DevSecOps Project

Supervisor :
Prof. Driss ALLAKI

Team :
Mouad Tigmouti Taha Safa
Khalid chbail Ilyas Kotbi

Presentation plan

Section 1

- Team Topology
- SCRUM

Section 2

- Backend CI/CD Pipeline
- Frontend CI/CD Pipeline
- Optimizing Pipelines

Section 3

- CI/CD Pipeline for Automated Security Testing
- SAST, SCA, CS, DAST

Section 4

- Creation of kubernetes cluster
- Creation of namespace
- Creation of necessary YAML files
- Execution of YAML files

1. Team Topology
2. Development cycle of this project
3. Project deliverable

Section 1

1. Team Topology

In this project, we chose as a team topology : The **Dev** and **Ops** Collaboration.
That is because it promotes seamless collaboration between development and operations teams, leading to improved efficiency and faster delivery of software products.

By working together, both teams can leverage their specialized skills and knowledge, resulting in better decision-making and a more agile development process. This model also encourages a shared goal of delivering reliable and frequent changes, aligning the efforts of both teams and improving overall software quality

1. Team Topology
2. Development cycle of this project
3. Project deliverable

Section 1

2. Development cycle of this project

Roles:

1-CI/CD Engineer:

Setting up CI/CD pipelines for the backend and frontend using tools like GitLab CI or GitHub Actions.Optimizing CI/CD processes to improve efficiency and reduce deployment time.Automating tests throughout the CI/CD pipeline, including unit tests, integration tests, and end-to-end tests.Configuring and managing build artifacts and releases.

2. Development cycle of this project

Roles:

2-Security Engineer :

Strengthening security measures throughout the CI/CD process.Implementing pre-commit hooks to enforce code quality and security standards.Integrating a "Vulnerability Management Tool" to scan for potential security vulnerabilities in the codebase.Conducting security audits and risk assessments to identify and address potential vulnerabilities.

2. Development cycle of this project

Roles:

3-DevOps Engineer:

Creating a local Kubernetes cluster for development and testing purposes.Deploying the database, backend, and frontend applications to the Kubernetes cluster.Verifying the created Kubernetes objects, such as pods, services, and deployments.Implementing scaling and monitoring strategies for the deployed applications.

2. Development cycle of this project

Evenements:

1-Sprint Planning Meeting: The development team gathers to select items from the Product Backlog to include in the upcoming sprint and define sprint goals.

2-Daily Stand-up: A short daily meeting where the development team shares progress, discusses any obstacles, and plans for the day.

3-Iterative Development: Developers work in iterations, focusing on specific features or assigned tasks. They follow Agile development principles such as continuous delivery and adaptability to changes.

4-Sprint Review: At the end of each sprint, the development team demonstrates completed features to stakeholders and gathers their feedback. Adjustments are made if needed to improve the product.

5-Sprint Retrospective: A meeting to evaluate the completed sprint, identify strengths and areas for improvement, and propose corrective actions for future sprints.

1. Team Topology
2. Development cycle of this project
3. Project deliverable

Section 1

3. Project deliverable:

Product backlog:

The backlog capture was done using Jira, a widely-used project tracking tool. I created a new product backlog in Jira for our project, which allows us to centralize and manage all the requirements, features, and tasks associated with our product. In this backlog, we expressed the needs to be addressed in the form of user stories and summarized the work to be done in phases 2, 3, and 4 as three main tasks.

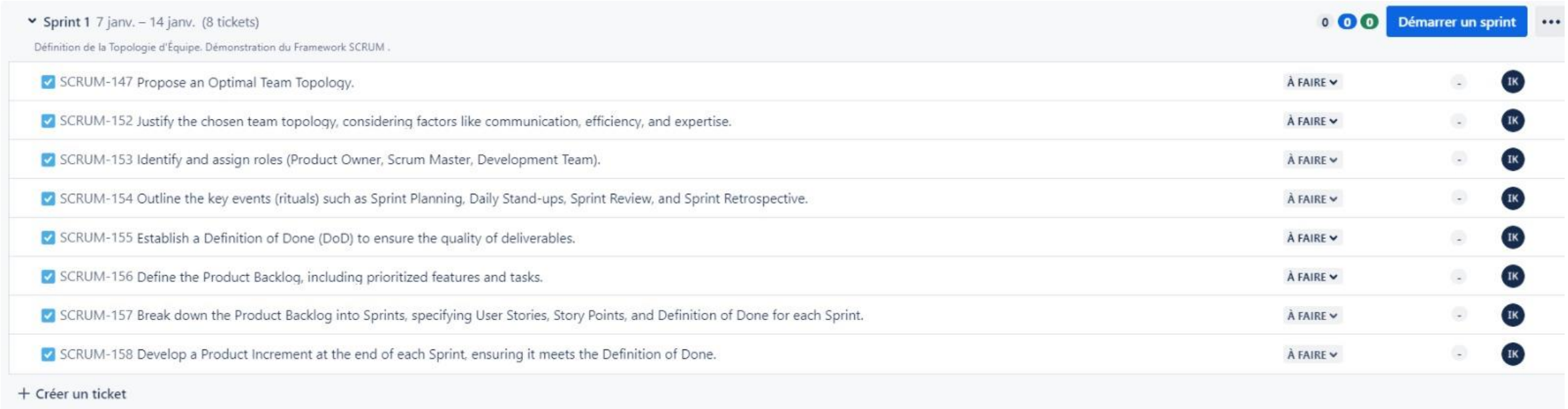
Backlog (19 tickets) — 78 — 0 — 0 — Créer un sprint

SCRUM-28 As a DevOps engineer, I want to use a specific GitLab runner (with, for example, a Docker executor) for faster and efficient pipeline execution.	À FAIRE	2	M
SCRUM-29 As a CI/CD manager, I want to improve control over pipeline execution using rules.	À FAIRE	5	M
SCRUM-30 As a user, I want to receive clear and understandable notifications in the event of a major security incident or the correction of vulnerabilities, to stay aware of the s...	À FAIRE	3	T
SCRUM-31 As a developer, I want to implement automated security tests such as 'Dynamic Application Security Testing (DAST)' in the CI/CD pipeline to assess the security of t...	À FAIRE	5	T
SCRUM-102 As a user, I want the application to be resilient to errors, with automatic recovery in case of failures, leveraging Kubernetes' built-in error handling, for an uninterru...	À FAIRE	3	KC
SCRUM-103 As a user, I want the application to be highly available, with minimal interruptions, through the use of Kubernetes, for a seamless user experience	À FAIRE	3	KC
SCRUM-34 As a user, I want the application to scale seamlessly based on demand	À FAIRE	5	KC
SCRUM-35 As a user, I want to be notified of input errors or successful actions during the creation, updating, or deletion of a tutorial.	À FAIRE	2	T
SCRUM-36 As a backend developer, I want to set up a CI/CD pipeline with GitLab CI (or GitHub Actions) to automate the build, packaging, and deployment of the application's...	À FAIRE	8	M
SCRUM-37 As a frontend developer, I want to set up a CI/CD pipeline with GitLab CI (or GitHub Actions) to automate the build, packaging, and deployment of the application'...	À FAIRE	8	M
SCRUM-38 As a developer, I want to create and use a specific GitLab runner (or equivalent for GitHub Actions) to execute pipelines optimally.	À FAIRE	5	M
SCRUM-39 As a developer, I want to use techniques such as pipeline rules, caching, and generic jobs to optimize CI/CD pipelines.	À FAIRE	5	M
SCRUM-40 As a developer, I want to add 'Pre-Commit Hooks' to ensure code security hygiene in the version control system (VCS).	À FAIRE	3	T
SCRUM-41 As a developer, I want to extend the CI/CD pipeline to include automated tests and security scans throughout the deployment process.	À FAIRE	8	T
SCRUM-42 As a developer, I want to use monitoring tools to supervise the infrastructure, MySQL database, backend, and frontend of the application, as well as corresponding ...	À FAIRE	5	T
SCRUM-43 As a developer, I want to create a local Kubernetes cluster using Minikube or an equivalent, and deploy the application into this cluster.	À FAIRE	8	KC
SCRUM-159 Setting up CI/CD pipelines for the backend and frontend of the application using GitLab CI (or GitHub Actions) and optimizing the processes.	À FAIRE	-	M
SCRUM-160 Strengthening security and automating tests throughout the CI/CD process, including the addition of pre-commit hooks and implementation of a "Vulnerability M...	À FAIRE	-	T
SCRUM-161 Creating a local Kubernetes cluster, deploying the database, backend, and frontend of the application, and verifying the created Kubernetes objects.	À FAIRE	-	KC

3. Project deliverable:

Sprints Backlogs(Kotbi Ilyas):

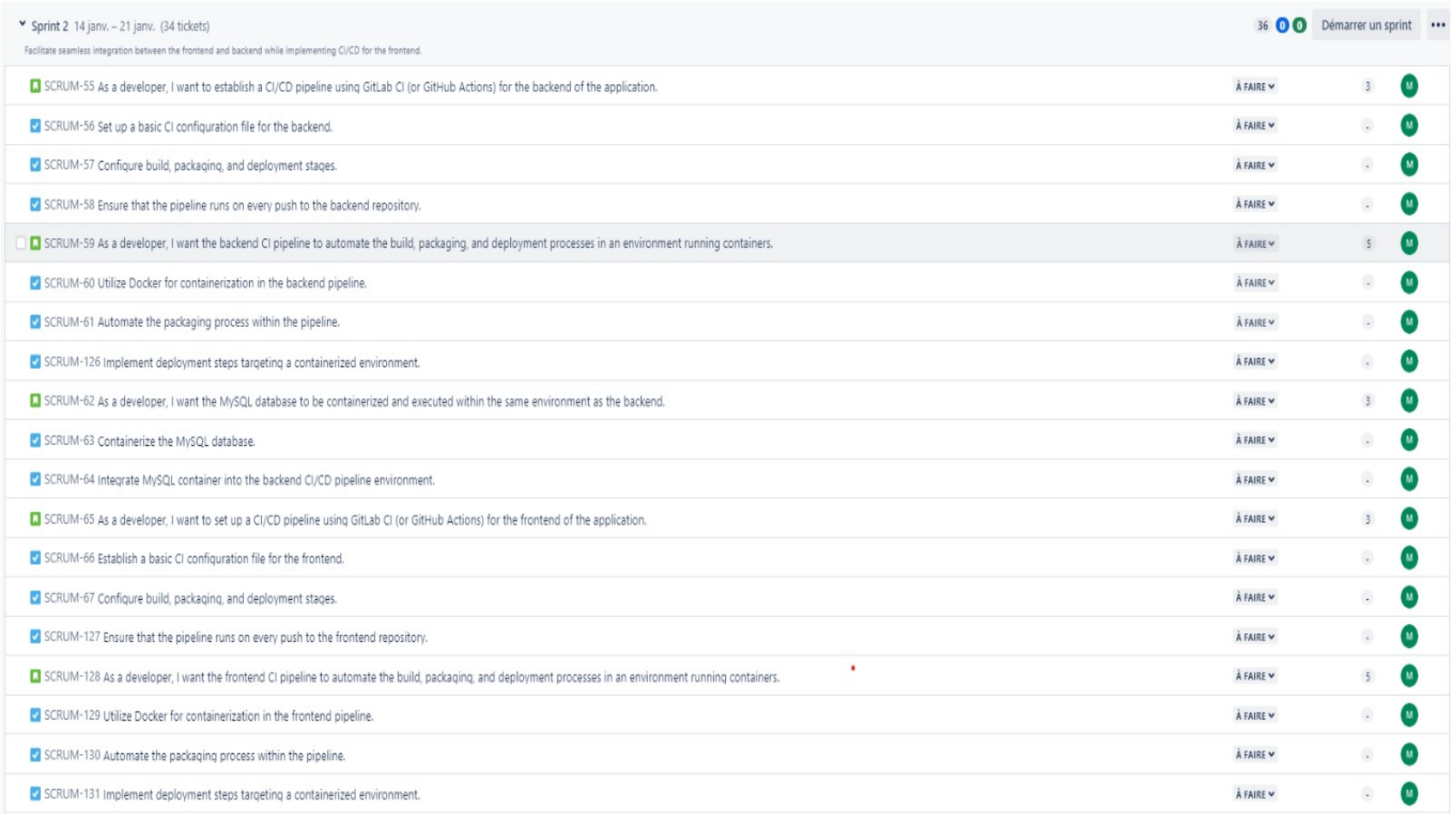
In this sprint, the tasks were assigned to a single team member who is responsible for their completion. This approach promotes individual focus and accountability for each task. The first objective of this sprint was to propose a team topology that would enable engineers in our company to be highly productive in developing and operating the application, as well as other similar applications. The team member assigned to this sprint conducted thorough research, collected relevant data, and analyzed different options to determine the best team configuration suited to our needs. The second objective was to demonstrate the use of the SCRUM framework to our development team. The team member detailed the development phase's process according to SCRUM, adhering to the specific roles and events (rituals). Additionally, they created the various deliverables of the project, including the product backlog, sprint backlogs, and the criteria for sprint completion (DoDs).



3. Project deliverable:

Sprints Backlogs (Tigmouti Mouad):

In this sprint, the main focus was on setting up CI/CD pipelines for the backend and frontend of the application using either GitLab CI or GitHub Actions. The first task involved implementing a CI/CD pipeline for the backend, while the second task focused on setting up a similar pipeline for the frontend. Both pipelines were required to automate the build, packaging, and deployment processes within a containerized environment. Additionally, the MySQL database was to be containerized and executed within the same environment. To optimize these pipelines, several techniques were recommended. The first technique involved creating and utilizing a custom GitLab runner, specifically using a Docker executor. This approach provided greater control and customization options for the pipeline execution. The second technique involved using rules to improve the execution control of the pipeline and its jobs, allowing for conditional execution based on specific criteria.



3. Project deliverable:

Sprints Backlogs(Tigmouti Mouad):

SCRUM-132 As a developer, I want the MySQL database to be containerized and executed within the same environment as the frontend.	À FAIRE	3	M
SCRUM-133 Containerize the MySQL database(frontend).	À FAIRE	-	M
SCRUM-134 Integrate MySQL container into the frontend CI/CD pipeline environment(frontend).	À FAIRE	-	M
SCRUM-135 As a developer, I want to create and utilize a specific GitLab runner for the backend pipeline (using, for example, a Docker executor).	À FAIRE	3	M
SCRUM-136 Configure a dedicated GitLab runner for the backend.	À FAIRE	-	M
SCRUM-137 Use a Docker executor to enhance isolation and flexibility.	À FAIRE	-	M
SCRUM-138 As a developer, I want to use rules for better control of pipeline and job execution.	À FAIRE	3	M
SCRUM-139 Implement rules for conditional execution based on specific conditions.	À FAIRE	-	M
SCRUM-140 Ensure that jobs run only when necessary to optimize the pipeline.	À FAIRE	-	M
SCRUM-141 As a developer, I want to employ caching techniques for more efficient management of project dependencies.	À FAIRE	5	M
SCRUM-142 Identify and cache dependencies where applicable.	À FAIRE	-	M
SCRUM-143 Implement cache restoration mechanisms to speed up subsequent builds.	À FAIRE	-	M
SCRUM-144 As a developer, I want to use generic Jobs (with includes and extends) for better job reuse across different projects.	À FAIRE	3	M
SCRUM-145 Identify common tasks that can be abstracted into generic jobs.	À FAIRE	-	M
SCRUM-146 Utilize includes and extends to incorporate generic jobs into specific projects.	À FAIRE	-	M

The third technique was to implement caching, which would optimize the management of project dependencies and reduce build times by reusing cached dependencies. Lastly, the fourth technique was to utilize generic jobs with includes and extends, enabling better job reuse between different projects and promoting code reusability. Overall, this sprint aimed to establish efficient and automated CI/CD pipelines for both the backend and frontend, while incorporating optimization techniques to enhance the development and deployment processes.

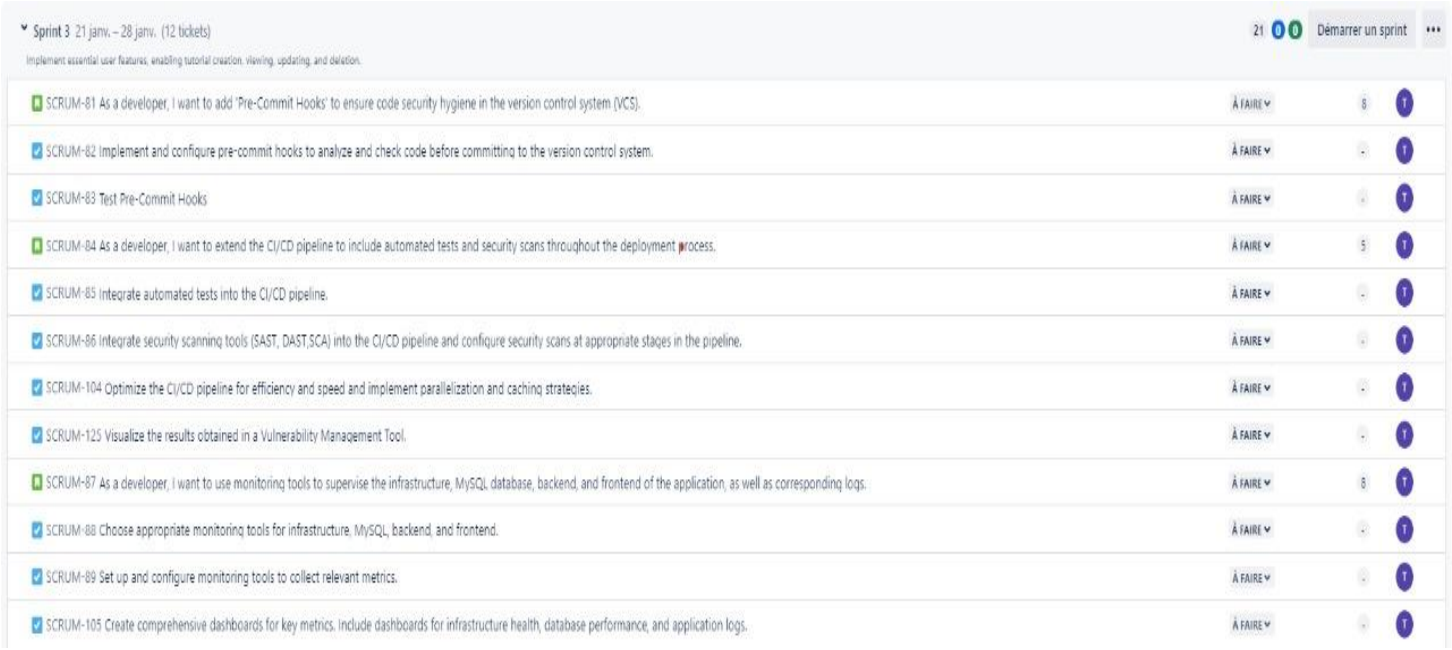
3. Project deliverable:

Sprints Backlogs (Safa Taha):

In this sprint, the primary focus was on improving the security and monitoring aspects of the application. Key tasks included implementing "Pre-Commit Hooks" in Git for enhanced security practices during code commits. The sprint also aimed to broaden the Continuous Integration/Continuous Deployment (CI/CD) pipeline with automated testing and security scans, incorporating practices like Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST). Results were visualized in a Vulnerability Management Tool, providing insights into potential vulnerabilities.

As a bonus, the sprint involved implementing monitoring tools for system, application, and log monitoring. System monitoring covered crucial infrastructure metrics within the Kubernetes cluster, while application monitoring supervised the MySQL database, backend, and frontend. Log monitoring was established to oversee logs generated by different components. These tools offered valuable insights for proactive issue identification and resolution.

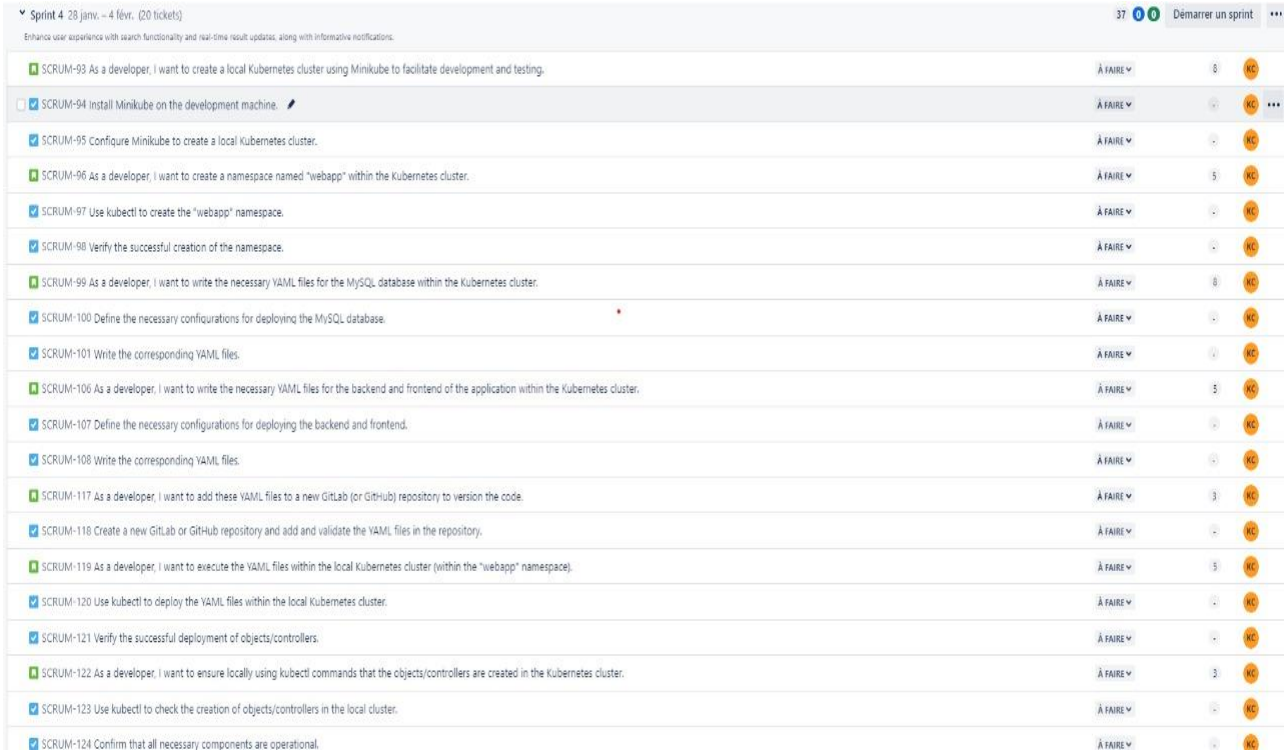
Overall, the sprint aimed to strengthen the application's security through enhanced checks and automation, coupled with improved visibility and control over system performance and logs.



3. Project deliverable:

Sprints Backlogs(Chbail Khalid):

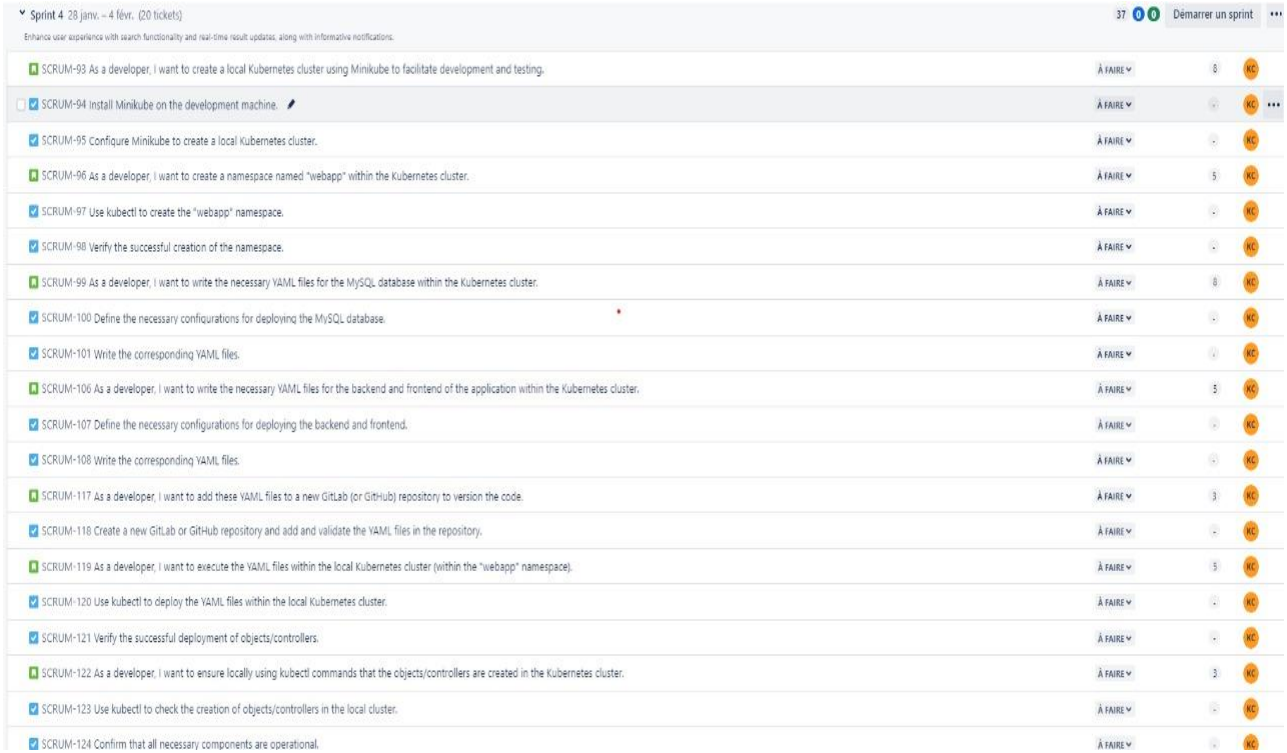
In this sprint, the focus was on setting up a local Kubernetes cluster using Minikube or an equivalent tool. The first task involved creating the cluster, providing a self-contained environment for application deployment and management. The next step was to create a namespace named "webapp" within the Kubernetes cluster, ensuring proper organization and isolation of resources.To enable database functionality, YAML files were created for the MySQL database, defining the necessary configurations and specifications. Similarly, YAML files were prepared for the backend and frontend components of the application, outlining container definitions, volume mounts, and service configurations.These YAML files were then added to a new GitLab or GitHub repository, facilitating version control and collaboration. Furthermore, the files were executed within the local Kubernetes cluster, specifically targeting the "webapp" namespace. This allowed for the deployment and orchestration of the application components within the Kubernetes environment.To validate the successful creation and operation of Kubernetes objects and controllers, local verification was performed using kubectl commands. These commands ensured that the desired resources, such as pods, services, and deployments, were created and running as intended within the cluster.Overall, this sprint focused on establishing a local Kubernetes cluster, configuring namespaces, and deploying the MySQL database, backend, and frontend components using YAML files. The utilization of version control repositories and local verification ensured a smooth and efficient setup of the Kubernetes infrastructure for the application.



3. Project deliverable:

Sprints Backlogs(Chbail Khalid):

In this sprint, the focus was on setting up a local Kubernetes cluster using Minikube or an equivalent tool. The first task involved creating the cluster, providing a self-contained environment for application deployment and management. The next step was to create a namespace named "webapp" within the Kubernetes cluster, ensuring proper organization and isolation of resources. To enable database functionality, YAML files were created for the MySQL database, defining the necessary configurations and specifications. Similarly, YAML files were prepared for the backend and frontend components of the application, outlining container definitions, volume mounts, and service configurations. These YAML files were then added to a new GitLab or GitHub repository, facilitating version control and collaboration. Furthermore, the files were executed within the local Kubernetes cluster, specifically targeting the "webapp" namespace. This allowed for the deployment and orchestration of the application components within the Kubernetes environment. To validate the successful creation and operation of Kubernetes objects and controllers, local verification was performed using kubectl commands. These commands ensured that the desired resources, such as pods, services, and deployments, were created and running as intended within the cluster. Overall, this sprint focused on establishing a local Kubernetes cluster, configuring namespaces, and deploying the MySQL database, backend, and frontend components using YAML files. The utilization of version control repositories and local verification ensured a smooth and efficient setup of the Kubernetes infrastructure for the application.



