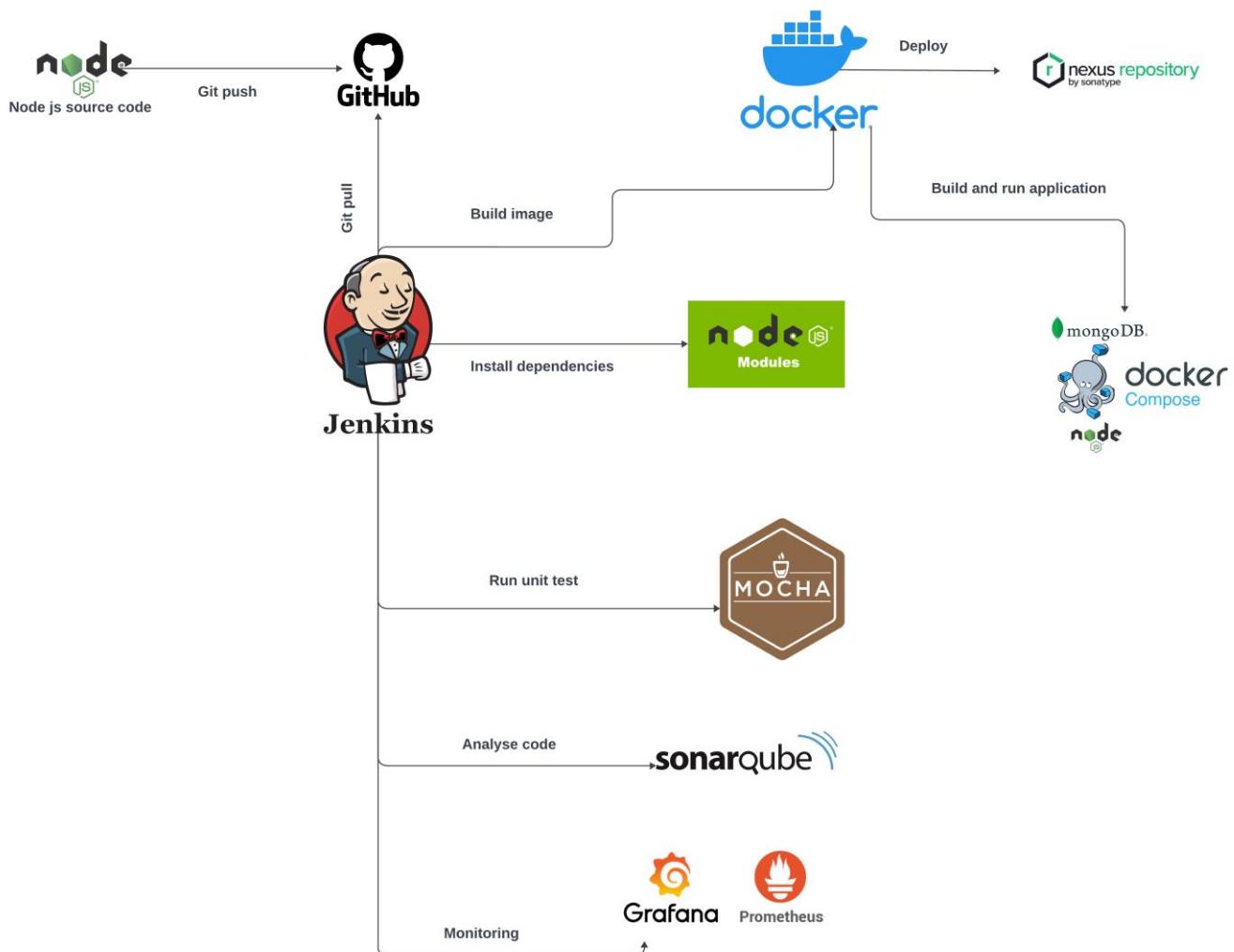


Atelier : Guide de Configuration de Pipeline CI/CD pour une Application Node.js avec MongoDB

Objectifs :

L'objectif de cet atelier est de fournir un guide étape par étape pour la mise en place d'un pipeline de CI/CD pour une application Node.js avec MongoDB. Ce pipeline inclut l'intégration avec Jenkins pour l'automatisation, SonarQube pour l'analyse statique du code, Mocha pour les tests unitaires, Docker pour la conteneurisation, Nexus Repository pour le stockage des images Docker, et Prometheus avec Grafana pour la surveillance.

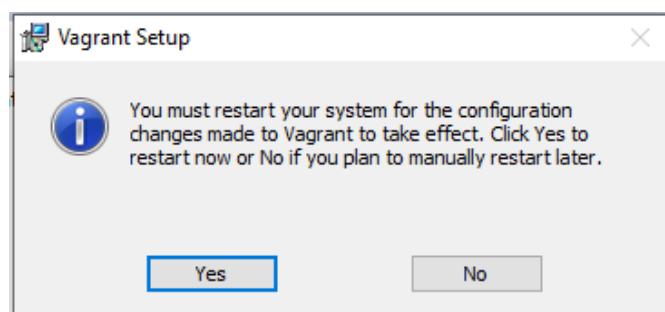


1. Installation et configuration Vagrant:

Avant l'installation de Vagrant, Vous devez avoir une boîte virtuelle installée. Vous pouvez télécharger Virtualbox via ce lien : <https://www.virtualbox.org/wiki/Downloads>.

Passons maintenant à l'installation de vagrant. La première étape à faire est de télécharger l'installation en accédant à ce lien : <https://www.vagrantup.com/downloads>

Redémarrez l'ordinateur après l'installation.

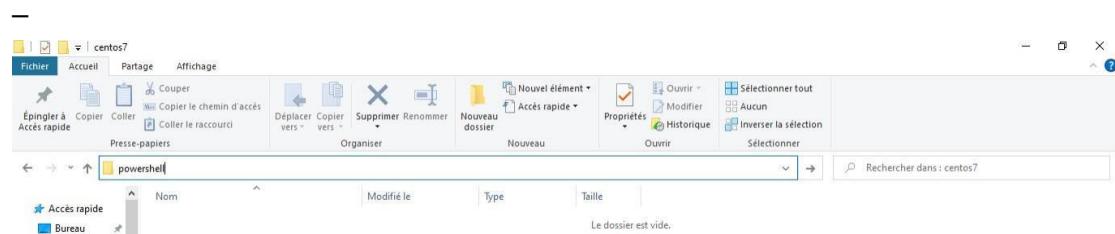


Pour vérifier que Vagrant est installé, ouvrez un PowerShell et exécutez la commande suivante :

```
vagrant --version  
Windows PowerShell  
Copyright (c) Microsoft corporation. Tous droits réservés.  
Testez le nouveau système multiplateforme Powershell https://aka.ms/pscore6  
PS C:\Users\Mehdy> vagrant --version  
Vagrant 2.4.0  
PS C:\Users\Mehdy>
```

Créer et configurer une machine virtuelle avec Vagrant :

- 1/ Par souci d'homogénéité, vous pouvez créer un dossier D:\Vagrant, dans lequel, vous allez trouver toutes les machines virtuelles dont vous aurez besoin par la suite.
- 2/ Créez un dossier nommé « Ubuntu » et ouvrez un PowerShell.



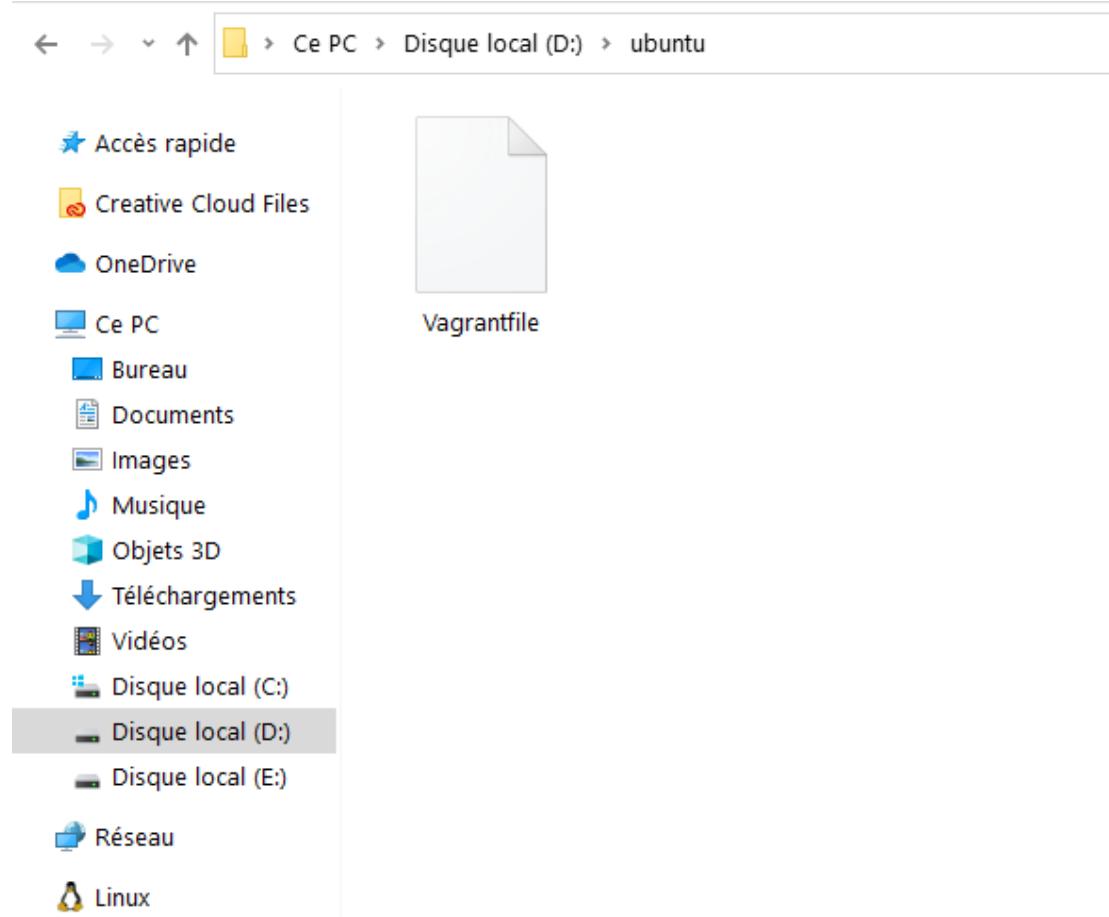
3/ Exécutez la commande **vagrant init** pour créer le fichier de configuration Vagrantfile.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS D:\ubuntu> vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
PS D:\ubuntu>
```

4/ Ouvrez le fichier Vagrantfile où vous trouverez une configuration par défaut avec les explications des différents champs en commentaire.



Pour créer une machine virtuelle Ubuntu avec une configuration basique, vous pouvez utiliser le Vagrantfile ci-dessous :

```
Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-22.04"
  config.vm.network "private_network", ip: "192.168.33.10"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "4000"
    vb.cpus = "2"
  end
end
```

NB : Après chaque modification du fichier Vagrantfile, vous devez exécuter la commande **vagrant reload**.

5/ Après, ouvrez VirtualBox et exécutez la commande **vagrant up** dans le PowerShell pour créer la machine.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://
PS D:\ubuntu> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'bento/ubuntu-22.04'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'bento/ubuntu-22.04' version '2...
...
==> default: Setting the name of the VM: ubuntu_default_1702...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configur...
  default: Adapter 1: nat
  default: Adapter 2: hostonly
==> default: Forwarding ports...
  default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few...
  default: SSH address: 127.0.0.1:2222
  default: SSH username: vagrant
  default: SSH auth method: private key
  default:
  default: Vagrant insecure key detected. Vagrant will aut...
  default: this with a newly generated keypair for better...
  default: Inserting generated public key within guest...
  default: Removing insecure key from the guest if it's pr...
  default: Key inserted! Disconnecting and reconnecting us...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
  default: /vagrant => D:/ubuntu
PS D:\ubuntu>
```

Pour connaître l'état de la machine virtuelle, il suffit de taper dans le PowerShell la commande **vagrant status**.

```
Windows PowerShell
PS D:\ubuntu> vagrant status
Current machine states:

default          running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
PS D:\ubuntu>
```

