Mise en place d'une Infrastructure DevSecOps : GitLab, Kubernetes, Helm et SonarQube sur AWS

Objectifs :

- Mettre en place un pipeline GitLab CI/CD
- Déployer un Cluster Kubernetes sur AWS EC2
- Déployer une application sur Kubernetes cluster avec Helm-chart
- Déployer SonarQube

Prérequis

- Un compte GitLab
- Accès à AWS Academy Learner Lab
- Git installé sur votre machine

Architecture Globale

Partie 1 : Préparation

1.1 Accéder à AWS Academy

1. Lancez une instance Ec2
2. Connecter via ssh sur Powershell $ ssh -i <key.pem> ubuntu@<Public-IP>

1.2 Cloner le répertoire de github

Accédez à https://github.com/samarth-p/College-ERP

$ mkdir projects

$ cd projects

$ git clone https://github.com/samarth-p/College-ERP

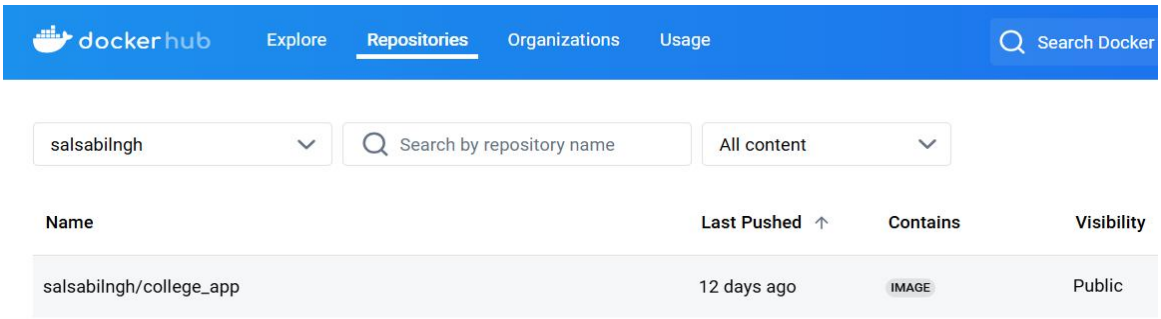
1.3 Création Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY . /app

RUN apt-get update && pip install --no-cache-dir -r requirements.txt

CMD ["sh","-c","python manage.py runserver 0.0.0.0:8000"]
```

1.4 Construire et pousser l' image Docker sur DockerHub

```
ubuntu@ip-172-31-18-77:~/project/College-ERP$ sudo docker build -t college_app .
[+] Building 14.5s (10/10) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                           0.0s
 => => transferring dockerfile: 218B                                                                           0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                            0.4s
 => [auth] library/python:pull token for registry-1.docker.io                                                  0.0s
 => [internal] load .dockerignore                                                                              0.0s
 => => transferring context: 2B                                                                                0.0s
 => [1/4] FROM docker.io/library/python:3.11-slim@sha256:370c586a6ffc8c619e6d652f81c094b34b14b8f2fb9251f092de23f1  0.2s
 => => resolve docker.io/library/python:3.11-slim@sha256:370c586a6ffc8c619e6d652f81c094b34b14b8f2fb9251f092de23f1  0.0s
 => => sha256:533df8de4eb83c1265bfb78f65b87dc9b45afbca418babfbed108c5c2497a322 5.29kB / 5.29kB                  0.0s
 => => sha256:370c586a6ffc8c619e6d652f81c094b34b14b8f2fb9251f092de23f16e299b78 9.13kB / 9.13kB                  0.0s
 => => sha256:b88b6b440e33679874d4f16011be1b5350406fb923b847d19d58d8bcd8099896 1.75kB / 1.75kB                  0.0s
 => [internal] load build context                                                                              0.4s
 => => transferring context: 22.56MB                                                                           0.4s
 => [2/4] WORKDIR /app                                                                                         0.2s
 => [3/4] COPY . /app                                                                                          0.2s
 => [4/4] RUN apt-get update && pip install --no-cache-dir -r requirements.txt                                11.9s
 => exporting to image                                                                                         1.3s
 => => exporting layers                                                                                        1.3s
 => => writing image sha256:7ff2e1d7c7ba52f80ca32288e187dd7f03d8589226ebb48ce3e73a3f50003b12                   0.0s
 => => naming to docker.io/library/college_app                                                                 0.0s
ubuntu@ip-172-31-18-77:~/project/College-ERP$
```