Well, that was a ton of reading

1. Backend pipeline
   - 1.1- Dockerfile
   - 1.2- .gitlab-ci.yml
   - 1.3- execution of the pipeline
2. Frontend pipeline
   - 2.1- Dockerfile
   - 2.2- .gitlab-ci.yml
   - 2.3- execution of the pipeline
3. Optimization techniques :
   - 3.1- Gitlab-runner
   - 3.2- Generic Jobs
   - 3.3- Rules

Section 2

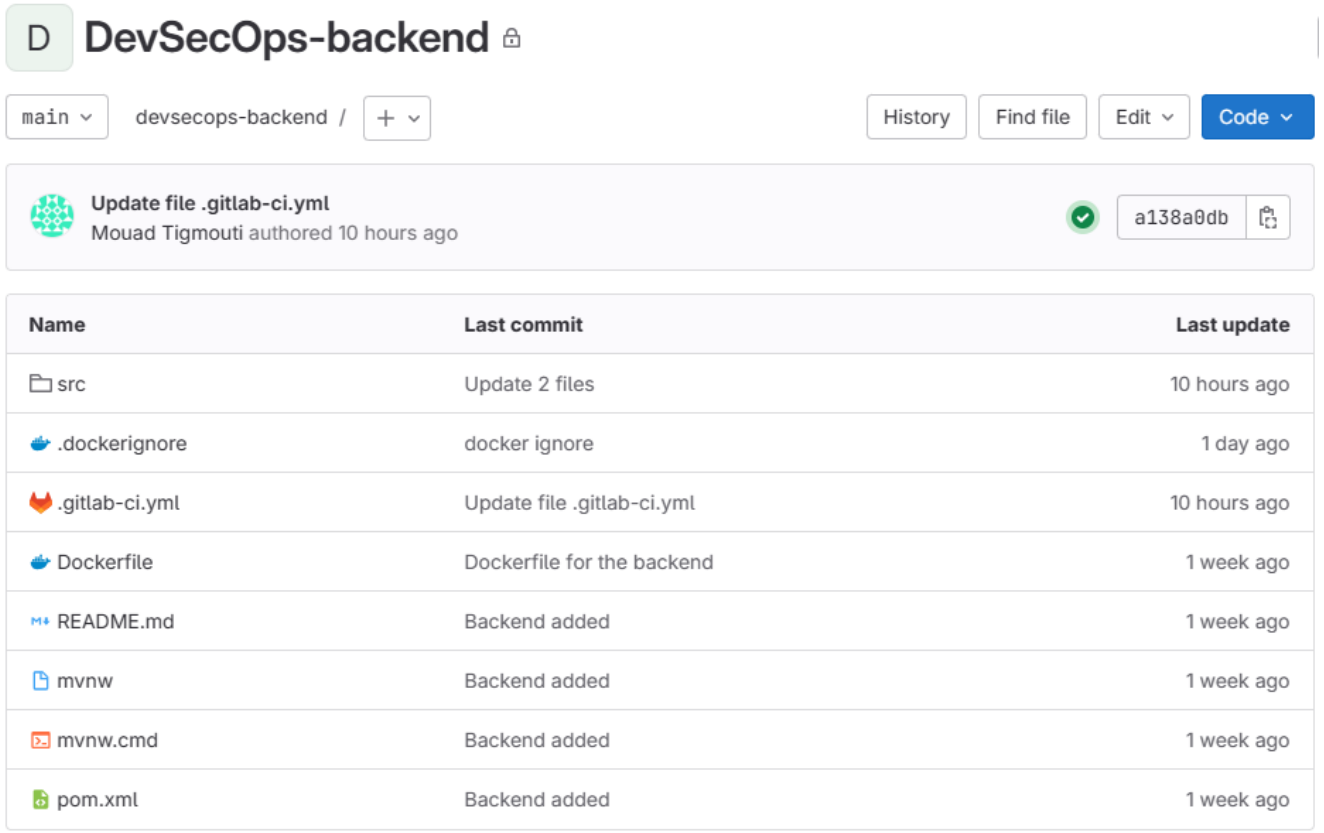
From this point, things started to get more exciting

1. Backend pipeline

First, we need to clone the repositories from GitHub and then push them into our GitLab repository. Here is the result :

Of course, these files don't exist in the original repository, we created them so they help us during the process.

Let's explain each one of them one by one.



1.1-Backend Dockerfile

This is the Dockerfile for the backend.

Since we are working with a Spring Boot project, we need to prepare the environment and use Java.
We are using OpenJDK version 17-jdk-slim.

Dockerfile | TutorialController.java | .gitlab-ci.yml

Dockerfile

```
1  FROM openjdk:17-jdk-slim
2  WORKDIR /app
3  COPY target/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar /app/spring-boot-app.jar
4  EXPOSE 8081
5  CMD ["java", "-jar", "spring-boot-app.jar"]
6
```

FROM openjdk:17-jdk-slim: This sets the base image to OpenJDK version 17 with a slim Linux distribution. It provides the necessary Java runtime for our application.

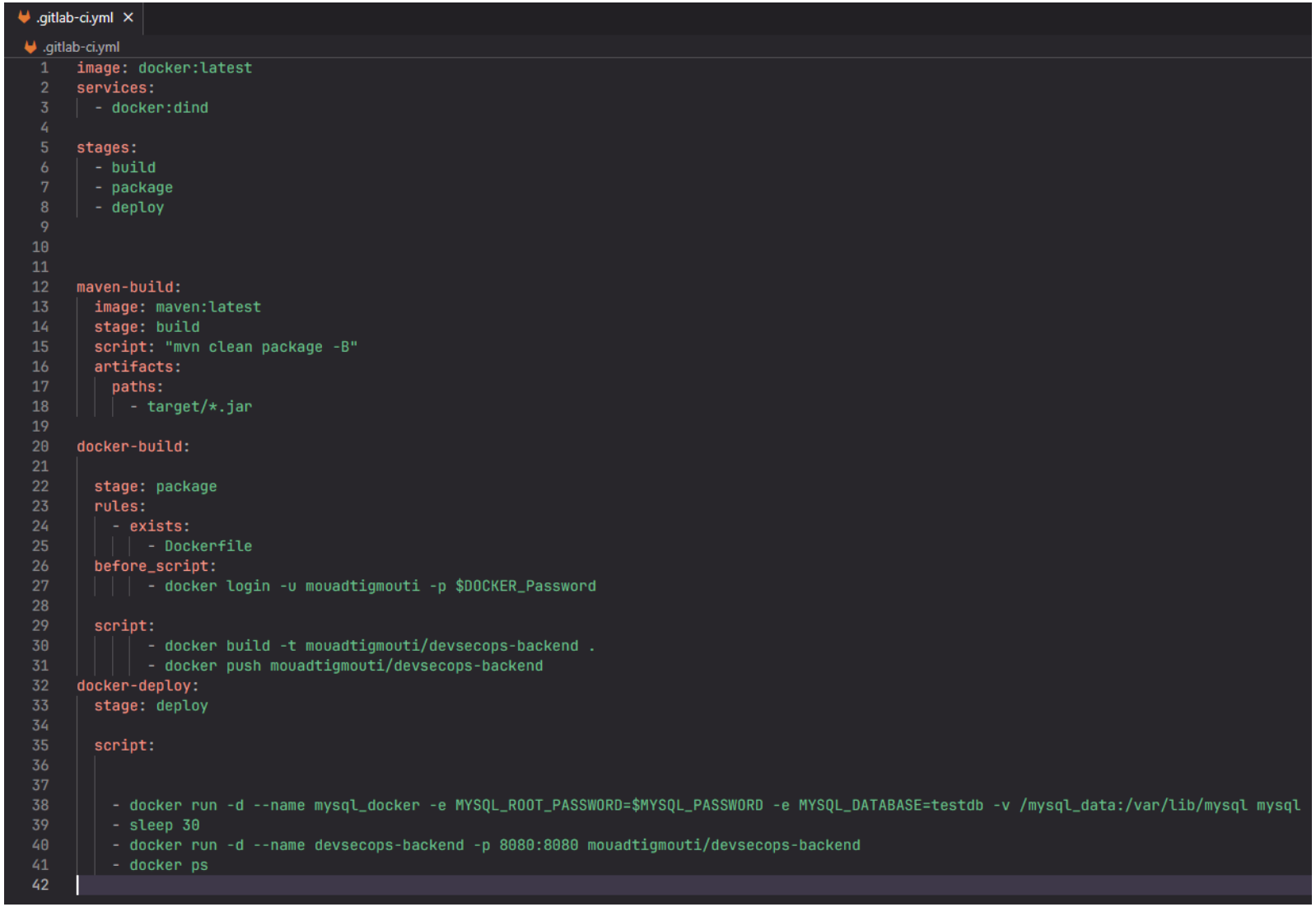
WORKDIR /app: This sets the working directory inside the container to /app.

COPY target/spring-boot-data-jpa-0.0.1-SNAPSHOT.jar /app/spring-boot-app.jar: This copies the Spring Boot application JAR file from the local target directory to the /app directory inside the container.

EXPOSE 8081: This informs Docker that the application inside the container will use port 8081.

CMD ["java", "-jar", "spring-boot-app.jar"]: This sets the default command to run when the container starts.

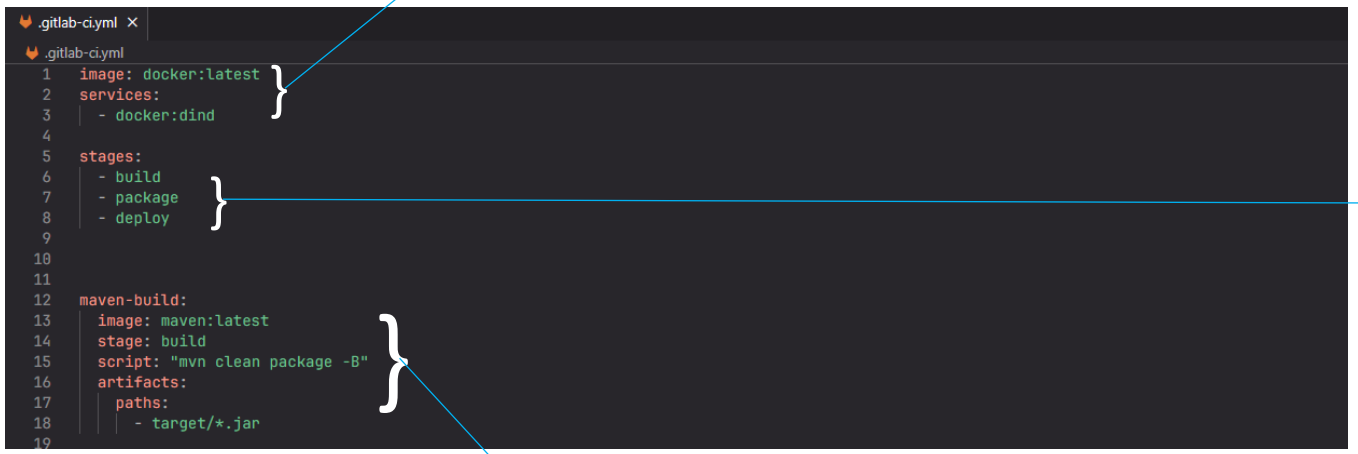
1.2-Backend .gitlab-ci.yml



This is the pipeline code; it is divided into 3 stages: build, package, and deploy. Let's see what each part does.

1.2-Backend .gitlab-ci.yml

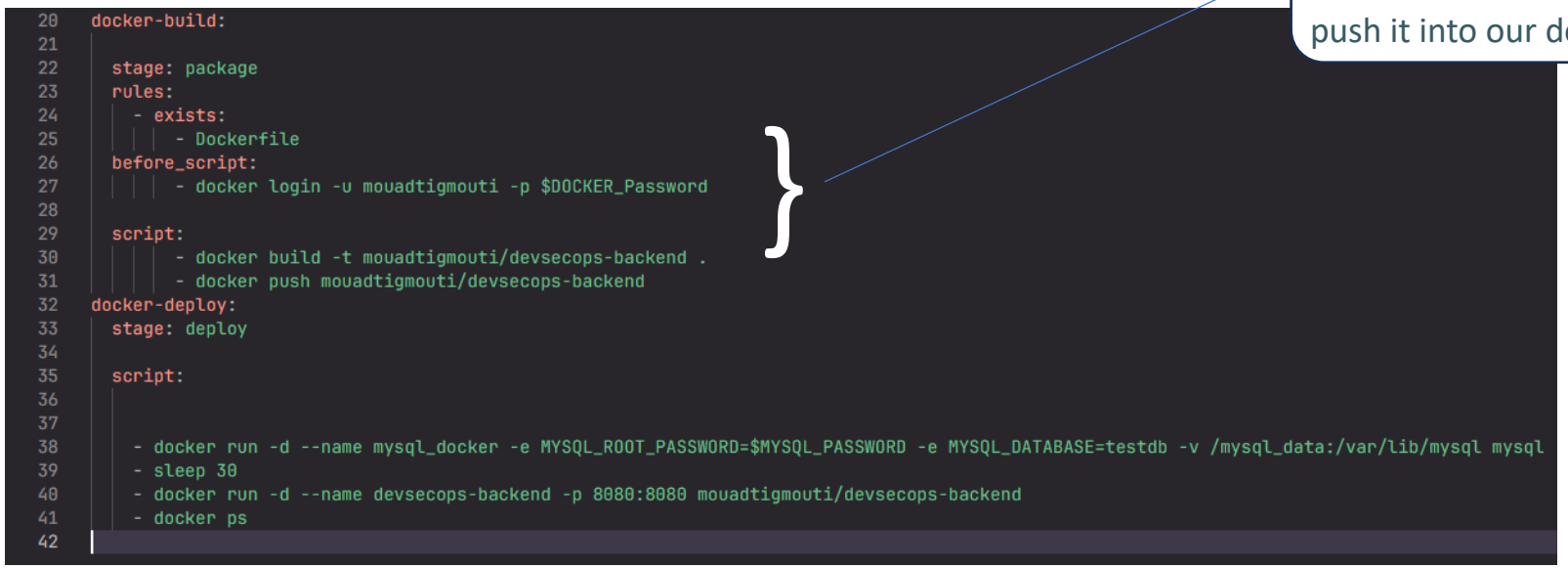
This part specifies the Docker image that will be used as the runner for the CI/CD pipeline (docker:latest).
Also, it sets up Docker-in-Docker (dind) as a service, allowing the CI/CD jobs to run Docker commands.



Here we declare that the stages of the CI/CD pipeline will be : build, package, and deploy. And the jobs of these stages will be executed sequentially.

Here we define a Maven build job, using the maven:latest Docker image, it executes the Maven build command, and stores the resulting JAR files as artifacts.

1.2-Backend .gitlab-ci.yml



```
20  docker-build:
21
22    stage: package
23    rules:
24      - exists:
25          - Dockerfile
26    before_script:
27          - docker login -u mouadtigmouti -p $DOCKER_Password
28
29    script:
30          - docker build -t mouadtigmouti/devsecops-backend .
31          - docker push mouadtigmouti/devsecops-backend
32  docker-deploy:
33    stage: deploy
34
35    script:
36
37
38      - docker run -d --name mysql_docker -e MYSQL_ROOT_PASSWORD=$MYSQL_PASSWORD -e MYSQL_DATABASE=testdb -v /mysql_data:/var/lib/mysql mysql
39      - sleep 30
40      - docker run -d --name devsecops-backend -p 8080:8080 mouadtigmouti/devsecops-backend
41      - docker ps
42
```

This is the configuration for packaging the application. It checks for the existence of a Dockerfile, logs into Docker registry, and builds the Docker image if the Dockerfile is present and then push it into our dockerhub account,

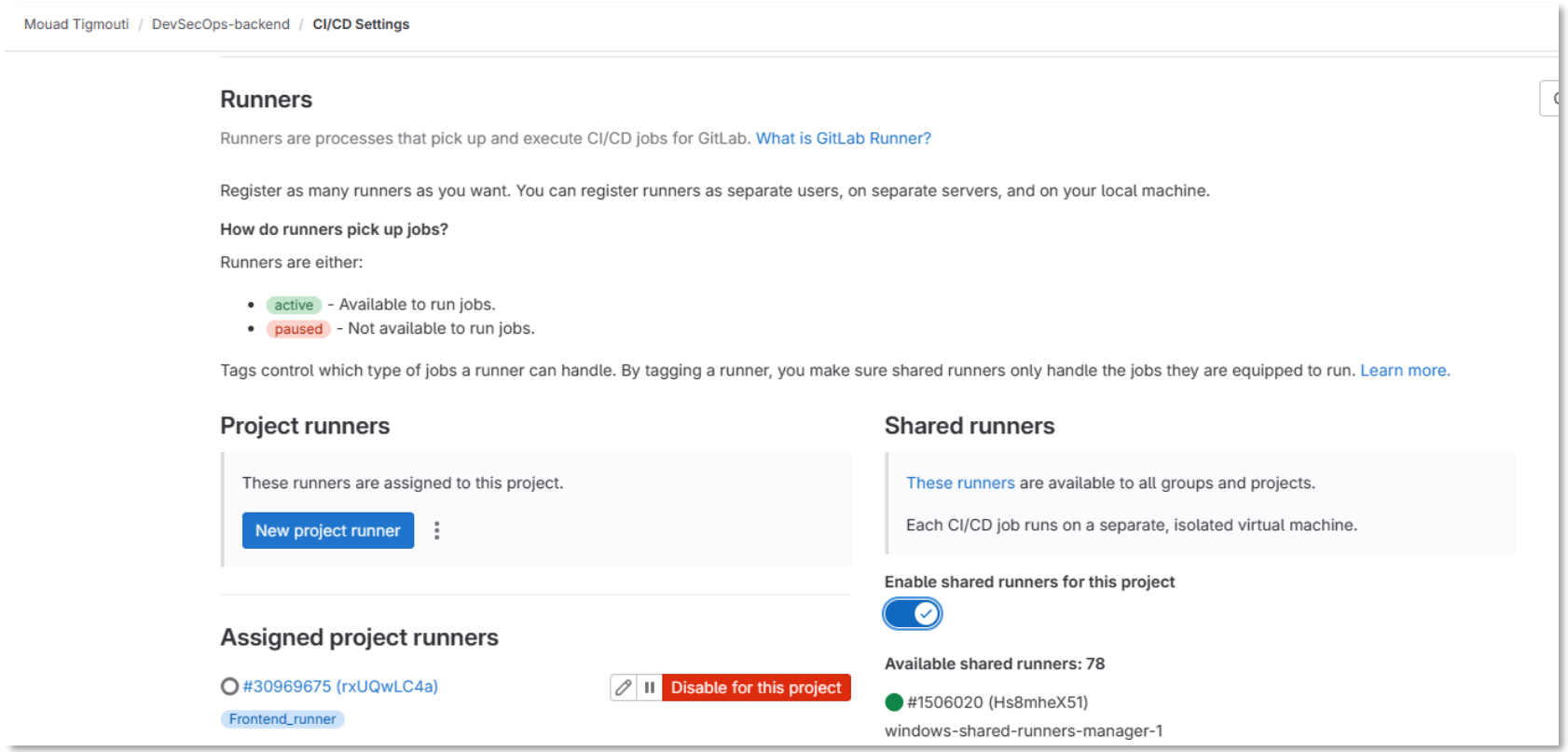
Here we define a Docker deployment job. It stops and removes existing Docker containers, and then runs MySQL and the application containers with specified configurations.

In this YAML file, I chose to simulate the scenario as if it were to be deployed on a machine. To achieve this, I created a volume to link the data of the MySQL container with the host file system. This ensures that even if something goes wrong with the MySQL container, the data will remain intact. This step will be useful for the subsequent sections and steps in this project

1.3-Backend .gitlab-ci.yml

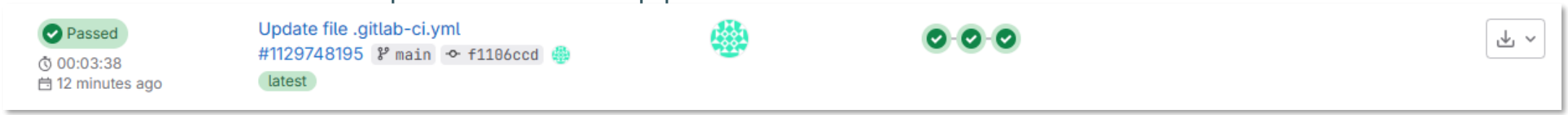
Now, let's execute the pipeline code and see the results :

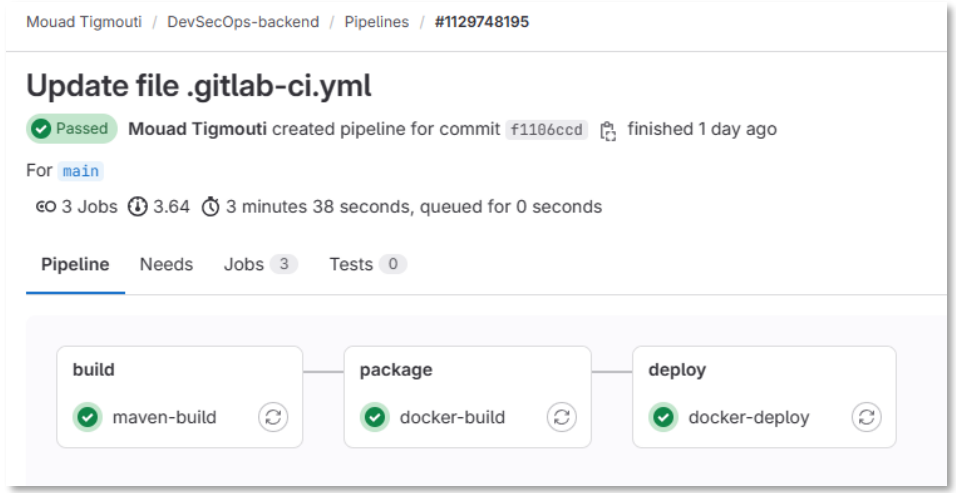
For this step we will use the shared runner offered by Gitlab :



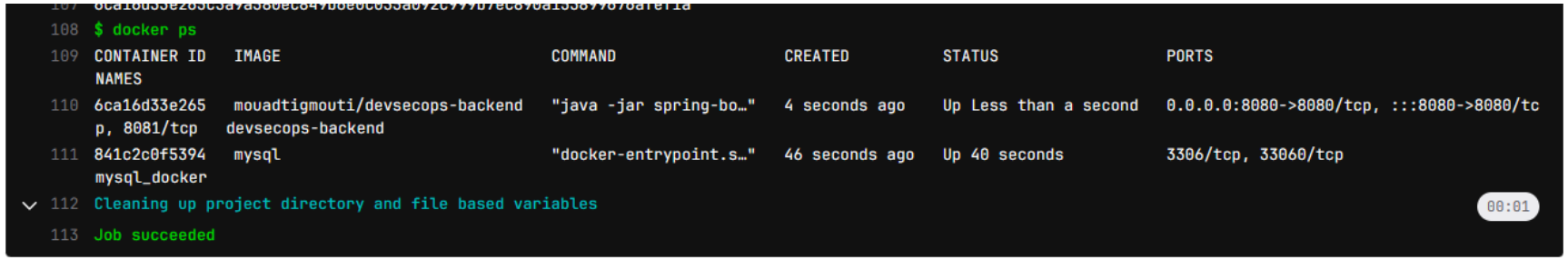
1.3-Backend .gitlab-ci.yml

Here it took 3 minutes to complete the backend pipeline :





This is the result of the last command in the pipeline:



1. Backend pipeline
   - 1.1- Dockerfile
   - 1.2- .gitlab-ci.yml
   - 1.3- execution of the pipeline

2. Frontend pipeline
   - 2.1- Dockerfile
   - 2.2- .gitlab-ci.yml
   - 2.3- execution of the pipeline

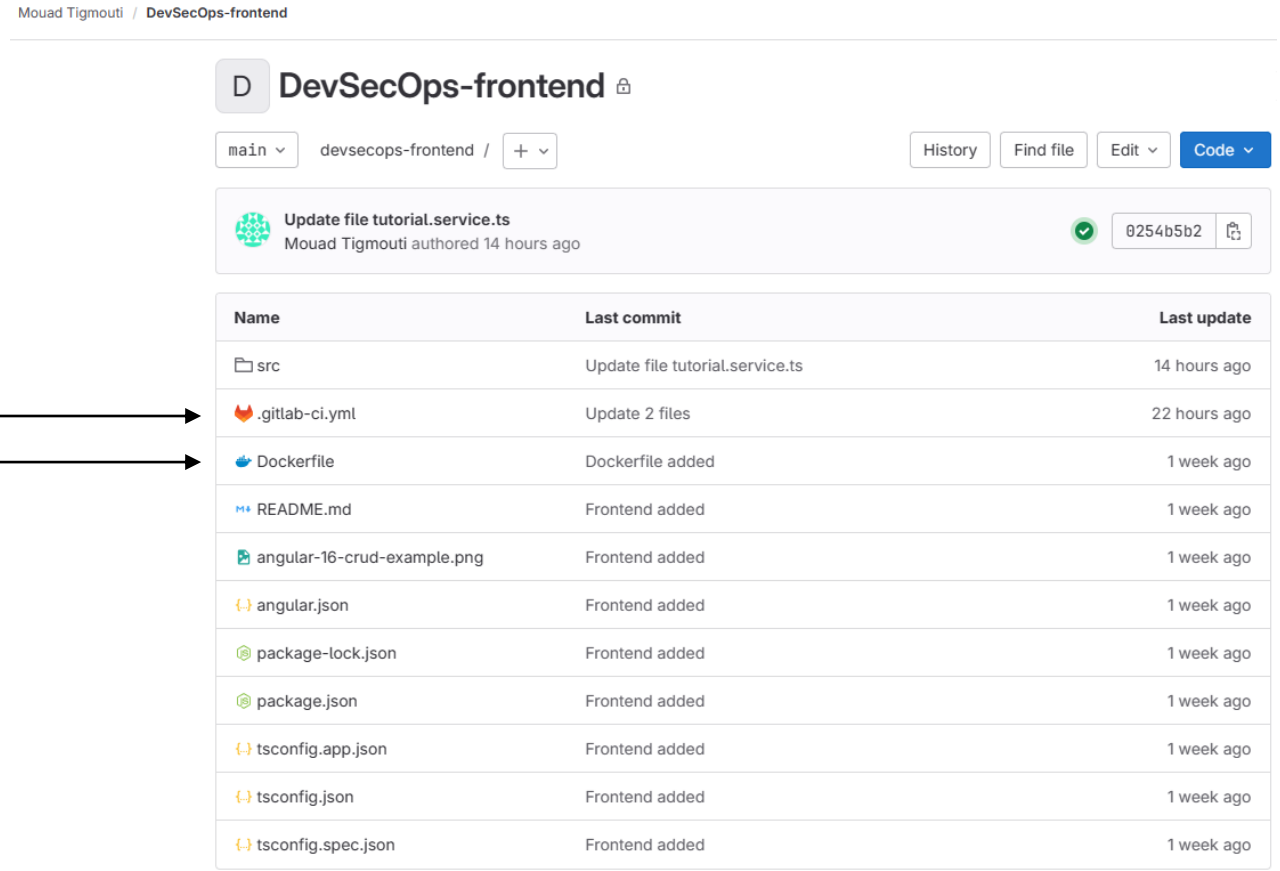
3. Optimization techniques :
   - 3.1- Gitlab-runner
   - 3.2- Generic Jobs
   - 3.3- Rules

Section 2

2. Frontend pipeline

As we did for the backend, we need to clone the repositories from GitHub and then push them into our GitLab repository. Here is the result :

Let’s explain the content of these file :



2.1-Frontend Dockerfile

Lets start with the dockerfile
Considering that we are working with an Angular project, we need to use node.
We are using latest version of Node.js.

