

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)				
<input type="checkbox"/> 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
<input type="checkbox"/> SCRUM-29	As a CI/CD manager, I want to improve control over pipeline execution using rules.	À FAIRE ▾	5	M
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> SCRUM-34	As a user, I want the application to scale seamlessly based on demand	À FAIRE ▾	5	KC
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input type="checkbox"/> 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
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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
<input checked="" type="checkbox"/> 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).

Sprint 1 7 janv. – 14 janv. (8 tickets)		0	0	0	Démarrer un sprint	...
Définition de la Topologie d'Équipe. Démonstration du Framework SCRUM .						
<input checked="" type="checkbox"/>	SCRUM-147 Propose an Optimal Team Topology.	À FAIRE	-	IK		
<input checked="" type="checkbox"/>	SCRUM-152 Justify the chosen team topology, considering factors like communication, efficiency, and expertise.	À FAIRE	-	IK		
<input checked="" type="checkbox"/>	SCRUM-153 Identify and assign roles (Product Owner, Scrum Master, Development Team).	À FAIRE	-	IK		
<input checked="" type="checkbox"/>	SCRUM-154 Outline the key events (rituals) such as Sprint Planning, Daily Stand-ups, Sprint Review, and Sprint Retrospective.	À FAIRE	-	IK		
<input checked="" type="checkbox"/>	SCRUM-155 Establish a Definition of Done (DoD) to ensure the quality of deliverables.	À FAIRE	-	IK		
<input checked="" type="checkbox"/>	SCRUM-156 Define the Product Backlog, including prioritized features and tasks.	À FAIRE	-	IK		
<input checked="" type="checkbox"/>	SCRUM-157 Break down the Product Backlog into Sprints, specifying User Stories, Story Points, and Definition of Done for each Sprint.	À FAIRE	-	IK		
<input checked="" type="checkbox"/>	SCRUM-158 Develop a Product Increment at the end of each Sprint, ensuring it meets the Definition of Done.	À FAIRE	-	IK		

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.

Sprint 2 14 janv. – 21 janv. (34 tickets)				
Facilitate seamless integration between the frontend and backend while implementing CI/CD for the frontend.				
<input checked="" type="checkbox"/> SCRUM-55	As a developer, I want to establish a CI/CD pipeline using GitLab CI (or GitHub Actions) for the backend of the application.	À FAIRE	3	M
<input checked="" type="checkbox"/> SCRUM-56	Set up a basic CI configuration file for the backend.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-57	Configure build, packaging, and deployment stages.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-58	Ensure that the pipeline runs on every push to the backend repository.	À FAIRE	-	M
<input type="checkbox"/> SCRUM-59	As a developer, I want the backend CI pipeline to automate the build, packaging, and deployment processes in an environment running containers.	À FAIRE	5	M
<input checked="" type="checkbox"/> SCRUM-60	Utilize Docker for containerization in the backend pipeline.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-61	Automate the packaging process within the pipeline.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-126	Implement deployment steps targeting a containerized environment.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-62	As a developer, I want the MySQL database to be containerized and executed within the same environment as the backend.	À FAIRE	3	M
<input checked="" type="checkbox"/> SCRUM-63	Containerize the MySQL database.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-64	Integrate MySQL container into the backend CI/CD pipeline environment.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-65	As a developer, I want to set up a CI/CD pipeline using GitLab CI (or GitHub Actions) for the frontend of the application.	À FAIRE	3	M
<input checked="" type="checkbox"/> SCRUM-66	Establish a basic CI configuration file for the frontend.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-67	Configure build, packaging, and deployment stages.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-127	Ensure that the pipeline runs on every push to the frontend repository.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-128	As a developer, I want the frontend CI pipeline to automate the build, packaging, and deployment processes in an environment running containers.	À FAIRE	5	M
<input checked="" type="checkbox"/> SCRUM-129	Utilize Docker for containerization in the frontend pipeline.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-130	Automate the packaging process within the pipeline.	À FAIRE	-	M
<input checked="" type="checkbox"/> SCRUM-131	Implement deployment steps targeting a containerized environment.	À FAIRE	-	M

3. Project deliverable:



Sprints Backlogs(Tigmouti Mouad):

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

Sprint 3: 21 janv. – 28 janv. (12 tickets)			
Implement essential user features, enabling tutorial creation, viewing, updating, and deletion.			
<input checked="" type="checkbox"/> SCRUM-81	As a developer, I want to add 'Pre-Commit Hooks' to ensure code security hygiene in the version control system (VCS).	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-82	Implement and configure pre-commit hooks to analyze and check code before committing to the version control system.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-83	Test Pre-Commit Hooks	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-84	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 1
<input checked="" type="checkbox"/> SCRUM-85	Integrate automated tests into the CI/CD pipeline.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-86	Integrate security scanning tools (SAST, DAST, SCA) into the CI/CD pipeline and configure security scans at appropriate stages in the pipeline.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-104	Optimize the CI/CD pipeline for efficiency and speed and implement parallelization and caching strategies.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-125	Visualize the results obtained in a Vulnerability Management Tool.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-87	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 logs.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-88	Choose appropriate monitoring tools for infrastructure, MySQL, backend, and frontend.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-89	Set up and configure monitoring tools to collect relevant metrics.	À FAIRE	8 1
<input checked="" type="checkbox"/> SCRUM-105	Create comprehensive dashboards for key metrics. Include dashboards for infrastructure health, database performance, and application logs.	À FAIRE	8 1

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.

Sprint 4 28 janv. – 4 févr. (20 tickets)		
Enhance user experience with search functionality and real-time result updates, along with informative notifications.		
<input type="checkbox"/> SCRUM-93 As a developer, I want to create a local Kubernetes cluster using Minikube to facilitate development and testing.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-94 Install Minikube on the development machine.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-95 Configure Minikube to create a local Kubernetes cluster.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-96 As a developer, I want to create a namespace named "webapp" within the Kubernetes cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-97 Use kubectl to create the "webapp" namespace.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-98 Verify the successful creation of the namespace.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-99 As a developer, I want to write the necessary YAML files for the MySQL database within the Kubernetes cluster.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-100 Define the necessary configurations for deploying the MySQL database.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-101 Write the corresponding YAML files.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-106 As a developer, I want to write the necessary YAML files for the backend and frontend of the application within the Kubernetes cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-107 Define the necessary configurations for deploying the backend and frontend.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-108 Write the corresponding YAML files.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-117 As a developer, I want to add these YAML files to a new GitLab (or GitHub) repository to version the code.	À FAIRE	3 KC
<input checked="" type="checkbox"/> SCRUM-118 Create a new GitLab or GitHub repository and add and validate the YAML files in the repository.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-119 As a developer, I want to execute the YAML files within the local Kubernetes cluster (within the "webapp" namespace).	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-120 Use kubectl to deploy the YAML files within the local Kubernetes cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-121 Verify the successful deployment of objects/controllers.	À FAIRE	3 KC
<input checked="" type="checkbox"/> SCRUM-122 As a developer, I want to ensure locally using kubectl commands that the objects/controllers are created in the Kubernetes cluster.	À FAIRE	3 KC
<input checked="" type="checkbox"/> SCRUM-123 Use kubectl to check the creation of objects/controllers in the local cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-124 Confirm that all necessary components are operational.	À FAIRE	5 KC

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.

Sprint 4 28 janv. – 4 févr. (20 tickets)		
Enhance user experience with search functionality and real-time result updates, along with informative notifications.		
<input checked="" type="checkbox"/> SCRUM-93 As a developer, I want to create a local Kubernetes cluster using Minikube to facilitate development and testing.	À FAIRE	8 KC
<input type="checkbox"/> SCRUM-94 Install Minikube on the development machine.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-95 Configure Minikube to create a local Kubernetes cluster.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-96 As a developer, I want to create a namespace named "webapp" within the Kubernetes cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-97 Use kubectl to create the "webapp" namespace.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-98 Verify the successful creation of the namespace.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-99 As a developer, I want to write the necessary YAML files for the MySQL database within the Kubernetes cluster.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-100 Define the necessary configurations for deploying the MySQL database.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-101 Write the corresponding YAML files.	À FAIRE	8 KC
<input checked="" type="checkbox"/> SCRUM-106 As a developer, I want to write the necessary YAML files for the backend and frontend of the application within the Kubernetes cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-107 Define the necessary configurations for deploying the backend and frontend.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-108 Write the corresponding YAML files.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-117 As a developer, I want to add these YAML files to a new GitLab (or GitHub) repository to version the code.	À FAIRE	3 KC
<input checked="" type="checkbox"/> SCRUM-118 Create a new GitLab or GitHub repository and add and validate the YAML files in the repository.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-119 As a developer, I want to execute the YAML files within the local Kubernetes cluster (within the "webapp" namespace).	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-120 Use kubectl to deploy the YAML files within the local Kubernetes cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-121 Verify the successful deployment of objects/controllers.	À FAIRE	3 KC
<input checked="" type="checkbox"/> SCRUM-122 As a developer, I want to ensure locally using kubectl commands that the objects/controllers are created in the Kubernetes cluster.	À FAIRE	3 KC
<input checked="" type="checkbox"/> SCRUM-123 Use kubectl to check the creation of objects/controllers in the local cluster.	À FAIRE	5 KC
<input checked="" type="checkbox"/> SCRUM-124 Confirm that all necessary components are operational.	À FAIRE	5 KC

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 :

D DevSecOps-backend

main devsecops-backend / +

History Find file Edit Code

Update file .gitlab-ci.yml Mouad Tigmouti authored 10 hours ago a138a0db

Name	Last commit	Last update
src	Update 2 files	10 hours ago
.dockerignore	docker ignore	1 day ago
.gitlab-ci.yml	Update file .gitlab-ci.yml	10 hours ago
Dockerfile	Dockerfile for the backend	1 week ago
README.md	Backend added	1 week ago
mvnw	Backend added	1 week ago
mvnw.cmd	Backend added	1 week ago
pom.xml	Backend added	1 week ago

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.

A screenshot of a code editor showing a Dockerfile. The Dockerfile contains the following code:

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

The code editor interface shows tabs for 'Dockerfile' (selected), 'TutorialController.java', and '.gitlab-ci.yml'. The Dockerfile tab displays the code above.

