



Rapport : État d'avancement Délégué / Scrum Master

Infrastructure-Déploiement

GR 12

22 décembre 2024

1. Introduction	2
2. Technologies et outils utilisés pour le déploiement	2
3. Étapes de mise en œuvre du déploiement	3
Voici un aperçu des étapes nécessaires pour configurer et exécuter le déploiement automatique.	3
3.2 Jenkins :	4
3.2.1 Créer une instance de VMS (Virtual Machine Server) EC2 pour installer Jenkins.	4
3.2.2 Connection entre Jenkins et Github	5
3.2.3 Connection entre Jenkins et SonarQube EC2	7
3.2.4 Connection entre Jenkins et Docker EC2	8
3.2.5 Exécuter les Commands Shell pour Build application	10
Docker-compose	12
Dockerfile :	14
3.3 Docker:	15
3.3.1 Créer une instance de VMS (Virtual Machine Server) EC2 pour installer Docker (deployment server).	15
3.3.2 installer Docker et Docker-compose :	15
3.4 SonarQube:	15
3.4.1 Créer une instance de VMS (Virtual Machine Server) EC2 pour installer SonarQube.	15
3.4.2 installer SonarQube dans EC2 :	15

1. Introduction

Le déploiement est une étape clé dans un pipeline CI/CD. Il garantit que les applications développées et testées sont mises en production ou sur des environnements appropriés de manière cohérente, fiable et automatisée .

et comme on travail avec un méthode Agile donc ce méthode nous permanent de faire les chose suivantes:

- Continuous Development
- Continuous Integration
- Continuous Testing
- Continuous Delivery → Besoin d'une approbation manuelle pour toute version de production.

2. Technologies et outils utilisés pour le déploiement

Pour réaliser le déploiement de notre application nous va utiliser les technologie suivants:

2.1 Github: Pour collecter le Code, et contrôler les versions.

2.2 Jenkins : Pour orchestrer les étapes de création d'images, d'analyse du code, et de déploiement.

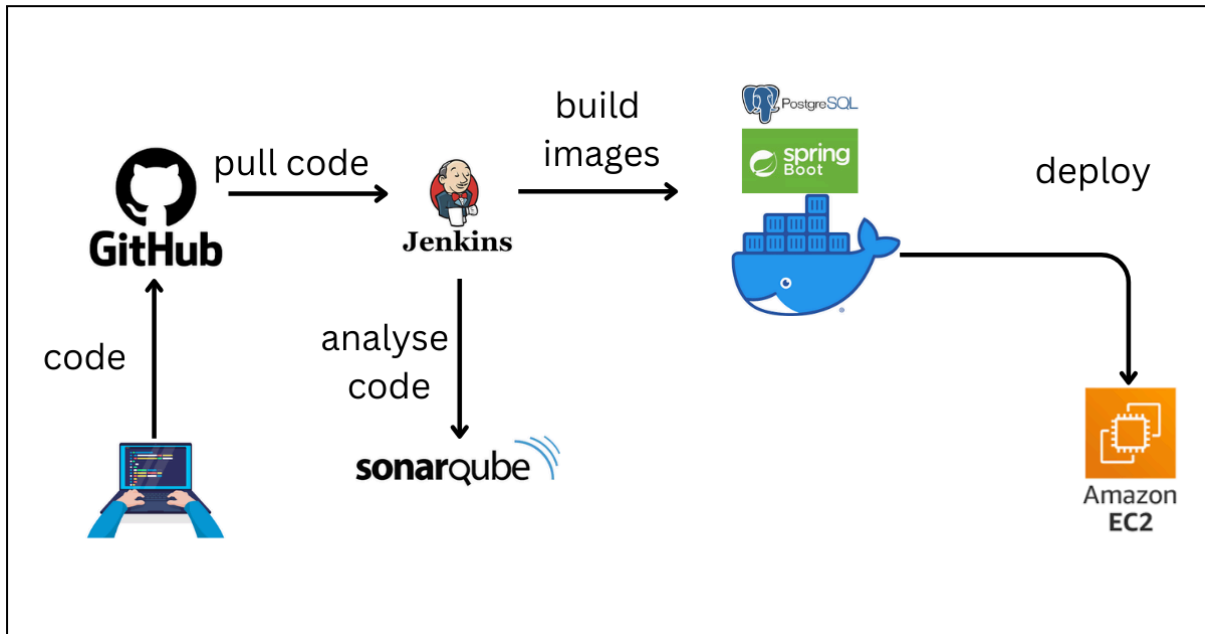
2.3 SonarQube: est un outil d'analyse statique du code qui permet de détecter les problèmes de qualité, les bugs, les vulnérabilités, garantissant ainsi un code plus fiable et maintenable.

