



Support de TP

**DevOps**

TP 1 : VM Ubuntu

#- Si votre ordinateur tourne déjà sous un **OS Ubuntu**, alors vous n'avez rien à faire.

#- Si vous avez déjà une **VM Ubuntu installée** sur votre PC et opérationnelle, alors vous n'avez rien à faire.

#- Sinon, **Installer une VM Ubuntu avec Vagrant** en suivant les étapes ci-dessous :

#### **1-Installer VirtualBox**

#### **2- Installer Vagrant**

**3- Verifier l'installation** en tapant les commandes suivantes « vagrant –version » and « VBoxManage –version » .

**4-** Créer un dossier D:/Vagrant/Ubuntu. Se placer dans ce dossier et lancer un PowerShell ou CMD à partir de ce dossier

#### **5- Démarrer VirtualBox**

**5-** Lancer la commande **vagrant init** : Cette commande crée un Vagrantfile dans votre répertoire de projet

**6-Écraser le contenu du fichier Vagranfile créé, par le fichier se trouvant sur le Drive (DevOps).**

**7-Lancer la commande vagrant up** : Cette commande téléchargera la version Ubuntu spécifiée si elle n'est pas déjà mise en cache, puis démarrera la machine virtuelle.

**8-lancer la commande vagrant ssh** : Cette commande établira une connexion SSH à votre machine virtuelle Ubuntu en cours d'exécution

(Pour arrêter la VM :**exit** de la VM, puis **vagrant halt**)

#### **9- Installer l'éditeur nano**

- sudo apt install nano
- nano --version

Support de TP

**DevOps**

# TP 2 : Intégration continue

## SOMMAIRE

I.	Installation du JDK.....	2
II.	Installation de Maven.....	2
III.	Installation de Git.....	3
IV.	Installation de Jenkins .....	3
V.	Jobs Jenkins.....	7

# I. Installation du JDK

1. Se placer (avec un CMD ou un PowerShell) au niveau du dossier de votre VM Ubuntu lancez votre VM (vagrant up), et lancer un shell (vagrant ssh).

```
PS E:\Vagrant\Ubuntu> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
--> default: Checking if box 'bento/ubuntu-22.04' version '202508.03.0' is up to date...
--> default: Clearing any previously set forwarded ports...
--> default: Clearing any previously set network interfaces...
--> default: Preparing network interfaces based on configuration...
--> 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 minutes...
--> default: SSH address: 127.0.0.1:2222
--> default: SSH username: vagrant
--> default: SSH auth method: private key
--> default: Vagrant insecure key detected. Vagrant will automatically replace
--> default: this with a newly generated keypair for better security.
--> default: Inserting generated public key within guest...
--> default: Removing insecure key from the guest if it's present...
--> default: Key inserted! Disconnecting and reconnecting using new SSH key...
--> default: Machine booted and ready!
--> default: Checking for guest additions in VM...
--> default: The guest additions on this VM do not match the installed version of
--> default: VirtualBox! In most cases this is fine, but in rare cases it can
--> default: prevent things such as shared folders from working properly. If you see
--> default: shared folder errors, please make sure the guest additions within the
--> default: virtual machine match the version of VirtualBox you have installed on
--> default: your host and reload your VM.
--> default: Guest Additions Version: 7.1.12
--> default: VirtualBox Version: 7.2
--> default: Configuring and enabling network interfaces...
--> default: Mounting shared folders...
--> default: E:/Vagrant -> /vagrant
PS E:\Vagrant\Ubuntu>
PS E:\Vagrant\Ubuntu> vagrant ssh
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-144-generic x86_64)
```

2. Installer la OpenJDK :

- sudo apt update
- sudo apt install openjdk-21-jdk

3. Pour vérifier que le JDK 21 est bien installé, exécuter la commande suivante:

```
vagrant@vagrant:~$ java -version
openjdk version "21.0.8" 2025-07-15
OpenJDK Runtime Environment (build 21.0.8+9-Ubuntu-0ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 21.0.8+9-Ubuntu-0ubuntu122.04.1, mixed mode, sharing)
vagrant@vagrant:~$
```

# II. Installation de Maven

4. Pour installer Maven, vous devez lancer le terminal et exécuter les commandes suivantes:

```
vagrant@vagrant: ~
vagrant@vagrant:~$ sudo apt install maven
```

5. Pour vérifier que le Maven est bien installé :

```
vagrant@vagrant: ~  
vagrant@vagrant:~$ mvn -version  
Apache Maven 3.6.3  
Maven home: /usr/share/maven  
Java version: 21.0.8, vendor: Ubuntu, runtime: /usr/lib/jvm/java-21-openjdk-amd64  
Default locale: en_US, platform encoding: UTF-8  
OS name: "linux", version: "5.15.0-144-generic", arch: "amd64", family: "unix"  
vagrant@vagrant:~$
```

## III. Installation de Git

6. Vérifier si **Git** est déjà installé en utilisant la commande « `git --version` ».

```
vagrant@vagrant: ~  
vagrant@vagrant:~$ git --version  
git version 2.34.1  
vagrant@vagrant:~$
```

7. Installer Git si ce n'est pas installé

- `sudo apt install git`

## IV. Installation de Jenkins

8. Pour installer Jenkins, vous devez exécuter les 4 commandes suivantes (faites juste un copier-coller de chaque commande) :

- `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-stable binary/ | sudo tee \ /etc/apt/sources.list.d/jenkins.list > /dev/null`
- `sudo apt-get update`
- `sudo apt-get install jenkins`

9. En utilisant la commande appropriée vérifier que Jenkins est en cours d'exécution.

- Pour lancer Jenkins (`start`), pour lancer jenkins automatiquement comme service au prochains démarriages (`enable`)
  - `sudo systemctl start jenkins.service`
- Pour vérifier l'installation de Jenkins : `status`
  - `sudo systemctl status jenkins`

```
vagrant@vagrant:~$ sudo systemctl start 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 Fri 2025-08-22 14:42:21 UTC; 2min 6s ago
       Main PID: 6554 (java)
          Tasks: 43 (limit: 5615)
        Memory: 558.6M
         CPU: 1min 5.457s
        CGroup: /system.slice/jenkins.service
                  └─6554 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Aug 22 14:42:04 vagrant jenkins[6554]: bc1a10cff1fd4900a61d718e40b3ca2e
Aug 22 14:42:04 vagrant jenkins[6554]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 22 14:42:04 vagrant jenkins[6554]: ****
Aug 22 14:42:04 vagrant jenkins[6554]: ****
Aug 22 14:42:04 vagrant jenkins[6554]: ****
Aug 22 14:42:21 vagrant jenkins[6554]: 2025-08-22 14:42:21.693+0000 [id=39]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed initialization
Aug 22 14:42:21 vagrant jenkins[6554]: 2025-08-22 14:42:21.769+0000 [id=30]      INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Aug 22 14:42:21 vagrant systemd[1]: Started Jenkins Continuous Integration Server.
Aug 22 14:42:23 vagrant jenkins[6554]: 2025-08-22 14:42:23.531+0000 [id=56]      INFO      h.m.DownloadService$Downloadable#load: Obtained the updated configuration
Aug 22 14:42:23 vagrant jenkins[6554]: 2025-08-22 14:42:23.537+0000 [id=56]      INFO      hudson.util.Retriger#start: Performed the action check update
lines 1-20/20 (END)
```

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

```
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:d1:a4:ba brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    altname enp0s3
      inet 10.0.2.15/24 metric 100 brd 10.0.2.255 scope global dynamic eth0
        valid_lft 73448sec preferred_lft 73448sec
        inet6 fd17:625c:f037:2:a00:27ff:fed1:a4a/64 scope global dynamic mngtmpaddr noprefixroute
          valid_lft 86221sec preferred_lft 14221sec
        inet6 fe80::a00:27ff:fed1:a4a/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:d5:09:6d brd ff:ff:ff:ff:ff:ff
    altname enp0s8
    altname enp0s8
      inet 192.168.50.4/24 brd 192.168.50.255 scope global eth1
        valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fed5:96d/64 scope link
          valid_lft forever preferred_lft forever
vagrant@vagrant:~$
```

Non sécurisé 192.168.50.4:8080/login?from=%2F

Démarrage

## Débloquer Jenkins

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

`/var/lib/jenkins/secrets/initialAdminPassword`

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

Mot de passe administrateur

.....

Continuer

11. 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@localhost ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
37a196e129c64114ae89f0e0b9df83e6
```

Mot de passe administrateur

.....

12. Installer les plugins suggérés de la chaîne de l'intégration continue.

curisé 192.168.50.4:8080

Démarrage

## Personnaliser Jenkins

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

Installer les plugins suggérés

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

Sélectionner les plugins à installer

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

Jenkins 2.516.2

### 13. Configurer les outils.

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

## Créer le 1er utilisateur Administrateur

Nom d'utilisateur:

Mot de passe:

Confirmation du mot de passe:

Nom complet:

Adresse courriel:

3.2 Continuer en tant qu'Administrateur (highlighted) Sauver et continuer

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

The screenshot shows the Jenkins Administration interface. At the top, there's a navigation bar with links like 'Statut', 'Constructions', 'Mes vues', 'Account', 'Appearance', 'Preferences', 'Security', 'Experiments', and 'Identifiants'. On the right, a sidebar also lists these same items. Below the sidebar, a 'Sign out' link is visible. The main content area contains a 'Mot de passe' section with fields for 'Mot de passe:' and 'Confirmer le mot de passe:', both containing placeholder text '.....'. At the bottom, there's a 'Fin de session' button with a 'Termine toutes les sessions' link and a 'Save' button.

## 14. Configurer de Jenkins – Plugins

- Pour installer des plugins, il suffit d'accéder à la fenêtre «Administrer Jenkins/ Plugins»

The screenshot shows the Jenkins Plugin Manager page. At the top, there's a navigation bar with links like "Importez les favoris", "Postman - DSI", "Office Esprit", "Blackboard Esprit", "Accueil - OneDrive", "Équipe", and "admin - Jenkins". Below the navigation is a search bar labeled "Rechercher les plugins disponibles". A large table lists several plugins:

Installer	Nom	Publié	Santé
<input type="checkbox"/>	Pipeline Graph Analysis 241.vc3d4fb_b_2582	Il y a 2 mo. 22 j	<span style="color: green;">100</span>
<input type="checkbox"/>	PAM Authentication 1.12	Il y a 5 mo. 21 j	<span style="color: green;">97</span>
<input type="checkbox"/>	JavaMail API 1.6.2-11	Il y a 6 mo. 0 j	<span style="color: green;">96</span>
<input type="checkbox"/>	Command Agent Launcher 123.v37cfcd92ef67	Il y a 4 mo. 14 j	<span style="color: green;">100</span>

On the left sidebar, there are tabs: "Mises à jour", "Plugins disponibles" (which is selected), "Plugins installés", "Paramètres avancés", and "Progression des téléchargements".

# V. Jobs Jenkins

- Pour créer notre chaîne d'intégration continue, on va installer les plugins suivants dans Jenkins  
(Installer ces plugins sans redémarrer, puis redémarrer Jenkins à la fin) :
  - Git plugin (normalement déjà installé, mais vérifier)
  - Maven Integration
  - Sonargraph Integration
  - SonarQube Scanner
  - Redémarrer Jenkins suite à l'installation des plugins
    - sudo systemctl restart jenkins.service