Dockerfile

```
1  FROM node:latest
2  WORKDIR /app
3  ENV PATH /app/node_modules/.bin:$PATH
4  COPY package.json /app/package.json
5  RUN npm install
6  RUN npm install -g @angular/cli@7.3.9
7  COPY . /app
8  CMD ng serve --host 0.0.0.0 --port 8081
9
```

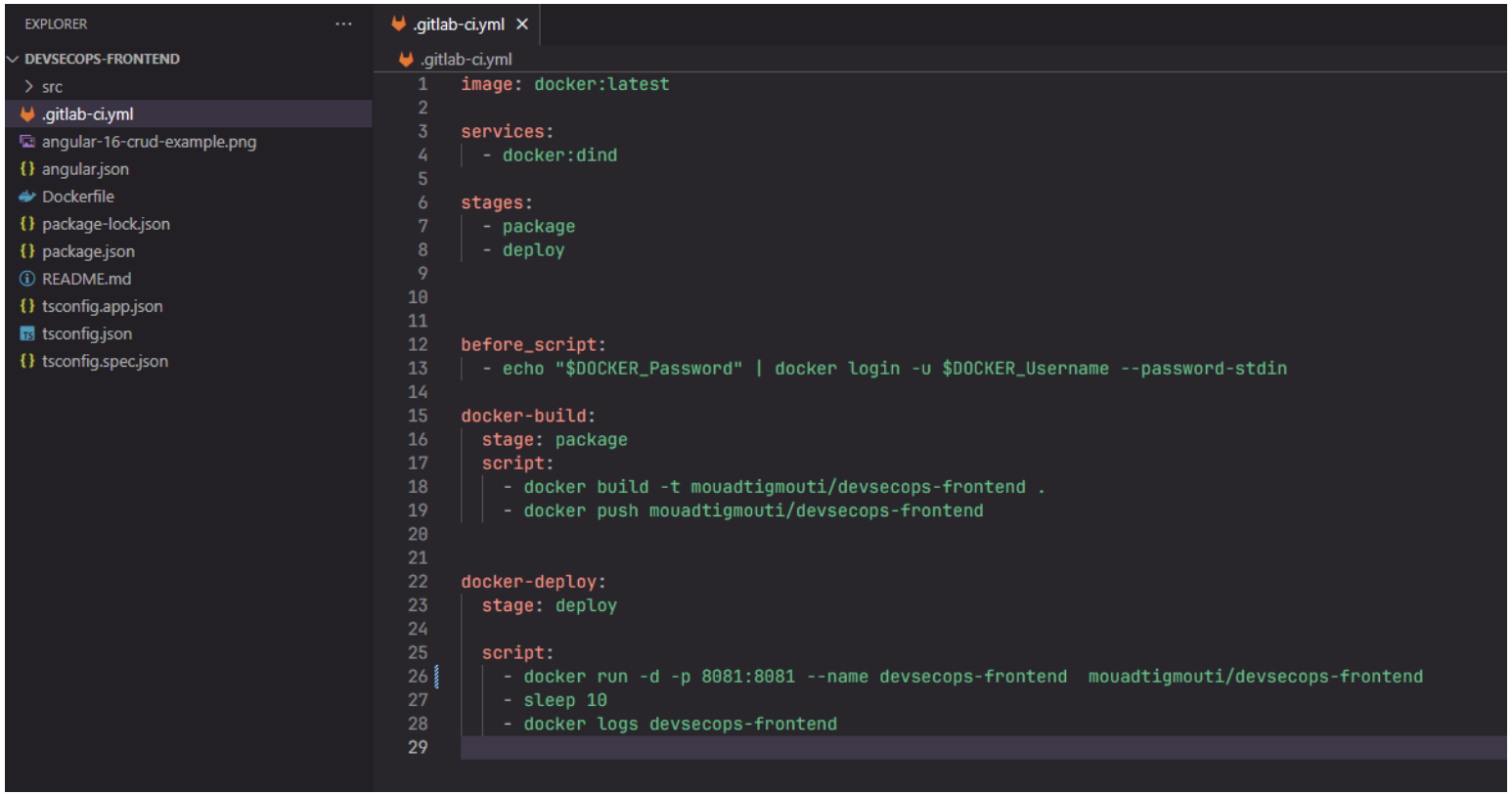
FROM node:latest: This line sets the base image for the Docker container. In this case, we are using the latest version of the official Node.js image.

WORKDIR /app: This sets the working directory inside the container to /app.

ENV PATH /app/node_modules/.bin:$PATH: This line updates the PATH environment variable to include the /app/node_modules/.bin directory. So that we make sure that locally installed binaries can be executed without any problem.

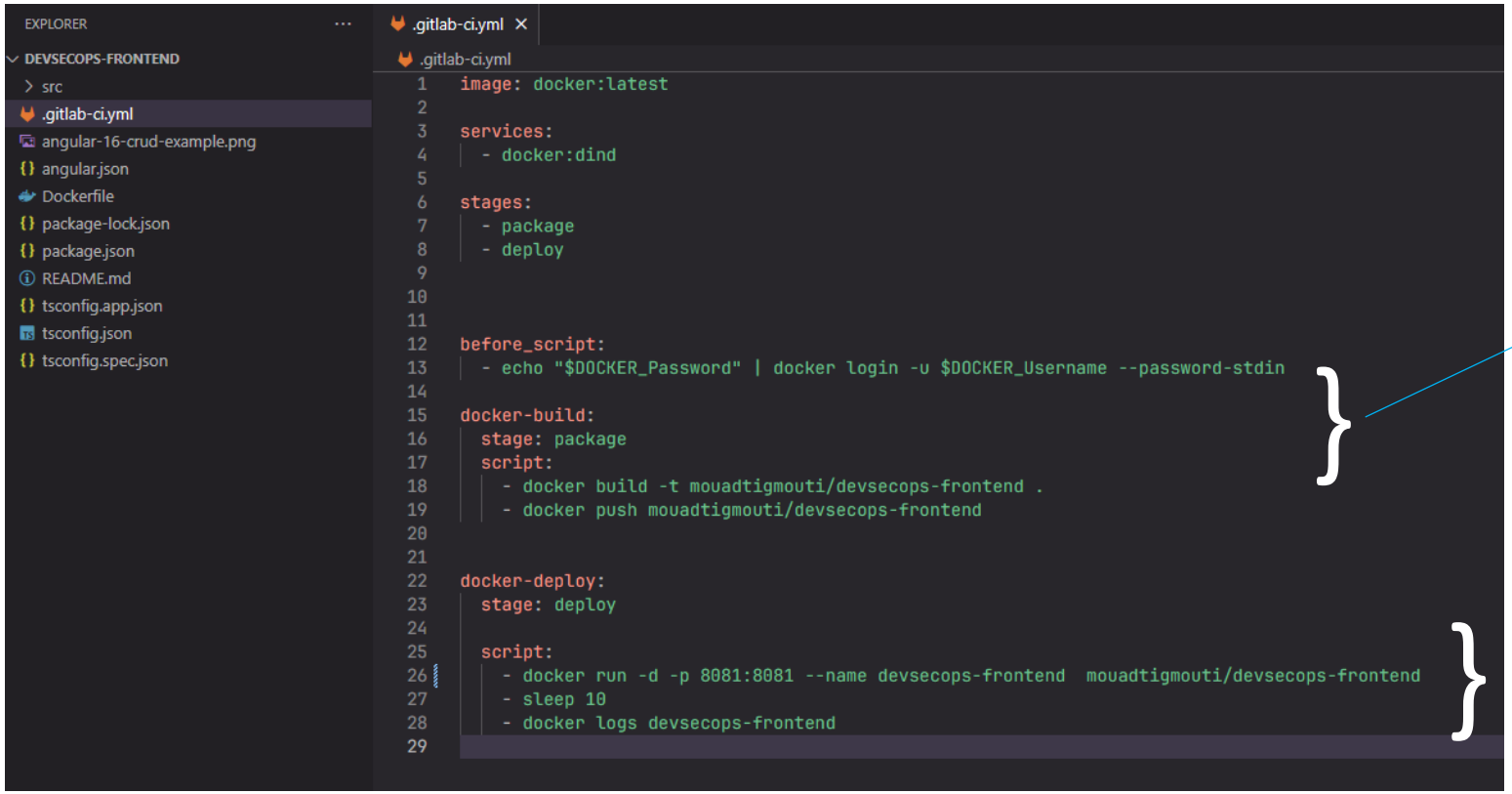
RUN npm install: This line runs the npm install command inside the container.
RUN npm install -g @angular/cli@7.3.9: This line installs the Angular CLI
COPY . /app: This line copies the entire content of the current directory where the Dockerfile is located to the /app directory inside the container.
CMD ng serve --host 0.0.0.0 --port 8081: This line specifies the default command to run when the container starts.

2.2-Frontend .gitlab-ci.yml



This time the pipeline has 2 stages : Package and Deploy.

2.2-Frontend .gitlab-ci.yml



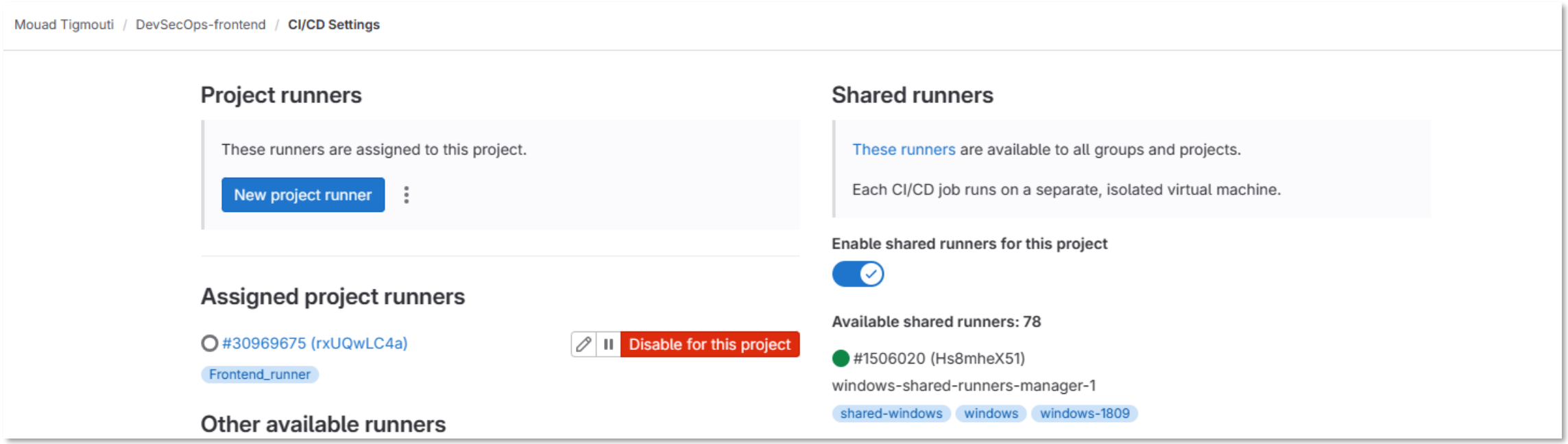
In this section, the pipeline executes two stages: before_script for Docker login and docker-build for building and pushing the Docker image to our dockerhub.

the docker-deploy job stops and removes any existing containers to avoid any conflict with the name devsecops-frontend, then runs a new container, sleep 10 seconds and then it shows the logs of the frontend container

2.3-Frontend .gitlab-ci.yml

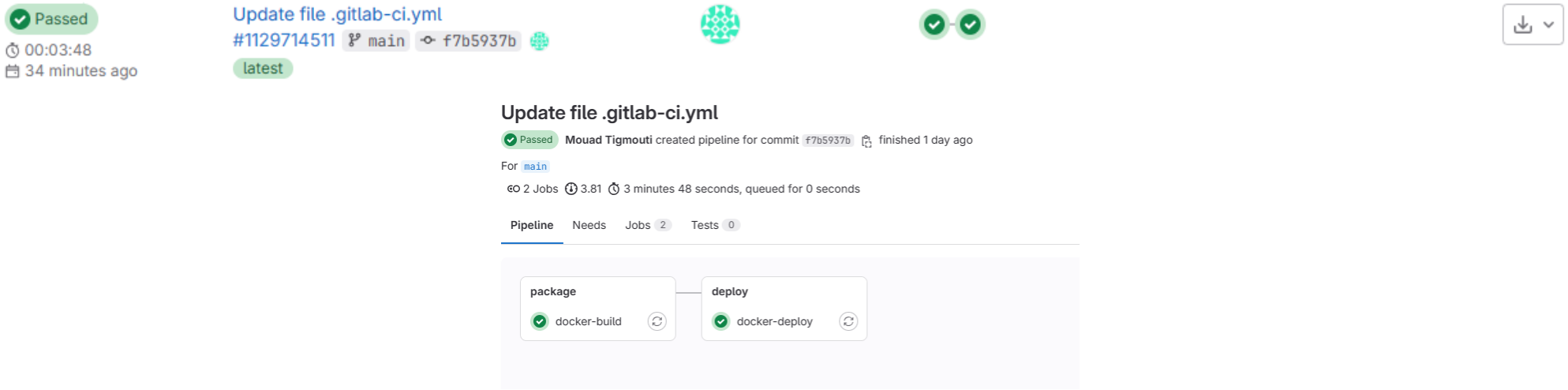
Now, everything is set let's execute the pipeline code :

For this step too we will use the shared runner offered by Gitlab :



2.3-Frontend .gitlab-ci.yml

Here it took 3 minutes to complete the frontend pipeline :



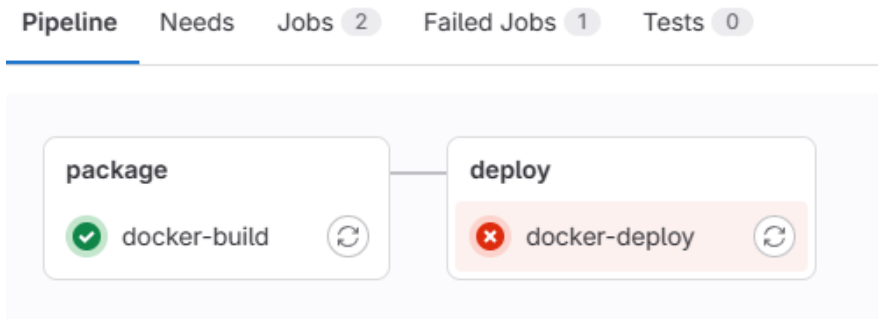
This is the result of the last command in the pipeline:

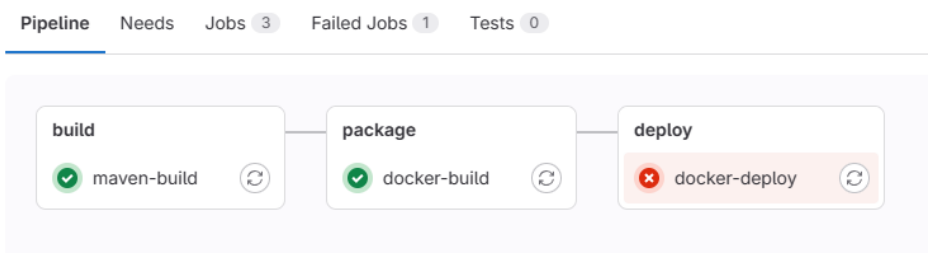
```
95  $ sleep 10
96  $ docker logs devsecops-frontend
97  Node.js version v21.5.0 detected.
98  Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please see https://nodejs.org/en/about/releases/.
99  Warning: This is a simple server for use in testing or debugging Angular applications
100 locally. It hasn't been reviewed for security issues.
101 Binding this server to an open connection can result in compromising your application or
102 computer. Using a different host than the one passed to the "--host" flag might result in
103 websocket connection issues. You might need to use "--disable-host-check" if that's the
104 case.
105
106 - Generating browser application bundles (phase: setup)...
107 Cleaning up project directory and file based variables                    00:00
108 Job succeeded
```

Our pipeline is running, of course not from the first time that’s for sure :

Passed	00:10:43 — 40 minutes ago
Passed	00:10:40 — 1 hour ago
Passed	00:12:42 — 1 hour ago
Passed	00:08:40 — 2 hours ago
Passed	00:06:42 — 2 hours ago
Passed	00:06:25 — 3 hours ago
Passed	00:04:40 — 3 hours ago

Failed	00:06:43 — 4 hours ago
Failed	00:04:42 — 4 hours ago
Failed	4 hours ago
Failed	4 hours ago
Failed	00:04:36 — 4 hours ago
Failed	00:03:49 — 5 hours ago





But we managed to fix the errors in our pipeline.

Now our pipeline is running, but it is not optimized :

Using a shared runner is not a practical solution since it is shared, and it can go out of service at any time.
Using a shared runner isn't great for reliable deployments because the shared runner might not always be available or perform consistently, which can affect the deployment process.

Also the code of the pipeline for both the frontend and the backend doesn't seem lightweight. It has all the stages in one place, which is not ideal for the upcoming steps where we plan to add more sections to the code.

For the next step we are going to optimize our pipeline.

1. Backend pipeline
   - 1.1- Dockerfile
   - 1.2- .gitlab-ci.yml
   - 1.3- execution of the pipeline
2. Frontend pipeline
   - 2.1- Dockerfile
   - 2.2- .gitlab-ci.yml
   - 2.3- execution of the pipeline
3. Optimization techniques :
   - 3.1- Gitlab-runner
   - 3.2- Generic Jobs
   - 3.3- Rules

Section 2

3.1-Gitlab-runner

To create our local GitLab-runner we need :

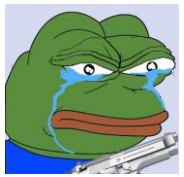
we need Ubuntu virtual machine, we are using Ubuntu 23.10 in VMWare

we need to install docker

we need to install gitlab-runner

we need to register the gitlab runner

Also we need a lot of patience



3.1-Gitlab-runner

We already did the steps for our Virtual machine

```
mouad@mouad-None:~$ gitlab-runner run
Runtime platform                                    arch=amd64 os=linux pid=5695
 revision=102c81ba version=16.7.0
Starting multi-runner from /home/mouad/.gitlab-runner/config.toml...  builds=0 m
ax_builds=0
WARNING: Running in user-mode.
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...

Configuration loaded                                builds=0 max_builds=1
listen_address not defined, metrics & debug endpoints disabled  builds=0 max_bui
lds=1
[session_server].listen_address not defined, session endpoints disabled  builds=
0 max_builds=1
Initializing executor providers                     builds=0 max_builds=1
Checking for jobs... received                       job=5882628381 repo_url=http
s://gitlab.com/mouad.tigmouti00/devsecops-backend.git runner=rxUQwLC4a
Added job to processing list                        builds=1 job=5882628381 max_
builds=1 project=53347837 repo_url=https://gitlab.com/mouad.tigmouti00/devsecops
-backend.git
Appending trace to coordinator...ok                 code=202 job=5882628381 job-
log=0-314 job-status=running runner=rxUQwLC4a sent-log=0-313 status=202 Accepted
 update-interval=1m0s
Appending trace to coordinator...ok                 code=202 job=5882628381 job-
```

Gitlab-runner

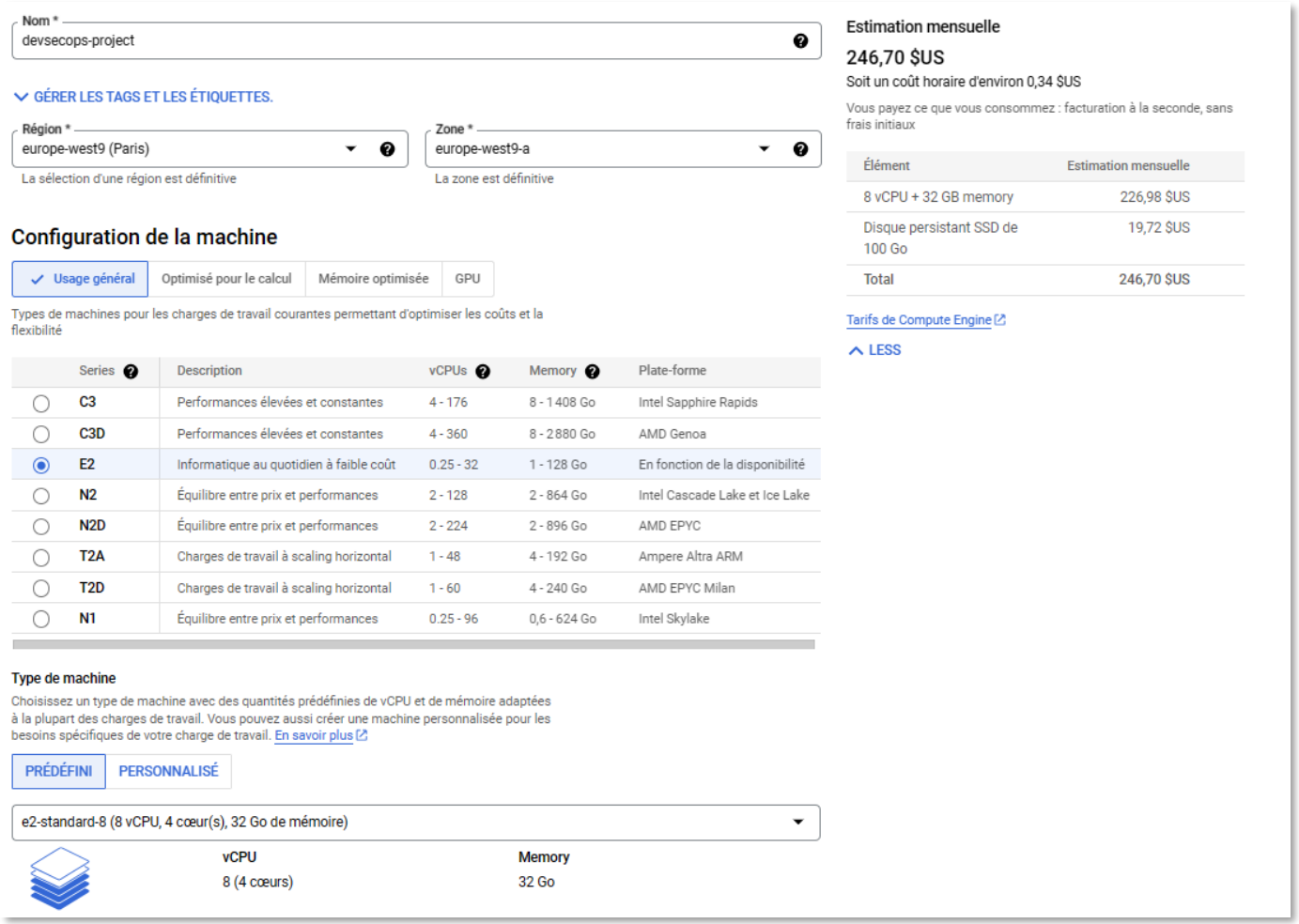
```
mouad@mouad-None:~$ docker --version
Docker version 24.0.7, build afdd53b
mouad@mouad-None:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CR
EATED         STATUS        PORTS                                     NAMES
9cadf4f8a422   mouadtigmouti/devsecops-backend      "java -jar spring-bo…"   6
hours ago     Up 6 hours    8081/tcp, 192.168.75.132:8083->8080/tcp   devsecops-backend
ab27f68611d8   mysql                                "docker-entrypoint.s…"   6
hours ago     Up 6 hours    33060/tcp, 192.168.75.132:3307->3306/tcp   mysql_docker
ec9b215ab6ad   defectdojo/defectdojo-django:latest   "/wait-for-it.sh mys…"   20
 hours ago    Up 10 hours                                               django-defectdojo-celerybeat-1
3f0e06771804   defectdojo/defectdojo-django:latest   "/wait-for-it.sh mys…"   20
 hours ago    Up 10 hours                                               django-defectdojo-celeryworker-1
db4152a93484   defectdojo/defectdojo-django:latest   "/wait-for-it.sh mys…"   20
 hours ago    Up 10 hours                                               django-defectdojo-uwsgi-1
40876495279f   mysql:5.7.44                         "docker-entrypoint.s…"   20
 hours ago    Up 10 hours   3306/tcp, 33060/tcp                         django-defectdojo-mysql-1
3db6cf3cab87   rabbitmq:3.12.11-alpine              "docker-entrypoint.s…"   20
 hours ago    Up 10 hours   4369/tcp, 5671-5672/tcp, 15691-15692/tcp, 25672/tcp   django-defectdojo-rabbitmq-1
2a1db629888f   defectdojo/defectdojo-nginx:latest    "/entrypoint-nginx.sh"   21
 hours ago    Up 10 hours   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 80/tcp, 0.
0.0.0:8443->8443/tcp, :::8443->8443/tcp   django-defectdojo-nginx-1
mouad@mouad-None:~$
```

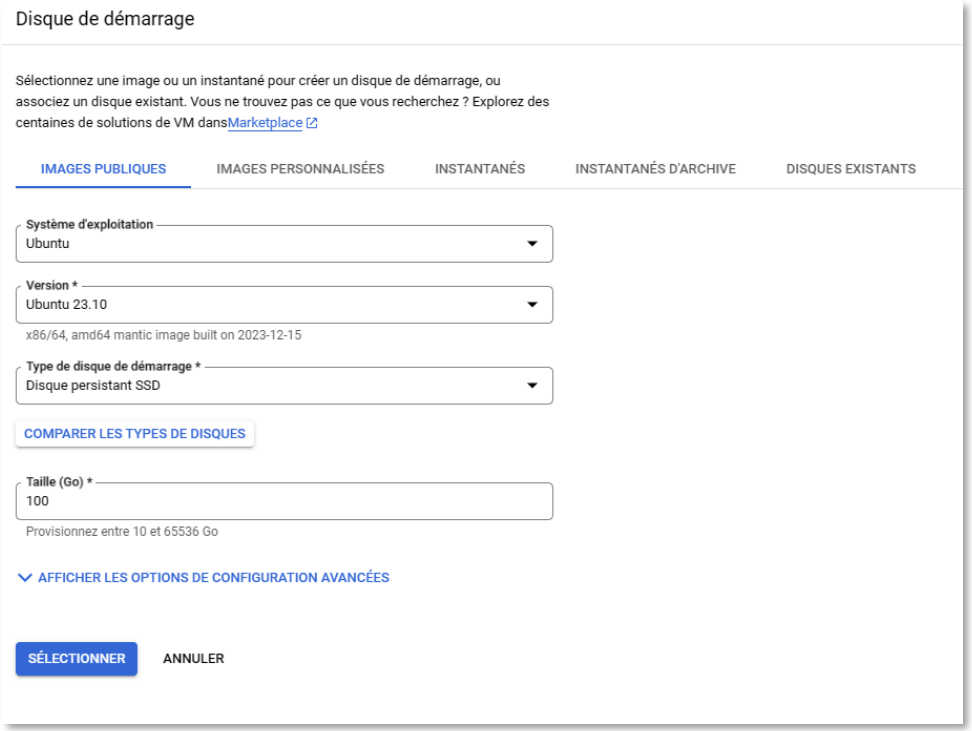
docker

But, we will repeat the steps once more as we have created a Google Cloud machine to host our project, providing an opportunity to demonstrate the steps again

3.1-Gitlab-runner

We created a Google Cloud machine with the same specifications mentioned earlier. However, this time, taking advantage of the free $300 credit for three months, we opted for a more powerful configuration: 8 cores, 32 GB RAM, and a 100 GB SSD. The cost for this configuration is $248 per month, which is more than enough for our needs.





