - postgresql_data:/var/lib/postgresql/data
    restart: always

volumes:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
  postgresql:
  postgresql_data:
```

b) Il faut configurer Linux avant de lancer le template en appliquant le paramètre suivant

```
root# sysctl -w vm.max_map_count=262144
root# echo "vm.max_map_count=262144" > /etc/sysctl.d/98-vm.max_map_count.conf
```

c) Exécutez maintenant le fichier de composition à l'aide de la commande Docker compose :

```
$ docker compose up -d
```

d) Répertoriez les conteneurs en cours d'exécution pour voir si les conteneurs SonarQube et PostgreSQL sont présents.

```
$ docker compose ps
```

e) Ouvrez maintenant dans votre navigateur <http://localhost:9000> et connectez-vous à votre compte par défaut **admin** dont le mot de passe est **admin**.

## Pour une installation rapide de Nexus

- a) Tout d'abord, nous commencerons par créer un volume pour conserver les données persistantes au cas où le conteneur s'arrêterait pour une raison quelconque.

```
$ docker volume create --name nexus-data
```

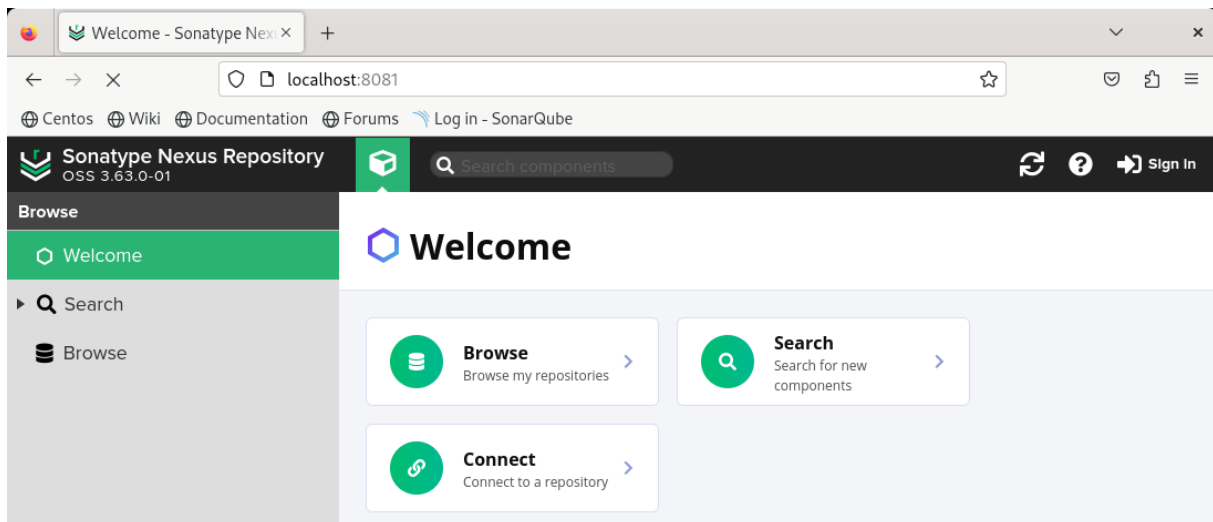
- b) Ensuite, nous allons exécuter le conteneur avec l'image [sonatype/nexus3](https://sonatype.github.io/nexus3)

```
$ docker run -d -p 8081:8081 --name nexus -v nexus-data:/nexus-data sonatype/nexus3
```

Par défaut, le gestionnaire de référentiel Nexus s'exécute sur le port 8081

Maintenant, nous allons lui donner une minute ou deux pour s'initialiser

Maintenant, en visitant l'url <http://localhost:8081>, nous pouvons voir que Nexus s'est initialisé



- c) La prochaine étape que nous devons faire est d'obtenir le mot de passe administrateur qui est généré automatiquement lors du lancement initial du conteneur.

Comme indiqué dans la documentation, l'utilisateur par défaut est admin et le mot de passe généré de manière unique se trouve dans le fichier `admin.password` à l'intérieur du volume.

### Obtenir le mot de passe administrateur

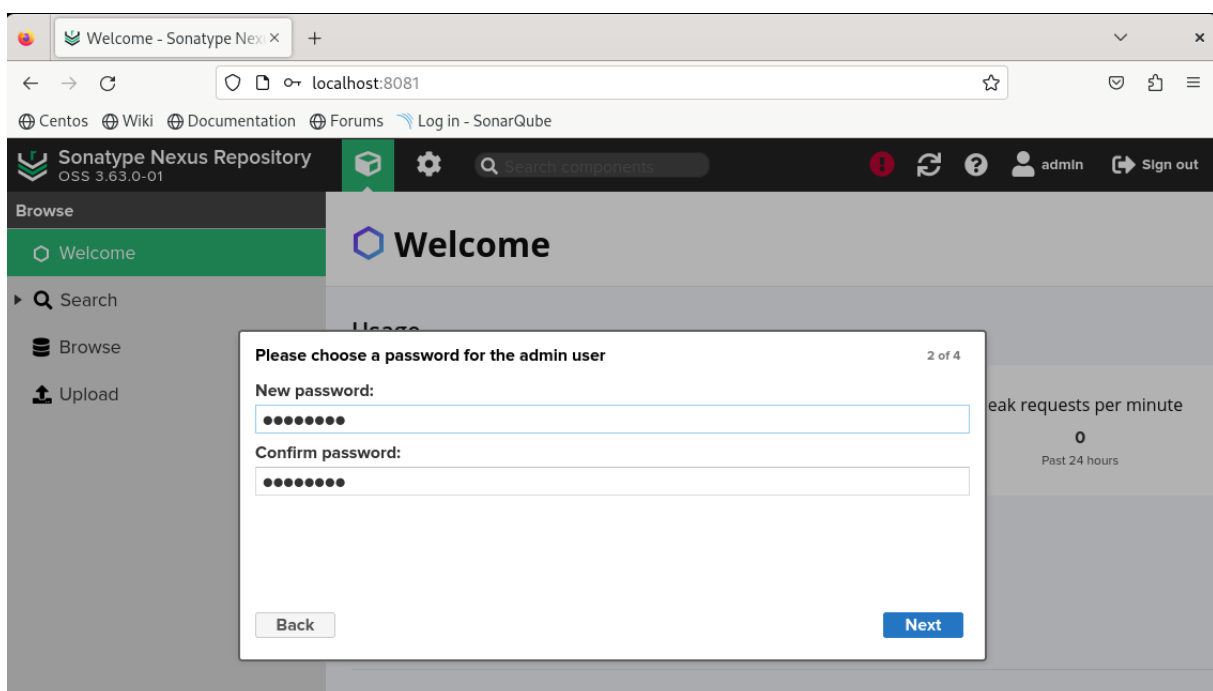
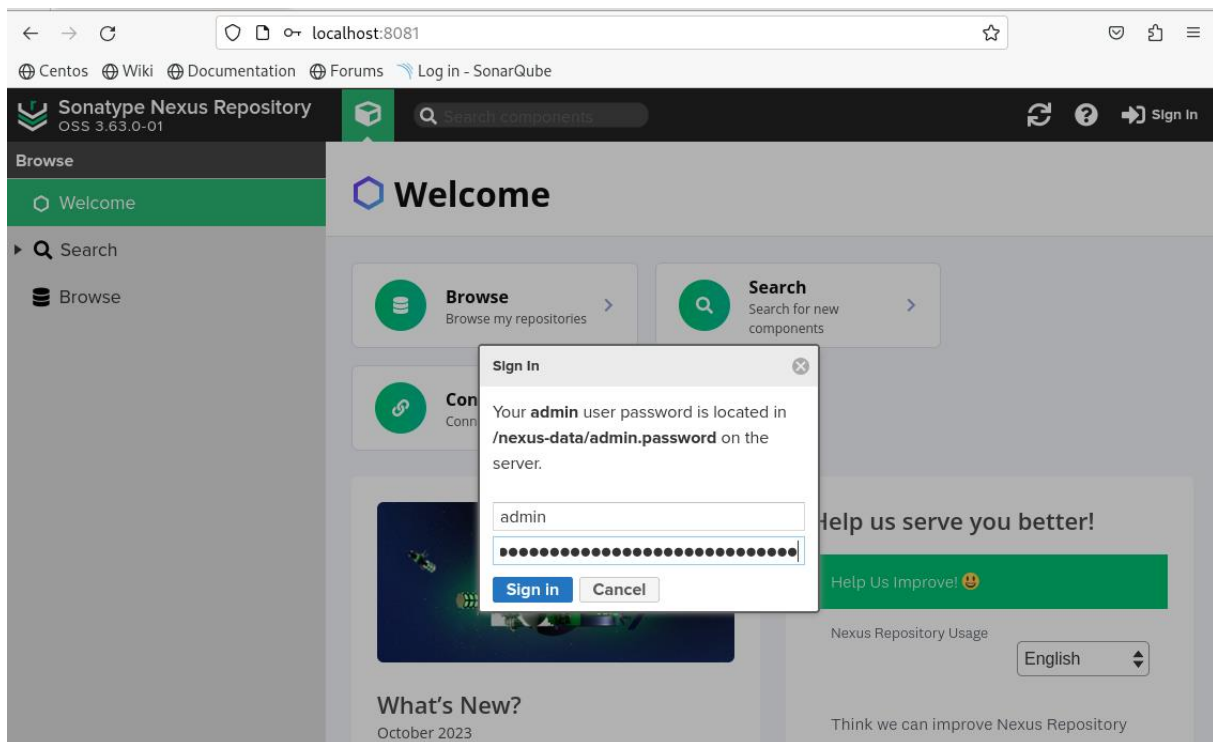
Nous transmettons le mot de passe à notre console en utilisant `docker exec`

```
$ docker container exec nexus cat /nexus-data/admin.password
```

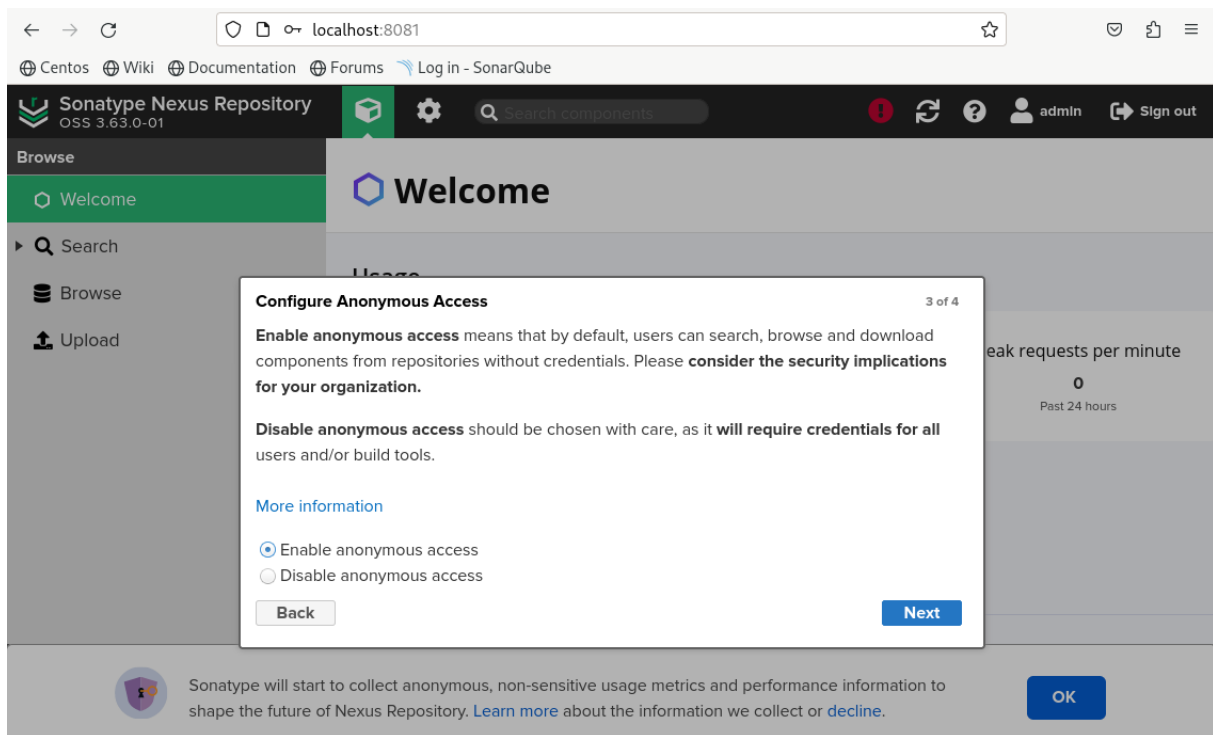
ce qui devrait renvoyer quelque chose comme ce qui suit