```
vagrant@vagrant:~$ sudo systemctl restart jenkins.service
vagrant@vagrant:~$
```

## 15. Configuration de Jenkins – Outils

- Pour configurer les outils, il faut accéder à la fenêtre «Administrer Jenkins/ Tools»

The screenshot shows the Jenkins Administration Tools page. At the top, there's a message: "Une construction sur le noeud principal peut engendrer des problèmes de sécurité. Vous devriez configurer des builds distribués. Consulter la documentation." Below the message are three buttons: "Set up agent", "Set up cloud", and "Dismiss".

The main area is titled "Configuration du système" and contains several sections:

- System**: Configurer les paramètres généraux et les chemins de fichiers.
- Tools** (highlighted with a red box): Configurer les outils, leur localisation et les installateurs automatiques.
- Clouds**: Ajouter, supprimer et configurer les instances de cloud afin de provisionner les agents à la demande.
- Plugins**: Ajouter, supprimer, activer ou désactiver des plugins qui peuvent étendre les fonctionnalités de Jenkins.
- Nodes**: Ajouter, supprimer, contrôler et montrer les divers nœuds que Jenkins utilise pour exécuter les jobs.
- Apparence générale**: Configurer l'apparence générale de Jenkins.

## 1- JDK

Fournisseur de réglages globaux par défaut  
Utiliser les réglages globaux Maven par défaut

Installations Sonargraph Build  
Ajouter Sonargraph Build

Installations JDK  
Ajouter JDK

Git installations

Enregistrer      Appliquer

Installations JDK

Ajouter JDK

**JDK**

Nom  
JAVA\_HOME

JAVA\_HOME  
/usr/lib/jvm/java-21-openjdk-amd64/

Install automatically ?  
Ajouter un installateur ▾

## 2- Maven

**Maven**

Nom  
M2\_HOME

MAVEN\_HOME  
/usr/share/maven

Install automatically ?

Ajouter Maven

Enregistrer      Appliquer

### 3- Git :

- Création Token Git

The screenshot shows the GitHub Home page. At the top, there's a search bar and a sidebar on the right with a user profile for "tarek-ayari". Below the sidebar, a "Latest changes" section lists recent updates. The main area displays trending repositories, including "github/spec-kit" and "agentscope-ai/agentscope". A "Feed" section follows, showing more repository cards.

The screenshot shows the GitHub Profile settings page at <https://github.com/settings/profile>. The left sidebar lists various settings categories like "Emails", "Bio", "Pronouns", "URL", "Social accounts", "Company", "Location", and "ORCID iD". The "Developer settings" link is highlighted with a red box. On the right, there are fields for "Bio" (with placeholder "Tell us a little bit about yourself"), "Pronouns" (set to "Don't specify"), and "Social accounts" (with four "Link to social profile" fields). The "Company" field is empty with a placeholder "You can @mention your company's GitHub organization to link it." The "Location" field is also empty. A checkbox for "Display current local time" is present with a note: "Other users will see the time difference from their local time." A "Connect your ORCID iD" button is at the bottom.

://github.com/settings/personal-access-tokens

... Postman- DSL Office Esprit Blackboard Esprit Accueil - OneDrive Équipe Devops

Developer Settings Type [ ] to search

Fine-grained personal access tokens Generate new token

A GitHub Apps OAuth Apps Personal access tokens ^

Personal access tokens Fine-grained tokens Tokens (classic)

These are fine-grained, repository-scoped tokens suitable for personal API use and for using Git over HTTPS.

devopsgit Never used • Expires on Fri, Sep 26 2025 Delete

© 2025 GitHub, Inc. Terms Privacy Security Status GitHub Community Docs Contact Manage cookies Do not share my personal information

**Expiration**

30 days (Oct 05, 2025) The token will expire on the selected date

**Repository access**

**Public repositories**  
Read-only access to public repositories.

**All repositories**  
This applies to all current and future repositories you own. Also includes public repositories (read-only).

**Only select repositories**  
Select at least one repository. Max 50 repositories. Also includes public repositories (read-only).

**Permissions**

Choose the minimal permissions necessary for your needs. [Learn more about permissions.](#)

Account 0 + Add permissions

No account permissions added yet

User permissions permit access to resources under your personal GitHub account.

Generate token Cancel

This token will be ready for use immediately.

## Espace jenkins :

The screenshot shows the Jenkins Administration interface. The 'Credentials' section is highlighted with a red box. Other sections visible include 'Nodes', 'Clouds', 'Apparence générale', 'Security', and 'Users'.

### - Ajouter le token Git

The screenshot shows the 'Update credentials' form for adding a GitHub credential. The 'Nom d'utilisateur' field contains 'tarek-ayari' and the 'Mot de passe' field contains 'Concealed'. Both fields are highlighted with a red box. The 'ID' field contains 'jenkins-github-https-cred' and the 'Description' field contains 'Access Github'. A 'Sauvegarder' button is at the bottom.

Jenkins / Administrer Jenkins / Identifiants / System / Identifiants globaux (illimité) / tarek-ayari/\*\*\*\*\*\*\*\* (Access Github)

mettre à jour

upprimer

éplacer

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

Nom d'utilisateur ? tarek-ayari

Treat username as secret ?

Mot de passe ? Concealed

ID ? jenkins-github-https-cred

Description ? Access Github

Sauvegarder

Change Password

## 16. Configuration d'un Job avec Jenkins

- Les Jobs ou tâches représentent le cœur du processus de « build » dans Jenkins.
- Un Job Jenkins est représenté et composé de plusieurs étapes de build.
- Tout projet sous Jenkins passe par les 3 étapes suivantes:
  - ✓ Création du job
  - ✓ Configuration du job (configuration des étapes du build)
  - ✓ Lancement du build
- Nous allons nous intéresser aux Jobs Jenkins de type Pipeline : Il s'agit de définir la configuration de notre Job grâce à un script (basé sur le langage Groovy).

## 17. Configuration d'un Job avec Jenkins – Pipeline

- Pour montrer le fonctionnement des Jobs Jenkins de type pipeline, nous allons en créer un, qui s'appelle «FirstPipeline» par exemple:

Nouveau Item

Saisissez un nom

FirstPipeline

Select an item type

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

OK

**Etape 1 :** 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 «Identifiants» pour que Jenkins soit capable de récupérer le code de Git.

**Attention :** Récupérer l'URL d'un projet de Git (Projet Java), je vous donnerai des exemples de repos, car Github peut nous bloquer si on utilise tous le même repo.

## Etape 1 : Récupération du code du git (main ou master selon github)

Definition

Pipeline script

```
Script ?  
1 v pipeline {  
2     agent any  
3  
4 v     stages {  
5 v         stage('GIT') {  
6 v             steps {  
7  
8                 git branch: 'main',  
9                 changelog: false,  
10                credentialsId: 'jenkins-github',  
11                url: 'https://github.com/tarek-ayari/devops.git'  
12            }  
13        }  
14    }  
15}
```

Use Groovy Sandbox ?

## Etape 2 : Exécuter une commande Maven (mvn compile par exemple)

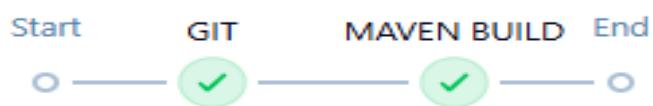
```
stage('MAVEN Build') {  
    steps {  
        // Use the 'mvn' command to compile, test, and package the project  
        // '-B' for non-interactive mode, '-DskipTests' to skip tests (remove if you want to run tests)  
        sh 'mvn -B -DskipTests clean'  
    }  
}
```

### 18. Build d'un Job avec Jenkins – Pipeline

- Pour lancer le Build du Job:

The screenshot shows the Jenkins dashboard with the 'FirstPipeline' job selected. The job status is green with a checkmark icon. A button labeled 'Lancer un build' is highlighted with a red box. To the right, there is a section titled 'Liens permanents' (Permanent links) containing a bulleted list of recent builds. At the bottom, there is a 'Builds' table with one entry.

Builds	...
...	...



Support de TP

**DevOps**

# Chapitre 3 : Livraison continue

## SOMMAIRE

I.	Installation de Docker .....	2
II.	Création d'une image.....	4
III.	Dépôt DockerHub .....	7

# I. Installation de Docker

1. Démarrer la machine virtuelle : se placer au niveau dossier de la VM et lancer là, puis lancer un client ssh:

```
vagrant@vagrant: ~$ E:\Vagrant\Ubuntu> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'bento/ubuntu-22.04' version '202508.03.0' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
```

```
vagrant@vagrant: ~$ E:\Vagrant\Ubuntu> vagrant ssh
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-144-generic x86_64)

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

2. Installer Docker à travers les 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 focal stable"
- sudo apt install docker-ce -y
- sudo systemctl status docker

3. Vérifier que docker est déjà lancé : 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 Wed 2025-08-27 13:32:52 UTC; 5min ago
     Tasks: 9
    Memory: 94.4M
      CPU: 2.039s
   CGroup: /system.slice/docker.service
           └─853 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

- CTRL C pour sortir de la commande de status.

4. Attribution des droits d'accès en lecture/écriture / exécution pour le user « vagrant », son groupe et tout autre utilisateur

- sudo chmod 666 /var/run/docker.sock
- 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
  - ✓ docker run hello-world

```
vagrant@vagrant:~$ vagrant@vagrant:~$ docker -v
Docker version 28.1.1, build 4eba377
vagrant@vagrant:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:a0dfb02aac212703bfcb339d77d47ec32c8706ff250850ecc0e19c8737b18567
Status: Downloaded newer image for hello-world:latest
```

```
vagrant@vagrant:~$ sudo chmod 666 /var/run/docker.sock
vagrant@vagrant:~$ docker run hello-world

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/
```

## II. Création d'une image

5. Au niveau du dossier utilisateur, créer un dossier nommé **docker** et créer un docker file.

6. Configurer une image qui se base sur l'image **alpine** : Une distribution Linux légère qui maintient la taille de votre conteneur petite.

```
vagrant@vagrant:~$ mkdir docker
vagrant@vagrant:~$ vi Dockerfile
vagrant@vagrant:~$ vagrant@vagrant:~$ vi Dockerfile
vagrant@vagrant:~$
```

```
vagrant@vagrant: ~
  GNU nano 6.2
FROM alpine

RUN apk add openjdk21
EXPOSE 80
CMD "java -version"
```

7. Dans le nom de l'image à créer ,il faut préciser votre login à dockerhub pour pouvoir envoyer ton image sur DockerHub (tarekayari1/alpine).

The screenshot shows the DockerHub search interface with the query 'alpine' entered. The search results page displays several Docker images related to Alpine Linux. The top result is the 'alpine' image from the 'DOCKER OFFICIAL IMAGE' repository, which is highlighted with a red border. This image has 1B+ downloads and 9.2K stars. Below it are other popular Alpine-related images, such as 'alpinelinux/alpine-gitlab-ci' and 'alpinelinux/docker-cli', both of which are sponsored OSS projects.