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

```
.gitlab-ci.yml x
.gitlab-ci.yml
1   image: docker:latest
2   services:
3     - docker:dind
4
5   stages:
6     - build
7     - package
8     - deploy
9
10
11 maven-build:
12   image: maven:latest
13   stage: build
14   script: "mvn clean package -B"
15   artifacts:
16     paths:
17       - target/*.jar
18
19 docker-build:
20
21   stage: package
22   rules:
23     - exists:
24       - Dockerfile
25   before_script:
26     - docker login -u mouadtigmouti -p $DOCKER_Password
27
28   script:
29     - docker build -t mouadtigmouti/devsecops-backend .
30     - docker push mouadtigmouti/devsecops-backend
31
32 docker-deploy:
33   stage: deploy
34
35   script:
36
37
38     - docker run -d --name mysql_docker -e MYSQL_ROOT_PASSWORD=$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 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

```
.gitlab-ci.yml x
.gitlab-ci.yml
1  image: docker:latest
2  services:
3  | - docker:dind
4
5  stages:
6  | - build
7  | - package
8  | - deploy
9
10
11
12 maven-build:
13   image: maven:latest
14   stage: build
15   script: "mvn clean package -B"
16   artifacts:
17     paths:
18       - target/*.jar
19
```

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

Mouad Tigmouti / DevSecOps-backend / CI/CD Settings

Runners

Runners are processes that pick up and execute CI/CD jobs for GitLab. [What is GitLab Runner?](#)

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine.

How do runners pick up jobs?

Runners are either:

- **active** - Available to run jobs.
- **paused** - Not available to run jobs.

Tags control which type of jobs a runner can handle. By tagging a runner, you make sure shared runners only handle the jobs they are equipped to run. [Learn more.](#)

Project runners

These runners are assigned to this project.

New project runner ⋮

Assigned project runners

#30969675 (rxUQwLC4a) ⋮ Disable for this project

Frontend_runner

Shared runners

These runners are available to all groups and projects.

Each CI/CD job runs on a separate, isolated virtual machine.

Enable shared runners for this project

Available shared runners: 78

⋮ #1506020 (Hs8mheX51)
windows-shared-runners-manager-1



1.3-Backend .gitlab-ci.yml

Here it took 3 minutes to complete the backend pipeline :

Update file .gitlab-ci.yml
#1129748195 ⚡ main ➔ f1106ccd ● latest

Mouad Tigmouti / DevSecOps-backend / Pipelines / #1129748195

Update file .gitlab-ci.yml

Passed Mouad Tigmouti created pipeline for commit f1106ccd ⚡ finished 1 day ago

For main

3 Jobs 3.64 3 minutes 38 seconds, queued for 0 seconds

Pipeline Needs Jobs 3 Tests 0

```
graph LR; build[maven-build] --> package[docker-build]; package --> deploy[docker-deploy]
```

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

```
107 8ca16d33e263c5a7a380e8470080c053a072c77707etc570a153877070a1511a
108 $ docker ps
109 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
  NAMES
110 6ca16d33e265 mouadtigmouti/devsecops-backend "java -jar spring-bo..." 4 seconds ago Up Less than a second 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp
     p, 8081/tcp devsecops-backend
111 841c2c0f5394 mysql "docker-entrypoint.s..." 46 seconds ago Up 40 seconds 3306/tcp, 33060/tcp
     mysql_docker
112 Cleaning up project directory and file based variables
113 Job succeeded 00:01
```

- 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 :

Mouad Tigmouti / DevSecOps-frontend

D DevSecOps-frontend

main devsecops-frontend / +

History Find file Edit Code

Update file tutorial.service.ts Mouad Tigmouti authored 14 hours ago 0254b5b2

Name	Last commit	Last update
src	Update file tutorial.service.ts	14 hours ago
.gitlab-ci.yml	Update 2 files	22 hours ago
Dockerfile	Dockerfile added	1 week ago
README.md	Frontend added	1 week ago
angular-16-crud-example.png	Frontend added	1 week ago
angular.json	Frontend added	1 week ago
package-lock.json	Frontend added	1 week ago
package.json	Frontend added	1 week ago
tsconfig.app.json	Frontend added	1 week ago
tsconfig.json	Frontend added	1 week ago
tsconfig.spec.json	Frontend added	1 week ago

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 X
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

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows a project structure for a 'DEVSECOPS-FRONTEND' repository. The files listed include 'src', '.gitlab-ci.yml', 'angular-16-crud-example.png', 'angular.json', 'Dockerfile', 'package-lock.json', 'package.json', 'README.md', 'tsconfig.app.json', 'tsconfig.json', and 'tsconfig.spec.json'. The '.gitlab-ci.yml' file is currently selected and open in the main editor area. The code itself defines a CI pipeline with two stages: 'Package' and 'Deploy'.

```
EXPLORER ... .gitlab-ci.yml x | .gitlab-ci.yml
1 image: docker:latest
2
3 services:
4   - docker:dind
5
6 stages:
7   - package
8   - deploy
9
10
11 before_script:
12   - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
13
14 docker-build:
15   stage: package
16   script:
17     - docker build -t mouadtigmouti/devsecops-frontend .
18     - docker push mouadtigmouti/devsecops-frontend
19
20
21 docker-deploy:
22   stage: deploy
23   script:
24     - docker run -d -p 8081:8081 --name devsecops-frontend mouadtigmouti/devsecops-frontend
25     - sleep 10
26     - docker logs devsecops-frontend
27
28
29
```

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



2.2-Frontend .gitlab-ci.yml

The screenshot shows the VS Code interface with the Explorer sidebar open, displaying files for a project named 'DEVSECOPS-FRONTEND'. The '.gitlab-ci.yml' file is selected in the Explorer. The code editor shows the following .gitlab-ci.yml configuration:

```
image: docker:latest
services:
  - docker:dind
stages:
  - package
  - deploy
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
  script:
    - docker run -d -p 8081:8081 --name devsecops-frontend mouadtigmouti/devsecops-frontend
    - sleep 10
    - docker logs devsecops-frontend
```

Two curly braces on the right side of the code editor group the 'before_script' block and the 'docker-build' and 'docker-deploy' blocks respectively, indicating they belong to the same stages.

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 :

Mouad Tigmouti / DevSecOps-frontend / CI/CD Settings

Project runners

These runners are assigned to this project.

New project runner ⋮

Assigned project runners

#30969675 (rxUQwLC4a) ⋮ Disable for this project

Frontend_runner

Other available runners

Shared runners

These runners are available to all groups and projects.

Each CI/CD job runs on a separate, isolated virtual machine.

Enable shared runners for this project

Available shared runners: 78

#1506020 (Hs8mheX51) windows-shared-runners-manager-1 ⋮

shared-windows windows windows-1809



2.3-Frontend .gitlab-ci.yml

Here it took 3 minutes to complete the frontend pipeline :

Passed
00:03:48
34 minutes ago

Update file .gitlab-ci.yml
#1129714511 main f7b5937b latest

Update file .gitlab-ci.yml
Passed Mouad Tigmouti created pipeline for commit f7b5937b finished 1 day ago
For main
2 Jobs 3.81 3 minutes 48 seconds, queued for 0 seconds

Pipeline Needs Jobs 2 Tests 0

```
graph LR; package[package<br/>docker-build] --> deploy[deploy<br/>docker-deploy]
```

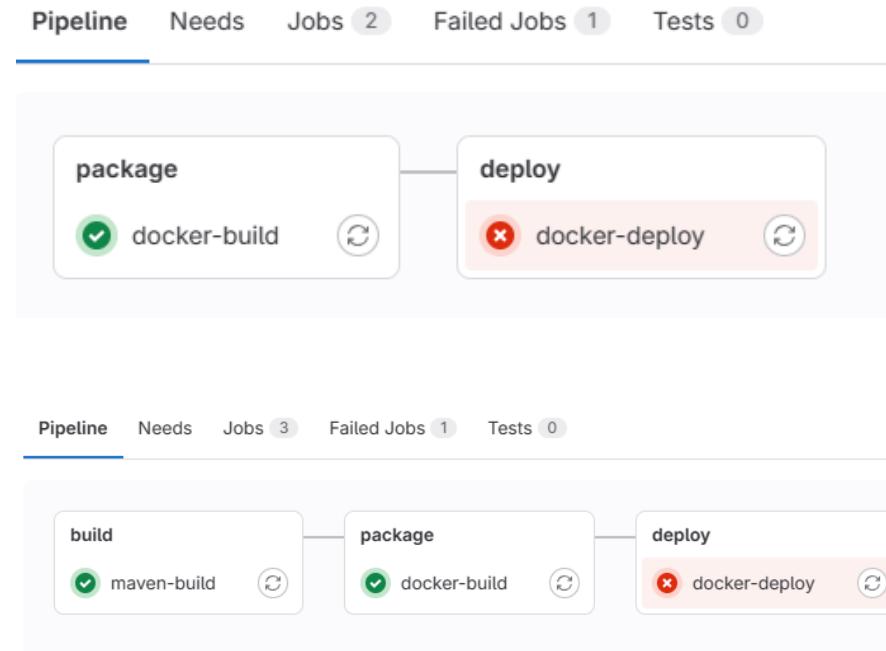
This is the result of the last command in the pipeline:

```
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
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



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



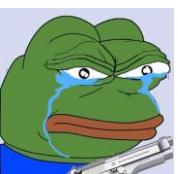
we need to install docker



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 max_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_builds=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://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
CREATED        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-django-celerybeat-1
3f0e06771804   defectdojo/defectdojo-django:latest "wait-for-it.sh mys..."  20
hours ago      Up 10 hours
django-defectdojo-django-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
2a1db62988bf   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.

Nom * devsecops-project

GÉRER LES TAGS ET LES ÉTIQUETTES.

Région * europe-west9 (Paris) Zone * europe-west9-a

La sélection d'une région est définitive La zone est définitive

Configuration de la machine

Usage général Optimisé pour le calcul Mémoire optimisée GPU

Types de machines pour les charges de travail courantes permettant d'optimiser les coûts et la flexibilité

Series	Description	vCPUs	Memory	Plate-forme
C3	Performances élevées et constantes	4 - 176	8 - 1408 Go	Intel Sapphire Rapids
C3D	Performances élevées et constantes	4 - 360	8 - 2880 Go	AMD Genoa
E2	Informatique au quotidien à faible coût	0.25 - 32	1 - 128 Go	En fonction de la disponibilité
N2	Équilibre entre prix et performances	2 - 128	2 - 864 Go	Intel Cascade Lake et Ice Lake
N2D	Équilibre entre prix et performances	2 - 224	2 - 896 Go	AMD EPYC
T2A	Charges de travail à scaling horizontal	1 - 48	4 - 192 Go	Ampere Altra ARM
T2D	Charges de travail à scaling horizontal	1 - 60	4 - 240 Go	AMD EPYC Milan
N1	Équilibre entre prix et performances	0.25 - 96	0,6 - 624 Go	Intel Skylake

Type de machine

Choisissez un type de machine avec des quantités prédéfinies de vCPU et de mémoire adaptées à la plupart des charges de travail. Vous pouvez aussi créer une machine personnalisée pour les besoins spécifiques de votre charge de travail. [En savoir plus](#)

PRÉDEFINI PERSONNALISÉ

e2-standard-8 (8 vCPU, 4 cœur(s), 32 Go de mémoire)

	VCPU 8 (4 coeurs)	Memory 32 Go
--	----------------------	-----------------

Estimation mensuelle
246,70 \$US

Soit un coût horaire d'environ 0,34 \$US

Vous payez ce que vous consommez : facturation à la seconde, sans frais initiaux

Élément	Estimation mensuelle
8 vCPU + 32 GB memory	226,98 \$US
Disque persistant SSD de 100 Go	19,72 \$US
Total	246,70 \$US

[Tarifs de Compute Engine](#)

[LESS](#)

Disque de démarrage

Sélectionnez une image ou un instantané pour créer un disque de démarrage, ou associez un disque existant. Vous ne trouvez pas ce que vous recherchez ? Explorez des centaines de solutions de VM dans [Marketplace](#)

IMAGES PUBLIQUES IMAGES PERSONNALISÉES INSTANTANÉS INSTANTANÉS D'ARCHIVE DISQUES EXISTANTS

Système d'exploitation Ubuntu

Version * Ubuntu 23.10

x86_64, amd64 mantic image built on 2023-12-15

Type de disque de démarrage * Disque persistant SSD

[COMPARER LES TYPES DE DISQUES](#)

Taille (Go) * 100

Provisionnez entre 10 et 65536 Go

[AFFICHER LES OPTIONS DE CONFIGURATION AVANCÉES](#)