6/ Pour accéder à la machine et créer un client vagrant, il suffit d'exécuter dans PowerShell la commande **vagrant ssh**

```
vagrant@vagrant:~$ PS_D:\ubuntu> vagrant ssh
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-83-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Mon Dec 11 10:40:05 AM UTC 2023

System load: 0.01904296875 Processes: 159
Usage of /: 12.0% of 30.34GB Users logged in: 0
Memory usage: 4% IPv4 address for eth0: 10.0.2.15
Swap usage: 0% IPv4 address for eth1: 192.168.33.10

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-83-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Mon Dec 11 10:40:05 AM UTC 2023

System load: 0.01904296875 Processes: 159
Usage of /: 12.0% of 30.34GB Users logged in: 0
Memory usage: 4% IPv4 address for eth0: 10.0.2.15
Swap usage: 0% IPv4 address for eth1: 192.168.33.10

This system is built by the Bento project by Chef Software
More information can be found at https://github.com/chef/bento
vagrant@vagrant:~$ 6/ Pour accéder à la machine et créer un client vagrant, il suffit d'exécuter dans powershell la
mande vagrant ssh-bash: 6/: No such file or directory
vagrant@vagrant:~$
```

2. Jenkins :

a) Installation de Java 17 :

- Se placer un PowerShell au niveau du dossier de votre VM Ubuntu et lancez votre VM (**vagrant up**), et lancer un shell (**vagrant ssh**).
- Installer open jdk 17 et open jre 17.
 - **sudo apt-get update**
 - **sudo apt install openjdk-17-jdk openjdk-17-jre**
- Pour vérifier que le JDK 17 est bien installé, exécuter la commande suivante :
 - **Java --version**

```
vagrant@vagrant:~$ java --version
openjdk 17.0.9 2023-10-17
OpenJDK Runtime Environment (build 17.0.9+9-Ubuntu-122.04)
OpenJDK 64-Bit Server VM (build 17.0.9+9-Ubuntu-122.04, mixed mode, sharing)
vagrant@vagrant:~$
```

- Configuration des variables d'environnement : Afin d'aider les applications basées sur Java à localiser correctement la machine virtuelle Java, vous devez définir deux variables d'environnement : "JAVA_HOME" et "JRE_HOME"
 - **sudo cp /etc/profile /etc/profile_backup**
 - **echo 'export JAVA_HOME=/usr/lib/jvm/default-java' | sudo tee -a /etc/profile**

- **echo 'export JRE_HOME=/usr/lib/jvm/default-java' | sudo tee -a /etc/profile**
- **source /etc/profile**

```
vagrant@vagrant:~$ sudo cp /etc/profile /etc/profile_backup
vagrant@vagrant:~$ echo 'export JAVA_HOME=/usr/lib/jvm/default-java' | sudo tee -a /etc/profile
export JAVA_HOME=/usr/lib/jvm/default-java
vagrant@vagrant:~$ echo 'export JRE_HOME=/usr/lib/jvm/default-java' | sudo tee -a /etc/profile
export JRE_HOME=/usr/lib/jvm/default-java
vagrant@vagrant:~$ source /etc/profile
vagrant@vagrant:~$
```

tee : écrit le contenu du stdout vers un fichier

source : exécute le contenu d'un fichier

- Enfin, exécuter les commandes suivantes pour vérifier :

- **echo \$JAVA_HOME**
- **echo \$JRE_HOME**

```
vagrant@vagrant:~$ echo $JAVA_HOME
/usr/lib/jvm/default-java
vagrant@vagrant:~$ echo $JRE_HOME
/usr/lib/jvm/default-java
vagrant@vagrant:~$
```

b) Installation de Node js:

Pour installer Node js, vous devez lancer le terminal et exécuter les commandes suivantes:

- **sudo apt update**
- **sudo apt install -y curl**
- **curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -**
- **sudo apt install nodejs**

```
Sélection vagrant@vagrant:
vagrant@vagrant:~$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  javascript-common libcurl4 libcurl4-openssl-dev libnode72 nodejs-doc
Suggested packages:
  apache2 | lighttpd | httpd npm
The following NEW packages will be installed:
  javascript-common libcurl4 libcurl4-openssl-dev libnode72 nodejs nodejs-doc
0 upgraded, 6 newly installed, 0 to remove and 76 not upgraded.
Need to get 13.7 MB of archives.
After this operation, 53.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

- **sudo apt install npm**

Pour vérifier que node et npm sont bien installés :

➤ **node -v**

```
vagrant@vagrant:~$  
vagrant@vagrant:~$ node -v  
v18.17.1
```

➤ **npm -v**

```
vagrant@vagrant:~$ npm -v  
9.6.7
```

c) Installation de Jenkins:

Pour installer Jenkins, vous devez exécuter les 4 commandes suivantes:

➤ **curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \ /usr/share/keyrings/jenkins-keyring.asc > /dev/null**

➤ **echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \ https://pkg.jenkins.io/debian binary/ | sudo tee \ /etc/apt/sources.list.d/jenkins.list > /dev/null**

➤ **sudo apt-get update**

➤ **sudo apt install jenkins**

Pour lancer Jenkins

➤ **sudo systemctl start jenkins.service**

Pour vérifier l'installation de jenkins

➤ **sudo systemctl status jenkins.service**

```
vagrant@vagrant:~$ sudo systemctl status jenkins  
● jenkins.service - Jenkins Continuous Integration Server  
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)  
   Active: active (running) since Mon 2023-12-11 11:41:04 UTC; 35s ago  
     Main PID: 9357 (java)  
        Tasks: 52 (limit: 5615)  
       Memory: 1.4G  
          CPU: 8.032s  
         cGroup: /system.slice/jenkins.service  
                 └─9357 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080  
  
Dec 11 11:40:34 vagrant jenkins[9357]: f95a0e9d51714336aa3b1b504f910101  
Dec 11 11:40:34 vagrant jenkins[9357]: This may also be found at: '/var/lib/jenkins/secrets/initialAdminPassword'  
Dec 11 11:40:34 vagrant jenkins[9357]: ****  
Dec 11 11:40:34 vagrant jenkins[9357]: ****  
Dec 11 11:40:34 vagrant jenkins[9357]: ****  
Dec 11 11:41:04 vagrant jenkins[9357]: 2023-12-11 11:41:04.036+0000 [id=32]      INFO    Jenkins.InitReactorRunner$1#onAttained: Complete  
Dec 11 11:41:04 vagrant jenkins[9357]: 2023-12-11 11:41:04.081+0000 [id=25]      INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is f  
Dec 11 11:41:04 vagrant systemd[1]: Started Jenkins Continuous Integration Server.  
Dec 11 11:41:08 vagrant jenkins[9357]: 2023-12-11 11:41:08.114+0000 [id=54]      INFO    h.m.DownloadService$Downloadable#load: Obtained  
Dec 11 11:41:08 vagrant jenkins[9357]: 2023-12-11 11:41:08.115+0000 [id=54]      INFO    hudson.util.Retriger#start: Performed the action  
lines 1-20/20 (END)
```

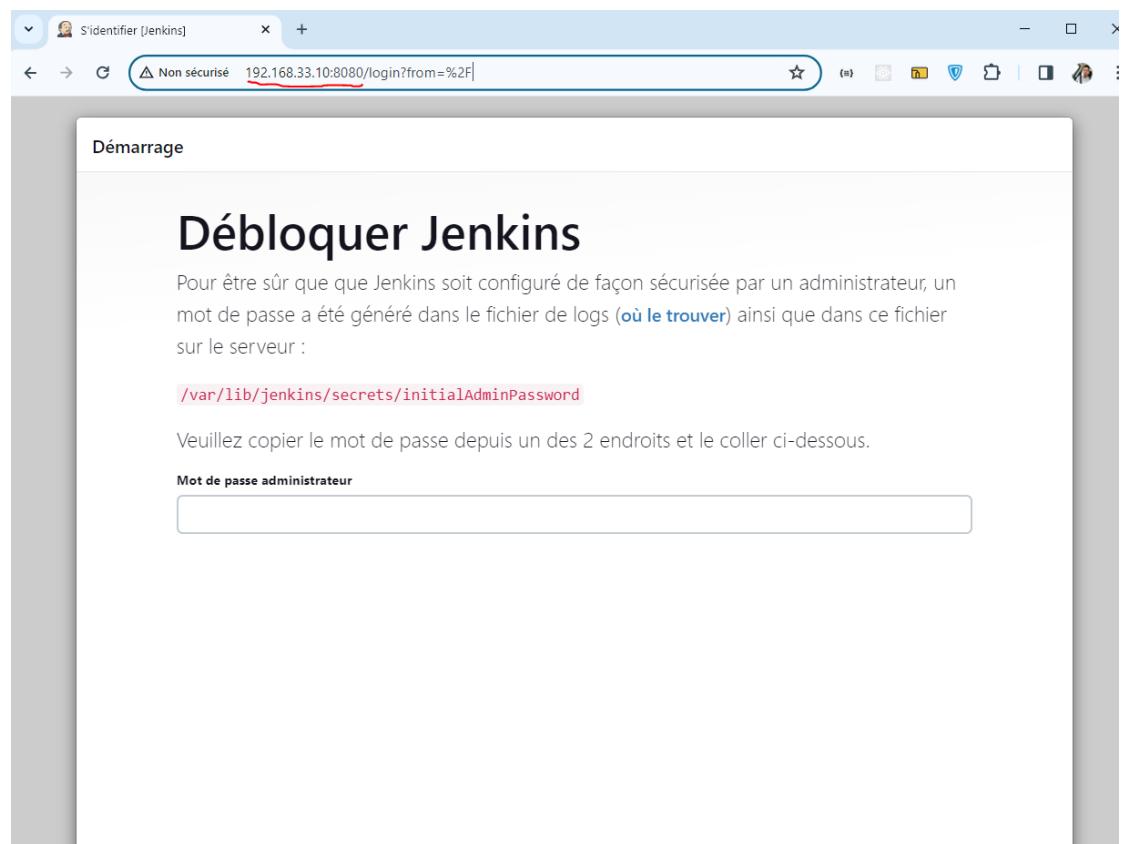
Pour lancer jenkins automatiquement comme service au prochains démarrages

➤ **sudo systemctl enable jenkins.service**

Pour accéder à Jenkins, vous devez récupérer l'adresse ip de la machine virtuelle à travers la commande :

➤ **ip addr show**

```
vagrant@vagrant:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3b:cf:90 brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic eth0
        valid_lft 81986sec preferred_lft 81986sec
        inet6 fe80::a00:27ff:fe3b:cf90/64 scope link
            valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:64:cf:08 brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    inet 192.168.33.10/24 brd 192.168.33.255 scope global eth1
        valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fe64:cf08/64 scope link
            valid_lft forever preferred_lft forever
vagrant@vagrant:~$
```

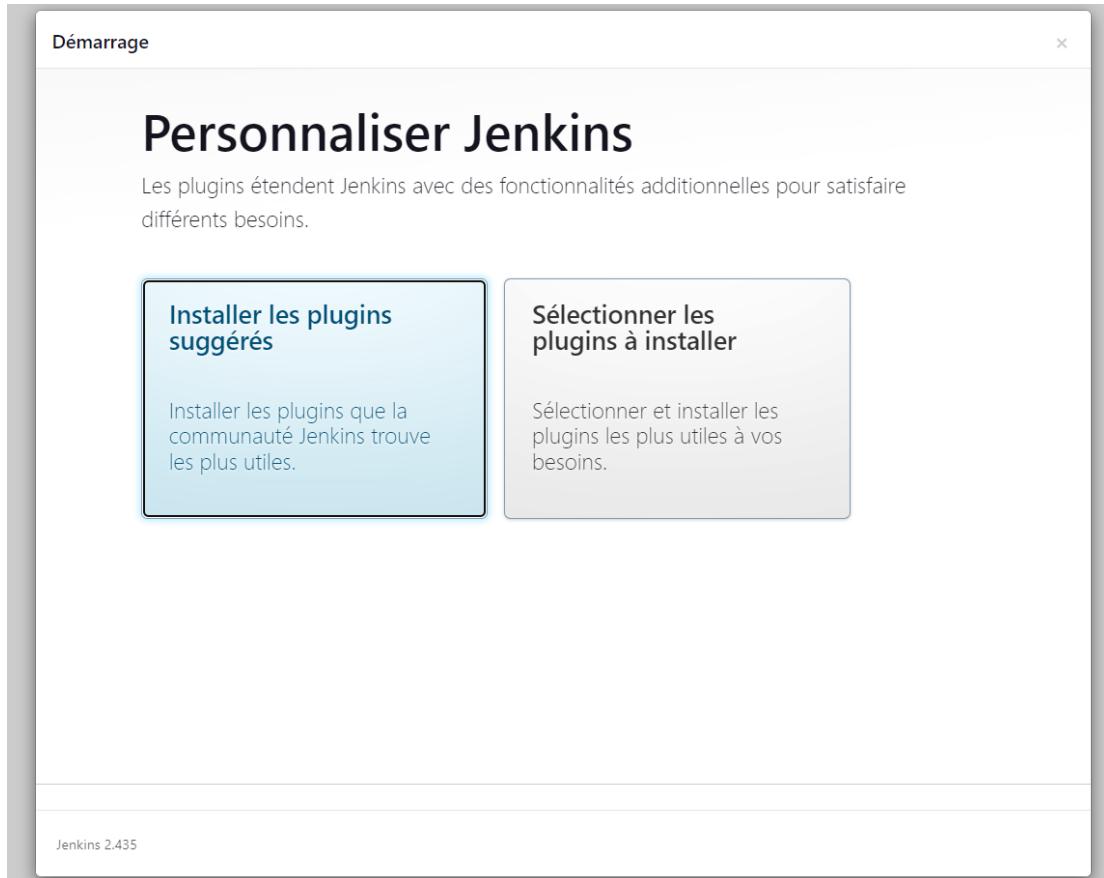


Pour la première fois, il faut débloquer Jenkins en tapant le mot de passe qui est stocké dans le fichier de log mentionné dans la fenêtre ci-dessus (utiliser la commande cat pour afficher le mot de passe)

➤ `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`