- A partir de l'image officielle de alpine (sur DockerHub), **on va créer notre propre image alpine:1.0.0 «customisée» avec java21**.

```
vagrant@vagrant:~/docker$ vi Dockerfile
vagrant@vagrant:~/docker$ docker build -t tarekayari1/alpine:1.0.0 .
[+] Building 77.6s (6/6) FINISHED                                            docker:default
    => [internal] load build definition from Dockerfile                      0.0s
    => => transferring dockerfile: 96B                                         0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 7) 0.0s
=> [internal] load metadata for docker.io/library/alpine:latest           0.0s
    => [internal] load .dockerignore                                         0.0s
    => => transferring context: 28                                         0.0s
=> CACHED [1/2] FROM docker.io/library/alpine:latest                      0.0s
=> [2/2] RUN apk add openjdk21                                              74.3s
=> exporting to image                                                       3.1s
=> => exporting layers                                                       3.0s
=> => writing image sha256:522ce8e2a6e729dba228520faf8a51dec97c6a946fd268914b4cab4938cb5bf 0.0s
=> => naming to docker.io/tarekayari1/alpine:1.0.0                         0.0s

1 warning found (use docker --debug to expand):
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 7)
vagrant@vagrant:~/docker$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
tarekayari1/alpine  1.0.0   522ce8e2a6e7  23 seconds ago  325MB
hello-world         latest   1b44b5a3e06a  2 weeks ago   10.1kB
alpine              latest   9234e8fb04c4  6 weeks ago   8.31MB
vagrant@vagrant:~/docker$
```

8. Télécharger une image d'un registre : docker pull «nom\_image»:«version», Si on ne spécifie pas la version, docker télécharge la dernière version

```
vagrant@vagrant:~/docker$ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
Digest: sha256:4bcff63911fcb4448bd4fdacec207030997caf25e9bea4045fa6c8c44de311d1
Status: Image is up to date for alpine:latest
docker.io/library/alpine:latest
vagrant@vagrant:~/docker$
```

9. Créer un conteneur : docker run «nom de l'image» . Si docker ne trouve pas l'image ,il la télécharge de Dockerhub et après il crée le conteneur.

```
vagrant@vagrant:~/docker$ vagrant@vagrant:~/docker$ docker run postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
396b1da763e6: Pull complete
f5465e2fc020: Pull complete
c166c949e1c3: Pull complete
7fa725c973af: Pull complete
1f6dfcaad4e9: Pull complete
b7a79609094c: Pull complete
901a9540064a: Pull complete
985f0a899c07: Pull complete
5d91a345d79a: Pull complete
f7f2afaalb41: Pull complete
36b4e7f51364: Pull complete
85558a0233ea: Pull complete
be9fdbdb096: Pull complete
ae28e2b99a62: Pull complete
Digest: sha256:29e0bb09c8e7e7fc265ea9f4367de9622e55bae6b0b97e7cce740c2d63c2ebc0
Status: Downloaded newer image for postgres:latest
Error: Database is uninitialized and superuser password is not specified.
You must specify POSTGRES_PASSWORD to a non-empty value for the
superuser. For example, "-e POSTGRES_PASSWORD=password" on "docker run".

You may also use "POSTGRES_HOST_AUTH_METHOD=trust" to allow all
connections without a password. This is *not* recommended.

See PostgreSQL documentation about "trust":
https://www.postgresql.org/docs/current/auth-trust.html
```

## 10. Vérifier que l'image a été créée localement : docker images

```
vagrant@vagrant:~/docker$
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tarekayari1/alpine	1.0.0	522ce8e2a6e7	9 minutes ago	325MB
postgres	latest	ca95f67fffb26	12 days ago	454MB
hello-world	latest	1b44b5a3e06a	2 weeks ago	10.1kB
alpine	latest	9234e8fb04c4	6 weeks ago	8.31MB

## 11. Récupérer la liste des conteneurs qui sont en exécution : docker ps -a

```
vagrant@vagrant:~/docker$
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
c679dfbde3ef	postgres	"docker-entrypoint.s..."	2 minutes ago	Exited (1) 2 minutes ago		jovial_mendel
bd6bfff59db1b	hello-world	"/hello"	57 minutes ago	Exited (0) 57 minutes ago		cranky_allen
765c39d5033	hello-world	"/hello"	58 minutes ago	Exited (0) 58 minutes ago		boring_chaum

- Pourquoi one ne voit pas de conteneur alpine:1.0.0?

## 12. Lancer un conteneur et le laisser tourner en background (avec l'option -d : docker run -itd ) pour pouvoir y accéder plus tard.

- docker run -itd tarekayari1/alpine:1.0.0 /bin/sh

```
vagrant@vagrant:~/docker$
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8683814ce05b	tarekayari1/alpine:1.0.0	"/bin/sh"	8 seconds ago	Up 7 seconds	80/tcp	friendly_dewdney

## 13. Pour accéder au conteneur déjà lancé (avec docker exec) / Ctrl-D pour sortir

```
vagrant@vagrant:~/docker$
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
8683814ce05b	tarekayari1/alpine:1.0.0	"/bin/sh"	8 seconds ago	Up 7 seconds	80/tcp	friendly_dewdney

```
/ #  
/ # pwd  
/ / # ls  
bin dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var  
/ #
```

## 14. Supprimer une image : docker image rm «nomimage» --force.

```
vagrant@vagrant:~/docker$
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tarekayari1/alpine	1.0.0	522ce8e2a6e7	25 minutes ago	325MB
postgres	latest	ca95f67fffb26	12 days ago	454MB
hello-world	latest	1b44b5a3e06a	2 weeks ago	10.1kB
alpine	latest	9234e8fb04c4	6 weeks ago	8.31MB

```
vagrant@vagrant:~/docker$ docker rmi hello-world --force  
Untagged: hello-world:latest  
Untagged: hello-world@sha256:a0dfb02aac212703bfcbb339d77d47ec32c8706ff250850ecc0e19c8737b18567  
Deleted: sha256:1b44b5a3e06a9aae883e7bf245c100be0bb81a0e01b32de604f3ac44711634  
vagrant@vagrant:~/docker$ docker images  
REPOSITORY
```

15. Supprimer un conteneur : docker remove «Id du conteneur».

```
vagrant@vagrant:~/docker$ vagrant@vagrant:~/docker$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
8683814ce05b tarekayari1/alpine:1.0.0 "/bin/sh" 16 minutes ago Up 16 minutes 80/tcp friendly_dewdney
c679dfbde3ef postgres "docker-entrypoint.s..." 21 minutes ago Exited (1) 21 minutes ago jovial_mendel
bd6bff59db1b 1b44b5a3e06a "/hello" About an hour ago Exited (0) About an hour ago cranky_allen
765c39d54033 1b44b5a3e06a "/hello" About an hour ago Exited (0) About an hour ago boring_chaum
vagrant@vagrant:~/docker$ docker remove 86838
Error response from daemon: cannot remove container "/friendly_dewdney": container is running: stop the container before removing or force remove
vagrant@vagrant:~/docker$ docker stop 86838
86838
vagrant@vagrant:~/docker$ docker remove 86838
vagrant@vagrant:~/docker$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c679dfbde3ef postgres "docker-entrypoint.s..." 23 minutes ago Exited (1) 23 minutes ago jovial_mendel
bd6bff59db1b 1b44b5a3e06a "/hello" About an hour ago Exited (0) About an hour ago cranky_allen
765c39d54033 1b44b5a3e06a "/hello" About an hour ago Exited (0) About an hour ago boring_chaum
vagrant@vagrant:~/docker$
```

## III. Dépôt DockerHub

16. Se connecter au site **DockerHub** (<https://hub.docker.com>) . Si vous n'avez pas de compte en créer un.

17. s'y connecter (dockerlogin).

```
vagrant@vagrant:~/docker$ docker login
USING WEB-BASED LOGIN
Info → To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: XSTJ-PDGH
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser...

WARNING! Your credentials are stored unencrypted in '/home/vagrant/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
vagrant@vagrant:~/docker$
```

18. Envoyer une image que vous avez créé sur dockerhub (dockerpush).

```
vagrant@vagrant:~/docker$ vagrant@vagrant:~/docker$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
tarekayari1/alpine 1.0.0 522ce8e2a6e7 37 minutes ago 325MB
postgres latest ca95f67ffb26 12 days ago 454MB
alpine latest 9234e8fb04c4 6 weeks ago 8.31MB
vagrant@vagrant:~/docker$ docker push tarekayari1/alpine:1.0.0
The push refers to repository [docker.io/tarekayari1/alpine]
b00718021e34: Pushed
418dcc7d85a: Mounted from library/alpine
1.0.0: digest: sha256:ff2b94daf43db487e84a0d65ef0e48e26f844722e0f5816afeb06ea21edda911 size: 741
vagrant@vagrant:~/docker$
```

19. Vérifier que l'image existe bien au niveau de DockerHub.

The screenshot shows the Docker Hub 'My Hub' interface. On the left, there's a sidebar with options like 'Repositories', 'Collaborations', 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The 'Repositories' option is selected. The main area is titled 'Repositories' and shows a single entry: 'tarekayari1/alpine'. This entry includes a timestamp '6 minutes ago', a 'IMAGE' icon, 'Public' visibility, and 'Inactive' status. There are also 'Last Pushed' and 'Contains' filters at the top of the list. A 'Create a repository' button is located in the top right corner.

Support de TP

**DevOps**

# Chapitre 4 : Vérification de la qualité du code

## SOMMAIRE

I.	Installation de SonarQube .....	2
II.	Intégration avec Jenkins .....	4

# I. Installation de SonarQube

1. Faites un chmod auparavant pour éviter les problèmes de droits d'accès.
  - sudo chmod 666 /var/run/docker.sock
2. Télécharger et installer la dernière version de l'image docker Sonar en tapant la commande suivante.
  - docker pull sonarqube

```
vagrant@vagrant:~$ vagrant@vagrant:~$  
vagrant@vagrant:~$ sudo chmod 666 /var/run/docker.sock  
vagrant@vagrant:~$ docker pull sonarqube  
Using default tag: latest  
latest: Pulling from library/sonarqube  
b71466b94f26: Pull complete  
95c1ceb63b80: Pull complete  
f62b74addfec: Pull complete  
f8ac95b0cea9: Pull complete  
8cce32af8d45: Pull complete  
5a04f5b1da55: Pull complete  
e4b4015f66e4: Pull complete  
4f4fb700ef54: Pull complete  
Digest: sha256:6d4b6cdec730c81a8ccbf266716f342d5c9d8a8c93426a2377e239d3ac79a38  
Status: Downloaded newer image for sonarqube:latest  
docker.io/library/sonarqube:latest  
vagrant@vagrant:~$
```

3. Vérifier que l'image docker de sonarqube est bien téléchargé

- docker images

