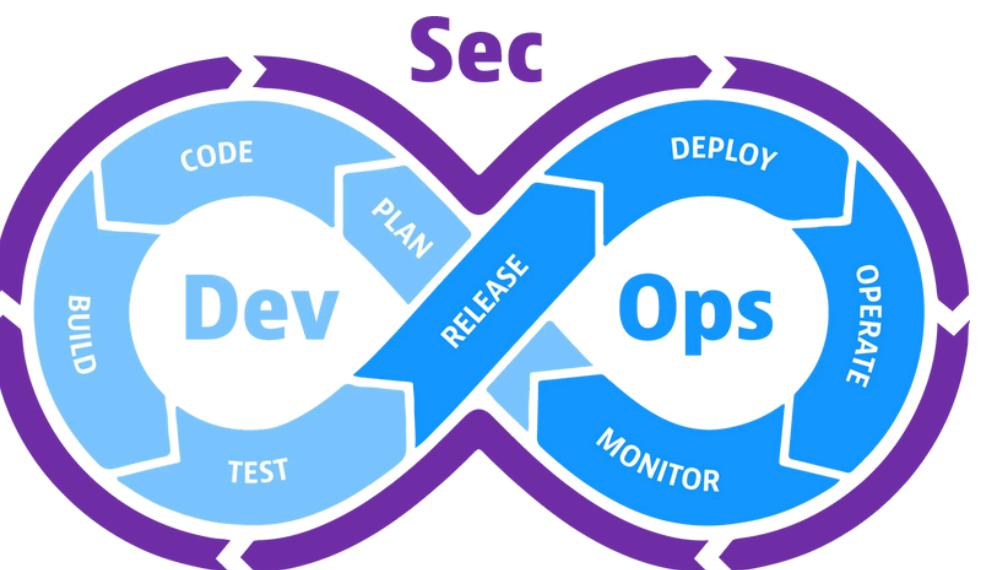


DEVSECOPS PROJECT



DONE BY:

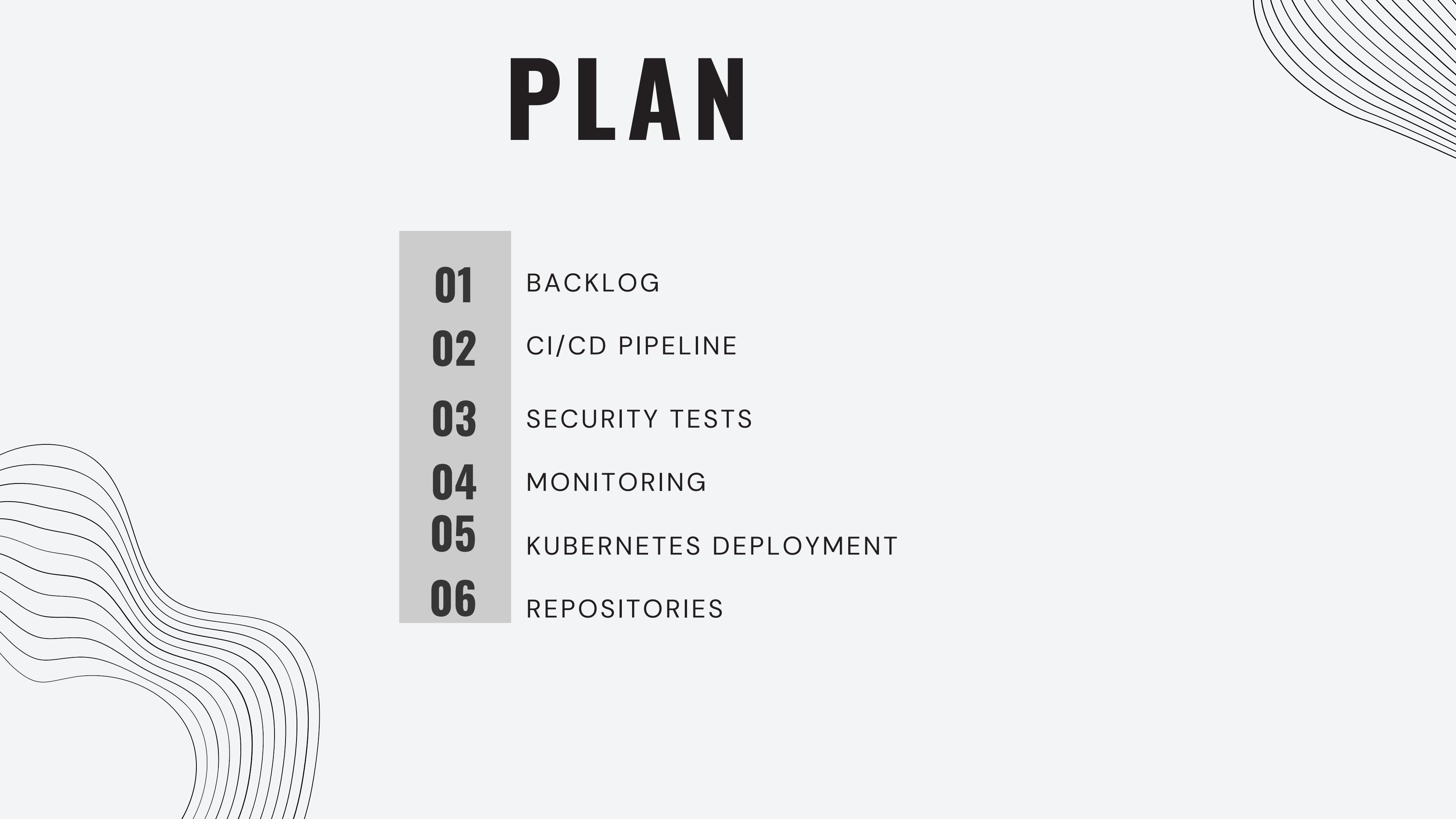
Amine Lachegur
Ait wahmane ELhoussaine
Ismail laraben
Abdellah Igajane
Abdelghafour Bouhdyd



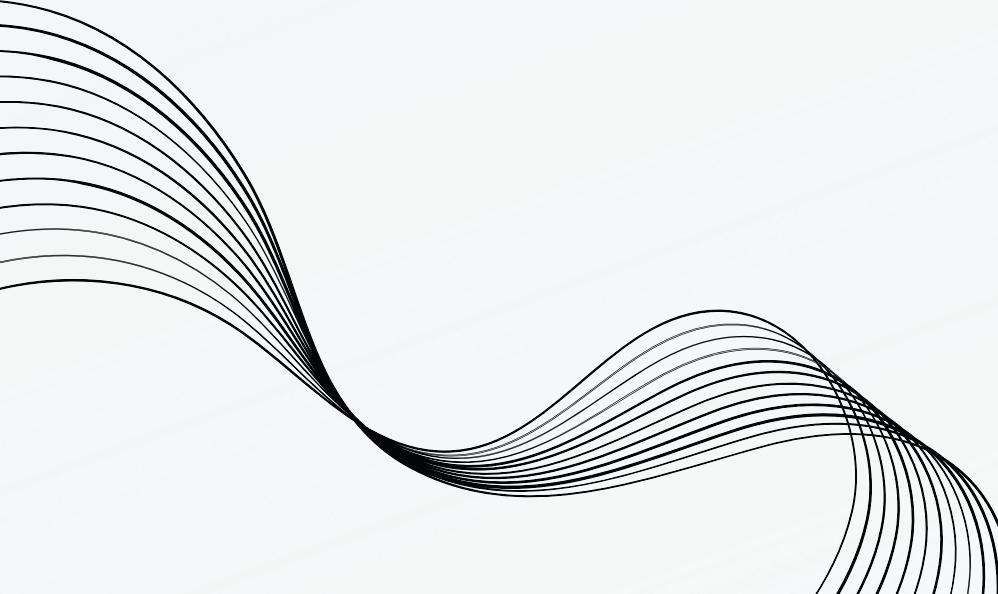
LED BY :

Mr. ALLAKI Driss

PLAN

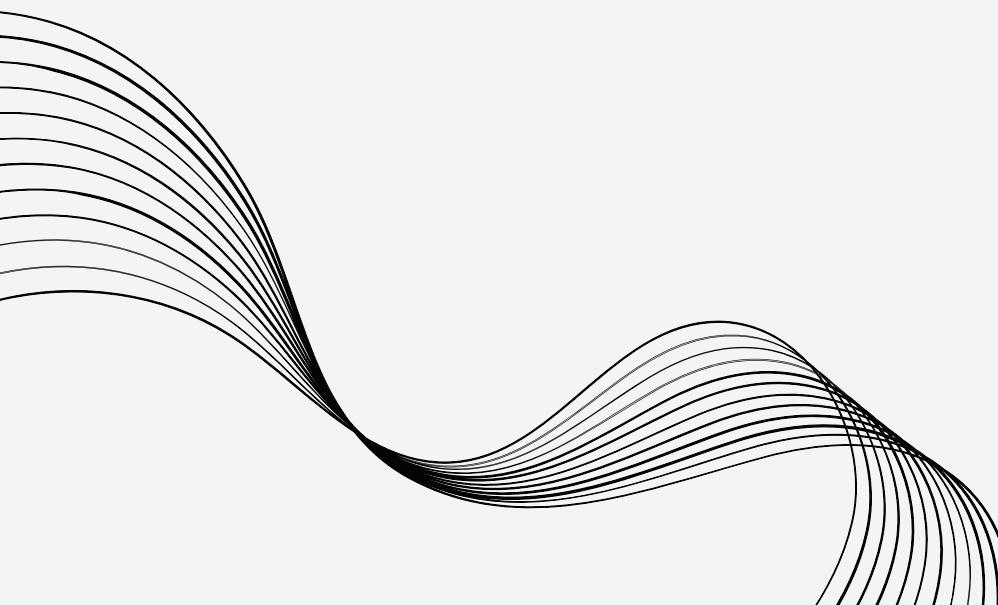
- 
- 01** BACKLOG
 - 02** CI/CD PIPELINE
 - 03** SECURITY TESTS
 - 04** MONITORING
 - 05** KUBERNETES DEPLOYMENT
 - 06** REPOSITORIES

1-BACKLOG



INTRODUCTION

In order to successfully carry out this project, a working methodology has been established within the team, utilizing the JIRA tool, specifically employing SCRUM boards for project management



Initially, the project was divided into four sprints

Projects / Projet DevSecOps

Backlog

The screenshot shows a Jira backlog interface for the 'Projet DevSecOps' project. The backlog is organized into four main items, each representing an epic:

- CI/CD pipeline**: Contains 3 issues. Status: 0 (grey), 0 (blue), 0 (green). Action buttons: 'Start sprint' (blue), '...'.
- Security Measures**: Contains 2 issues. Status: 0 (grey), 0 (blue), 0 (green). Action buttons: 'Start sprint' (grey), '...'.
- Monitoring**: Contains 3 issues. Status: 0 (grey), 0 (blue), 0 (green). Action buttons: 'Start sprint' (grey), '...'.
- Application Deployment**: Contains 3 issues. Status: 0 (grey), 0 (blue), 0 (green). Action buttons: 'Start sprint' (grey), '...'.

Filtering options at the top include a search bar, priority filters (AL, EW, IA, II, AB), an 'Epic' dropdown, and three icons for cloud, line chart, and filter.

Product Backlog

Sprint 1: CI/CD pipeline

Sprint 1 includes 3 tickets :

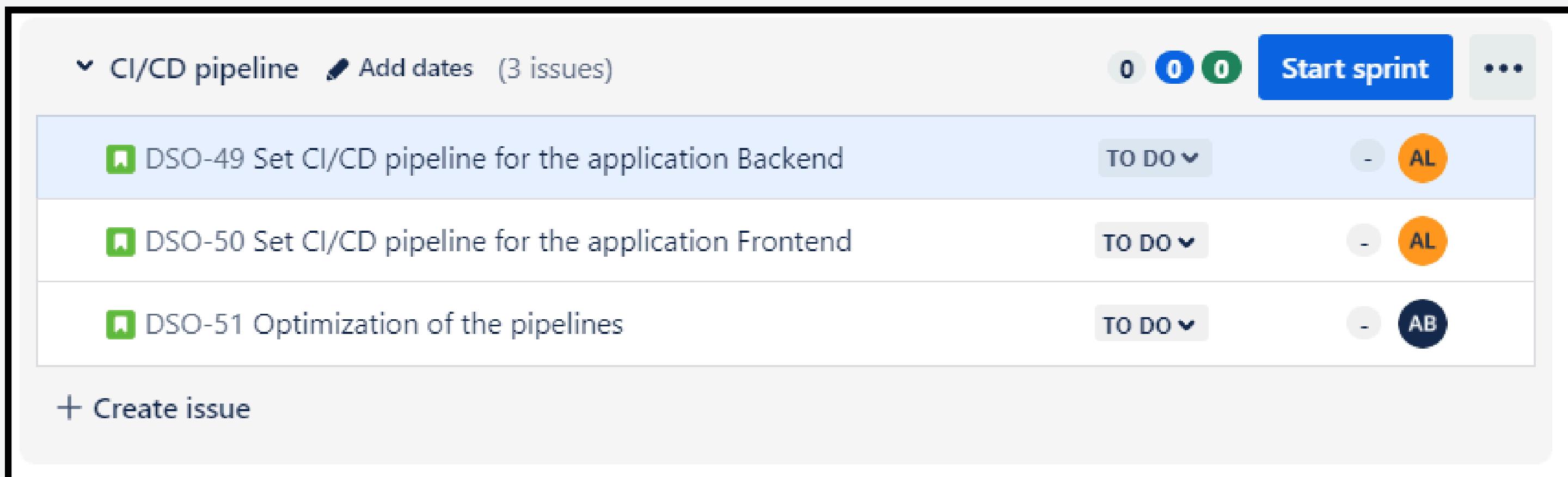
- Set CI/CD Pipeline for backend.
- Set CI/CD Pipeline for frontend.
- Pipelines optimizations.

▼ CI/CD pipeline ✍ Add dates (3 issues)

0 0 0 Start sprint ...

| | | | |
|--|-------|---|----|
| DSO-49 Set CI/CD pipeline for the application Backend | TO DO | - | AL |
| DSO-50 Set CI/CD pipeline for the application Frontend | TO DO | - | AL |
| DSO-51 Optimization of the pipelines | TO DO | - | AB |

+ Create issue



Ticket 1: Set CI/CD Pipeline for backend.

The screenshot shows a Jira ticket interface. At the top, there are buttons for 'Add epic' and a slash character, followed by a lock icon, a circular progress bar with the number '1', a thumbs-up icon, a share icon, three dots, and a close button. Below this, the ticket ID 'DSO-49' is displayed. The main title of the ticket is 'Set CI/CD pipeline for the application Backend'. Underneath the title are four small icons: a clipboard, a person, a gear, and three dots. Below these icons are two buttons: 'To Do' and 'Actions'. The 'Actions' button has a lightning bolt icon. The 'Description' section contains the following text: 'As a **Backend developer**, I want to set up a **CI/CD pipeline** for the application backend. This will automate the build, test, and deployment processes, ensuring efficient and reliable updates to the backend application.'

Add epic /

DSO-49

Set CI/CD pipeline for the application Backend

To Do Actions

Description

As a **Backend developer**, I want to set up a **CI/CD pipeline** for the application backend. This will automate the build, test, and deployment processes, ensuring efficient and reliable updates to the backend application.

Ticket 2 : Set CI/CD Pipeline for frontend.

The screenshot shows a Jira ticket card with the following details:

- Key:** Add epic / DSO-50
- Status:** Locked (blue lock icon)
- Assignee:** 1 (blue circle icon)
- Likes:** 1 (blue thumbs-up icon)
- Share:** Share icon
- More:** More options icon
- X:** Close button

Set CI/CD pipeline for the application Frontend

Buttons:

To Do Actions

Description

As a **front-end developer**, I want to set up a **CI/CD pipeline** for the application frontend. This will automate the build, test, and deployment processes, ensuring efficient and reliable updates to the frontend application.

Ticket 3 : Pipelines optimization.

[Add epic](#) / [DSO-51](#) [1](#) ...

Optimization of the pipelines

...

To Do Actions

Description

As a **DevOps engineer**, I want to **optimize the pipelines** for both backend and frontend applications to improve efficiency and reduce deployment time.

Sprint 2 : Security Measures

Sprint 2 includes 2 tickets :

- Include Pre-commit Hooks.
- Include automated security tests and scans.

▼ Security Measures ✍ Add dates (2 issues)

0 0 0 Start sprint ...

| | | |
|---|---------|------|
| DK DSO-32 Include Pre-Commit Hooks | TO DO ▾ | - II |
| DK DSO-52 Include automated security tests and scans | TO DO ▾ | - II |

+ Create issue

Ticket 1: Include Pre-commit Hooks

Add epic /
DSO-32

🔒 1 ⌂ ⌂ ... X

Include Pre-Commit Hooks

📎 🚧 🕋 ...

To Do Actions

Description

As a **security-conscious developer**, I want to include **pre-commit hooks in the CI/CD pipeline** for both backend and frontend applications. This will enforce security best practices and catch potential vulnerabilities before committing code changes.