```
ubuntu@ip-172-31-18-77:~/project/College-ERP$ sudo docker images
REPOSITORY                TAG              IMAGE ID       CREATED         SIZE
college_app               latest           a4c03db98f88   12 days ago     242MB
```



Partie 2 : Créer et déployer un cluster Kubernetes on AWS EC2

2.1 Création du cluster K3S

On commence par ajouter 2 autres instances EC2 pour fonctionner en tant que ‘worker-nodes’

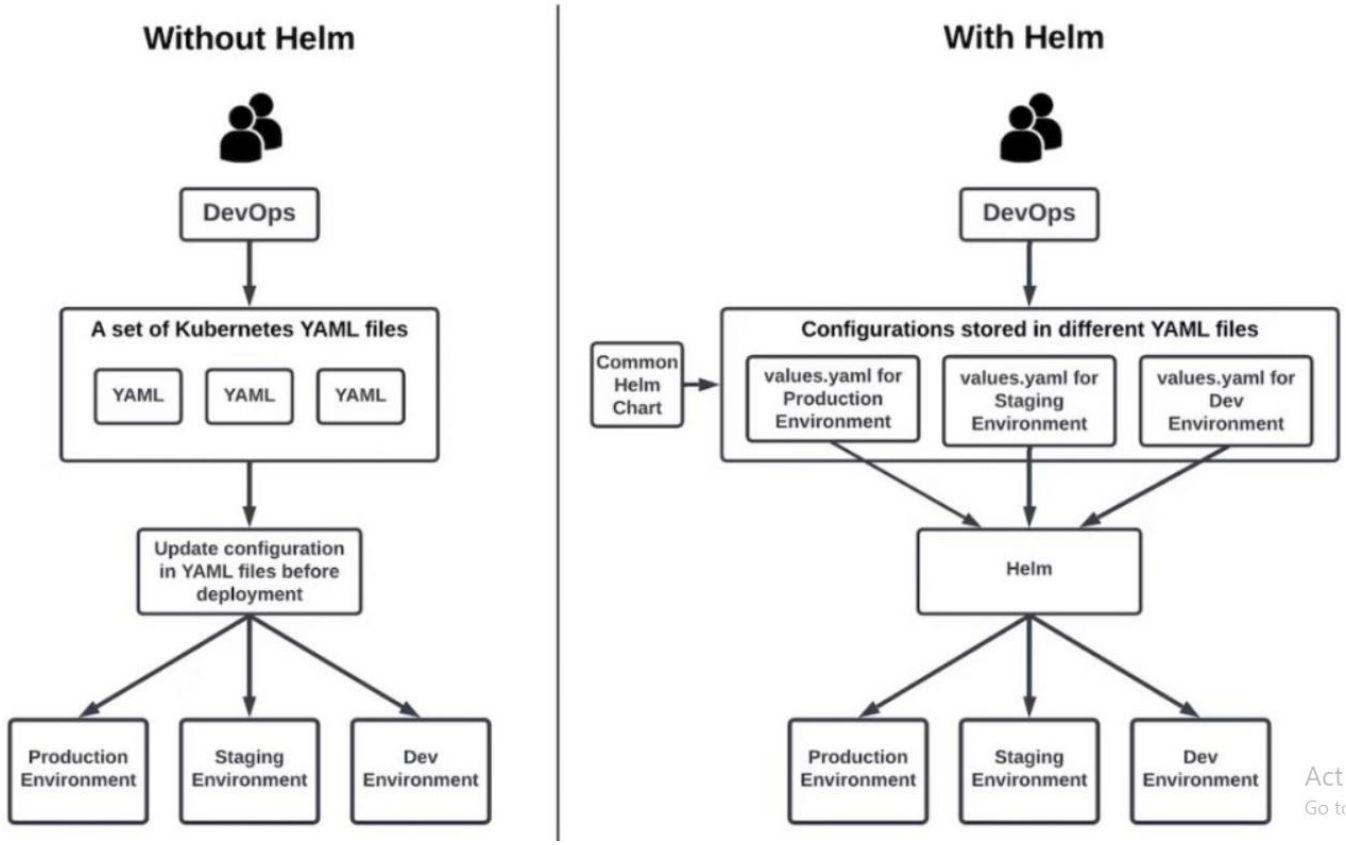
Et sur l’instance master on éxécute cette commande pour installer le cluster k3s
$ curl -sfL https://get.k3s.io | sh –
une fois terminé , on ajoute les nœuds à notre cluster kubernetes