```
vagrant@vagrant:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
f95a0e9d51714336aa3b1b504f910101
vagrant@vagrant:~$
```

Installer les plugins suggérés.



Vous pouvez ne créer aucun nouvel utilisateur et continuer à utiliser le utilisateur « admin » :

Créer le 1er utilisateur Administrateur

Nom d'utilisateur:	<input type="text"/>
Mot de passe:	<input type="password"/>
Confirmation du mot de passe:	<input type="password"/>
Nom complet:	<input type="text"/>
Adresse courriel:	<input type="text"/>

3.2

[Continuer en tant qu'Administrateur](#)

[Sauver et continuer](#)

Vous pouvez passer cette étape et ne rien modifier :

Démarrage

Configuration de l'instance

URL de Jenkins :

L'URL de Jenkins est utilisée pour fournir l'URL de base pour les liens absous vers les diverses ressources Jenkins. Cela signifie que cette valeur est nécessaire pour le bon fonctionnement de nombreuses fonctionnalités de Jenkins, notamment les notifications par mail, les mises à jour des statuts des pull requests, et la variable d'environnement BUILD_URL fournie pour les étapes de build.

La valeur par défaut affichée **n'est pas encore sauvegardée** et est générée à partir de la requête actuelle, lorsque c'est possible. Il est fortement recommandé d'utiliser comme valeur l'URL qui est censée être utilisée par les utilisateurs. Cela évitera des confusions lors du partage ou de la visualisation de liens.

Jenkins 2.303.2 Passer cette étape et terminer Sauver et terminer

Changer le mot de passe de « admin » à « jenkins » par exemple :

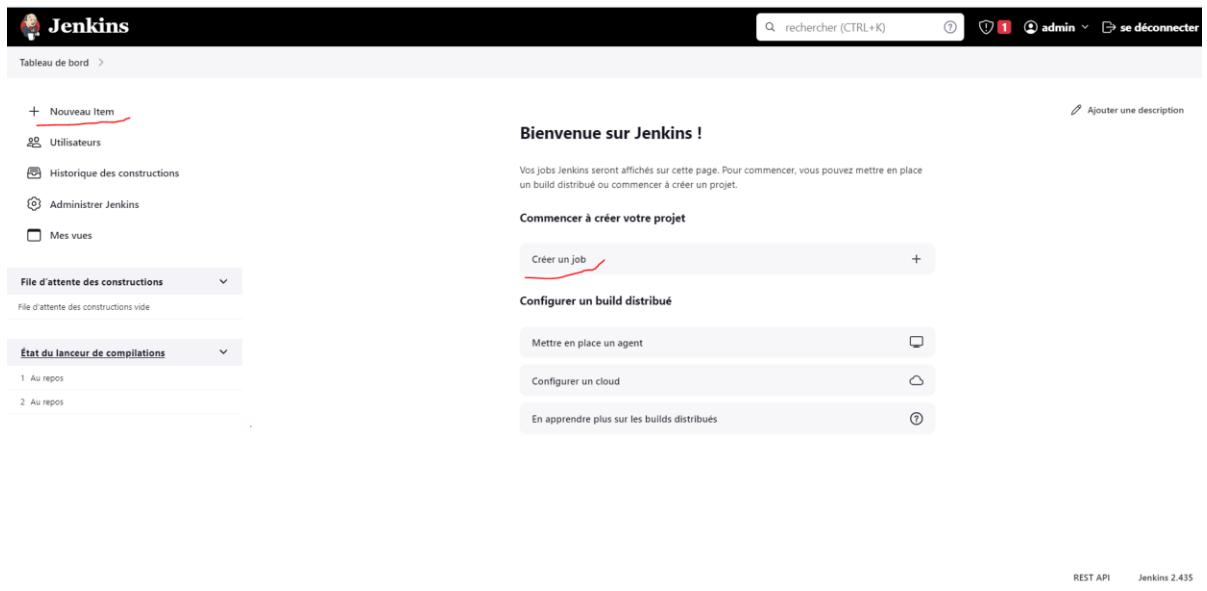
The screenshot shows the Jenkins sidebar menu. The 'Configure' item, which is under the 'Admin' section, is highlighted with a red underline. Other items visible in the menu include 'Constructions', 'Mes vues', and 'Identifiants'. The top right corner shows the user is logged in as 'admin' with 1 notification, and there is a 'Logout' button.

Mot de passe

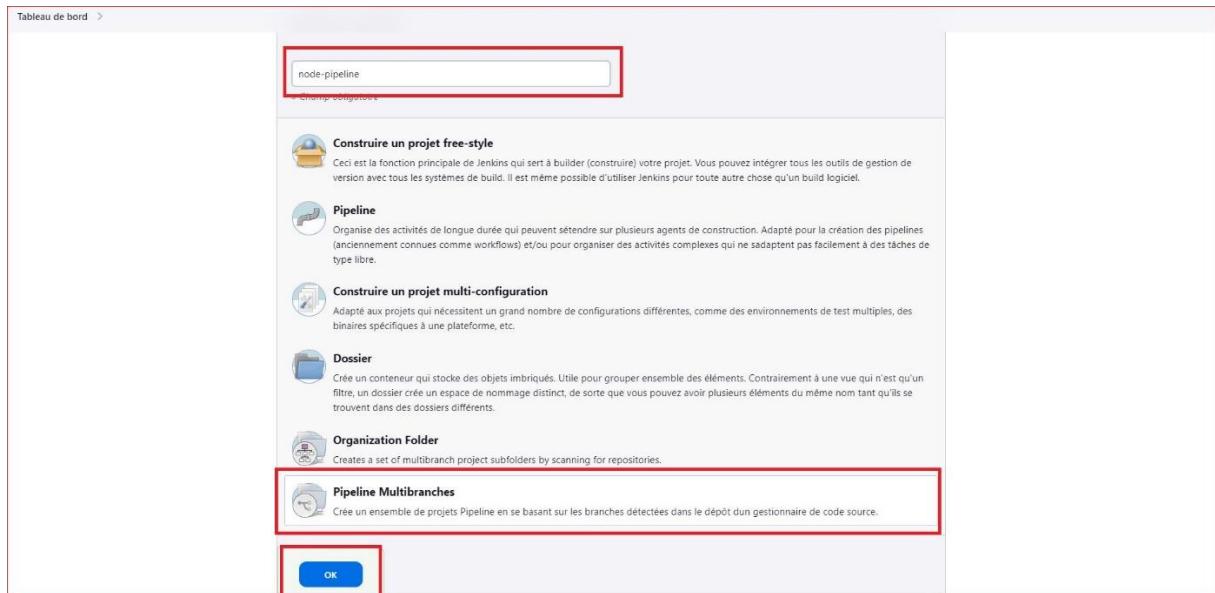
Mot de passe:

Confirmer le mot de passe:

➤ Création d'un pipeline multibranche avec Git

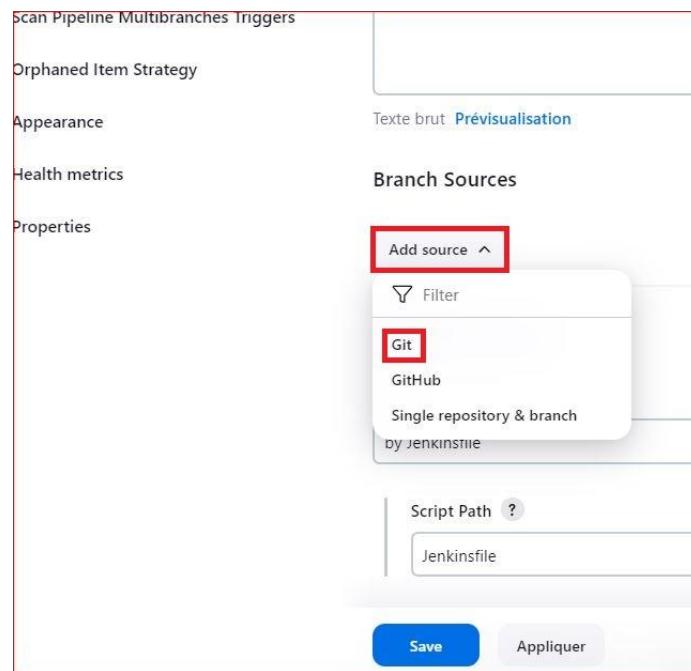


The screenshot shows the Jenkins dashboard. On the left sidebar, there is a red underline under the 'Nouveau Item' button. The main area features a 'Bienvenue sur Jenkins!' message and a 'Commencer à créer votre projet' section with a 'Créer un job' button also underlined in red. The 'File d'attente des constructions' and 'État du lanceur de compilations' sections are visible on the left.



The screenshot shows the 'Create New Item' dialog. A red box highlights the 'node-pipeline' input field. Another red box highlights the 'Pipeline Multibranches' option under the 'Pipeline' category. An 'OK' button at the bottom is also highlighted with a red box.

Récupération du code de Git : Utiliser des repo Git publics ou ajouter des clés SSH ou définir des noms d'utilisateur et des mots de passe dans la partie « Credentials » pour que Jenkins soit capable de récupérer le code de Git.



Lien github : <https://github.com/mehdybenromdhane/nodeProject.git>

The screenshot shows the Jenkins Configuration page. On the left, a sidebar lists configuration options: General, Branch Sources (selected), Build Configuration, Scan Pipeline Multibranches Triggers, Orphaned Item Strategy, Appearance, Health metrics, and Properties. The main area is titled 'Branch Sources'. It shows a 'Git' section with a 'Project Repository' field containing 'https://github.com/mehdybenromdhane/nodeProject.git'. Below it is a 'Credentials' section with a dropdown menu showing 'aucun'. A 'Discover branches' button is also present. At the bottom of the 'Branch Sources' section is an 'Add' button.

Ajouter email ou username et le mot de passe de votre compte Github.

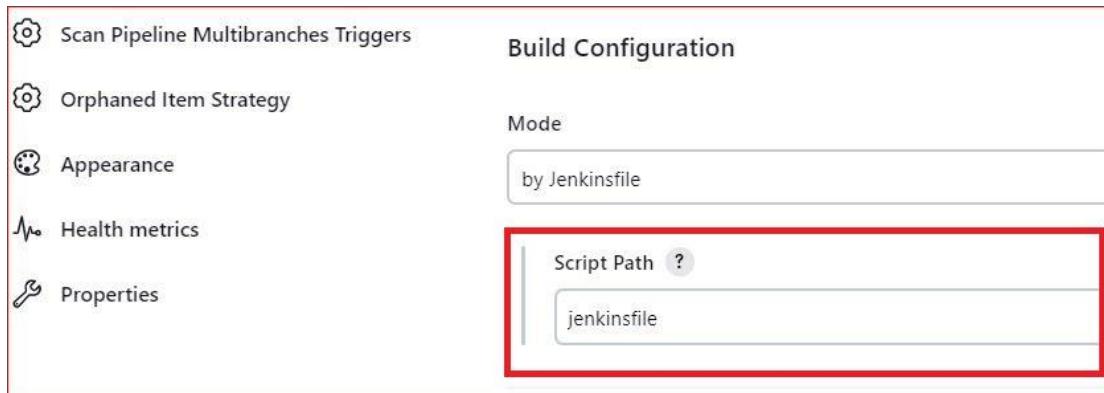
The screenshot shows a Jenkins configuration interface for adding a credential. The 'Domain' dropdown is set to 'Identifiants globaux (illimité)'. The 'Type' dropdown is set to 'Nom d'utilisateur et mot de passe'. The 'Portée' dropdown is set to 'Global (Jenkins, agents, items, etc...)'. The 'Nom d'utilisateur' field and the 'Mot de passe' field are highlighted with a red border, indicating they are required fields. A checkbox labeled 'Treat username as secret' is present but not checked. Below these fields are 'ID' and 'Description' fields, which are also highlighted with a red border.

Découvrir des branches par noms :

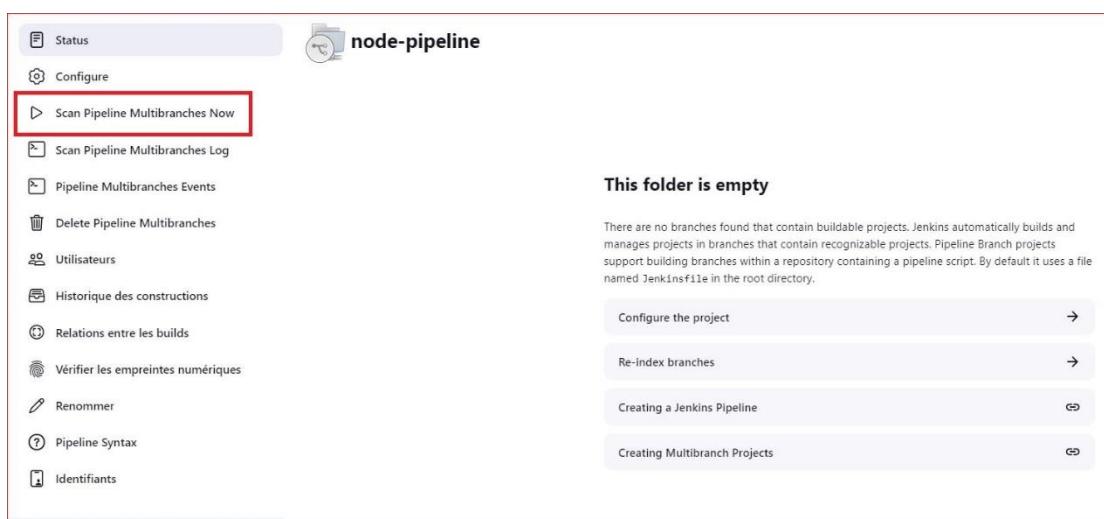
The screenshot shows the 'Behaviours' section in Jenkins. It features a 'Discover branches' card with a question mark icon. Below it is an 'Add' button with a dropdown arrow. A 'Filter' section is expanded, showing options like 'Within Repository', 'Discover branches', 'Discover other refs', 'Discover tags', and 'Filter by name (with regular expression)'. The 'Filter by name (with regular expression)' input field is highlighted with a red border. Other options like 'Filter by name (with wildcards)' and 'Additional' are also visible.

Par exemple il sera scanné uniquement depuis la branche **main**

Changer Jenkinsfile en jenkinsfile et cliquer sur appliquer et save.



Cliquer scan pour lire le fichier jenkinsfile à partir de répertoire Github



Ajouter les 3 stages suivantes dans votre jenkinsfile et faire les commit sur github