Ticket 2 : Include automated security tests and scans

Add epic / DSO-52

Include automated security tests and scans

To Do Actions

Description

As a **security-conscious developer**, I want to include **automated security tests and scans** in the **CI/CD pipeline** for both the application backend and frontend. This will help identify and mitigate potential security vulnerabilities before deploying the applications to production.

Sprint 3 : Monitoring

Sprint 3 includes 3 tickets :

- System Monitoring.
- Application Monitoring.
- Log Monitoring.

▼ Monitoring ✍ Add dates (3 issues)

0 0 0 Start sprint ...

| | | | |
|--|-------|---|----|
| ⌚ DSO-30 System Monitoring | TO DO | - | EW |
| ⌚ DSO-45 Application Monitoring | TO DO | - | EW |
| ⌚ DSO-46 Log Monitoring | TO DO | - | EW |

+ Create issue

Ticket 1: System Monitoring

The image shows a Jira ticket interface for a ticket titled "System Monitoring". The ticket has the ID "DSO-30". At the top right, there are icons for locking, monitoring, thumbs up, sharing, and more. Below the title, there are four small action buttons: a clipboard, a person, a circular arrow, and three dots. Underneath these are two dropdown menus: "To Do" and "Actions". The "Description" section contains the following text:

As a **DevOps engineer**, I want to implement **system monitoring** for the application backend and frontend. This will allow us to proactively identify and address performance issues, errors, and anomalies, ensuring the stability and optimal functioning of the system.

Ticket 2 : Application Monitoring

The screenshot shows a Jira ticket interface. At the top left, there are buttons for 'Add epic' and a dropdown menu. To the right are icons for a lock (blue), a circular progress bar with the number '1', a thumbs up, a share symbol, three dots, and an 'X'. Below this, the ticket title 'DSO-45' is displayed. The main title of the ticket is 'Application Monitoring'. Underneath the title are four small icons: a clipboard, a network, a gear, and three dots. Below these are two dropdown menus: 'To Do' and 'Actions'. The 'Actions' menu has a lightning bolt icon. The 'Description' section contains the following text:

As a **DevOps engineer**, I want to implement **application monitoring** for the backend and frontend applications. This will allow us to proactively identify and address performance issues, errors, and anomalies specific to the applications, ensuring their optimal functioning and user experience.

Ticket 3 : Log Monitoring

The image shows a Jira ticket interface with the following details:

- Project:** Add epic / DSO-46
- Issue Type:** Lock (blue padlock icon)
- Assignee:** 1 (circle icon)
- Likes:** 1 (hand icon)
- Share:** Share icon
- More Options:** Three dots icon
- X:** Close button

Log Monitoring

Buttons:

To Do Actions

Description

As a **DevOps engineer**, I want to implement **log monitoring** for the backend and frontend applications. This will enable us to proactively identify and troubleshoot issues by monitoring and analyzing application logs, ensuring the stability and optimal functioning of the system.

Sprint 4 : Security Measures

Sprint 4 includes 2 tickets :

- Create Kubernetes cluster.
- Configuration of the cluster.
- Deployment of the Application's cluster.

The screenshot shows a Jira backlog for the 'Application Deployment' sprint. The backlog lists three issues:

- DSO-47 Create Kubernetes Cluster
- DSO-48 Configuration of Cluster
- DSO-53 Deployment of the Application Cluster

Each issue is currently in the 'TO DO' status and is categorized under 'IA' (Information Assurance). There are three empty circles at the top right of the backlog area, likely representing remaining story points or tasks. A 'Start sprint' button is also present.

Ticket 1: Create Kubernetes Cluster

[Add epic](#) / [DSO-47](#) [1](#) ...

Create Kubernetes Cluster

...

To Do Actions

Description

As a **DevOps engineer**, I want to create a **Kubernetes cluster** to efficiently manage and orchestrate containerized applications. This will enable us to leverage the scalability, resilience, and ease of deployment provided by Kubernetes.

Ticket 2 : Configuration of the cluster

The image shows a Jira ticket interface with the following details:

- Key:** Add epic / DSO-48
- Labels:** Lock icon, Circular 1 icon, Like icon, Share icon, More options icon, Close icon.
- Title:** Configuration of Cluster
- Buttons:** Attach, Version, Link, More options.
- Status:** To Do
- Actions:** Actions dropdown.
- Description:** As a **DevOps engineer**, I want to **configure** an existing **Kubernetes cluster** to ensure it meets the requirements of our applications and infrastructure. This will enable us to optimize the cluster's performance, security, and functionality.

Ticket 3 : Deployment of the Application's cluster

Add epic / DSO-53

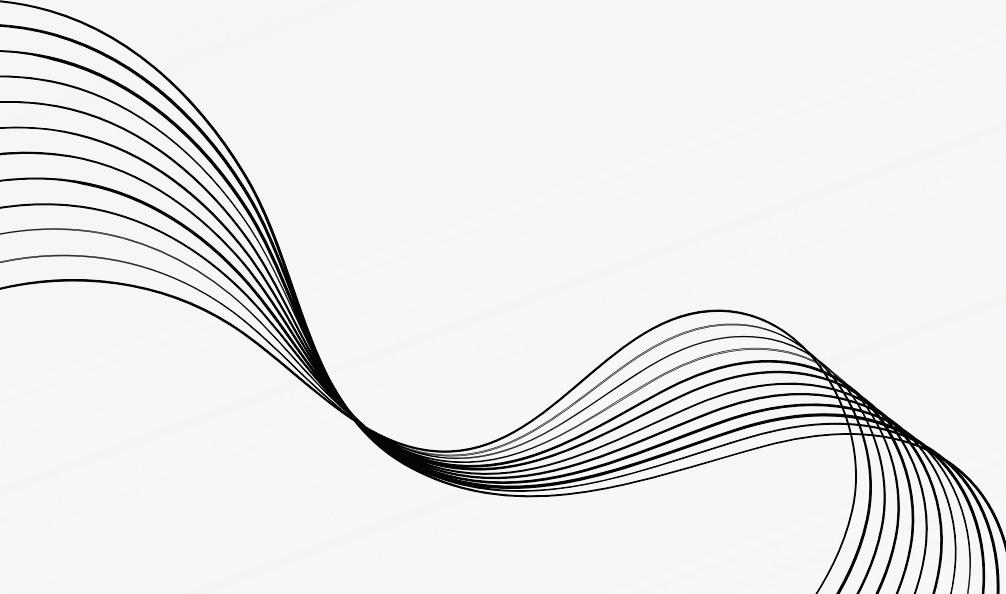
Deployment of the Application Cluster

To Do Actions

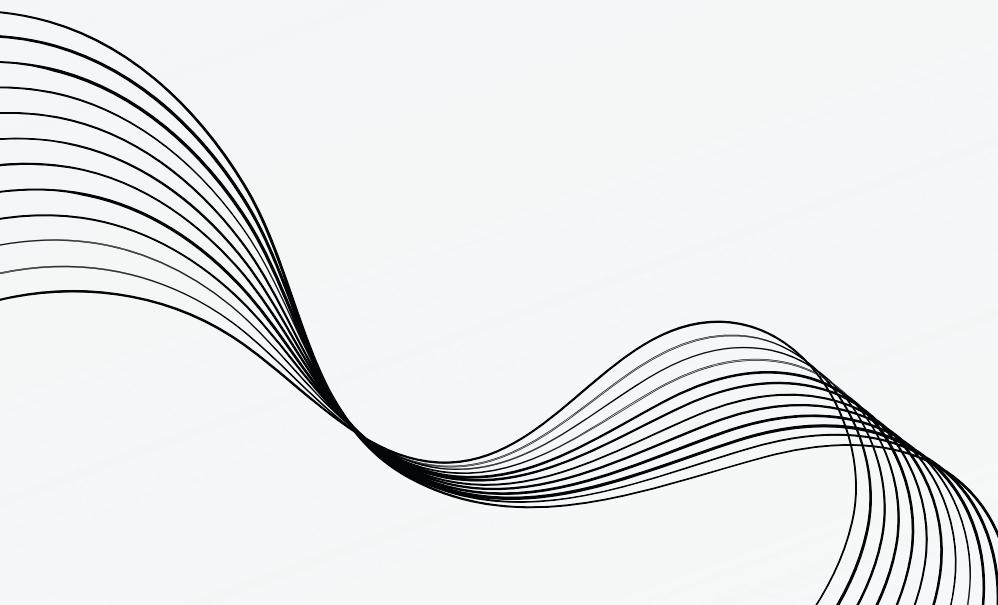
Description

As a **DevOps engineer**, I want to **deploy our application cluster** to the **Kubernetes** environment. This will enable us to leverage the benefits of containerization and orchestration, ensuring the efficient deployment and management of our applications.

2-CI/CD PIPELINE

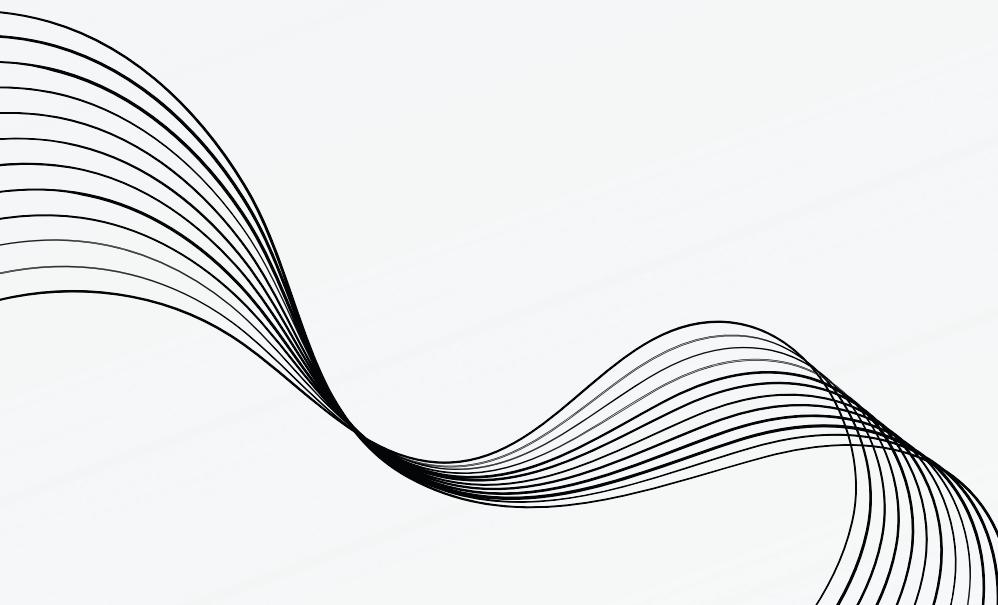


GitLab proved to be the ideal choice for our CI/CD needs due to its comprehensive set of features and seamless integration with source code management. By utilizing GitLab, we were able to streamline our software delivery process, improve collaboration among team members, and achieve faster and more reliable deployments.



1) Backend CI/CD pipeline

First of all , we have to create a **Dockerfile** to define the instructions and configuration needed to build a **Docker image**.



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 8080
CMD ["java", "-jar", "spring-boot-app.jar"]
```

The Dockerfile has 5 numbered lines. The code editor interface includes a file icon, a file size of 171B, and buttons for Blame, Edit, Replace, Delete, and download.

1) Backend CI/CD pipeline

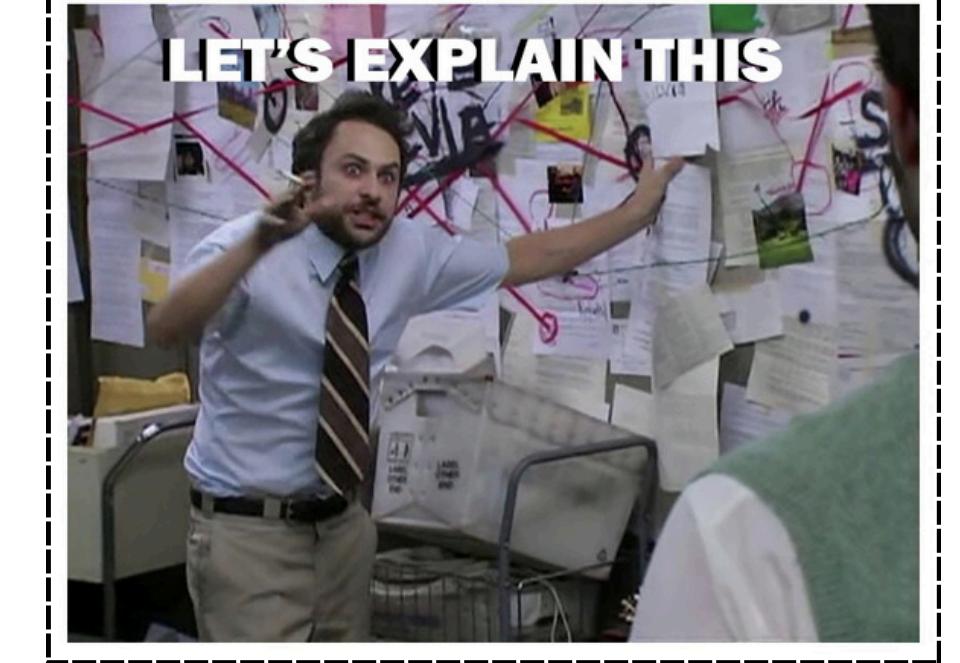
Let's explain the content of the **Dockerfile** :

- **Line 1** : specifies the base image for the Docker image. In this case, it uses the OpenJDK 17 image.
- **Line 2** : sets the working directory inside the Docker image to /app. This is the directory where subsequent commands will be executed.
- **Line 3** : copies the JAR file from the target directory of the host machine into the /app directory of the Docker image.
- **Line 4** : This line specifies that the Docker container will listen on port 8080.
- **Line 5** : defines the command that will be executed when the Docker container starts.

1) Backend CI/CD pipeline

also , we have to configure our pipeline using the YAML file :

```
🔥 .gitlab-ci.yml 882 B
Blame Edit Replace Delete
1 image: docker:latest
2 services:
3   - docker:dind
4
5 stages:
6   - build
7   - package
8   - deploy
9
10 maven-build:
11   image: maven:latest
12   stage: build
13   script: "mvn clean package -B"
14   artifacts:
15     paths:
16       - target/*.jar
17
18 docker-build:
19
20   stage: package
21   rules:
22     - exists:
23       - Dockerfile
24   before_script:
25     - docker login -u 4bd3lgh4f0r -p $DOCKER_HUB_PASSWORD
26   script:
27     - docker build -t 4bd3lgh4f0r/devsecops_backend .
28     - docker push 4bd3lgh4f0r/devsecops_backend
29
30 docker-deploy:
31   stage: deploy
32   before_script:
33     - apk add --update curl && rm -rf /var/cach/apk/*
34   script:
35     - docker network create docker_net
36
37     - docker run -d --name mysql_name --network docker_net -e MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD -e MYSQL_DATABASE=testdb mysql
38
39     - docker run -d --name backend_sb --network docker_net -p 8080:8080 4bd3lgh4f0r/devsecops_backend:1.0
```



1) Backend CI/CD pipeline

Here are some key values used on the YAML file:

- The image directive specifies the Docker image that will be used as the base environment for the pipeline jobs. In this case, it is set to **docker:latest**.
- The services section defines additional services that will be available during the pipeline execution. Here, **docker:dind** service is specified, which provides a Docker-in-Docker environment.
- The stages section lists the different stages of the pipeline, which are :
 - **Build** : responsible for compiling and packaging the project.
 - **Package** : responsible for the creation and preparation of the Docker image.
 - **Deploy** : focuses on the deployment of the application.
- The YAML file defines 3 jobs :
 - **maven-build**
 - **docker-build**
 - **docker-deploy**

1) Backend CI/CD pipeline

Now let's run the backend pipeline

4bd3lgh4f0r / devsecops_backend / Pipelines / #1129505100

Update .gitlab-ci.yml file

Passed 4bd3lgh4f0r created pipeline for commit 0c6fd047 finished 1 day ago

For main

latest 3 Jobs 3.24 3 minutes 14 seconds, queued for 1 seconds

DONE

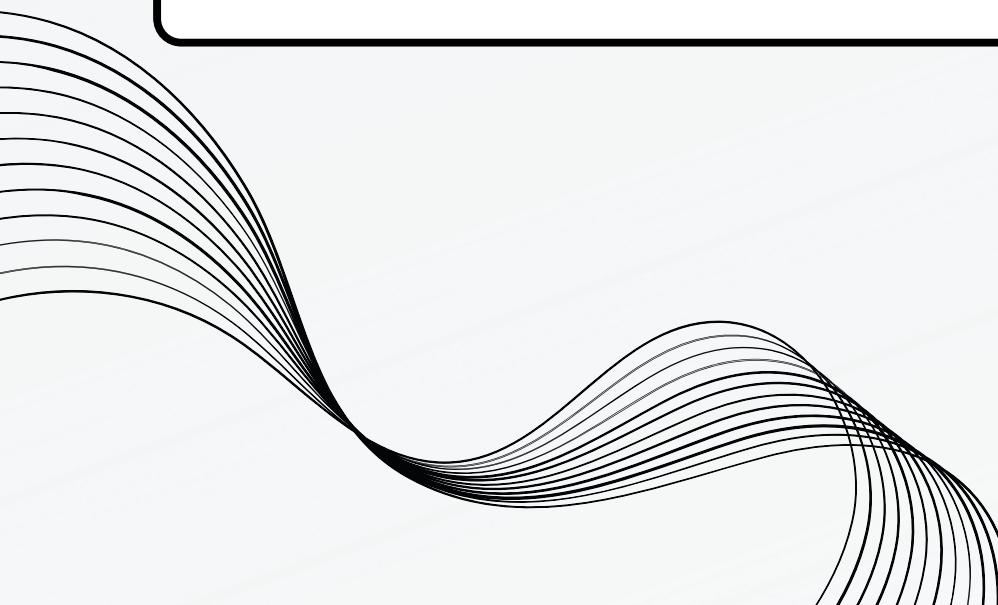
Pipeline Needs Jobs 3 Tests 0

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

The pipeline ran successfully

2) Frontend CI/CD pipeline

let's create a **Dockerfile** to define the instructions and configuration needed to build a **Docker image**.



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