```
ubuntu@ip-172-31-29-80:~$ curl -sfL https://get.k3s.io | K3S_NODE_NAME=k3s-worker-01 K3S_URL=https://172.31.18.77:6443 K3S_TOKEN=K1017bf54d58adefa624cedc469aff6157cda1f8c
4a5a8f18058e43145f798e07bb::server:45c8b3e7fdc704ef1d92f93373b44b34 sh -
[INFO]  Finding release for channel stable
[INFO]  Using v1.30.6+k3s1 as release
[INFO]  Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.30.6+k3s1/sha256sum-amd64.txt
[INFO]  Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.30.6+k3s1/k3s
[INFO]  Verifying binary download
[INFO]  Installing k3s to /usr/local/bin/k3s
[INFO]  Skipping installation of SELinux RPM
[INFO]  Creating /usr/local/bin/kubectl symlink to k3s
[INFO]  Creating /usr/local/bin/crictl symlink to k3s
[INFO]  Creating /usr/local/bin/ctr symlink to k3s
[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
[INFO]  Creating uninstall script /usr/local/bin/k3s-agent-uninstall.sh
[INFO]  env: Creating environment file /etc/systemd/system/k3s-agent.service.env
[INFO]  systemd: Creating service file /etc/systemd/system/k3s-agent.service
[INFO]  systemd: Enabling k3s-agent unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s-agent.service → /etc/systemd/system/k3s-agent.service.
[INFO]  systemd: Starting k3s-agent
```