3.1-Gitlab-runner

Now that we have created the machine, let's proceed with installing Docker using the following commands:

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl gnupg
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor
-o /etc/apt/keyrings/docker.gpg
```

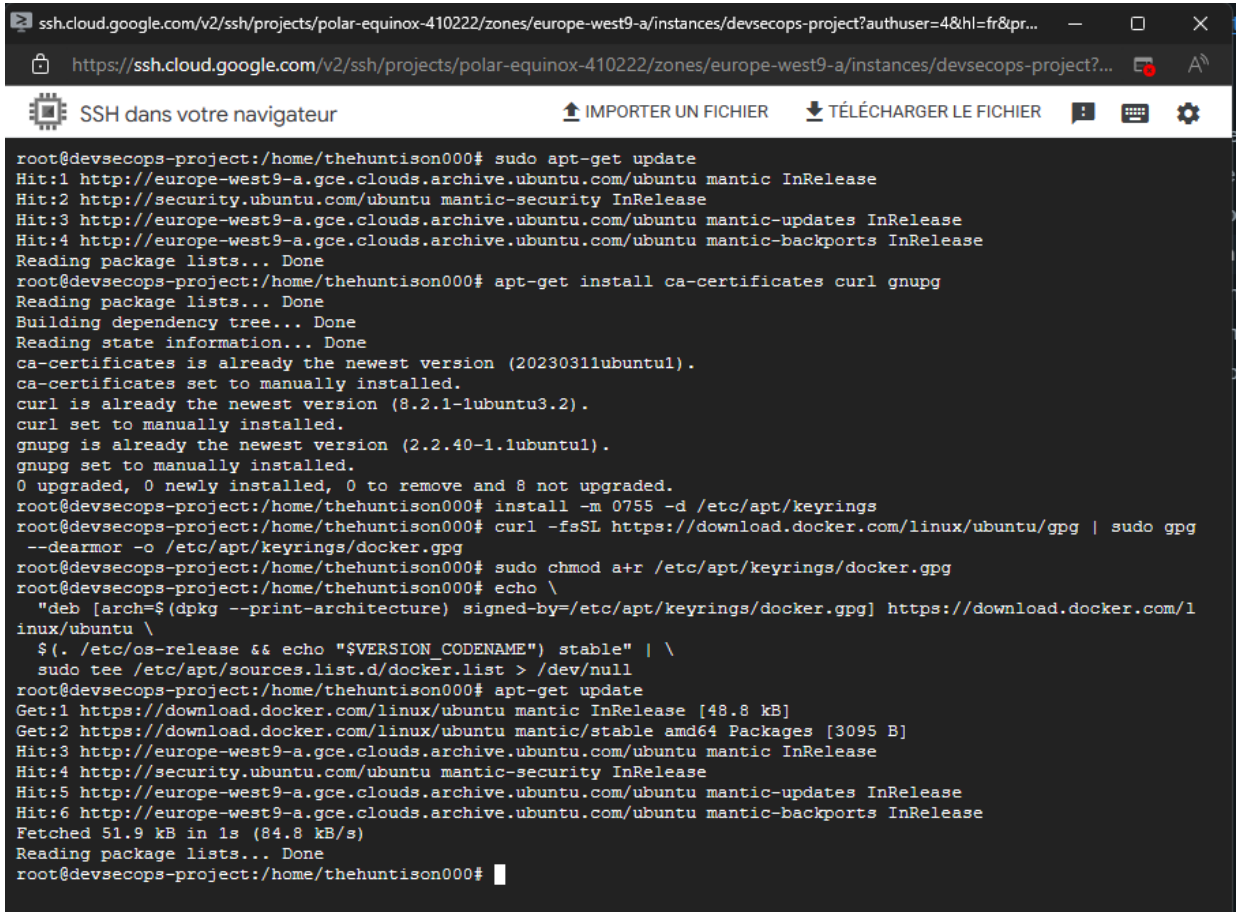
```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

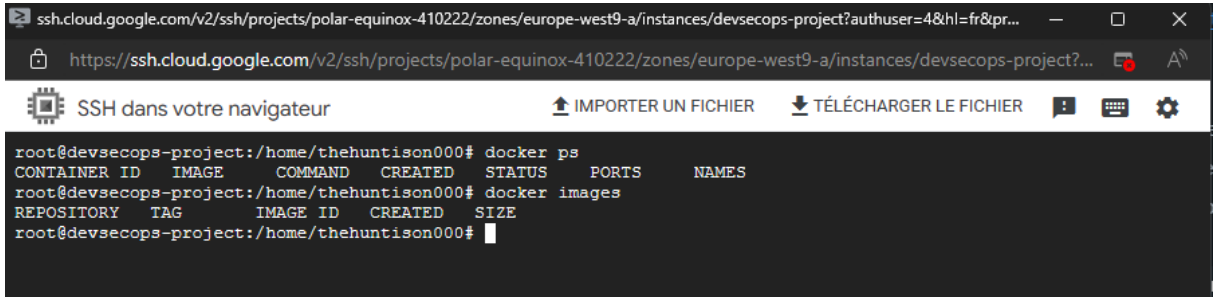
```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

```
 sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

3.1-Gitlab-runner





Now we have docker installed successfully

3.1-Gitlab-runner

After we installed docker it is time to install GitLab-runner
Using the following commands :

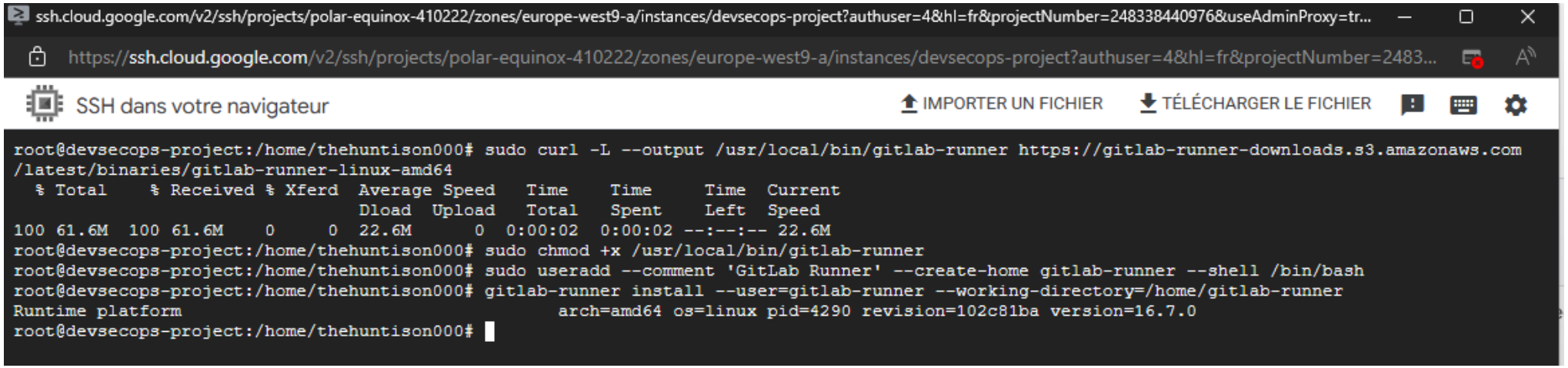
```
sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64
```

```
sudo chmod +x /usr/local/bin/gitlab-runner
```

```
sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash
```

```
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner
```

```
sudo gitlab-runner start
```



3.1-Gitlab-runner

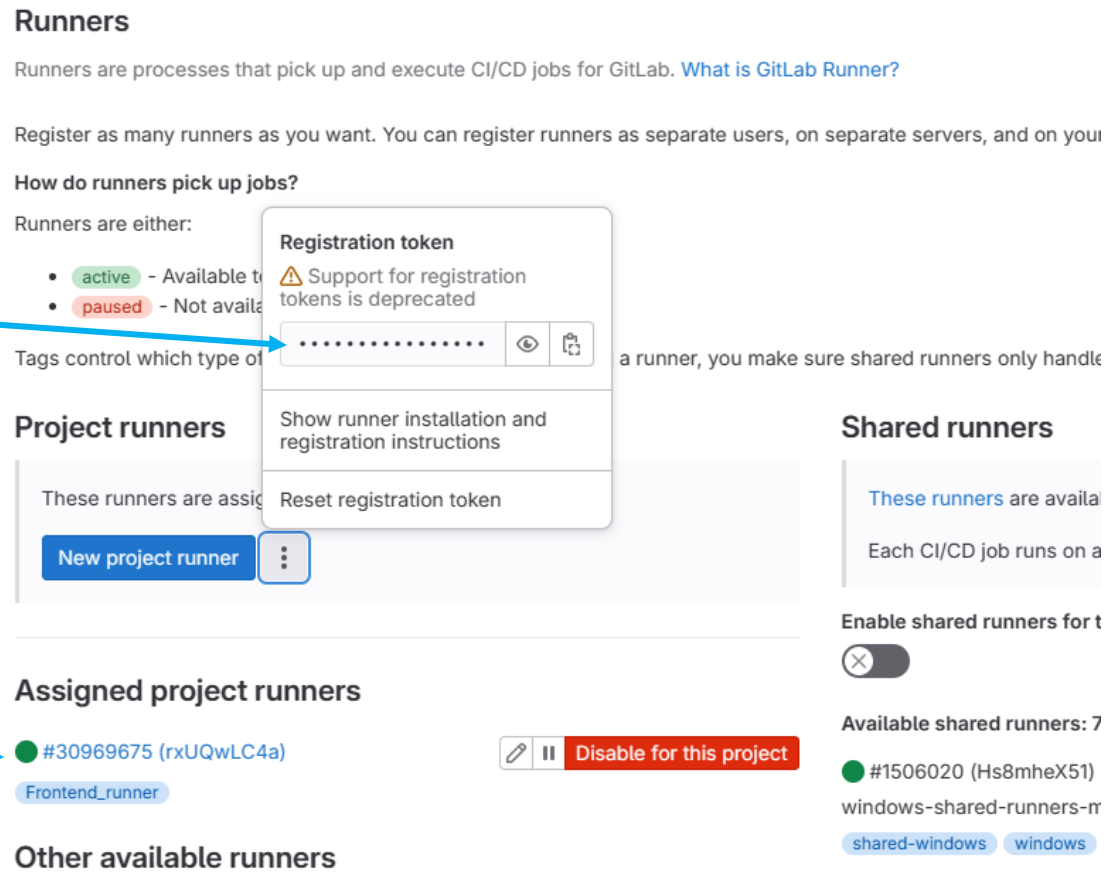
GitLab-runner now is installed, let’s register so we can use it :
We can register by running the following commands

Gitlab-runner register

https://gitlab.com

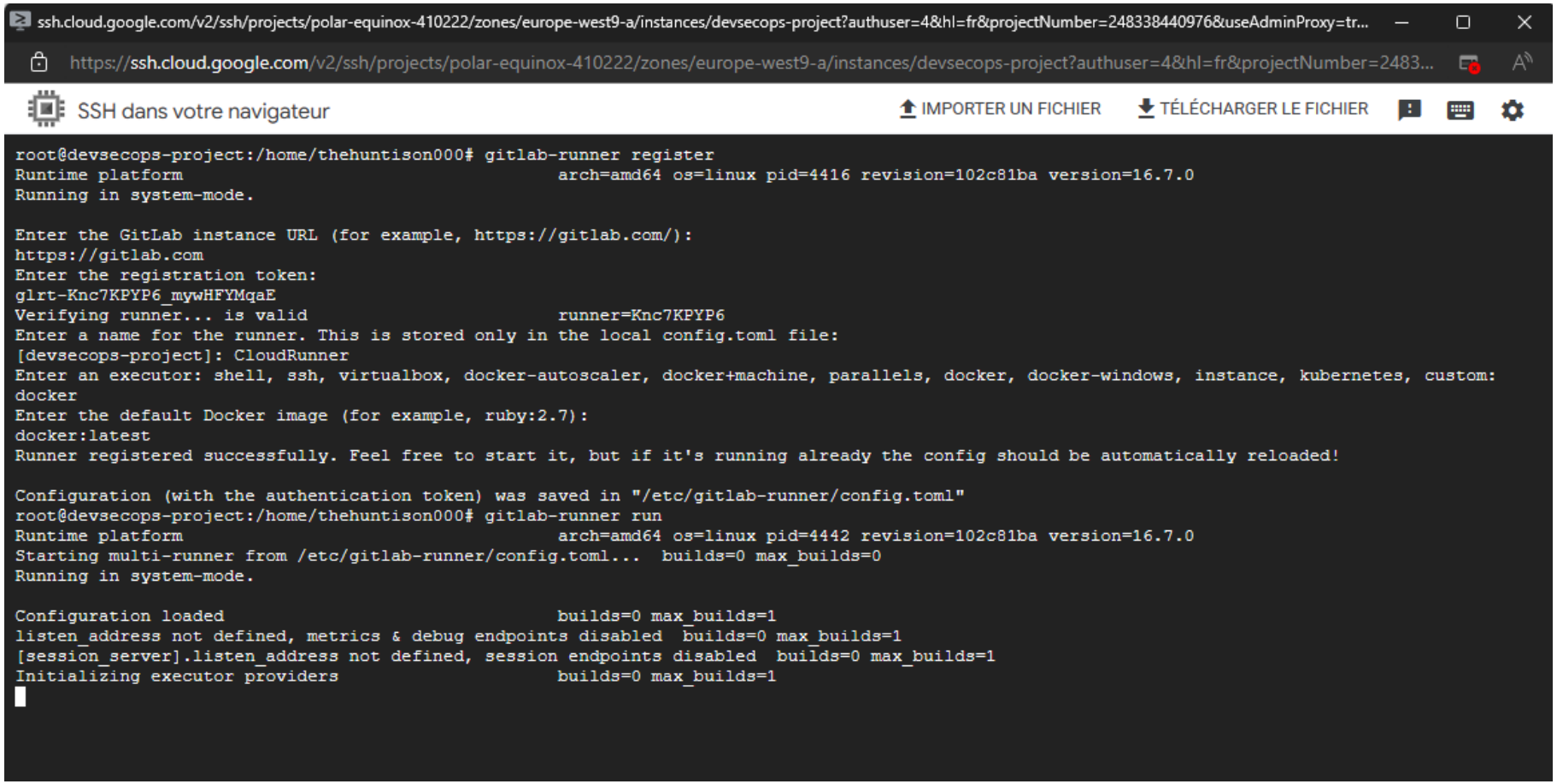
Then we give the token we got from Gitlab

Our runner that we are using we our local Ubuntu machine



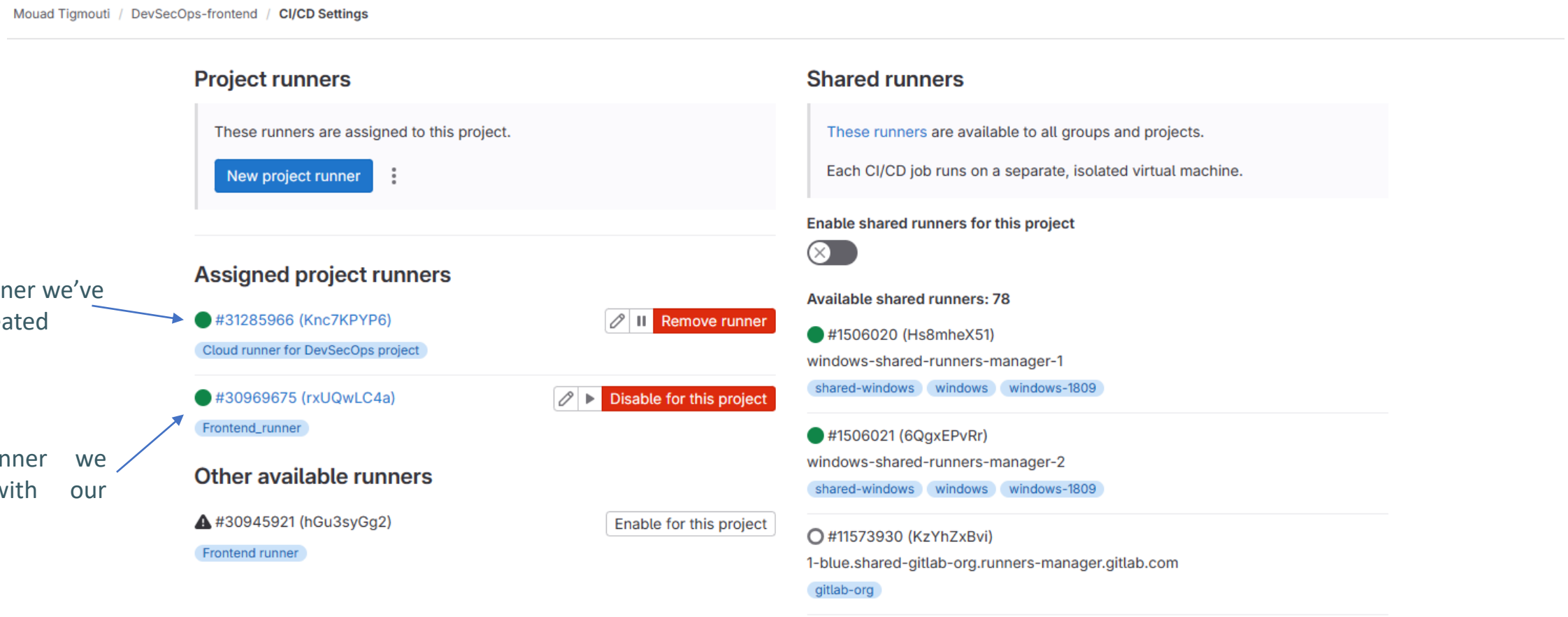
3.1-Gitlab-runner

With that, the GitLab runner is registered and ready to go :



3.1-Gitlab-runner

In Gitlab, we turned of the shared runners so that our local runner can start to receive the jobs.



The runner we've just created

The runner we used with our local VM

For the upcoming steps, we won't be using the Google Cloud Runner since we have been utilizing our local runner to run pipelines in our project before creating the Google Cloud machine.

3.1-Gitlab-runner

Backend

With our Gitlab-runner is prepared let’s run the pipeline :
First we start with the backend, it necessary to make some changes in the pipeline code so everything goes well :

.gitlab-ci.yml

```yaml
image: docker:latest
services:
  - docker:dind

stages:
  - build
  - package
  - deploy

variables:
  DOCKER_HOST: tcp://172.17.0.1:2375
  DOCKER_DRIVER: overlay2

maven-build:
  image: maven:latest
  stage: build
  script: "mvn clean package -B"
  artifacts:
    paths:
      - target/*.jar

docker-build:

  stage: package
  rules:
    - exists:
        - Dockerfile
  before_script:
        - docker login -u mouadtigmouti -p $DOCKER_Password

  script:
        - docker build -t mouadtigmouti/devsecops-backend .
        - docker push mouadtigmouti/devsecops-backend
```

We added these variables so that the runner can communicate with the local Docker, using the specified Docker host (tcp://172.17.0.1:2375) and the storage driver overlay2 that is responsible for managing the filesystem space used by Docker containers

3.1-Gitlab-runner

Backend

The other part of the pipeline code remains the same :

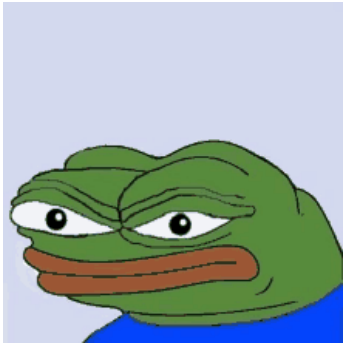
```
34  docker-deploy:
35    stage: deploy
36    before_script:
37      - docker stop devsecops-backend || true
38      - docker rm devsecops-backend || true
39      - docker stop mysql_docker || true
40      - docker rm mysql_docker || true
41      - docker network create backend_network || true
42    script:
43
44
45      - docker run -d --name mysql_docker --network backend_network -e MYSQL_ROOT_PASSWORD=$MYSQL_PASSWORD -e MYSQL_DATABASE=testdb
46        -v /mysql_data:/var/lib/mysql mysql
47      - sleep 30
48      - docker run -d --name devsecops-backend --network backend_network -p 8080:8080 mouadtigmouti/devsecops-backend
49      - docker ps
50
```

To stop all the previous containers and remove them to avoid conflicts

To keep the data safe by creating a volume in the host machine

To connect the backend and the database in the same network

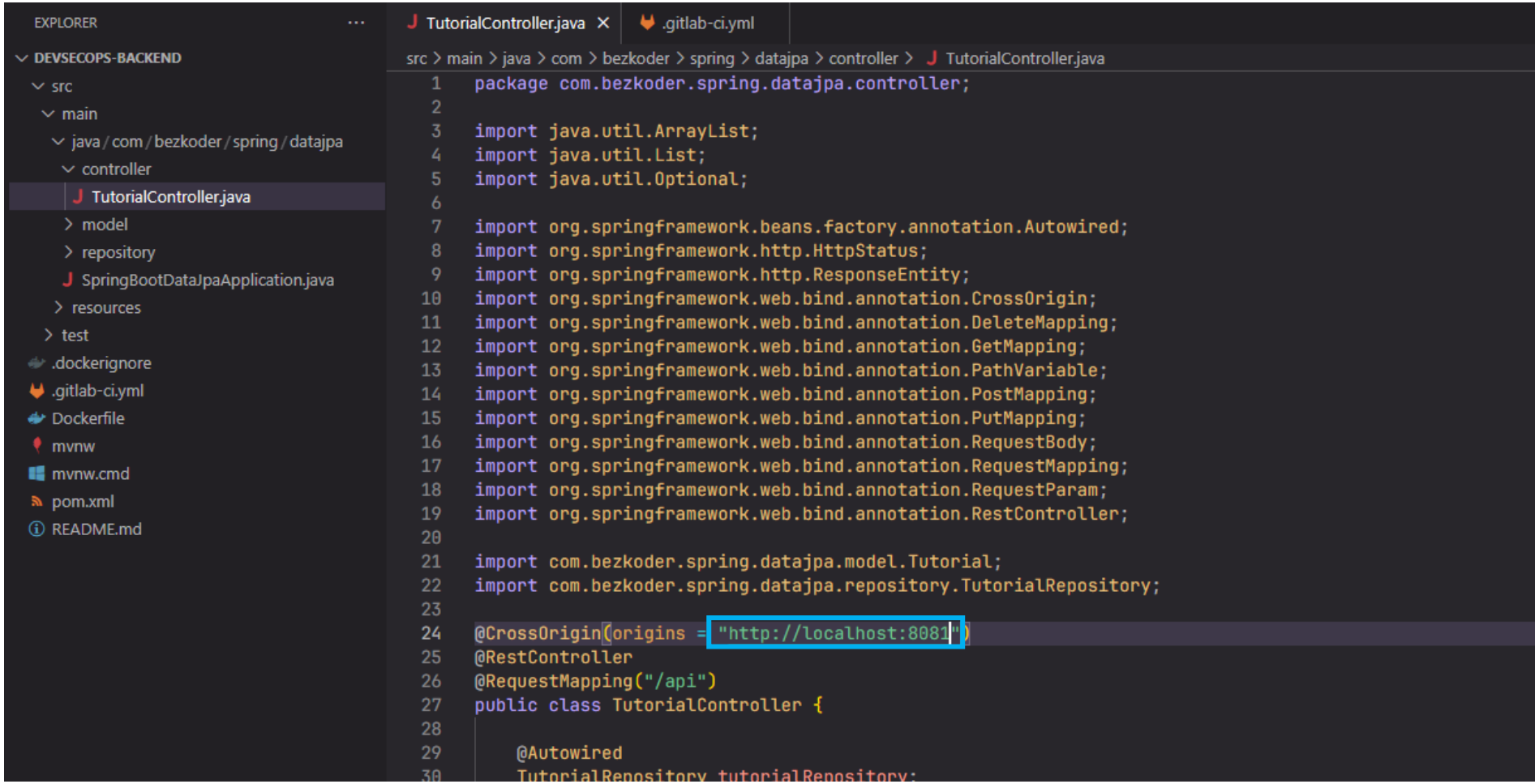
"Sleep 30" command is so important, I spent an entire day running the pipeline and the backend starts only for 10 seconds before exiting. Only to find out that the MySQL container doesn't find the enough time to run correctly, so when the backend attempts to connect to MySQL it simply not running yet.



3.1-Gitlab-runner

Backend

Before we commit the changes we have to make sure that the file TutorialController.java has for the origin http://localhost:8081:



3.1-Gitlab-runner

Frontend

For the frontend we will make only two changes for the pipeline code:

.gitlab-ci.yml

```yaml
image: docker:latest

services:
  - docker:dind

stages:
  - package
  - deploy

variables:
  DOCKER_HOST: tcp://172.17.0.1:2375
  DOCKER_DRIVER: overlay2

before_script:
  - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin

docker-build:
  stage: package
  script:
    - docker build -t mouadtigmouti/devsecops-frontend .
    - docker push mouadtigmouti/devsecops-frontend


docker-deploy:
  stage: deploy
  before_script:
    - docker stop devsecops-frontend || true
    - docker rm devsecops-frontend || true
  script:
    - docker run -d -p 8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
    - sleep 10
    - docker logs devsecops-frontend

```

We added these variables so that the runner can communicate with the local Docker, using the specified Docker host (tcp://172.17.0.1:2375) and the storage driver overlay2 that is responsible for managing the filesystem space used by Docker containers

To connect the frontend to the rest of the app (backend & MySQL)

3.1-Gitlab-runner

Frontend

Also, we need to modify the file tutorial.service.ts and provide it with the port where the backend runs, which is : http://localhost:8080/api/tutorials:

3.1-Gitlab-runner

Frontend

Without forgetting to config the database, it is necessary to ensure that the backend can communicate with the MySQL container. I learned that the hard way.

To achieve this, we connect all containers through the network called backend_network that we created earlier. Now, we need to obtain the IP address of the MySQL container within this network and provide it to the backend.

```
mouad@mouad-None:~$ docker network inspect backend_network
[
    {
        "Name": "backend_network",
        "Id": "f37d9fcdbceb2c4d3752464ff0261d6283a42fdfa5bce50ffbe8394aae88d171",
        "Created": "2024-01-06T10:56:55.802700385+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "766d6ba55bbccd1c686bfe4e0b4b939b0a3bb31094005013fb731016beffb2b0": {
                "Name": "mysql_docker",
                "EndpointID": "0361ae3b977a6f0c07d1e83ccd5b83de74180f30a236a683c0a2c1d19b506c61",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
```

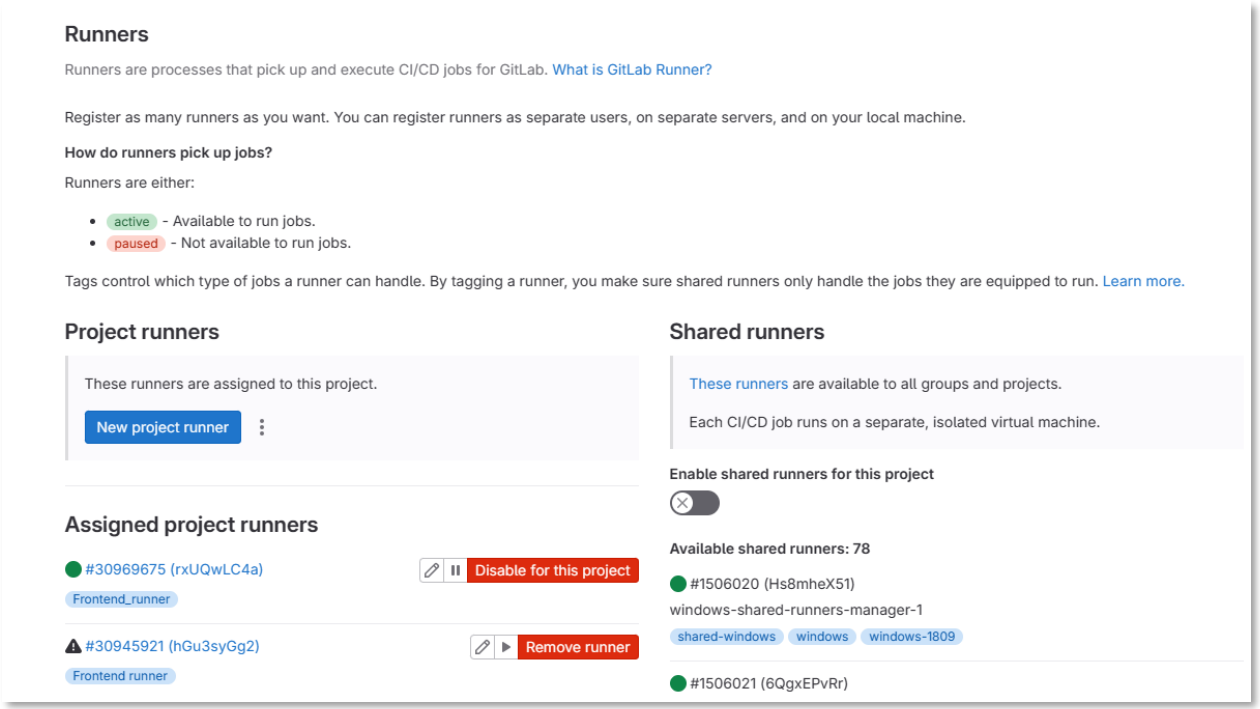
Now we simply add the address 172.19.0.2 to the application.properties file.

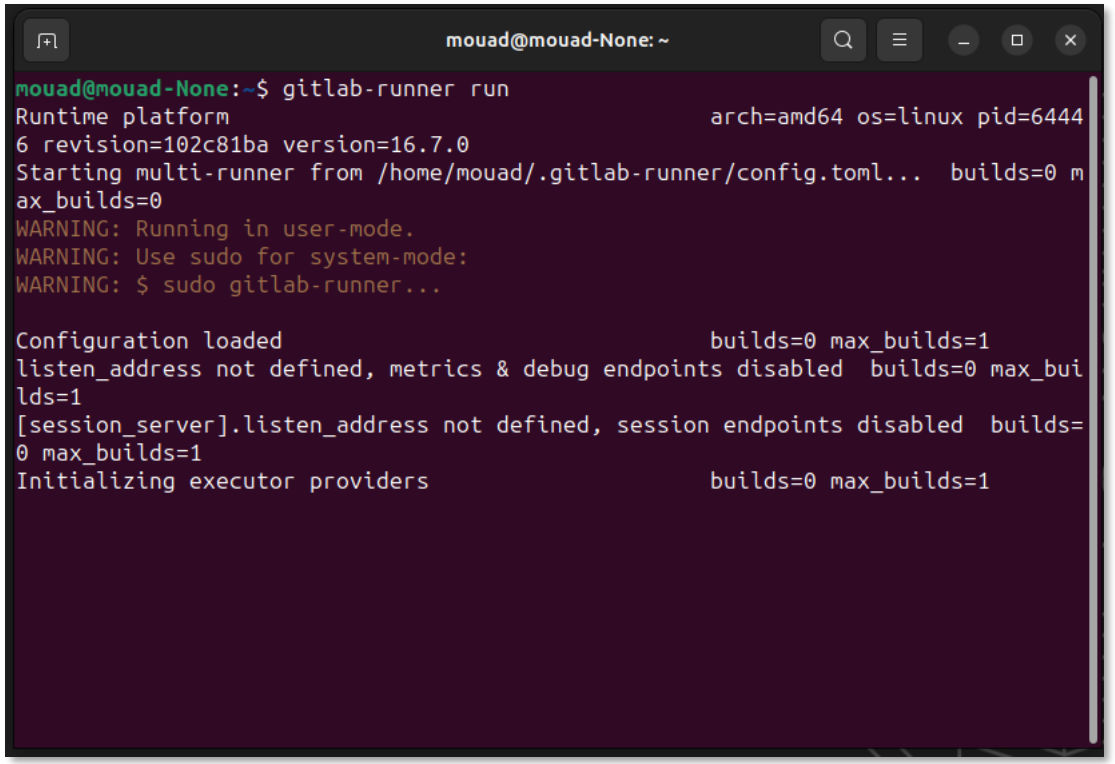
```
.gitlab-ci.yml  TutorialController.java  application.properties

src > main > resources > application.properties
1  spring.datasource.url= jdbc:mysql://172.19.0.2:3306/testdb?useSSL=false&allowPublicKeyRetrieval=true
2  spring.datasource.username= root
3  spring.datasource.password= ICCN2023
4
5  spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQLDialect
6  spring.jpa.hibernate.ddl-auto= update
7
```

To make sure that the MySQL container take the same address everytime, we just need to start the pipeline of the backend first.