```
1 FROM node:latest
2
3 WORKDIR /app
4 ENV PATH /app/node_modules/.bin:$PATH
5
6 COPY package.json /app/package.json
7 RUN npm install
8 RUN npm install -g @angular/cli@7.3.9
9
10 COPY . /app
11
12 CMD ng serve --host 0.0.0.0 --port 4200
```

The Dockerfile is 215 B in size. The interface includes standard GitHub-style buttons for Blame, Edit, Replace, and Delete, along with download icons.

2) Frontend CI/CD pipeline

Let's explain the content of the **Dockerfile** :

- **Line 1** : specifies the base image for the Docker image. In this case, it uses the latest Node.js image.
- **Line 3** : sets the working directory inside the Docker image to /app. This is the directory where subsequent commands will be executed.
- **Line 4** : sets an environment variable PATH to allow access to locally installed Node.js binaries within the project.
- **Line 6** : copies the **package.json** file from the host machine to the /app directory in the Docker image.
- **Line 7** : runs the **npm install** command inside the Docker image installs all the required packages.
- **Line 8** : installs the Angular CLI globally in the Docker image.
- **Line 10** : copies the remaining project files and directories from the host machine to the /app directory in the Docker image.
- **Line 11** : uses the **ng serve** command to start the Angular development server.

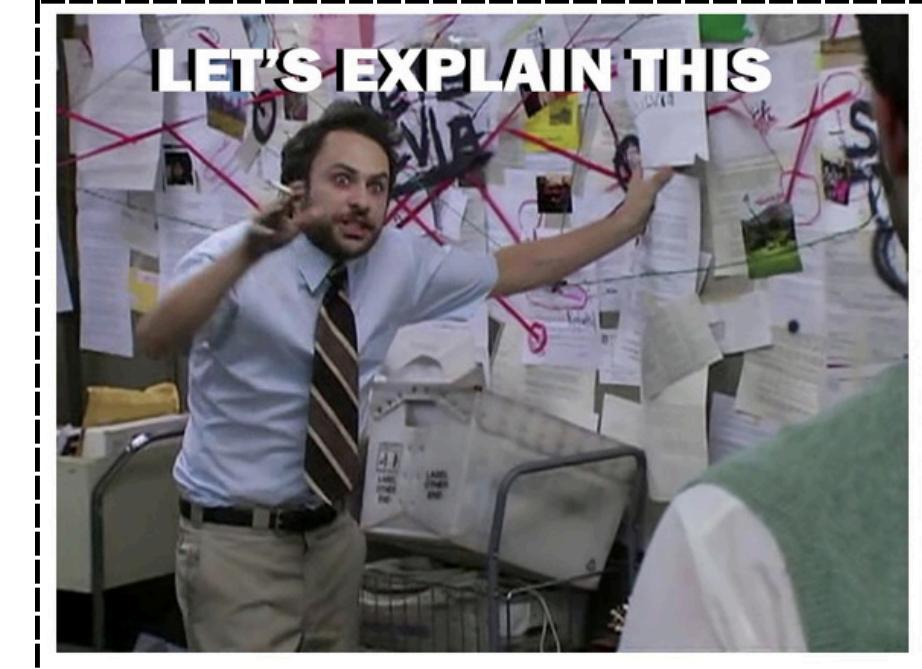
2) Frontend CI/CD pipeline

Now we configure the pipeline file :

.gitlab-ci.yml 723 B

Blame Edit ▾

```
1 image : docker:latest
2 include:
3   - template: Security/Dependency-Scanning.gitlab-ci.yml
4   - template: Security/SAST.gitlab-ci.yml
5 services:
6   - docker:dind
7 stages:
8   - test
9   - package
10  - deploy
11 docker-build:
12   stage: package
13   before_script:
14     - docker login -u 4bd3lgh4f0r -p $DOCKER_USER_PASSWORD
15   script:
16     - docker build -t 4bd3lgh4f0r/devsecops_frontend .
17     - docker push 4bd3lgh4f0r/devsecops_frontend
18
19 docker-deploy:
20   stage: deploy
21   before_script :
22     - apk add --update curl && rm -rf /var/cache/apk/*
23   script :
24     - docker run -d -p 9001:4200 --name frontend 4bd3lgh4f0r/devsecops_frontend
25     - sleep 30
26     - docker logs frontend
27     - curl -v "http://docker:9001"
28     docker logs frontend
```



2) Frontend CI/CD pipeline

Here are some key values used on the YAML file:

- The `image` directive specifies the Docker image that will be used as the base environment for the pipeline jobs. In this case, it is set to **`docker:latest`**.
- The `services` section defines the service **`docker:dind`**, which provides a Docker-in-Docker environment to be used during the pipeline execution.
- The `stages` section lists the different stages of the pipeline, which are :
 - ***Test***
 - ***Package***
 - ***Deploy***
- The YAML file defines 3 jobs :
 - **`docker-build`** : responsible for building and pushing a Docker image to Dockerhub.
 - **`docker-deploy`** : responsible for deploying the Docker image to the target environment.

2) Frontend CI/CD pipeline

Now let's run the frontend pipeline

4bd3lgh4f0r / devsecops_frontend / Pipelines / #1119011102

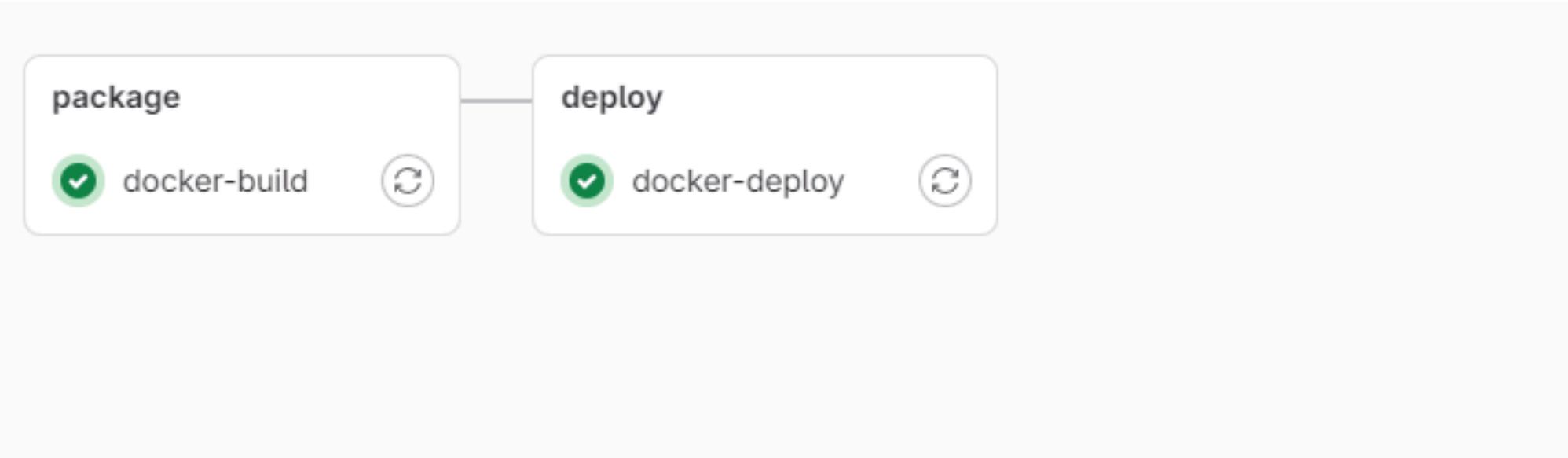
Update .gitlab-ci.yml

Passed 4bd3lgh4f0r created pipeline for commit 6cf325a8 finished 1 week ago

For main

2 Jobs 4.39 4 minutes 23 seconds, queued for 0 seconds

Pipeline Needs Jobs 2 Tests 0



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

The diagram shows a CI pipeline with two stages: 'package' and 'deploy'. The 'package' stage contains a job named 'docker-build'. The 'deploy' stage contains a job named 'docker-deploy'. Arrows indicate a sequential flow from the package stage to the deploy stage.

DONE



The pipeline ran successfully

3) Deployment of the images:

As we can see , the 2 images were successfully pushed to **DockerHub**

 4bd3lgh4f0r / devsecops_backend

Description
This repository does not have a description 

 Last pushed: a few seconds ago

Tags
This repository contains 2 tag(s).

| Tag | OS | Type | Pulled | Pushed |
|--|---|-------|--------|-------------------|
|  latest |  | Image | -- | a few seconds ago |

 4bd3lgh4f0r / devsecops_frontend

Description
This repository does not have a description 

 Last pushed: 12 hours ago

Tags
This repository contains 1 tag(s).

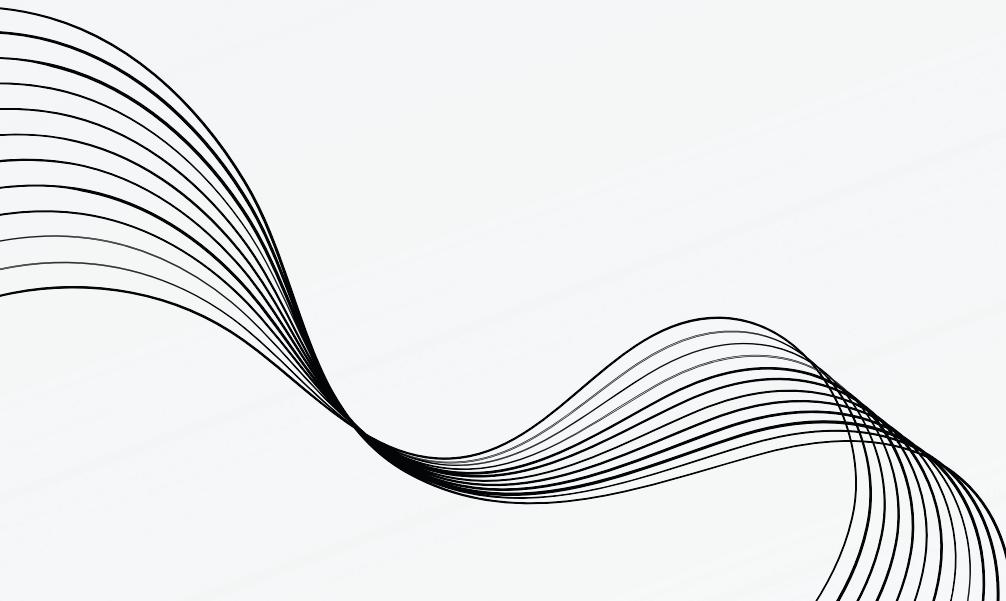
| Tag | OS | Type | Pulled | Pushed |
|--|---|-------|-------------|--------------|
|  latest |  | Image | 3 hours ago | 12 hours ago |



PIPELINES OPTIMIZATION

After establishing the two pipelines, we will endeavor to optimize them.

But first we will add some rules to the pipelines



RULES

Refers to a section in the CI/CD configuration file where you define conditions that determine when a particular job should be executed



These are the rules that we have used in the 2 pipelines

the building stage

```
docker-build:  
  stage: build  
  rules:  
    - exists:  
      - Dockerfile  
    - exists:  
      - $CI_COMMIT_BRANCH == "main"
```

Concerning the docker-build job we have created 2 rules :

the first : this rule ensures that the associated job is executed only when the "Dockerfile" is present in the project

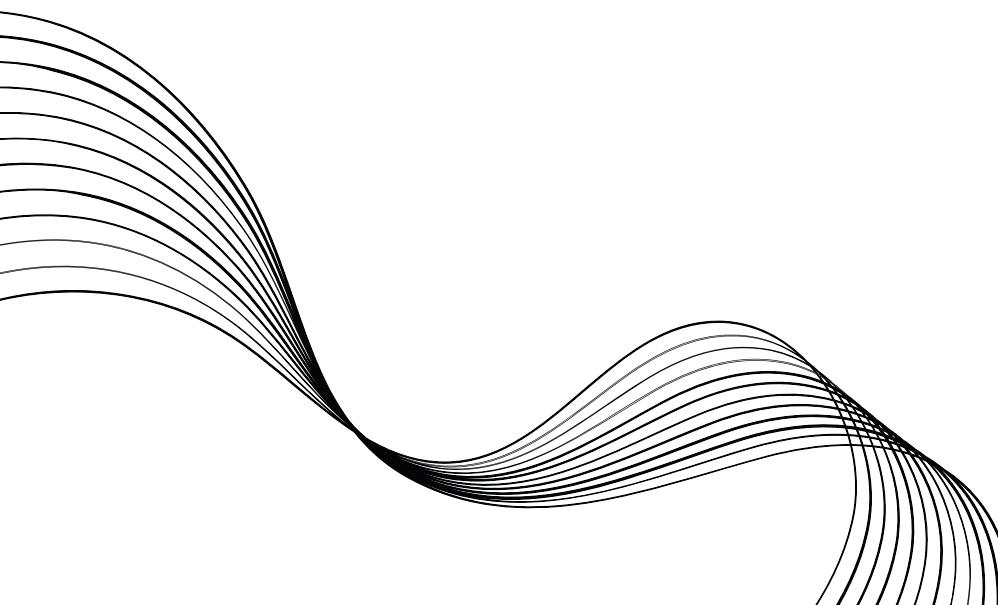
the second : the job will be executed only on main branch