```
pipeline{
    agent any

    stages {

        stage('Install dependencies') {
            steps{
                script {
                    sh('npm install')
                }
            }
        }

        stage('Unit Test') {
            steps{

```

```

script {
    sh('npm test')
}
}

stage('Build application') {
    steps{
        script {
            sh('npm run build-dev')
        }
    }
}
}

```

Cliquer sur lancer un build

Status

main

Full project name: node-pipeline/main

Stage View

Declarative: Checkout SCM	Install dependencies	Unit Test	Build application
2s	1s	3s	6s

Average stage times:
(Average full run time: ~19s)

#3 déc. 11 1 commit
23:50

Historique des builds **tendance** /

Filter...

#3 11 déc. 2023 22:50

Atom feed des builds Atom feed des échecs

Liens permanents

- Dernier build (#3), il y a 45 s
- Dernier build stable (#3), il y a 45 s
- Dernier build avec succès (#3), il y a 45 s
- Dernier build complété (#3), il y a 45 s

d) Installation de Docker:

Pour installer docker dans notre machine virtuelle il faut exécuter les 7 commandes suivantes.

➤ **sudo apt update**

- `sudo apt install apt-transport-https ca-certificates curl software-properties-common -y`
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
- `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
- `apt-cache policy docker-ce`
- `sudo apt install docker-ce -y`
-

Docker est déjà lancé après l'installation, vérifier avec :

- `sudo systemctl status docker`

```
vagrant@vagrant:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-12-11 23:01:42 UTC; 4min 18s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
      Main PID: 8091 (dockerd)
        Tasks: 10
       Memory: 26.0M
          CPU: 4.058s
        CGroup: /system.slice/docker.service
                 └─8091 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Donner les droits d'accès en lecture et écriture, mais pas exécution pour le user « vagrant », son groupe et tout autre utilisateur (6 = (R=1 / W=1 X=0))

- `sudo chmod 666 /var/run/docker.sock`

La notation "666" dans le contexte de la commande chmod fait référence aux permissions sur un fichier dans un système Unix/Linux. Les permissions sur un fichier sont divisées en trois parties : propriétaire (user), groupe (group), et autres (others). Dans le système de notation octale utilisé par chmod, chaque type de permission est représenté par un chiffre :

- 4 représente la permission de lecture (r).
- 2 représente la permission d'écriture (w).
- 1 représente la permission d'exécution (x).

Ainsi, pour chaque partie (propriétaire, groupe, autres), vous ajoutez ces chiffres pour obtenir la combinaison correcte de permissions. Dans le cas de "666", cela signifie :

- Propriétaire : lecture (4) + écriture (2) = 6
- Groupe : lecture (4) + écriture (2) = 6
- Autres : lecture (4) + écriture (2) = 6

Pour s'assurer que tout est bon, vérifier la version de Docker et lancer l'image « hello-world » (qui sera récupérer de Docker Hub automatiquement) :

- **docker -v**

```
vagrant@vagrant:~$ docker -v
Docker version 24.0.7, build afdd53b
vagrant@vagrant:~$
```

- **docker run hello-world**

```
vagrant@vagrant:~$ docker run hello-world
unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:c79d06dfdf3d3eb04cafd0dc2bacab0992ebc243e083cab208bac4dd7759e0
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

e) Installation et configuration de SonarQube:

On va utiliser la commande docker pull pour télécharger une image Docker de SonarQube depuis Docker Hub.

- **docker pull sonarqube**

```
vagrant@vagrant:~$ docker pull sonarqube
Using default tag: latest
latest: Pulling from library/sonarqube
cbe3537751ce: Downloading [=====>] 11.5MB/30.45MB
6cd63fc495d1: Download complete
9c42674dea4f: Downloading [=====>] 8.563MB/47.15MB
a992519bbaee: Download complete
506583e9517b: Download complete
1f5ec129d7c9: Waiting
55466329b2ef: Waiting
-
```

Après avoir téléchargé l'image, vous pouvez lancer un conteneur SonarQube en utilisant la commande :

- **docker run -d --name sonar -p 9000:9000 sonarqube**

Cela démarre un conteneur SonarQube en arrière-plan (-d), nommé "sonar", en exposant le port 9000 pour accéder à l'interface web.

```
vagrant@vagrant:~$ docker run -d --name sonar -p 9000:9000 sonarqube
c78a36dfb896fcde806b4fd3b514bb3ec577c65126b9d2d8ee58a0e49c20dbe
```

Taper la commande **docker ps** pour afficher la liste des conteneurs Docker en cours d'exécution sur votre machine. Par défaut, cette commande affiche uniquement les conteneurs en cours d'exécution. Si vous souhaitez voir tous les conteneurs, y compris ceux qui ont déjà été arrêtés, vous pouvez utiliser : **docker ps -a**