```
ubuntu@ip-172-31-18-77:~/project/College-ERP$ sudo kubectl get nodes
NAME              STATUS   ROLES                  AGE    VERSION
ip-172-31-18-77   Ready    control-plane,master   132m   v1.30.6+k3s1
k3s-worker-01     Ready    <none>                 112m   v1.30.6+k3s1
k3s-worker-02     Ready    <none>                 108m   v1.30.6+k3s1
ubuntu@ip-172-31-18-77:~/project/College-ERP$ sudo kubectl get pods
NAME                                    READY   STATUS    RESTARTS      AGE
collegeapp-helm-chart-7dd9f8c676-cgj9c   1/1     Running   4 (18m ago)   76m
ubuntu@ip-172-31-18-77:~/project/College-ERP$ sudo kubectl get svc
NAME                    TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
collegeapp-helm-chart   NodePort    10.43.160.13   <none>        8000:31183/TCP   82m
kubernetes              ClusterIP   10.43.0.1      <none>        443/TCP          133m
ubuntu@ip-172-31-18-77:~/project/College-ERP$
```

2.2 Installation Helm Chart

Helm est un gestionnaire de paquets pour Kubernetes. Il permet de simplifier le déploiement, la gestion et la mise à jour d'applications sur un cluster Kubernetes en utilisant des charts (modèles de ressources Kubernetes préconfigurés et personnalisables).

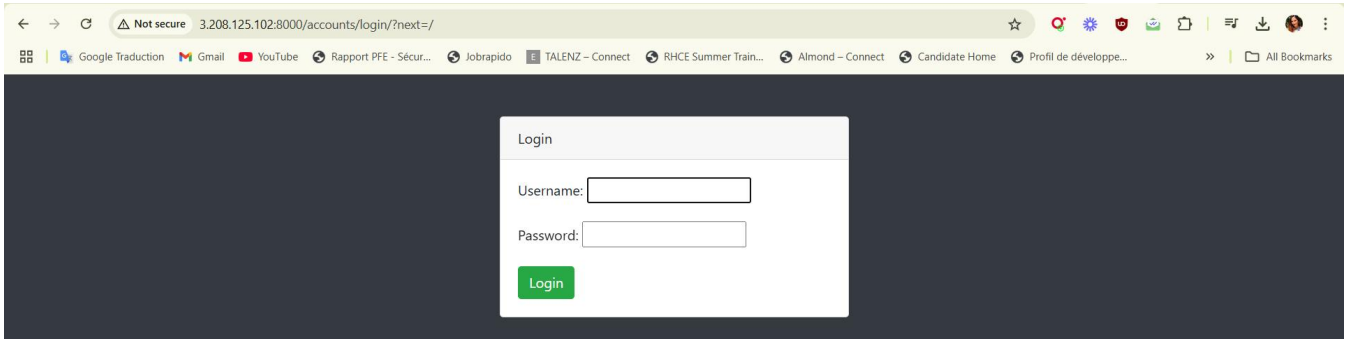


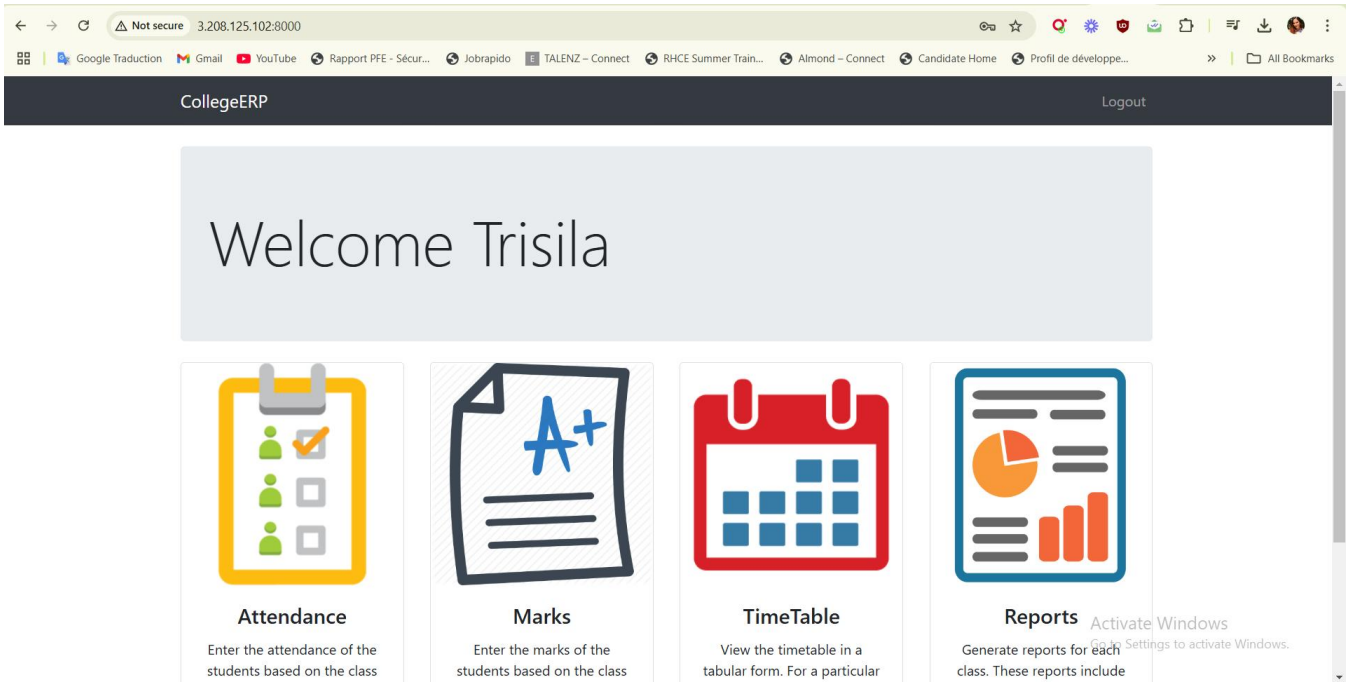

```
$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
$ chmod 700 get_helm.sh
$ ./get_helm.sh
```

```
ubuntu@ip-172-31-18-77:~$ helm version
version.BuildInfo{Version:"v3.16.3", GitCommit:"cfd07493f46efc9debd9cc1b02a0961186df7fdf", GitTreeState:"clean", GoVersion:"go1.22.7"}
ubuntu@ip-172-31-18-77:~$
```