```
docker-deploy:  
  stage: deploy  
  rules:  
    - needs:  
      - job: "docker-build"  
    - exists:  
      - $CI_COMMIT_BRANCH == "main"
```

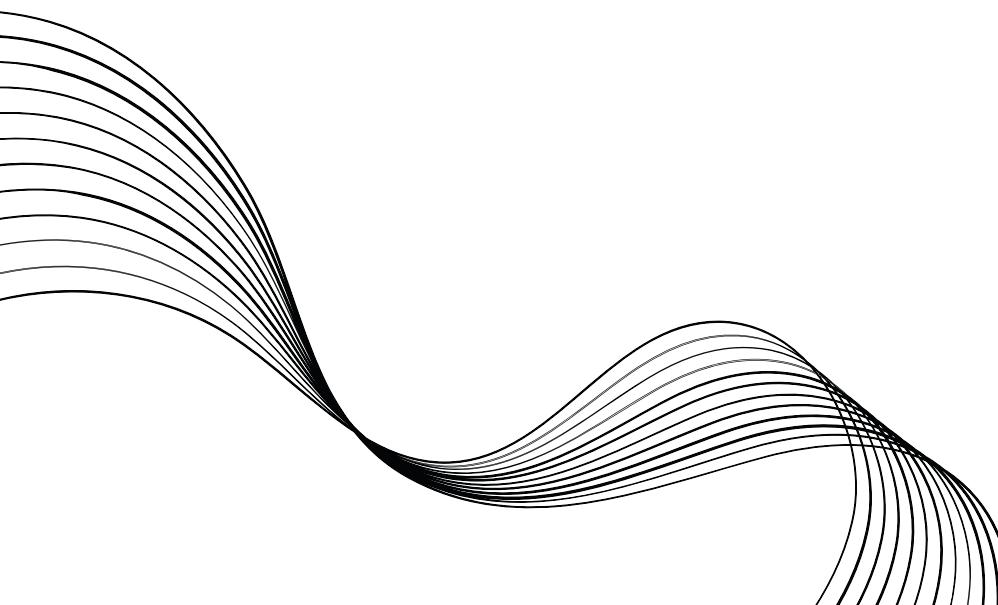
Concerning the docker-deploy job we have created 2 rules :
the first : ensures that the deployment occurs only after the successful completion of the Docker image build
the second : the job will be executed only on main branch

After adding some rules, we will Enhance GitLab CI/CD by adjusting **generic Jobs**
Store job configurations in an external YAML file for easy management and reuse in
shared job definitions.

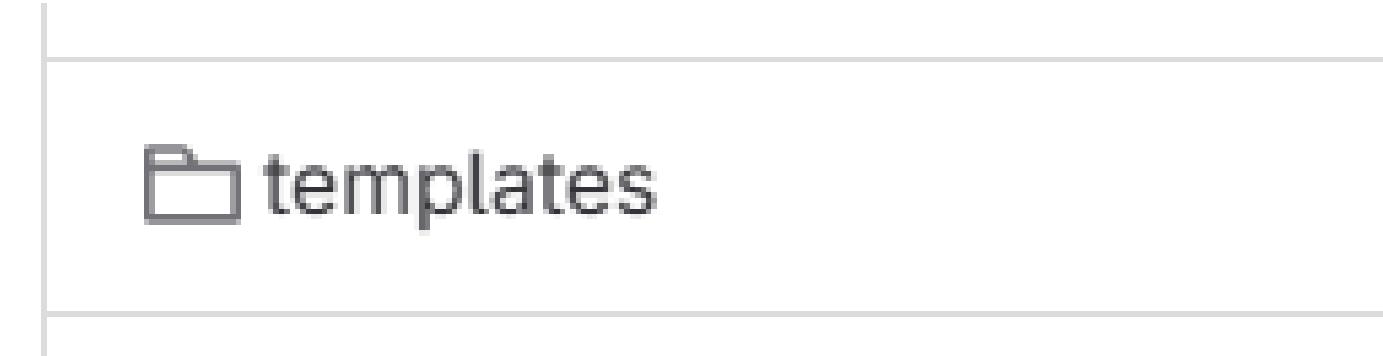
what are generic jobs



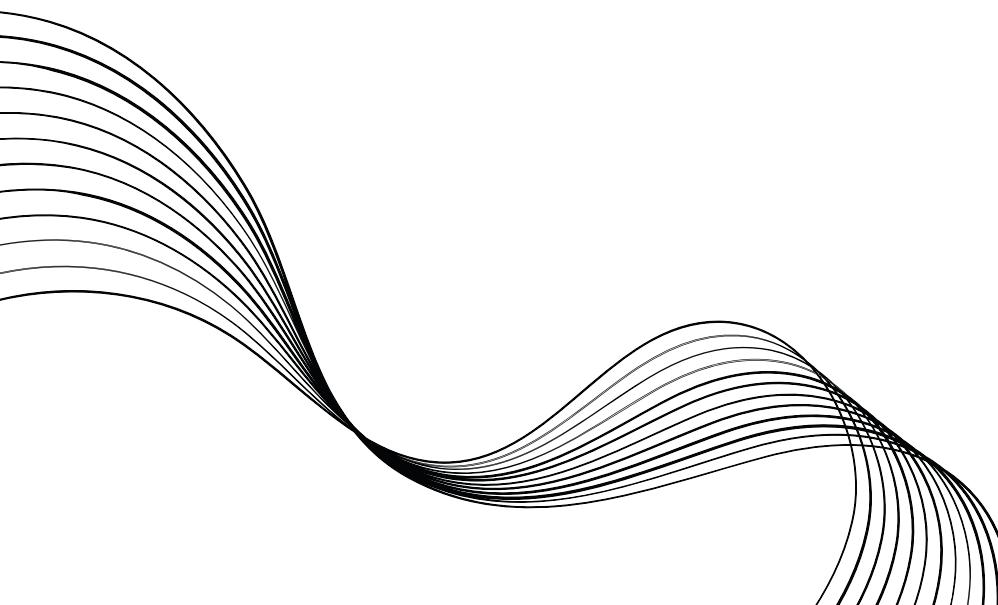
generic jobs is about storing job configurations in an external YAML file for easy management and reuse in shared job definitions.



First we create a directory named “templates” where we will store YAML files



then we create the YAML files for each job :



for the frontend:

docker-build.yml 430 B

```
1 docker-build:
2   stage: build
3   rules:
4     - exists:
5       - Dockerfile
6     - exists:
7       - $CI_COMMIT_BRANCH == "main"
8   before_script:
9     - docker login -u 4bd3lgh4f0r -p $DOCKER_USER_PASSWORD
10  script:
11    - docker pull 4bd3lgh4f0r/devsecops_frontend:latest || true
12    - docker build --cache-from 4bd3lgh4f0r/devsecops_frontend:latest -t 4bd3lgh4f0r/devsecops_frontend .
13    - docker push 4bd3lgh4f0r/devsecops_frontend
14
```

docker-deploy.yml 728 B

```
1 docker-deploy:
2   stage: deploy
3   rules:
4     - needs:
5       - job: "docker-build"
6     - exists:
7       - $CI_COMMIT_BRANCH == "main"
8   before_script:
9     - apk add --update curl && rm -rf /var/cache/apk/*
10  script:
11    - docker run -d -p 9001:4200 --name frontend 4bd3lgh4f0r/devsecops_frontend
12    - sleep 30
13    - docker logs frontend
14    - curl -v "http://docker:9001"
15    - docker logs frontend
```

then we Specify the paths for each job in the YAML pipeline file:

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

for the backend:

 docker-package.yml  130 B

```
1 maven-build:  
2   image: maven:latest  
3   stage: package  
4   script: "mvn clean package -B"  
5   artifacts:  
6     paths:  
7       - target/*.jar
```

 docker-deploy.yml  493 B

```
1 docker-deploy:  
2   stage: deploy  
3   rules:  
4     - needs:  
5       - job: "docker-build"  
6     - exists:  
7       - $CI_COMMIT_BRANCH == "main"  
8   before_script:  
9     - apk add --update curl && rm -rf /var/cach/apk/*  
10  script:  
11    - docker network create docker_net  
12    - docker run -d --name mysql_name --network docker_net -e MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD -  
13    - docker run -d --name backend_sb --network docker_net -p 8080:8080 4bd3lgh4f0r/devsecops_backend  
14  
15  
16
```

 docker-build.yml  313 B

```
1 docker-build:  
2  
3   stage: build  
4   rules:  
5     - exists:  
6       - Dockerfile  
7     - exists:  
8       - $CI_COMMIT_BRANCH == "main"  
9   before_script:  
10  - docker login -u 4bd3lgh4f0r -p $DOCKER_HUB_PASSWORD  
11  script:  
12  - docker build -t 4bd3lgh4f0r/devsecops_backend .  
13  - docker push 4bd3lgh4f0r/devsecops_backend  
14
```

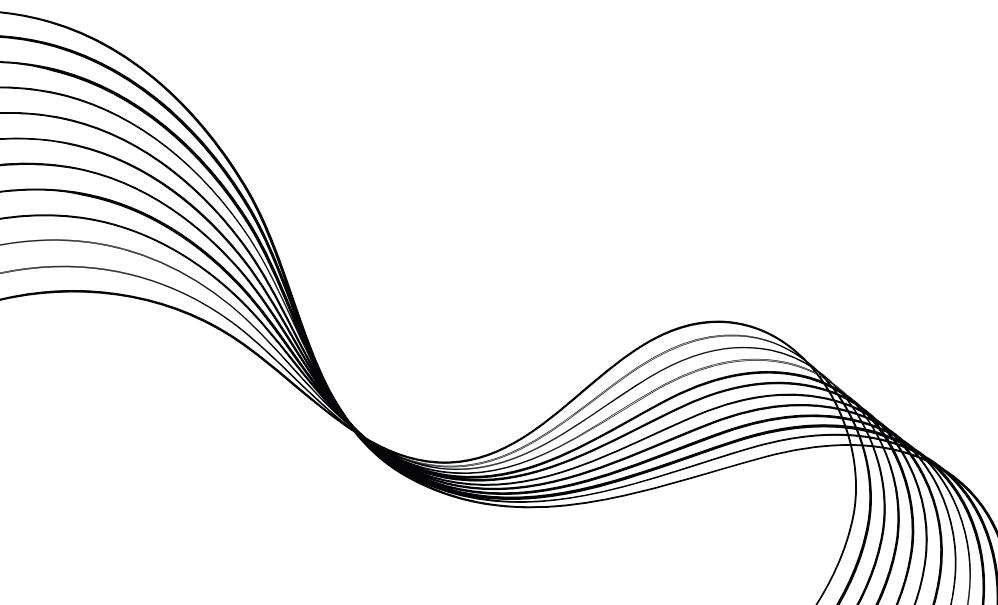
the paths :

- templates/docker-package.yml
- templates/docker-build.yml
- templates/docker-deploy.yml

GITLAB-RUNNER

GitLab Runner is the component responsible for running the jobs defined in a GitLab CI/CD pipeline. It can be installed on various platforms, such as Linux, macOS, and Windows. Runners can be shared among projects or dedicated to a specific project

in this task our goal is to configure our GitLab Runner to enhance the efficiency of job execution.



Initially, we select the operating system, and in our case , we are opting for Linux

Platform

Operating systems

 Linux  macOS  Windows

Containers

 Docker  Kubernetes

Tags

Tags

Add tags to specify jobs that the runner can run. [Learn more.](#)

`my_runner`

Separate multiple tags with a comma. For example, `macos, shared`.

Run untagged jobs
Use the runner for jobs without tags in addition to tagged jobs.

Here is the relevant information pertaining to our GitLab Runner configuration :

Register runner

GitLab Runner must be installed before you can register a runner. [How do I install GitLab Runner?](#)

Step 1

Copy and paste the following command into your command line to register the runner.

```
$ gitlab-runner register  
--url https://gitlab.com  
--token glrt-gz4T6tdZQ1t9Y2g3pA3n
```

 The runner authentication token `glrt-gz4T6tdZQ1t9Y2g3pA3n`  displays here for a short time only. After you register the runner, this token is stored in the `config.toml` and cannot be accessed again from the UI.

Step 2

Choose an executor when prompted by the command line. Executors run builds in different environments. [Not sure which one to select?](#) 

Step 3 (optional)

Manually verify that the runner is available to pick up jobs.

```
$ gitlab-runner run
```

This may not be needed if you manage your runner as a [system or user service](#) .

[Go to runners page](#)

now in the terminal of the virtual machine (we have used an ubuntu VM)
we register our runner :

```
abdelghafour@abdelghafour:~$ gitlab-runner register
Runtime platform          arch=amd64 os=linux pid=4552 revision=102c81ba version=16.7.0
WARNING: Running in user-mode.
WARNING: The user-mode requires you to manually start builds processing:
WARNING: $ gitlab-runner run
WARNING: Use sudo for system-mode:
WARNING: $ sudo gitlab-runner...
Enter the GitLab instance URL (for example, https://gitlab.com/):
https://gitlab.com
Enter the registration token:
glrt-gz4T6tdZQ1t9Y2g3pA3n
Verifying runner... is valid           runner=gz4T6tdZQ
Enter a name for the runner. This is stored only in the local config.toml file:
[abdelghafour]: runner
Enter an executor: docker-windows, ssh, kubernetes, docker-autoscaler, docker+machine, instance, custom, docker, parallels, shell
virtualbox:
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 "/home/abdelghafour/.gitlab-runner/config.toml"
abdelghafour@abdelghafour:~$
```

after registering the runner we add the following lines into the YAML file :

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

then we disable the shared runner

Shared runners

These runners are available to all groups and projects.

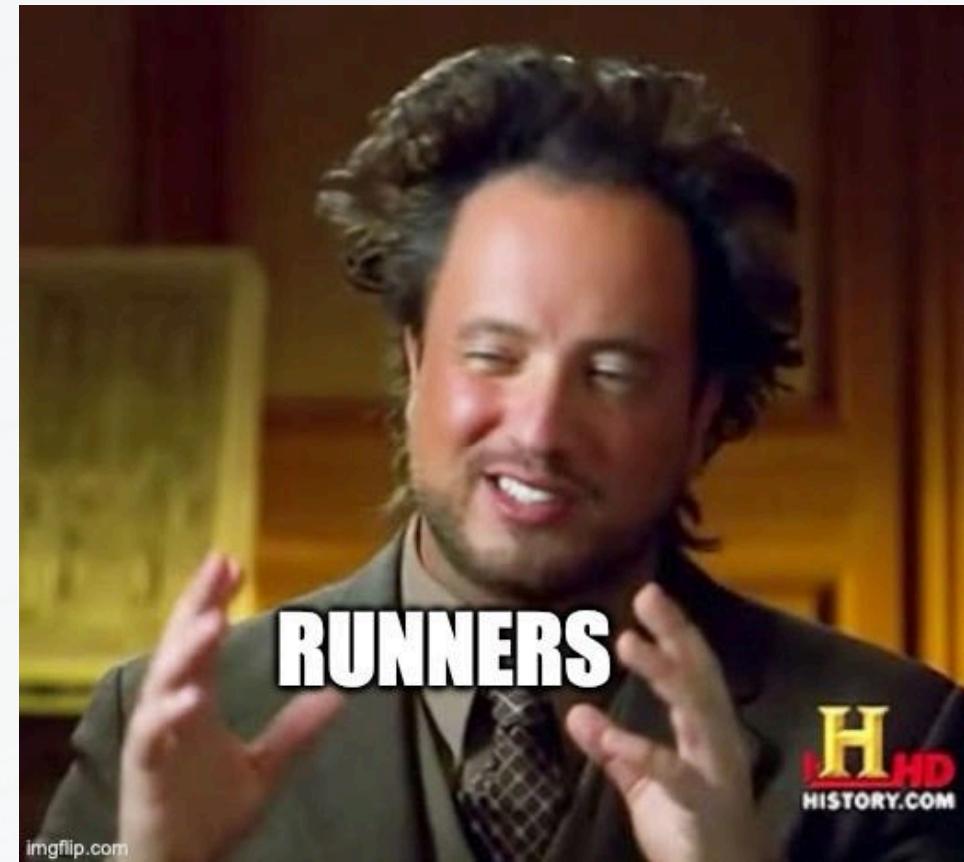
Each CI/CD job runs on a separate, isolated virtual machine.

Enable shared runners for this project

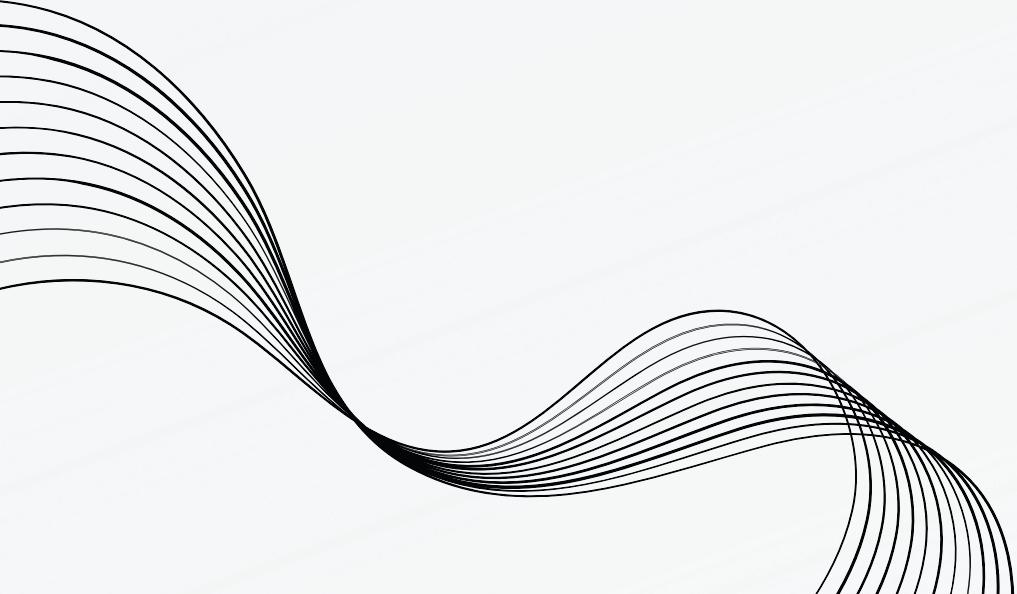


and finally we start the runner :