```
vagrant@vagrant:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
c78a36dfb896        sonarqube          "/opt/sonarqube/dock..."   35 seconds ago    Up 29 seconds      0.0.0.0:9000->9000/tcp, :::9000->9000/tcp
vagrant@vagrant:~$
```

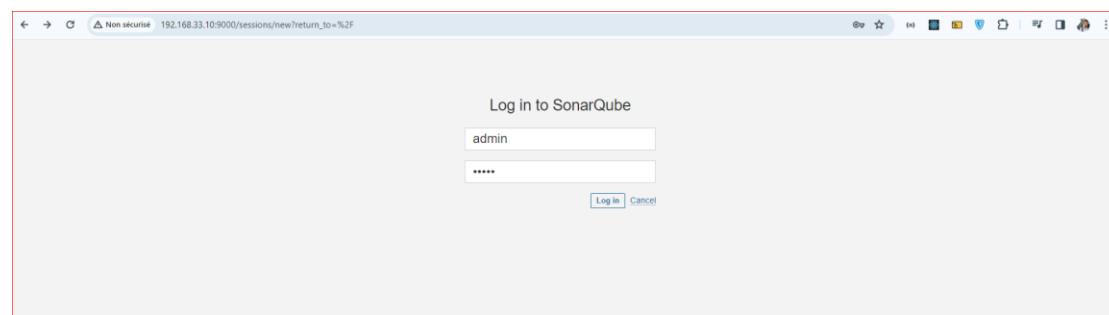
Attention : Si vous arrêtez le conteneur sonarqube, il ne faut pas lancer un docker run sur l'image sonarqube, car cela créera un nouveau conteneur. Il faut juste faire :

- **docker start id-conteneur-sonarqube**

Ou directement

- **docker start sonar**

Sur votre machine Windows, aller à l'url <http://<ip-vm>:9000> et se connecter avec admin/admin :



Changer le mot de passe à sonar par exemple (admin/sonar).

Après la connexion, cliquer sur **Create a local project**

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration More Q

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)?
Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

Import from Azure DevOps Import from Bitbucket Cloud Import from Bitbucket Server

Import from GitHub Import from GitLab

Are you just testing or have an advanced use-case? Create a local project.

Create a local project

Create a local project

Project display name *

nodeapp ✓

Up to 255 characters. Some scanners might override the value you provide.

Project key *

nodeapp ✓

The project key is a unique identifier for your project. It may contain up to 400 characters.
Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

main

The name of your project's default branch [Learn More ↗](#)

Next

Choose the baseline for new code for this project

Use the global setting

Previous version

Any code that has changed since the previous version is considered new code.

Recommended for projects following regular versions or releases.

Define a specific setting for this project

Previous version

Puisque nous n'avons pas de projets à analyser sur notre VM, nous allons utiliser Jenkins, pour récupérer un projet de Git, puis l'analyser avec Sonarqube .

Choisir la méthode d'analyse « avec Jenkins »

The screenshot shows a configuration page for analysis methods. At the top, a heading says 'Analysis Method' and a sub-instruction says 'Use this page to manage and set-up the way your analyses are performed.' Below, a section titled 'How do you want to analyze your repository?' contains five options: 'With Jenkins' (highlighted with a red box), 'With GitHub Actions', 'With GitLab CI', 'With Azure Pipelines', and 'Locally'. A note below 'Locally' says: 'Use this for testing or advanced use-case. Other modes are recommended to help you set up your CI environment.'

Se connecter à Jenkins via <http://192.168.33.10:8080/>

Administrer Jenkins > Plugins et installer la plugin SonarQube Scanner

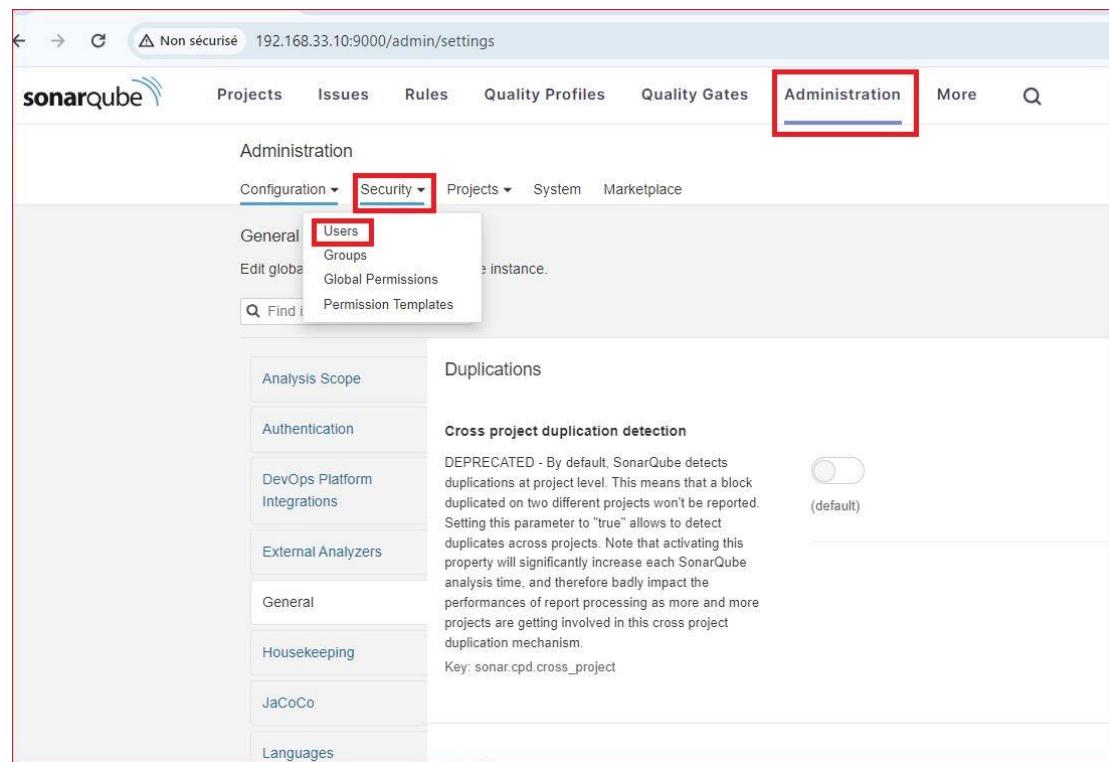
The screenshot shows the Jenkins 'Plugins' management interface. The 'Plugins disponibles' tab is selected. A search bar at the top has 'sonar' typed into it. In the search results, the 'SonarQube Scanner' plugin is listed with a checked checkbox next to its name. The 'Install' button for this plugin is highlighted with a red box.

The screenshot shows the Jenkins 'Progression du téléchargement' (Download Progress) page. The 'Progression des téléchargements' tab is selected. It displays two items: 'Préparation' (Preparation) and 'SonarQube Scanner'. Under Preparation, there are three bullet points: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. Under SonarQube Scanner, there are two entries: 'Loading plugin extensions' and 'Success' (indicated by a green checkmark). At the bottom, there are links to 'Revenir en haut de la page' (Return to top of page) and 'Redémarrer Jenkins quand l'installation est terminée et qu'aucun job n'est en cours' (Restart Jenkins when installation is finished and no jobs are running).

→ Jenkins doit être redémarré pour que la mise à jour soit effective.

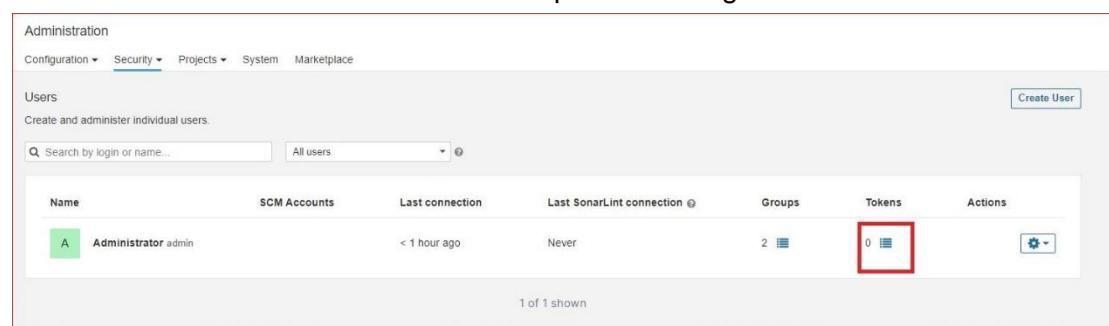
Pour récupérer le token d'accès nécessaire pour l'intégration de SonarQube dans Jenkins, suivez ces étapes :

- Connectez-vous à SonarQube :
- Accédez à l'Administration de SonarQube :



The screenshot shows the SonarQube administration interface at the URL `192.168.33.10:9000/admin/settings`. The top navigation bar has tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration (which is highlighted with a red box), More, and a search icon. Below the navigation is a sub-navigation bar with Configuration, Security (highlighted with a red box), Projects, System, and Marketplace. A dropdown menu for Security is open, showing options: General, Users (highlighted with a red box), Groups, Edit global permissions, Global Permissions, and Permission Templates. The main content area is titled 'Administration' and contains sections for Analysis Scope, Duplications, Authentication, DevOps Platform Integrations, External Analyzers, General, Housekeeping, JaCoCo, and Languages. A 'Cross project duplication detection' section is shown with a toggle switch set to off (disabled).

- Créez un nouveau token en fournissant un nom descriptif pour le référencer dans Jenkins. Assurez-vous de copier le token généré.



The screenshot shows the SonarQube users management interface. The top navigation bar includes Configuration, Security (selected and highlighted with a red box), Projects, System, and Marketplace. Below the navigation is a 'Users' section with a 'Create and administer individual users' link and a 'Create User' button. A search bar and a dropdown menu for filtering users ('All users') are also present. The main table lists users with columns: Name, SCM Accounts, Last connection, Last SonarLint connection, Groups, Tokens (highlighted with a red box), and Actions. One user, 'Administrator admin', is listed with 0 tokens. At the bottom of the page, it says '1 of 1 shown'.

Tokens of Administrator

Generate Tokens

Name	Expires in	Generate
Enter Token Name	30 days	Generate

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

Copy squ_7258fc17293f81db60d7af7c784a87b68cf9e1fb

Name	Type	Project	Last use	Created	Expiration	Actions
jenkins	User		Never	December 12, 2023	January 11, 2024	Revoke

[Done](#)

- Accédez à la section "Administrer Jenkins"
- Accédez à la section "Credentials"
- Sélectionnez le domaine Global

Credentials

T	P	Store	Domain	ID	Name
		System	(global)	git	mehdybenromdhane9@gmail.com/*****

Stores scoped to Jenkins

P	Store	Domains
	System	(global)

- Cliquez sur "Add Credentials."
- Choisissez le type d'informations d'identification approprié. Pour SonarQube, vous pouvez utiliser "Secret text"
- Remplissez ou collez le token secret de SonarQube.
- Cliquez sur "Create" pour enregistrer les informations d'identification.

New credentials

Type	Secret text
Portée	Global (Jenkins, agents, items, etc...)
Secret
ID	scanned
Description	
Create	

Naviguer sur Administrer Jenkins > System > SonarQube Server

- Sélectionnez Environnement variables
- Donnez un nom à votre installation, par exemple, "scanner".
- Ajoutez le lien de serveur SonarQube : <http://192.168.33.10:9000>
- Sélectionnez le token scanner.
- Enregistrez la configuration.

SonarQube servers

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

Environment variables

Installations de SonarQube

Liste des installations de SonarQube:

Nom	scanner
URL du serveur	Par défaut à http://localhost:9000 http://192.168.33.10:9000
Server authentication token	scanner + Ajouter ▾

Avancé ▾

Enregistrer **Appliquer**

Naviguer sur Administrer Jenkins > Tools > Installations SonarQube Scanner

Installations SonarQube Scanner

Installations SonarQube Scanner ^ Edited

Ajouter SonarQube Scanner

SonarQube Scanner

Name	scanner
<input checked="" type="checkbox"/> Install automatically ?	

Installer depuis Maven Central

Version	SonarQube Scanner 5.0.1.3006
---------	------------------------------

Ajouter un installateur ▾

Enregistrer **Appliquer**

- Donnez un nom à votre installation, par exemple, "scanner".
- Sélectionnez **Install automatically**. Et installer depuis maven central.
- Enregistrez la configuration.
- Créez un fichier **sonar-project.properties** dans votre projet et collez le code suivant `_sonar.projectKey=nodeapp`
- Au niveau de fichier Jenkinsfile dans votre projet ajoutez le code suivant avant le stage **Build application**:

```
stage('SonarQube Analysis') {
    steps{
        script {
            def scannerHome = tool 'scanner'
            withSonarQubeEnv {
                sh "${scannerHome}/bin/sonar-scanner"
            }
        }
    }
}
```

- Push sur github et lancer le pipeline

The screenshot shows the Jenkins Pipeline main view for the 'node-pipeline/main' build. The build status is green with a checkmark icon. The pipeline has one stage named 'main' which is also green. Below the stage name, it says 'Full project name: node-pipeline/main'. The 'Stage View' section shows the stages: 'Declarative: Checkout SCM', 'Install dependencies', 'Unit Test', 'SonarQube Analysis', and 'Build application'. Each stage has its average time listed: 3s, 40s, 17s, 2min 53s, and 27s respectively. A summary table at the bottom provides a detailed breakdown of the total run time:

	Declarative: Checkout SCM	Install dependencies	Unit Test	SonarQube Analysis	Build application
Average stage times:	3s	40s	17s	2min 53s	27s
(Average full run time: ~10min 33s)					
#4	1s	4s	19s	8min 39s	1min 21s
déc. 13 14:36					
No Changes					

The screenshot shows the SonarQube interface for a project named 'nodeapp'. The 'Measures' section is visible, displaying the following metrics:

- Reliability**: 0 Bugs (Status: A)
- Maintainability**: 0 Code Smells (Status: A)
- Security**: 1 Vulnerabilities (Status: E)
- Security Review**: 2 Security Hotspots (Status: E)
- Coverage**: 0.0% Coverage (Coverage on 80 Lines to cover) (Status: O)
- Duplications**: 0.0% Duplications (Duplications on 153 Lines) (Status: G)
- Unit Tests**: 0 Duplicated Blocks (Status: G)

f) Construire les images Docker (Node et mongoDB) :

Pour installer docker dans notre machine virtuelle il faut exécuter les 3 commandes suivantes :

- **sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-\$(uname -s)-\$(uname -m)" -o /usr/local/bin/docker-compose**

Ensuite, définissez les autorisations correctes afin que la commande docker-compose soit exécutable :

- **sudo chmod +x /usr/local/bin/docker-compose**

Pour vérifier que l'installation a réussi, exécuter :

- **docker-compose --version**

```
vagrant@vagrant:~$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
```

- Créer le fichier dockerfile dans votre projet node et coller le code suivant :

```
FROM node:16-alpine
WORKDIR /app
COPY . /app
RUN npm install
RUN npm run build-dev
EXPOSE 5000
CMD ["npm", "start"]
```

- Créer le fichier docker-compose.yml dans votre projet node et coller le code suivant :

```
version: '3.8'
services:
  db:
    image: mongo:4.2
    container_name: db
    restart: always

    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example
  ports:
    - 27017:27017
  volumes:
    - ./data/dump:/dump

  app:
    build: .
    image: 192.168.33.10:8083/nodemongoapp:6.0
    restart: always
    container_name: back
    ports:
      - '5000:5000'
    depends_on:
      - db
  volumes:
    mongo-data:
```

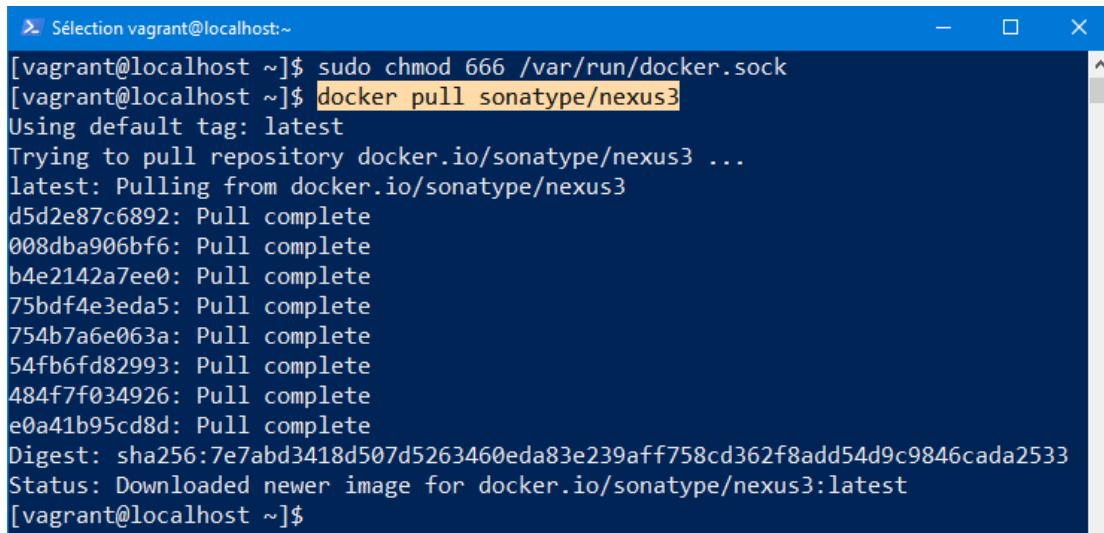
- Au niveau de fichier Jenkinsfile dans votre projet ajoutez le code suivant après le stage **Build application**:

```
// Building Docker images
stage('Building images (node and mongo)') {
    steps{
        script {
            sh('docker-compose build')
        }
    }
}
```

- Push sur github et lancer le pipeline

g) Installation et configuration Nexus :

- Nous allons utiliser une image Docker.
- Pour télécharger une image **nexus3**, vous devez exécuter la commande suivante: **docker pull sonatype/nexus3** (docker doit être up, faites un chmod avant) :



The screenshot shows a terminal window titled "Sélection vagrant@localhost:~". The command entered is "sudo chmod 666 /var/run/docker.sock" followed by "docker pull sonatype/nexus3". The output shows the progress of the pull, including layer names and their download status, until it reaches the final digest and status message: "Status: Downloaded newer image for docker.io/sonatype/nexus3:latest".