```
ubuntu@ip-172-31-18-77:~/project/College-ERP$ ls helm-chart/
Chart.yaml  charts  templates  values.yaml
```

2.3 Déploiement en Kubernetes

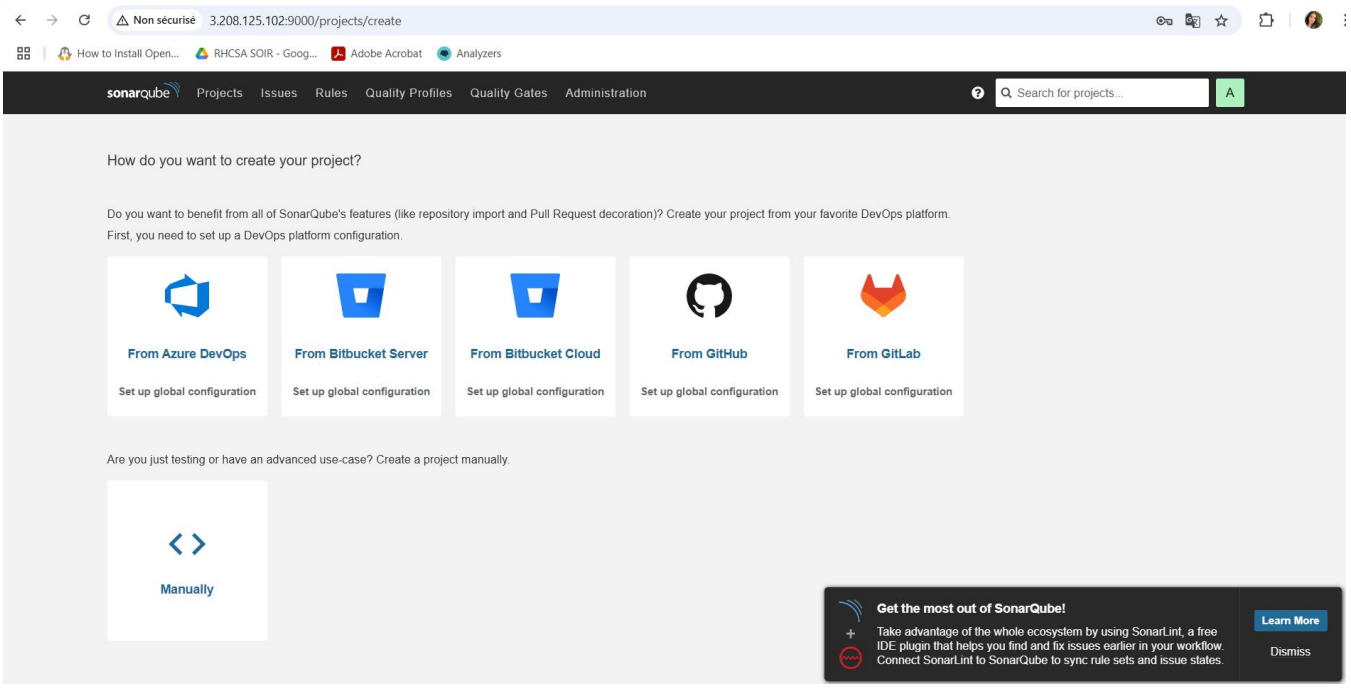




Partie3: Déploiement d'un SonarQube

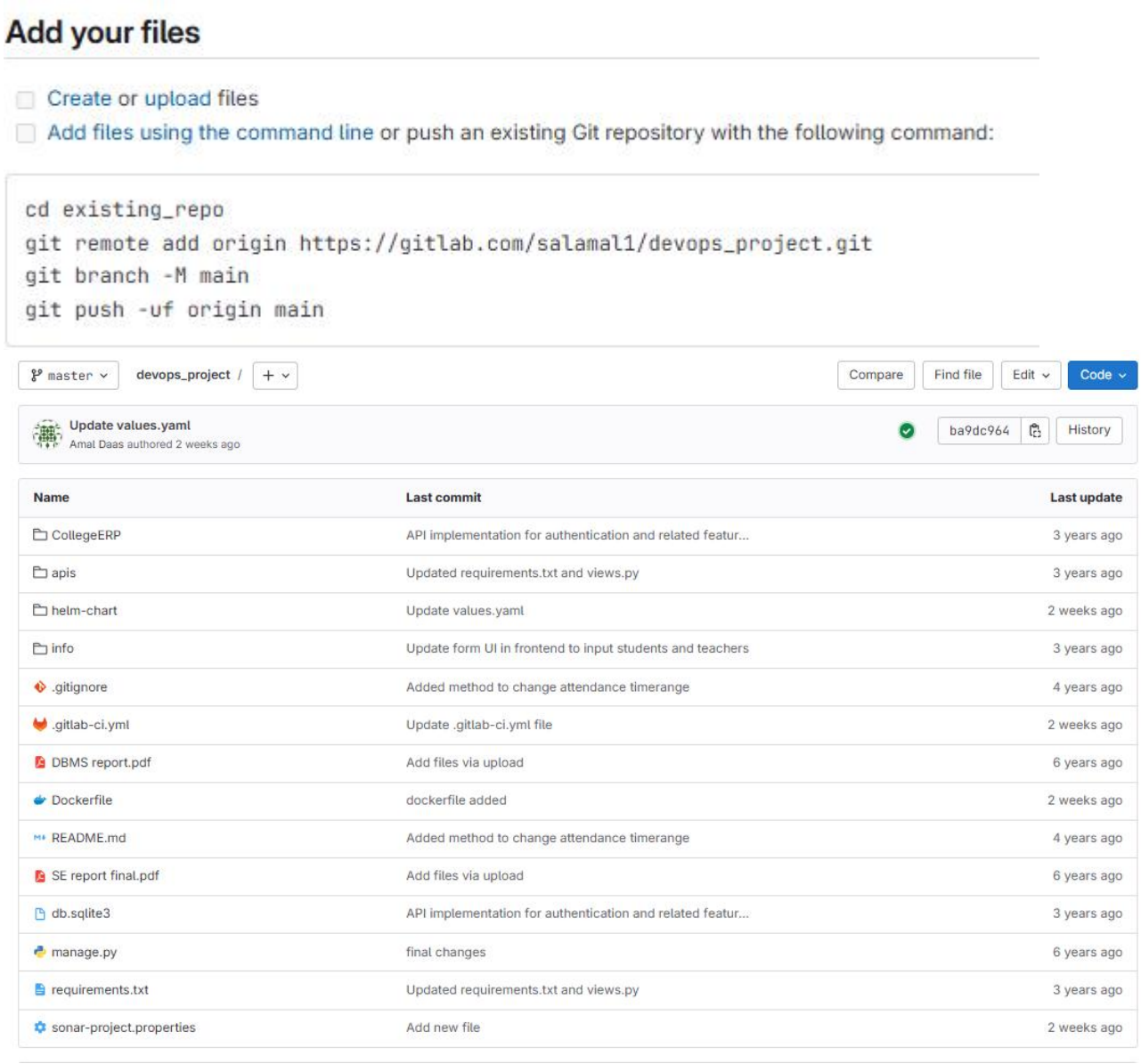
SonarQube est une plateforme open-source utilisée pour l'analyse de la qualité du code et la sécurité dans les projets logiciels. Elle est particulièrement utile dans une démarche DevOps, car elle permet l'analyse statique du code , l'intégration continue , le suivi des métriques .