```
vagrant@vagrant:~$ vagrant@vagrant:~$  
vagrant@vagrant:~$ docker images  
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE  
tarekayari1/alpine  1.0.0    eff478ce97d9  6 days ago   325MB  
postgres        latest    2e4922a83e90  7 days ago   454MB  
sonarqube       latest    c23ddf1f0583  4 weeks ago  1.22GB  
alpine          latest    9234e8fb04c4  2 months ago  8.31MB  
vagrant@vagrant:~$
```

4. Lancer le conteneur Sonarqube en arrière plan:

- docker run -d -p 9000:9000 sonarqube.

(-p pour exposer le port 9000 et pour que sonarqube soit accessible de l'extérieur), Cela prend du temps . C'est possible de lancer Sonar en background (option -d ou un docker start sur le conteneur).

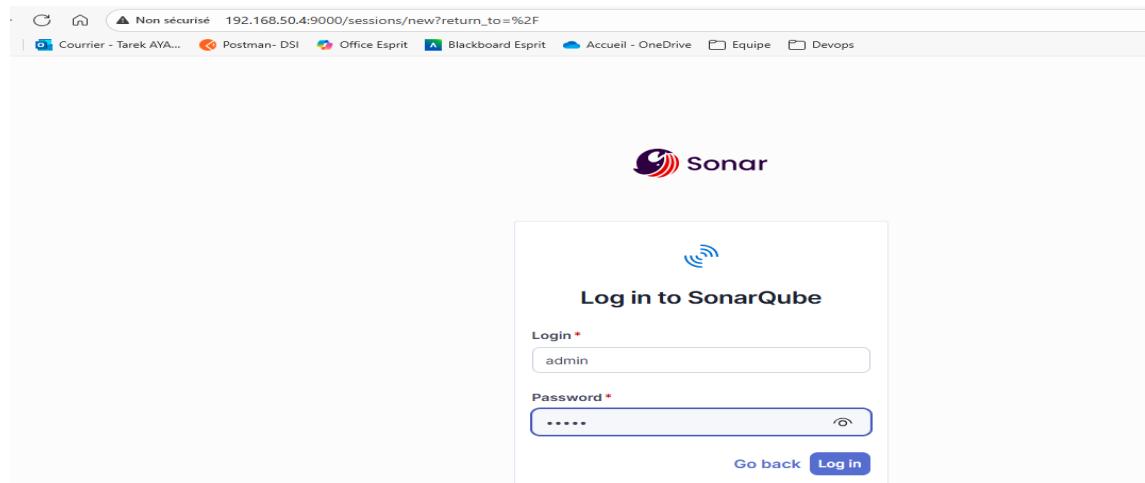
```
vagrant@vagrant:~$ docker run -p 9000:9000 sonarqube
2025.08.28 16:37:22 INFO app[][o.s.a.AppFileSystem] Cleaning or creating temp directory /opt/sonarqube/temp
2025.08.28 16:37:22 INFO app[][o.s.a.es.EsSettings] Elasticsearch listening on [HTTP: 127.0.0.1:9001, TCP: 127.0.0.1:{}]
2025.08.28 16:37:23 INFO app[][o.s.a.ProcessLauncherImpl] Launch process[ELASTICSEARCH] from [/opt/sonarqube/elasticsearch]: /opt/java/openjdk/bin/java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.script=./bin/elasticsearch -Dcli.libs=lib/tools/server-cli -Des.path.home=/opt/sonarqube/elasticsearch -Des.path.conf=/opt/sonarqube/temp/conf/es -Des.distribution.type=tar -cp /opt/sonarqube/elasticsearch/lib/*:/opt/sonarqube/elasticsearch/lib/cli-launcher/* org.elasticsearch.launcher.CliToolLauncher
2025.08.28 16:37:23 INFO app[][o.s.a.ProcessLauncherImpl] Process[ELASTICSEARCH] is up and running
```

## 5. Vérifier que le conteneur docker de Sonarqube est bien lancé

- docker ps -a

```
vagrant@vagrant:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
NAMES
f80ab18fb6fe   sonarqube     "/opt/sonarqube/dock..."   59 seconds ago   Up 58 seconds   0.0.0.0:9000->9000/tcp,
[::]:9000->9000/tcp
ca79d83e848b   postgres      "docker-entrypoint.s..."   6 days ago    Exited (1) 6 days ago
              condescending_kowalevski
57dafa04ca63   1b44b5a3e06a  "/hello"
              amazing_sinoussi
ce16ec1c3702   1b44b5a3e06a  "/hello"
              priceless_mclaren
vagrant@vagrant:~$
```

## 6. Se connecter à l'interface graphique, modifier le mot de passe et vérifier qu'il n'y a aucun projet analysé : Sur votre machine Windows, aller à l'url http://<ip-vm>:9000 et se connecter avec admin/admin



7. Changer le mot de passe à sonar par exemple (admin/Sonarqube@25).
8. Attention : Si vous arrêtez le conteneur sonarqube (CTRL S ou en arrêtant la VM), 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

```
vagrant@vagrant:~$ sudo chmod 666 /var/run/docker.sock
vagrant@vagrant:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND
f87df4171155        sonarqube          "/opt/sonarqube/dock...
2ed331260ff0        hello-world        "/hello"
313002c32776        mouradhassini/alpine:1.0.0   "/bin/sh"
9f903569b63c        postgres            "docker-entrypoint.s...
cf063f7757d8        hello-world        "/hello"
vagrant@vagrant:~$ docker start f87df
f87df
vagrant@vagrant:~$
```

## II. Intégration avec Jenkins

9. 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
10. Se connecter à Jenkins via l'url http://<ip-vm>:8080
11. Ajouter le token sonarqube dans jenkins, aller dans « Credential » et suivre les étapes suivantes :

## - Générer le token Sonarqube

The screenshot shows the SonarQube Community interface for creating a new project. At the top, there's a navigation bar with links like 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. A sidebar on the right is titled 'Administrator' and includes 'My Account' and 'Log out' options. The main content area is titled 'How do you want to create your project?'. It lists several import options:

- Import from Azure DevOps (Setup)
- Import from Bitbucket Cloud (Setup)
- Import from Bitbucket Server (Setup)
- Import from GitHub (Setup)
- Import from GitLab (Setup)

Below these options, a note says 'Are you just testing or have an advanced use-case?' followed by a link to 'Advanced setup'.

The screenshot shows the 'Administrator' section of the SonarQube Community interface. At the top, there's a navigation bar with links like 'Profile', 'Security', 'Notifications', and 'Projects'. A note below the navigation bar states: '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 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.' Below this, there's a section titled 'Generate Tokens' with a table for managing tokens. The table has columns: Name, Type, Project, Last use, Created, and Expiration. One row is shown with the following values:

Name	Type	Project	Last use	Created	Expiration
jenkins	Global Analysis Token				30 days

At the bottom of the table, it says 'No tokens'.

## - Configurer le token Sonarqube dans l'outil Jenkins

The screenshot shows the Jenkins Administration interface under the 'System' section. It includes links for 'Nodes', 'Clouds', 'Apparence générale', 'Security', 'Credentials' (which is highlighted), 'Users', and 'Credential Providers'. A search bar and a help icon are also present.

The screenshot shows the Jenkins Administration interface under the 'Identifiers' section. It displays a table of global identifiers, with one entry for 'jenkins-github-https-cred' (Type: Secret text, ID: sonarqube, Description: sonarqube). A blue button '+ Add Credentials' is visible at the top right.

ID	Nom	Type	Description
jenkins-github-https-cred	tarek-ayari/******** (Access Github)	Nom d'utilisateur et mot de passe	Access Github

The screenshot shows the Jenkins Administration interface under the 'Identifiers' section, with a new credential creation form. The form fields include 'Type' (Secret text), 'Portée' (Global), 'Secret' (sonarqube), 'ID' (sonarqube), and 'Description' (sonarqube). A 'Create' button is at the bottom.

## 12. Vérifier les plugins installés sur jenkins :

The screenshot shows the Jenkins 'Plugins' page. A search bar at the top contains the text 'sonar'. Below it, a sidebar has tabs: 'Mises à jour' (0), 'Plugins disponibles' (0), 'Plugins installés' (highlighted), and 'Paramètres avancés'. The main area lists two plugins: 'Sonargraph Integration Jenkins Plugin 5.0.2' and 'SonarQube Scanner for Jenkins 2.18'. Both are marked as healthy ('Santé') and active ('Activé').

## 13. Ajouter les outils Sonar sur Jenkins, Aller dans le menu Tools et chercher SonarQube Scanner :

The screenshot shows the Jenkins 'Configuration du système' page. Under the 'Tools' section, there is a link to 'Configurer les outils, leur localisation et les installateurs automatiques.' Below this, under 'Sécurité', there is a 'Security' section for securing Jenkins.

The screenshot shows the Jenkins 'Tools' configuration page. It displays a 'SonarQube Scanner' configuration card. The 'Name' field is set to 'sonarqube' and the 'Install automatically' checkbox is checked. Below this, a 'Installer depuis Maven Central' section shows the selected version 'SonarQube Scanner 7.2.0.5079'.

14. Ajouter un stage dans le Job « FirstPipeline » sonarqube avec tous le paramètres de configurations :

```

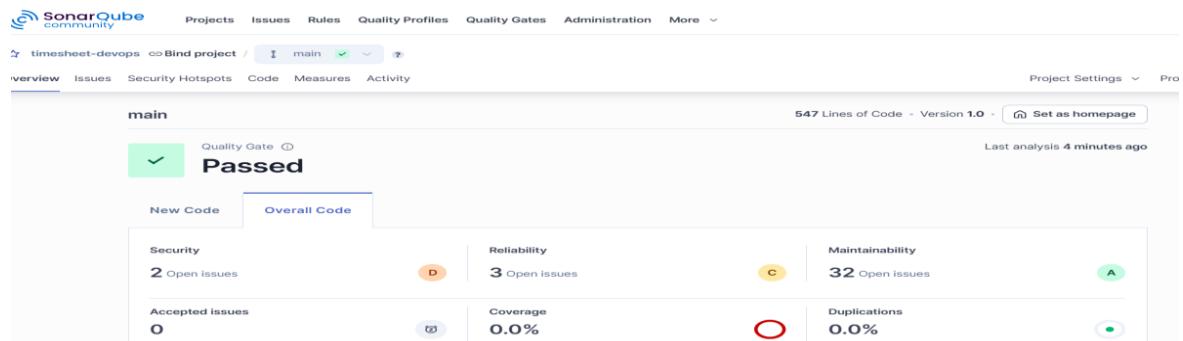
stage('MAVEN Build') {
    steps {
        // Use the 'mvn' command to compile, test, and package the project
        // '-B' for non-interactive mode, '-DskipTests' to skip tests (remove if you want to run tests)
        sh 'mvn clean compile'
    }
}

stage('SONARQUBE') {
    environment{
        SONAR_HOST_URL='http://192.168.50.4:9000/'
        SONAR_AUTH_TOKEN= credentials('sonarqube')
    }