```
abdelghafour@abdelghafour:~$ gitlab-runner run
Runtime platform                                arch=amd64 os=linux pid=4562 revision=102c81ba version=16.7.0
Starting multi-runner from /home/abdelghafour/.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
```



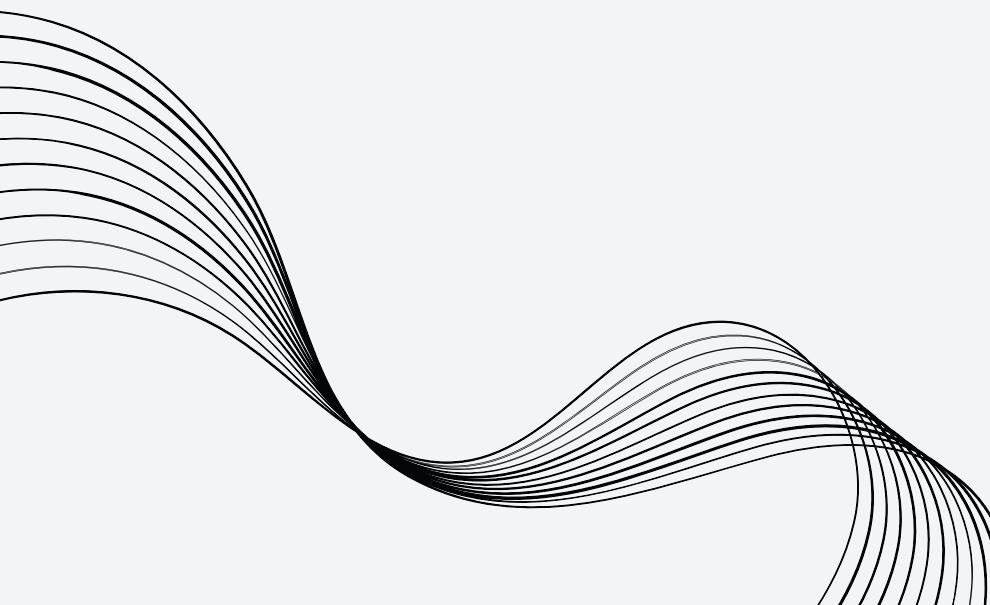
3-Security Tests



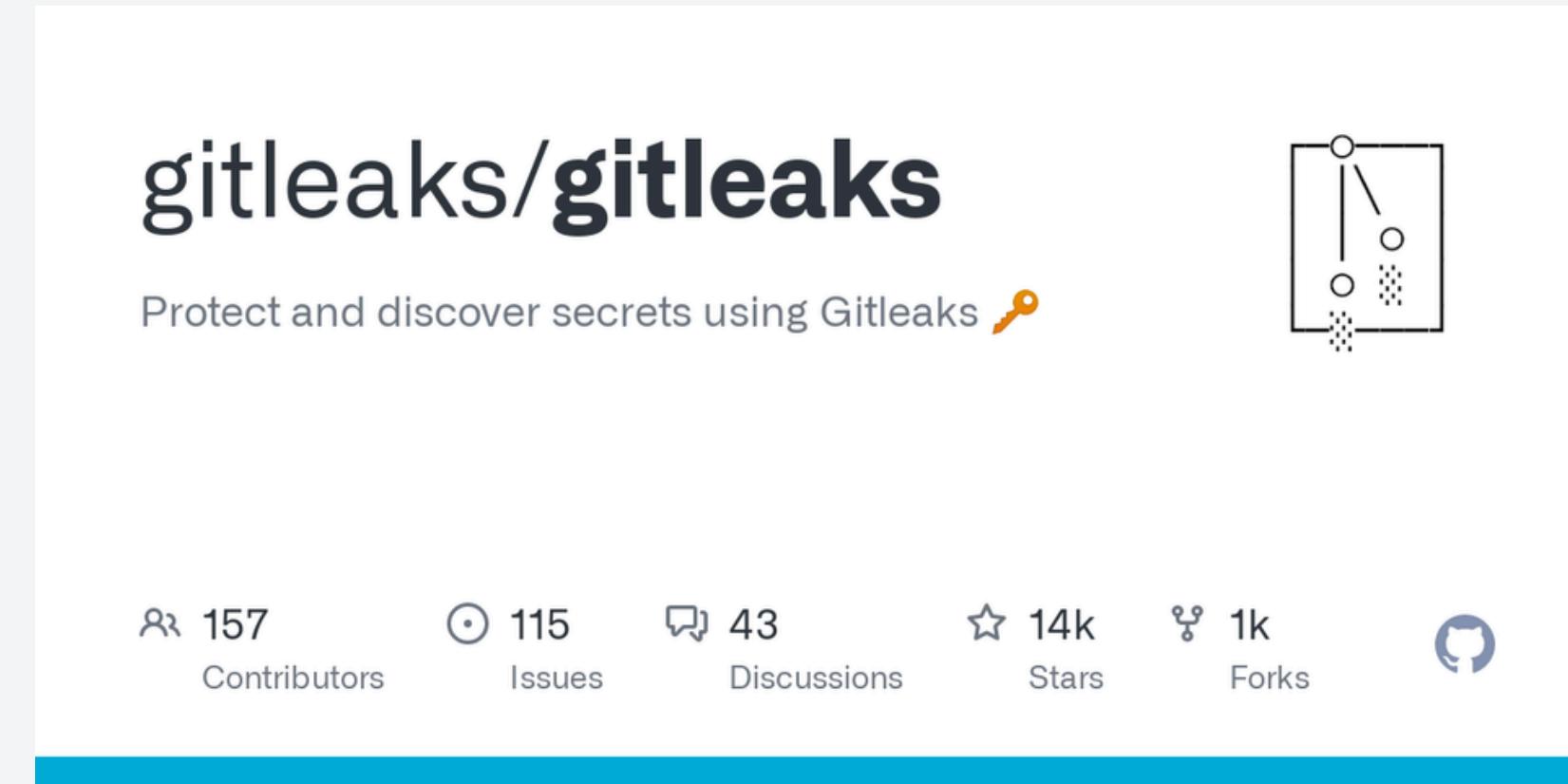
Pre-commit Hooks

Pre-commit hooks are scripts or actions that are executed automatically before a developer's changes are committed to the version control system (e.g., Git). The purpose of pre-commit hooks is to catch issues early in the development process and prevent code that doesn't meet certain criteria from being committed.

In our project, we will use pre-commit hooks to detect if hard coded secrets (passwords, credentials, api keys...) exist in our code, so we remove them before they are committed publicly.

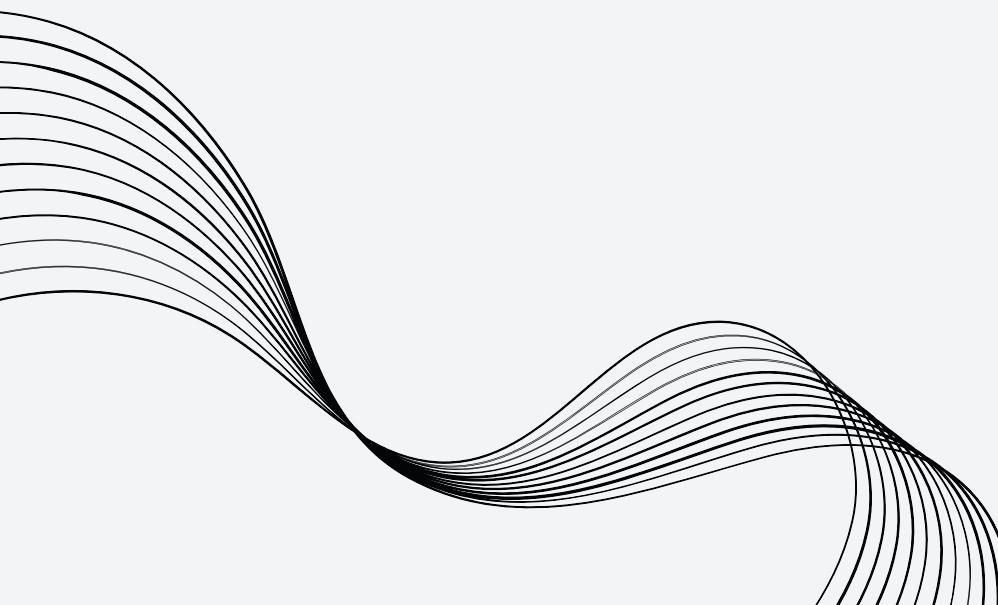


Tool used:



Link: <https://github.com/gitleaks/gitleaks>

Configuration file:



A screenshot of a GitHub code editor showing a configuration file named `.pre-commit-config.yaml`. The file contains YAML code for setting up a pre-commit hook to check for git leaks. The code is color-coded: teal for `repos`, red for URLs and `rev`, and green for `hooks` and `id`.

```
1 repos:
2   - repo: https://github.com/gitleaks/gitleaks
3     rev: v8.18.0
4   hooks:
5     - id: gitleaks
6
```

The file has a size of 103 B. The GitHub interface shows standard actions like Blame, Edit, Lock, Replace, and Delete, along with download icons.

Result:

```
(ismail㉿kali)-[~/devsecops/devsecops_backend]
└─$ echo "password: dokcer1234uuuu" >> password

(ismail㉿kali)-[~/devsecops/devsecops_backend]
└─$ git add password

(ismail㉿kali)-[~/devsecops/devsecops_backend]
└─$ git commit -m "testing pre-commit hooks"
[WARNING] Unstaged files detected.
[INFO] Stashing unstaged files to /home/ismail/.cache/pre-commit/patch1703809669-19071.
Detect hardcoded secrets.....(no files to check)Skipped
[INFO] Restored changes from /home/ismail/.cache/pre-commit/patch1703809669-19071.
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add/rm <file> ..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:   .pre-commit-config.yaml
    deleted:   docker_credentials

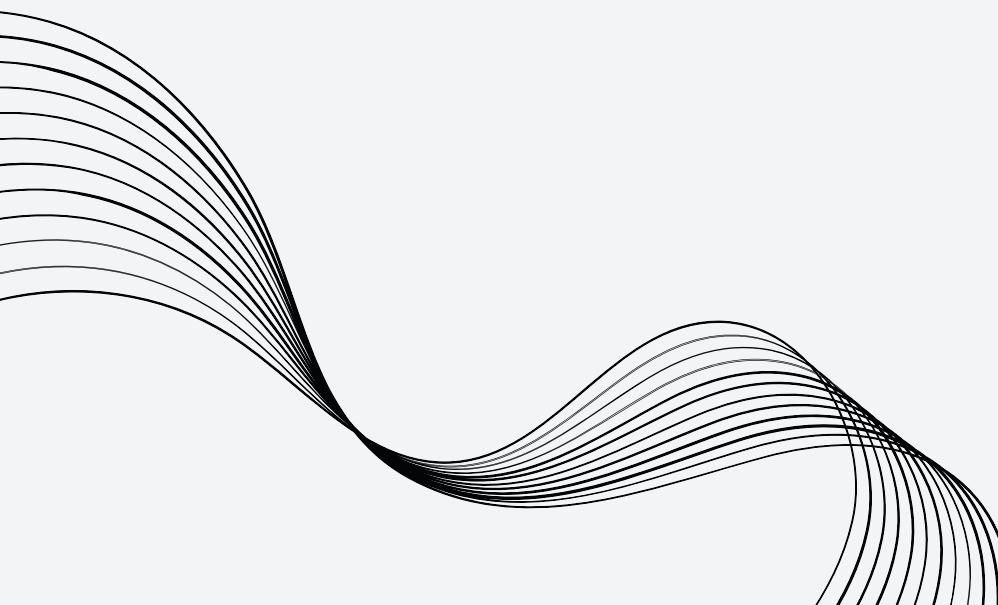
Untracked files:
  (use "git add <file> ..." to include in what will be committed)
    .pre-commit-config.yaml

no changes added to commit (use "git add" and/or "git commit -a")
```

SAST

Static Application Security Testing (SAST) is a critical component of a comprehensive security strategy in software development. SAST involves analyzing the source code or compiled binaries of an application to identify and mitigate security vulnerabilities and coding flaws early in the development process. It helps developers catch issues such as SQL injection, cross-site scripting (xss), and other vulnerabilities before the code is deployed.

We will use Gitlab for all security scans.



To add SAST scans to our pipeline, we should simply include Gitlab SAST file to our YAML file:

```
include:  
- template: Security/SAST.gitlab-ci.yml
```

More info: https://docs.gitlab.com/ee/user/application_security/sast/

After running our pipeline, we can now look at the scan results by downloading artifact, or using Gitlab vulnerability management tool.

Results

Vulnerability report

[+ Submit vulnerability](#)[Export](#)

The Vulnerability Report shows results of successful scans on your project's default branch, manually added vulnerability records, and vulnerabilities found from scanning operational environments. [Learn more.](#)

Development vulnerabilities 0

Operational vulnerabilities 0

Security reports last updated 1 week ago [#1121952268](#) • SBOMs last updated 1 week ago [#1121952268](#)

◆ Critical

0

◆ High

0

▼ Medium

0

● Low

0

● Info

0

?

 Unknown

0

Status

Needs triage +1 more

Severity

All severities

Tool

SAST

Activity

Still detected



The same way, we add scans for dependency scanning and container security, here is the full configuration file:

```
4 stages:
5   - test
6   - build
7   - container_scanning
8   - deploy
9
10 include:
11   - template: Security/Dependency-Scanning.gitlab-ci.yml
12   - template: Security/SAST.gitlab-ci.yml
13   - template: Security/Container-Scanning.gitlab-ci.yml
14
15 docker-build:
16   stage: build
17   before_script:
18     - docker login -u 4bd3lgh4f0r -p $DOCKER_USER_PASSWORD
19   script:
20     - docker build -t 4bd3lgh4f0r/devsecops_frontend .
21     - docker push 4bd3lgh4f0r/devsecops_frontend
22
23 container_scanning:
24   stage: container_scanning
25   variables:
26     CS_IMAGE: 4bd3lgh4f0r/devsecops_frontend
27
```

Results

Vulnerability report

The Vulnerability Report shows results of successful scans on your project's default branch, manually added vulnerability records, and vulnerabilities found from scanning operational environments. [Learn more](#).

Development vulnerabilities 782 Operational vulnerabilities 0

Security reports last updated 1 week ago #1121952268 • SBOMs last updated 1 week ago #1121952268



Status: Needs triage +1 more ▾ Severity: All severities ▾ Tool: All tools ▾ Activity: Still detected ▾

Group by: None ▾

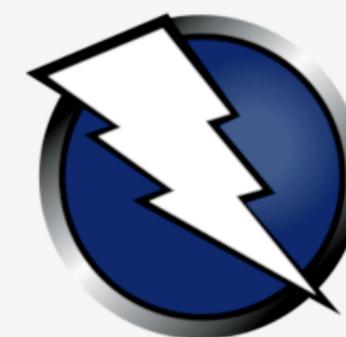
| <input type="checkbox"/> Detected | Status | Severity | Description | Identifier | Tool | Activity |
|-----------------------------------|------------|--------------|---|----------------|--------------------|----------|
| <input type="checkbox"/> | 2023-12-28 | Needs Triage | ● Critical CVE-2023-45853 in :zlib1g 4bd3lgh4f0r/devsecops_frontend | CVE-2023-45853 | Container Scanning | |
| <input type="checkbox"/> | 2023-12-28 | Needs Triage | ● Critical CVE-2023-51385 in :openssh-client 4bd3lgh4f0r/devsecops_frontend | CVE-2023-51385 | Container Scanning | |



DAST

Dynamic Application Security Testing (DAST) is a security testing methodology that assesses the vulnerability of a software application by actively examining its running version. Unlike static analysis that reviews the source code, DAST operates during runtime, simulating real-world attack scenarios to identify potential vulnerabilities. DAST is crucial for ensuring the security of software as it provides a dynamic and comprehensive assessment of an application's security posture, detecting issues such as input validation flaws, authentication weaknesses, and other runtime vulnerabilities. By mimicking the behavior of malicious actors, DAST helps uncover vulnerabilities that may not be evident in static analysis, allowing developers and security teams to address weaknesses and enhance the overall security of the software.

Tool used:



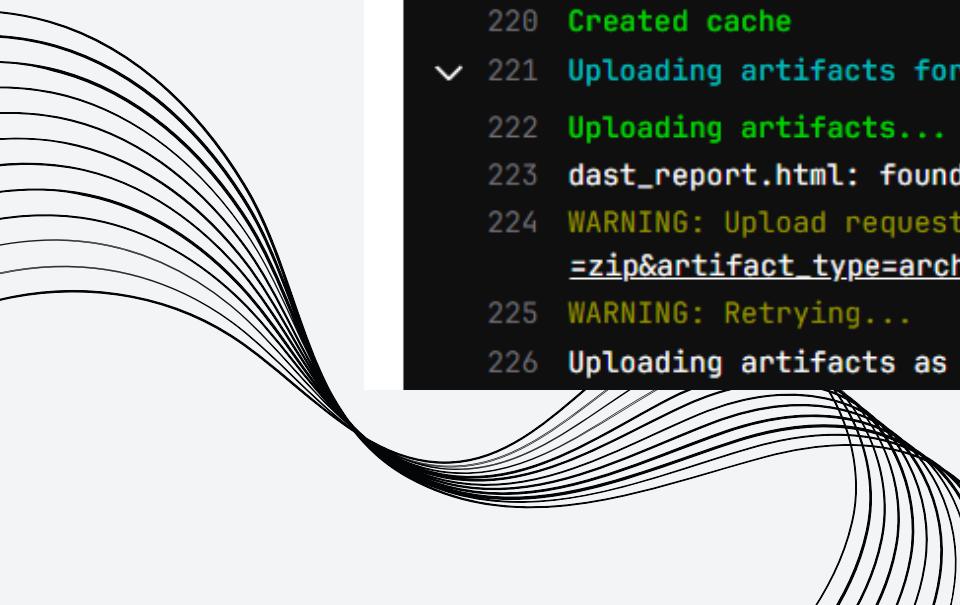
OWASP
Zed Attack Proxy

Configuration:

```
dast:
  stage: dast
  rules:
    - needs:
        - job: "docker-build"
        - job: "container_scanning"
    - exists:
        - $CI_COMMIT_BRANCH == "main"
  script:
    - docker network create my_network
    - docker run -d -p 9001:4200 --name frontend --network my_network 4bd3lgh4f0r/devsecops_frontend
    - docker pull softwaresecurityproject/zap-stable
    - docker run --name dast --network my_network --rm -v "$(pwd):/zap/wrk" -t softwaresecurityproject/zap-stable zap.sh -cmd -quickurl http://frontend:4200 -quickprogress -quickout /zap/wrk/dast_report.html

  artifacts:
    paths:
      - dast_report.html
```

We download the artifact after the job is finish:



iccn_devsecops / Devsecops Frontend / Jobs / #5887454418

Search job log ? Copy Print Up Down Left Right

204 1255 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/zap/.ZAP/session
205 1256 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/zap/.ZAP/dirbuster
206 1257 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/zap/.ZAP/fuzzers
207 1257 [main] INFO org.parosproxy.paros.Constant - Creating directory /home/zap/.ZAP/plugin
208 Jan 08, 2024 11:14:12 PM java.util.prefs.FileSystemPreferences\$1 run
209 INFO: Created user preferences directory.
210 Accessing URL
211 Using traditional spider
212 Active scanning
213 [=====] 100%
214 Attack complete
215 Writing results to /zap/wrk/dast_report.html
216 Saving cache for successful job 00:01
217 Creating cache main-protected...
218 WARNING: No matching files. Path is empty.
219 Archive is up to date!
220 Created cache
221 Uploading artifacts for successful job 00:02
222 Uploading artifacts...
223 dast_report.html: found 1 matching artifact files and directories
224 WARNING: Upload request redirected location=https://gitlab.com/api/v4/jobs/5887454418/artifacts?artifact_format=zip&artifact_type=archive new-url=https://gitlab.com
225 WARNING: Retrying... context=artifacts-uploader error=request redirected
226 Uploading artifacts as "archive" to coordinator... 201 Created id=5887454418 responseStatus=201 Created token=65_s2vc9

Duration: 2 minutes 51 seconds
Finished: 59 minutes ago
Queued: 0 seconds
Timeout: 1h (from project) ?
Runner: #12270835 (zxwgkjAPH) 3-blue.saas-linux-small-amd64.runners-manager.gitlab.com/default

Job artifacts ?
These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Download Browse

Commit e0e73873 View
Update docker-deploy.yml

Pipeline #1131239660 Passed for main View

dast

Result:



Site: <http://frontend:4200>

Generated on Mon, 8 Jan 2024 23:14:36

ZAP Version: 2.14.0

Summary of Alerts

| Risk Level | Number of Alerts |
|------------------|------------------|
| High | 1 |
| Medium | 3 |
| Low | 2 |
| Informational | 0 |
| False Positives: | 0 |

Alerts

| Name | Risk Level | Number of Instances |
|---|------------|---------------------|
| Cloud Metadata Potentially Exposed | High | 1 |
| .htaccess Information Leak | Medium | 1 |
| Content Security Policy (CSP) Header Not Set | Medium | 3 |
| Missing Anti-clickjacking Header | Medium | 3 |
| Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) | Low | 3 |
| X-Content-Type-Options Header Missing | Low | 3 |

The same thing is done to backend pipeline:

Result:

Vulnerability report

The Vulnerability Report shows results of successful scans on your project's default branch, manually added vulnerability records, and vulnerabilities found from scanning operational environments. [Learn more.](#)

Development vulnerabilities 180 Operational vulnerabilities 0

Security reports last updated 1 day ago #1132323145 • SBOMs last updated 1 day ago #1132323145

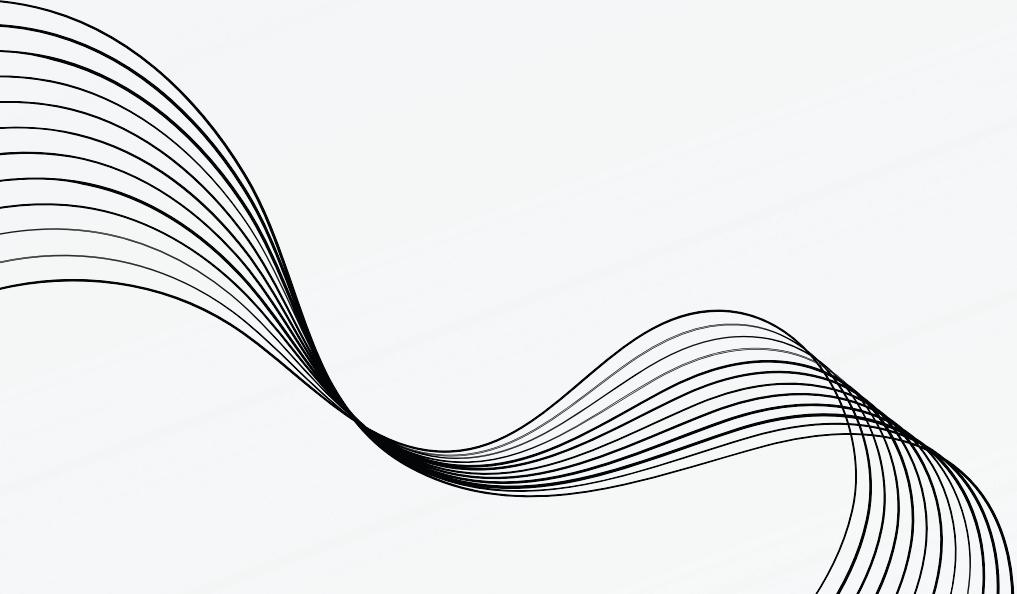
| | | | | | |
|---|---|--|---|---|--|
| ● Critical 12 | ◆ High 45 | ▼ Medium 49 | ● Low 74 | i Info 0 | ? Unknown 0 |
|---|---|--|---|---|--|

Status: Needs triage +1 more ▾ Severity: All severities ▾ Tool: All tools ▾ Activity: Still detected ▾

Group by: None ▾

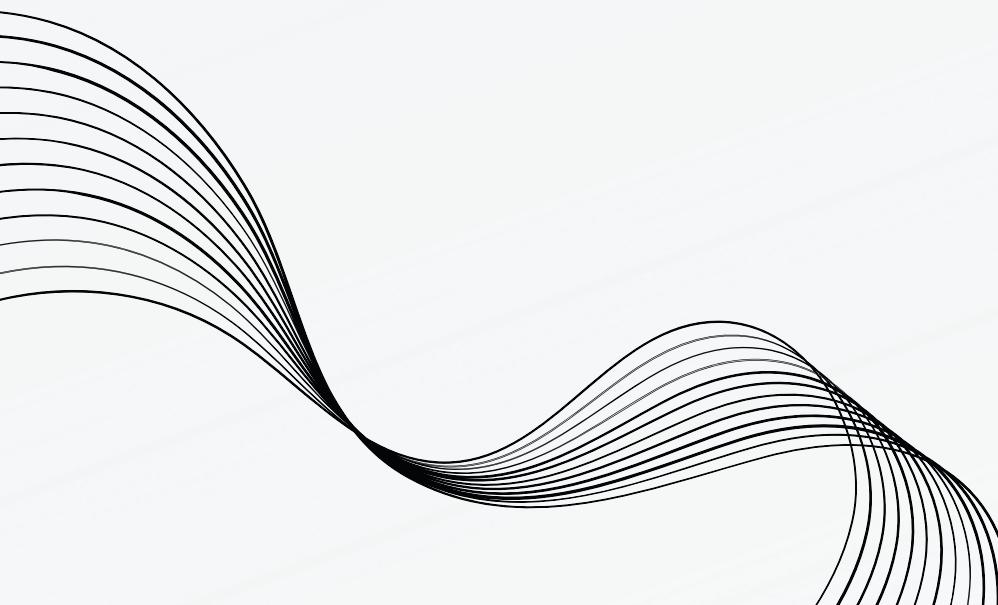
| <input type="checkbox"/> Detected | Status | ↓ Severity | Description | Identifier | Tool | Activity |
|-----------------------------------|------------|--------------|--|---------------|--------------------|----------|
| <input type="checkbox"/> | 2024-01-08 | Needs Triage | ● Critical CVE-2022-1292 in :openssl 4bd3lgh4f0r/devsecops_backend | CVE-2022-1292 | Container Scanning | |
| <input type="checkbox"/> | 2024-01-08 | Needs Triage | ● Critical CVE-2019-8457 in :libdb5.3 4bd3lgh4f0r/devsecops_backend | CVE-2019-8457 | Container Scanning | |

4 - MONITORING



What is Monitoring?

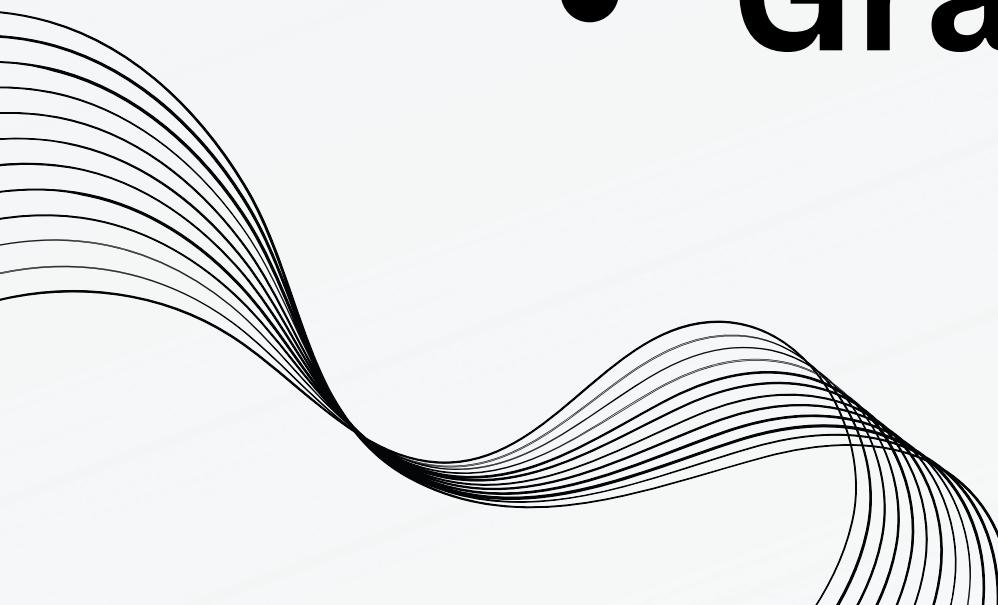
Monitoring is an essential aspect of managing and ensuring the health and performance of our cluster and applications. Monitoring allows us to collect and analyze data about various components in our Kubernetes environment.



How to do it?

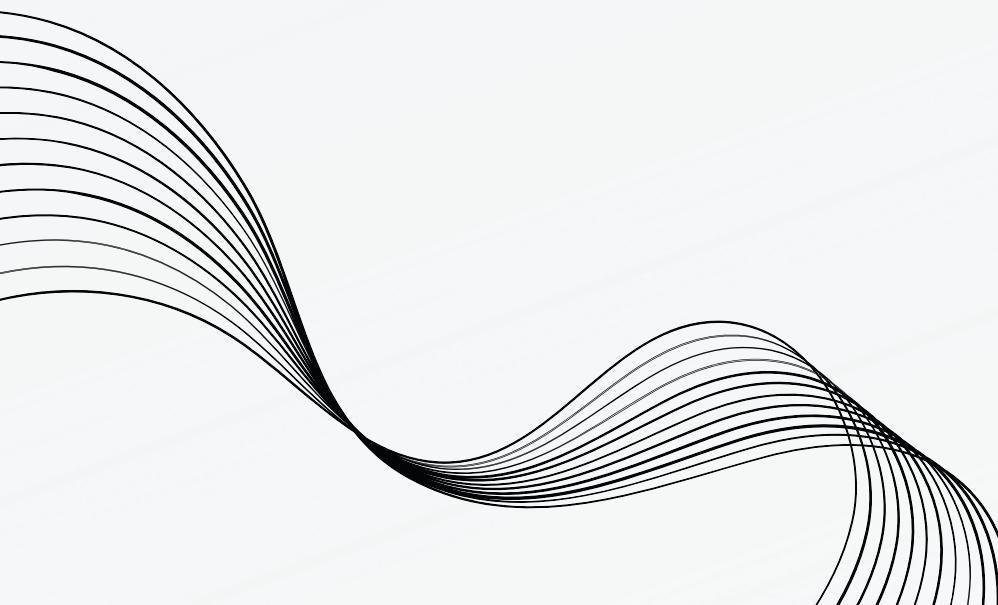
There are several tools to monitor your environment in Kubernetes and ensure the tracking of your system resources and in this project we will use:

- **Prometheus**
- **Grafana**



Prometheus

Prometheus is an open-source monitoring system that collects and analyzes time series data, providing powerful querying, alerting, and integration capabilities for monitoring applications and infrastructure in dynamic environments like Kubernetes.



Grafana

Grafana is an open-source visualization tool that provides customizable dashboards for monitoring and analyzing metrics, while Prometheus is an open-source monitoring system that collects and stores time series data from various sources.



Grafana

Adding prometheus repo with helm

```
C:\Users\DELL>helm repo add prometheus-community https://prometheus-community.github.io/helm-charts  
"prometheus-community" already exists with the same configuration, skipping
```

Adding the charts repo to my repositories

```
C:\Users\DELL>helm repo add stable https://charts.helm.sh/stable  
"stable" has been added to your repositories
```

Updating repo with helm

```
C:\Users\DELL>helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "prometheus-community" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. *Happy Helming!*
```

Installing Prometheus

```
C:\Users\DELL>helm install prometheus prometheus-community/kube-prometheus-stack
NAME: prometheus
LAST DEPLOYED: Fri Dec 29 11:19:07 2023
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.

List all resources in Kubernetes cluster

```
C:\Users\DELL>minikube kubectl get all
NAME                                         READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-kube-prometheus-alertmanager-0   0/2     Init:0/1      0          6s
pod/prometheus-grafana-57cc5d6996-2mz7k                    0/3     ContainerCreating  0          41s
pod/prometheus-kube-prometheus-operator-558444bb59-qsw5p    1/1     Running      0          41s
pod/prometheus-kube-state-metrics-6cd846d5cf-f8w99        1/1     Running      0          41s
pod/prometheus-prometheus-kube-prometheus-prometheus-0    0/2     Init:0/1      0          6s
pod/prometheus-prometheus-node-exporter-l74gq                1/1     Running      0          41s

NAME                           TYPE        CLUSTER-IP       EXTERNAL-IP      PORT(S)           AGE
service/alertmanager-operated ClusterIP  None            <none>          9093/TCP,9094/TCP,9094/UDP  6s
service/kubernetes             ClusterIP  10.96.0.1       <none>          443/TCP          7m6s
service/prometheus-grafana     ClusterIP  10.105.192.224  <none>          80/TCP           42s
service/prometheus-kube-prometheus-alertmanager   ClusterIP  10.107.248.98  <none>          9093/TCP,8080/TCP  42s
service/prometheus-kube-prometheus-operator       ClusterIP  10.99.104.83   <none>          443/TCP          42s
service/prometheus-kube-prometheus-prometheus    ClusterIP  10.109.153.168 <none>          9090/TCP,8080/TCP  42s
service/prometheus-kube-state-metrics           ClusterIP  10.105.97.250  <none>          8080/TCP          42s
service/prometheus-operated            ClusterIP  None            <none>          9090/TCP          6s
service/prometheus-prometheus-node-exporter     ClusterIP  10.107.141.222 <none>          9100/TCP          42s

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  41s

NAME                                         READY   UP-TO-DATE  AVAILABLE   AGE
deployment.apps/prometheus-grafana          0/1     1           0          41s
deployment.apps/prometheus-kube-prometheus-operator  1/1     1           1          41s
deployment.apps/prometheus-kube-state-metrics  1/1     1           1          41s

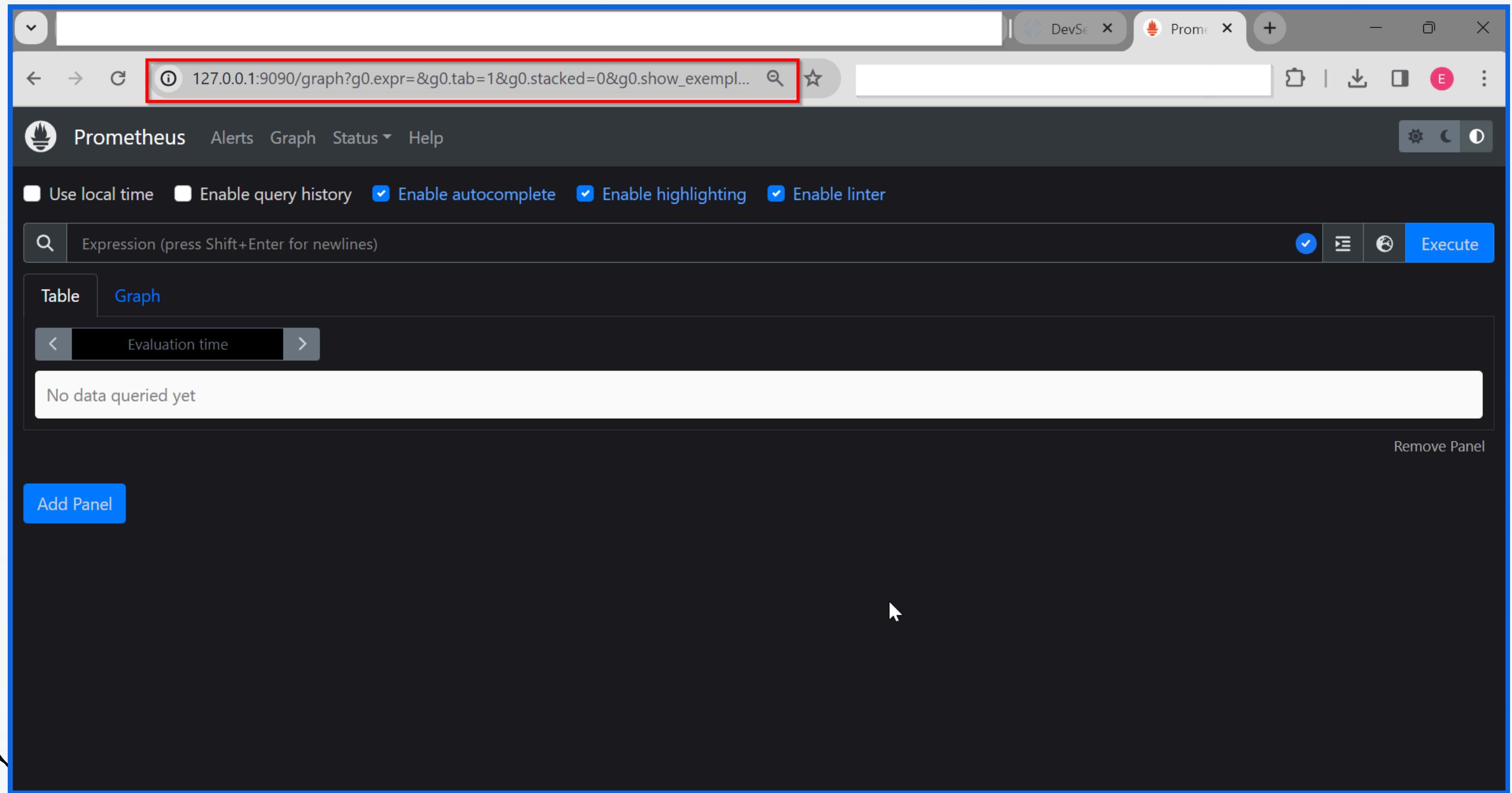
NAME                               DESIRED  CURRENT  READY   AGE
replicaset.apps/prometheus-grafana-57cc5d6996  1        1        0       41s
replicaset.apps/prometheus-kube-prometheus-operator-558444bb59  1        1        1       41s
replicaset.apps/prometheus-kube-state-metrics-6cd846d5cf      1        1        1       41s

NAME                                         READY   AGE
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager  0/1     6s
statefulset.apps/prometheus-prometheus-kube-prometheus-prometheus    0/1     6s
```