SÉLECTIONNER ANNULER



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

```
ssh.cloud.google.com/v2/ssh/projects/polar-equinox-410222/zones/europe-west9-a/instances/devsecops-project?authuser=4&hl=fr&pr...
https://ssh.cloud.google.com/v2/ssh/projects/polar-equinox-410222/zones/europe-west9-a/instances/devsecops-project?authuser=4&hl=fr&pr...

SSH dans votre navigateur IMPORTER UN FICHIER TÉLÉCHARGER LE FICHIER

root@devsecops-project:/home/thehuntison000# sudo apt-get update
Hit:1 http://europe-west9-a.gce.clouds.archive.ubuntu.com/ubuntu mantic InRelease
Hit:2 http://security.ubuntu.com/ubuntu mantic-security InRelease
Hit:3 http://europe-west9-a.gce.clouds.archive.ubuntu.com/ubuntu mantic-updates InRelease
Hit:4 http://europe-west9-a.gce.clouds.archive.ubuntu.com/ubuntu mantic-backports InRelease
Reading package lists... Done
root@devsecops-project:/home/thehuntison000# apt-get install ca-certificates curl gnupg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu1).
ca-certificates set to manually installed.
curl is already the newest version (8.2.1-1ubuntu3.2).
curl set to manually installed.
gnupg is already the newest version (2.2.40-1.1ubuntu1).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
root@devsecops-project:/home/thehuntison000# install -m 0755 -d /etc/apt/keyrings
root@devsecops-project:/home/thehuntison000# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
root@devsecops-project:/home/thehuntison000# sudo chmod a+r /etc/apt/keyrings/docker.gpg
root@devsecops-project:/home/thehuntison000# 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
root@devsecops-project:/home/thehuntison000# apt-get update
Get:1 https://download.docker.com/linux/ubuntu mantic InRelease [48.8 kB]
Get:2 https://download.docker.com/linux/ubuntu mantic/stable amd64 Packages [3095 B]
Hit:3 http://europe-west9-a.gce.clouds.archive.ubuntu.com/ubuntu mantic InRelease
Hit:4 http://security.ubuntu.com/ubuntu mantic-security InRelease
Hit:5 http://europe-west9-a.gce.clouds.archive.ubuntu.com/ubuntu mantic-updates InRelease
Hit:6 http://europe-west9-a.gce.clouds.archive.ubuntu.com/ubuntu mantic-backports InRelease
Fetched 51.9 kB in 1s (84.8 kB/s)
Reading package lists... Done
root@devsecops-project:/home/thehuntison000#
```

```
ssh.cloud.google.com/v2/ssh/projects/polar-equinox-410222/zones/europe-west9-a/instances/devsecops-project?authuser=4&hl=fr&pr...
https://ssh.cloud.google.com/v2/ssh/projects/polar-equinox-410222/zones/europe-west9-a/instances/devsecops-project?authuser=4&hl=fr&pr...

SSH dans votre navigateur IMPORTER UN FICHIER TÉLÉCHARGER LE FICHIER

root@devsecops-project:/home/thehuntison000# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
root@devsecops-project:/home/thehuntison000# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
root@devsecops-project:/home/thehuntison000#
```

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
```

The screenshot shows a terminal window with the following session:

```
ssh.cloud.google.com/v2/ssh/projects/polar-equinox-410222/zones/europe-west9-a/instances/devsecops-project?authuser=4&hl=fr&projectNumber=248338440976&useAdminProxy=true
https://ssh.cloud.google.com/v2/ssh/projects/polar-equinox-410222/zones/europe-west9-a/instances/devsecops-project?authuser=4&hl=fr&projectNumber=248338440976&useAdminProxy=true
SSH dans votre navigateur
IMPORTER UN FICHIER TÉLÉCHARGER LE FICHIER
root@devsecops-project:/home/thehuntison000# sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time     Time      Current
               Dload  Upload Total   Spent    Left Speed
100 61.6M  100 61.6M    0     0  22.6M    0:00:02  0:00:02  --:--:-- 22.6M
root@devsecops-project:/home/thehuntison000# sudo chmod +x /usr/local/bin/gitlab-runner
root@devsecops-project:/home/thehuntison000# sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash
root@devsecops-project:/home/thehuntison000# gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner
Runtime platform
arch=amd64 os=linux pid=4290 revision=102c81ba version=16.7.0
root@devsecops-project:/home/thehuntison000#
```



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

Runners

Runners are processes that pick up and execute CI/CD jobs for GitLab. [What is GitLab Runner?](#)

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your

How do runners pick up jobs?

Runners are either:

- active - Available to
- paused - Not availa

Tags control which type of

Project runners

These runners are assig

New project runner

Registration token

⚠ Support for registration tokens is deprecated

.....



Show runner installation and
registration instructions

Reset registration token

Shared runners

These runners are availa

Each CI/CD job runs on a

Enable shared runners for t



Available shared runners: 7

● #30969675 (rxUQwLC4a)

Frontend_runner

✖ Disable for this project

Assigned project runners

Other available runners



3.1-Gitlab-runner

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

The screenshot shows a terminal window titled "SSH dans votre navigateur" with the URL <https://ssh.cloud.google.com/v2/ssh/projects/polar-equinox-410222/zones/europe-west9-a/instances/devsecops-project?authuser=4&hl=fr&projectNumber=248338440976&useAdminProxy=true>. The terminal output is as follows:

```
root@devsecops-project:/home/thehuntison000# gitlab-runner register
Runtime platform          arch=amd64 os=linux pid=4416 revision=102c81ba version=16.7.0
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com
Enter the registration token:
glrt-Knc7KPYP6_mywHfYMqaE
Verifying runner... is valid           runner=Knc7KPYP6
Enter a name for the runner. This is stored only in the local config.toml file:
[devsecops-project]: CloudRunner
Enter an executor: shell, ssh, virtualbox, docker-autoscaler, docker+machine, parallels, docker, docker-windows, instance, kubernetes, custom:
docker
Enter the default Docker image (for example, ruby:2.7):
docker:latest
Runner registered successfully. Feel free to start it, but if it's running already the config should be automatically reloaded!

Configuration (with the authentication token) was saved in "/etc/gitlab-runner/config.toml"
root@devsecops-project:/home/thehuntison000# gitlab-runner run
Runtime platform          arch=amd64 os=linux pid=4442 revision=102c81ba version=16.7.0
Starting multi-runner from /etc/gitlab-runner/config.toml... builds=0 max_builds=0
Running in system-mode.

Configuration loaded          builds=0 max_builds=1
listen_address not defined, metrics & debug endpoints disabled  builds=0 max_builds=1
[session_server].listen_address not defined, session endpoints disabled  builds=0 max_builds=1
Initializing executor providers  builds=0 max_builds=1
```



3.1-Gitlab-runner

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

Mouad Tigmouti / DevSecOps-frontend / CI/CD Settings

The screenshot shows the 'CI/CD Settings' page for the 'DevSecOps-frontend' project. It has two main sections: 'Project runners' and 'Shared runners'.

Project runners: These runners are assigned to this project. It includes a 'New project runner' button and a '⋮' menu. Under 'Assigned project runners', there are two entries:

- #31285966 (Knc7KPYP6) - Cloud runner for DevSecOps project
- #30969675 (rxUQwLC4a) - Frontend_runner

For each entry, there are edit, disable, and remove buttons. A note says 'Available shared runners: 78'.

Shared runners: These runners are available to all groups and projects. Each CI/CD job runs on a separate, isolated virtual machine. A toggle switch is shown as off. The section lists three runners:

- #1506020 (Hs8mheX51) - windows-shared-runners-manager-1 (shared-windows, windows, windows-1809)
- #1506021 (6QgxEPvRr) - windows-shared-runners-manager-2 (shared-windows, windows, windows-1809)
- #11573930 (KzYhZxBvi) - 1-blue.shared-gitlab-org.runners-manager.gitlab.com (gitlab-org)

Annotations on the left side point to specific runners:

- An arrow points to the first runner in the 'Assigned project runners' list with the text 'The runner we've just created'.
- An arrow points to the second runner in the 'Assigned project runners' list with the text 'The runner 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
1  image: docker:latest
2  services:
3    - docker:dind
4
5  stages:
6    - build
7    - package
8    - deploy
9
10 variables:
11   DOCKER_HOST: tcp://172.17.0.1:2375
12   DOCKER_DRIVER: overlay2
13
14 maven-build:
15   image: maven:latest
16   stage: build
17   script: "mvn clean package -B"
18   artifacts:
19     paths:
20       - target/*.jar
21
22 docker-build:
23
24   stage: package
25   rules:
26     - exists:
27       - Dockerfile
28   before_script:
29     - docker login -u mouadtigmouti -p $DOCKER_Password
30
31   script:
32     - docker build -t mouadtigmouti/devsecops-backend .
33     - 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

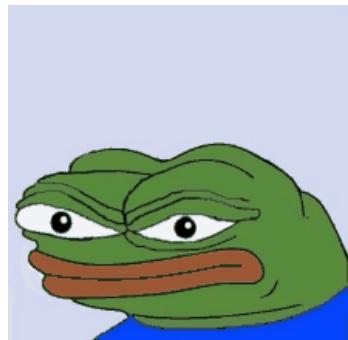
The other part of the pipeline code remains the same :

```
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

“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.



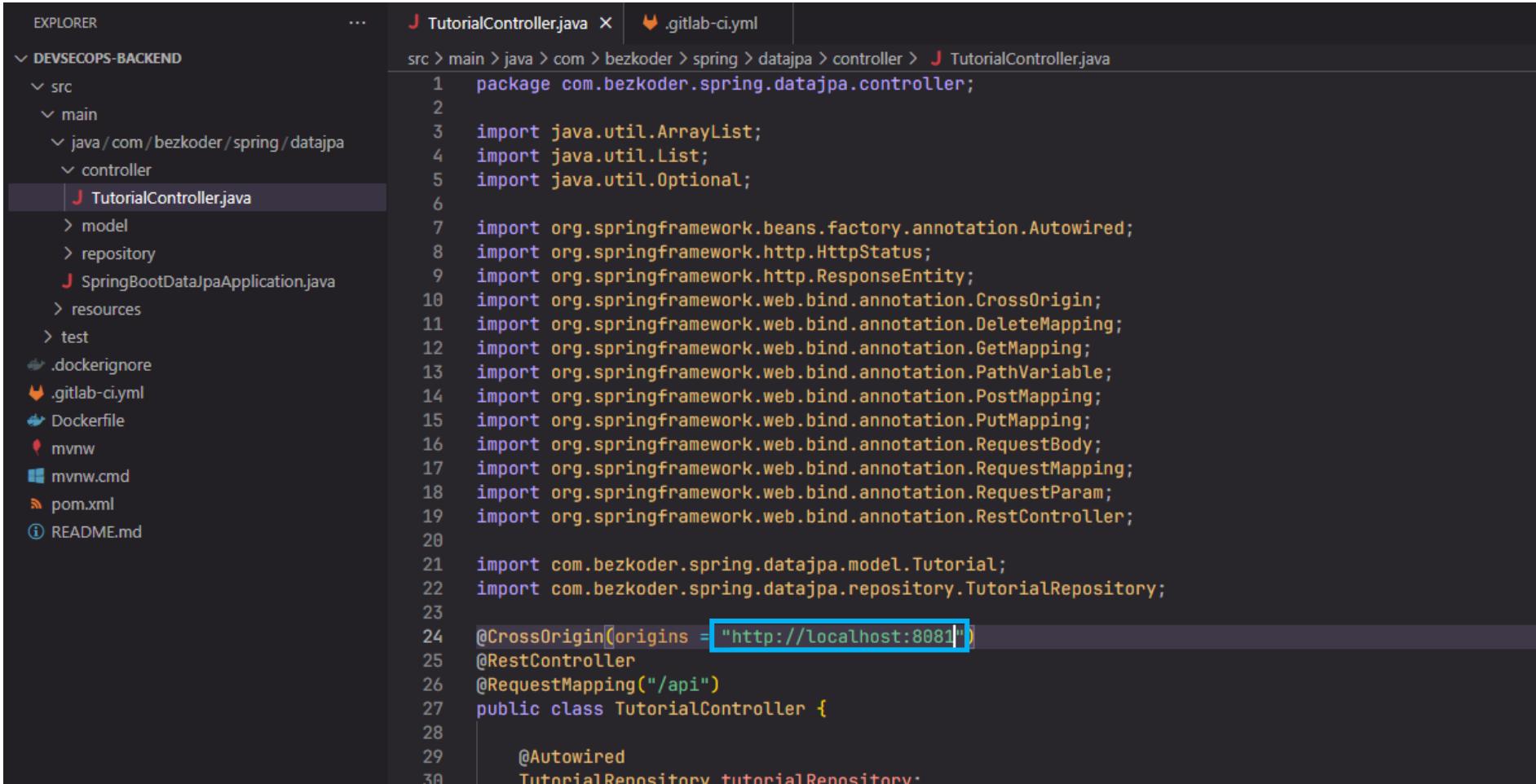
To connect the backend and the database in the same network