    steps {
        sh 'mvn sonar:sonar -Dsonar.projectKey=devops_git -Dsonar.host.url=$SONAR_HOST_URL -Dsonar.token=$SONAR_AUTH_TOKEN'
    }
}

```

15. Lancer le build Pipeline et aller dans <http://<ip-vm>:9000> et regarder le résultat de l'analyse de votre projet.



- **Bug:** Un code qui peut engendrer un mauvais comportement de l'application
- **Vulnerability:** Faille de sécurité dans notre code.
- **Hotspots Reviewed:** Code à revoir pour être sûr que ce n'est pas une faille de sécurité.
- **Code Smells:** Ce n'est pas un bug, mais c'est un code qui peut retarder l'équipe de développement



Support de TP

**DevOps**

# Chapitre 5 : Supervision continue

## SOMMAIRE

I.	Démarrage de l'application Prometheus .....	2
II.	Intégration de Grafana .....	5
III.	Surveillance de l'application.....	7

# I. Démarrage de l'application Prometheus

1. Nous allons surveiller Jenkins avec Prometheus et afficher les métriques avec Grafana. Vous pouvez surveiller d'autres composants comme le conteneur Spring Boot ou le conteneur de base de données de votre application finale.
2. 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

The screenshot shows the Jenkins Plugins management interface. In the top navigation bar, there are links for 'Courrier - Tarek AYA...', 'Postman- DS...', 'Office Esprit', 'Blackboard Esprit', 'Accueil - OneDrive', 'Equipe', 'Devops', and the Jenkins logo. Below the navigation, the page title is 'Jenkins / Administrer Jenkins / Plugins'. On the left, there's a sidebar with tabs: 'Mises à jour' (0), 'Plugins disponibles' (selected), 'Plugins installés', and 'Paramètres avancés'. The main area has a search bar with the query 'promet'. A table lists three available plugins:

- Prometheus metrics** (819.v50953a\_c560dd)
- Otel agent host metrics monitoring** (1.4.2)
- Cortex Metrics** (1.0.1)

Each row includes a 'Installer' button, the plugin name, its version, its category (e.g., 'monitoring', 'Divers'), a brief description, the last update date ('Il y a 6 mo. 11 j', 'Il y a 1 mo. 23 j', 'Il y a 4 an. 5 mo.'), and a green circular badge with the number of reviews (96, 100, 91).

The screenshot shows the Jenkins Plugins management interface during a plugin download. The top navigation bar and sidebar are identical to the previous screenshot. The main area has a title 'Progression du téléchargement'. On the left, the sidebar shows the 'Progression des téléchargements' tab is selected. The right side displays the download status for the 'Prometheus metrics' plugin:

- Préparation:** A bulleted list: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'.
- Pipeline Graph Analysis:** Status 'Succès' (green checkmark).
- Pipeline: REST API:** Status 'Succès' (green checkmark).
- Prometheus metrics:** A note: 'Le plugin prometheus ne supporte pas le chargement dynamique. Jenkins doit être redémarré pour que la mise à jour soit effective.' followed by 'Loading plugin extensions' and 'Success' (green checkmark).

At the bottom, there are two links:

- [Revenir en haut de la page](#) (you can start using the installed plugins immediately)
- Redémarrer Jenkins quand l'installation est terminée et qu'aucun job n'est en cours

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

3. 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

```
vagrant@vagrant:~$ sudo systemctl restart jenkins.service
vagrant@vagrant:~$ sudo chmod 666 /var/run/docker.sock
vagrant@vagrant:~$ docker run -d --name prometheus -p 9090:9090 prom/prometheus
Unable to find image 'prom/prometheus:latest' locally
latest: Pulling from prom/prometheus
9fa9226be034: Pull complete
1617e25568b2: Pull complete
097a69c6efe6: Pull complete
2ee6cb77beb7: Pull complete
a4e782810d03: Pull complete
76619c1908eb: Pull complete
2dfc70ad9941: Pull complete
fd1d3a5a5f79: Pull complete
5e4c02bc6754: Pull complete
208063e2dcbb: Pull complete
Digest: sha256:63805ebb8d2b3920190daf1cb14a60871b16fd38bed42b857a3182bc621f4996
Status: Downloaded newer image for prom/prometheus:latest
491136bfddaa1e736a2c1f418245f50afbbf1af62ca2c07375e32c1c451676b1
vagrant@vagrant:~$ docker exec -it prometheus sh
/prometheus $
```

4. Mettre à jour le conteneur prometheus. récupérer la commande complète du fichier prometheus.yml.txt

```
tee -a /etc/prometheus/prometheus.yml <<EOF
  - job_name: FirstPipeline
    metrics_path: /prometheus
    static_configs:
      - targets: ['192.168.50.4:8080']
EOF
```

- docker restart prometheus

5. Tester les différents url prometheus pour vérifier que la configuration est correcte.

- [http://@IP\\_VM:9090/targets](http://@IP_VM:9090/targets)

The screenshot shows the Prometheus interface at <http://192.168.50.4:9090/targets>. It displays two sections: 'jenkins' and 'prometheus'. Each section lists an endpoint with its labels and the last scrape time. Both targets are marked as 'UP'.

Endpoint	Labels	Last scrape	State
<a href="http://192.168.50.4:8080/prometheus">http://192.168.50.4:8080/prometheus</a>	instance="192.168.50.4:8080", job="jenkins"	12.813s ago	66ms UP
<a href="http://localhost:9090/metrics">http://localhost:9090/metrics</a>	app="prometheus", instance="localhost:9090", job="prometheus"	5.021s ago	27ms UP

- Les logs seront visibles dans l'url suivante : [http://@IP\\_VM:8080/prometheus/](http://@IP_VM:8080/prometheus/)

The screenshot shows the Prometheus metrics endpoint at <http://192.168.50.4:9090/metrics>. It displays a long list of metrics related to Go's garbage collection, including counters for completed GC cycles, forced GC cycles, and duration seconds, as well as histograms for heap allocations by size.