2.4 Docker : Pour emballer les applications Spring Boot et les bases de données PostgreSQL sous forme de conteneurs.

2.5 Amazon EC2 : Pour héberger et exécuter les conteneurs Docker dans un environnement de production fiable.

3. Étapes de mise en œuvre du déploiement

Voici un aperçu des étapes nécessaires pour configurer et exécuter le déploiement automatique.



Les trois vms EC2 instances:

Instances (3/6) Info

Last updated less than a minute ago
⌂
Connect
Instance state ▾
Actions ▾
Launch instances
▾

All states ▾
< 1 >
⚙️

	Name ↗	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability
<input type="checkbox"/>	Jenkins1	i-04dde6aec13c54e23	⊖ Stopped 🔍 🔍	t2.medium	-	View alarms +	us-east-1d
<input checked="" type="checkbox"/>	JenkinsFree	i-0D9f08d795cc5a9fa	✔ Running 🔍 🔍	t2.micro	✔ 2/2 checks passed	View alarms +	us-east-1d
<input type="checkbox"/>	SonarQube	i-00f3bd8d75f28651c	⊖ Stopped 🔍 🔍	t2.medium	-	View alarms +	us-east-1d
<input type="checkbox"/>	docker1	i-0b46e48ab795a7a86	⊖ Stopped 🔍 🔍	t2.medium	-	View alarms +	us-east-1d
<input checked="" type="checkbox"/>	SonarQubeFree	i-0c7038c305a1015f9	✔ Running 🔍 🔍	t2.micro	🕒 Initializing	View alarms +	us-east-1d
<input checked="" type="checkbox"/>	DockerFree	i-0465293127d5956b6	✔ Running 🔍 🔍	t2.micro	✔ 2/2 checks passed	View alarms +	us-east-1a

3.1 Développeur : Soumet du code dans le dépôt **GitHub**.

3.2 Jenkins :

3.2.1 Créer une instance de VMS (Virtual Machine Server) EC2 pour installer Jenkins.

Instances (6) [Info](#) Last updated 9 minutes ago [Refresh](#) [Connect](#) [Instance state](#) [Actions](#) [Launch instances](#)

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

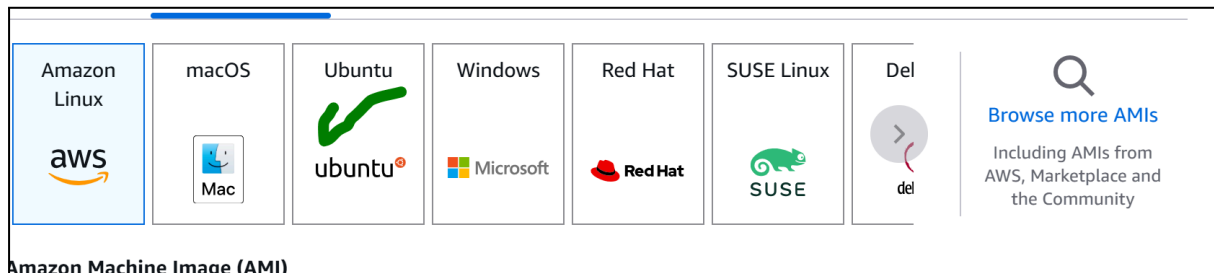
Name and tags [Info](#)

Name

JenkinsFree

Add additional tags

choisir un OS Ubuntu



choisir la configuration nécessaire pour de Ram,CPU,etc..