```
[admuser@server sonarqube]$ docker container exec nexus cat /nexus-data/admin.password
40cbba1e-e29e-42c7-9cb6-e589b059e4c8 [admuser@server sonarqube]$
```

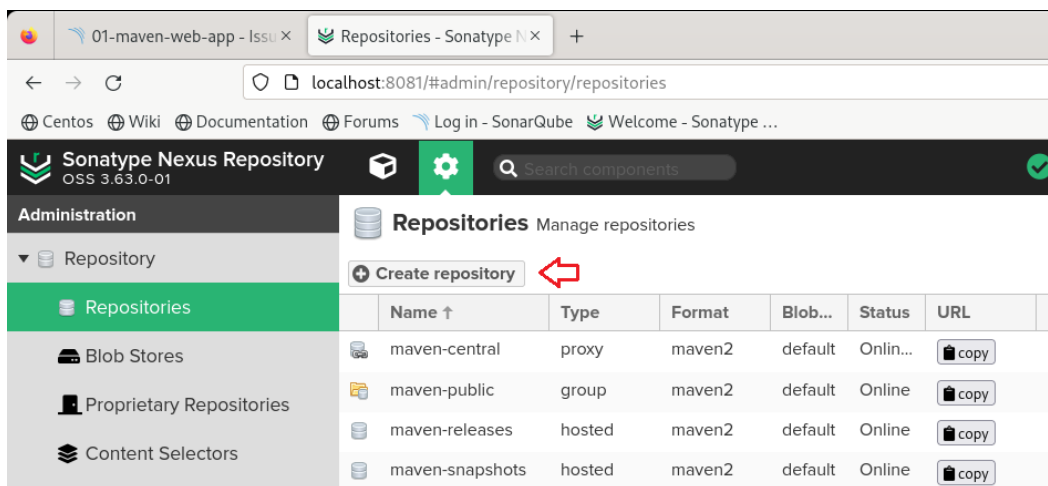
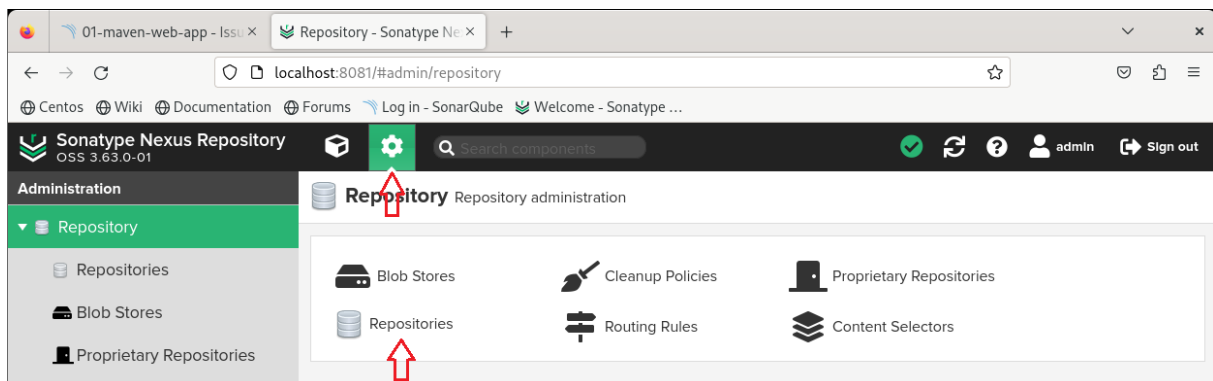
Nous pouvons maintenant nous connecter au panneau d'administration de Nexus

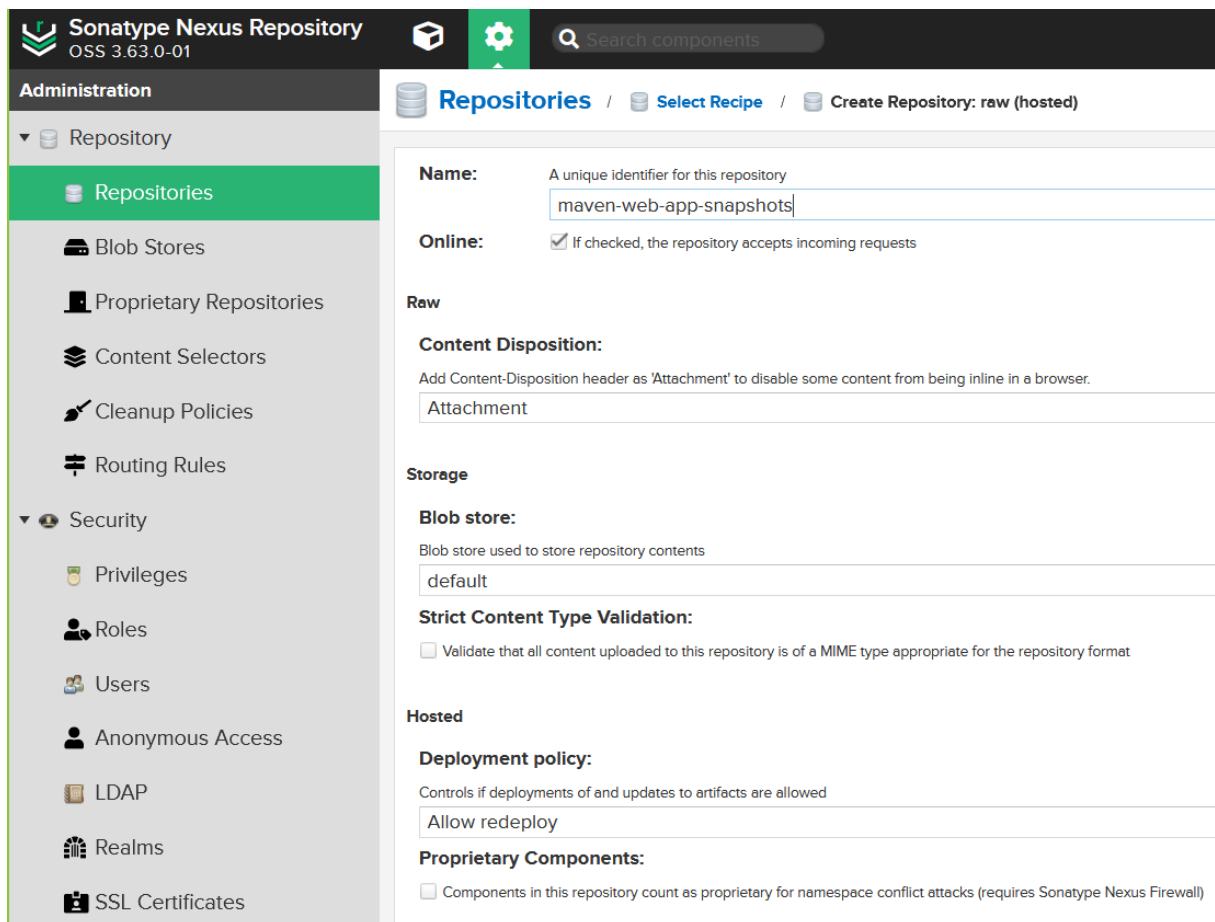
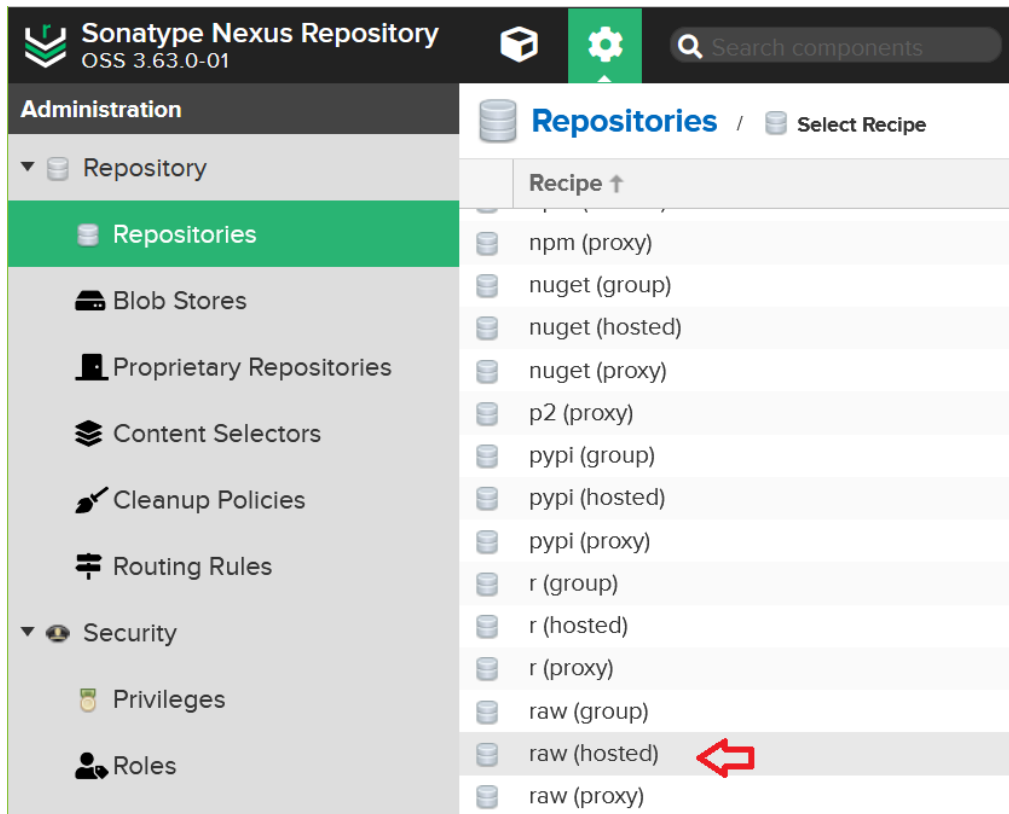






et à partir de là, nous pouvons configurer Nexus et créer différents référentiels en fonction des besoins de notre application.



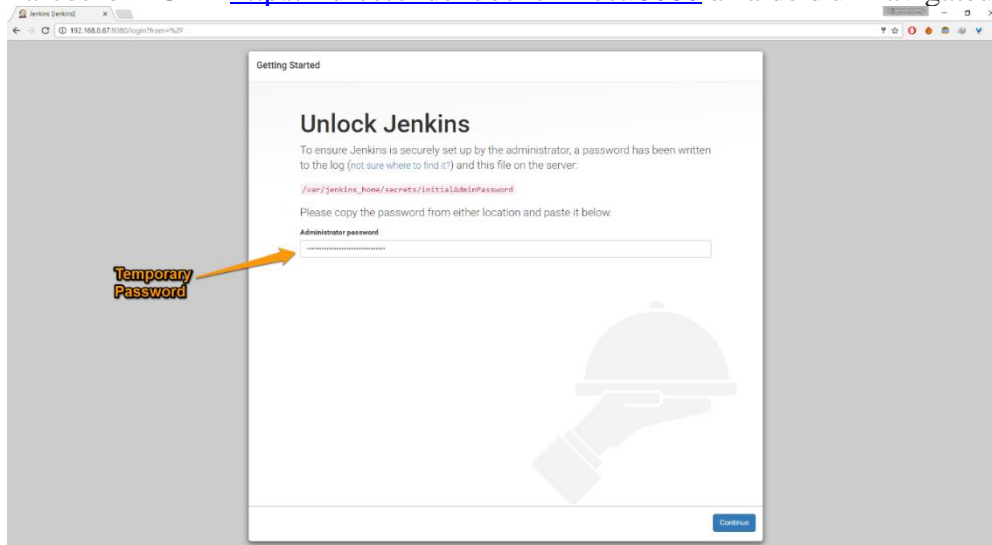


## Pour l'installation de Jenkins