Running Prometheus using port-forward

```
C:\Users\DELL\Desktop\DevSecOps>kubectl -- port-forward prometheus-prometheus-kube-prometheus-prometheus-0 9090 -n webapp
Forwarding from 127.0.0.1:9090 -> 9090
Forwarding from [::1]:9090 -> 9090
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090
|
```

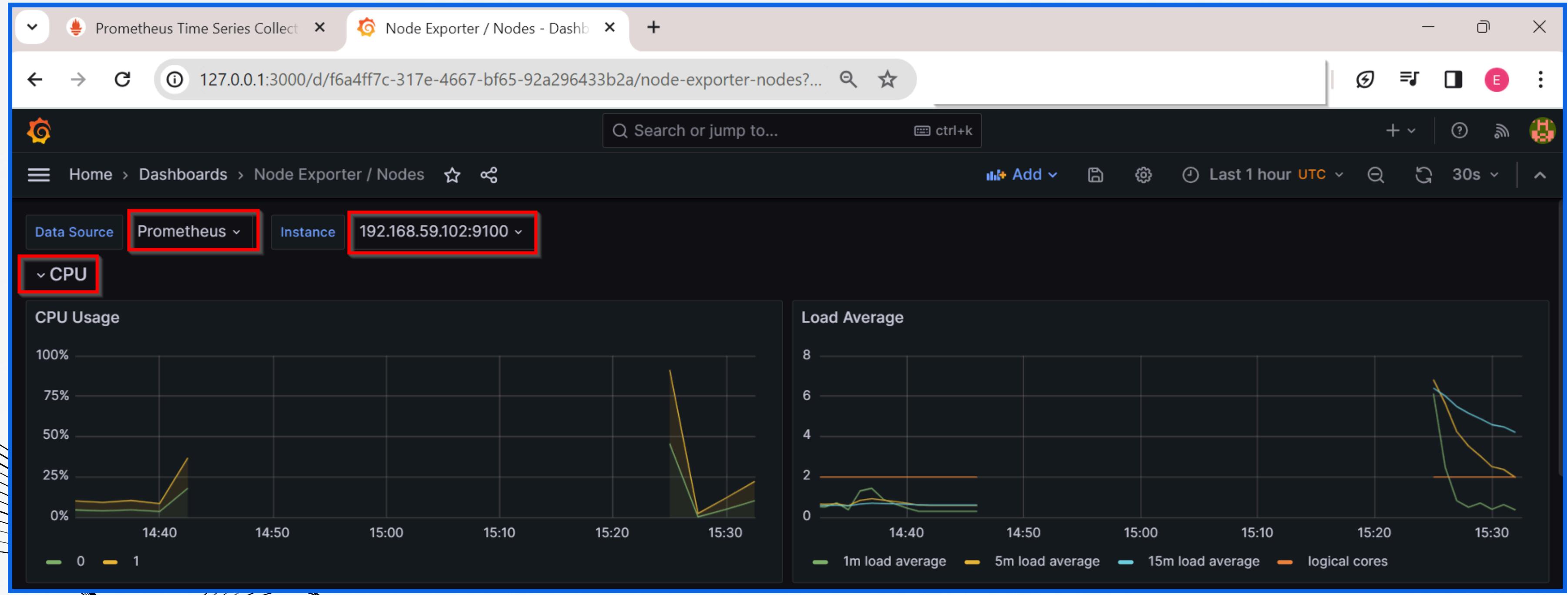
Prometheus Web-UI



Running Grafana using port-forward

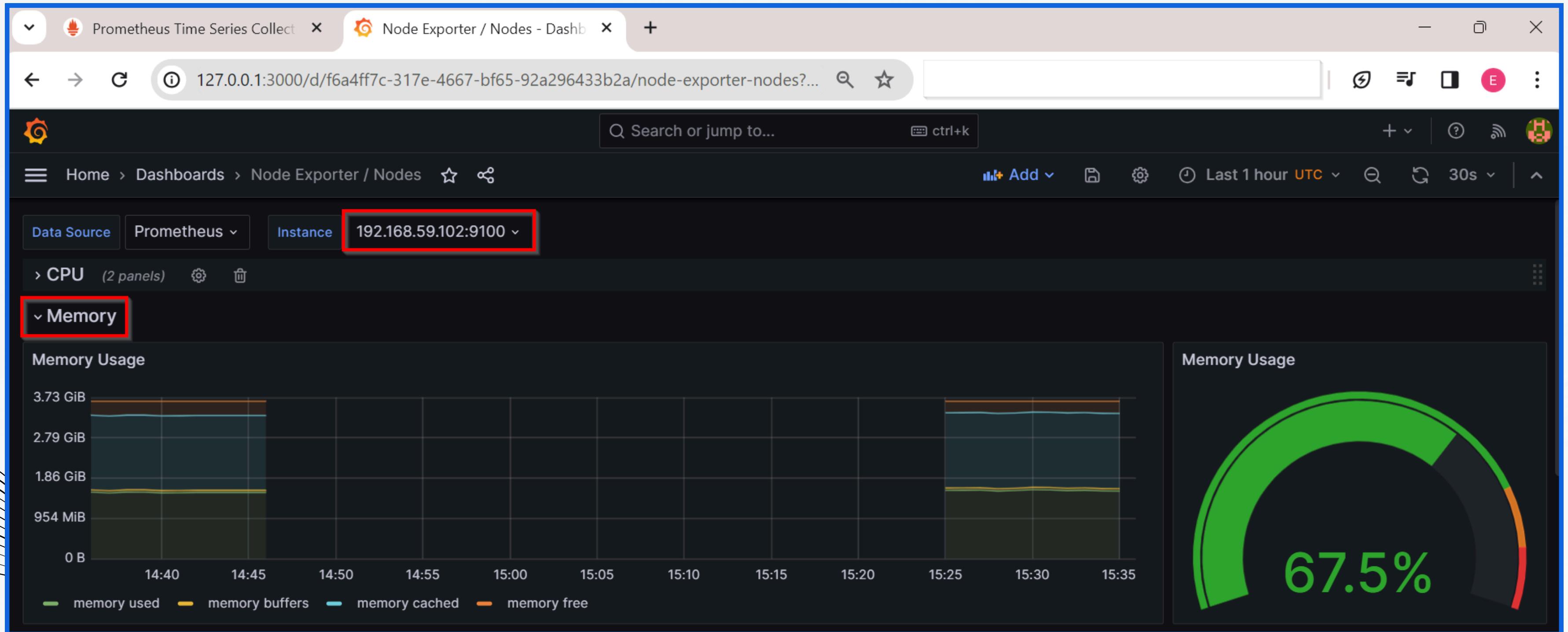
System Monitoring

=> CPU:



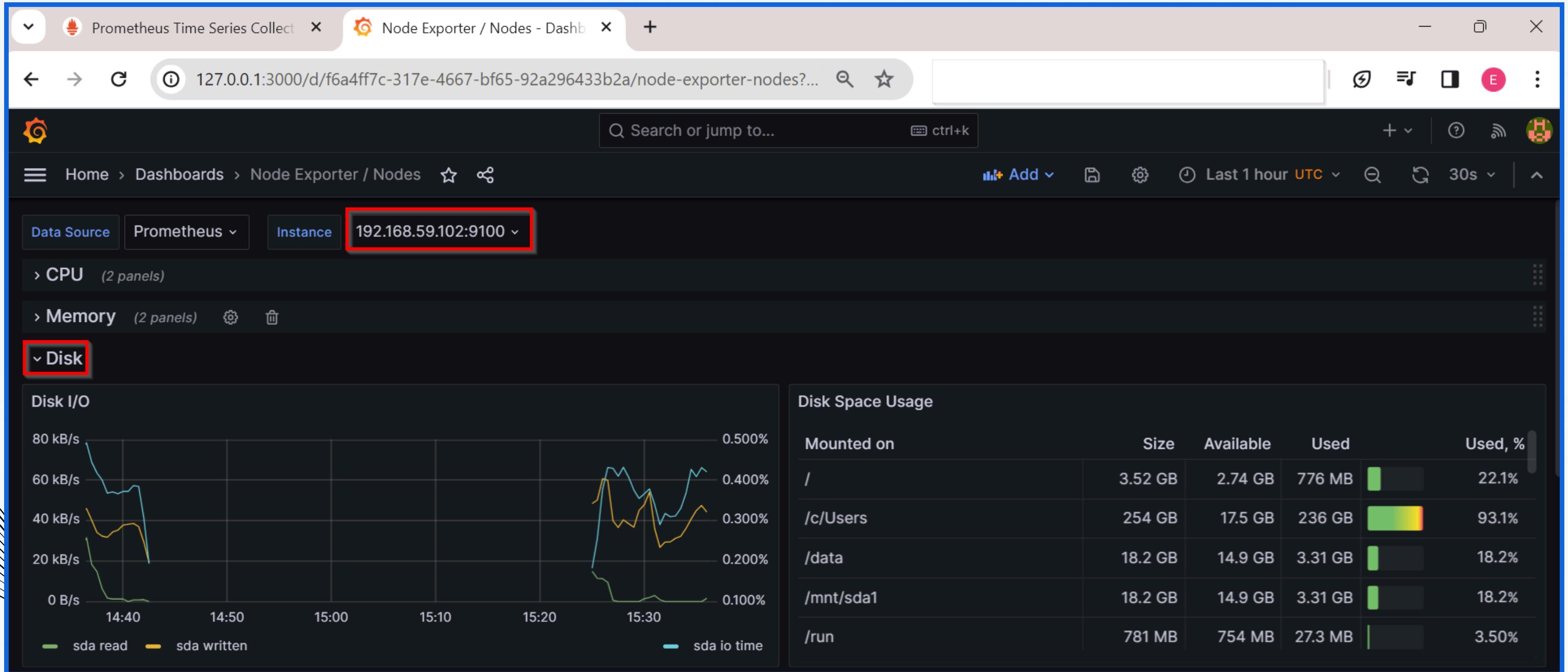
System Monitoring

=> RAM:



System Monitoring

=> Disk:



Application Monitoring

=> Mysql deployment:

```
C:\Users\DELL\Desktop\DevSecOps>kubectl -- apply -f mysql-deployment.yaml -n webapp
deployment.apps/mysql-deployment created
service/mysql-service unchanged
```

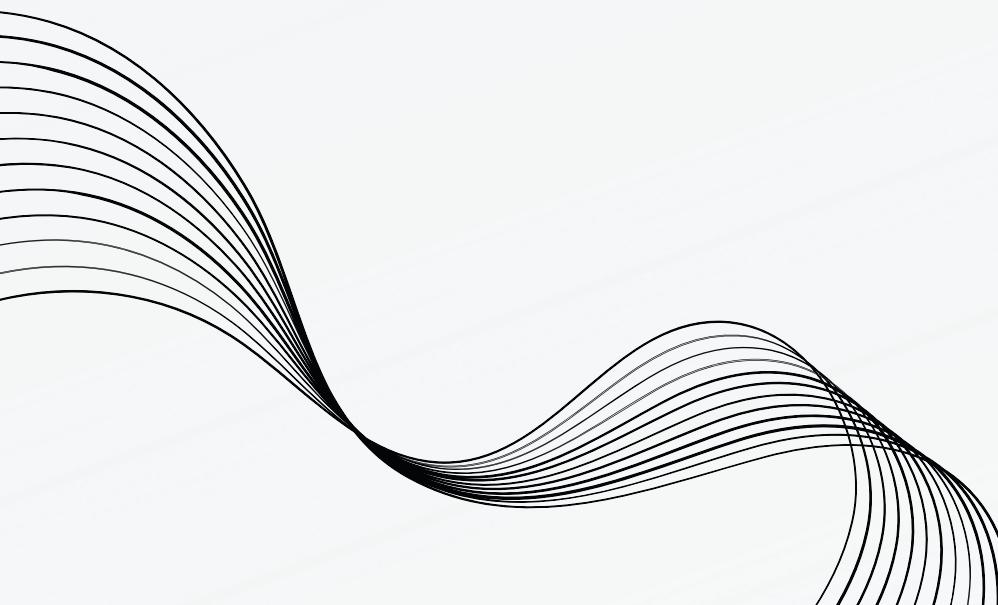
Application Monitoring

=> Installation of mysql-exporter:

```
C:\Users\DELL\Desktop\DevSecOps>helm install mysql-exporter prometheus-community/prometheus-mysql-exporter -f values.yaml -n webapp
NAME: mysql-exporter
LAST DEPLOYED: Sun Jan  7 18:17:22 2024
NAMESPACE: webapp
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace webapp -l "app.kubernetes.io/name=prometheus-mysql-exporter,app.kubernetes.io/instance=mysql-exporter" -o jsonpath="{.items[0].metadata.name}")
  echo "Visit http://127.0.0.1:9104 to use your application"
  kubectl --namespace webapp port-forward $POD_NAME 9104
```