▼ Instance type [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Windows base pricing: 0.0162 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour

On-Demand RHEL base pricing: 0.026 USD per Hour

On-Demand Linux base pricing: 0.0116 USD per Hour

☐ All generations

[Compare instance types](#)

[Additional costs apply for AMIs with pre-installed software](#)

générer un SSH key est affecté l'instance ::

<input type="checkbox"/>	Name	Type	Created	Fingerprint	ID
<input type="checkbox"/>	SSH-KEY-Jenkins	rsa	2024/12/07 18:12 GMT+1	3c:cd:96:c4:85:7f:3a:35:3c:5f:...	key-0c7221f7b68145df2

Connection avec Jenkins from local Ubuntu

```
root@DESKTOP-Q81GVVI:/mnt/c/Users/AL AZAMI/Downloads# ssh -i SSH-KEY-Jenkins.pem ubuntu@3.84.125.237
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1018-aws x86_64)
```

```
*** System restart required ***
Last login: Sun Dec 15 16:16:16 2024 from 102.97.135.239
ubuntu@jenkinsFree:~$
```

installation de jenkins dans l'instance EC2

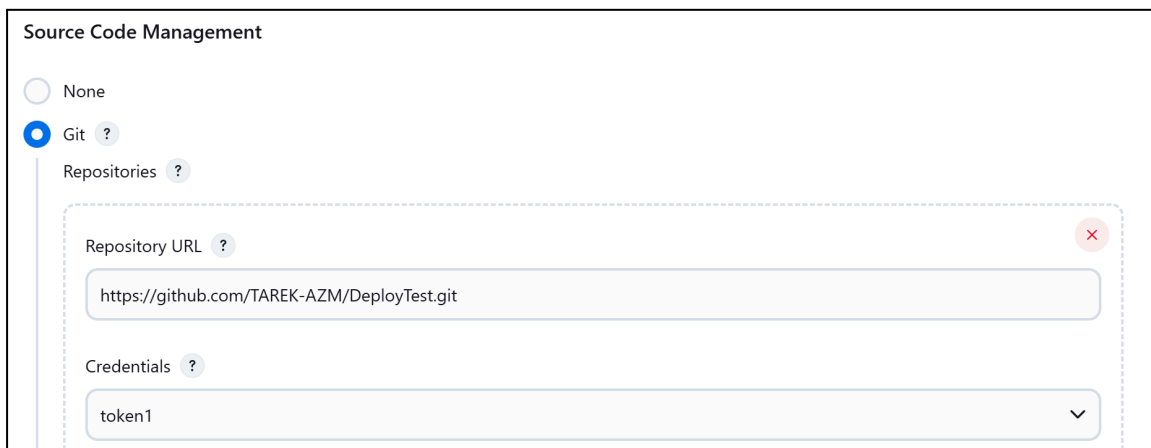
sudo apt update

sudo apt install openjdk-17-jre

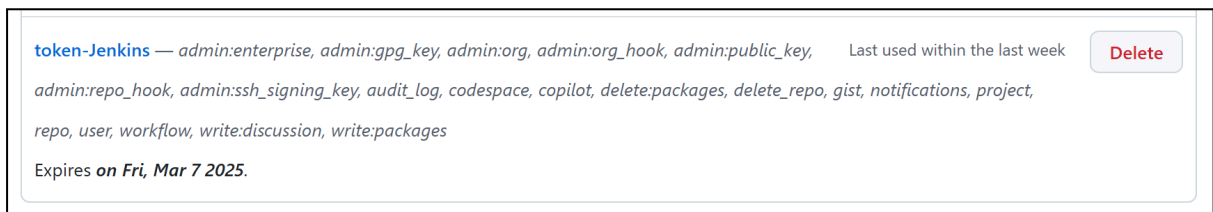
Après on installer Jenkins on suivent les démarche dans (Jenkins Docs)

3.2.2 Connection entre Jenkins et Github

ici on spécifie notre repository qui on va connecter avec jenkins
apres on crée un << Access Token >> dans github and inject dans jenkins
credentials comme suivent.