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>:



The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure under `DEVSECOPS-BACKEND`, including `src`, `main`, `java/com/bezkoder/spring/datajpa/controller`, `TutorialController.java`, and other files like `.gitlab-ci.yml` and `Dockerfile`.
- CODE EDITOR:** Displays the `TutorialController.java` file with the following code:

```
1 package com.bezkoder.spring.datajpa.controller;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Optional;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpStatus;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.CrossOrigin;
11 import org.springframework.web.bind.annotation.DeleteMapping;
12 import org.springframework.web.bind.annotation.GetMapping;
13 import org.springframework.web.bind.annotation.PathVariable;
14 import org.springframework.web.bind.annotation.PostMapping;
15 import org.springframework.web.bind.annotation.PutMapping;
16 import org.springframework.web.bind.annotation.RequestBody;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.bind.annotation.RequestParam;
19 import org.springframework.web.bind.annotation.RestController;
20
21 import com.bezkoder.spring.datajpa.model.Tutorial;
22 import com.bezkoder.spring.datajpa.repository.TutorialRepository;
23
24 @CrossOrigin(origins = "http://localhost:8081")
25 @RestController
26 @RequestMapping("/api")
27 public class TutorialController {
28
29     @Autowired
30     TutorialRepository tutorialRepository;
```



3.1-Gitlab-runner

Frontend

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

```
! .gitlab-ci.yml
1  image: docker:latest
2
3  services:
4    - docker:dind
5
6  stages:
7    - package
8    - deploy
9
10 variables:
11   DOCKER_HOST: tcp://172.17.0.1:2375
12   DOCKER_DRIVER: overlay2
13
14 before_script:
15   - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
16
17 docker-build:
18   stage: package
19   script:
20     - docker build -t mouadtigmouti/devsecops-frontend .
21     - docker push mouadtigmouti/devsecops-frontend
22
23
24 docker-deploy:
25   stage: deploy
26   before_script:
27     - docker stop devsecops-frontend || true
28     - docker rm devsecops-frontend || true
29   script:
30     - docker run -d -p 8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
31     - sleep 10
32     - docker logs devsecops-frontend
33
```

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>:

The screenshot shows a code editor with the following file structure:

- DEVSECOPS-FRONTEND
- src
- app
 - components
 - models
 - services
 - tutorial.service.spec.ts
 - tutorial.service.ts**
- assets
- favicon.ico
- index.html

The content of **tutorial.service.ts** is as follows:

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { Tutorial } from '../models/tutorial.model';
5
6 const baseUrl = 'http://localhost:8080/api/tutorials';
7
8 @Injectable({
9   providedIn: 'root',
10 })
11 export class TutorialService {
12   constructor(private http: HttpClient) {}
13
14   getAll(): Observable<Tutorial[]> {
15     return this.http.get<Tutorial[]>(baseUrl);
16   }
17
18   get(id: any): Observable<Tutorial> {
19     return this.http.get<Tutorial>(`${baseUrl}/${id}`);
20   }
21 }
```



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": {
     "766d6ba55bccd1c686bfe4e0b4b939b0a3bb31094005013fb731016beffb2b0": {
         "Name": "mysql_docker",
         "EndpointID": "0361ae3b977a6f0c07d1e83cccd5b83de74180f30a236a683c0a2c1d19b506c61",
         "MacAddress": "07:42: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 X
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

Runners

Runners are processes that pick up and execute CI/CD jobs for GitLab. [What is GitLab Runner?](#)

Register as many runners as you want. You can register runners as separate users, on separate servers, and on your local machine.

How do runners pick up jobs?

Runners are either:

- **active** - Available to run jobs.
- **paused** - Not available to run jobs.

Tags control which type of jobs a runner can handle. By tagging a runner, you make sure shared runners only handle the jobs they are equipped to run. [Learn more.](#)

Project runners

These runners are assigned to this project.

[New project runner](#) :

Assigned project runners

#30969675 (rxUQwLC4a)	 Disable for this project
Frontend_runner	
⚠ #30945921 (hGu3syGg2)	 Remove runner
Frontend runner	

Shared runners

These runners are available to all groups and projects.

Each CI/CD job runs on a separate, isolated virtual machine.

Enable shared runners for this project

Available shared runners: 78

#1506020 (Hs8mheX51)	windows-shared-runners-manager-1	
#1506021 (6QgxEPvRr)		

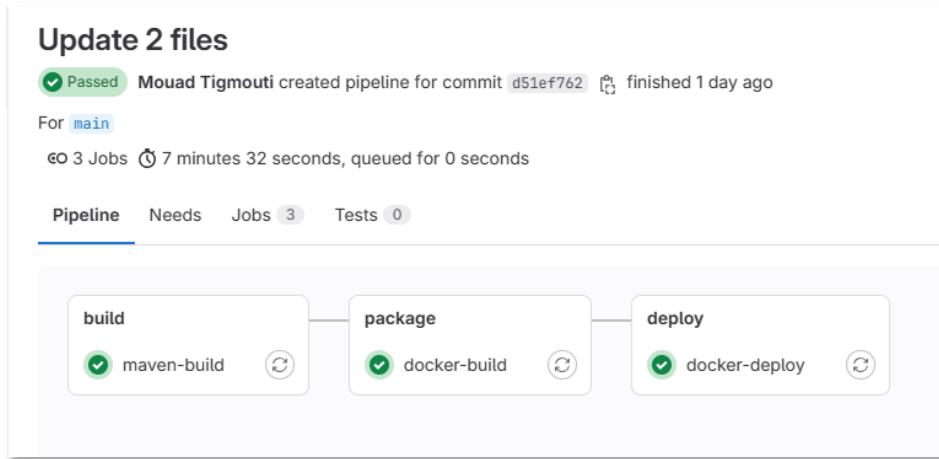
```
mouad@mouad-None:~$ gitlab-runner run
Runtime platform
6 revision=102c81ba version=16.7.0
Starting multi-runner from /home/mouad/.gitlab-runner/config.toml... builds=0 max_builds=0
WARNING: Running in user-mode.
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...