```
$ sudo -i
# wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
# rpm --import https://pkg.jenkins.io/redhat-
stable/jenkins.io.key
# yum install java-11-openjdk java-11-openjdk-devel
# cat > /etc/profile.d/java.sh <<'EOF'
export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink
$(which java))))))
export PATH=$PATH:$JAVA_HOME/bin
EOF
# source /etc/profile.d/java.sh
# yum install jenkins
# systemctl enable --now jenkins
# hostnamectl set-hostname jenkins-server.lab.local
# IP=$(hostname -I | awk '{print $1}') eval 'echo "$IP
jenkins-server jenkins-server.lab.local" >> /etc/hosts'
```

## Accès à l'interface utilisateur Web de Jenkins:

Parcourez l'URL <http://Adresse de docker host:8080> à l'aide d'un navigateur.



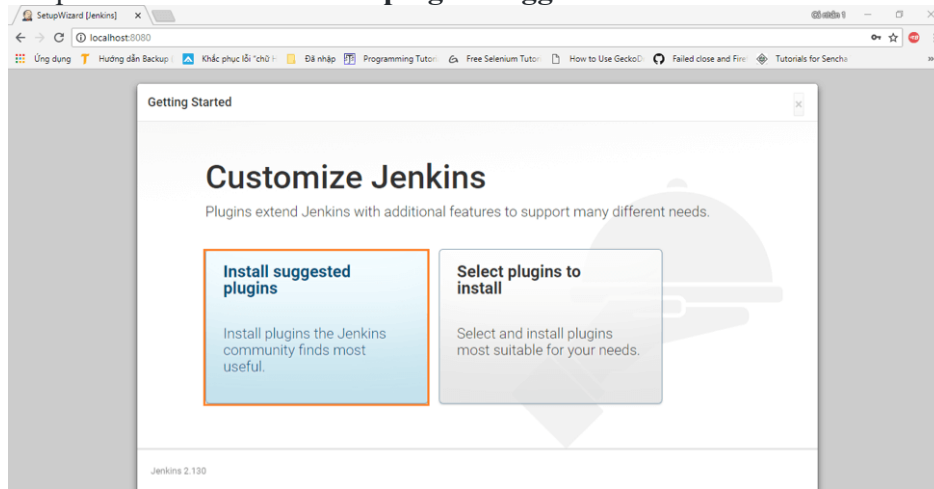
- Puisque, nous accédons à l'interface Web de Jenkins pour la première fois, nous avons besoin du mot de passe administrateur pour nous y connecter.

Le chemin d'accès au fichier de mot de passe a été fourni par l'interface Web Jenkins.

```
# cat /var/lib/jenkins/secrets/initialAdminPassword
```

Connectez-vous à l'interface Web de Jenkins en utilisant ce mot de passe.

- Cliquez sur **Sélectionner les plugins suggérés**.



Créer un utilisateur administrateur

Getting Started

## Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Getting Started

## Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

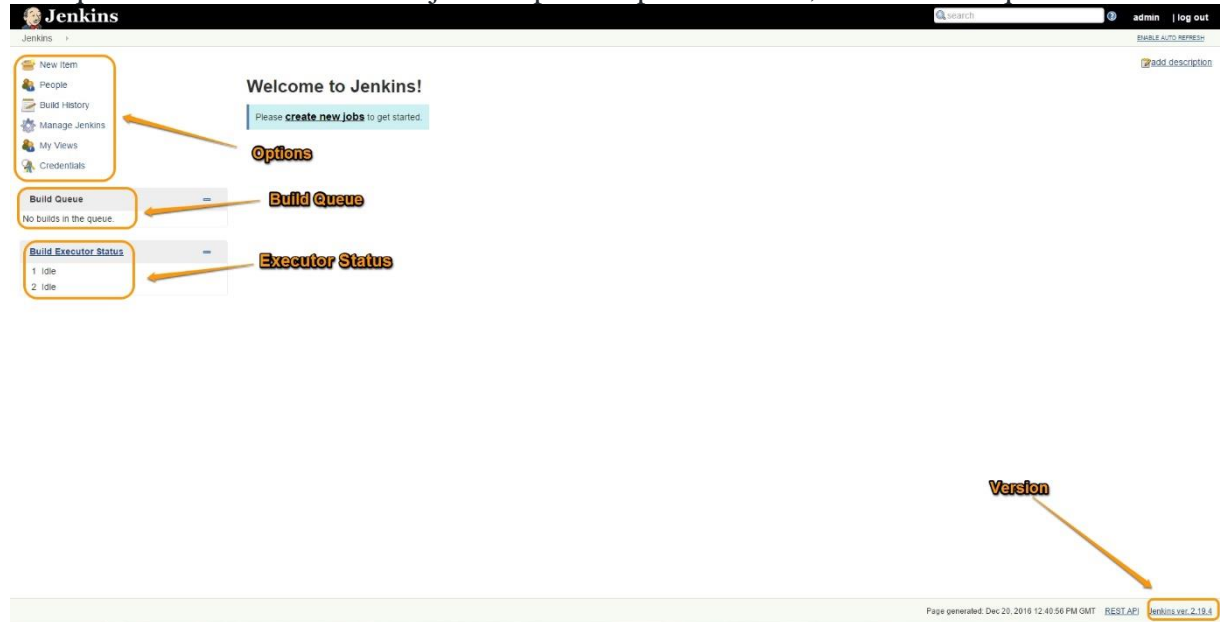
Jenkins 2.426.1 Not now Save and Finish

Maintenant, nous avons installé Jenkins avec succès et nous pouvons procéder aux configurations

## Configurations Jenkins

### Se familiariser avec la console Jenkins

Lorsque vous vous connectez à Jenkins pour la première fois, voici l'écran que vous verriez.



- Sur le côté gauche de l'écran, en haut se trouve le menu pour créer de nouveaux projets, pour gérer Jenkins, pour créer des utilisateurs, etc.
- Juste en dessous du menu se trouve la file d'attente de construction. Tous les travaux programmés pour s'exécuter sont ajoutés à la file d'attente et apparaîtront ici.
- Sous la file d'attente de construction se trouve le statut de l'exécuteur de compilation. Cela montre l'état des travaux en cours d'exécution en temps réel.
- En bas à droite de la page se trouvent les informations sur la version de Jenkins affichées.

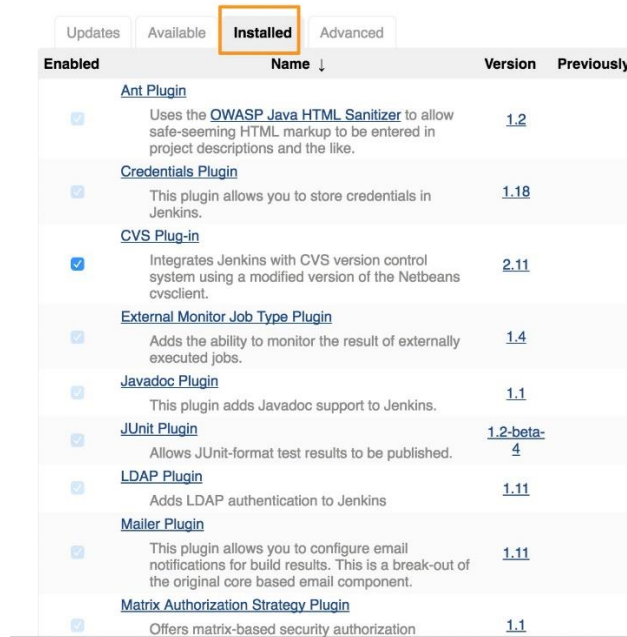
## Plugins

La vraie force de Jenkins réside dans son écosystème riche en plugins. C'est ainsi que les outils s'intègrent à Jenkins pour créer un flux de travail CI. Vous voulez déclencher des jobs Jenkins après chaque changement dans Git, vous avez un plugin pour cela. Vous souhaitez envoyer une notification à vos développeurs sur une compilation réussie ou échouée, vous disposez d'un plugin de notification. Vous souhaitez utiliser un outil pour récupérer ou pousser les artefacts de construction, vous avez un plugin pour cela. C'est ainsi que la plupart des outils parlent à Jenkins.

Dans ce Lab, nous allons voir un processus simple pour installer des plugins. Dans ce cadre, nous allons installer un plugin qui nous aidera à intégrer Jenkins à notre référentiel Git.

## Explorer les configurations de plugins

- Dans «Administrer Jenkins», sélectionnez l'option «Gestion des plugins».
- Dans le volet Gérer les plugins, vous verrez les onglets suivants,
  - Mises à jour
  - Disponibles
  - Installés



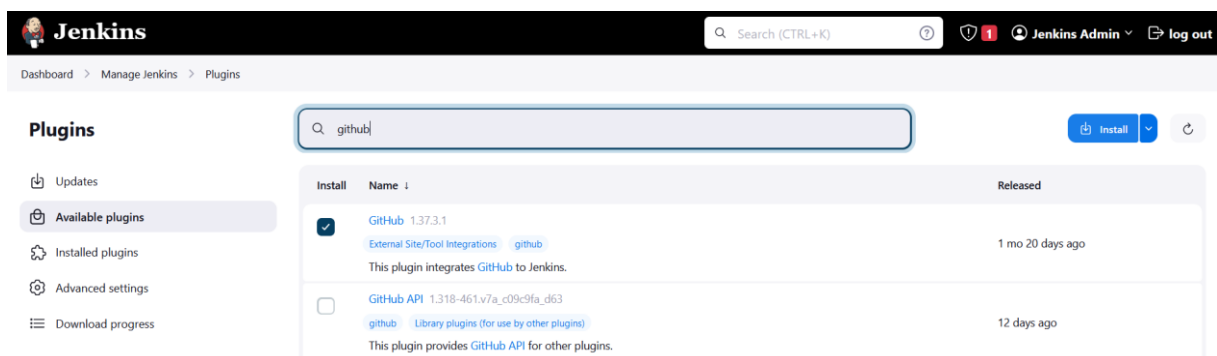
Enabled	Name ↓	Version	Previously
<input checked="" type="checkbox"/>	<a href="#">Ant Plugin</a> Uses the <a href="#">OWASP Java HTML Sanitizer</a> to allow safe-seeming HTML markup to be entered in project descriptions and the like.	1.2	
<input checked="" type="checkbox"/>	<a href="#">Credentials Plugin</a> This plugin allows you to store credentials in Jenkins.	1.18	
<input checked="" type="checkbox"/>	<a href="#">CVS Plug-in</a> Integrates Jenkins with CVS version control system using a modified version of the Netbeans cvsclient.	2.11	
<input checked="" type="checkbox"/>	<a href="#">External Monitor Job Type Plugin</a> Adds the ability to monitor the result of externally executed jobs.	1.4	
<input checked="" type="checkbox"/>	<a href="#">Javadoc Plugin</a> This plugin adds Javadoc support to Jenkins.	1.1	
<input checked="" type="checkbox"/>	<a href="#">JUnit Plugin</a> Allows JUnit-format test results to be published.	1.2-beta-4	
<input checked="" type="checkbox"/>	<a href="#">LDAP Plugin</a> Adds LDAP authentication to Jenkins	1.11	
<input checked="" type="checkbox"/>	<a href="#">Mailer Plugin</a> This plugin allows you to configure email notifications for build results. This is a break-out of the original core based email component.	1.11	
<input checked="" type="checkbox"/>	<a href="#">Matrix Authorization Strategy Plugin</a> Offers matrix-based security authorization	1.1	

- Avancé

Sélectionnez "Installés" pour afficher la liste des plugins préinstallés avec Jenkins.

## Installer le plugin Github

- Dans "Gérer les plugins", sélectionnez l'onglet **Disponibles**.
- Dans le coin supérieur droit, vous devriez voir une zone de filtre, commencez à taper le terme de recherche dans cette zone. Pour cet exemple, nous taperions **github**.



Jenkins

Search (CTRL+K)

Dashboard > Manage Jenkins > Plugins

Plugins

Search: github

Install

Install	Name ↓	Released
<input checked="" type="checkbox"/>	<a href="#">GitHub</a> 1.37.3.1 <a href="#">External Site/Tool Integrations</a> github This plugin integrates <a href="#">GitHub</a> to Jenkins.	1 mo 20 days ago
<input type="checkbox"/>	<a href="#">GitHub API</a> 1.318-461.v7a_c09c9fa_d63 <a href="#">github</a> Library plugins (for use by other plugins) This plugin provides <a href="#">GitHub API</a> for other plugins.	12 days ago

## Installer Git sur le serveur jenkins