3.1-Gitlab-runner

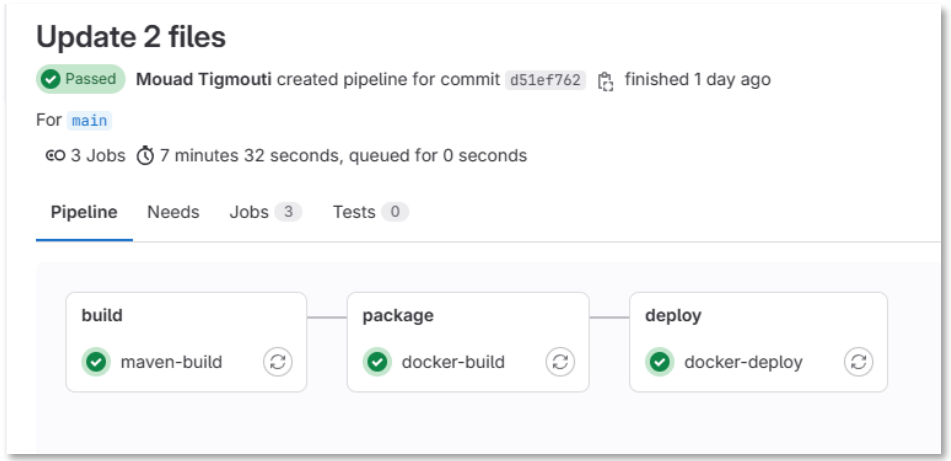
With our local runner prepared and listening, we can commit the changes for the backend first and then the frontend



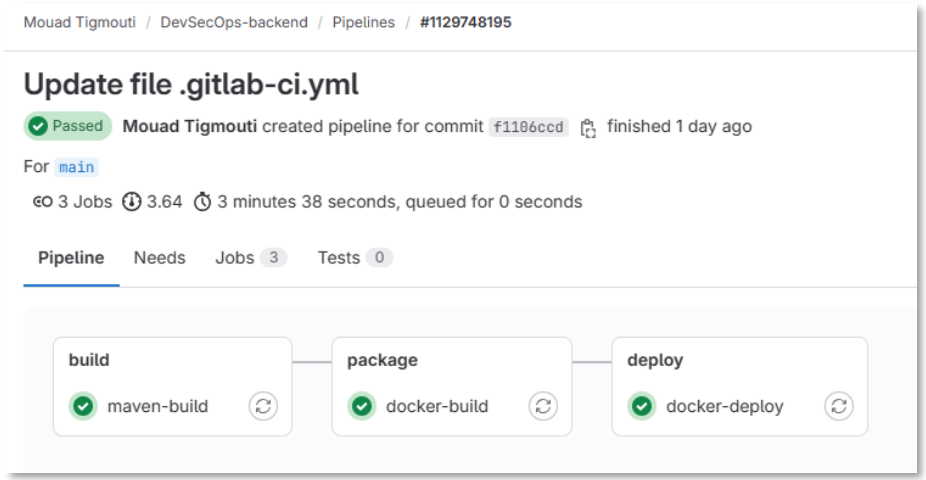


3.1-Gitlab-runner

For our local runner, the execution took 7 minutes. This duration depends on the local machine, specifically because it has an HDD hard disk. Approximately 5 minutes of the total time were spent in the push job, primarily due to the internet upload speed and the read/ write speed of the hard disk.



Our local runner

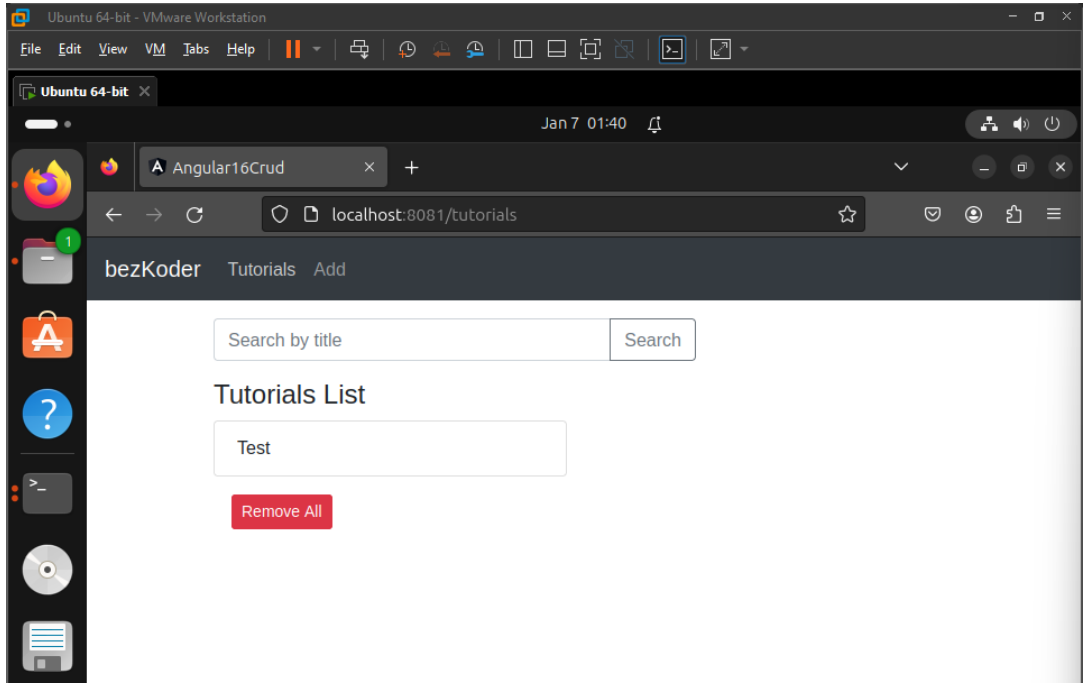


Gitlab shared runner

3.1-Gitlab-runner

Here is the result :

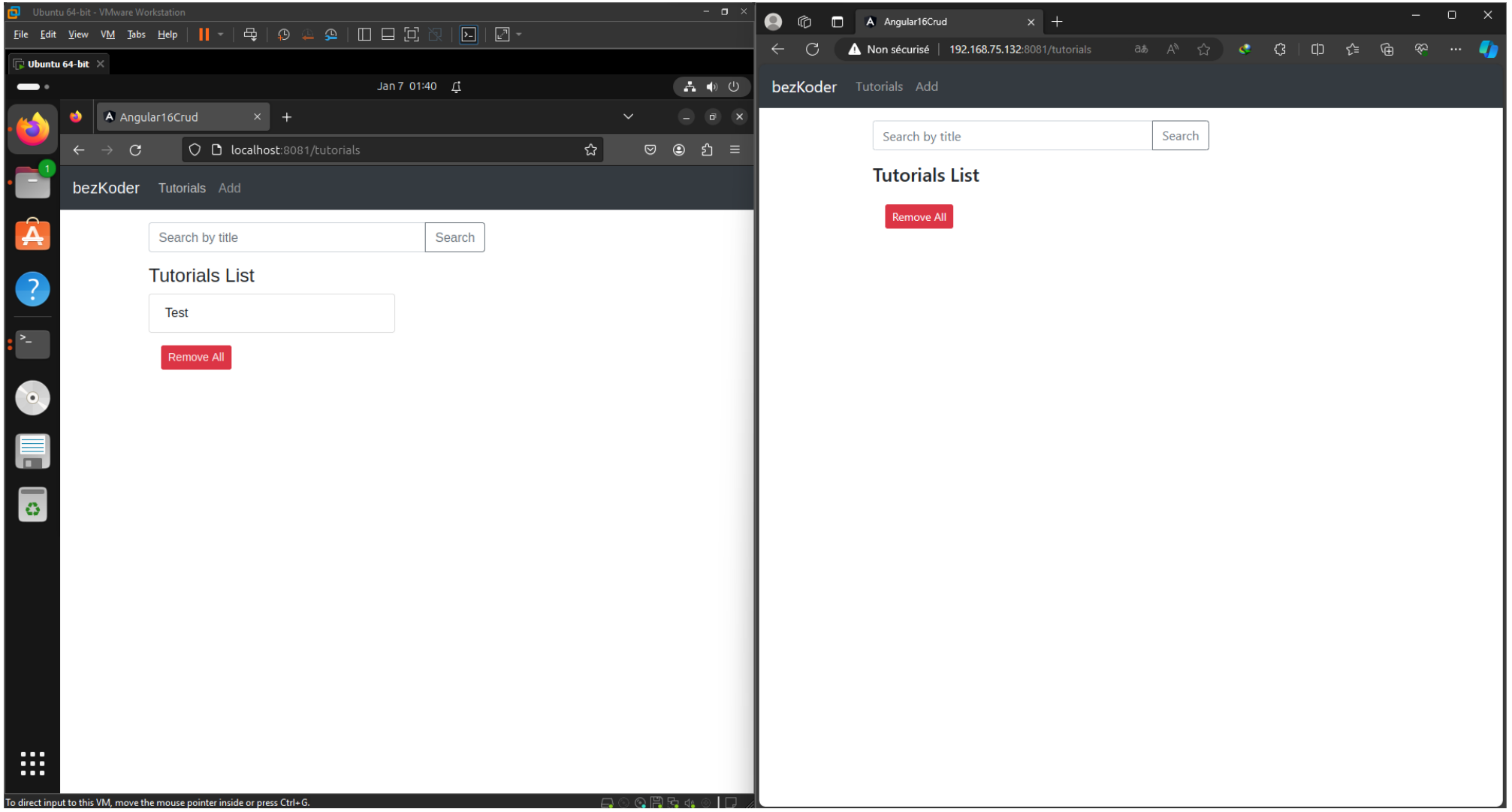
```
mouad@mouad-None:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED       STATUS          PORTS                                                  NAMES
979e3e5224fb   mysql                                "docker-entrypoint.s…"   2 hours ago   Up 47 minutes   3306/tcp, 33060/tcp                                    mysql_docker
d224014553e9   mouadtigmouti/devsecops-backend      "java -jar spring-bo…"   2 hours ago   Up 47 minutes   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 8081/tcp    devsecops-backend
1c6d42a9cf87   mouadtigmouti/devsecops-frontend     "docker-entrypoint.s…"   2 hours ago   Up 46 minutes   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp              devsecops-frontend
mouad@mouad-None:~$
```



With this configuration everything should work fine inside the VM.

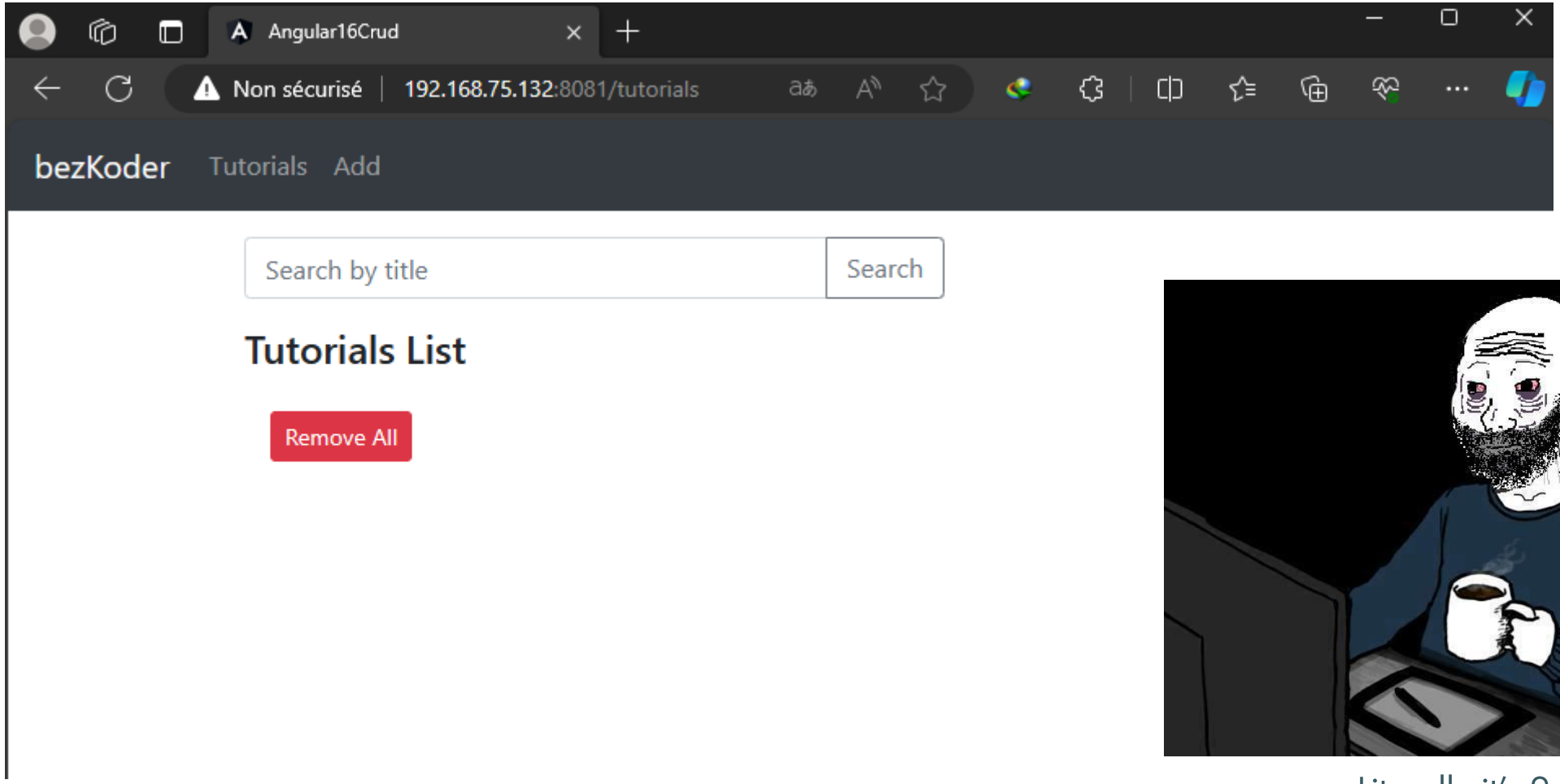
3.1-Gitlab-runner

Now it is working, but only inside of the VM machine :



3.1-Gitlab-runner

The host machine has only access to the frontend but not the backend.



Literally it's 2.30 AM

3.1-Gitlab-runner

To solve this, we simply need to use the IP address of the VM machine instead of localhost, let's change that in the frontend and backend files.

First we need the VM IP address :
It is 192.168.75.132
Now we change the files

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.75.132  netmask 255.255.255.0  broadcast 192.168.75.255
        inet6 fe80::20c:29ff:fe37:2c7f  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:37:2c:7f  txqueuelen 1000  (Ethernet)
        RX packets 32612  bytes 21140898 (21.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 31922  bytes 15686969 (15.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Dockerfile | TutorialController.java | .gitlab-ci.yml

src > main > java > com > bezkoder > spring > datajpa > controller > TutorialController.java

```java
package com.bezkoder.spring.datajpa.controller;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.bezkoder.spring.datajpa.model.Tutorial;
import com.bezkoder.spring.datajpa.repository.TutorialRepository;

@CrossOrigin(origins = "http://192.168.75.132:8081")
@RestController
@RequestMapping("/api")
public class TutorialController {
```

Dockerfile | tutorial.service.ts

src > app > services > tutorial.service.ts > baseUrl

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Tutorial } from '../models/tutorial.model';

const baseUrl = 'http://192.168.75.132:8080/api/tutorials';

@Injectable({
  providedIn: 'root',
})
```

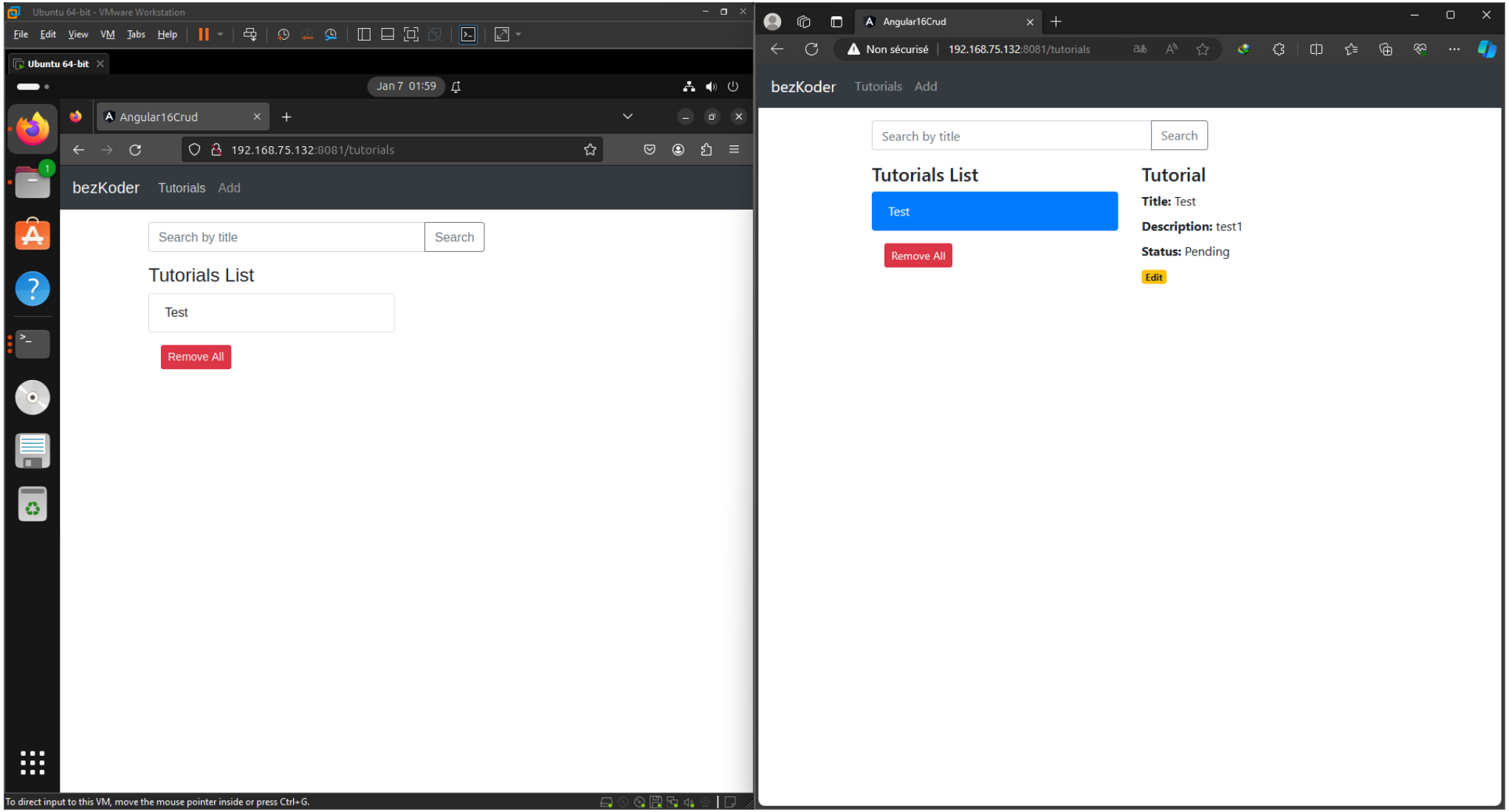
3.1-Gitlab-runner

At this stage, these changes are enough to fix the problem.
After making the changes, I forgot to update the deploy job so that the frontend and backend run with the VM IP address.
Since this oversight is in the deploy section, we can simply stop the containers and restart them with the correct values:

```
mouad@mouad-None: ~
mouad@mouad-None:~$ docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS                                                 N
AMES
0fc66d52fd4e   mouadtigmouti/devsecops-backend    "java -jar spring-bo…"   3 minutes ago    Up 3 minutes    0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 8081/tcp   d
evsecops-backend
3a47bdee7c3e   mysql                              "docker-entrypoint.s…"   4 minutes ago    Up 4 minutes    3306/tcp, 33060/tcp                                   m
ysql_docker
d09360524af8   mouadtigmouti/devsecops-frontend   "docker-entrypoint.s…"   26 minutes ago   Up 26 minutes   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp             d
evsecops-frontend
mouad@mouad-None:~$ docker stop 0fc66d52fd4e d09360524af8
0fc66d52fd4e
d09360524af8
mouad@mouad-None:~$ docker rm -f 0fc66d52fd4e d09360524af8
0fc66d52fd4e
d09360524af8
mouad@mouad-None:~$ docker run -d --name devsecops-backend --network backend_network -p 192.168.75.132:8080:8080 mouadtigmouti/devsecops-backend
f566f74f287e1d75b3807cd7a67244125742c4168e8b1f483a9336244baacd7e
mouad@mouad-None:~$ docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
a62f4cd9508a1d30b5739306e95356f2c0dd86548c8e86b435dd186fc38a624b
mouad@mouad-None:~$ docker ps
CONTAINER ID   IMAGE                              COMMAND                  CREATED          STATUS          PORTS                                                 NAMES
a62f4cd9508a   mouadtigmouti/devsecops-frontend   "docker-entrypoint.s…"   4 seconds ago    Up 3 seconds    192.168.75.132:8081->8081/tcp                         devsecops-fro
ntend
f566f74f287e   mouadtigmouti/devsecops-backend    "java -jar spring-bo…"   36 seconds ago   Up 35 seconds   192.168.75.132:8080->8080/tcp, 8081/tcp               devsecops-bac
kend
3a47bdee7c3e   mysql                              "docker-entrypoint.s…"   7 minutes ago    Up 7 minutes    3306/tcp, 33060/tcp                                   mysql_docker
mouad@mouad-None:~$
```

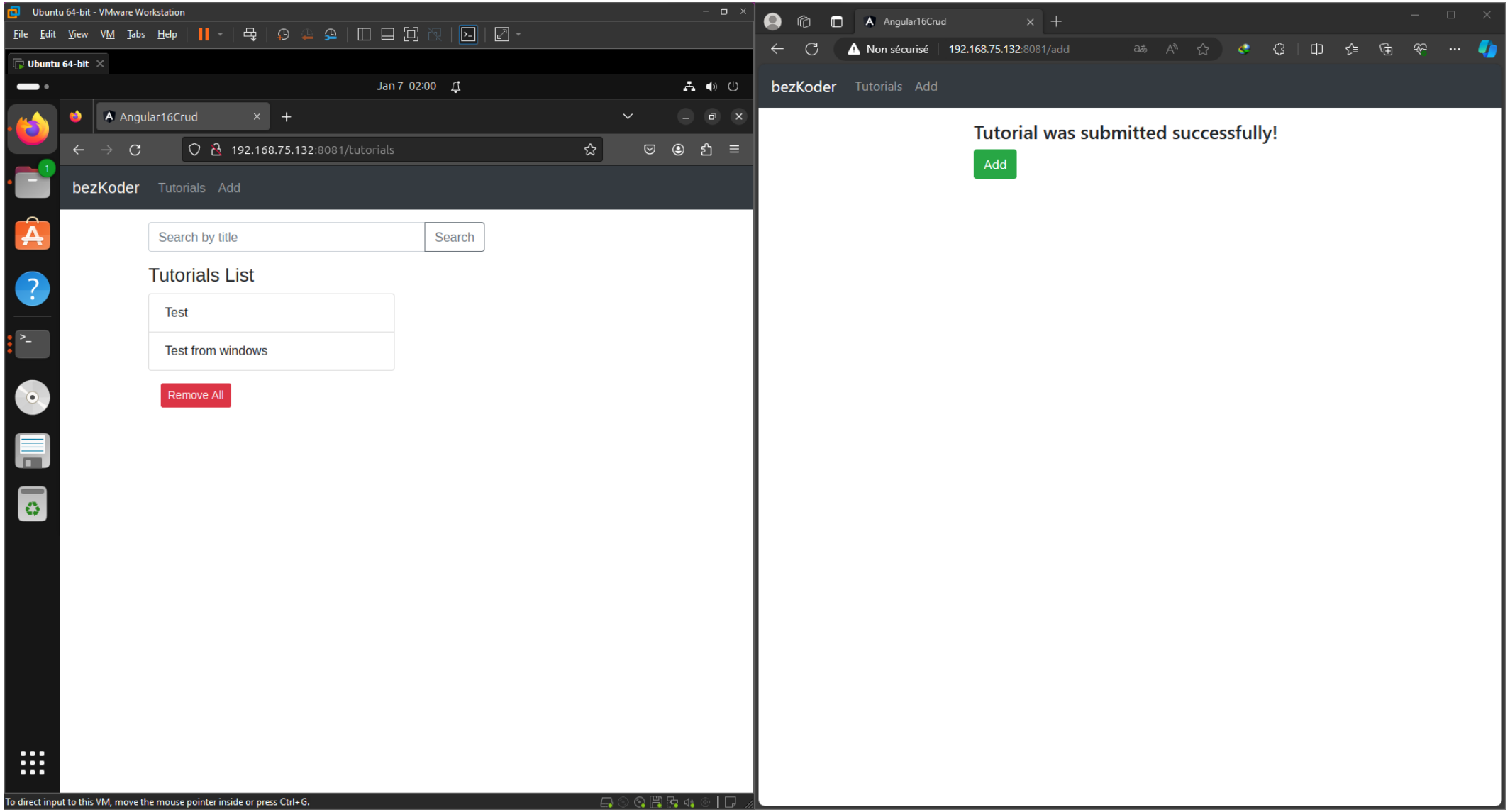
3.1-Gitlab-runner

Here is the result :



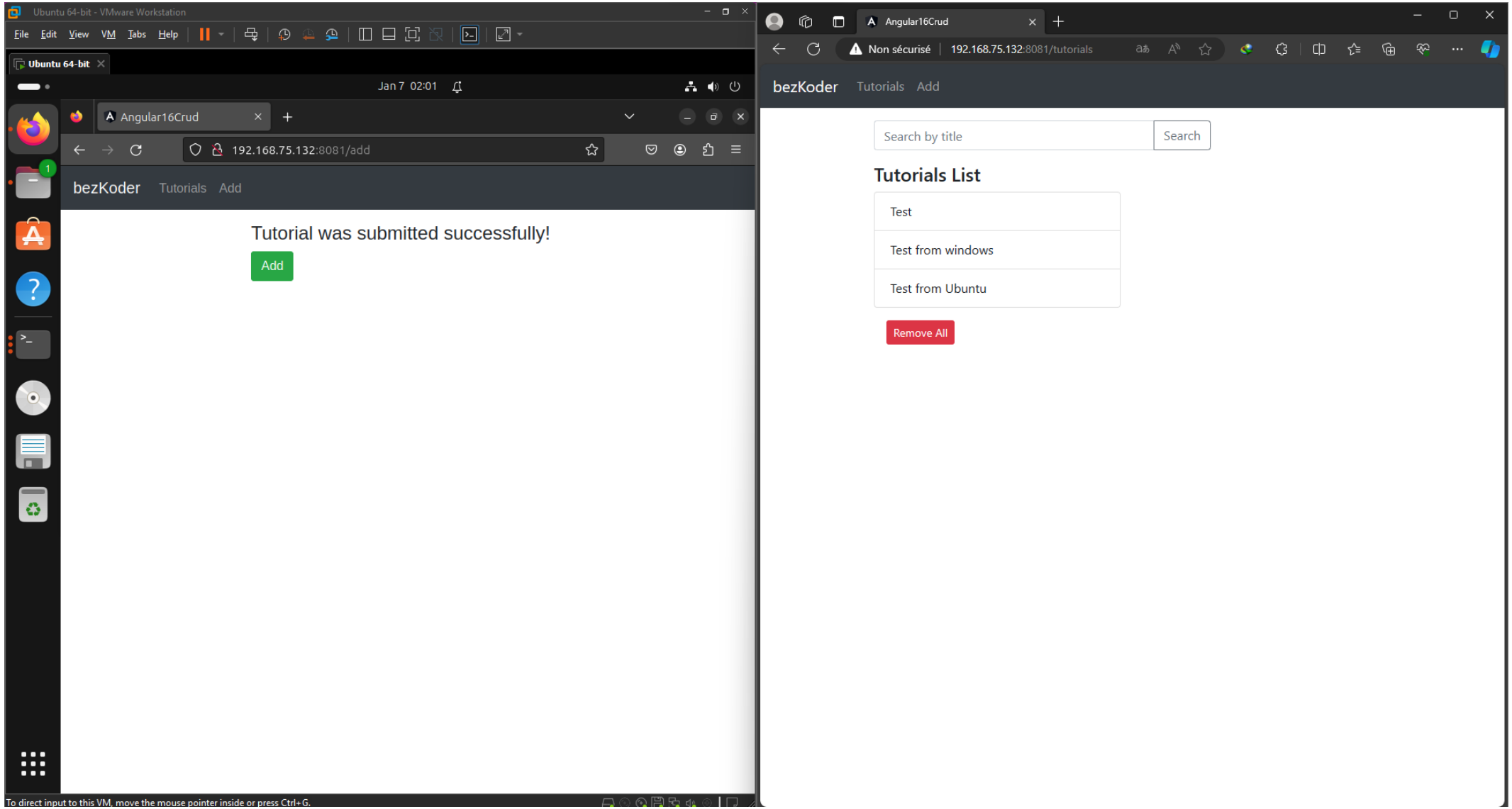
3.1-Gitlab-runner

Now we can affect changes from both machines



3.1-Gitlab-runner

Now we can affect changes from both machines



3.1-Generic Jobs

To explain the important of generic jobs let's take a look at the backend pipeline code for example :

.gitlab-ci.yml