Configuration loaded
builds=0 max_builds=1
listen_address not defined, metrics & debug endpoints disabled builds=0 max_builds=1
[session_server].listen_address not defined, session endpoints disabled builds=0 max_builds=1
Initializing executor providers
builds=0 max_builds=1
```

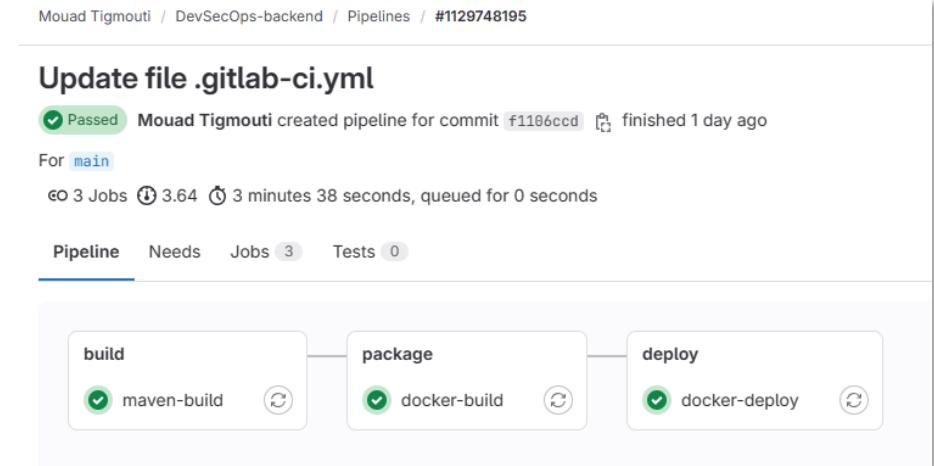


3.1-Gitlab-runner

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



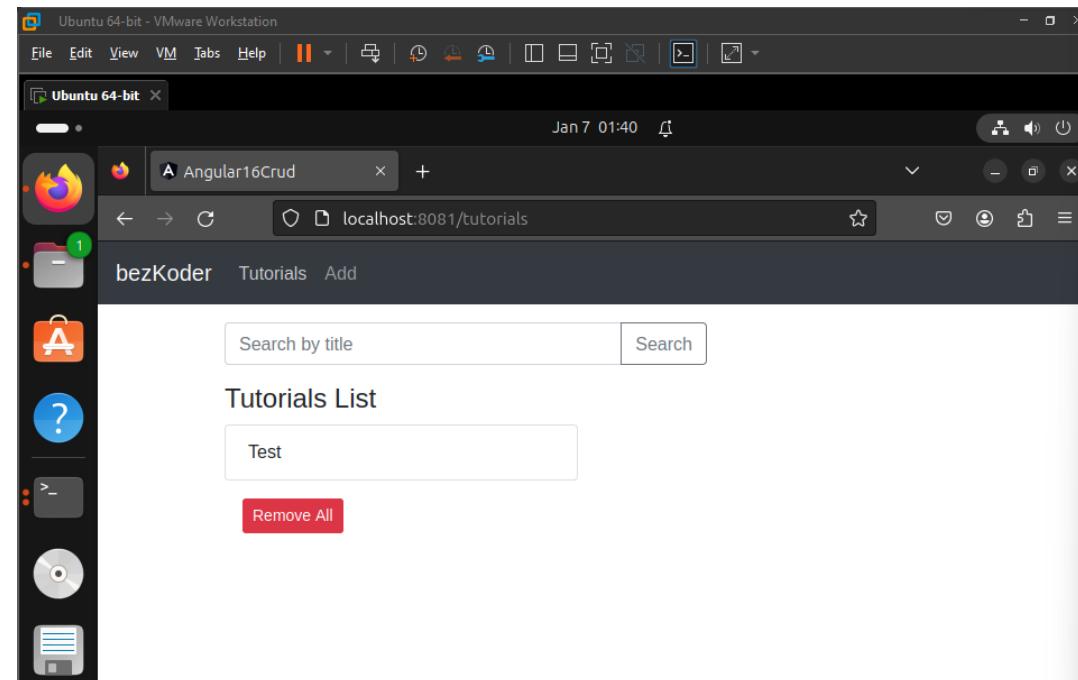
Gitlab shared runner



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   mouaditmouti/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   mouaditmouti/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 :

The screenshot shows a VMware Workstation interface with two browser windows running on an Ubuntu 64-bit VM.

- Left Window:** Displays the Angular16Crud application at `localhost:8081/tutorials`. The page title is "Angular16Crud". It shows a search bar with placeholder "Search by title" and a "Search" button. Below the search bar is a table with one row containing the text "Test". A red "Remove All" button is located below the table.
- Right Window:** Displays the same Angular16Crud application at `localhost:8081/tutorials`. The page title is "Angular16Crud". It shows a search bar with placeholder "Search by title" and a "Search" button. Below the search bar is a heading "Tutorials List" followed by a red "Remove All" button. The table below is empty.

Both windows have a header bar with "bezKoder", "Tutorials", and "Add" buttons. The left window's header bar also includes a "Search by title" input field and a "Search" button.

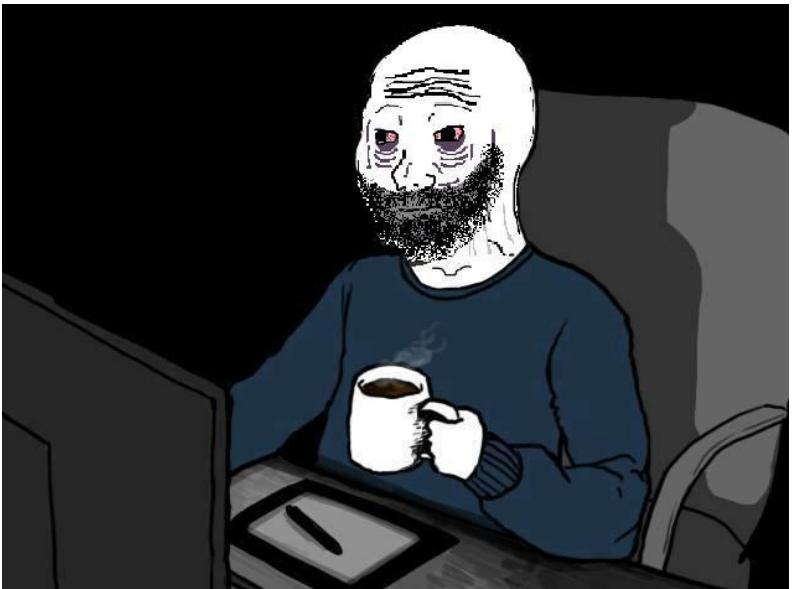
At the bottom of the VMware window, there is a message: "To direct input to this VM, move the mouse pointer inside or press Ctrl+G."



3.1-Gitlab-runner

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

A screenshot of a Microsoft Edge browser window. The title bar says "Angular16Crud". The address bar shows a warning icon and the URL "192.168.75.132:8081/tutorials". Below the address bar, there are navigation links for "bezKoder", "Tutorials", and "Add". The main content area displays a "Tutorials List" with a search bar containing "Search by title" and a "Search" button. A large red button labeled "Remove All" is visible. The background of the page is dark.



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 | J TutorialController.java | .gitlab-ci.yml
src > main > java > com > bezkoder > spring > datajpa > controller > J TutorialController.java
1 package com.bezkoder.spring.datajpa.controller;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Optional;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpStatus;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.CrossOrigin;
11 import org.springframework.web.bind.annotation.DeleteMapping;
12 import org.springframework.web.bind.annotation.GetMapping;
13 import org.springframework.web.bind.annotation.PathVariable;
14 import org.springframework.web.bind.annotation.PostMapping;
15 import org.springframework.web.bind.annotation.PutMapping;
16 import org.springframework.web.bind.annotation.RequestBody;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.bind.annotation.RequestParam;
19 import org.springframework.web.bind.annotation.RestController;
20
21 import com.bezkoder.spring.datajpa.model.Tutorial;
22 import com.bezkoder.spring.datajpa.repository.TutorialRepository;
23
24 @CrossOrigin(origins = "http://192.168.75.132:8081")
25 @RestController
26 @RequestMapping("/api")
27 public class TutorialController {
```

```
Dockerfile | TS tutorial.service.ts |
src > app > services > TS tutorial.service.ts > [?] baseUrl
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { Tutorial } from '../models/tutorial.model';
5
6 const baseUrl = 'http://192.168.75.132:8080/api/tutorials';
7
8 @Injectable({
9   providedIn: 'root',
10 })
```



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:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
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 devsecops-backend
evsecops-backend
3a47bdee7c3e mysql
mysql_docker
"docker-entrypoint.s..." 4 minutes ago Up 4 minutes 3306/tcp, 33060/tcp mysql_docker
d09360524af8 mouadtigmouti/devsecops-frontend "docker-entrypoint.s..." 26 minutes ago Up 26 minutes 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp devsecops-frontend
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-frontend
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-backend
kend
3a47bdee7c3e mysql
mysql_docker
"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 :

The screenshot displays a VMware workstation interface with two running Ubuntu 64-bit VMs. Both VMs are running Firefox browsers.

Left VM (Ubuntu 64-bit - VMware Workstation):

- The title bar says "Ubuntu 64-bit - VMware Workstation".
- The window title is "Ubuntu 64-bit".
- The address bar shows "192.168.75.132:8081/tutorials".
- The page content shows a "Tutorials List" section with a single entry: "Test".
- Below the list is a red "Remove All" button.

Right VM (Ubuntu 64-bit - VMware Workstation):

- The title bar says "Angular16Crud".
- The address bar shows "Non sécurisé | 192.168.75.132:8081/tutorials".
- The page content shows a "Tutorials List" section with a single entry: "Test".
- To the right of the list, there is a "Tutorial" panel with the following details:
 - Title:** Test
 - Description:** test1
 - Status:** Pending
- Below the panel are "Edit" and "Remove" buttons.



3.1-Gitlab-runner

Now we can affect changes from both machines

The image displays a dual-machine setup. On the left, a VMware Workstation window titled "Ubuntu 64-bit - VMware Workstation" shows a Firefox browser window. The browser is displaying a page titled "Angular16Crud" with the URL "192.168.75.132:8081/tutorials". The page content shows a "Tutorial was submitted successfully!" message and a list of tutorials with titles "Test" and "Test from windows". On the right, a Microsoft Edge browser window titled "Angular16Crud" shows the same "Tutorial was submitted successfully!" message and the list of tutorials. The desktop environment on the Linux host includes a dock with icons for various applications like a file manager, terminal, and browser.



3.1-Gitlab-runner

Now we can affect changes from both machines

The screenshot displays a VMware Workstation interface with two active virtual machines:

- Ubuntu 64-bit**: This machine is currently selected. A browser window titled "Angular16Crud" is open at the URL "192.168.75.132:8081/add". The page shows a success message: "Tutorial was submitted successfully!" and a green "Add" button.
- Windows 10**: This machine is visible in the background. Its browser window also displays the "Angular16Crud" page at "192.168.75.132:8081/tutorials", showing the same list of tutorials as the Ubuntu machine.

The list of tutorials on both screens includes:

- Test
- Test from windows
- Test from Ubuntu

A red "Remove All" button is located below the list on the Windows screen.



3.1-Generic Jobs

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

```
.gitlab-ci.yml
1  image: docker:latest
2  services:
3  | - docker:dind
4
5  stages:
6  | - build
7  | - package
8  | - deploy
9
10 variables:
11   DOCKER_HOST: tcp://172.17.0.1:2375
12   DOCKER_DRIVER: overlay2
13
14 maven-build:
15   image: maven:latest
16   stage: build
17   script: "mvn clean package -B"
18   artifacts:
19     paths:
20       - target/*.jar
21
22 docker-build:
23
24   stage: package
25   rules:
26     - exists:
27       - Dockerfile
28   before_script:
29     - docker login -u mouadtigmouti -p $DOCKER_Password
30
31   script:
32     - docker build -t mouadtigmouti/devsecops-backend .
33     - docker push mouadtigmouti/devsecops-backend
```

```
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
```

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 x
.gitlab-ci.yml
1  image: docker:latest
2
3  services:
4  | - docker:dind
5
6  stages:
7  | - package
8  | - deploy
9
10 variables:
11   DOCKER_HOST: tcp://172.17.0.1:2375
12   DOCKER_DRIVER: overlay2
13
14 before_script:
15 | - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
16
17 docker-build:
18   stage: package
19   script:
20   | - docker build -t mouadtigmouti/devsecops-frontend .
21   | - docker push mouadtigmouti/devsecops-frontend
22
23
24 docker-deploy:
25   stage: deploy
26   before_script:
27   | - docker stop devsecops-frontend || true
28   | - docker rm devsecops-frontend || true
29   | script:
30   | - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
31   | - sleep 10
32   | - 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.



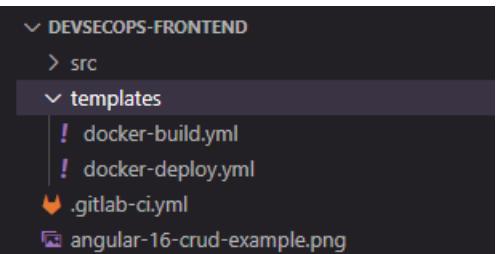
3.2-Generic Jobs

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

```
gitlab-ci.yml X | .gitlab-ci.yml
1  image: docker:latest
2
3  services:
4  | - docker:dind
5
6  stages:
7  | - package
8  | - deploy
9
10 variables:
11 | DOCKER_HOST: tcp://172.17.0.1:2375
12 | DOCKER_DRIVER: overlay2
13
14 before_script:
15 | - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
16
17 docker-build:
18 | stage: package
19 | script:
20 | | - docker build -t mouadtigmouti/devsecops-frontend .
21 | | - docker push mouadtigmouti/devsecops-frontend
22
23
24 docker-deploy:
25 | stage: deploy
26 | before_script:
27 | | - docker stop devsecops-frontend || true
28 | | - docker rm devsecops-frontend || true
29 | script:
30 | | - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
31 | | - sleep 10
32 | | - 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 x
templates > ! docker-build.yml
1 docker-build:
2   stage: package
3   before_script:
4     - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
5   script:
6     - docker build -t mouadtigmouti/devsecops-frontend .
7     - docker push mouadtigmouti/devsecops-frontend
```

```
! docker-deploy.yml x
templates > ! docker-deploy.yml
1 docker-deploy:
2   stage: deploy
3   before_script:
4     - docker stop devsecops-frontend || true
5     - docker rm devsecops-frontend || true
6   script:
7     - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
8     - sleep 10
9     - docker logs devsecops-frontend
```

```
! .gitlab-ci.yml x
!.gitlab-ci.yml
1 image: docker:latest
2
3 services:
4   - docker:dind
5
6 include:
7   - templates/docker-build.yml
8   - templates/docker-deploy.yml
9
10 stages:
11   - package
12   - deploy
13
14 variables:
15   DOCKER_HOST: tcp://172.17.0.1:2375
16   DOCKER_DRIVER: overlay2
17
```