```

# HELP go_gc_cycles_automatic_gc_cycles_total Count of completed GC cycles generated by the Go runtime. Sourced from /gc/cycles/automatic:gc-cycles.
# TYPE go_gc_cycles_automatic_gc_cycles_total counter
go_gc_cycles_automatic_gc_cycles_total 11
# HELP go_gc_cycles_forced_gc_cycles_total Count of completed GC cycles forced by the application. Sourced from /gc/cycles/forced:gc-cycles.
# TYPE go_gc_cycles_forced_gc_cycles_total counter
go_gc_cycles_forced_gc_cycles_total 0
# HELP go_gc_cycles_total_gc_cycles_total Count of all completed GC cycles. Sourced from /gc/cycles/total:gc-cycles.
# TYPE go_gc_cycles_total_gc_cycles_total counter
go_gc_cycles_total_gc_cycles_total 11
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 6.991e-05
go_gc_duration_seconds{quantile="0.25"} 0.000125184
go_gc_duration_seconds{quantile="0.5"} 0.000191004
go_gc_duration_seconds{quantile="0.75"} 0.000378795
go_gc_duration_seconds{quantile="1"} 0.00087621
go_gc_duration_seconds_sum 0.003095185
go_gc_duration_seconds_count 11
# HELP go_gc_gogo_percent Heap size target percentage configured by the user, otherwise 100. This value is set by the GOGC environment variable, and the runtime/debug.SetGCPerc function. S
/gc/gogo:percent.
# TYPE go_gc_gogo_percent gauge
go_gc_gogo_percent 75
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This value is set by the GOMEMLIMIT environment variable, and the runtime/debug.SetMemor
function. Sourced from /gc/gomemlimit:bytes.
# TYPE go_gc_gomemlimit_bytes gauge
go_gc_gomemlimit_bytes 4.52408279e+09
# HELP go_gc_heap_allocs_by_size_bytes Distribution of heap allocations by approximate size. Bucket counts increase monotonically. Note that this does not include tiny objects as defined by
/gc/heap/tiny/allocs:objects, only tiny blocks. Sourced from /gc/heap/allocs-by-size:bytes.
# TYPE go_gc_heap_allocs_by_size_bytes histogram
go_gc_heap_allocs_by_size_bytes_bucket{le="8.999999999999998"} 7732
go_gc_heap_allocs_by_size_bytes_bucket{le="24.99999999999996"} 66116
go_gc_heap_allocs_by_size_bytes_bucket{le="64.9999999999999"} 176336
go_gc_heap_allocs_by_size_bytes_bucket{le="144.9999999999997"} 236363
go_gc_heap_allocs_by_size_bytes_bucket{le="320.9999999999994"} 248960
go_gc_heap_allocs_by_size_bytes_bucket{le="704.9999999999999"} 250819
go_gc_heap_allocs_by_size_bytes_bucket{le="1536.9999999999998"} 251934

```

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

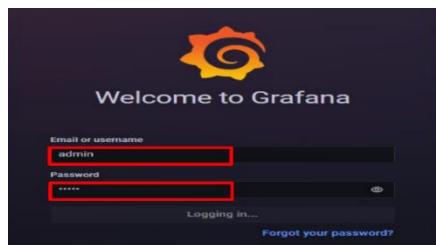
- Vous pouvez configurer prometheus avec Jenkins en accédant au menu suivant : Jenkins > System > Prometheus

The screenshot shows the Jenkins 'System' configuration page under the 'Data Sources' section. A new data source named 'Prometheus' is being added. The 'Path' field contains 'prometheus'. The 'Default Namespace' is set to 'default'. The 'Collecting metrics period in seconds' is set to '5'. Under 'Metrics to collect', several checkboxes are checked: 'Count duration of successful builds', 'Count duration of unstable builds', 'Count duration of failed builds', 'Count duration of not-built builds', 'Count duration of aborted builds', and 'Fetch the test results of builds'. Other options like 'Add build parameter label to metrics', 'Add build status label to metrics', and 'Process disabled jobs' are unchecked. The 'Job attribute name' is set to 'jenkins\_job'. The 'Build parameters that will be added as separate labels to metrics' field is empty. At the bottom, there are two checkboxes: 'Collect disk usage' and 'Collect memory usage'.

## II. Intégration de Grafana

### 6. Création d'un conteneur Docker Grafana.

- Docker run -d --name grafana -p 3000:3000 grafana/grafana
- Accéder à Grafana : [http://@IP\\_VM:3000](http://@IP_VM:3000) : L'utilisateur et le mot de passe par défaut sont "admin/admin".

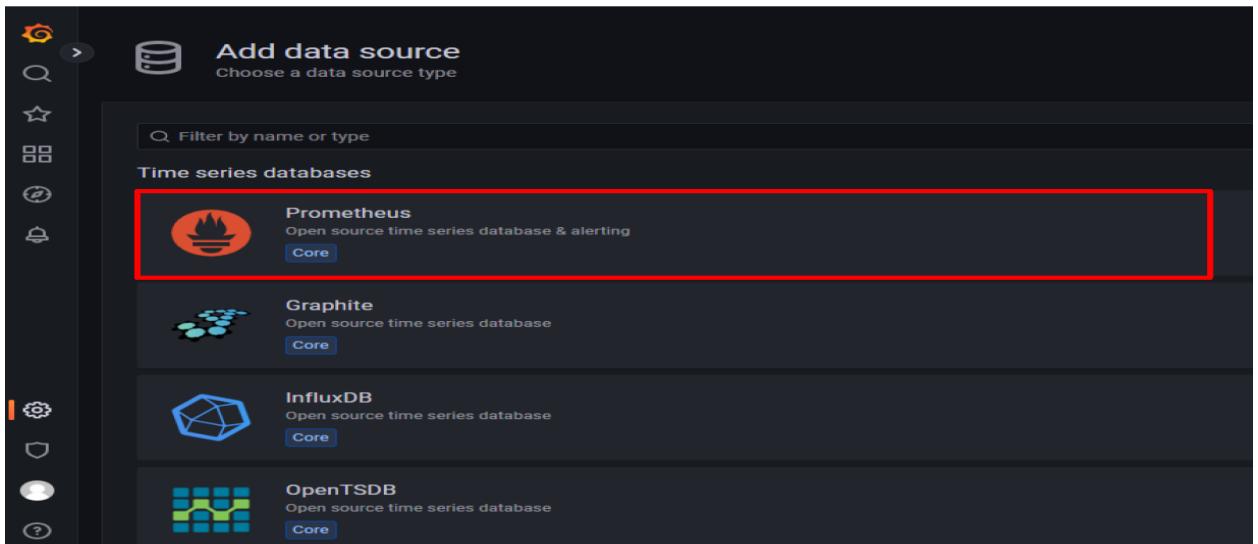


### 7. Changer le mot de passe (par exemple login : admin password: grafana)

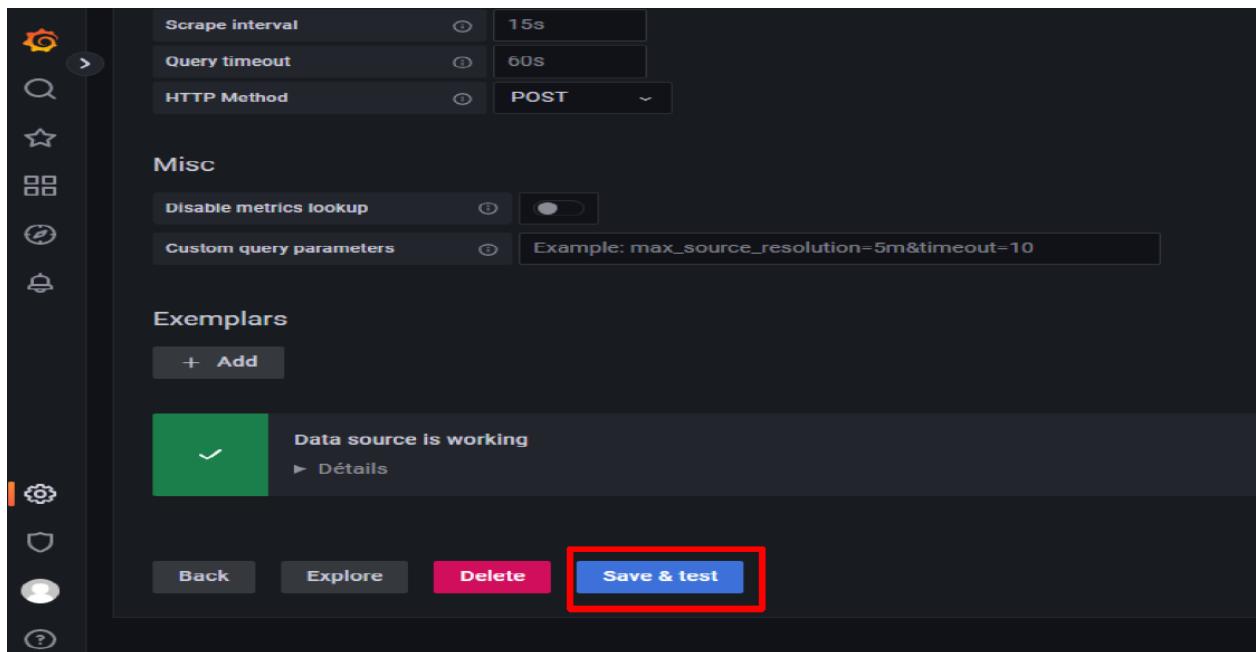
### 8. Ajouter la source des données → Prometheus

The screenshot shows the Grafana home dashboard. On the left, there is a sidebar with various icons. The main area has a 'Basic' section with instructions for setting up Grafana. To the right, there are several cards: 'TUTORIAL', 'DATA SOURCE AND DASHBOARDS', 'Grafana fundamentals', 'DATA SOURCES' (which is highlighted with a red box), and 'DASHBOARDS'. The 'DATA SOURCES' card contains the text 'Add your first data source' and a database icon. Below it is a link 'Learn how in the docs'.

- Récupérer l'adresse IP du conteneur docker Grafana.

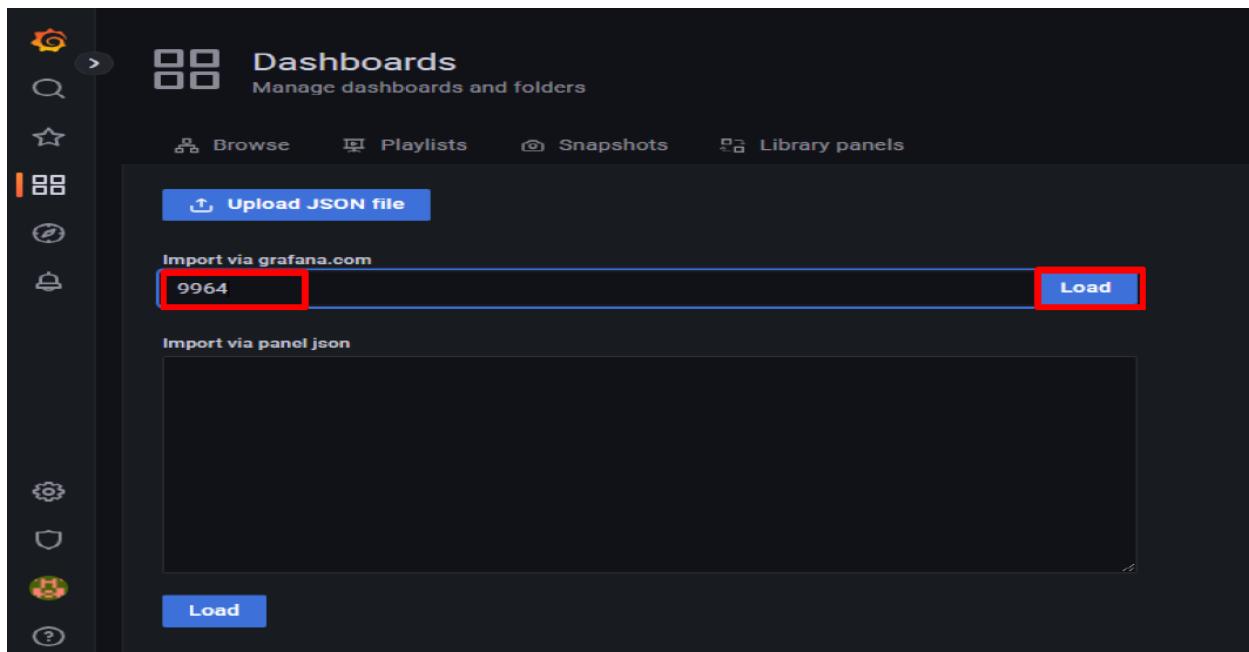


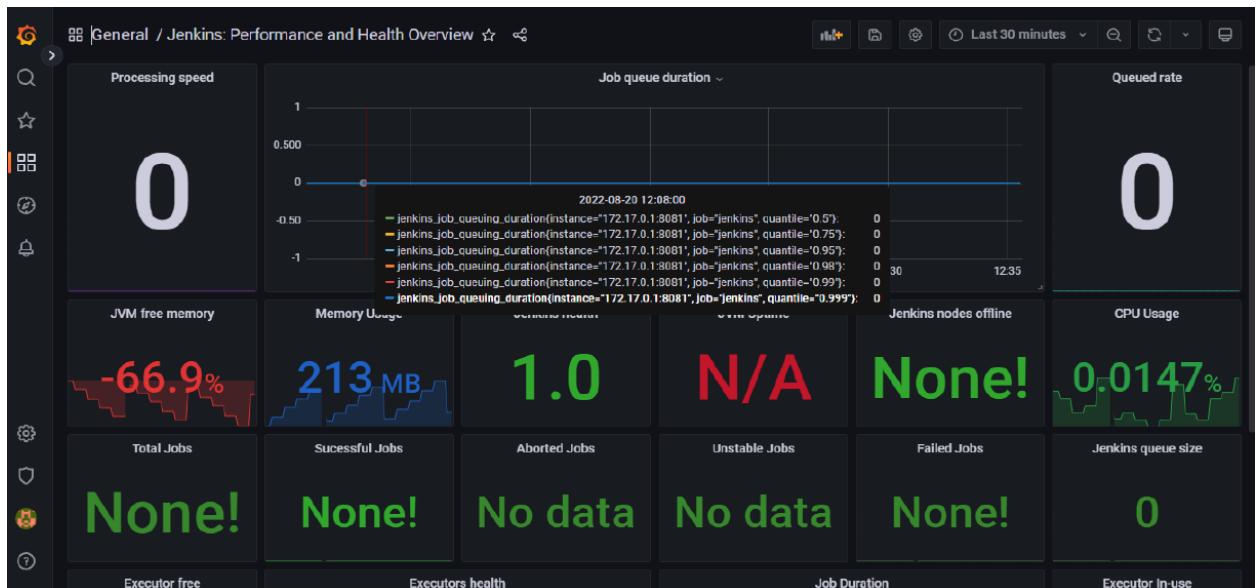
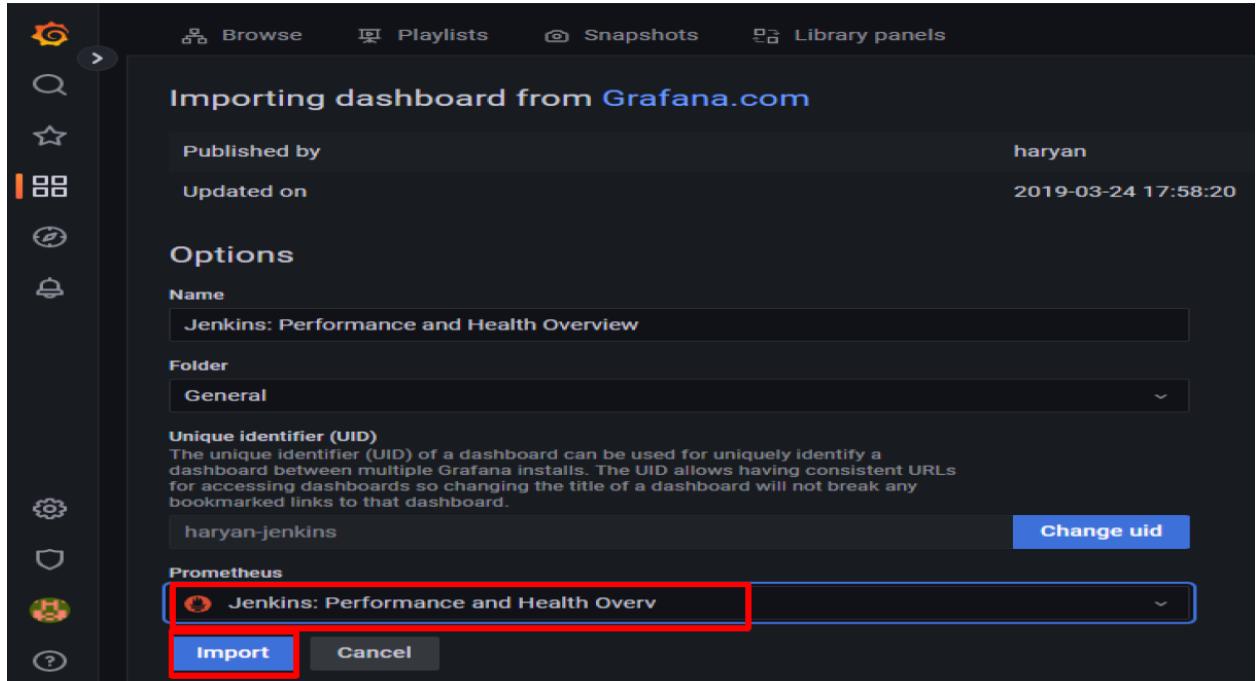
The screenshot shows the 'Data Sources / Prometheus' configuration screen. The 'Settings' tab is active. A data source named 'Jenkins: Performance and Health Overview' is selected. The 'Name' field contains the value 'Jenkins: Performance and Health Overview' and the 'URL' field contains 'http://192.168.1.14:9090', both of which are highlighted with a red box. The 'HTTP' section includes fields for 'Access', 'Allowed cookies', and 'Timeout'. The 'Auth' section includes options for 'Basic auth', 'TLS Client Auth', 'Skip TLS Verify', 'With Credentials', 'With CA Cert', and 'Forward OAuth Identity', none of which have checkboxes checked.