```
# yum install git -y
# which git
/usr/bin/git
```

Une fois installé il faut configurer le plugin :

### Dashboard -> Tools

The screenshot shows the Jenkins 'Tools' configuration page. At the top, there's a breadcrumb: 'Dashboard > Manage Jenkins > Tools'. Under 'Git installations', there's a form with fields for 'Name' (set to 'Git') and 'Path to Git executable' (set to '/usr/bin/git'). There's an unchecked checkbox for 'Install automatically'. Below this is an 'Add Git' button. Under 'Maven installations', there's an 'Add Maven' button. At the bottom are 'Save' and 'Apply' buttons.

### Apply & Save

## Configurer le Global Tool JDK

Sur le serveur jenkins :

```
# echo $JAVA_HOME
/usr/lib/jvm/java-17-openjdk-17.0.6.0.10-3.el9.x86_64
```

The screenshot shows the Jenkins 'Tools' configuration page, specifically the 'JDK installations' section. It has a breadcrumb: 'Dashboard > Manage Jenkins > Tools'. There's an 'Add JDK' button. Below it is a form with fields for 'Name' (set to 'JAVA\_HOME') and 'JAVA\_HOME' (set to '/usr/lib/jvm/java-17-openjdk-17.0.6.0.10-3.el9.x86\_64'). There's an unchecked checkbox for 'Install automatically'. Below this is another 'Add JDK' button. At the bottom, there's a 'Git installations' section which is currently empty. A green 'Saved' notification banner is visible at the very bottom.

### Apply & Save

## Le Projet

Nous allons donc décomposer le projet en cinq phases qui sont les suivantes

Étape 1 : Cloner le référentiel

Étape 2 : Build Maven

Étape 3 : Révision du code

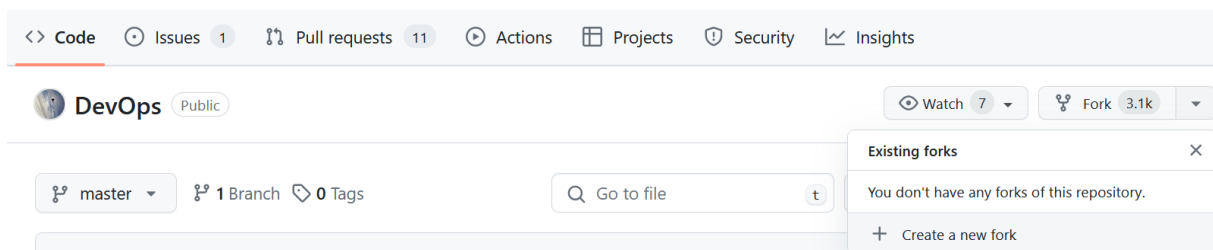
Étape 4 : Télécharger des artefacts

Étape 5 : Déployer l'application


### I. Première étape : cloner le référentiel

Donc, dans la première étape de ce pipeline, nous allons cloner le référentiel git <https://github.com/eliesjebri/maven-web-app.git> depuis votre compte Github.

Depuis votre compte Github allez à « Create a new fork »



Puis créez le fork


Owner \*  eliesjebri / Repository name \*

✓ Your new repository will be created as DevOps-.  
The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the **master** branch only  
Contribute back to ravdy/DevOps by adding your own branch. [Learn more.](#)

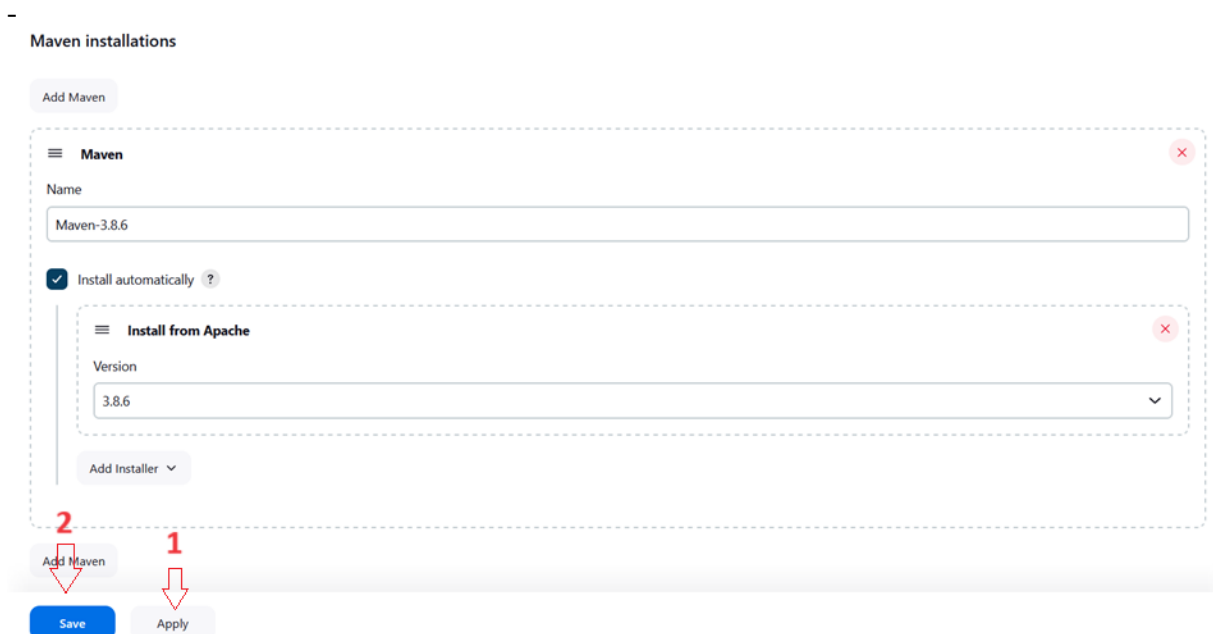
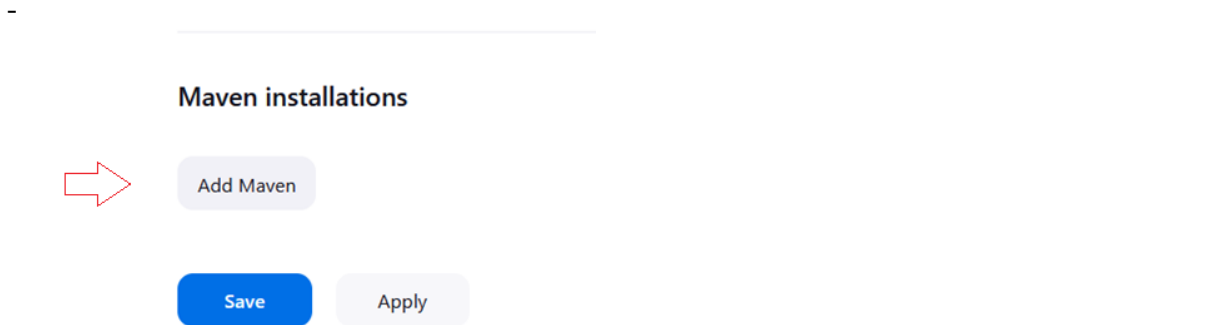
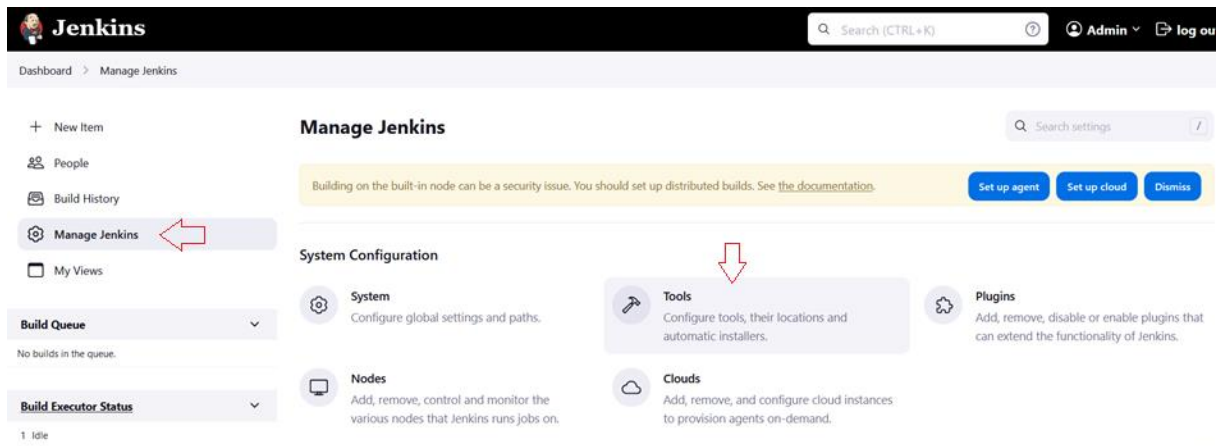
 You are creating a fork in your personal account.

[Create fork](#)

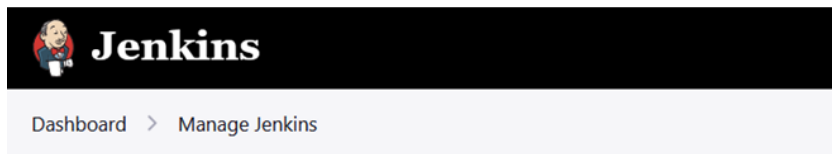


a) Commençons maintenant à configurer Jenkins :

Tout d'abord, nous devons configurer maven en tant que configuration globale d'outil dans le tableau de bord Jenkins.



b) Commençons maintenant à construire le pipeline :

This is the 'Enter an item name' form in Jenkins. It has a text input field containing 'maven-web-app-pipeline', with a red arrow pointing to it. Below the field is a note '» Required field'. Underneath, there are two options: 'Freestyle project' and 'Pipeline'. The 'Pipeline' option is selected, indicated by a red arrow. The 'Pipeline' description states: '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.'This is the Jenkins Pipeline configuration page. The breadcrumb trail is 'Dashboard > maven-web-app-pipeline > Configuration'. On the left, under 'Configure', the 'Pipeline' tab is selected, with a red arrow pointing to it. The main area is titled 'Pipeline' and shows the 'Definition' dropdown set to 'Pipeline script'. Below this is a 'Script' editor with a red arrow pointing to a dropdown menu that lists 'try sample Pipeline...', 'try sample Pipeline...', 'Hello World', 'GitHub + Maven', and 'Scripted Pipeline'. At the bottom, there's a checkbox for 'Use Groovy Sandbox' which is checked, and a 'Pipeline Syntax' link. 'Save' and 'Apply' buttons are at the bottom.

Dashboard > maven-web-app-pipeline > Configuration

### Configure

- General
- Advanced Project Options
- Pipeline**

### Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Hello') {
6       steps {
7         echo 'Hello World'
8       }
9     }
10  }
11 }
12
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

[Save](#) [Apply](#)

Dashboard > maven-web-app-pipeline > Pipeline Syntax

### Snippet Generator

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

git: Git

- archiveArtifacts: Archive the artifacts
- bat: Windows Batch Script
- build: Build a job
- catchError: Catch error and set build result to failure
- checkout: Check out from version control
- cleanWs: Delete workspace when build is done
- deleteDir: Recursively delete the current directory from the workspace
- dir: Change current directory
- echo: Print Message
- emailExt: Extended Email
- emailExtrecipients: Extended Email Recipients
- error: Error signal
- fileExists: Verify if file exists in workspace
- findBuildScans: Find published build scans
- fingerprint: Record fingerprints of files to track usage
- git: Git**
- input: Wait for interactive input

eliesjebri / maven-web-app

Code Pull requests Actions Projects Wiki Security Insights Settings

maven-web-app Public

forked from Shauryan/maven-web-app

Pin Watch 0 Fork 1.1k Star 0

main 1 Branch 0 Tags

This branch is up to date with Shauryan/maven-web-app:master

ashokitschool Update pom.xml

src/main/webapp Update in

Go to file

Local Codespaces

Clone

HTTPS SSH GitHub CLI

Copy url to clipboard

https://github.com/eliesjebri/maven-web-app.git

About

maven-web-app

- Activity
- 0 stars
- 0 watching
- 1.1k forks

## Steps

Sample Step

git: Git

git ?

Repository URL ?

<https://github.com/eliesjebri/maven-web-app.git>

Failed to connect to repository : Error performing git command: ls-remote -h <https://github.com/eliesjebri/maven-web-app.git> HEAD

Branch ?

main

Credentials ?

- none -

+ Add

☒ Include in polling? ?

Poll remote repository for changes. Default is 'true'.

If poll is false, then the remote repository will not be polled for changes. If poll is true or is not set, then the remote repository will be polled for changes.

(from Git plugin)

☒ Include in changelog? ?

Compute changelog for this job. Default is 'true'.

If changelog is false, then the changelog will not be computed for this job. If changelog is true or is not set, then the changelog will be computed.

(from Git plugin)

Generate Pipeline Script

git branch: 'main', url: 'https://github.com/eliesjebri/maven-web-app.git'



Dashboard > maven-web-app-pipeline > Configuration

### Configure

- General
- Advanced Project Options
- Pipeline**

### Pipeline

Definition

Pipeline script

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Clone Repo') {
6       steps {
7         git branch: 'main', url: 'https://github.com/eliesjebri/maven-web-app.git'
8       }
9     }
10  }
11 }
12
```

☒ Use Groovy Sandbox ?