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 x
templates > ! docker-build.yml
1 docker-build:
2   stage: package
3   before_script:
4     - echo "$DOCKER_Password" | docker login -u $DOCKER_Username --password-stdin
5   script:
6     - docker build -t mouadtigmouti/devsecops-frontend .
7     - docker push mouadtigmouti/devsecops-frontend
```

```
! docker-deploy.yml x
templates > ! docker-deploy.yml
1 docker-deploy:
2   stage: deploy
3   before_script:
4     - docker stop devsecops-frontend || true
5     - docker rm devsecops-frontend || true
6   script:
7     - docker run -d -p 192.168.75.132:8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
8     - sleep 10
9     - docker logs devsecops-frontend
```

```
!.gitlab-ci.yml x
.gitlab-ci.yml
1 image: docker:latest
2
3 services:
4   - docker:dind
5
6 include:
7   - templates/docker-build.yml
8   - templates/docker-deploy.yml
9
10 stages:
11   - package
12   - deploy
13
14 variables:
15   DOCKER_HOST: tcp://172.17.0.1:2375
16   DOCKER_DRIVER: overlay2
17
```

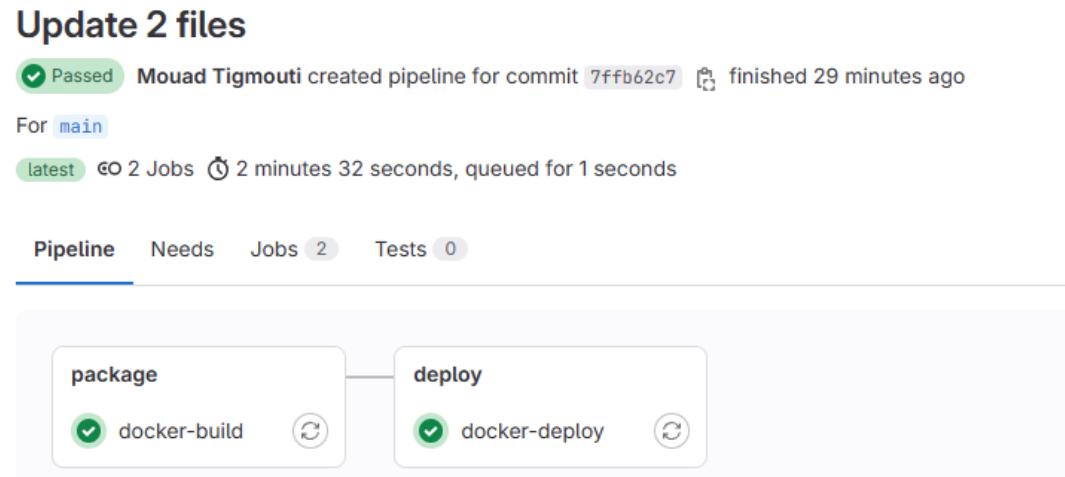
The pipeline code now looks lightweight and much better

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:~$ 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

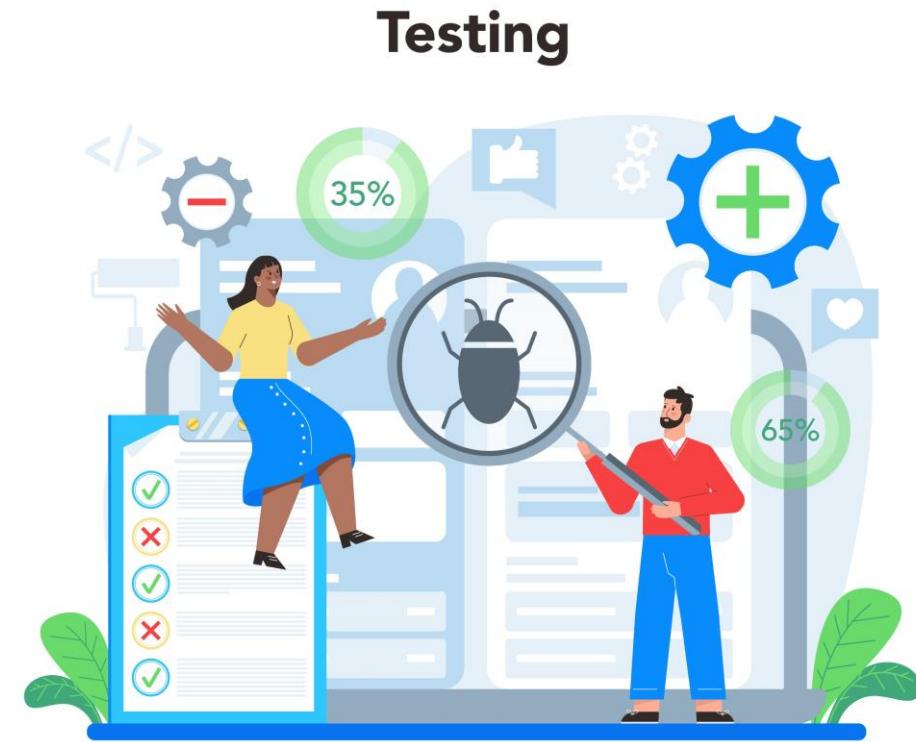
```
!.gitlab-ci.yml | ts tutorial.service.ts | docker-deploy.yml x
templates > ! docker-deploy.yml
1 docker-deploy:
2   stage: deploy
3   before_script:
4     - docker stop devsecops-frontend || true
5     - docker rm devsecops-frontend || true
6   rules:
7     - exists:
8       - Dockerfile
9     - exists:
10      - '$CI_COMMIT_BRANCH == "main"'
11   script:
12     - docker run -d -p 8081:8081 --name devsecops-frontend --network backend_network mouadtigmouti/devsecops-frontend
13     - sleep 10
14     - 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 these 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_secrets":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
    }
  },
  "results": []
}
```

```
{"_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 these tools for the SCA phase :



refireJS



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

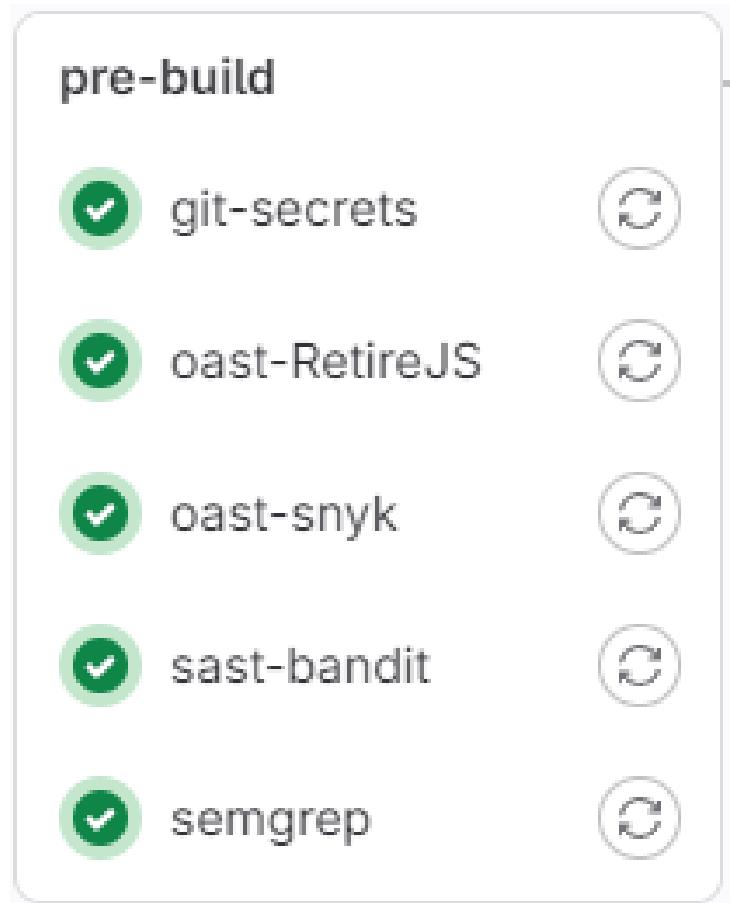


The result of the test :

```
70: Found 0 vulnerabilities
71: $ snyk auth 35589080-81a0-4492-af3d-82ee5f1140d0
72: Your account has been authenticated. Snyk is now ready to be used.
73: $ snyk test --json > snyk-results.json
74: $ cat snyk-results.json
75: {
76:   "vulnerabilities": [],
77:   "ok": true,
78:   "dependencyCount": 0,
79:   "org": "thehuntison000",
80:   "policy": "# Snyk (https://snyk.io) policy file, patches or ignores known vulnerabilities.\nversion: v1.25.1\nignore: {}\npatch: {}\\n",
81:   "isPrivate": true,
82:   "licensesPolicy": {
83:     "severities": {},
84:     "orgLicenseRules": {
85:       "AGPL-1.0": {
86:         "licenseType": "AGPL-1.0",
87:         "severity": "high",
88:         "instructions": ""
89:       },
90:       "AGPL-3.0": {
91:         "licenseType": "AGPL-3.0",
92:         "severity": "high",
93:         "instructions": ""
94:       },
95:       "Artistic-1.0": {
96:         "licenseType": "Artistic-1.0",
97:         "severity": "medium",
98:         "instructions": ""
99:       },
100:      "Artistic-2.0": {
101:        "licenseType": "Artistic-2.0",
102:        "severity": "medium",
103:        "instructions": ""
104:      },
105:      "CDDL-1.0": {
106:        "licenseType": "CDDL-1.0",
107:        "severity": "medium",
108:        "instructions": ""
109:      },
110:      "CPOL-1.02": {
111:        "licenseType": "CPOL-1.02",
112:        "severity": "high",
113:        "instructions": ""
114:      },
115:      "EPL-1.0": {
116:        "licenseType": "EPL-1.0",
117:        "severity": "medium",
118:        "instructions": ""
119:      }
120:    }
121:  }
122: }
```



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 severity 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 validate gpg keys with the master keyring, leading to a potential man-in-the-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 problems.	
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 compiled with Readline support. The Readline library uses an "INPUTRC" environment variable to get a path to the library config file. When the library cannot parse the specified file, it prints an error message containing data from the file. This flaw allows an unprivileged user to read 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 TIOCGSTI ioctl call, which 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 not prevent replacement of a plain file with a symlink during use of the POSIX "-R -L" options, which allows local users to modify the ownership of arbitrary files by leveraging a race condition.	
Unapproved	Critical	dpkg	1.20.9	Dpkg::Source::Archive in dpkg, the Debian package management system, before 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-place extraction of files from the archive can be controlled by the user.	