```
ubuntu@ip-172-31-18-77:~/project/College-ERP$ sudo docker run --name sonar -d -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
6414378b6477: Pull complete
17da8ec43a12: Pull complete
d12988e90d61: Pull complete
f4d133ca2b7f: Pull complete
143733ae87a4: Pull complete
8438621478bb: Pull complete
3d0284140b24: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:c337c407849de45a727f09dbd875779ad7b5784e0b02b096c1f8cd72e27a9fdc
Status: Downloaded newer image for sonarqube:lts-community
b4f1756a762f2c577716e54ddf5039855d64767e37e6ba28f398bd6b349c9da2
ubuntu@ip-172-31-18-77:~/project/College-ERP$
```



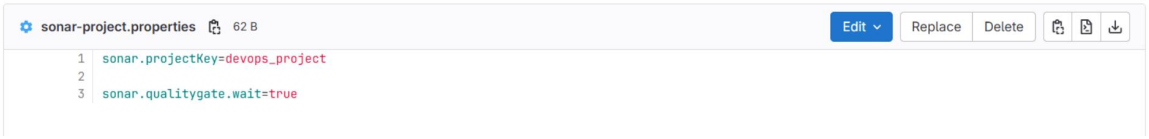
Partie4: Création d'un pipeline Gitlab CI/CD

1.1 Pousser le dépôt vers gitlab



1.2 Créer le pipeline initial pour la compilation de SonarQube

➢ Permet de vérifier la qualité et la sécurité du code source



Pipeline #1575391120 Passed for 757c0432: Add new file

Pipeline syntax is correct. Learn more

Edit | Visualize | Validate NEW | Full configuration

CI/CD Catalog | Help