```yaml
image: docker:latest
services:
  - docker:dind

stages:
  - build
  - package
  - deploy

variables:
  DOCKER_HOST: tcp://172.17.0.1:2375
  DOCKER_DRIVER: overlay2

maven-build:
  image: maven:latest
  stage: build
  script: "mvn clean package -B"
  artifacts:
    paths:
      - target/*.jar

docker-build:

  stage: package
  rules:
    - exists:
        - Dockerfile
  before_script:
        - docker login -u mouadtigmouti -p $DOCKER_Password

  script:
        - docker build -t mouadtigmouti/devsecops-backend .
        - docker push mouadtigmouti/devsecops-backend
```

```yaml
docker-deploy:
  stage: deploy
  before_script:
    - docker stop devsecops-backend || true
    - docker rm devsecops-backend || true
    - docker stop mysql_docker || true
    - docker rm mysql_docker || true
    - docker network create backend_network || true
  script:


    - docker run -d --name mysql_docker --network backend_network -e MYSQL_ROOT_PASSWORD=$MYSQL_PASSWORD -e MYSQL_DATABASE=testdb
      -v /mysql_data:/var/lib/mysql mysql
    - sleep 30
    - docker run -d --name devsecops-backend --network backend_network -p 8080:8080 mouadtigmouti/devsecops-backend
    - docker ps

```

The current pipeline is quite long, and it's a bit messy to have all the stages in one place. It becomes even more challenging when we consider adding the security tests section. It's not the most efficient setup to have all the stages in just one file. That's why we're opting to use generic jobs to improve our code.

3.2-Generic Jobs

Generic jobs in GitLab CI/CD are job templates that encapsulate common tasks, such as building and deploying Docker images. These templates are designed to be included in the main pipeline configuration, allowing for modular and reusable CI/CD definitions.

Generic jobs are typically stored in a separate folder, often named "templates." Each job is defined in its own file, containing the specific steps and configurations for a particular task, such as building a Docker image or deploying an application.

They have many benefits such as :

+ Reducing the risk of configuration errors and promoting best practices.
+ Reducing the likelihood of errors introduced during updates.
+ make it easier to scale CI/CD configurations by providing a standardized approach that can be applied to new projects.

3.2-Generic Jobs

First, let's apply the concept of generic jobs to our frontend pipeline

.gitlab-ci.yml

```yaml
image: docker:latest

services:
  - docker:dind

stages:
  - package
  - deploy

variables:
  DOCKER_HOST: tcp://172.17.0.1:2375
  DOCKER_DRIVER: overlay2

before_script:
  - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin

docker-build:
  stage: package
  script:
    - docker build -t mouadtigmouti/devsecops-frontend .
    - docker push mouadtigmouti/devsecops-frontend


docker-deploy:
  stage: deploy
  before_script:
    - docker stop devsecops-frontend || true
    - docker rm devsecops-frontend || true
  script:
    - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
    - sleep 10
    - docker logs devsecops-frontend
```

We will start by creating a folder in GitLab repository called templates.

Inside of this folder we will create two files .yml, one for the package stage and one for the deploy stage.

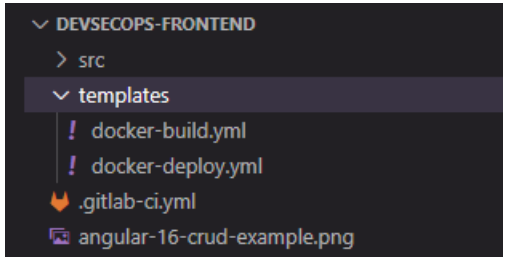
3.2-Generic Jobs

First, let's apply the concept of generic jobs to our frontend pipeline

.gitlab-ci.yml

```yaml
image: docker:latest

services:
  - docker:dind

stages:
  - package
  - deploy

variables:
  DOCKER_HOST: tcp://172.17.0.1:2375
  DOCKER_DRIVER: overlay2

before_script:
  - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin

docker-build:
  stage: package
  script:
    - docker build -t mouadtigmouti/devsecops-frontend .
    - docker push mouadtigmouti/devsecops-frontend


docker-deploy:
  stage: deploy
  before_script:
    - docker stop devsecops-frontend || true
    - docker rm devsecops-frontend || true
  script:
    - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
    - sleep 10
    - docker logs devsecops-frontend
```

We will start by creating a folder in GitLab repository called templates.
Inside of this folder we will create two files .yml, one for the package stage and one for the deploy stage.



3.2-Generic Jobs

Let's take a look inside each file :

docker-build.yml

```yaml
docker-build:
  stage: package
  before_script:
  - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
  script:
    - docker build -t mouadtigmouti/devsecops-frontend .
    - docker push mouadtigmouti/devsecops-frontend
```

docker-deploy.yml

```yaml
docker-deploy:
  stage: deploy
  before_script:
    - docker stop devsecops-frontend || true
    - docker rm devsecops-frontend || true
  script:
    - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
    - sleep 10
    - docker logs devsecops-frontend
```

.gitlab-ci.yml

```yaml
image: docker:latest

services:
  - docker:dind

include:
  - templates/docker-build.yml
  - templates/docker-deploy.yml

stages:
  - package
  - deploy

variables:
  DOCKER_HOST: tcp://172.17.0.1:2375
  DOCKER_DRIVER: overlay2
```

The pipeline code now looks lightweight and much better

My colleague is currently working on the security tests for the backend. Once she finishes, we will present you with the optimized pipeline in Section 3, including the incorporation of the security tests.

3.2-Generic Jobs

Let's take a look inside each file :

docker-build.yml

```yaml
docker-build:
  stage: package
  before_script:
  - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
  script:
    - docker build -t mouadtigmouti/devsecops-frontend .
    - docker push mouadtigmouti/devsecops-frontend
```

docker-deploy.yml

```yaml
docker-deploy:
  stage: deploy
  before_script:
    - docker stop devsecops-frontend || true
    - docker rm devsecops-frontend || true
  script:
    - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
    - sleep 10
    - docker logs devsecops-frontend
```

.gitlab-ci.yml

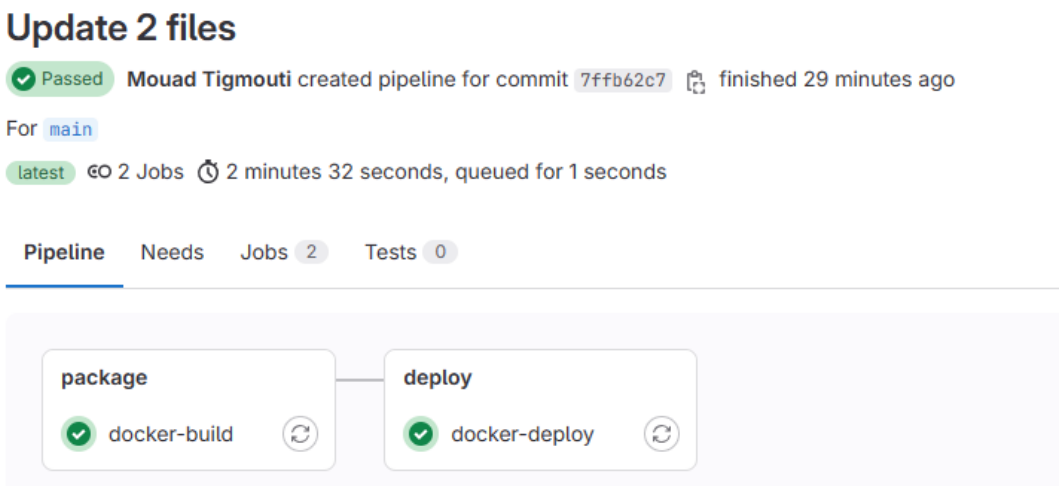
```yaml
image: docker:latest

services:
  - docker:dind

include:
  - templates/docker-build.yml
  - templates/docker-deploy.yml

stages:
  - package
  - deploy

variables:
  DOCKER_HOST: tcp://172.17.0.1:2375
  DOCKER_DRIVER: overlay2
```

The pipeline code now looks lightweight and much better

My colleague is currently working on the security tests for the backend. Once she finishes, we will present you with the optimized pipeline in Section 3, including the incorporation of the security tests.

3.2-Generic Jobs

It works perfectly fine.




```
mouad@mouad-None: ~
mouad@mouad-None:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                                NAMES
dc1efb6715e9   mouadtigmouti/devsecops-frontend   "docker-entrypoint.s…"   31 minutes ago   Up 31 minutes   192.168.75.132:8081->8081/tcp   devsecops-frontend
mouad@mouad-None:~$
```

3.3-Rules

GitLab CI/CD rules define when jobs should be created and executed in the pipeline. These rules are based on conditions like branch names, tags, and changes in specific files, allowing for customized and conditional automation of CI/CD processes.

Let's add some rules to the frontend pipeline

.gitlab-ci.yml | tutorial.service.ts | docker-deploy.yml

templates > docker-deploy.yml

```yaml
docker-deploy:
  stage: deploy
  before_script:
    - docker stop devsecops-frontend || true
    - docker rm devsecops-frontend || true
  rules:
    - exists:
        - Dockerfile
    - exists:
        - '$CI_COMMIT_BRANCH == "main"'
  script:
    - docker run -d -p 8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
    - sleep 10
    - docker logs devsecops-frontend
```

These rules ensure that a Dockerfile exists for building Docker images, and the CI process runs only when changes are made to the main branch.

These are simple rules; as we move into the next section, we will be using more rules.

1. Backend :
   - 1.1- SAST
   - 1.2- SCA
   - 1.3- CS
   - 1.4- DAST
2. Frontend
   - 2.1- SAST
   - 2.2- SCA
   - 2.3- CS
   - 2.4- DAST

Section 3

1.1-SAST

Static application security testing is a type of software test used for inspecting and analyzing code to identify security vulnerabilities. Software security tools — such as static code analyzers — scan your code as it's being written to identify potential weaknesses, errors and bugs.

For this project we are going to use theses tools :

TruffleHog

Bandit

Semgrep

1.1-SAST

TruffleHog

Embed TruffleHog into Gitlab CI/CD:

```
git-secrets:
 stage: build
 script:
  - docker run -v $(pwd):/src --rm hysnsec/trufflehog filesystem --directory=/src --json | tee trufflehog-output.json
 artifacts:
  paths: [trufflehog-output.json]
  when: always
  expire_in: one week
```

1.1-SAST

TruffleHog

The result of the test :

{"level":"info-0","ts":"2024-01-09T15:18:39Z","logger":"trufflehog","msg":"--directory flag is deprecated, please pass directories as arguments"}
{"level":"info-0","ts":"2024-01-09T15:18:39Z","logger":"trufflehog","msg":"running source","source_manager_worker_id":"iXgIF","with_units":true}
{"level":"info-0","ts":"2024-01-09T15:18:39Z","logger":"trufflehog","msg":"finished scanning","chunks":14,"bytes":105566,"verified_secrets":0,"unverified_secrets":0,"scan_duration":"417.125512ms"}


```
$ docker run -v $(pwd):/src --rm hysnsec/trufflehog filesystem --directory=/src --json | tee trufflehog-output.json
{"level":"info-0","ts":"2024-01-09T15:18:39Z","logger":"trufflehog","msg":"--directory flag is deprecated, please pass directories as arguments"}
{"level":"info-0","ts":"2024-01-09T15:18:39Z","logger":"trufflehog","msg":"running source","source_manager_worker_id":"iXgIF","with_units":true}
{"level":"info-0","ts":"2024-01-09T15:18:39Z","logger":"trufflehog","msg":"finished scanning","chunks":14,"bytes":105566,"verified_secrets":0,"unverified_s
ecrets":0,"scan_duration":"417.125512ms"}
```

1.1-SAST

Bandit

The Bandit is a tool designed to find common security issues in Python code.
To do this Bandit, processes each file, builds an AST, and runs appropriate plugins against the AST nodes. Once Bandit has finished scanning all the files it generates a report.

Embed Bandit into Gitlab CI/CD:

```
sast-bandit:
  stage: build
  before_script:
    - apk update
    - apk add --no-cache python3 py3-pip
  script:
    - pip3 install bandit==1.7.4 --break-system-packages
    - pip3 install requests --break-system-packages
    - bandit -r . -f json | tee bandit-output.json
    - cat bandit-output.json
  artifacts:
    paths: [bandit-output.json]
    when: always
```

1.1-SAST

Bandit

The result of the test :

```
$ bandit -r . -f json | tee bandit-output.json
[main]      INFO      profile include tests: None
[main]      INFO      profile exclude tests: None
[main]      INFO      cli include tests: None
[main]      INFO      cli exclude tests: None
{
 "errors": [],
 "generated_at": "2024-01-09T15:19:38Z",
 "metrics": {
  "./upload-results.py": {
   "CONFIDENCE.HIGH": 0,
   "CONFIDENCE.LOW": 0,
   "CONFIDENCE.MEDIUM": 0,
   "CONFIDENCE.UNDEFINED": 0,
   "SEVERITY.HIGH": 0,
   "SEVERITY.LOW": 0,
   "SEVERITY.MEDIUM": 0,
   "SEVERITY.UNDEFINED": 0,
   "loc": 54,
   "nosec": 0,
   "skipped_tests": 0
  },
```

```
 "_totals": {
   "CONFIDENCE.HIGH": 0,
   "CONFIDENCE.LOW": 0,
   "CONFIDENCE.MEDIUM": 0,
   "CONFIDENCE.UNDEFINED": 0,
   "SEVERITY.HIGH": 0,
   "SEVERITY.LOW": 0,
   "SEVERITY.MEDIUM": 0,
   "SEVERITY.UNDEFINED": 0,
   "loc": 54,
   "nosec": 0,
   "skipped_tests": 0
  }
 },
 "results": []
}
```

```
$ bandit -r . -f json | tee bandit-output.json
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
{
  "errors": [],
  "generated_at": "2024-01-09T15:19:38Z",
  "metrics": {
    "./upload-results.py": {
      "CONFIDENCE.HIGH": 0,
      "CONFIDENCE.LOW": 0,
      "CONFIDENCE.MEDIUM": 0,
      "CONFIDENCE.UNDEFINED": 0,
      "SEVERITY.HIGH": 0,
      "SEVERITY.LOW": 0,
      "SEVERITY.MEDIUM": 0,
      "SEVERITY.UNDEFINED": 0,
      "loc": 54,
      "nosec": 0,
      "skipped_tests": 0
    },
    "_totals": {
      "CONFIDENCE.HIGH": 0,
      "CONFIDENCE.LOW": 0,
      "CONFIDENCE.MEDIUM": 0,
      "CONFIDENCE.UNDEFINED": 0,
      "SEVERITY.HIGH": 0,
      "SEVERITY.LOW": 0,
      "SEVERITY.MEDIUM": 0,
      "SEVERITY.UNDEFINED": 0,
      "loc": 54,
      "nosec": 0,
      "skipped_tests": 0
    }
  },
```

1.1-SAST

Semgrep

Embed Semgrep into Gitlab CI/CD:

```
semgrep:
  stage: build
  script:
   - docker run --rm -v ${PWD}:/src returntocorp/semgrep semgrep --config auto --output semgrep-output.json --json --metrics=off
```

1.1-SAST

Semgrep

The result of the test :

METRICS: Using configs from the Registry (like --config=p/ci) reports pseudonymous rule metrics to semgrep.dev.
To disable Registry rule metrics, use "--metrics=off".
Using configs only from local files (like --config=xyz.yml) does not enable metrics.
More information: https://semgrep.dev/docs/metrics

| Scan Status |

Scanning 7 files tracked by git with 1102 Code rules:

Language	Rules	Files	Origin	Rules
<multilang>	55	14	Community	1102
json	4	4		
yaml	28	1		

| Scan Summary |

(need more rules? `semgrep login` for additional free Semgrep Registry rules)
Ran 86 rules on 7 files: 0 findings.

```
$ docker run --rm -v ${PWD}:/src returntocorp/semgrep semgrep --config auto --output semgrep-output.json --json
METRICS: Using configs from the Registry (like --config=p/ci) reports pseudonymous rule metrics to semgrep.dev.
To disable Registry rule metrics, use "--metrics=off".
Using configs only from local files (like --config=xyz.yml) does not enable metrics.
More information: https://semgrep.dev/docs/metrics


┌──────────────┐
│ Scan Status  │
└──────────────┘
  Scanning 7 files tracked by git with 1102 Code rules:

  Language      Rules   Files          Origin      Rules
 ─────────────────────────────        ──────────────────
  <multilang>      55      14          Community    1102
  json              4       4
  yaml             28       1


┌──────────────┐
│ Scan Summary │
└──────────────┘
(need more rules? `semgrep login` for additional free Semgrep Registry rules)
Ran 86 rules on 7 files: 0 findings.
```

1.2-SCA

Software Composition Analysis (SCA) provides visibility into the open source components and libraries being incorporated into the software that development teams create.

For this project we are going to use theses tools for the SCA phase :

retireJS

Bandit

1.2-SCA

retireJS

Embed retireJS into Gitlab CI/CD:

```
sast-RetireJS:
  stage: build
  image: node:alpine3.10
  script:
    - npm install
    - npm install -g retire # Install retirejs npm package.
    - retire --outputformat json --outputpath retirejs-report.json --severity high
    - cat retirejs-report.json
  artifacts:
    paths: [retirejs-report.json]
    when: always
```

1.2-SCA

retireJS

The result of the test :

{"version":"4.3.4","start":"2024-01-09T15:31:01.504Z","data":[],"messages":[],"errors":[],"time":0.261}

```
$ cat retirejs-report.json
{"version":"4.3.4","start":"2024-01-09T15:31:01.504Z","data":[],"messages":[],"errors":[],"time":0.261}
```

1.2-SCA

snyk

Embed snyk into Gitlab CI/CD:

```
oast-snyk:
  stage: build
  image: node:alpine3.10
  before_script:
    - wget -O snyk https://github.com/snyk/cli/releases/download/v1.1156.0/snyk-alpine
    - chmod +x snyk
    - mv snyk /usr/local/bin/
  script:
    - npm install
    - snyk auth 35589080-01a0-4492-af3d-82ee5f1140d0
    - snyk test --json > snyk-results.json
    - cat snyk-results.json
  artifacts:
    paths: [snyk-results.json]
    when: always
    expire_in: one week
```

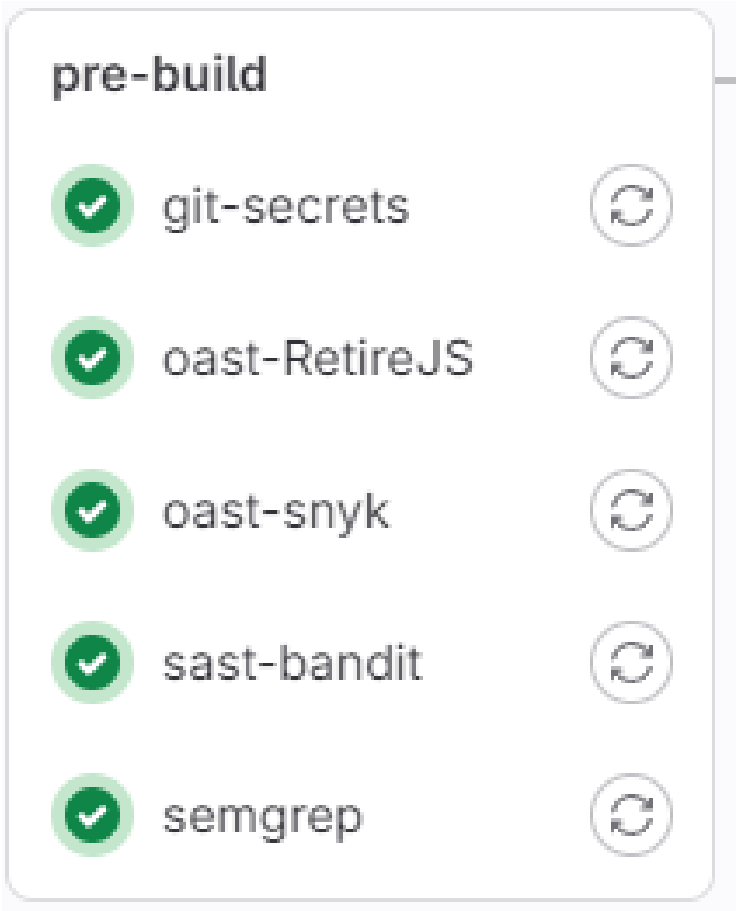
1.2-SCA

snyk

The result of the test :

```
$ snyk auth 35589080-01a0-4492-af3d-82ee5f1140d0
Your account has been authenticated. Snyk is now ready to be used.
$ snyk test --json > snyk-results.json
$ cat snyk-results.json
{
  "vulnerabilities": [],
  "ok": true,
  "dependencyCount": 0,
  "org": "thehuntison000",
  "policy": "# Snyk (https://snyk.io) policy file, patches or ignores known vulnerabilities.\nversion: v1.25.1\nignore: {}\npatch: {}\n",
  "isPrivate": true,
  "licensesPolicy": {
    "severities": {},
    "orgLicenseRules": {
      "AGPL-1.0": {
        "licenseType": "AGPL-1.0",
        "severity": "high",
        "instructions": ""
      },
      "AGPL-3.0": {
        "licenseType": "AGPL-3.0",
        "severity": "high",
        "instructions": ""
      },
      "Artistic-1.0": {
        "licenseType": "Artistic-1.0",
        "severity": "medium",
        "instructions": ""
      },
      "Artistic-2.0": {
        "licenseType": "Artistic-2.0",
        "severity": "medium",
        "instructions": ""
      },
      "CDDL-1.0": {
        "licenseType": "CDDL-1.0",
        "severity": "medium",
        "instructions": ""
      },
      "CPOL-1.02": {
        "licenseType": "CPOL-1.02",
        "severity": "high",
        "instructions": ""
      },
      "EPL-1.0": {
        "licenseType": "EPL-1.0",
        "severity": "medium",
        "instructions": ""
      },
```

The pipeline of the SAST and SCA phase :



1.3-Container Security

Container Scanning is often considered part of Software Composition Analysis (SCA). SCA can contain aspects of inspecting the items your code uses. These items typically include application and system dependencies that are almost always imported from external sources, rather than sourced from items you wrote yourself.

```
#Container Scanning

include:
  - template: Security/Container-Scanning.gitlab-ci.yml

container_scanning:
  variables:
    CS_IMAGE: mouadtigmouti/devsecops-backend
```

1.3-Container Security

The result of container security scan :

```
[INFO] [2024-01-09 19:31:20 +0000] [container-scanning]  >  Scanning container from registry mouadtigmouti/devsecops-backend for vulnerabilities with sever
ity level UNKNOWN or higher, with gcs 6.6.2 and Trivy Version: 0.48.1, advisories updated at 2024-01-09T12:13:28+00:00
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
|   STATUS   | CVE SEVERITY |   PACKAGE NAME  |     PACKAGE VERSION     |                                CVE DESCRIPTION                                |
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
| Unapproved |     Low      |       apt       |          2.2.4          | It was found that apt-key in apt, all versions, do not correctly valid        |
|            |              |                 |                         | ate gpg keys with the master keyring, leading to a potential man-in-th        |
|            |              |                 |                         |                              e-middle attack.                                 |
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
| Unapproved |     High     |       bash      |         5.1-2+b3        | A flaw was found in the bash package, where a heap-buffer overflow can        |
|            |              |                 |                         |  occur in valid parameter_transform. This issue may lead to memory pro        |
|            |              |                 |                         |                                    blems.                                     |
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
| Unapproved |     Low      |       bash      |         5.1-2+b3        |                          TEMP-0841856-B18BAF                                  |
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
| Unapproved |     Low      |     bsdutils    |   1:2.36.1-8+deb11u1    | A flaw was found in the util-linux chfn and chsh utilities when compil        |
|            |              |                 |                         | ed with Readline support. The Readline library uses an "INPUTRC" envir        |
|            |              |                 |                         | onment variable to get a path to the library config file. When the lib        |
|            |              |                 |                         | rary cannot parse the specified file, it prints an error message conta        |
|            |              |                 |                         | ining data from the file. This flaw allows an unprivileged user to rea        |
|            |              |                 |                         | d root-owned files, potentially leading to privilege escalation. This         |
|            |              |                 |                         |          flaw affects util-linux versions prior to 2.37.4.                    |
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
| Unapproved |     Low      |    coreutils    |         8.32-4+b1       | chroot in GNU coreutils, when used with --userspec, allows local users        |
|            |              |                 |                         |  to escape to the parent session via a crafted TIOCSTI ioctl call, whi        |
|            |              |                 |                         |         ch pushes characters to the terminal's input buffer.                  |
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
| Unapproved |     Low      |    coreutils    |         8.32-4+b1       | In GNU Coreutils through 8.29, chown-core.c in chown and chgrp does no        |
|            |              |                 |                         | t prevent replacement of a plain file with a symlink during use of the        |
|            |              |                 |                         |  POSIX "-R -L" options, which allows local users to modify the ownersh        |
|            |              |                 |                         |        ip of arbitrary files by leveraging a race condition.                  |
+------------+--------------+-----------------+-------------------------+-------------------------------------------------------------------------------+
| Unapproved |   Critical   |       dpkg      |         1.20.9          | Dpkg::Source::Archive in dpkg, the Debian package management system, b        |
|            |              |                 |                         | efore version 1.21.8, 1.20.10, 1.19.8, 1.18.26 is prone to a directory        |
|            |              |                 |                         |  traversal vulnerability. When extracting untrusted source packages in        |
|            |              |                 |                         |  v2 and v3 source package formats that include a debian.tar, the in-pl        |
```

1.4-DAST

Dynamic Application Security Testing (DAST) is an application security testing methodology in which the application is tested at runtime to discover security vulnerabilities. DAST tools don't have access to the application and API's source code, so they detect vulnerabilities by performing actual attacks, similar to a real hacker.

For this project we are going to use theses tools for the DAST phase:

Nikto

NMAP

Zed Atttack Proxy

1.4-DAST

Nikto

Embed Nikto into Gitlab CI/CD:

```
nikto:
  stage: test
  script:
    - docker pull hysnsec/nikto
    - docker run --rm -v $(pwd):/tmp hysnsec/nikto -h http://192.168.75.132:8083
```

1.4-DAST

Nikto

The result of the test :

```
$ docker run --rm -v $(pwd):/tmp hysnsec/nikto -h http://192.168.75.132:8083
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.75.132
+ Target Hostname:    192.168.75.132
+ Target Port:        8083
+ Start Time:         2024-01-09 16:28:26 (GMT0)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME typ
e. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /G8qOUp9m.php3: Uncommon header 'content-disposition' found, with contents: inline;filename=f.txt.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS .
+ HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ 8100 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:           2024-01-09 16:28:53 (GMT0) (27 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

1.4-DAST

NMAP

Nmap (“Network Mapper”) is a free and open source (license) utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Embed NMAP into Gitlab CI/CD:

```
nmap:
  stage: test
  script:
    - docker pull hysnsec/nmap
    - docker run --rm -v $(pwd):/tmp hysnsec/nmap -sSVC -A -p 8083 192.168.75.132
```

1.4-DAST

NMAP

The result of the test :

```
$ docker run --rm -v $(pwd):/tmp hysnsec/nmap -sSVC -A -p 8083 192.168.75.132
Starting Nmap 7.94 ( https://nmap.org ) at 2024-01-09 16:29 UTC
Nmap scan report for 192.168.75.132
Host is up (0.000062s latency).
PORT     STATE SERVICE VERSION
8083/tcp open  us-srv?
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 404
|     Vary: Origin
|     Vary: Access-Control-Request-Method
|     Vary: Access-Control-Request-Headers
|     Content-Disposition: inline;filename=f.txt
|     Content-Type: application/json
|     Date: Tue, 09 Jan 2024 16:29:53 GMT
|     Connection: close
|     {"timestamp":"2024-01-09T16:29:53.520+00:00","status":404,"error":"Not Found","path":"/nice%20ports%2C/Tri%6Eity.txt%2ebak"}
|   GetRequest:
|     HTTP/1.1 404
|     Vary: Origin
|     Vary: Access-Control-Request-Method
|     Vary: Access-Control-Request-Headers
|     Content-Type: application/json
|     Date: Tue, 09 Jan 2024 16:29:53 GMT
|     Connection: close
|     {"timestamp":"2024-01-09T16:29:53.514+00:00","status":404,"error":"Not Found","path":"/"}
|   HTTPOptions:
|     HTTP/1.1 404
|     Vary: Origin
|     Vary: Access-Control-Request-Method
|     Vary: Access-Control-Request-Headers
|     Content-Type: application/json
|     Date: Tue, 09 Jan 2024 16:29:58 GMT
|     Connection: close
|     {"timestamp":"2024-01-09T16:29:58.529+00:00","status":404,"error":"Not Found","path":"/"}
|   RPCCheck, RTSPRequest:
|     HTTP/1.1 400
|     Content-Type: text/html;charset=utf-8
|     Content-Language: en
|     Content-Length: 435
|     Date: Tue, 09 Jan 2024 16:29:58 GMT
|     Connection: close
|     <!doctype html><html lang="en"><head><title>HTTP Status 400
```

1.4-DAST

Zed Atttack Proxy

OWASP
Zed Attack Proxy

ZAP is an open-source web application security scanner to perform security testing (Dynamic Testing) on web applications. OWASP ZAP is the flagship OWASP project used extensively by penetration testers. ZAP can also run in a daemon mode for hands-off scans for CI/CD pipeline. ZAP provides extensive API (SDK) and a REST API to help users create custom scripts.