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 these tools for the DAST phase:



Nikto



NMAP



Zed Attack 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 type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /68q0Up9m.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:

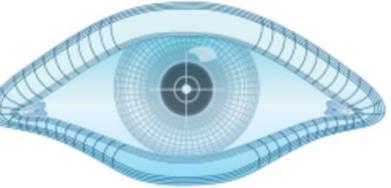
```
nmap:  
  stage: test  
  script:  
    - docker pull hysnsec/nmap  
    - docker run --rm -v $(pwd):/tmp hysnsec/nmap -sSVC -A -p 8083 192.168.75.132
```

1.4-DAST



NMAP

The result of the test :



NMAP

```
$ docker run --rm -v $(pwd):/tmp hybssec/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%20/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/1.1 408
|     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 408</title></head><body><font>Sorry, we're temporarily unavailable</font></body></html>
Request /title=style type="text/html; charset=UTF-8; font-family:Tahoma,arial, sans-serif; font-size:14px; font-weight:bold; color:#000000; background-color:#FFFFFF; margin-top:10px; margin-bottom:10px; border:1px solid black; padding:5px; border-radius:5px; width:300px; height:100px; overflow:scroll; position: absolute; left: 50%; top: 50%; transform: translate(-50%, -50%);</style>
```



1.4-DAST

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 Attack Proxy



The result of the test :

```
60 $ zap-full-scan.py -t http://192.168.75.132:8083 -r report.html
61 Total of 4 URLs
62 PASS: Directory Browsing [0]
63 PASS: Vulnerable JS Library (Powered by Retire.js) [10003]
64 PASS: In Page Banner Information Leak [10009]
65 PASS: Cookie No HttpOnly Flag [10010]
66 PASS: Cookie Without Secure Flag [10011]
67 PASS: Re-examine Cache-control Directives [10015]
68 PASS: Cross-Domain JavaScript Source File Inclusion [10017]
69 PASS: Content-Type Header Missing [10019]
70 PASS: Anti-clickjacking Header [10020]
71 PASS: X-Content-Type-Options Header Missing [10021]
72 PASS: Information Disclosure - Debug Error Messages [10023]
73 PASS: Information Disclosure - Sensitive Information in URL [10024]
74 PASS: Information Disclosure - Sensitive Information in HTTP Referrer Header [10025]
75 PASS: HTTP Parameter Override [10026]
76 PASS: Information Disclosure - Suspicious Comments [10027]
77 PASS: Open Redirect [10028]
78 PASS: Cookie Poisoning [10029]
```

1.4-DAST

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	
Attack	

- 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

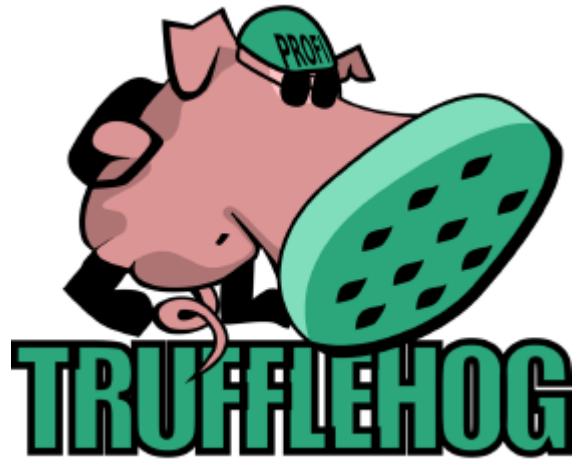


Section 3

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_secrets":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 0.0.0:8080 --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



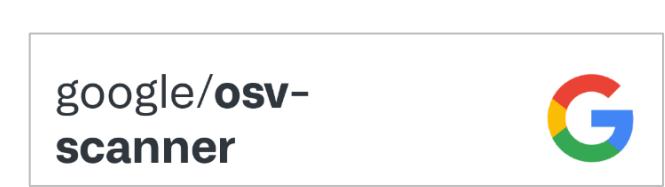
Same as the backend we are going to use the same tools with another new tool :



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



osv-scanner

google/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 found. 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 severity 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 validate gpg keys with the master keyring, leading to a potential man-in-the-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 problems.
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 compiled with Readline support. The Readline library uses an "INPUTRC" environment variable to get a path to the library config file. When the library cannot parse the specified file, it prints an error message containing data from the file. This flaw allows an unprivileged user to read 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, which 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 not correctly handle the --userspec option when combined with the -R recursive option.

2.4-DAST



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



Nikto



NMAP



Zed Attack 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 type. 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



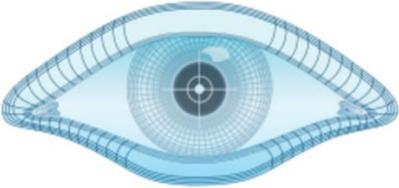
NMAP



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 Attack 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 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`

```
mouad@mouad-None:~$ sudo snap install kubectl --classic
[sudo] password for mouad:
kubectl 1.28.5 from Canonical✓ installed
mouad@mouad-None:~$
```

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
```

```
mouad@mouad-None:~$ sudo curl -Lo /usr/local/bin/minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
          % Received % Xferd  Average Speed   Time     Time     Time  Current
          Dload  Upload Total   Spent    Left  Speed
100 89.3M  100 89.3M    0      0  9239k      0  0:00:09  0:00:09  ---:--- 10.8M
mouad@mouad-None:~$ sudo chmod +x /usr/local/bin/minikube
mouad@mouad-None:~$
```



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 getContexts/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 getContexts/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](#)

```
mouad@mouad-None:~$ kubectl create namespace webapp
namespace/webapp created
mouad@mouad-None:~$ kubectl get namespaces
NAME      STATUS   AGE
default   Active   7m58s
kube-node-lease   Active   7m58s
kube-public   Active   7m58s
kube-system   Active   7m58s
webapp     Active   14s
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 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 x | ! backend.yaml | ! frontend.yaml  
! mysql.yaml  
1 apiVersion: apps/v1  
2 kind: Deployment  
3 metadata:  
4   name: mysql-deployment  
5   namespace: webapp  
6 spec:  
7   replicas: 1  
8   selector:  
9     matchLabels:  
10    app: mysql  
11   template:  
12     metadata:  
13       labels:  
14         app: mysql  
15     spec:  
16       containers:  
17         - name: mysql  
18           image: mysql:latest  
19           ports:  
20             - containerPort: 3306  
21           volumeMounts:  
22             - name: mysql-persistent-storage  
23               mountPath: /var/lib/mysql  
24 ---
```

```
24 ---  
25 apiVersion: v1  
26 kind: Service  
27 metadata:  
28   name: mysql-service  
29   namespace: webapp  
30 spec:  
31   selector:  
32     app: mysql  
33   ports:  
34     - protocol: TCP  
35       port: 3306  
36       targetPort: 3306  
37     type: ClusterIP
```



2-Creation necessary YAML files

Frontend:

```
! mysql.yaml | ! backend.yaml | ! frontend.yaml X
! frontend.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: devsecops-frontend-deployment
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: devsecops-frontend
10   template:
11     metadata:
12       labels:
13         app: devsecops-frontend
14     spec:
15       containers:
16         - name: devsecops-frontend
17           image: mouadtigmouti/devsecops-frontend
18           imagePullPolicy: Always
19         ports:
20           - name: http
21             containerPort: 8081
22
```

```
22
23   ---
24   apiVersion: v1
25   kind: Service
26   metadata:
27     name: devsecops-frontend-service
28   spec:
29     type: ClusterIP
30     selector:
31       app: devsecops-frontend
32     ports:
33       - name: http
34         port: 8081
35         targetPort: 8081
36
```



2-Creation necessary YAML files

Backend :

```
! mysql.yaml | ! backend.yaml X | ! frontend.yaml |
! backend.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: devsecops-backend-deployment
5  spec:
6    replicas: 1
7    selector:
8      matchLabels:
9        app: devsecops-backend
10   template:
11     metadata:
12       labels:
13         app: devsecops-backend
14   spec:
15     containers:
16       - name: devsecops-backend
17         image: mouadtigmouti/devsecops-
18           imagePullPolicy: Always
19         ports:
20           - containerPort: 8080
21
22 ---
```

```
23   apiVersion: v1
24   kind: Service
25   metadata:
26     name: devsecops-backend-service
27   labels:
28     app: devsecops-backend
29   spec:
30     type: ClusterIP
31   selector:
32     app: devsecops-backend
33   ports:
34     - name: http
35       port: 8080
36       targetPort: 8080
37
```



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
1 package com.bezkoder.spring.datajpa.controller;
2
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Optional;
6
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpStatus;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.CrossOrigin;
11 import org.springframework.web.bind.annotation.DeleteMapping;
12 import org.springframework.web.bind.annotation.GetMapping;
13 import org.springframework.web.bind.annotation.PathVariable;
14 import org.springframework.web.bind.annotation.PostMapping;
15 import org.springframework.web.bind.annotation.PutMapping;
16 import org.springframework.web.bind.annotation.RequestBody;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.bind.annotation.RequestParam;
19 import org.springframework.web.bind.annotation.RestController;
20
21 import com.bezkoder.spring.datajpa.model.Tutorial;
22 import com.bezkoder.spring.datajpa.repository.TutorialRepository;
23
24 @CrossOrigin(origins = "http://devsecops-frontend-service:8081")
25 @RestController
26 @RequestMapping("/api")
27 public class TutorialController {
```

```
TutorialController.java application.properties .gitlab-ci.yml
src > main > resources > application.properties
1 spring.datasource.url= jdbc:mysql://mysql-service: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
```

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 :

```
ts tutorial.service.ts X
src > app > services > ts tutorial.service.ts > [o] baseUrl
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { Tutorial } from '../models/tutorial.model';
5
6 const baseUrl = 'http://devsecops-backend-service:8080/api/tutorials';
7
```

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
```

The screenshot shows a terminal window with a dark theme. The title bar reads "mouad@mouad-None: ~/Desktop/k8s". The terminal content displays the output of three consecutive `kubectl apply` commands:

```
mouad@mouad-None:~/Desktop/k8s$ kubectl apply -f frontend.yaml -n webapp  
deployment.apps/devsecops-frontend-deployment created  
service/devsecops-frontend-service created  
mouad@mouad-None:~/Desktop/k8s$ kubectl apply -f mysql.yaml -n webapp  
deployment.apps/mysql-deployment created  
service/mysql-service created  
mouad@mouad-None:~/Desktop/k8s$ kubectl apply -f backend.yaml -n webapp  
deployment.apps/devsecops-backend-deployment created  
service/devsecops-backend-service created  
mouad@mouad-None:~/Desktop/k8s$
```



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 can't 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 minikube
  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: too many requests: You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limit
  Warning   Failed     87s (x6 over 3m17s)   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