Pipeline Syntax

Save Apply

Maintenant, Appliquez et enregistrez.

- c) Exécutez simplement le pipeline et voyez si la première étape est en cours de traitement ou non. Alors cliquez sur Construire maintenant :

Jenkins

Dashboard > maven-web-app-pipeline >

- Status
- Changes
- Build Now** ←
- Configure
- Delete Pipeline
- Full Stage View
- Rename
- Pipeline Syntax

The screenshot displays the Jenkins dashboard for the 'maven-web-app-pipeline'. The top navigation bar shows 'Dashboard > maven-web-app-pipeline >'. The main content area is divided into two sections: 'Stage View' and 'Permalinks'.

**Stage View:** This section shows the pipeline's progress. A green checkmark indicates the pipeline is successful. The 'Hello' stage is highlighted with a green bar, showing a duration of 1s. The 'Average stage times' are listed as 1s, and the 'Average full run time' is approximately 2s. A table below shows the build history for the 'Hello' stage, with the most recent build (#2) on Dec 20, 2023, at 18:39, showing 'No Changes'.

**Permalinks:** This section provides links to various build-related information:

- Last build (#2), 9 min 57 sec ago
- Last stable build (#2), 9 min 57 sec ago
- Last successful build (#2), 9 min 57 sec ago
- Last completed build (#2), 9 min 57 sec ago

**Console Output:** The console output for build #2 is displayed, showing the pipeline's execution steps. The output is as follows:

```
Started by user Admin
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/maven-web-app-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Hello)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/elsesjebri/maven-web-app.git
> /usr/bin/git init /var/lib/jenkins/workspace/maven-web-app-pipeline # timeout=10
Fetching upstream changes from https://github.com/elsesjebri/maven-web-app.git
> /usr/bin/git --version # timeout=10
> git --version # 'git version 2.39.3'
> /usr/bin/git fetch --tags --force --progress -- https://github.com/elsesjebri/maven-web-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> /usr/bin/git config remote.origin.url https://github.com/elsesjebri/maven-web-app.git # timeout=10
> /usr/bin/git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> /usr/bin/git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 0ff9fcd2bf396d3f3e1ef8728cbabe3dc887dfb (refs/remotes/origin/main)
> /usr/bin/git config core.sparsecheckout # timeout=10
> /usr/bin/git checkout -f 0ff9fcd2bf396d3f3e1ef8728cbabe3dc887dfb # timeout=10
> /usr/bin/git branch -a -v --no-abbrev # timeout=10
> /usr/bin/git checkout -b main 0ff9fcd2bf396d3f3e1ef8728cbabe3dc887dfb # timeout=10
Commit message: "Update pom.xml"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Nous avons donc terminé notre première étape. Passons maintenant à la deuxième étape qui est Maven Build.

## II. Deuxième étape : construction de Maven

### Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Clone Repo') {
6       steps {
7         git branch: 'main', url: 'https://github.com/eliesjebri/maven-web-app.git'
8       }
9     }
10    stage('Build Stage') {
11      steps {
12        sh "mvn clean package"
13      }
14    }
15    tools {
16      maven 'Maven-3.8.6'
17    }
18  }
19 }
20
21
```

☒ Use Groovy Sandbox ?[Pipeline Syntax](#)

Save

Apply

Maintenant, appliquez et enregistrez.

Exécutez simplement le pipeline et voyez si la deuxième étape est en cours de traitement ou non. Alors cliquez sur **Construire maintenant**.

The screenshot shows the Jenkins dashboard for the 'maven-web-app-pipeline'. The top navigation bar includes 'Dashboard' and 'maven-web-app-pipeline'. The main content area shows the pipeline status as 'maven-web-app-pipeline' with a green checkmark. Below this, there are links for 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'Rename', and 'Pipeline Syntax'. The 'Stage View' section displays a table with stage names and their average times:

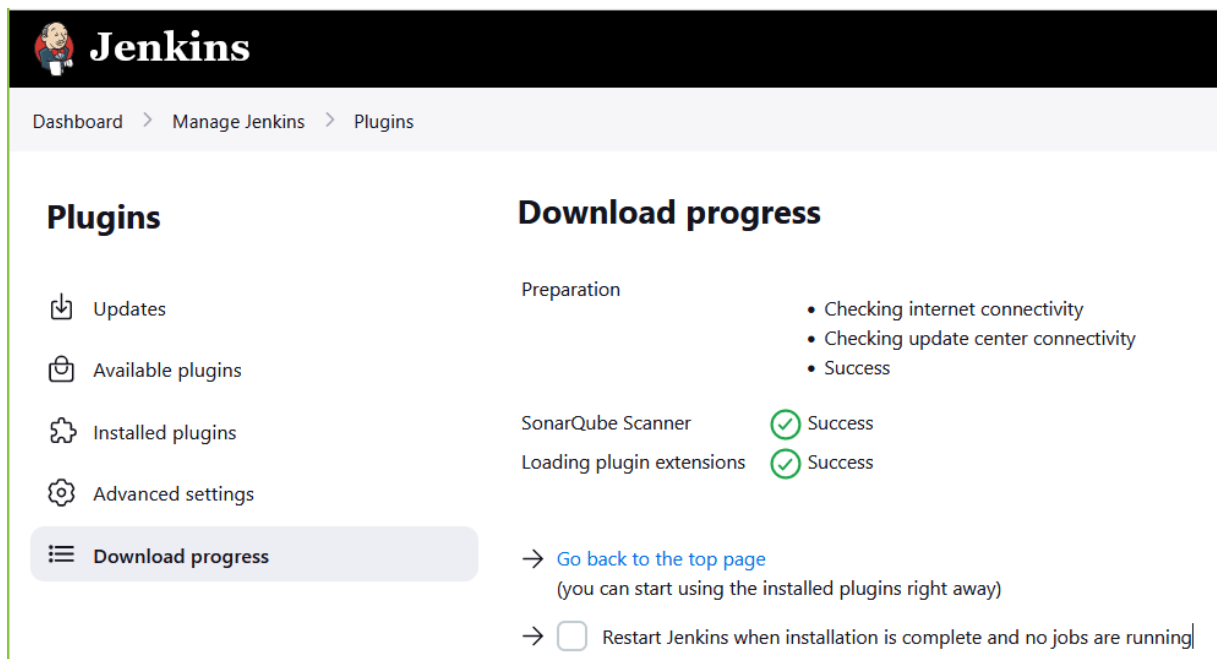
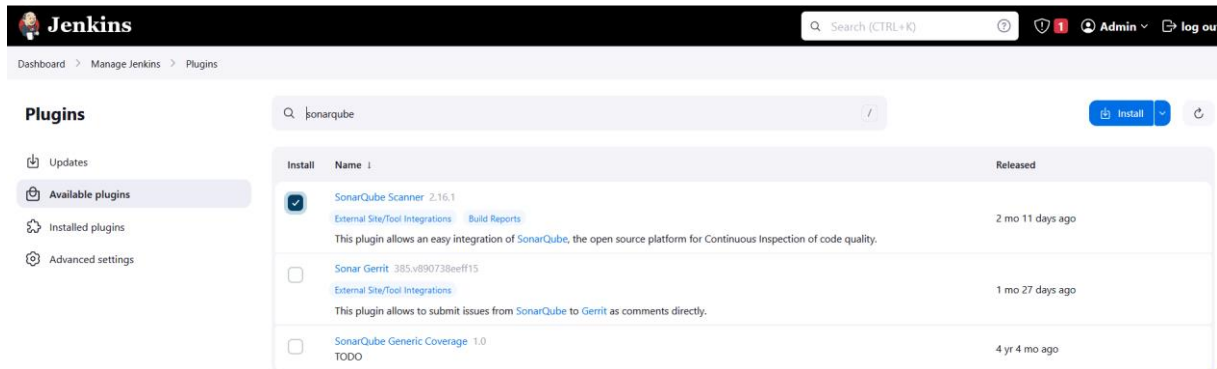
Stage	Average time
Clone Repo	1s
Build Stage	53s

The 'Build History' section shows a list of builds with columns for build number, status, and time. The first build is #10, completed on Dec 20, 2023, at 10:11 PM. The second build is #2, completed on Dec 20, 2023, at 6:39 PM. The 'Permalinks' section provides links to the last build, last stable build, last successful build, and last completed build, all of which are #10, completed 18 minutes ago.

Ainsi, alors que la deuxième étape est terminée, à savoir le build avec Maven, nous allons maintenant passer à la troisième étape, à savoir la révision du code.

### III. Étape 3 : révision du code

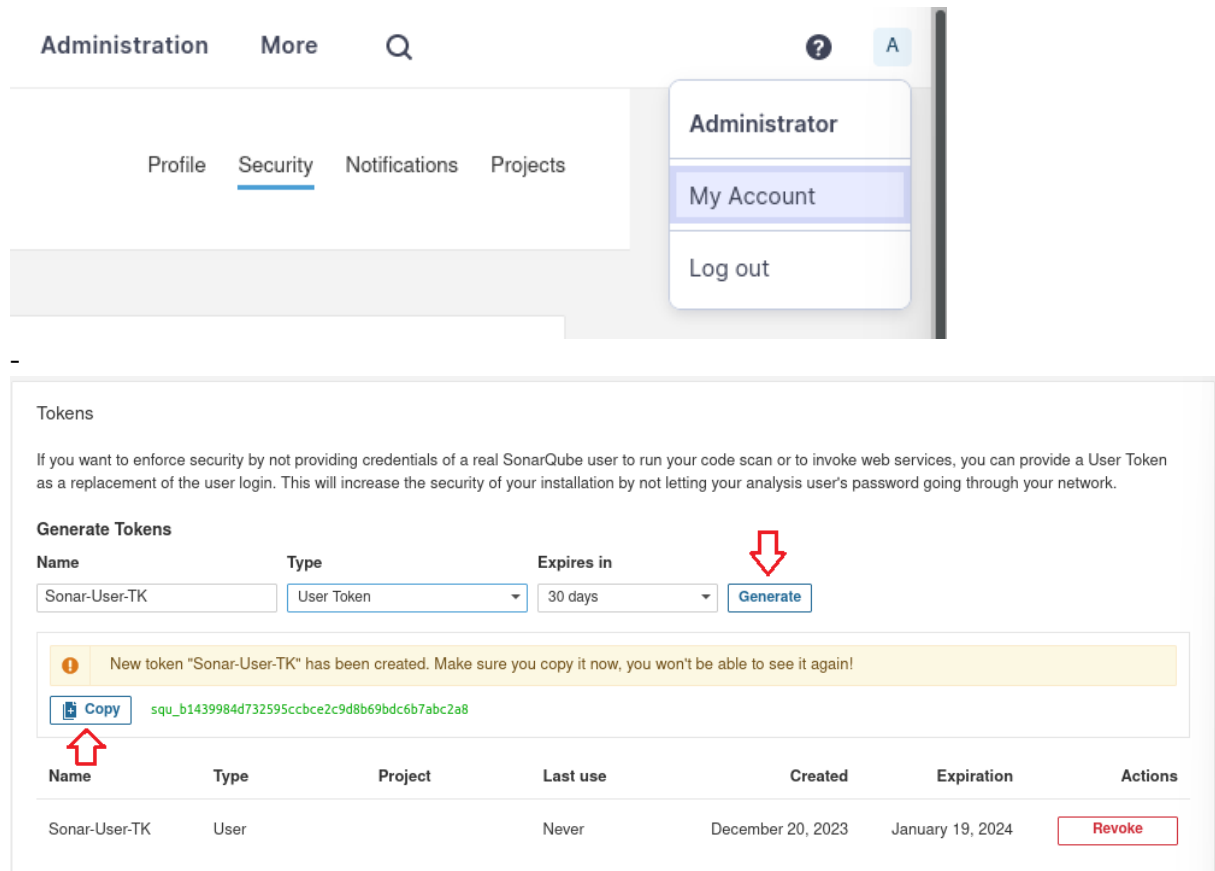
Cette étape sera entièrement prise en charge par Sonarqube pour la revue et l'analyse du code. Pour cela, nous devons intégrer Sonarqube avec Jenkins Server. Afin d'intégrer Sonarqube à Jenkins, nous devons installer un plugin nommé SonarQube Scanner. Sélectionnez ce plugin et installez-le sans redémarrer.