```
[vagrant@localhost ~]$ sudo chmod 666 /var/run/docker.sock
[vagrant@localhost ~]$ docker pull sonatype/nexus3
Using default tag: latest
Trying to pull repository docker.io/sonatype/nexus3 ...
latest: Pulling from docker.io/sonatype/nexus3
d5d2e87c6892: Pull complete
008dba906bf6: Pull complete
b4e2142a7ee0: Pull complete
75bdf4e3eda5: Pull complete
754b7a6e063a: Pull complete
54fb6fd82993: Pull complete
484f7f034926: Pull complete
e0a41b95cd8d: Pull complete
Digest: sha256:7e7abd3418d507d5263460eda83e239aff758cd362f8add54d9c9846cada2533
Status: Downloaded newer image for docker.io/sonatype/nexus3:latest
[vagrant@localhost ~]$
```

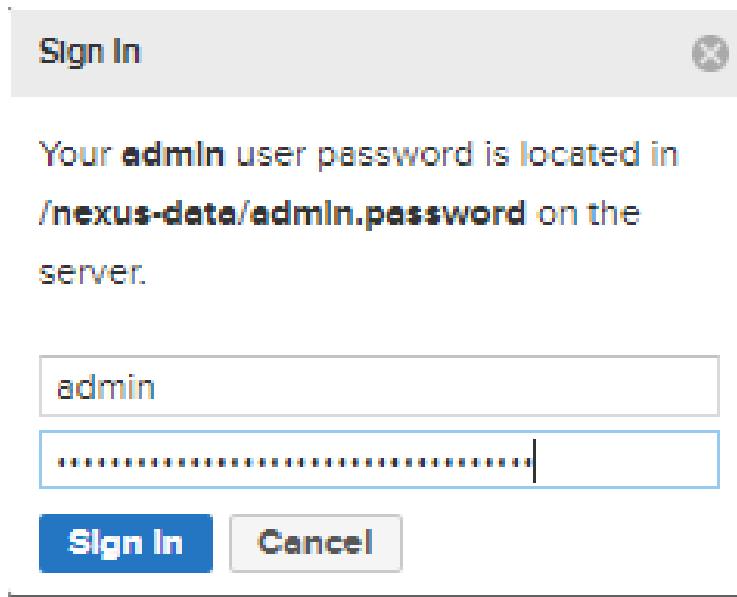
- Pour créer le conteneur:
➤ **docker run -d -p 8081:8081 -p 8083:8083 --name nexus sonatype/nexus3**
- Pour vérifier que le conteneur est fonctionnel :
➤ **docker ps**

```
vagrant@vagrant:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
4928db21bdcc        sonatype/nexus3    "/opt/sonatype/nexus..."   7 seconds ago      Up 5 seconds      0.0.0.0:808
```

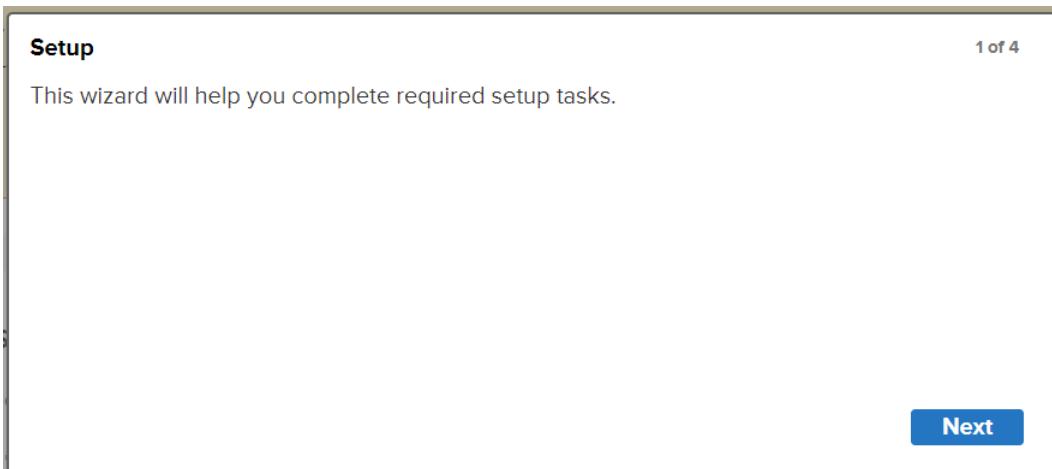
- En local (Windows), sur un navigateur, aller à <http://192.168.33.10:8081>
- Connecter avec les paramètres suivants :
- Username : admin
- Password : Accéder au conteneur (avec docker exec -i idconteneur), et lire le fichier mentionné dans l'interface d'authentification (avec cat) :



```
[vagrant@localhost ~]$ docker exec -i 18eefef10a94 cat /nexus-data/admin.password
6f808d07-59f2-48fc-b35b-984bcdedb6a3[vagrant@localhost ~]$
```



- Changer le mot de passe à nexus par exemple, et accepter les accès anonymes.



Please choose a password for the admin user

2 of 4

New password:

! This field is required

Confirm password:

[Back](#)

[Next](#)

Configure Anonymous Access

3 of 4

Enable anonymous access means that by default, users can search, browse and download components from repositories without credentials. Please **consider the security implications for your organization.**

Disable anonymous access should be chosen with care, as it **will require credentials for all** users and/or build tools.

[More information](#)

- Enable anonymous access
 Disable anonymous access

[Back](#)

[Next](#)

Complete

4 of 4

The setup tasks have been completed, enjoy using Nexus Repository Manager!

[Finish](#)

- Dans l'interface utilisateur Nexus, accédez à l'onglet "configuration" sur la gauche.
- Cliquez sur le bouton "Create Repository"

Sonatype Nexus Repository
OSS 3.63.0-01

Administration

- Repository
 - Repositories**
 - Blob Stores
 - Proprietary Repositories
 - Content Selectors
 - Cleanup Policies
 - Routing Rules
- Security
 - Privileges
 - Roles

Repositories Manage repositories

Create repository

Name ↑	Type	Format
maven-central	proxy	maven2
maven-public	group	maven2
maven-releases	hosted	maven2
maven-snapshots	hosted	maven2
nuget-group	group	nuget
nuget-hosted	hosted	nuget
nuget.org-proxy	proxy	nuget

- Sélectionnez "docker (hosted)"

Repository

- Repositories**
- Blob Stores
- Proprietary Repositories
- Content Selectors
- Cleanup Policies
- Routing Rules

Recipe ↑

- apt (hosted)
- apt (proxy)
- bower (group)
- bower (hosted)
- bower (proxy)
- cocoapods (proxy)
- conan (proxy)
- conda (proxy)
- docker (group)**
- docker (hosted)**
- docker (proxy)
- gitlfs (hosted)

- Remplissez les détails requis.

The screenshot shows the 'Repositories' section of the Nexus configuration. A new repository named 'docker-repo' is being created. The 'Online' checkbox is checked. Under 'HTTP', port 8083 is specified. The 'Enable Docker V1 API' checkbox is checked. Other sections like 'HTTPS', 'Allow anonymous docker pull', and 'Docker Registry API Support' are also visible.

- Cliquez sur le bouton "Create Repository".
- Accédez à la section "Administrer Jenkins"
- Accédez à la section "Credentials"
- Sélectionnez le domaine Global

Credentials					
T	P	Store	Domain	ID	Name
		System	(global)	git	mehdybenromdhane9@gmail.com/*****
		System	(global)	scanner	scanner

- Cliquez sur "Add Credentials".
- Choisissez le type d'informations d'identification approprié. Pour Nexus, vous pouvez utiliser "nom d'utilisateur et mot de passe"
- Remplissez le nom d'utilisateur et mot de passe de votre compte Nexus avec ID = nexus.
- Cliquez sur "Create" pour enregistrer les informations d'identification.

New credentials

Type: Nom d'utilisateur et mot de passe

Portée: Global (Jenkins, agents, items, etc...)

Nom d'utilisateur: admin

Treat username as secret

Mot de passe:

ID: nexus

Description:

Create

- Installer les plugins Docker et Docker pipeline
- Administrer Jenkins > Plugins > plugins disponibles

Jenkins

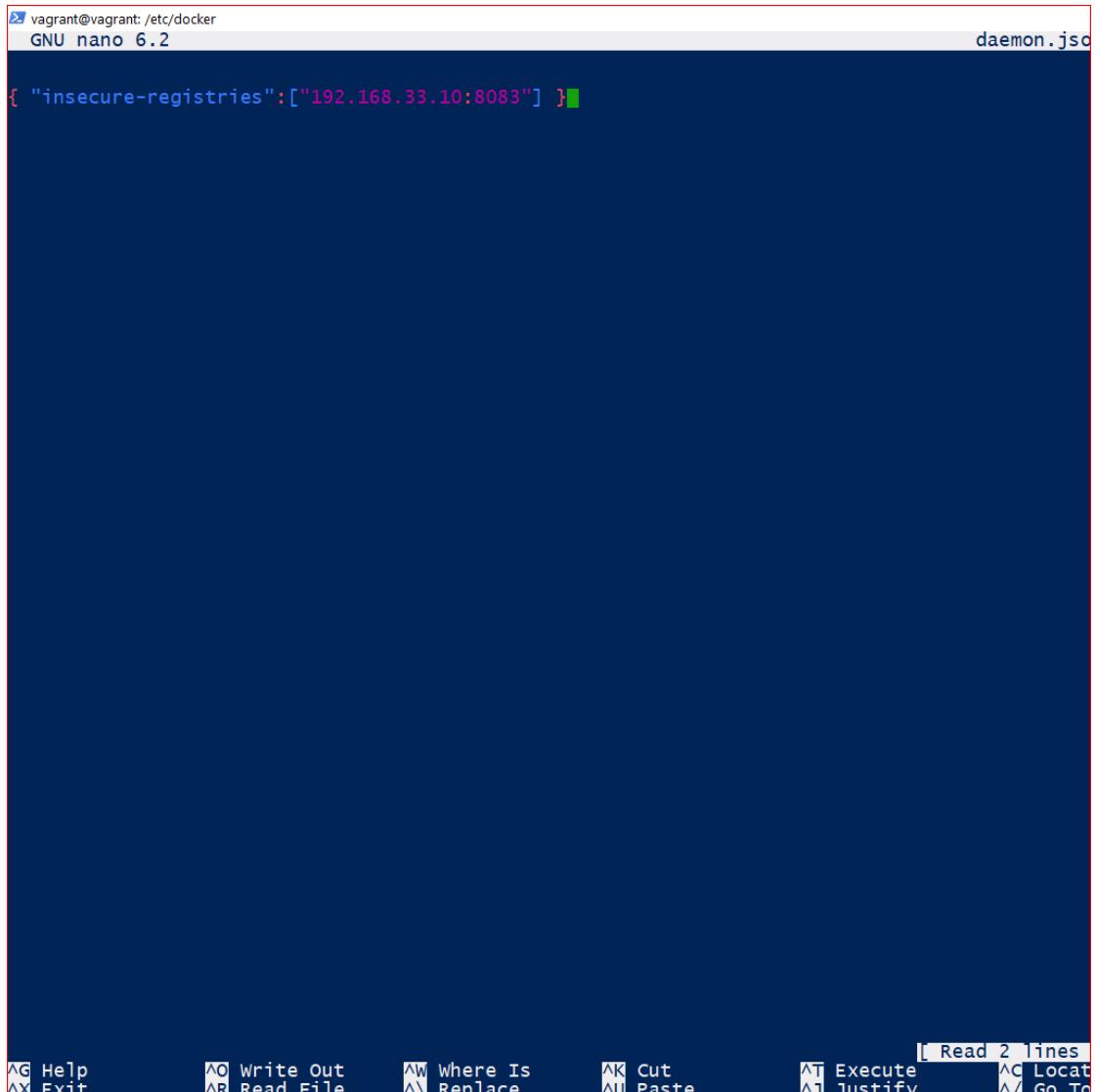
Tableau de bord > Administre Jenkins > Plugins

Plugins	Progression du téléchargement
Mises à jour	Préparation
Plugins disponibles	<ul style="list-style-type: none"> • Checking internet connectivity • Checking update center connectivity • Success
Plugins installés	<ul style="list-style-type: none"> Succès Succès Succès Succès Succès Success Succès Succès
Paramètres avancés	
Progression des téléchargements	
	<ul style="list-style-type: none"> → Revenir en haut de la page (vous pouvez commencer à utiliser les plugins installés dès maintenant) → <input type="checkbox"/> Redémarrer Jenkins quand l'installation est terminée et qu'aucun job n'est en cours

Créer le fichier daemon.json sous /etc/docker et ajouter le code suivant :