```
mouad@mouad-None:~/Desktop/k8s$ kubectl get pods -n webapp
NAME                               READY   STATUS    RESTARTS   AGE
devsecops-backend-deployment-85775b8f4-th2qt   0/1     Error      6 (3m1s ago)  7m21s
devsecops-frontend-deployment-57c7cfb7d5-x2r6z   1/1     Running     0          7m36s
mysql-deployment-7987cdd5fc-vxjkg   0/1     ImagePullBackOff  0          7m27s
mouad@mouad-None:~/Desktop/k8s$ kubectl port-forward service/devsecops-frontend-service 8081:8081 -n webapp
Forwarding from 127.0.0.1:8081 -> 8081
Forwarding from [::1]:8081 -> 8081
Handling connection for 8081
bezKoder Tutorials Add
Search by title Search
Tutorials List
Remove All
```

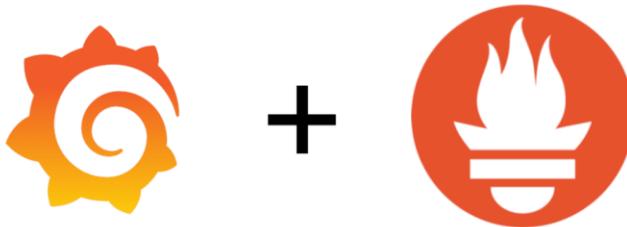


Bonus section

Monitoring



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



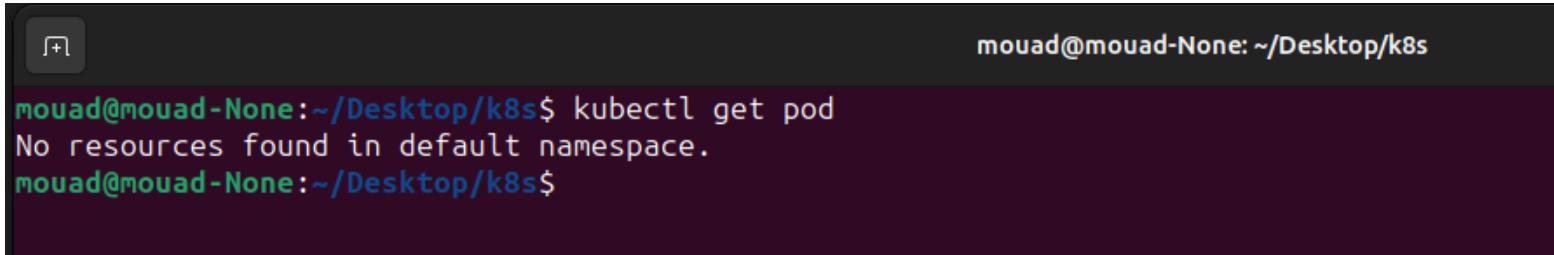
[Prometheus](#) is an open-source monitoring and alerting toolkit designed for reliability and scalability. It collects metrics from configured targets, stores them, and allows querying and alerting on this data.

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



Monitoring

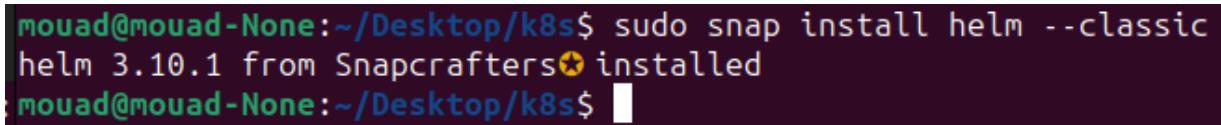
Before we start installation let's clean minikubes state and check if there is any running pods :



A screenshot of a terminal window titled "mouad@mouad-None: ~/Desktop/k8s". The command "kubectl get pod" is run, resulting in the output "No resources found in default namespace." The terminal has a dark theme with light-colored text.

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

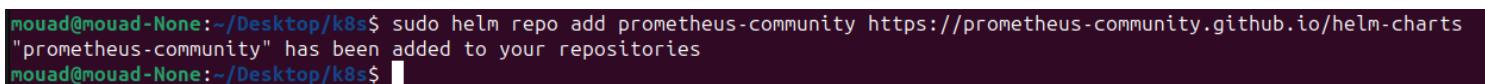
Then we need to install helm :



A screenshot of a terminal window titled "mouad@mouad-None: ~/Desktop/k8s". The command "sudo snap install helm --classic" is run, showing the output "helm 3.10.1 from Snapcrafters★ installed". The terminal has a dark theme with light-colored text.

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

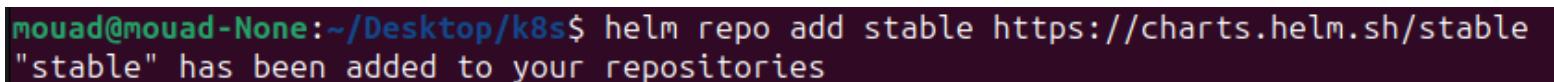
Now let's install Prometheus, to do so we need to add the repo with helm first :



A screenshot of a terminal window titled "mouad@mouad-None: ~/Desktop/k8s". The command "sudo helm repo add prometheus-community https://prometheus-community.github.io/helm-charts" is run, showing the output "'prometheus-community' has been added to your repositories". The terminal has a dark theme with light-colored text.

```
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



A screenshot of a terminal window titled "mouad@mouad-None: ~/Desktop/k8s". The command "helm repo add stable https://charts.helm.sh/stable" is run, showing the output "'stable' has been added to your repositories". The terminal has a dark theme with light-colored text.

```
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 Helm-ing!*
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-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-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 :

The screenshot displays two windows. The top window is a terminal session titled 'mouad@mouad-None: ~'. It shows the command `kubectl port-forward deployment/prometheus-grafana 3000` being run, followed by output indicating port forwarding from both 127.0.0.1:3000 and ::1:3000 to 3000, and a message about handling a connection for 3000. The bottom window is a web browser titled 'Grafana' showing the 'localhost:3000/login' page. The page features the Grafana logo (a sun-like icon) and the text 'Welcome to Grafana'. It includes input fields for 'Email or username' and 'Password', a 'Log in' button, and a 'Forgot your password?' link. The browser's address bar also shows 'localhost:3000/login'. At the bottom of the browser window, there is a footer with links to 'Documentation', 'Support', 'Community', 'Open Source', a version number 'v10.2.2 (161e3cac5075540918e3a39004f2364ad104d5bb)', and a 'New version available!' notification.

Monitoring



With the credentials [admin](#) and the password "prom-operator"

The screenshot shows the Grafana web interface at localhost:3000/?orgId=1. The title bar says "Grafana". The main header includes a search bar and links for "Documentation", "Tutorials", "Community", and "Public Slack".

Welcome to Grafana

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

TUTORIAL
[DATA SOURCE AND DASHBOARDS](#)
[Grafana fundamentals](#)
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

COMPLETE
Add your first data source

[Learn how in the docs](#)

COMPLETE
Create your first dashboard

[Learn how in the docs](#)

[Remove this panel](#)

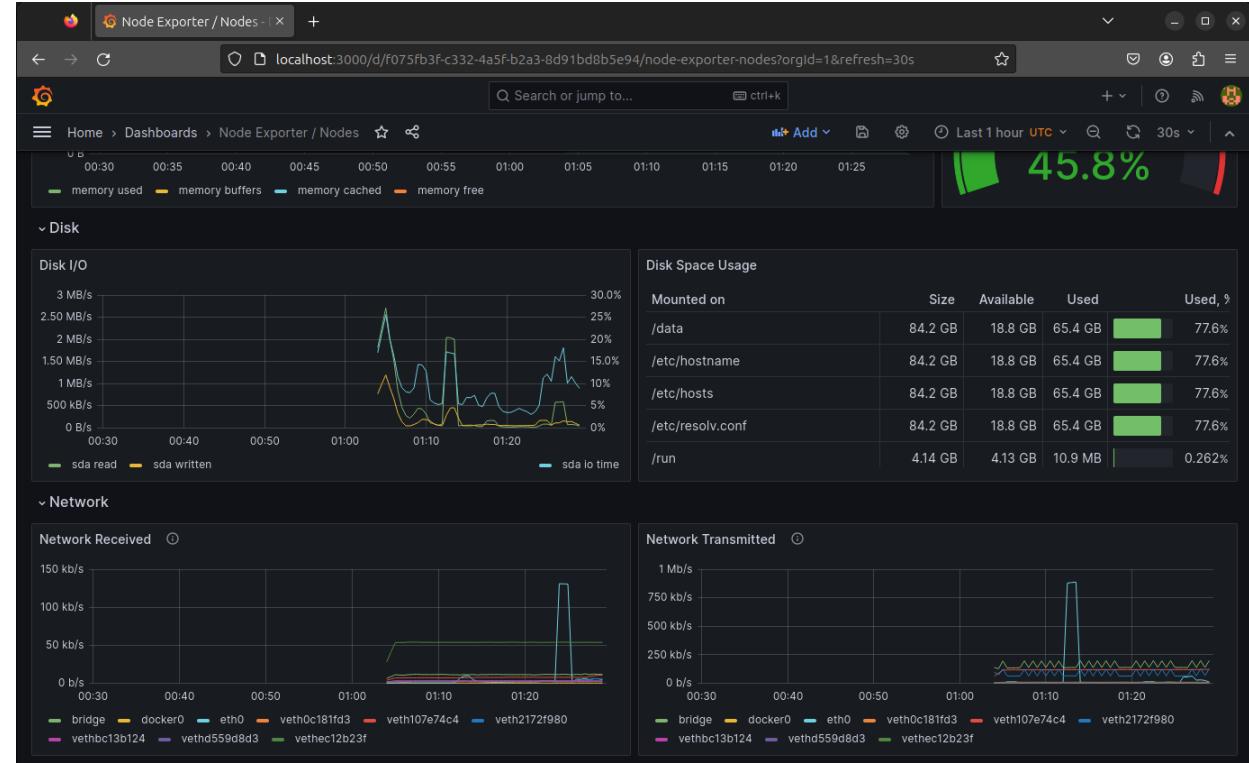
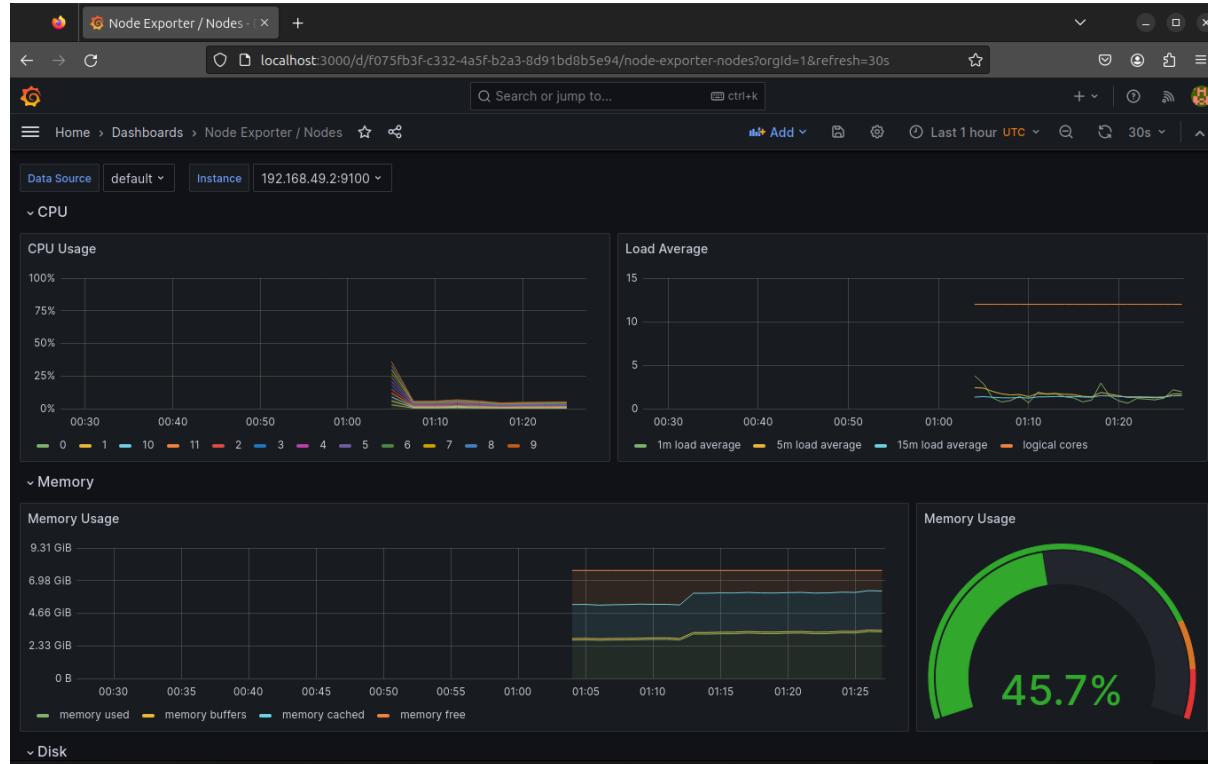
Dashboards
[Starred dashboards](#)

Latest from the blog
Jan 10

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