## III. Surveillance de l'application

9. Accéder à [http://@IP\\_VM:3000/dashboard/import](http://@IP_VM:3000/dashboard/import) et utiliser 9964 l'identifiant d'un cd





Support de TP

**DevOps**

# Chapitre 6 : Orchestration des conteneurs

## SOMMAIRE

I.	Installation de Minikube .....	2
II.	Installation Serveur Nginx .....	2
III.	Création des services et des déploiements.....	4
	Minikube est un choix justifié par les contraintes suivantes : .....	10
▪	Simplicité d'installation .....	10
▪	Environnement idéal pour l'apprentissage.....	10
▪	Consommation réduite de ressources.....	10
▪	Support des fonctionnalités Kubernetes.....	10
▪	Pas besoin d'infrastructure cloud.....	10
▪	Bonne intégration avec les outils DevOps.....	11

# I. Installation de Minikube

1. Telecharger Minikube.
  - curl -LO <https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64>
2. Install the Minikube binary
  - sudo install minikube-linux-amd64 /usr/local/bin/minikube
3. Supprimer le fichier d'installation
  - rm minikube-linux-amd64
4. Telecharger Kubectl
  - curl -LO [https://dl.k8s.io/release/\\$\(curl -L -s https://dl.k8s.io/release/stable.txt\)/bin/linux/amd64/kubectl](https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl)
5. Installer Kubectl: C'est l'outil en ligne de commande officiel pour interagir avec un cluster Kubernetes
  - sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
6. Supprimer le fichier d'installation
  - rm kubectl
7. Une fois l'installation minikube finalisé, démarrer minikube
  - minikube start --driver=docker
8. Vérifier l'installation
  - minikube version
  - minikube status
  - kubectl version --client
  - kubectl config get-contexts

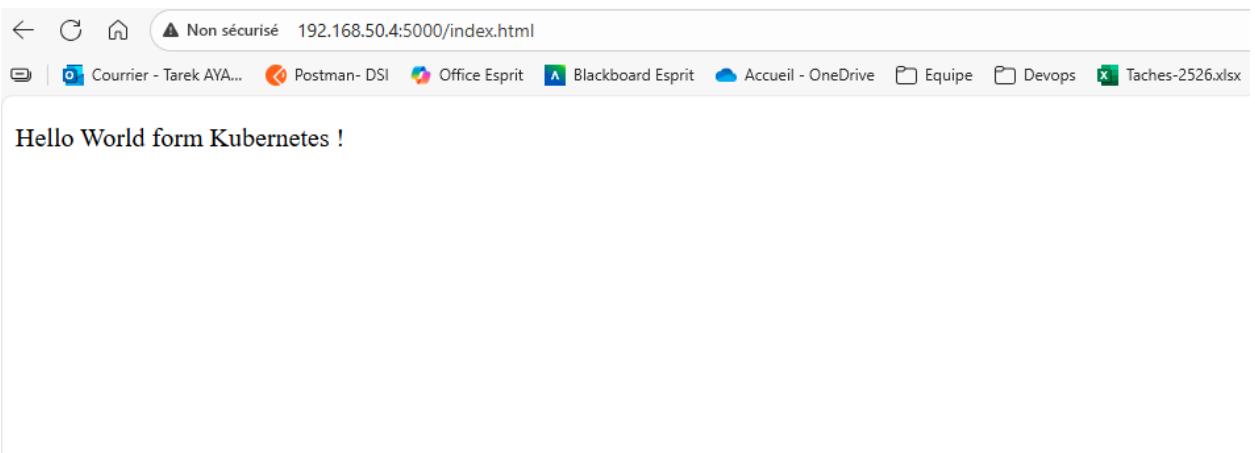
# II. Installation Serveur Nginx

1. Lancer la commande d'installation suivante
  - docker run -it --rm -d -p 5000:80 --name web nginx

By default, Nginx looks in the /usr/share/nginx/html directory inside of the container for files to serve.

2. Stopper le serveur
  - docker stop web

3. Créer le répertoire « kubernetes » puis y accéder pour bien structurer et centraliser les fichiers de configuration relatives à cet atelier
  - mkdir Kubernetes
4. Créer le fichier Index.html
5. Lancer la commande suivante pour monter le répertoire kubernetest dans la racine de stockage de fichier html cote nginx
  - docker run -it --rm -d -p 5000:80 --name web -v ~/kubernet:/usr/share/nginx/html nginx
6. Vous pouvez tester l'affichage de la page html
  - <http://192.168.50.4:5000/index.html>



# III. Création des services et des déploiements

Dans cet atelier vous allez :

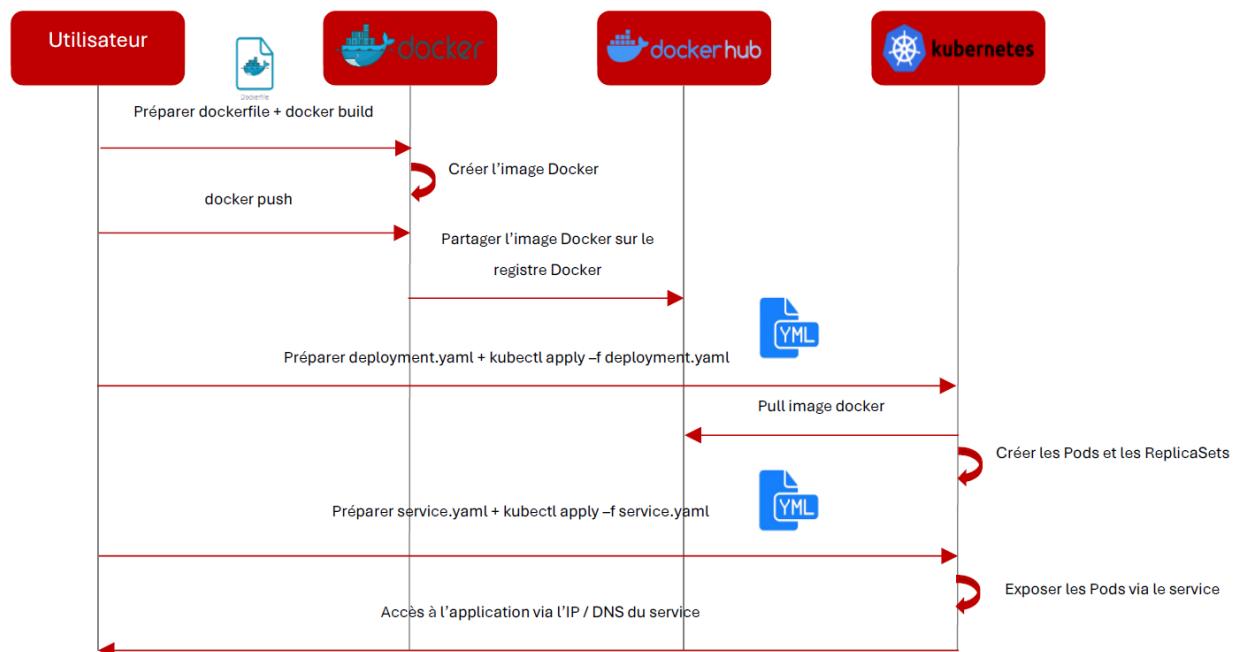
1.  Créer une image Docker
2.  Déployer sur Minikube (Kubernetes)
3.  Accéder à votre site 'Hello World'
4.  Diagnostiquer des problèmes courants

**Objectif : Déployer un site web statique "Hello World" avec Kubernetes en suivant un enchaînement précis.**

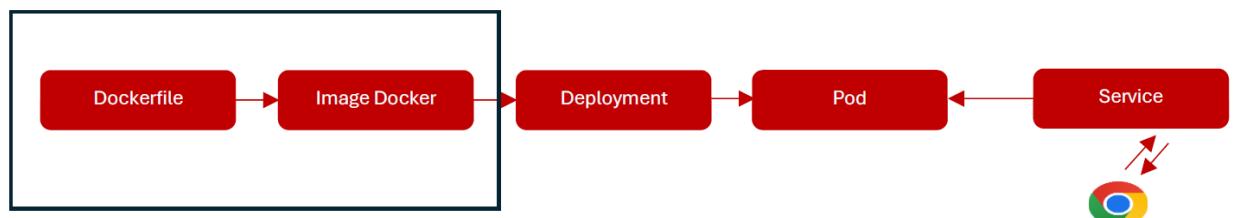
**Livrables :**

- Les fichiers :
  - index.html
  - Dockerfile
  - deployment.yaml
  - service.yaml
- Une URL accessible (via minikube) affichant "Hello World".

## Étapes à suivre dans cet atelier pour atteindre l'objectif :



### Partie 01 : Préparer l'image Docker à déployer



1. Dans le même répertoire, Créer le fichier « Dockerfile » pour créer l'image afin de déployer la page web « Hello world ».

2. Exécuter les commandes nécessaires pour partager l'image créée sur « DockerHub » (Étape Optionnelle)

- docker build . -t test-app
- docker run -p 8085:80 -d test-app

### Verification Application

Pour vérifier l'application s'exécute correctement vous pouvez lancer le navigateur avec le port utilisé

- <http://192.168.50.4:8085/index.html>

Non sécurisé 192.168.50.4:8085/index.html

Courrier - Tarek AYA... Postman- DS1 Office Esprit Blackboard Esprit Accueil - OneDrive Equipe Devops Taches-2526.xlsx SU-ASI1 - Document... DotNet

Hello World form Kubernetes !

### 3. Se connecter au Docker Hub pour déployer l'image créé

- docker login -u tarekayari1
- docker tag test-app:latest tarekayari1/test-app:1.0.0
- docker push tarekayari1/test-app:1.0.0

[Repositories](#) / [test-app](#) / [General](#)

## tarekayari1/test-app

Last pushed in 3 minutes

Add a description

Add a category

[General](#) Tags Image Management BETA Collaborators Webhooks Se

### Tags

DOCKER SCOUT INACTIVE

[Activate](#)

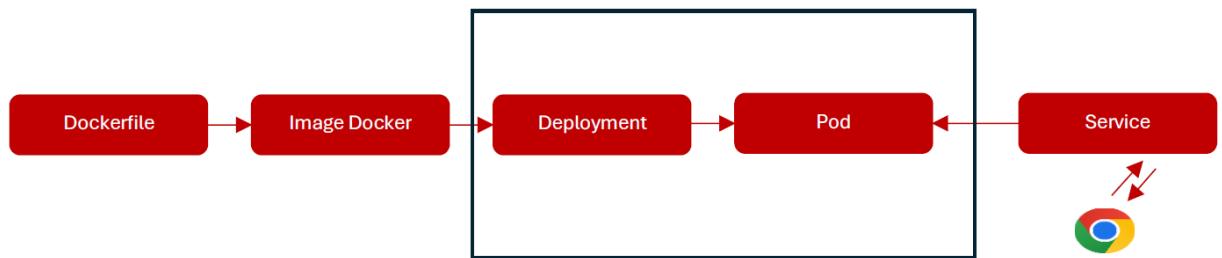
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
1.0.0		Image	less than 1 day	3 minutes

[See all](#)

→ A ce niveau, l'image Docker est déposée et prête pour être utilisé afin de faire le déploiement avec minikube (Kubernetes)

## Partie 02 : Déployer l'image Docker avec Minikube (Kubernetes)



1. Créer le fichier « deployment.yaml »

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-app
  template:
    metadata:
      labels:
        app: test-app
    spec:
      containers:
        - name: test-app
          image: tarekayari1/test-app:1.0.0
          ports:
            - containerPort: 80
```