Connectez-vous maintenant à SonarQube, puis accédez à Administrator >> Compte >> Sécurité >> Générer un jeton

Ici, générez un jeton utilisateur et copiez le jeton dans le panneau Jenkins.



**Administration** More ? A

Profile **Security** Notifications Projects

**Administrator**

My Account

Log out

**Tokens**

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

**Generate Tokens**

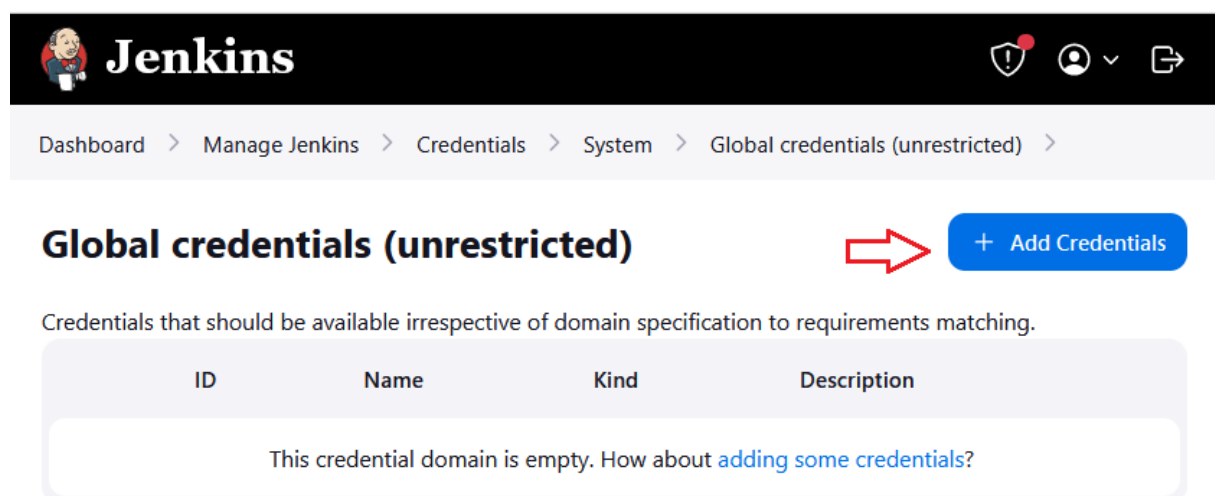
Name: Sonar-User-TK Type: User Token Expires in: 30 days **Generate**

New token "Sonar-User-TK" has been created. Make sure you copy it now, you won't be able to see it again!

**Copy** squ\_b1439984d732595ccbc2c9d8b69bdc6b7abc2a8

Name	Type	Project	Last use	Created	Expiration	Actions
Sonar-User-TK	User		Never	December 20, 2023	January 19, 2024	<b>Revoke</b>

Ajoutez vos nouvelles informations d'identification de SonarQube dans Jenkins en vous déplaçant vers le chemin Dashboard - Manage Jenkins Credentials - System - Global credentials (unrestricted). Ajoutez votre clé secrète ici.



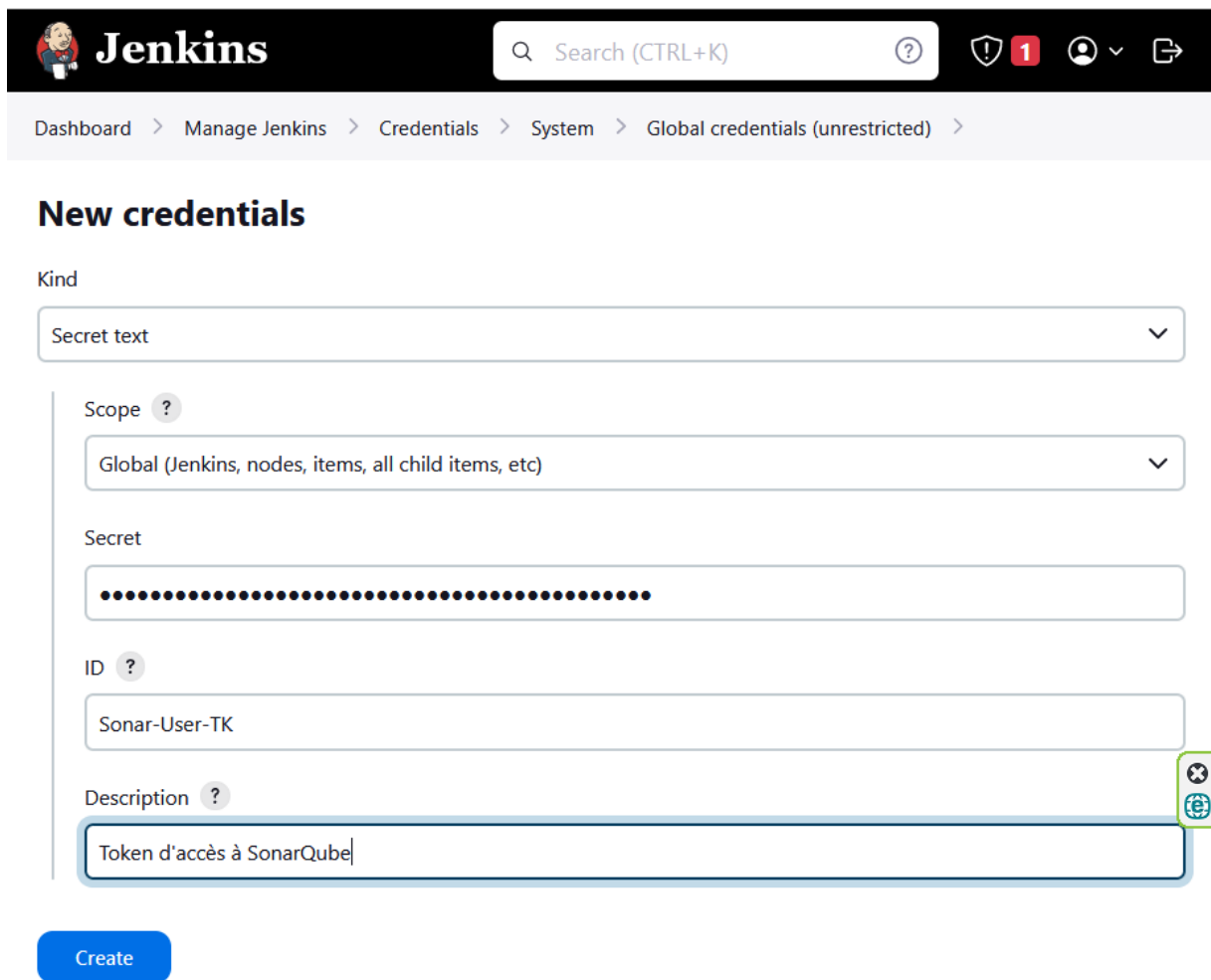
**Jenkins**

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

**Global credentials (unrestricted)** **+ Add Credentials**

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
This credential domain is empty. How about <a href="#">adding some credentials</a> ?			



The image shows the 'New credentials' form in Jenkins. The breadcrumb trail is 'Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted)'. The form has the following fields:

- Kind:** A dropdown menu with 'Secret text' selected.
- Scope:** A dropdown menu with 'Global (Jenkins, nodes, items, all child items, etc)' selected.
- Secret:** A text input field filled with dots, representing a masked password or token.
- ID:** A text input field containing 'Sonar-User-TK'.
- Description:** A text input field containing 'Token d'accès à SonarQube'.

At the bottom of the form is a blue 'Create' button.

Dans le tableau de bord Jenkins, accédez à Dashboard - Manage Jenkins - System – SonarQube servers.


### SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ **Environment variables** Enable injection of SonarQube server configuration as build environment variables

### SonarQube installations

List of SonarQube installations

Add SonarQube 


### Pipeline Speed / Durability

Save

Apply

## SonarQube servers


If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☒ Environment variables 

### SonarQube installations


List of SonarQube installations

Name

SonarQube-community-10.3.0 


Server URL

Default is http://localhost:9000

http://192.168.0.155:9000 

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Token d'accès à SonarQube 

+ Add ▾

Advanced ▾

Save

Apply

Cliquez sur Appliquer et Enregistrer. Avec cela, vous avez configuré Jenkins avec SonarQube. Passons maintenant au pipeline et ajoutons une étape pour l'étape SonarQube.

## Pipeline

Definition

Pipeline script

Script ?