```
{ "insecure-registries":["192.168.33.10:8083"] }
```

- cd /etc/docker
- sudo nano daemon.json



```
vagrant@vagrant: /etc/docker
GNU nano 6.2                                         daemon.json

{ "insecure-registries": ["192.168.33.10:8083"] }

^G Help      ^O Write Out    ^W Where Is      ^K Cut          ^T Execute     ^C Locat
^X Exit      ^R Read File    ^\ Replace       ^U Paste        ^J Justify     ^/ Go To
                                         [ Read 2 lines ]
```

➤ CTRL + X
➤ Y

Pour vérifier

Exécuter la commande suivante :

```
➤ cat daemon.json
```

Maintenant il faut redémarrer docker avec :

```
➤ sudo systemctl restart docker
```

- Au niveau de fichier Jenkinsfile dans votre projet ajoutez le code suivant avant **stages** :

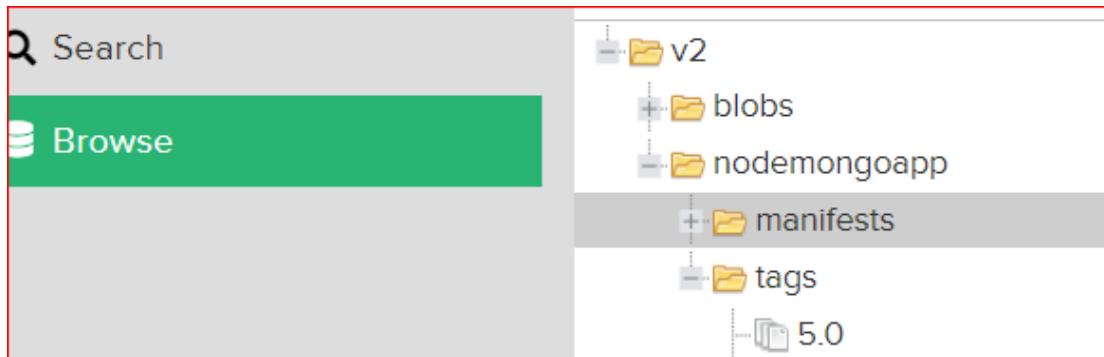
```
environment {  
    registryCredentials = "nexus"  
    registry = "192.168.33.10:8083"  
}
```

- Et après le stage **Building images** ajoutez le code suivant :

```
// Uploading Docker images into Nexus Registry  
stage('Deploy to Nexus') {  
    steps{  
        script {  
            docker.withRegistry("http://"+registry,  
registryCredentials ) {  
                sh('docker push $registry/nodemongoapp:5.0 ')  
            }  
        }  
    }  
}
```

- Push sur github et lancer le pipeline

Stage View



Ajoutez le code suivant (jenkinsfile) pour exécuter notre application avec docker compose :

```
stage('Run application') {
    steps{
        script {
            docker.withRegistry("http://"+registry, registryCredentials
) {

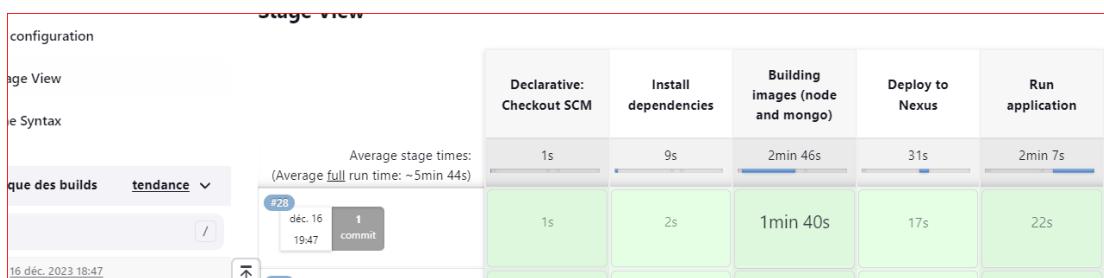
                sh('docker pull $registry/nodemongoapp:6.0')

                sh('docker-compose up -d')
            }
        }
    }
}
```

- sh 'docker pull \$registry/nodemongoapp:6.0' : Cette étape télécharge l'image Docker nodemongoapp:6.0 depuis le registre spécifié.(nexus)

- sh 'docker-compose up -d' : Cette étape démarre l'application définie dans votre fichier docker-compose.yml en mode détaché (-d). Cela signifie que les conteneurs s'exécuteront en arrière-plan.

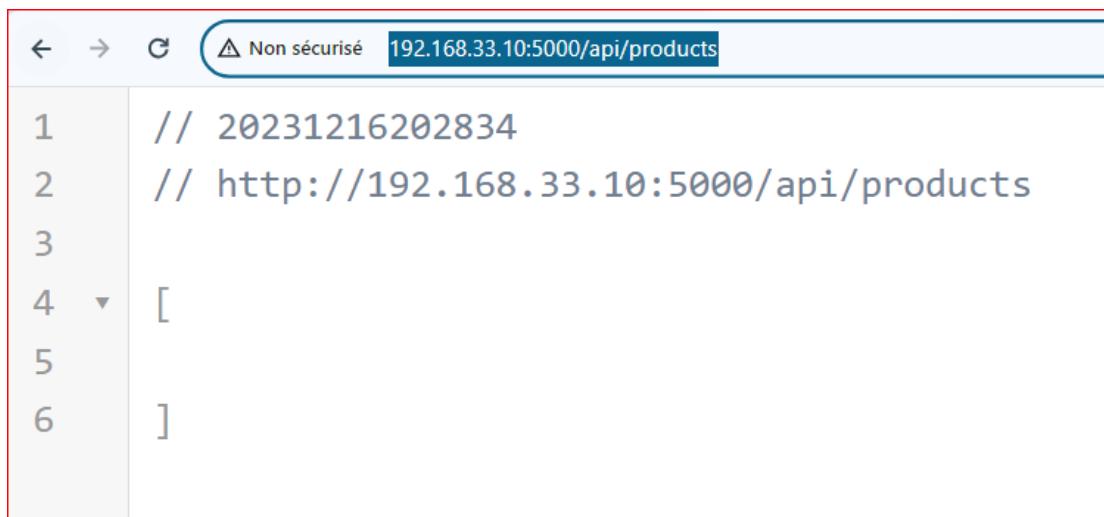
Push le code et lancez le pipeline



Naviguer sur <http://192.168.33.10:5000/api/>:



<http://192.168.33.10:5000/api/products>



et bien sûr vous pouvez tester l'application déployée avec postman.

h) Mise en place du Monitoring de l'outil Jenkins, avec Prometheus + Grafana :

- Aller dans "Administre Jenkins" puis "Gestion des plugins" et installer "Prometheus metrics". Ce plugin permettra d'exposer les métriques pour Prometheus sur le endpoint **/prometheus**



→ Jenkins doit être redémarré pour que la mise à jour soit effective.

- Création d'un conteneur Docker Prometheus.
- **sudo chmod 666 /var/run/docker.sock**
- **docker run -d --name prometheus -p 9090:9090 prom/prometheus**
- **docker exec -it prometheus sh**
- Puis dans le conteneur prometheus (8080 est le port pour accéder à Jenkins)
:

```
tee -a /etc/prometheus/prometheus.yml <<EOF
- job_name: jenkins
  metrics_path: /prometheus
  static_configs:
    - targets: ['172.17.0.1:8080']
EOF
```

- Vérifier le contenu du fichier créé avec
 - **cat /etc/prometheus/prometheus.yml** puis **exit**
 - **docker restart prometheus**

- Taper <http://192.168.33.10:9090/targets> dans votre navigateur

The screenshot shows the Prometheus Targets page. At the top, there are tabs for 'All', 'Unhealthy', and 'Collapse All'. A search bar is also present. Below the tabs, there are two sections: 'jenkins (1/1 up)' and 'prometheus (1/1 up)'. Each section contains a table with columns: Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error. The 'jenkins' section shows one endpoint: http://172.17.0.1:8080/prometheus, which is UP and was scraped 6.20s ago. The 'prometheus' section shows one endpoint: http://localhost:9090/metrics, which is UP and was scraped 15.65s ago.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://172.17.0.1:8080/prometheus	UP	instance="172.17.0.1:8080" job="jenkins"	6.20s ago	6.311ms	

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	15.65s ago	3.635ms	

- Taper <http://192.168.33.10:9090/metrics> dans votre navigateur

The screenshot shows the Prometheus Metrics page. The URL is 192.168.33.10:9090/metrics. The page displays a large list of Go metrics, each with a help string, type, and description. The metrics listed include go_gc_duration_seconds, go_gc_duration_seconds_summary, go_gc_duration_seconds{quantile="0"}, go_gc_duration_seconds{quantile="0.25"}, go_gc_duration_seconds{quantile="0.5"}, go_gc_duration_seconds{quantile="0.75"}, go_gc_duration_seconds{quantile="1"}, go_gc_duration_seconds_sum, go_gc_duration_seconds_count, go_goroutines, go_info, go_info{version="go1.21.5"}, go_memstats_alloc_bytes, go_memstats_alloc_bytes{quantile="0"}, go_memstats_alloc_bytes_total, go_memstats_alloc_bytes_total{quantile="0"}, go_memstats_buck_hash_sys_bytes, go_memstats_gc_sys_bytes, go_memstats_heap_alloc_bytes, go_memstats_heap_idle_bytes, go_memstats_heap_inuse_bytes, and go_memstats_heap_objects.