## 2. Appliquer la configuration du service avec la commande

- `kubectl apply -f deployment.yaml`

```
vagrant@vagrant:~/hello-kubernetes$ kubectl apply -f deployment.yaml
deployment.apps/hello-deployment created
vagrant@vagrant:~/hello-kubernetes$
```

## 3. Vérifier le Déploiement

### 3.1 Lister tous les pods pour vérifier leur statut

- `kubectl get pods`

```
vagrant@vagrant:~/hello-kubernetes$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-deployment-78bf47f886-8w9wg   1/1     Running   0          22s
vagrant@vagrant:~/hello-kubernetes$
```

### 3.2 Lister les Deployments pour vérifier la disponibilité

- `kubectl get deployments`

```
vagrant@vagrant:~/hello-kubernetes$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
hello-deployment   1/1       1          1          27m
vagrant@vagrant:~/hello-kubernetes$
```

→ À ce stade, le déploiement de l'image Docker est effectué avec Minikube et l'application est opérationnelle au sein du cluster Kubernetes.

## Partie 03 : Exposer l'application avec un Service Minikube (Kubernetes)

### 1. Créer le fichier « service.yaml »

```
apiVersion: v1
kind: Service
metadata:
  name: test-service
spec:
  type: NodePort
  selector:
    app: test-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30036
```

## 2. Appliquer la configuration du service avec la commande

- **kubectl apply -f service.yaml**

```
vagrant@vagrant:~/kubernetes$ kubectl apply -f deployment.yaml
deployment.apps/test-deployment created
```

## 3. Vérifier le Service

Lister les Services pour obtenir l'IP externe

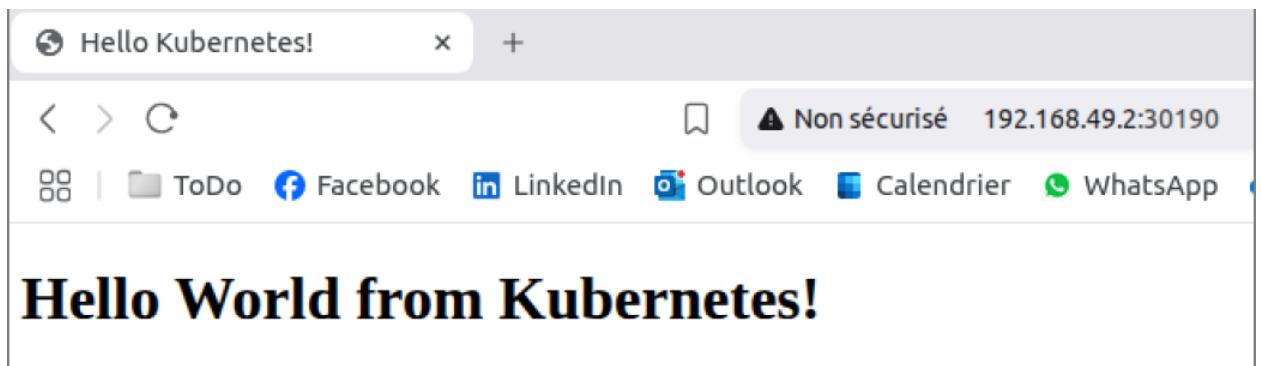
- **kubectl get services**

```
vagrant@vagrant:~/kubernetes$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
hello-deployment-78bf47f886-8w9wg   1/1     Running   0          121m
test-deployment-75bccb9b59-7j48c   1/1     Running   0          68s
```

## 4. Récupérer l'URL d'accès externe générée par Minikube pour le service « hello-service »

- **minikube service hello-service --url**

```
vagrant@vagrant:~/kubernetes$ minikube service test-service
NAME         NAMESPACE   TARGET PORT   URL
test-service default      80           http://192.168.49.2:31075
[1]  Opening service default/test-service in default browser...
http://192.168.49.2:31075
```



### Problème/Solution :

- **ImagePullBackOff** : Vérifier le nom de l'image dans deployment.yaml et les accès Docker Hub.
- **Service en Pending (Minikube)** : Lancer minikube tunnel dans un terminal séparé.
- **Port déjà utilisé** : Changer le port dans service.yaml (exemple : 8080:80).
- **Erreur 403** : Vérifier que index.html est bien copié dans /usr/share/nginx/html.

# Annexe

## 1. Justification choix minikube

**Minikube** est un outil open source qui permet de **créer un cluster Kubernetes local** (mononœud) sur une machine personnelle ou un serveur léger.

Il fournit :

- Un cluster Kubernetes **fonctionnel** (API Server, kubelet, scheduler...)
- Un environnement léger pour **tester, apprendre ou développer** des applications Kubernetes
- Une installation facile avec des drivers comme VirtualBox, Docker, KVM ou Hyper-V

**Minikube** est un choix justifié par les contraintes suivantes :

### ■ Simplicité d'installation

- Une seule commande suffit pour installer et démarrer : `minikube start`
- Pas besoin de configurer un cluster multi-nœuds complexe
- Tout tourne localement (VM ou container)

### ■ Environnement idéal pour l'apprentissage

- Parfait pour **les étudiants**, enseignants ou formateurs souhaitant enseigner :
  - Déploiement de services Kubernetes
  - Gestion de pods, secrets, volumes, etc.
  - Concepts comme les Ingress, les Probes, les Volumes Persistants, etc.

- Très utilisé pour les **TPs ou ateliers DevOps / Cloud / Microservices**

### ■ Consommation réduite de ressources

- Fonctionne sur des machines avec **4 à 8 Go de RAM**
- Parfait pour des postes étudiants ou des VM dans des salles de cours
- Possibilité de limiter les ressources : `--memory`, `--cpus`

### ■ Support des fonctionnalités Kubernetes

- Supporte :
  - `kubectl`, `helm`, `ingress`, `dashboard`, `metrics-server`, etc.
  - Les volumes persistants locaux
  - Le load balancer via `minikube tunnel`
  - L'exposition des services via `minikube service`

### ■ Pas besoin d'infrastructure cloud

- Ne dépend **d'aucune solution cloud payante** (GKE, AKS, EKS)
- Pas de connexion Internet nécessaire après le setup initial

- Idéal pour travailler **hors ligne ou en intranet académique**
- **Bonne intégration avec les outils DevOps**
  - Compatible avec kubectl, Helm, Kustomize, skaffold, Docker
  - Peut être utilisé avec GitLab CI, Jenkins, GitHub Actions en mode local

Minikube est **le choix idéal dans un cadre académique** pour apprendre Kubernetes, car :

- Il est **léger et accessible** même sur des machines modestes
- Il offre une **expérience complète et réaliste** de Kubernetes
- Il **ne nécessite pas de cloud ni de ressources importantes**
- Il permet aux étudiants de se former en toute autonomie, localement

## 2. Kubectl Command

### 2.1 Configuration files commands

In Kubernetes, configuration files (often called manifest files or YAML files) define the desired state of cluster resources using a declarative format. These files tell Kubernetes what to create, modify, or manage, such as pods, deployments, services, or volumes.

Each file typically contains:

An apiVersion to specify the Kubernetes API version

A kind to define the type of resource (e.g., Deployment, Service)

metadata for naming and labeling

A spec section that outlines the configuration details (like container images, replicas, or ports)

The core command to apply a Kubernetes configuration file is:

kubectl apply -f <configuration file>

This command creates or updates the resources defined in the YAML file to match the desired state.

kubectl create -f <configuration file> – Create objects.

kubectl create -f <configuration file directory> – Create objects in all manifest files in a directory.

kubectl create -f <'url'> – Create objects from a URL.

kubectl get -f <configuration file> – Retrieve the status of the resources.

kubectl diff -f <configuration file> – Show differences between the live cluster state and the config file.

kubectl delete -f <configuration file> – Delete an object

## 2.2. Deployment commands

A Kubernetes Deployment is a controller that manages the lifecycle of application Pods by declaratively defining the desired state and ensuring the system continuously aligns with it.

A Deployment specifies how many Pods should run, what container images they use, and how they should be updated. It works with ReplicaSets to ensure the correct number of pod replicas are always running. The Deployment Controller monitors the cluster and automatically replaces or scales Pods as needed to match the defined state.

kubectl get deployment – List one or more Deployments.

kubectl describe deployment <deployment\_name> – Display the detailed state of one or more Deployments.

kubectl edit deployment <deployment\_name> – Edit and update the definition of one or more Deployments on the server.

kubectl create deployment <deployment\_name> – Create a new Deployment.

kubectl delete deployment <deployment\_name> – Delete Deployments.

kubectl rollout status deployment <deployment\_name> – See the rollout status of a Deployment.

kubectl set image deployment/<deployment name> <container name>=image:<new image version> – Perform a rolling update (K8S default), set the image of the container to a new version for a particular Deployment.

`kubectl rollout undo deployment/<deployment name>` – Roll back a previous Deployment.

`kubectl replace --force -f <configuration file>` – Perform a replace Deployment: Force replace, delete, and then re-create the resource.

`kubectl scale deployment <deployment name> --replicas=<n>` – Manually scale the number of pod replicas in a Deployment.

Read more about Kubernetes deployment strategies: [Different Types of Kubernetes Deployment Strategies \(Examples\)](#)

### 2.3. Events commands

A Kubernetes event is a record of a state change or occurrence within the cluster, typically used for debugging or auditing. Events are generated by the Kubernetes system and components such as the scheduler, controllers, or kubelet.

`kubectl get events` – List recent events for all resources in the system.

`kubectl get events --field-selector type=Warning` – List Warnings only.

`kubectl get events --sort-by=.metadata.creationTimestamp` – List events sorted by timestamp.

`kubectl get events --field-selector involvedObject.kind!=Pod` – List events but exclude Pod events.

`kubectl get events --field-selector involvedObject.kind=Node, involvedObject.name=<node_name>` – Pull events for a single node with a specific name.

`kubectl get events --field-selector type!=Normal` – Filter out normal events from a list of events.

## 2.4. Log commands

Kubernetes logs refer to the output generated by containers, pods, nodes, and cluster components, used to monitor and debug applications running within a Kubernetes environment.

`kubectl logs <pod_name>` – Print the logs for a pod.

`kubectl logs --since=6h <pod_name>` – Print the logs for the last 6 hours for a pod.

`kubectl logs --tail=50 <pod_name>` – Get the most recent 50 lines of logs.

`kubectl logs -f <pod_name>` – Print the logs for a pod and follow new logs.

`kubectl logs -c <container_name> <pod_name>` – Print the logs for a container in a pod.

`kubectl logs <pod_name> pod.log` – Output the logs for a pod into a file named “pod.log”.

`kubectl logs --previous <pod_name>` – View the logs for a previously failed pod.

`kubectl logs --timestamps <pod_name>` – Include timestamps with each log line