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Clone Repo') {
6       steps {
7         git branch: 'main', url: 'https://github.com/eliesjebri/maven-web-app.git'
8       }
9     }
10    stage('Build Stage') {
11      steps {
12        sh "mvn clean package"
13      }
14
15      tools {
16        maven 'Maven-3.8.6'
17      }
18    }
19    stage('Code Review') {
20      steps {
21        withSonarQubeEnv('SonarQube-community-10.3.0') {
22          sh "mvn clean package sonar:sonar"
23          echo 'Static Analysis Completed'
24        }
25      }
26
27      tools {
28        maven 'Maven-3.8.6'
29      }
30    }
31  }
32 }
33
```

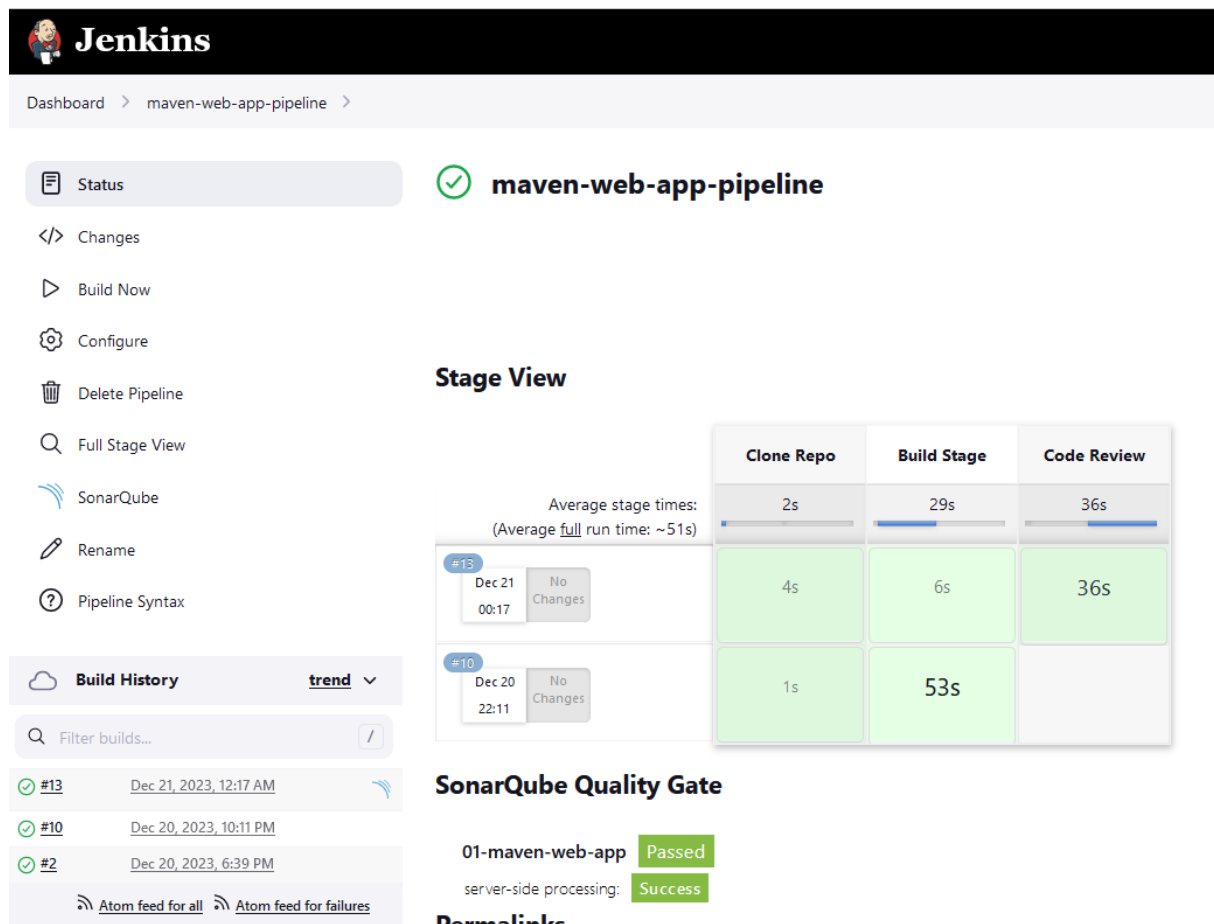
☒ Use Groovy Sandbox ?

Save

Apply

Cliquez sur Appliquer et enregistrez-le pour d'autres procédures.

Maintenant, exécutez Build Now et voyez si le pipeline est entièrement fonctionnel ou non. C'est le moment où nous avons trois étapes en cours.



The Jenkins dashboard shows the 'maven-web-app-pipeline' in a 'Stage View'. The pipeline consists of three stages: 'Clone Repo', 'Build Stage', and 'Code Review'. The average stage times are 2s, 29s, and 36s respectively, with an average full run time of approximately 51s. The pipeline is currently in a 'Success' state.

Stage	Clone Repo	Build Stage	Code Review
Average stage times	2s	29s	36s
#13 (Dec 21, 00:17)	4s	6s	36s
#10 (Dec 20, 22:11)	1s	53s	

**SonarQube Quality Gate**

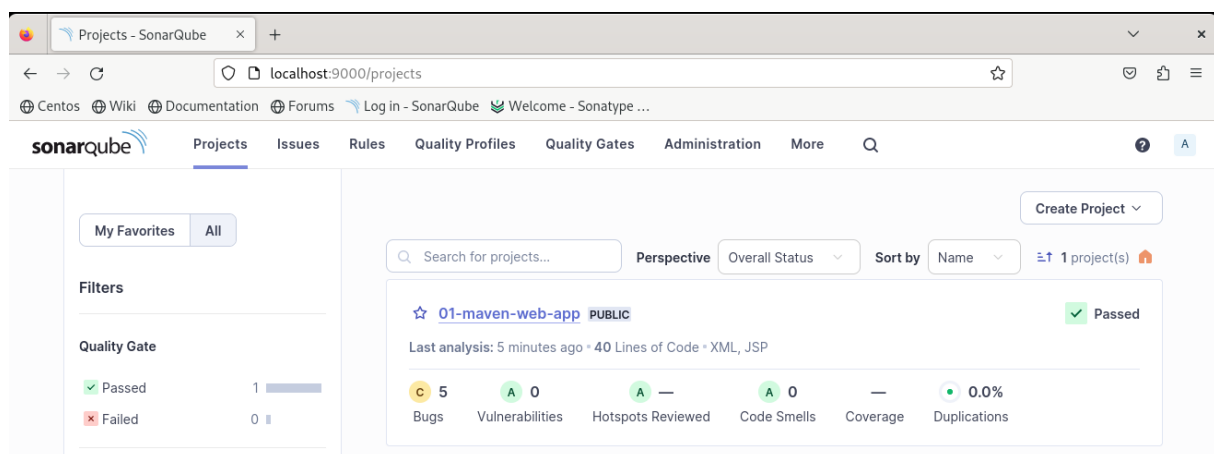
01-maven-web-app **Passed**  
server-side processing: **Success**

**Permalinks**

**Build History**

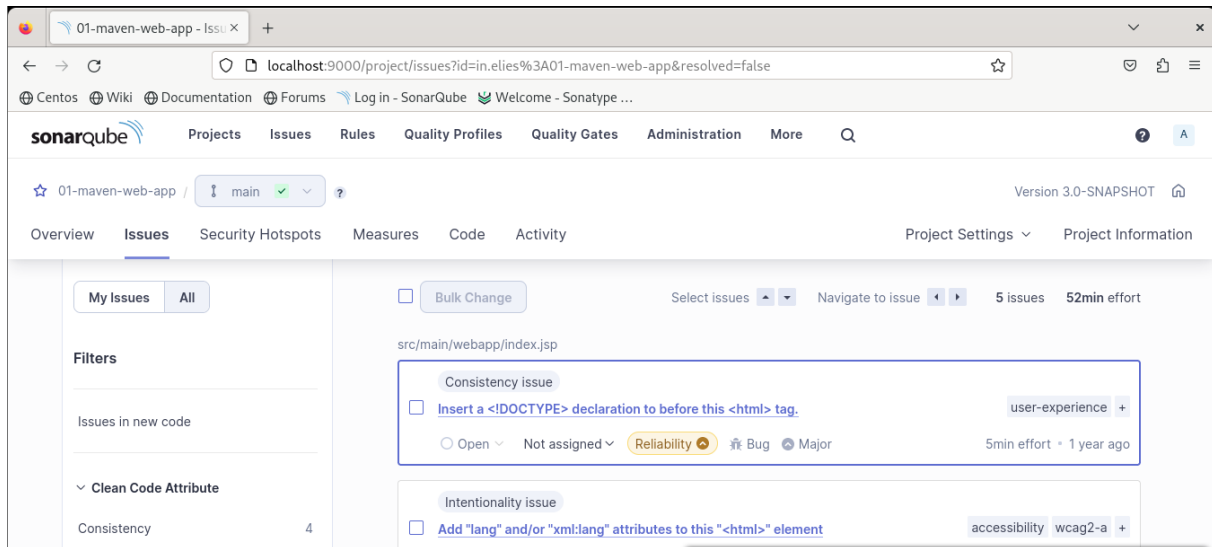
Build	Time
#13	Dec 21, 2023, 12:17 AM
#10	Dec 20, 2023, 10:11 PM
#2	Dec 20, 2023, 6:39 PM

Si la construction est en phase de réussite, vous constaterez que votre application Web Maven se trouve dans SonarQube.



The SonarQube interface shows the project '01-maven-web-app' in a 'Passed' state. The last analysis was performed 5 minutes ago, resulting in 40 Lines of Code (XML, JSP). The project has 5 Bugs, 0 Vulnerabilities, 0 Hotspots Reviewed, 0 Code Smells, 0.0% Coverage, and 0 Duplications.

Metric	Value
Bugs	5
Vulnerabilities	0
Hotspots Reviewed	0
Code Smells	0
Coverage	0.0%
Duplications	0



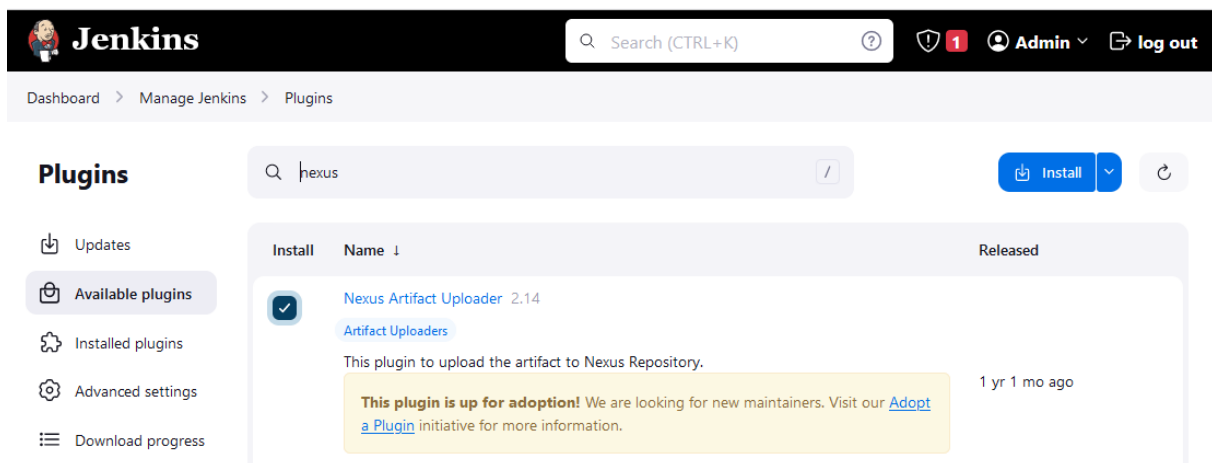
Construisons maintenant notre quatrième étape.

#### IV. Étape 4 : Télécharger les artefacts

Pour cette étape, nous devons configurer le système de référentiel d'artefacts Nexus dans Jenkins.

Accédez donc au tableau de bord Jenkins Dashboard - Manage Jenkins - Plugins

Installez Nexus Artifact Uploader



Ajoutons la quatrième étape dans le Pipeline en vous aidant avec l'assistant.

## Configuration

- General
- Advanced Project Options
- Pipeline**

### Script ?

```
10 sh "${mavenCMD} clean package"
11 }
12
13 stage('Code Review'){
14     withSonarQubeEnv('Sonar-Server-7.8'){
15         def mavenHome = tool name: "Maven-3.8.6", type: "maven"
16         def mavenCMD = "${mavenHome}/bin/mvn"
17         sh "${mavenCMD} sonar:sonar"
18     }
19 }
20
21 stage('Upload Build Artifact'){
22     |
23 }
24
25 }
26 }
```

☒ Use Groovy Sandbox ?

[Pipeline Syntax](#)

**Save** **Apply**

Utilisez le générateur de syntaxe du pipeline pour générer l'étape du script.

Dashboard > maven-web-app-pipeline > Pipeline Syntax

### Snippet Generator

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 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.)

#### Steps

Sample Step: **nexusArtifactUploader: Nexus Artifact Uploader**

**Nexus Details**

Nexus Version: **NEXUS3**

Protocol: **HTTP**

Nexus URL: **http://192.168.0.155:8081**  
**URL must not start with http:// or https://**

Credentials: **- none -**

Ajoutez les paramètres d'accès à Nexus

Dashboard > maven-web-app-pipeline > Pipeline Syntax

**URL must not start with http:// or https://**

Credentials: **- none -**

**+ Add**

**Jenkins**

Domain  
Global credentials (unrestricted)

Kind  
Username with password

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

Username ?  
admin

☐ Treat username as secret ?

Password ?  
\*\*\*\*\*

ID ?  
Nexus-Creds

Description ?  
Login-pass-Nexus

Récupérez les paramètres suivants depuis le fichier pom.xml

### Credentials

admin/\*\*\*\*\* (Login-pass-Nexus)

+ Add ▾

### GroupId

in.elies

### Version

1.0-SNAPSHOT

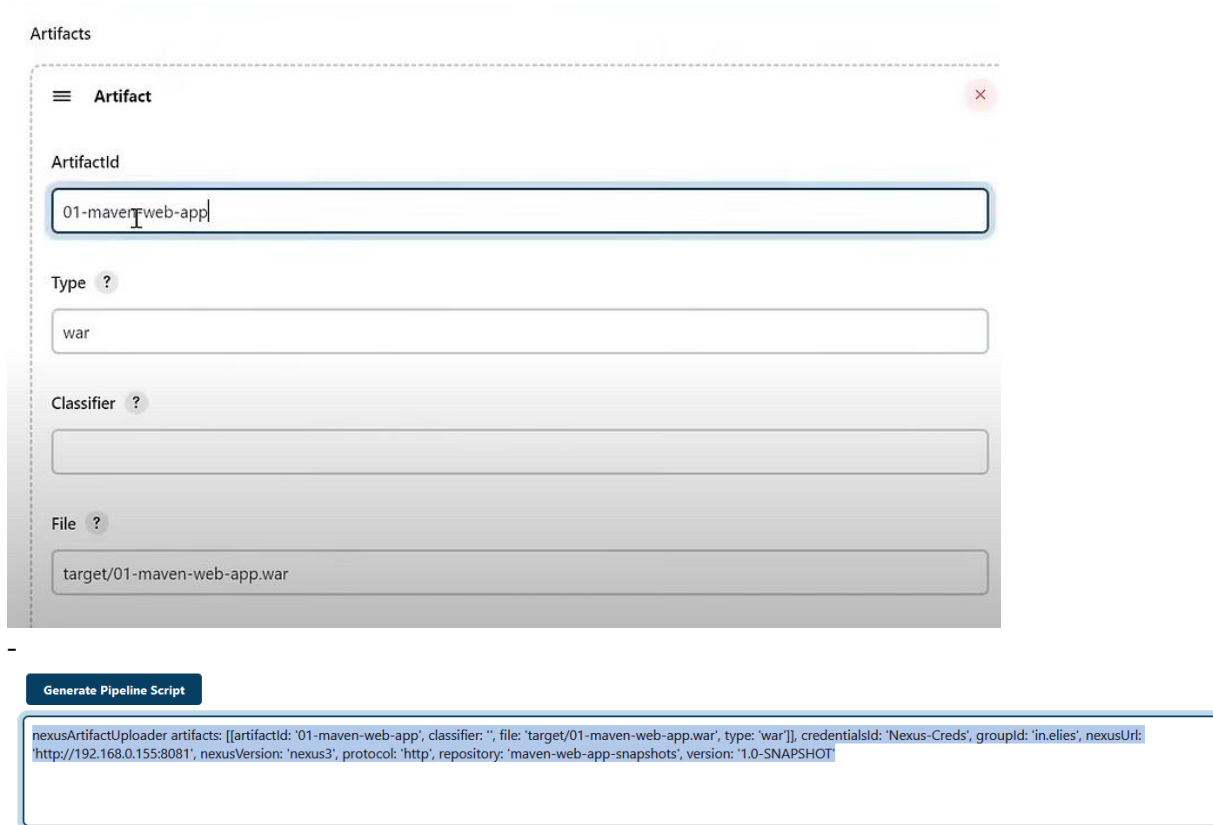
### Repository ?

maven-web-app-snapshots



Fournissez les détails du référentiel Nexus selon votre configuration dans la syntaxe du pipeline Jenkins.

Ajoutez maintenant les détails aux onglets Artefact.



Artifacts

Artifact

ArtifactId

01-maven-web-app

Type ?

war

Classifier ?

File ?

target/01-maven-web-app.war

Generate Pipeline Script