The screenshot shows the 'Source Code Management' configuration page in Jenkins. The 'Git' option is selected under 'Source Code Management'. Under 'Repositories', the 'Repository URL' field contains 'https://github.com/TAREK-AZM/DeployTest.git' and the 'Credentials' dropdown shows 'token1'.



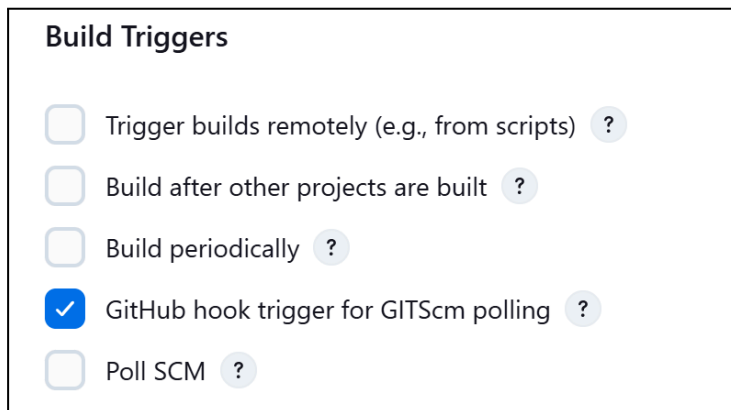
The screenshot shows a GitHub Access Token configuration. The token is named 'token-Jenkins' and has permissions for 'admin:enterprise', 'admin:gpg_key', 'admin:org', 'admin:org_hook', 'admin:public_key', 'admin:repo_hook', 'admin:ssh_signing_key', 'audit_log', 'codespace', 'copilot', 'delete:packages', 'delete_repo', 'gist', 'notifications', 'project', 'repo', 'user', 'workflow', 'write:discussion', and 'write:packages'. It was last used within the last week and expires on Friday, March 7, 2025. A 'Delete' button is visible.

après on spécifie le branch qui on va utiliser dans notre cas c'est **main** branch.



The screenshot shows the 'Add Repository' configuration page in Jenkins. Under 'Branches to build', the 'Branch Specifier (blank for 'any')' field contains '*/main'.

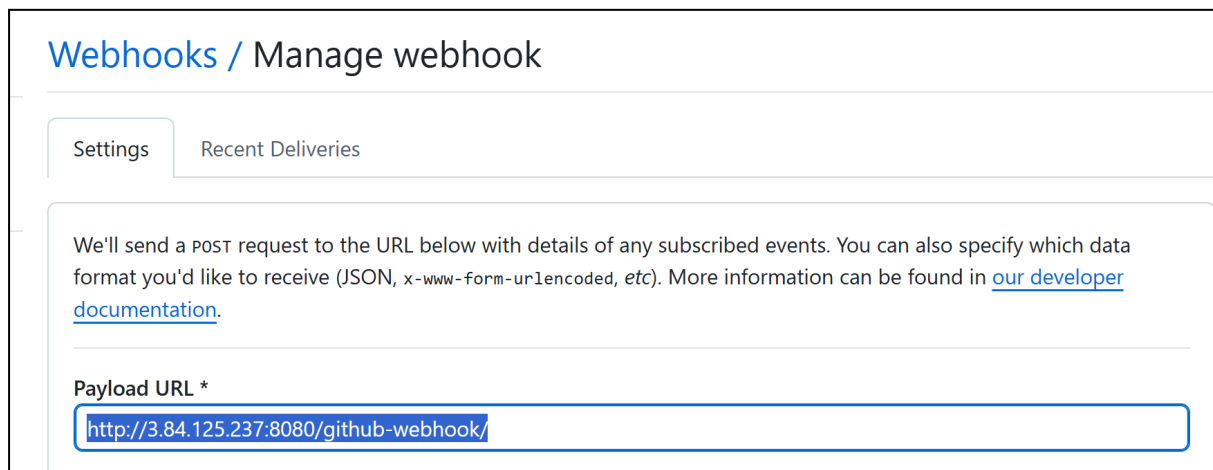
Après on coche cette option pour puisque jenkins reçoit les WebHook de github lorsque des mise a jour peuvent être faites sur le github.



Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

dans cette parti en github en créer un WebHook , et on spécifier URL de Jenkins qui reçoit les cette hooks qui envoyer par github.



Webhooks / Manage webhook

Settings Recent Deliveries


We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

ici en déclare dans quelle context le github va envoyer Webook vers Jenkins, dans notre cas je specifie dans chaque push, et pull request (si on a fait des merging)

3.2.3 Connection entre Jenkins et SonarQube EC2

Dans Jenkins ON installer plugins de SonarQube pour faire des configurations, on ajouter URL de SonarQube, et aon ajouter un Token de SonarQube ici pour connection



SonarQube servers

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

☒ Environment variables

SonarQube installations

List of SonarQube installations

Name

Server URL

Default is `http://localhost:9000`

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Name	Type	Project	Last use	Created	Expiration	
jenkins-token	Global		11 days ago	December 8, 2024	March 8, 2025	Revoke

☒ Prepare SonarQube Scanner environment ?

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled. Will default to the one defined in the SonarQube installation.

sonartoken1

≡

Execute SonarQube Scanner

✕

JDK ?

JDK to be used for this SonarQube analysis

(Inherit From Job)

Path to project properties ?

Analysis properties ?

sonar.projectKey=Onex-deploy-key
sonar.projectName=Onex-deploy-key
sonar.projectVersion=1.0
sonar.exclusions=**/*.java
sonar.sourceEncoding=UTF-8

3.2.4 Connection entre Jenkins et Docker EC2

Pour configurer cette partie on aller vers manage jenkins > system ou fait cette instruction :

dans la VMS EC2 de docker en connectant Docker VMS par SSH avec jenkins .

- créer un user "tarek" avec password
- je aller ver root user et fair cette instruction

On Jenkins server

```
sudo su Jenkins
ssh-keygen rsa (create new ssh)
#option 1
cat ~/.ssh/id_rsa.pub (Copy Public Key to Docker Server)
#option 2
ssh-copy-id username@3.91.8.203
#et on enter le password de user tarek
```

On Docker

```
sudo su root
sudo nano /etc/ssh/sshd_config

AuthorizedKeysFile .ssh/authorized_keys # uncomment this line
Match User tarek # add this command
PasswordAuthentication Yes # uncomment this line
#sudo systemctl daemon-reload
$ sudo systemctl restart ssh.service
```

- après on prend le private key de Docker , et créer dans Jenkins une nouvelle Credentials avec options SSH , et on injecte le private key.

Après on aller vers manage jenkins > system et on fait ça

ici on ajoute user et son password de Docker server.

Server Groups Center

Server Group List :
Create the server groups for your projects

Group Name ? ✕
Docker-servers

SSH Port ?
22

User Name ?
tarek

Password ?
.....

Server List :
add the server under this server group for your projects

Server Group: ✕
Docker-servers

Server Name ?
docker-1

Server IP ?
3.91.8.203

3.2.5 Exécuter les Commands Shell pour Build application

dans la partie de Build on Ajouter un Remote Shell on run les commands

```
cd /home/tarek/website  
docker-compose down # pour arrêter les containers qui on mode running
```



The screenshot shows the 'Remote Shell' configuration window in Jenkins. It includes a 'Disable' checkbox, a 'Target Server' dropdown menu with the selected value 'Docker-servers~~docker-1~~3.91.8.203', and a 'shell' text area containing the commands 'cd /home/tarek/website' and 'docker-compose down'.

on copie le contenu de notre jenkins workspace qui contient la dernière version de code de github.