Embed Zed Atttack Proxy into Gitlab CI/CD:

```
zap-baseline:
  stage: test
  image: owasp/zap2docker-stable
  when: always
  script:
    - mkdir -p /zap/wrk/
    - zap-full-scan.py -t http://192.168.75.132:8083 -r report.html
    - cp /zap/wrk/report.html .
  artifacts:
    when: always
    paths: [report.html]
```

1.4-DAST

Zed Atttack Proxy

OWASP
Zed Attack Proxy

The result of the test :

```
$ zap-full-scan.py -t http://192.168.75.132:8083 -r report.html
Total of 4 URLs
PASS: Directory Browsing [0]
PASS: Vulnerable JS Library (Powered by Retire.js) [10003]
PASS: In Page Banner Information Leak [10009]
PASS: Cookie No HttpOnly Flag [10010]
PASS: Cookie Without Secure Flag [10011]
PASS: Re-examine Cache-control Directives [10015]
PASS: Cross-Domain JavaScript Source File Inclusion [10017]
PASS: Content-Type Header Missing [10019]
PASS: Anti-clickjacking Header [10020]
PASS: X-Content-Type-Options Header Missing [10021]
PASS: Information Disclosure - Debug Error Messages [10023]
PASS: Information Disclosure - Sensitive Information in URL [10024]
PASS: Information Disclosure - Sensitive Information in HTTP Referrer Header [10025]
PASS: HTTP Parameter Override [10026]
PASS: Information Disclosure - Suspicious Comments [10027]
PASS: Open Redirect [10028]
PASS: Cookie Poisoning [10029]
```

1.4-DAST

Zed Atttack Proxy

OWASP
Zed Attack Proxy

ZAP Scanning Report

Site: http://192.168.75.132:8083

Generated on Tue, 9 Jan 2024 18:15:45

ZAP Version: 2.14.0

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	0
Low	0
Informational	2
False Positives:	0

Alerts

Name	Risk Level	Number of Instances
Storable and Cacheable Content	Informational	4
User Agent Fuzzer	Informational	48

Alert Detail

Informational	Storable and Cacheable Content
Description	The response contents are storable by caching components such as proxy servers, and may be retrieved directly from the cache, rather than from the origin server by the caching servers, in response to similar requests from other users. If the response data is sensitive, personal or user-specific, this may result in sensitive information being leaked. In some cases, this may even result in a user gaining complete control of the session of another user, depending on the configuration of the caching components in use in their environment. This is primarily an issue where "shared" caching servers such as "proxy" caches are configured on the local network. This configuration is typically found in corporate or educational environments, for instance.
URL	http://192.168.75.132:8083
Method	GET
Parameter	
Attack	
Evidence	
Other Info	In the absence of an explicitly specified caching lifetime directive in the response, a liberal lifetime heuristic of 1 year was assumed. This is permitted by rfc7234.
URL	http://192.168.75.132:8083/
Method	GET
Parameter	

1. Backend :
   - 1.1- SAST
   - 1.2- SCA
   - 1.3- CS
   - 1.4- DAST
2. Frontend
   - 2.1- SAST
   - 2.2- SCA
   - 2.3- CS
   - 2.4- DAST

Section 3

2.1-SAST

This time for the frontend we are going to use almost the same tools :

TruffleHog

Bandit

2.1-SAST

TruffleHog

Embed TruffleHog into Gitlab CI/CD:

```
git-secrets:
  stage: build
  script:
    - docker run -v $(pwd):/src --rm hysnsec/trufflehog filesystem --directory=/src --json | tee trufflehog-output.json
    - cat trufflehog-output.json
  artifacts:
    paths: [trufflehog-output.json]
    when: always
    expire_in: one week
```

2.1-SAST

TruffleHog

The result of the test :

```
$ docker run -v $(pwd):/src --rm hysnsec/trufflehog filesystem --directory=/src --json | tee trufflehog-output.json
{"level":"info-0","ts":"2024-01-09T20:06:49Z","logger":"trufflehog","msg":"--directory flag is deprecated, please pass directories as arguments"}
{"level":"info-0","ts":"2024-01-09T20:06:49Z","logger":"trufflehog","msg":"running source","source_manager_worker_id":"XZwq5","with_units":true}
{"level":"info-0","ts":"2024-01-09T20:06:49Z","logger":"trufflehog","msg":"finished scanning","chunks":0,"bytes":0,"verified_secrets":0,"unverified_secret
s":0,"scan_duration":"40.182481ms"}
```

2.1-SAST

Bandit

The Bandit is a tool designed to find common security issues in Python code.
To do this Bandit, processes each file, builds an AST, and runs appropriate plugins against the AST nodes. Once Bandit has finished scanning all the files it generates a report.

Embed Bandit into Gitlab CI/CD:

```
sast-bandit:
  stage: build
  before_script:
    - apk update
    - apk add --no-cache python3 py3-pip
  script:
    - pip3 install bandit==1.7.4 --break-system-packages
    - pip3 install requests --break-system-packages
    - bandit -r . -f json | tee bandit-output.json
    - cat bandit-output.json
    - whoami
    - hostname
  after_script:
    - python3 upload-results.py --host 0.0.0.0:8080 --api_key 3d49f5330984a68a20f4dc75a59bc313ddc3509d --engagement_id 1 --product_id 1 --lead_id 1 --environment "Production" --result_file bandit-output.json --scanner "Bandit Scan"
  artifacts:
    paths: [bandit-output.json]
    when: always
```

2.1-SAST

Bandit

The result of the test :

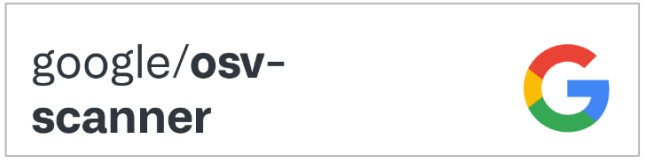
```
$ bandit -r . -f json | tee bandit-output.json
[main]  INFO    profile include tests: None
[main]  INFO    profile exclude tests: None
[main]  INFO    cli include tests: None
[main]  INFO    cli exclude tests: None
{
  "errors": [],
  "generated_at": "2024-01-09T20:08:20Z",
  "metrics": {
    "_totals": {
      "CONFIDENCE.HIGH": 0,
      "CONFIDENCE.LOW": 0,
      "CONFIDENCE.MEDIUM": 0,
      "CONFIDENCE.UNDEFINED": 0,
      "SEVERITY.HIGH": 0,
      "SEVERITY.LOW": 0,
      "SEVERITY.MEDIUM": 0,
      "SEVERITY.UNDEFINED": 0,
      "loc": 0,
      "nosec": 0,
      "skipped_tests": 0
    }
  },
  "results": []
```

2.2-SCA

Same as the backend we are going to use the same tools with another new tool :

retireJS

Bandit



osv-scanner

2.2-SAST

retireJS

Embed retireJS into Gitlab CI/CD:

```
oast-RetireJS:
  stage: build
  image: node:alpine3.10
  script:
    - npm install
    - npm install -g retire # Install retirejs npm package.
    - retire --outputformat json --outputpath retirejs-report.json --severity high
    - cat retirejs-report.json
  artifacts:
    paths: [retirejs-report.json]
    when: always
```

2.2-SAST

retireJS

The result of the test :

```
$ cat retirejs-report.json
{"version":"4.3.4","start":"2024-01-09T20:00:43.360Z","data":[],"messages":[],"errors":[],"time":40.595}
```

2.2-SAST

snyk

Embed snyk into Gitlab CI/CD:

```
oast-snyk:
  stage: build
  image: node:alpine3.10
  before_script:
    - wget -O snyk https://github.com/snyk/cli/releases/download/v1.1156.0/snyk-alpine
    - chmod +x snyk
    - mv snyk /usr/local/bin/
  script:
    - npm install
    - snyk auth 35589080-01a0-4492-af3d-82ee5f1140d0
    - snyk test --json > snyk-results.json
    - cat snyk-results.json
  artifacts:
    paths: [snyk-results.json]
    when: always
    expire_in: one week
```

2.2-SAST

snyk

The result of the test :

```
$ cat snyk-results.json
{
  "vulnerabilities": [],
  "ok": true,
  "dependencyCount": 12,
  "org": "thehuntison000",
  "policy": "# Snyk (https://snyk.io) policy file, patches or ignores known vulnerabilities.\nversion: v1.25.1\nignore: {}\npatch: {}\n",
  "isPrivate": true,
  "licensesPolicy": {
    "severities": {},
    "orgLicenseRules": {
      "AGPL-1.0": {
        "licenseType": "AGPL-1.0",
        "severity": "high",
        "instructions": ""
      },
      "AGPL-3.0": {
        "licenseType": "AGPL-3.0",
        "severity": "high",
        "instructions": ""
      },
      "Artistic-1.0": {
        "licenseType": "Artistic-1.0",
        "severity": "medium",
        "instructions": ""
      },
      "Artistic-2.0": {
        "licenseType": "Artistic-2.0",
        "severity": "medium",
        "instructions": ""
      },
```

2.2-SAST

osv-scanner

google/**osv-scanner**

Embed osv-scanner into Gitlab CI/CD:

```
osv-scanner:
  stage: pre-build
  image: golang:1.21.1-alpine3.18
  before_script:
    - apk add npm
    - npm install
  script:
    - go install github.com/google/osv-scanner/cmd/osv-scanner@v1
    - osv-scanner .
```

2.2-SAST

google/**osv-scanner**

osv-scanner

The result of the test :

```
$ osv-scanner .
Scanning dir .
Scanning /builds/mouad.tigmouti00/devsecops-frontend/ at commit 4d7b8e588e53e4abae976c8c788ce813a0baace3
Scanned /builds/mouad.tigmouti00/devsecops-frontend/package-lock.json file and found 847 packages
+-------------------------------------+------+-----------+------------------+---------+-------------------+
| OSV URL                             | CVSS | ECOSYSTEM | PACKAGE          | VERSION | SOURCE            |
+-------------------------------------+------+-----------+------------------+---------+-------------------+
| https://osv.dev/GHSA-67hx-6x53-jw92 | 9.3  | npm       | @babel/traverse  | 7.21.5  | package-lock.json |
| https://osv.dev/GHSA-jchw-25xp-jwwc | 6.1  | npm       | follow-redirects | 1.15.2  | package-lock.json |
| https://osv.dev/GHSA-7fh5-64p2-3v2j | 5.3  | npm       | postcss          | 8.4.23  | package-lock.json |
| https://osv.dev/GHSA-c2qf-rxjj-qqgw | 5.3  | npm       | semver           | 5.7.1   | package-lock.json |
| https://osv.dev/GHSA-c2qf-rxjj-qqgw | 5.3  | npm       | semver           | 6.3.0   | package-lock.json |
| https://osv.dev/GHSA-c2qf-rxjj-qqgw | 5.3  | npm       | semver           | 7.4.0   | package-lock.json |
| https://osv.dev/GHSA-353f-5xf4-qw67 | 7.5  | npm       | vite             | 4.3.1   | package-lock.json |
+-------------------------------------+------+-----------+------------------+---------+-------------------+
```

2.3-Container Security

Container Scanning is often considered part of Software Composition Analysis (SCA). SCA can contain aspects of inspecting the items your code uses. These items typically include application and system dependencies that are almost always imported from external sources, rather than sourced from items you wrote yourself.

```
#Container Scanning

include:
  - template: Security/Container-Scanning.gitlab-ci.yml

container_scanning:
  variables:
    CS_IMAGE: mouadtigmouti/devsecops-frontend
```

2.3-Container Security

The result of container security scan :

```
$ gtcs scan
[INFO] [2024-01-09 20:49:14 +0000] [container-scanning]  >  Remediation is disabled; /builds/mouad.tigmouti00/devsecops-frontend/Dockerfile cannot be fo
d. Have you set `GIT_STRATEGY` and
`CS_DOCKERFILE_PATH`?
See https://docs.gitlab.com/ee/user/application_security/container_scanning/#solutions-for-vulnerabilities-auto-remediation
[INFO] [2024-01-09 20:49:28 +0000] [container-scanning]  >  Scanning container from registry [MASKED]/devsecops-backend for vulnerabilities with severit
evel UNKNOWN or higher, with gcs 6.6.2 and Trivy Version: 0.48.1, advisories updated at 2024-01-09T12:13:28+00:00
+------------+--------------+-----------------+------------------------+-----------------------------------------------------------------------------+
|   STATUS   | CVE SEVERITY |   PACKAGE NAME  |    PACKAGE VERSION     |                                CVE DESCRIPTION                              |
+------------+--------------+-----------------+------------------------+-----------------------------------------------------------------------------+
| Unapproved |     Low      |       apt       |         2.2.4          | It was found that apt-key in apt, all versions, do not correctly valid      |
|            |              |                 |                        | ate gpg keys with the master keyring, leading to a potential man-in-th      |
|            |              |                 |                        |                              e-middle attack.                               |
+------------+--------------+-----------------+------------------------+-----------------------------------------------------------------------------+
| Unapproved |     High     |       bash      |        5.1-2+b3        | A flaw was found in the bash package, where a heap-buffer overflow can      |
|            |              |                 |                        |  occur in valid parameter_transform. This issue may lead to memory pro      |
|            |              |                 |                        |                                    blems.                                   |
+------------+--------------+-----------------+------------------------+-----------------------------------------------------------------------------+
| Unapproved |     Low      |       bash      |        5.1-2+b3        |                          TEMP-0841856-B18BAF                                |
+------------+--------------+-----------------+------------------------+-----------------------------------------------------------------------------+
| Unapproved |     Low      |     bsdutils    |   1:2.36.1-8+deb11u1   | A flaw was found in the util-linux chfn and chsh utilities when compil      |
|            |              |                 |                        | ed with Readline support. The Readline library uses an "INPUTRC" envir      |
|            |              |                 |                        | onment variable to get a path to the library config file. When the lib      |
|            |              |                 |                        | rary cannot parse the specified file, it prints an error message conta      |
|            |              |                 |                        | ining data from the file. This flaw allows an unprivileged user to rea      |
|            |              |                 |                        | d root-owned files, potentially leading to privilege escalation. This       |
|            |              |                 |                        |          flaw affects util-linux versions prior to 2.37.4.                  |
+------------+--------------+-----------------+------------------------+-----------------------------------------------------------------------------+
| Unapproved |     Low      |    coreutils    |        8.32-4+b1       | chroot in GNU coreutils, when used with --userspec, allows local users      |
|            |              |                 |                        |  to escape to the parent session via a crafted TIOCSTI ioctl call, whi      |
|            |              |                 |                        |         ch pushes characters to the terminal's input buffer.                |
+------------+--------------+-----------------+------------------------+-----------------------------------------------------------------------------+
| Unapproved |     Low      |    coreutils    |        8.32-4+b1       | In GNU Coreutils through 8.29, chown-core.c in chown and chgrp does no      |
```

2.4-DAST

Exactly like the backend we are going to use the exact same tools:

Nikto

NMAP

Zed Atttack Proxy

2.4-DAST

Nikto

Embed Nikto into Gitlab CI/CD:

```
nikto:
  stage: test
  script:
    - docker pull hysnsec/nikto
    - docker run --rm -v $(pwd):/tmp hysnsec/nikto -h http://192.168.75.132:8080 -o nikto-output.xml
    - cat nikto-output.xml
  artifacts:
    paths: [nikto-output.xml]
    when: always
```

2.4-DAST

Nikto

The result of the test :

```
$ docker run --rm -v $(pwd):/tmp hysnsec/nikto -h http://192.168.75.132:8081
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.75.132
+ Target Hostname:    192.168.75.132
+ Target Port:        8081
+ Start Time:         2024-01-09 20:55:42 (GMT0)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ /: Retrieved x-powered-by header: Express.
+ /: Retrieved access-control-allow-origin header: *.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME typ
e. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 8102 requests: 0 error(s) and 4 item(s) reported on remote host
+ End Time:           2024-01-09 20:58:10 (GMT0) (148 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

2.4-DAST

NMAP

Embed NMAP into Gitlab CI/CD:

```
nmap:
  stage: test
  script:
    - docker pull hysnsec/nmap
    - docker run --rm -v $(pwd):/tmp hysnsec/nmap -p 8080 192.168.75.132 -oX nmap-output.xml
    - nmap-output.xml
  artifacts:
    paths: [nmap-output.xml]
    when: always
```

2.4-DAST

NMAP

The result of the test :

```
$ docker run --rm -v $(pwd):/tmp hysnsec/nmap -sSVC -A -p 8081 192.168.75.132
Starting Nmap 7.94 ( https://nmap.org ) at 2024-01-09 20:59 UTC
Nmap scan report for 192.168.75.132
Host is up (0.00023s latency).
PORT     STATE SERVICE VERSION
8081/tcp open  http    Node.js (Express middleware)
|_http-title: ICCN-Project
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
TRACEROUTE (using port 8081/tcp)
HOP RTT     ADDRESS
1   0.21 ms 192.168.75.132
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.27 seconds
```

2.4-DAST

Zed Atttack Proxy

Embed Zed Atttack Proxy into Gitlab CI/CD:

```
zap-baseline:
  stage: test
  script:
    - docker run --user $(id -u):$(id -g) -w /zap -v $(pwd):/zap/wrk:rw --rm softwaresecurityproject/zap-stable:2.13.0 zap-baseline.py -t http://192.168.75.132:8080 -J zap-output.json
    - cat zap-output.json
  artifacts:
    paths: [zap-output.json]
    when: always
```

2.4-DAST

Zed Atttack Proxy

OWASP
Zed Attack Proxy

The result of the test :

```
$ zap-full-scan.py -t http://192.168.75.132:8081 -r report.html
Total of 15 URLs
PASS: Vulnerable JS Library (Powered by Retire.js) [10003]
PASS: In Page Banner Information Leak [10009]
PASS: Cookie No HttpOnly Flag [10010]
PASS: Cookie Without Secure Flag [10011]
PASS: Re-examine Cache-control Directives [10015]
PASS: Cross-Domain JavaScript Source File Inclusion [10017]
PASS: Content-Type Header Missing [10019]
PASS: Information Disclosure - Debug Error Messages [10023]
PASS: Information Disclosure - Sensitive Information in URL [10024]
PASS: Information Disclosure - Sensitive Information in HTTP Referrer Header [10025]
PASS: HTTP Parameter Override [10026]
PASS: Information Disclosure - Suspicious Comments [10027]
```

1. Creation of kubernetes cluster
2. Creation of namespace
3. Creation of necessary YAML files
4. Execution of YAML files

Section 4

1-Creation of Kubernetes cluster

For this step we are going to use Minikube in order to install a local cluster in our machine.

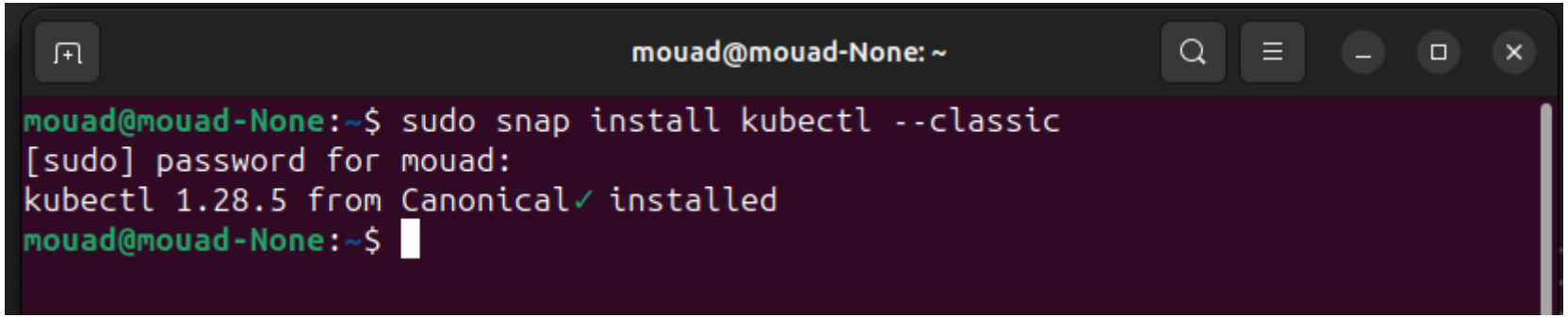
minikube

Minikube is an open-source tool that allows developers to run a local Kubernetes cluster on their machine. It provides a lightweight and portable way to experiment with Kubernetes features and deploy applications locally. Minikube sets up a virtual machine and replicates the core components of a full-scale Kubernetes cluster.

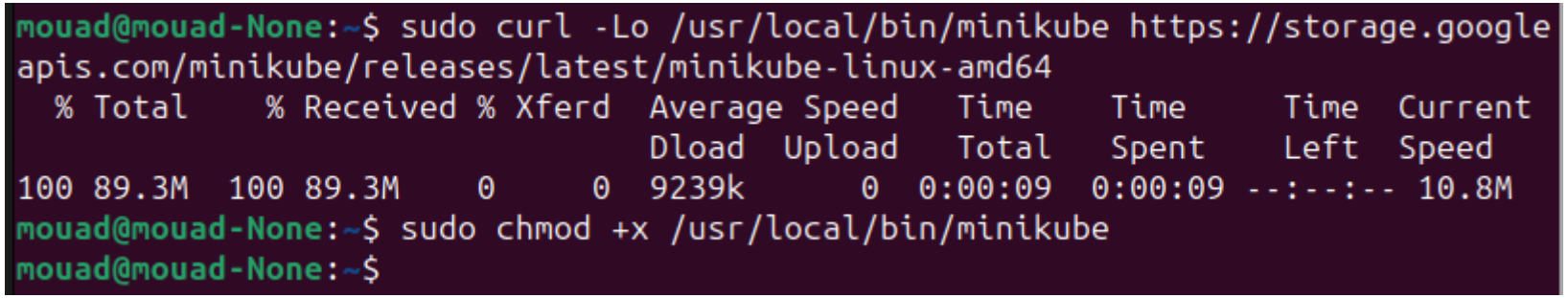
1-Creation of Kubernetes cluster

Let's install minikube in our local Ubuntu vm where we did the pipeline section :

First we will install kubectl, Using the command : sudo snap install kubectl --classic



Now, let's install Minikube using the commands :
sudo curl -Lo /usr/local/bin/minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo chmod +x /usr/local/bin/minikube



1-Creation of Kubernetes cluster

Let's start Minikube and test it :

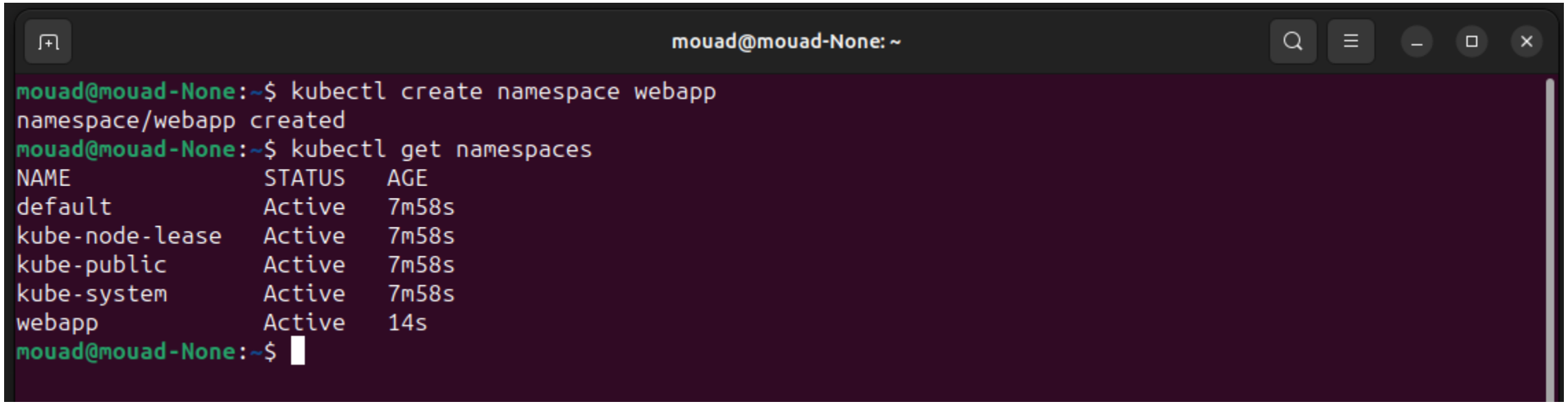
```
mouad@mouad-None:~$ minikube start
W0110 18:35:44.516265    4723 main.go:291] Unable to resolve the current Docker CLI context "default": context
"default": context not found: open /home/mouad/.docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c495
8ba141d5f4133a33f0688f/meta.json: no such file or directory
😄  minikube v1.32.0 on Ubuntu 23.10
✨  Automatically selected the docker driver. Other choices: qemu2, none, ssh
📌  Using Docker driver with root privileges
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
💾  Downloading Kubernetes v1.28.3 preload ...
    > preloaded-images-k8s-v18-v1...:  403.35 MiB / 403.35 MiB  100.00% 6.22 Mi
    > gcr.io/k8s-minikube/kicbase...:  453.90 MiB / 453.90 MiB  100.00% 4.15 Mi
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
mouad@mouad-None:~$ minikube status
W0110 18:41:10.595397    9056 main.go:291] Unable to resolve the current Docker CLI context "default": context
"default": context not found: open /home/mouad/.docker/contexts/meta/37a8eec1ce19687d132fe29051dca629d164e2c495
8ba141d5f4133a33f0688f/meta.json: no such file or directory
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

mouad@mouad-None:~$
```

1. Creation of kubernetes cluster
2. Creation of namespace
3. Creation of necessary YAML files
4. Execution of YAML files

Section 4

2-Creation of namespace

At this stage we are going to create a namespace called ‘webapp’
For that we are going to execute this command : kubectl create namespace webapp
And to verify its status : Kubectl get namespaces



1. Creation of kubernetes cluster
2. Creation of namespace
3. Creation of necessary YAML files
4. Execution of YAML files

Section 4

2-Creation necessary YAML files

For this step we are going to create 3 YAML files one file for each of the MySQL, the frontend and the backend

mysql :

mysql.yaml | backend.yaml | frontend.yaml

mysql.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  namespace: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:latest
        ports:
        - containerPort: 3306
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
  namespace: webapp
spec:
  selector:
    app: mysql
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
  type: ClusterIP
```

2-Creation necessary YAML files

Frontend:

mysql.yaml | backend.yaml | frontend.yaml

frontend.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: devsecops-frontend-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: devsecops-frontend
  template:
    metadata:
      labels:
        app: devsecops-frontend
    spec:
      containers:
      - name: devsecops-frontend
        image: mouadtigmouti/devsecops-frontend
        imagePullPolicy: Always
        ports:
          - name: http
            containerPort: 8081

---
apiVersion: v1
kind: Service
metadata:
  name: devsecops-frontend-service
spec:
  type: ClusterIP
  selector:
    app: devsecops-frontend
  ports:
  - name: http
    port: 8081
    targetPort: 8081
```

2-Creation necessary YAML files

Backend :

mysql.yaml | backend.yaml | frontend.yaml

backend.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: devsecops-backend-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: devsecops-backend
  template:
    metadata:
      labels:
        app: devsecops-backend
    spec:
      containers:
        - name: devsecops-backend
          image: mouadtigmouti/devsecops-
          imagePullPolicy: Always
          ports:
            - containerPort: 8080
```

```yaml

---
apiVersion: v1
kind: Service
metadata:
  name: devsecops-backend-service
  labels:
    app: devsecops-backend
spec:
  type: ClusterIP
  selector:
    app: devsecops-backend
  ports:
  - name: http
    port: 8080
    targetPort: 8080

```

2-Creation necessary YAML files

We need to adjust the source code of our app to match these changes :

For the backend source code :

TutorialController.java | application.properties | .gitlab-ci.yml

src > main > java > com > bezkoder > spring > datajpa > controller > TutorialController.java

```java
package com.bezkoder.spring.datajpa.controller;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.bezkoder.spring.datajpa.model.Tutorial;
import com.bezkoder.spring.datajpa.repository.TutorialRepository;

@CrossOrigin(origins = "http://devsecops-frontend-service:8081")
@RestController
@RequestMapping("/api")
public class TutorialController {
```

TutorialController.java | application.properties | .gitlab-ci.yml

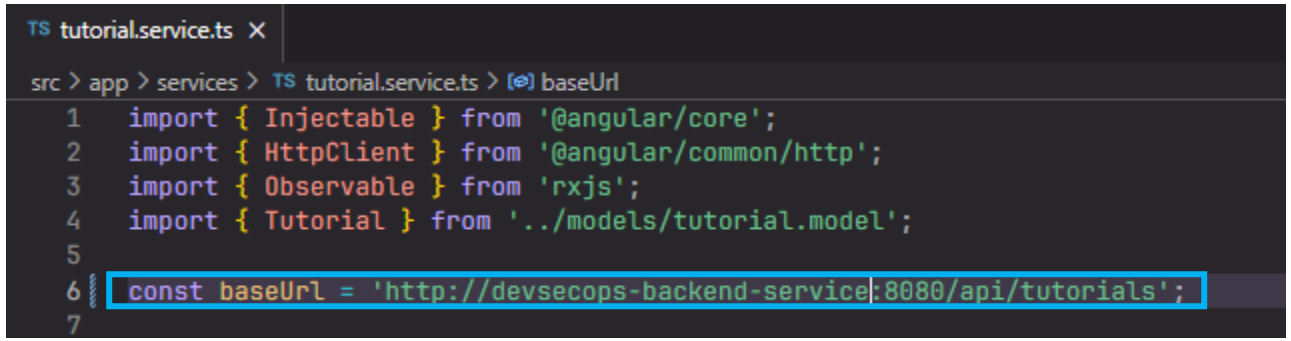
src > main > resources > application.properties

```
spring.datasource.url= jdbc:mysql://mysql-service:3306/testdb?useSSL=false&allowPublicKeyRetrieval=true
spring.datasource.username= root
spring.datasource.password= ICCN2023

spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto= update
```

We need to adjust the address of the frontend by changing it to the name of the frontend service. The same change should be applied to the database; the IP address should be replaced with the name of the MySQL service.