```yaml
stages:
  - sonarqube-check
  - build
  - push
  - deploy

sonarqube-check:
  stage: sonarqube-check
  image:
    name: sonarsource/sonar-scanner-cli:latest
    entrypoint: [""]
  variables:
    SONAR_USER_HOME: "${CI_PROJECT_DIR}/.sonar"  # Defines the location of the analysis task cache
    GIT_DEPTH: "0"  # Tells git to fetch all the branches of the project, required by the analysis task
  cache:
    key: "${CI_JOB_NAME}"
    paths:
      - .sonar/cache
  script:
    - sonar-scanner
  allow_failure: true
  only:
    - master
```

On vérifie l'état du pipeline

salamal / devops_project / Pipelines / #1575391120

Add new file

Delete

Passed **Amal Daas** created pipeline for commit 757c0432 1 minute ago, finished just now

For master

latest 1 job 1.33 1 minute 19 seconds, queued for 3 seconds

Pipeline | Jobs 1 | Tests 0

sonarqube-check

sonarqube-check

```
205 Saving cache for successful job                                   00:08
206 Creating cache sonarqube-check-non_protected...
207 .sonar/cache: found 55 matching artifact files and directories
208 Uploading cache.zip to https://storage.googleapis.com/gitlab-com-runners-cache/project/65103033/sonarqube-check-non_protected
209 Created cache
210 Cleaning up project directory and file based variables              00:01
211 Job succeeded
```

On accéde à l'interface de SonarQube pour vérifier le déploiement du stage "sonarqube-check"

Not secure 3.208.125.102:9000/projects

sonarqube | Projects | Issues | Rules | Quality Profiles | Quality Gates | Administration

Search for projects...

My Favorites | All

Search by project name or key

Create Project

3 project(s)

Perspective: Overall Status | Sort by: Name

Filters

Quality Gate

Passed 2

Failed 0

Reliability (Bugs)

A rating 0

B rating 0

C rating 2

D rating 0

E rating 0

Security (Vulnerabilities)

A rating 2

B rating 0

C rating 0

D rating 0

E rating 0

Security Review (Security Hotspots)

devops

Project's Main Branch is not analyzed yet. Configure analysis

devops-_ Passed — Last analysis: 18 days ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
21 C	0 A	0.0% E	73 A	0.0%	15.2%	3.9k HTML, Py...

devops1 Passed — Last analysis: 17 days ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
21 C	0 A	0.0% E	73 A	0.0%	15.2%	3.9k HTML, Py...

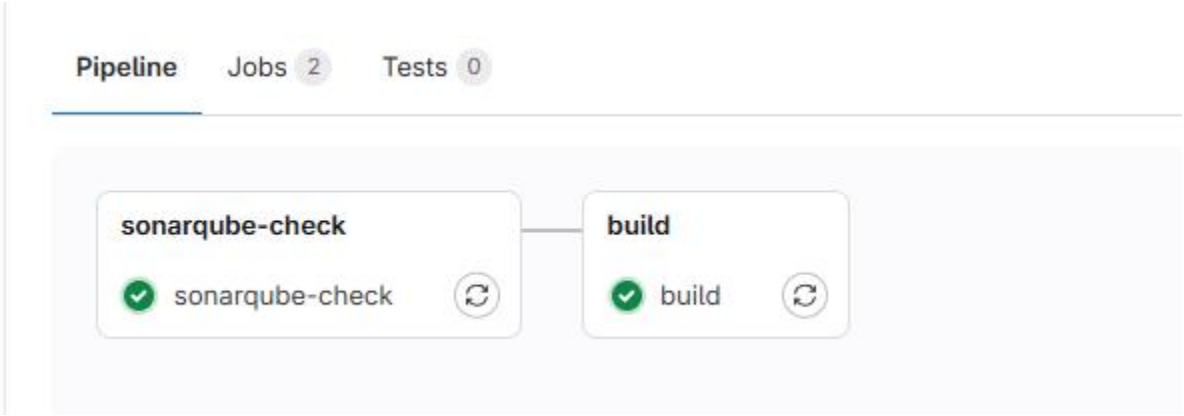
3 of 3 shown

1.3 Configuration de job Build

- Permet la construction de l'image Docker de l'application