The screenshot shows the 'Execute shell' configuration window in Jenkins. It includes a 'Command' text area with the command 'scp -r ./* tarek@3.91.8.203:~/website/'.

cette commande va faire nouvelle build pour générer .Jar file.



The screenshot shows the 'Remote Shell' configuration window in Jenkins, similar to the first one but with updated commands in the 'shell' text area: 'cd /home/tarek/website' and 'docker-compose build'.

après on run le docker-compose pour construire les images sprint boot app et postgres.



Docker-compose

```
version: "3.9"
services:
  postgres_db:
    container_name: postgres_db
    image: postgres:16.2
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: pgufc
      POSTGRES_DB: springboot
      PGDATA: /var/lib/postgresql/data/pgdata
    ports:
      - "5433:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - spring_network
    restart: unless-stopped
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres"]
      interval: 10s
      timeout: 5s
      retries: 5
```

```
spring_app:
  container_name: spring_app
  build:
    context: ./api
    args:
      - PROFILE=dev
      - APP_VERSION=1.0.0
  image: spring_app_image
  ports:
    - "9090:9090"

  environment:
    -
    SPRING_DATASOURCE_URL=jdbc:postgresql://postgres_db:5432/springboot
    - SPRING_DATASOURCE_USERNAME=postgres
    - SPRING_DATASOURCE_PASSWORD=pgufc
    - SPRING_PROFILES_ACTIVE=dev
  networks:
    - spring_network
  depends_on:
    postgres_db:
      condition: service_healthy
  restart: on-failure

# Optional: Add additional services as needed
# For example, you might want to add a mail service, frontend, etc

networks:
  spring_network:
    driver: bridge

volumes:
  postgres_data:
    driver: local
```

Dockerfile :

```
dockerfile d'app spring boot .
# Build stage
FROM maven:3.9.9 AS build
WORKDIR /build
COPY pom.xml .
RUN mvn dependency:go-offline
COPY src ./src
RUN mvn clean package -DskipTests

# Runtime stage
FROM openjdk:17-jdk-slim
ARG PROFILE=dev
ARG APP_VERSION=1.0.0

WORKDIR /app

# Copy the built jar from the build stage
COPY --from=build /build/target/*.jar /app/ServicePlatform.jar

# Expose the application port
EXPOSE 9090

# Environment variables with default values
ENV SPRING_PROFILES_ACTIVE=${PROFILE}
ENV APP_VERSION=${APP_VERSION}
ENV DB_URL=jdbc:postgresql://postgres_db:5432/springboot
ENV SPRING_DATASOURCE_URL=${DB_URL}
ENV SPRING_DATASOURCE_USERNAME=postgres
ENV SPRING_DATASOURCE_PASSWORD=pgufc

# Enable remote debugging (optional)
CMD ["java", "-jar", "ServicePlatform.jar"]
```

3.3 Docker:

3.3.1 Créer une instance de VMS (Virtual Machine Server) EC2 pour installer Docker (deployment server).

on suive la même démarche de jenkins pour construire cette instance

3.3.2 installer Docker et Docker-compose :

on Suive la démarche d'installation docker et docker-compose dans Ubuntu dans Docker Documentation

3.4 SonarQube:

3.4.1 Créer une instance de VMS (Virtual Machine Server) EC2 pour installer SonarQube.

on suive la même démarche de jenkins pour construire cette instance

3.4.2 installer SonarQube dans EC2 :

```
$ wget
https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-
24.12.0.100206.zip
#unzip the SonarQube.zip file
$ sudo apt install unzip
$ unzip sonarqube-24.12.0.100206.zip
$ cd sonarqube-24.12.0.100206/bin/linux-x86-64
$ ./sonar.sh console # running the SonarQube
```

3.4.3 Test la configuration

Résultat d'Analyse de la qualité du code via SonarQube d'après un Build .