```

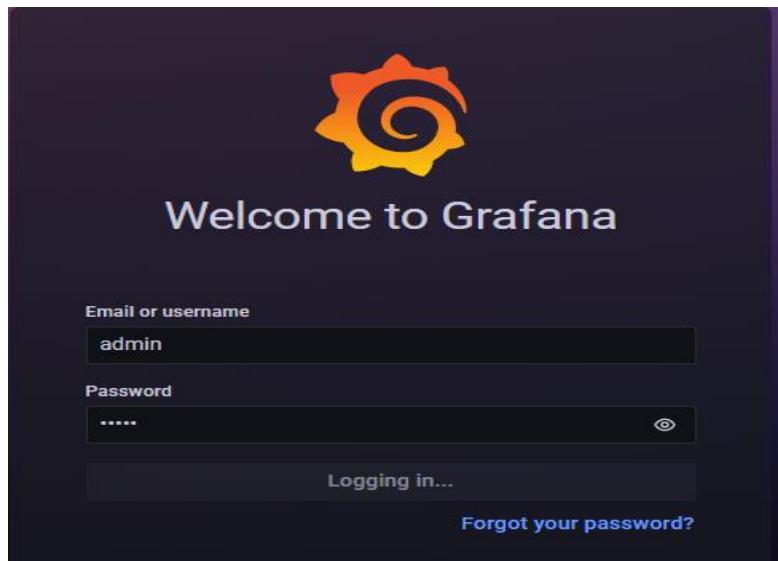
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.9269e-05
go_gc_duration_seconds{quantile="0.25"} 0.000337961
go_gc_duration_seconds{quantile="0.5"} 0.00052229
go_gc_duration_seconds{quantile="0.75"} 0.001408499
go_gc_duration_seconds{quantile="1"} 0.003303809
go_gc_duration_seconds_sum 0.0110463
go_gc_duration_seconds_count 11
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 36
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.21.5"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.1037488e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.01608808e+08
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.476198e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 177130
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 4.715784e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 2.1037488e+07
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 2.0455424e+07
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 2.4666112e+07
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 36

```

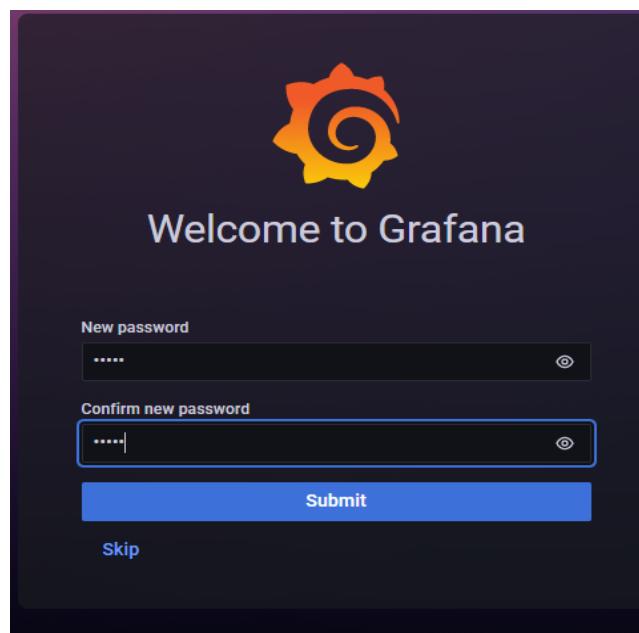
C'est très difficile d'exploiter et de comprendre ces données textuelles que **Prometheus** nous met à disposition.

Nous allons donc utiliser l'outil **Grafana** qui va récupérer ces données, les mettre en forme et les afficher graphiquement.

- Création d'un conteneur Docker Grafana.
- `docker run -d --name grafana -p 3000:3000 grafana/Grafana`
- Taper <http://192.168.33.10:3000> dans votre navigateur
- L'utilisateur et le mot de passe par défaut sont "admin/admin".



- Changer le mot de passe



- Ajouter la source des données → Prometheus

Welcome to Grafana

Need help? Documentation Tutorials Community Public

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

DATA SOURCES
Add your first data source

DASHBOARDS
Create your first dashboard

Add data source

Choose a data source type

Filter by name or type

Time series databases

- Prometheus** Open source time series database & alerting Core
- Graphite Open source time series database Core
- InfluxDB Open source time series database Core
- OpenTSDB Open source time series database Core

Récupérer l'adresse IP du conteneur docker Prometheus.

Connections

Add new connection

Data sources

Settings

Configure your Prometheus data source below

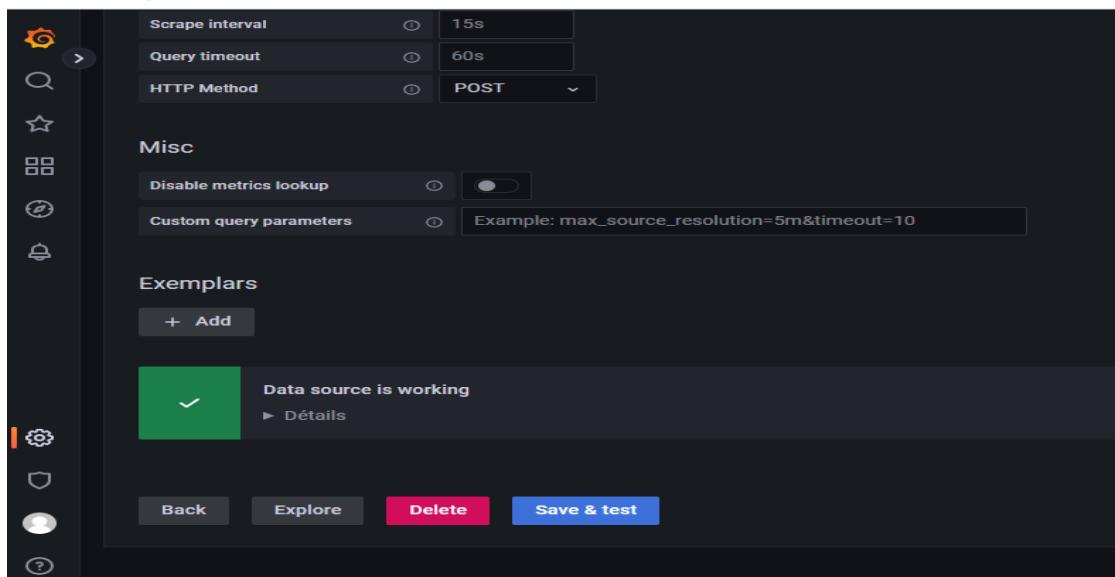
Name: Jenkins : Performance and Health Overview

Prometheus server URL * http://192.168.33.10:9090

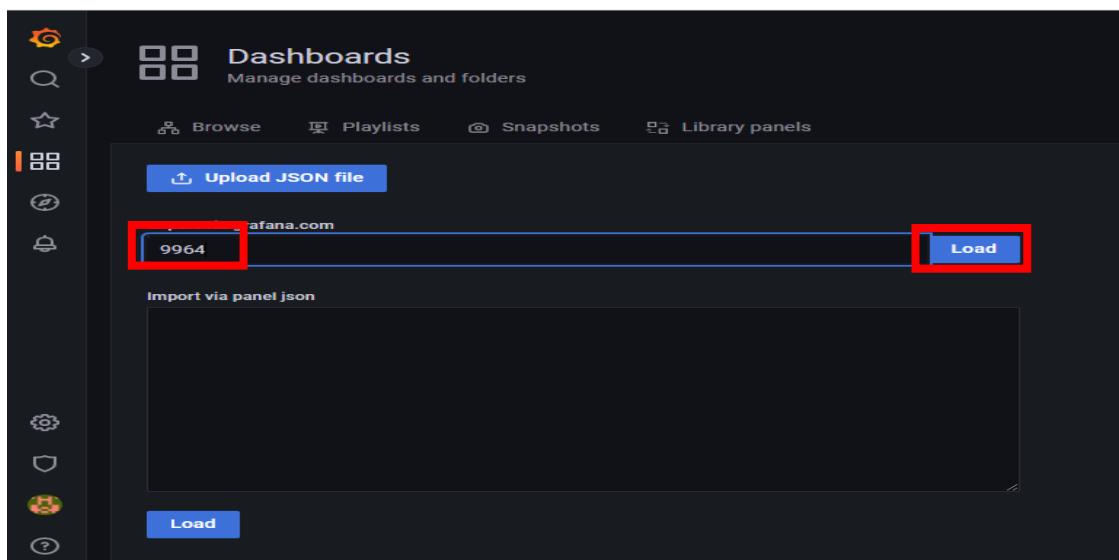
Connection

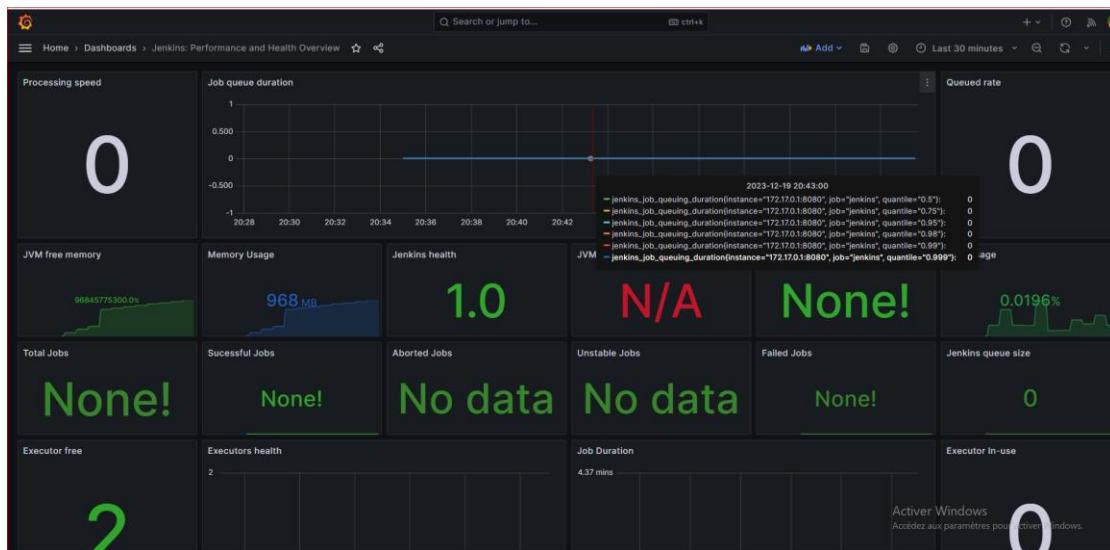
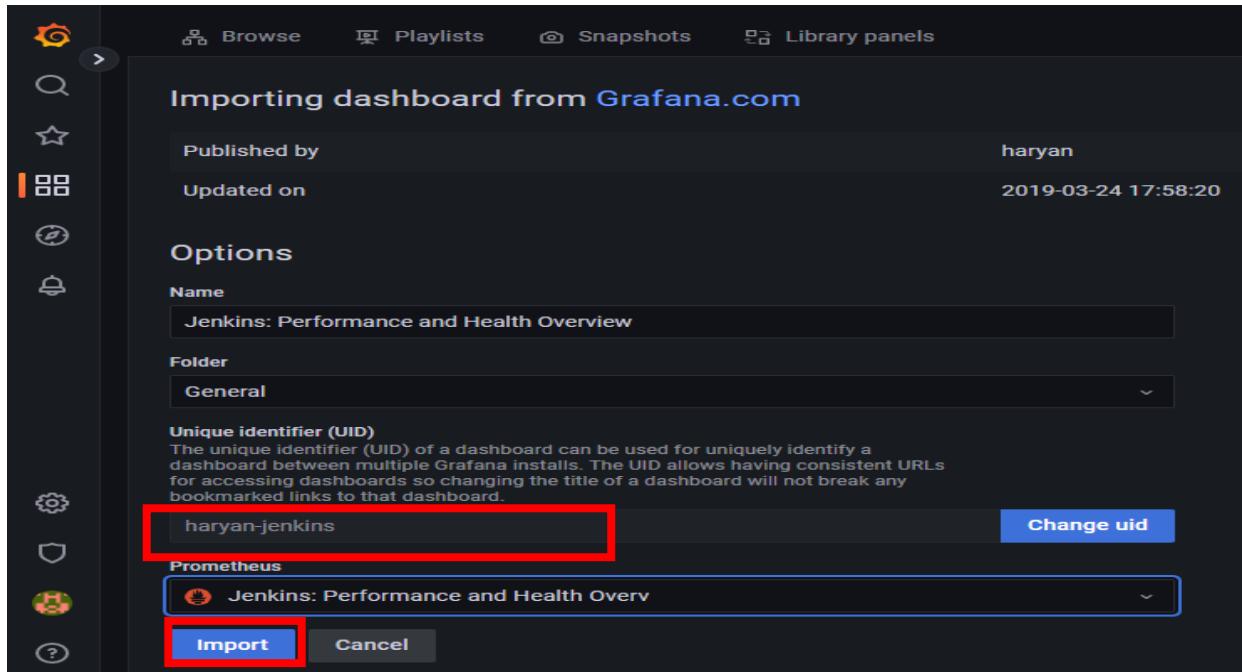
Authentication

- Cliquer Save & Test.

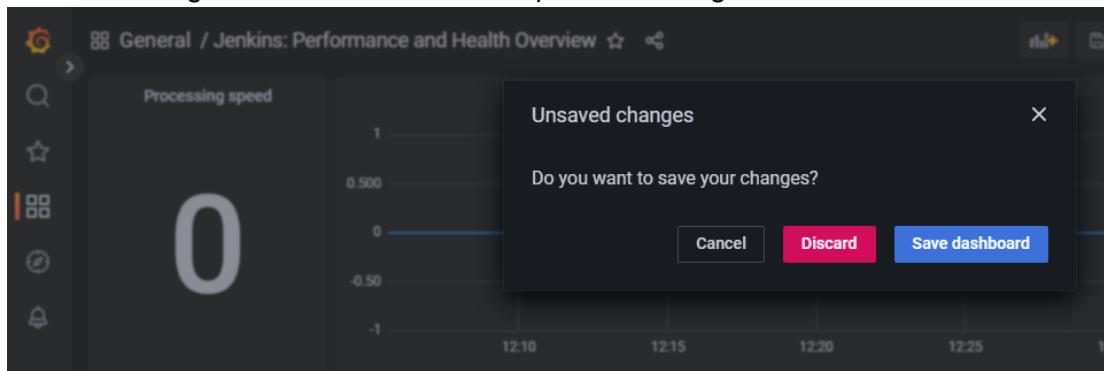


- Accéder à <http://192.168.33.10:3000/dashboard/import> et utiliser **9964** l'identifiant d'un template d'un dashboard.





- Sauvegarder le dashboard en cliquant sur le logo de Grafana



Jenkins

Tableau de bord > node-pipeline > main >

Status: **main** (green checkmark)

Full project name: node-pipeline/main

Stage View

Declarative: Checkout SCM Install dependencies Unit Test SonarQube Analysis Build application Building images (node and mongo) Deploy to Nexus Run application Run Prometheus Run Grafana

Average stage times:
(Average full run time: ~11min 49s)

#	décl. 19	No Changes
1	6s	46s
2	1min 7s	5min 21s
3	9s	1min 26s
4	14s	42s
5	7s	2s

Historique des builds: **tendance** / Filter...

Outil Capture d'écran

rechercher (CTRL+K)

admin

se déconnecter

À bientôt