```
build:
    stage: build
    image: docker:latest
    services:
        - docker:dind
    script:
       - echo "Building Docker Image"
       - docker build -t $DOCKER_IMAGE .
       - docker save -o docker_image.tar $DOCKER_IMAGE
    artifacts:
        paths:
            - docker_image.tar
    only:
      - master
```

Vérification du pipeline

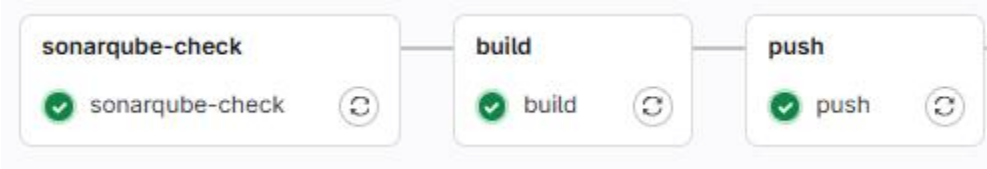


1.4 Configuration de job Push

- Permet de pousser l'image Docker sur DockerHub

```
push:
  stage: push
  image: docker:latest
  services:
    - docker:dind
  dependencies:
    - build
  script:
    - echo "login in DockerHub"
    - docker login -u $DOCKER_USERNAME -p $DOCKER_HUB_TOKEN
    - echo "loading docker image"
    - docker load -i docker_image.tar
    - echo "tagging docker images"
    - docker tag $DOCKER_IMAGE $DOCKER_USERNAME/$DOCKER_IMAGE
    - echo "pushing docker image"
    - docker push $DOCKER_USERNAME/$DOCKER_IMAGE
  only:
    - master
```

Vérification de job push

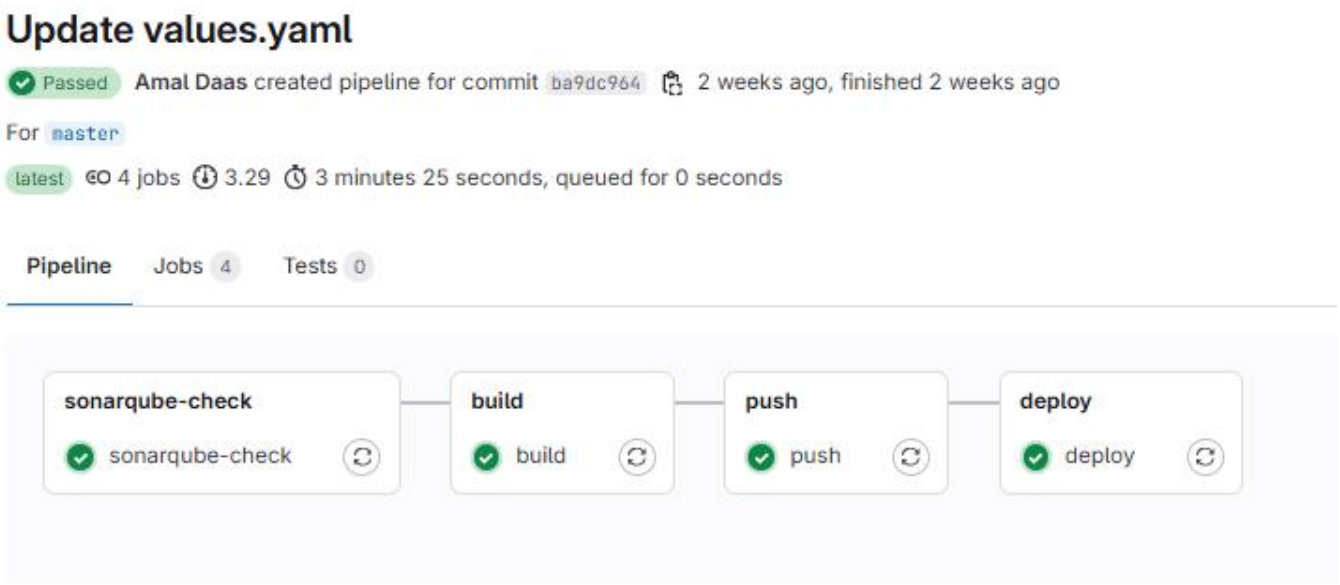


1.6 Déploiement de l'application (Job Deploy)

- Permet le déploiement de l'application dans kubernetes en utilisant Helm

```
deploy:
  stage: deploy
  script:
    - echo "deploy to kubernetes cluster using helm"
    - helm upgrade --install collegeapp ./helm-chart --set image.repository=$DOCKER_USERNAME/DOCKER_IMAGE --set image.tag=latest
  tags:
    - master_runner
```

Vérification du pipeline



Ce pipeline est conçu pour automatiser le processus de développement , de test , de conteneurisation et de déploiement de l'application , s'intégrant ainsi parfaitement dans une approche Devops .