2-Creation necessary YAML files

For the frontend source code :



We need to adjust the address of the backend by changing it to the name of the backend service.

Now we just need to run them and see the result

1. Creation of kubernetes cluster
2. Creation of namespace
3. Creation of necessary YAML files
4. Execution of YAML files

Section 4

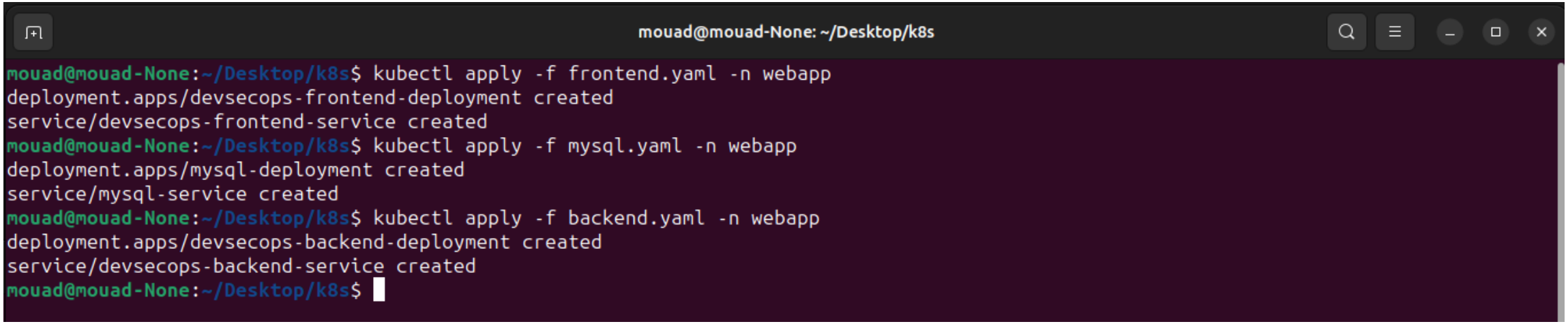
2-Creation necessary YAML files

To run the yaml files we simply have to run :

Kubectl apply –f mysql.yaml –n webapp

Kubectl apply –f frontend.yaml –n webapp

Kubectl apply –f mysql.yaml –n webapp



2-Creation necessary YAML files

Let’s check their state :


```
mouad@mouad-None:~/Desktop/k8s$ kubectl get pods -n webapp
NAME                                        READY   STATUS         RESTARTS      AGE
devsecops-backend-deployment-85775b8f4-th2qt     0/1     Error          1 (20s ago)   55s
devsecops-frontend-deployment-57c7cfb7d5-x2r6z   1/1     Running        0             70s
mysql-deployment-7987cdd5fc-vxjkg                0/1     ErrImagePull   0             61s
mouad@mouad-None:~/Desktop/k8s$
```

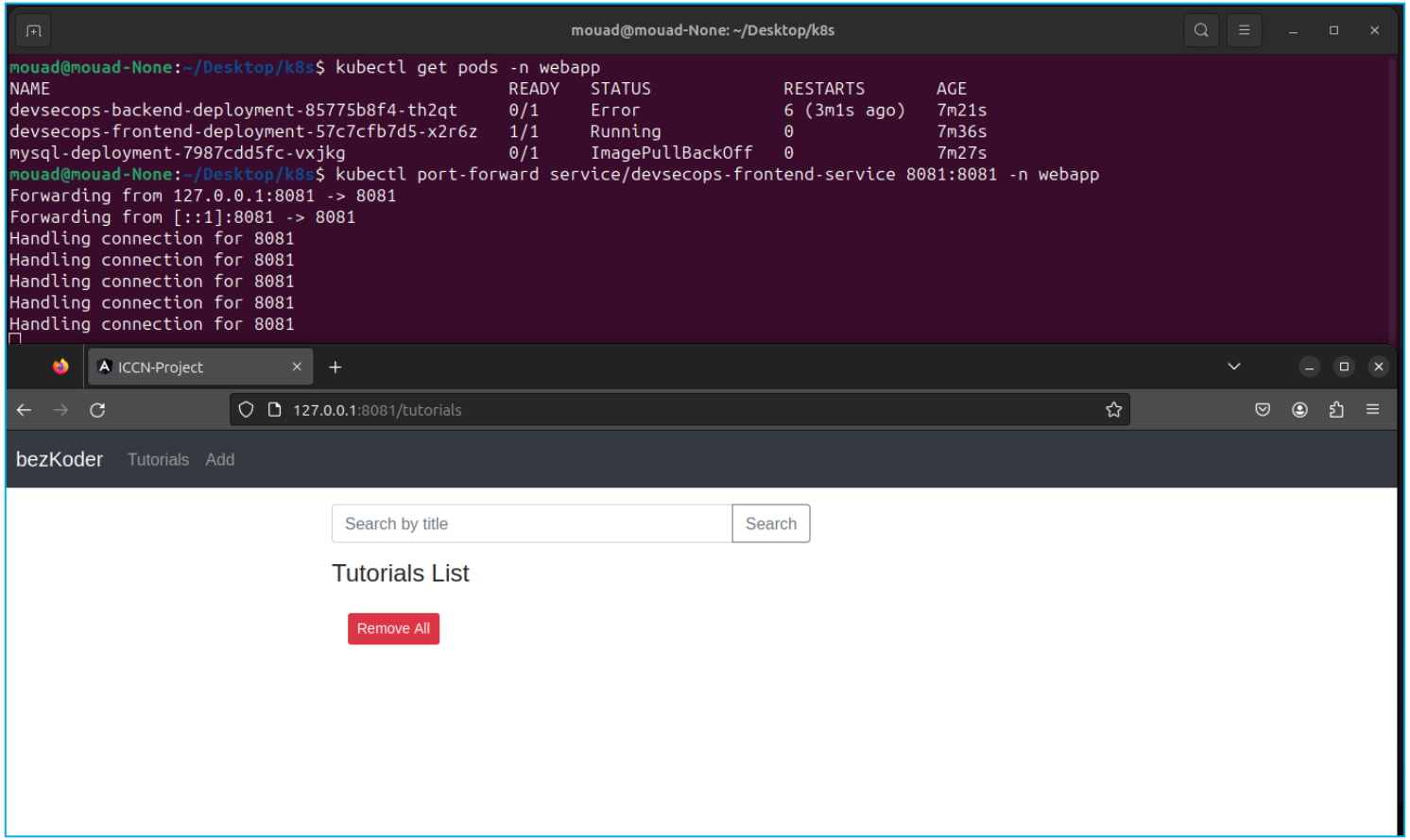
Unfortunately there is something wrong.
When I checked the logs, I found out that I reached the limit number of pulling the MySQL image, and that is why I cant start the MySQL pod which causes a problem for the backend because there is no database to connect to,

```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  3m21s                default-scheduler  Successfully assigned webapp/mysql-deployment-7987cdd5fc-vxjkg to miniku
be
  Normal   Pulling    113s (x4 over 3m20s)  kubelet           Pulling image "mysql:latest"
  Warning  Failed     111s (x4 over 3m18s)  kubelet           Failed to pull image "mysql:latest": Error response from daemon: toomany
requests: You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/inc
rease-rate-limit
  Warning  Failed     111s (x4 over 3m18s)  kubelet           Error: ErrImagePull
  Warning  Failed     87s (x6 over 3m17s)   kubelet           Error: ImagePullBackOff
  Normal   BackOff    73s (x7 over 3m17s)   kubelet           Back-off pulling image "mysql:latest"
```

2-Creation necessary YAML files

Let’s run the only pod we have since it has no problems, and :

it is working



Bonus section

Monitoring

For this section, we are going to monitor our environment using Prometheus and Grafana.

Prometheus is an open-source monitoring and alerting toolkit designed for reliability and scalability. It collects metrics from configured targets, stores them, and allows querying and alerting on this data.

Grafana complements Prometheus by providing visualization capabilities, allowing users to create interactive and customizable dashboards for monitoring metrics.

Monitoring

Before we start installation let's clean minikubes state and check if there is any running pods :

```
mouad@mouad-None: ~/Desktop/k8s
mouad@mouad-None:~/Desktop/k8s$ kubectl get pod
No resources found in default namespace.
mouad@mouad-None:~/Desktop/k8s$
```

Then we need to install helm :

```
mouad@mouad-None:~/Desktop/k8s$ sudo snap install helm --classic
helm 3.10.1 from Snapcrafters✪ installed
mouad@mouad-None:~/Desktop/k8s$
```

Now let's install Prometheus, to do so we need to add the repo with helm first :

```
mouad@mouad-None:~/Desktop/k8s$ sudo helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
mouad@mouad-None:~/Desktop/k8s$
```

Now let's add the official Helm "stable" chart repository

```
mouad@mouad-None:~/Desktop/k8s$ helm repo add stable https://charts.helm.sh/stable
"stable" has been added to your repositories
```

Monitoring

Let's update the repo

```
mouad@mouad-None:~$ helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
Update Complete. ⎈Happy Helming!⎈
mouad@mouad-None:~$
```

One more repo to update :

```
mouad@mouad-None:~$ helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
"prometheus-community" has been added to your repositories
```

Now let's install Prometheus :

```
mouad@mouad-None:~$ helm install prometheus prometheus-community/kube-prometheus-stack
NAME: prometheus
LAST DEPLOYED: Thu Jan 11 01:58:55 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace default get pods -l "release=prometheus"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
mouad@mouad-None:~$
```

With one command we were able to create all theses components :

```
mouad@mouad-None:~$ kubectl get pod
NAME                                                     READY   STATUS            RESTARTS   AGE
alertmanager-prometheus-kube-prometheus-alertmanager-0   0/2     PodInitializing   0          92s
prometheus-grafana-57cc5d6996-99j5l                      3/3     Running           0          2m17s
prometheus-kube-prometheus-operator-7bf77fdffc-tnq42     1/1     Running           0          2m17s
prometheus-kube-state-metrics-77f874cf44-tlch7           1/1     Running           0          2m17s
prometheus-prometheus-kube-prometheus-prometheus-0       0/2     PodInitializing   0          91s
prometheus-prometheus-node-exporter-xfntx                1/1     Running           0          2m17s
mouad@mouad-None:~$
```

Monitoring

Let's take a look at all the parts that were created :

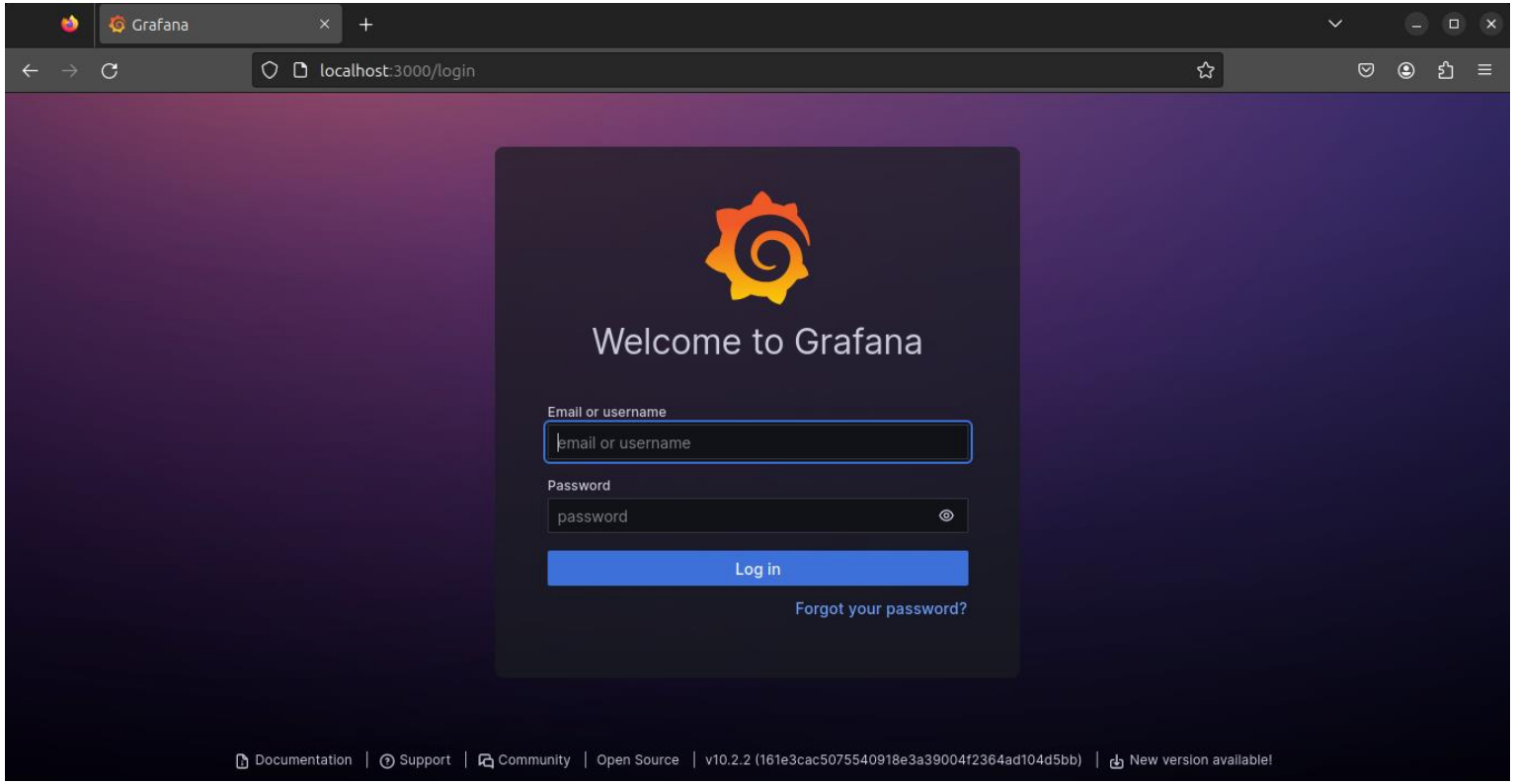
```
mouad@mouad-None:~$ kubectl get all
NAME                                                     READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0          4m32s
pod/prometheus-grafana-57cc5d6996-99j5l                  3/3     Running   0          5m17s
pod/prometheus-kube-prometheus-operator-7bf77fdffc-tnq42 1/1     Running   0          5m17s
pod/prometheus-kube-state-metrics-77f874cf44-tlch7       1/1     Running   0          5m17s
pod/prometheus-prometheus-kube-prometheus-prometheus-0   2/2     Running   0          4m31s
pod/prometheus-prometheus-node-exporter-xfntx            1/1     Running   0          5m17s

NAME                                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
service/alertmanager-operated                     ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP   4m32s
service/kubernetes                                ClusterIP   10.96.0.1        <none>        443/TCP                      7h23m
service/prometheus-grafana                        ClusterIP   10.98.117.229    <none>        80/TCP                       5m17s
service/prometheus-kube-prometheus-alertmanager   ClusterIP   10.100.230.152   <none>        9093/TCP,8080/TCP            5m17s
service/prometheus-kube-prometheus-operator       ClusterIP   10.102.66.123    <none>        443/TCP                      5m17s
service/prometheus-kube-prometheus-prometheus     ClusterIP   10.109.89.245    <none>        9090/TCP,8080/TCP            5m17s
service/prometheus-kube-state-metrics             ClusterIP   10.101.162.36    <none>        8080/TCP                     5m17s
service/prometheus-operated                       ClusterIP   None             <none>        9090/TCP                     4m31s
service/prometheus-prometheus-node-exporter       ClusterIP   10.103.85.212    <none>        9100/TCP                     5m17s

NAME                                                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/prometheus-prometheus-node-exporter   1         1         1       1            1           kubernetes.io/os=linux   5m17s

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/prometheus-grafana            1/1     1            1           5m17s
deployment.apps/prometheus-kube-prometheus-operator   1/1     1            1           5m17s
deployment.apps/prometheus-kube-state-metrics         1/1     1            1           5m17s

NAME                                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/prometheus-grafana-57cc5d6996           1         1         1       5m17s
replicaset.apps/prometheus-kube-prometheus-operator-7bf77fdffc   1         1         1       5m17s
replicaset.apps/prometheus-kube-state-metrics-77f874cf44         1         1         1       5m17s

NAME                                                               READY   AGE
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager   1/1     4m32s
statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus       1/1     4m31s
```

To start Grafana we are going to use port forwarding to expose it

Monitoring

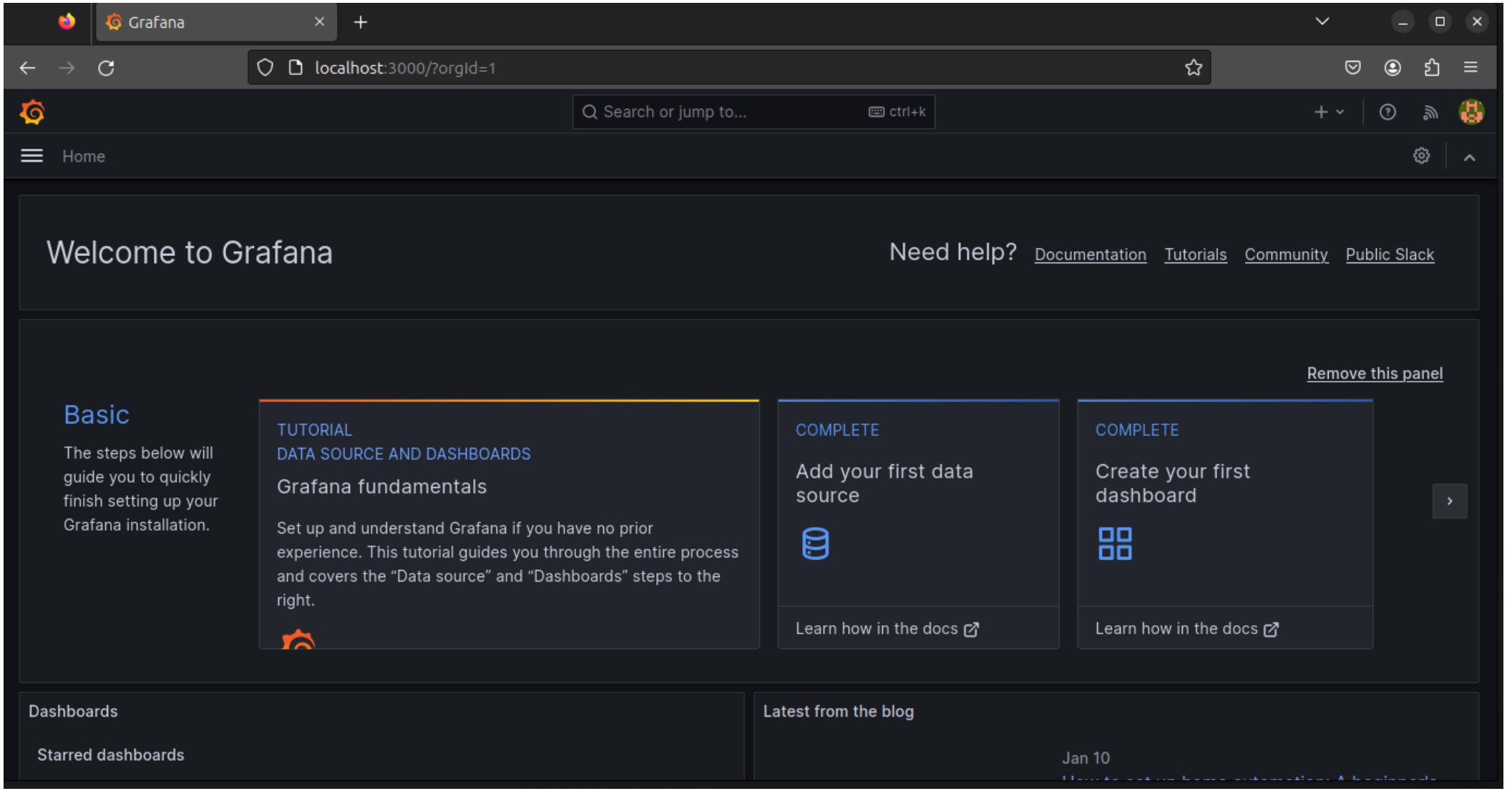
Let's run Grafana using port forwarding :

```
mouad@mouad-None: ~
mouad@mouad-None:~$ kubectl port-forward deployment/prometheus-grafana 3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
Handling connection for 3000
```



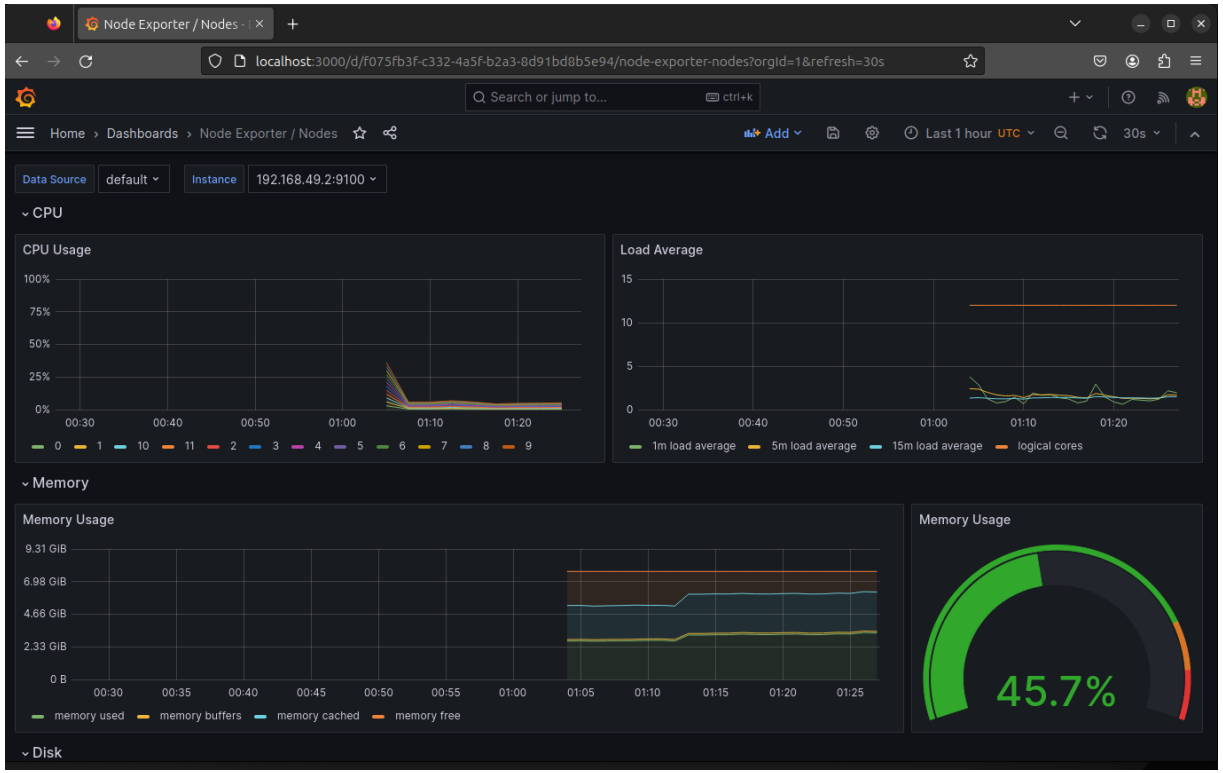
Monitoring

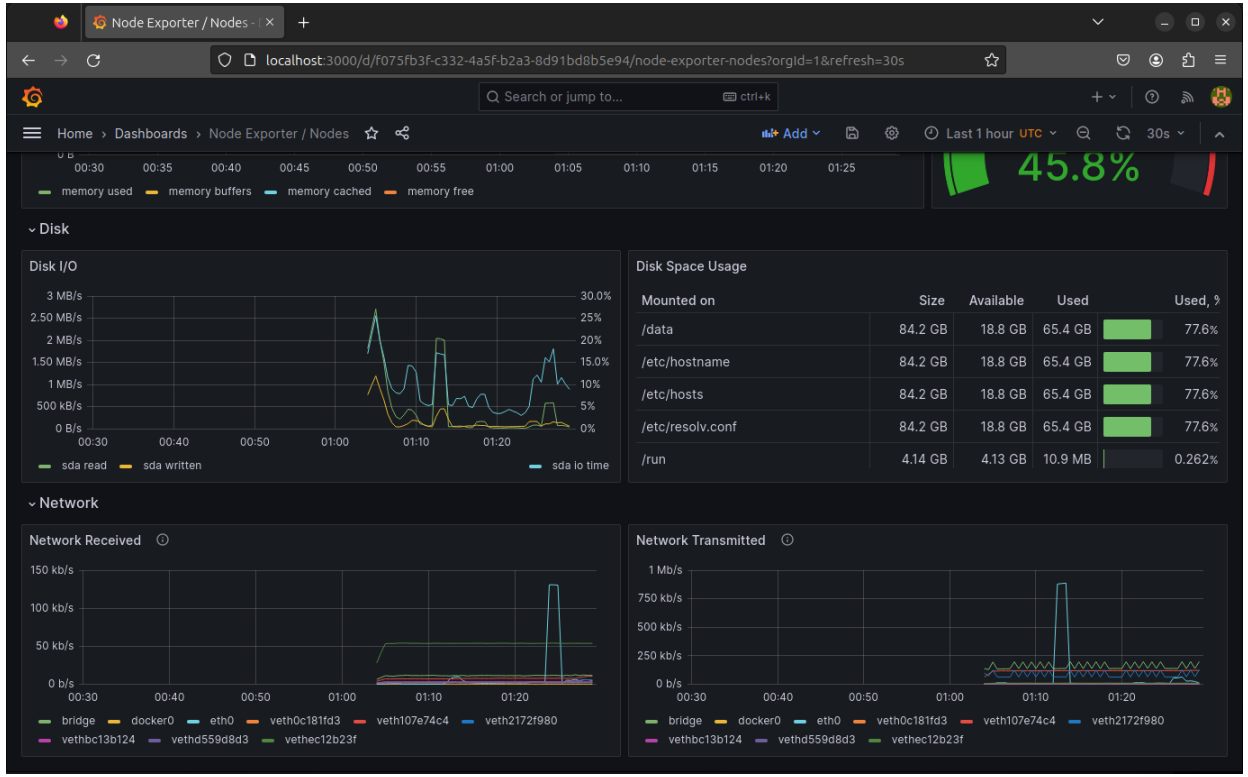
With the credentials admin and the password "prom-operator"



Monitoring

This dashboard in Grafana typically provides insights and metrics related to the nodes in a Kubernetes cluster. It allows us to explore and monitor various aspects of your cluster nodes





Our GitLab repo :

https://gitlab.com/mouad.tigmouti00/devsecops-backend
https://gitlab.com/mouad.tigmouti00/devsecops-frontend
https://gitlab.com/mouad.tigmouti00/k8s

Our Docker repo :

https://hub.docker.com/r/mouadtigmouti/devsecops-frontend

https://hub.docker.com/r/mouadtigmouti/devsecops-backend

Security tests report :

https://github.com/TMouad101/Files/blob/main/scans%20report.pdf

Our app :

http://34.163.16.217:8081/tutorials

Please use your 4G internet