```
nexusArtifactUploader artifacts: [[artifactId: '01-maven-web-app', classifier: '', file: 'target/01-maven-web-app.war', type: 'war']], credentialsId: 'Nexus-Creds', groupId: 'in.elies', nexusUrl: 'http://192.168.0.155:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-web-app-snapshots', version: '1.0-SNAPSHOT']
```

Passez maintenant le script dans l'étape Nexus.

```
nexusArtifactUploader artifacts: [[artifactId: '01-maven-web-app', classifier: '', file: 'target/01-maven-web-app.war', type: 'war']], credentialsId: 'Nexus-Creds', groupId: 'in.elies', nexusUrl: '192.168.0.155:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-web-app-snapshots', version: '1.0-SNAPSHOT']
```

```
Script ?
1 pipeline {
2   agent any
3
4   stages {
5     stage('Clone Repo') {
6       steps {
7         git branch: 'main', url: 'https://github.com/eliesjebri/maven-web-app.git'
8       }
9     }
10    stage('Build Stage') {
11      steps {
12        sh "mvn clean package"
13      }
14
15      tools {
16        maven 'Maven-3.8.6'
17      }
18    }
19    stage('Code Review') {
20      steps {
21        withSonarQubeEnv('SonarQube-community-10.3.0') {
22          sh "mvn clean package sonar:sonar"
23          echo 'Static Analysis Completed'
24        }
25      }
26
27      tools {
28        maven 'Maven-3.8.6'
29      }
30    }
31    stage('Upload Build Artifact') {
32      steps {
33        nexusArtifactUploader artifacts: [[artifactId: '01-maven-web-app', classifier: '', file: 'target/01-maven-web-app.war', type: 'war']]
34      }
35    }
36  }
37 }
```

Save

Apply

Cliquez sur Appliquer et Enregistrer.

Maintenant, exécutons le build maintenant et vérifions si le pipeline **Télécharger les artefacts** fonctionne ou non. À l'heure actuelle, nous avons 4 étapes différentes à parcourir.

**Jenkins**

Dashboard > maven-web-app-pipeline

Status ✓ maven-web-app-pipeline

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

SonarQube

Rename

Pipeline Syntax

**Build History** trend

Filter builds...

#23 Dec 21, 2023, 2:00 PM

#13 Dec 21, 2023, 12:17 AM

#10 Dec 20, 2023, 10:11 PM

#2 Dec 20, 2023, 6:39 PM

Atom feed for all Atom feed for failures

**Stage View**

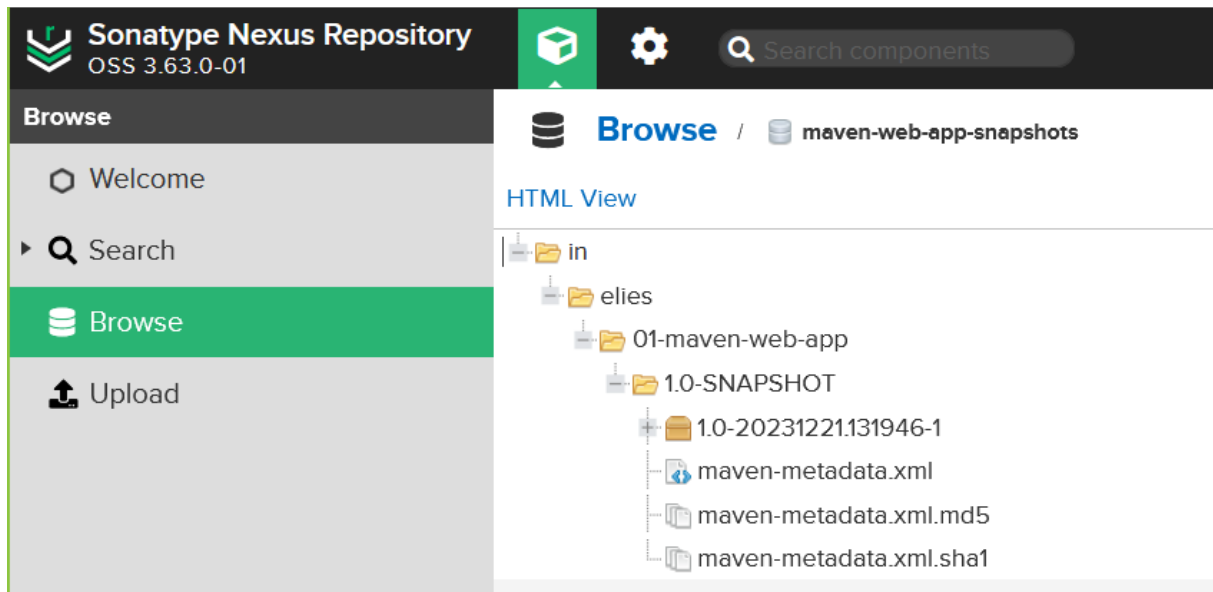
	Clone Repo	Build Stage	Code Review	Upload Build Artifact
Average stage times: (Average full run time: ~43s)	2s	21s	27s	1s
#23 Dec 21 14:00 No Changes	861ms	4s	18s	1s
#13 Dec 21 00:17 No Changes	4s	6s	36s	
#10 Dec 20 22:11 No Changes	1s	53s		

**SonarQube Quality Gate**

01-maven-web-app Passed

server-side processing: Success

Vérifiez maintenant votre référentiel Nexus, vous trouverez les artefacts téléchargés dans le système d'annuaire.



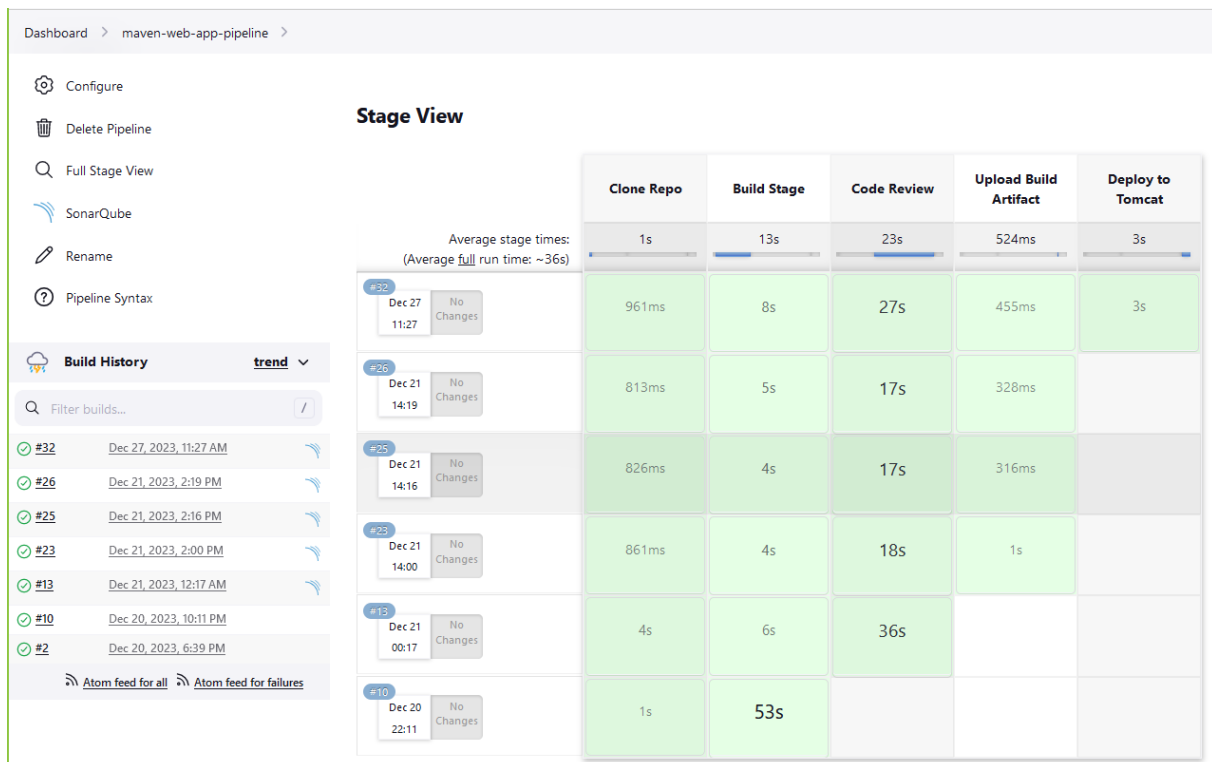
Nous pouvons désormais constater que les 4 étapes ont été franchies avec succès. Nous allons maintenant passer à notre dernière étape de déploiement de l'application.

## Étape 5 : déployer l'application

Ajoutez le stage.

```
36 stage('Deploy to Tomcat') {
37     steps {
38         script {
39             def remote = [:]
40             remote.name = '192.168.0.154'
41             remote.host = '192.168.0.154'
42             remote.user = 'root'
43             remote.password = 'password'
44             remote.allowAnyHosts = true
45             writeFile file: 'target/01-maven-web-app.war', text: 'ls -lrt'
46             sshPut remote: remote, from: 'target/01-maven-web-app.war', into: '/var/lib/tomcat/webapps'
47         }
48     }
49 }
```

Cliquez sur Appliquer et Enregistrer et exécutez la construction maintenant.



Une fois votre build lancé avec succès, vous pouvez maintenant vérifier sur le serveur Tomcat l'application Web en cours d'exécution.

Manager						
<a href="#">List Applications</a>	<a href="#">HTML Manager Help</a>			<a href="#">Manager Help</a>		<a href="#">Server Sta</a>
Applications						
Path	Version	Display Name	Running	Sessions	Commands	
/	None specified	Welcome to Tomcat	true	0	<div>StartStopReloadUndeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>	
/01-maven-web-app	None specified	Archetype Created Web Application	true	0	<div>StartStopReloadUndeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>	
/docs	None specified	Tomcat Documentation	true	0	<div>StartStopReloadUndeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>	
/examples	None specified	Servlet and JSP Examples	true	0	<div>StartStopReloadUndeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>	
/host-manager	None specified	Tomcat Host Manager Application	true	0	<div>StartStopReloadUndeploy</div> <div>Expire sessions with idle ≥ 30 minutes</div>	

Voici donc le pipeline CICD complet.