Application Monitoring

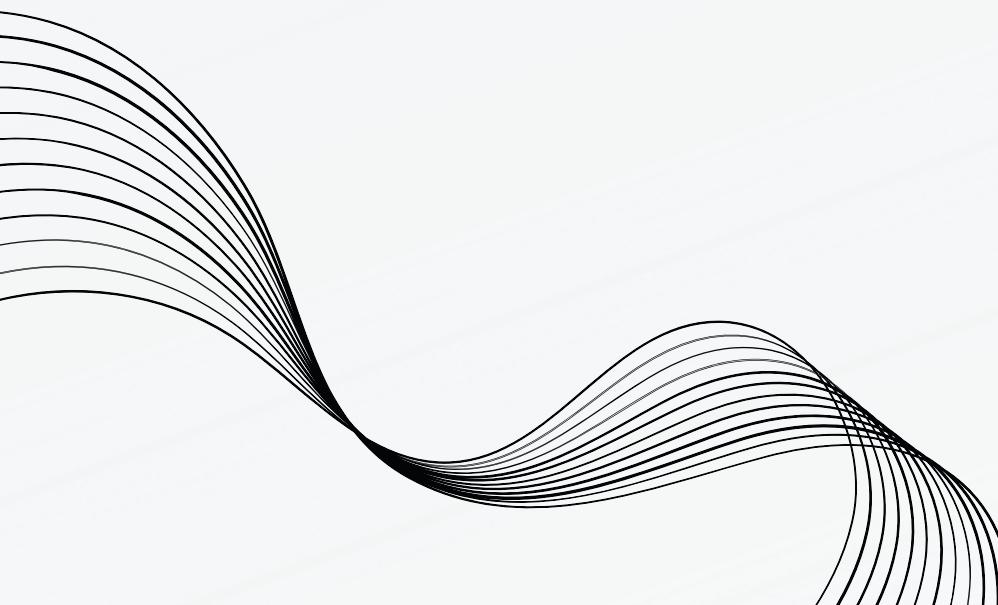
=> Collected metrics of mysql:



```
  Prometheus Time Series Collect | Prometheus / Overview - Dashb | 127.0.0.1:9104/metrics
  127.0.0.1:9104/metrics

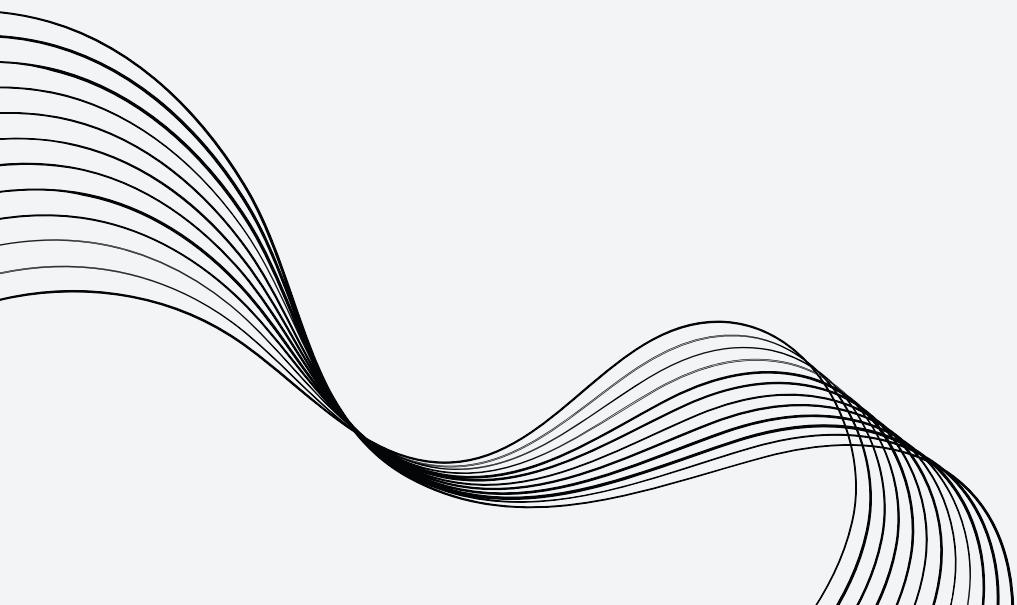
HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
TYPE go_gc_duration_seconds summary
_gc_duration_seconds{quantile="0"} 6.74e-05
_gc_duration_seconds{quantile="0.25"} 6.74e-05
_gc_duration_seconds{quantile="0.5"} 6.74e-05
_gc_duration_seconds{quantile="0.75"} 6.74e-05
_gc_duration_seconds{quantile="1"} 6.74e-05
_gc_duration_seconds_sum 6.74e-05
_gc_duration_seconds_count 1
HELP go_goroutines Number of goroutines that currently exist.
TYPE go_goroutines gauge
goroutines 9
HELP go_info Information about the Go environment.
TYPE go_info gauge
_info{version="go1.20.5"} 1
HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
TYPE go_memstats_alloc_bytes gauge
_memstats_alloc_bytes 3.305792e+06
HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
TYPE go_memstats_alloc_bytes_total counter
_memstats_alloc_bytes_total 4.258768e+06
HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
TYPE go_memstats_buck_hash_sys_bytes gauge
_memstats_buck_hash_sys_bytes 4580
HELP go_memstats_frees_total Total number of frees.
TYPE go_memstats_frees_total counter
_memstats_frees_total 7217
HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
TYPE go_memstats_gc_sys_bytes gauge
_memstats_gc_sys_bytes 7.440512e+06
HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
TYPE go_memstats_heap_alloc_bytes gauge
_memstats_heap_alloc_bytes 3.305792e+06
HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
TYPE go_memstats_heap_idle_bytes gauge
_memstats_heap_idle_bytes 3.211264e+06
HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
TYPE go_memstats_heap_inuse_bytes gauge
_memstats_heap_inuse_bytes 4.75136e+06
HELP go_memstats_heap_objects Number of allocated objects.
TYPE go_memstats_heap_objects gauge
_memstats_heap_objects 3913
HELP go_memstats_heap_released_bytes Number of heap bytes released to os
```

5 - KUBERNETES DEPLOYMENT



Minikube

Minikube is a lightweight tool that enables developers to run and manage a single-node Kubernetes cluster on their local machine, facilitating the development and testing of containerized applications.



minikube installation with choco :

```
PS C:\WINDOWS\system32> choco install minikube
Chocolatey v2.2.2
Installing the following packages:
minikube
By installing, you accept licenses for the packages.
Progress: Downloading kubernetes-cli 1.29.0... 100%

kubernetes-cli v1.29.0 [Approved]
kubernetes-cli package files install completed. Performing other installation steps.
The package kubernetes-cli wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint): y

Extracting 64-bit C:\ProgramData\chocolatey\lib\kubernetes-cli\tools\kubernetes-client-windows-amd64.tar.gz to C:\ProgramData\chocolatey\lib\kubernetes-cli\tools...
C:\ProgramData\chocolatey\lib\kubernetes-cli\tools
Extracting 64-bit C:\ProgramData\chocolatey\lib\kubernetes-cli\tools\kubernetes-client-windows-amd64.tar to C:\ProgramData\chocolatey\lib\kubernetes-cli\tools...
C:\ProgramData\chocolatey\lib\kubernetes-cli\tools
ShimGen has successfully created a shim for kubectl-convert.exe
ShimGen has successfully created a shim for kubectl.exe
The install of kubernetes-cli was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\kubernetes-cli\tools'
Progress: Downloading Minikube 1.32.0... 100%

Minikube v1.32.0 [Approved]
Minikube package files install completed. Performing other installation steps.
ShimGen has successfully created a shim for minikube.exe
The install of Minikube was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\Minikube'

Chocolatey installed 2/2 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
```

=> Starting a Kubernetes instance locally using Minikube. :

```
PS C:\WINDOWS\system32> minikube start
W1228 21:40:46.991056 31312 main.go:291] Unable to resolve the current Docker CLI context "default": context "default"
: context not found: open C:\Users\LENOVO\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a
33f0688f\meta.json: The system cannot find the path specified.
* minikube v1.32.0 on Microsoft Windows 11 Pro 10.0.22631.2861 Build 22631.2861
* Automatically selected the docker driver. Other choices: virtualbox, ssh
* Using Docker Desktop driver with root privileges
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Downloading Kubernetes v1.28.3 preload ...
    > gcr.io/k8s-minikube/kicbase...: 453.90 MiB / 453.90 MiB 100.00% 1.41 Mi
    > preloaded-images-k8s-v18-v1...: 403.35 MiB / 403.35 MiB 100.00% 1.19 Mi
* Creating docker container (CPUs=2, Memory=4000MB) ...
* 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
PS C:\WINDOWS\system32> minikube status
W1228 21:49:33.151781 38048 main.go:291] Unable to resolve the current Docker CLI context "default": context "default"
: context not found: open C:\Users\LENOVO\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a
33f0688f\meta.json: The system cannot find the path specified.
minikube
  type: Control Plane
  host: Running
  kubelet: Running
  apiserver: Running
  kubeconfig: Configured
```

Creating the 'webapp' namespace:

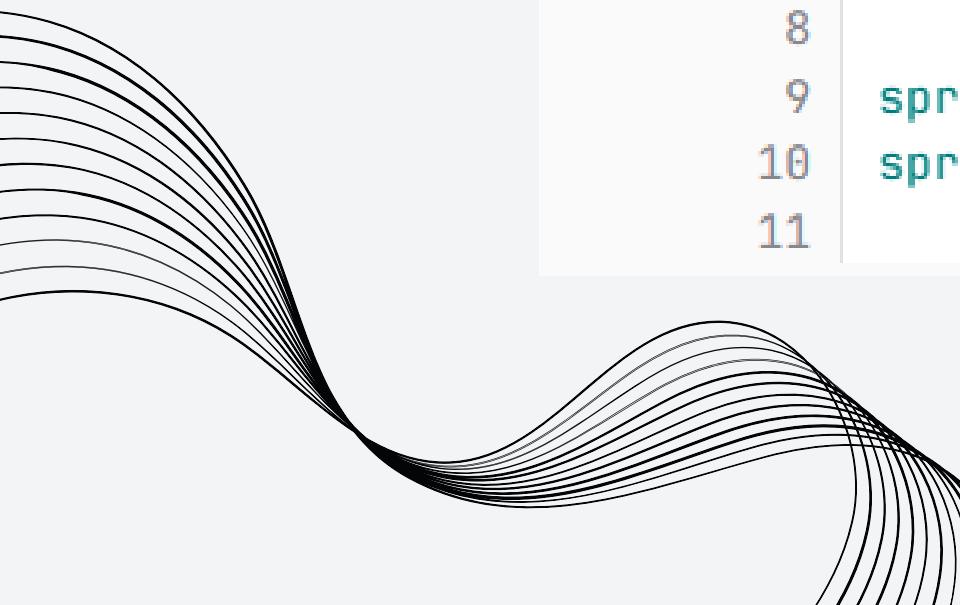
```
PS C:\WINDOWS\system32> kubectl create namespace webapp  
>>  
namespace/webapp created
```

We created a new Kubernetes namespace named 'webapp.' The output "namespace/webapp created" confirms the successful creation of the 'webapp' namespace, providing an isolated environment for managing resources within a Kubernetes cluster.



Use of environment variables

Before starting the creation of manifests for deployment, small modifications need to be made to the source code of the "application.properties" file. We will use environment variables for the database connection.



application.properties 422 B

```
1 spring.datasource.url= jdbc:mysql://${SPRING_DATASOURCE_URL}:3306/testdb?useSSL=false
2 spring.datasource.username= root
3 spring.datasource.password= root
4
5 spring.datasource.url=${SPRING_DATASOURCE_URL}
6 spring.datasource.username=${SPRING_DATASOURCE_USERNAME}
7 spring.datasource.password=${MYSQL_ROOT_PASSWORD}
8
9 spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQLDialect
10 spring.jpa.hibernate.ddl-auto= update
11
```

Deployment

=> backend.yaml (deployment) :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: backend-container
          image: 4bd3lgh4f0r/devsecops_backend:latest
          ports:
            - containerPort: 8080
          env:
            - name: SPRING_DATASOURCE_URL
              value: "jdbc:mysql://frontend-service:3306/testdb"
            - name: FRONTEND_IP
              value: "frontend-service"
```

Deployment

=> frontend.yaml (deployment) :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend-container
          image: 4bd3lgh4f0r/devsecops_frontend
          imagePullPolicy: Always
          ports:
            - name: http
              containerPort: 4200
          env:
            - name: backendIP
              value: "backend-service"
```

Deployment

=> mysq.yaml (deployment) :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deployment
  labels:
    app: mysql
  namespace: webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:latest
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: root_password
            - name: MYSQL_DATABASE
              value: webapp_database
```

Service

=> Backend.yaml (Service) :

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

Service

=> Frontend.yaml (Service) :

```
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
  namespace: webapp
spec:
  type: ClusterIP
  selector:
    app: frontend
  ports:
  - name: http
    port: 4200
    targetPort: 4200
```

Service

=> mysql.yaml (Service) :

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

Application of Configurations in the Kubernetes Cluster:

```
PS C:\Users\LENOVO\webapp-k8s-config-main> kubectl apply -f backend.yaml -n webapp
deployment.apps/backend-deployment created
service/backend-service created
PS C:\Users\LENOVO\webapp-k8s-config-main> kubectl apply -f frontend.yaml -n webapp
deployment.apps/frontend-deployment created
service/frontend-service created
PS C:\Users\LENOVO\webapp-k8s-config-main> kubectl apply -f mysql.yaml -n webapp
deployment.apps/mysql-deployment created
service/mysql-service created
```

the pods are currently running

```
PS C:\Users\LENOVO\webapp-k8s-config-main> kubectl get pods -n webapp
NAME                           READY   STATUS    RESTARTS   AGE
backend-deployment-786b959994-n894m   1/1     Running   1 (48s ago)   99s
frontend-deployment-55b4f46d6c-jxk2p   1/1     Running   1 (21m ago)   22m
frontend-deployment-55b4f46d6c-6dwqb   1/1     Running   0           20m
mysql-deployment-67d6cd6bbc-rwk4p    1/1     Running   0           20m
```

Accessing the Application Locally with kubectl port-forward

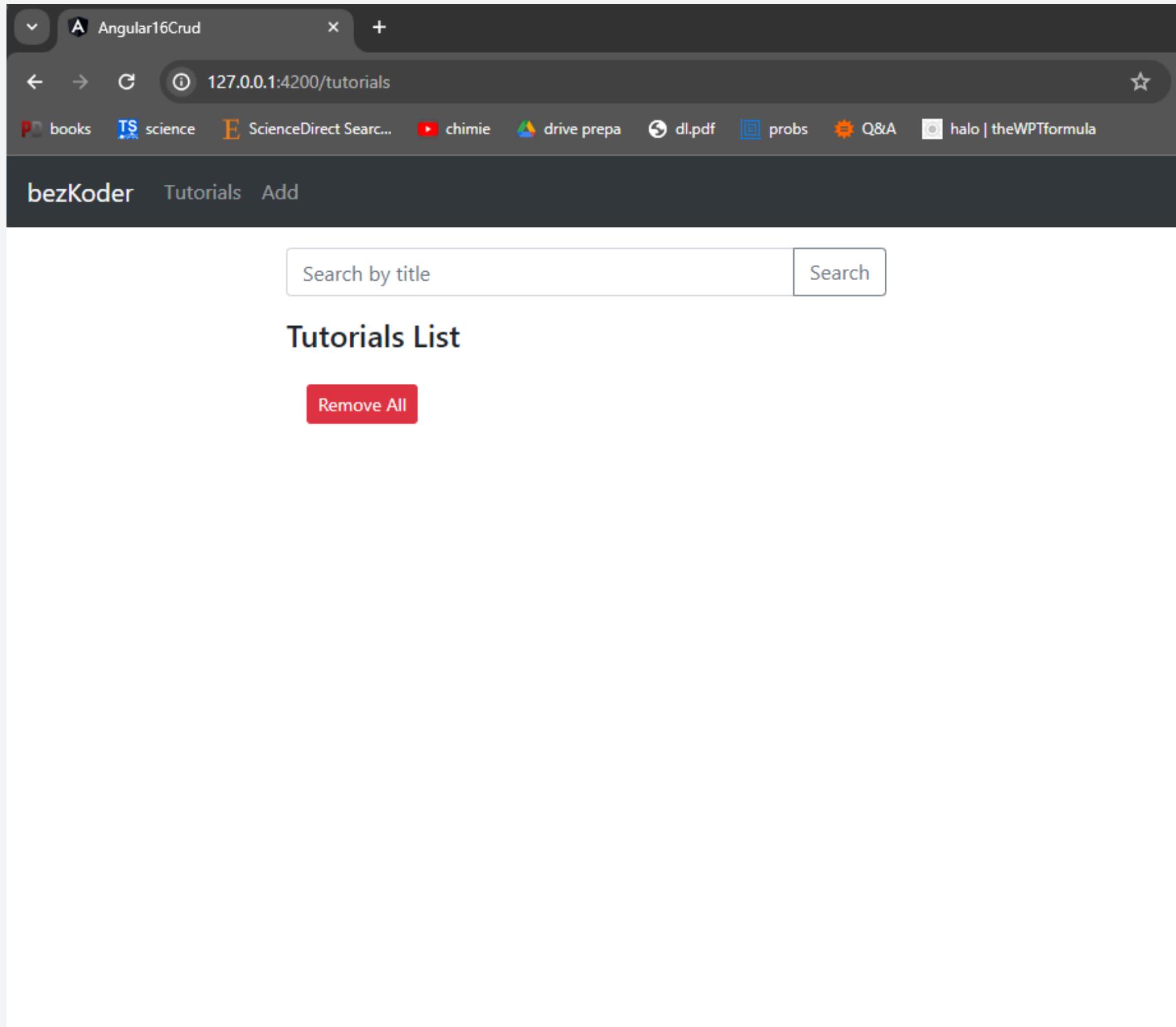
backend:

```
PS C:\Users\LENOVO> kubectl port-forward service/backend-service 8080:8080 -n webapp
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
|
```

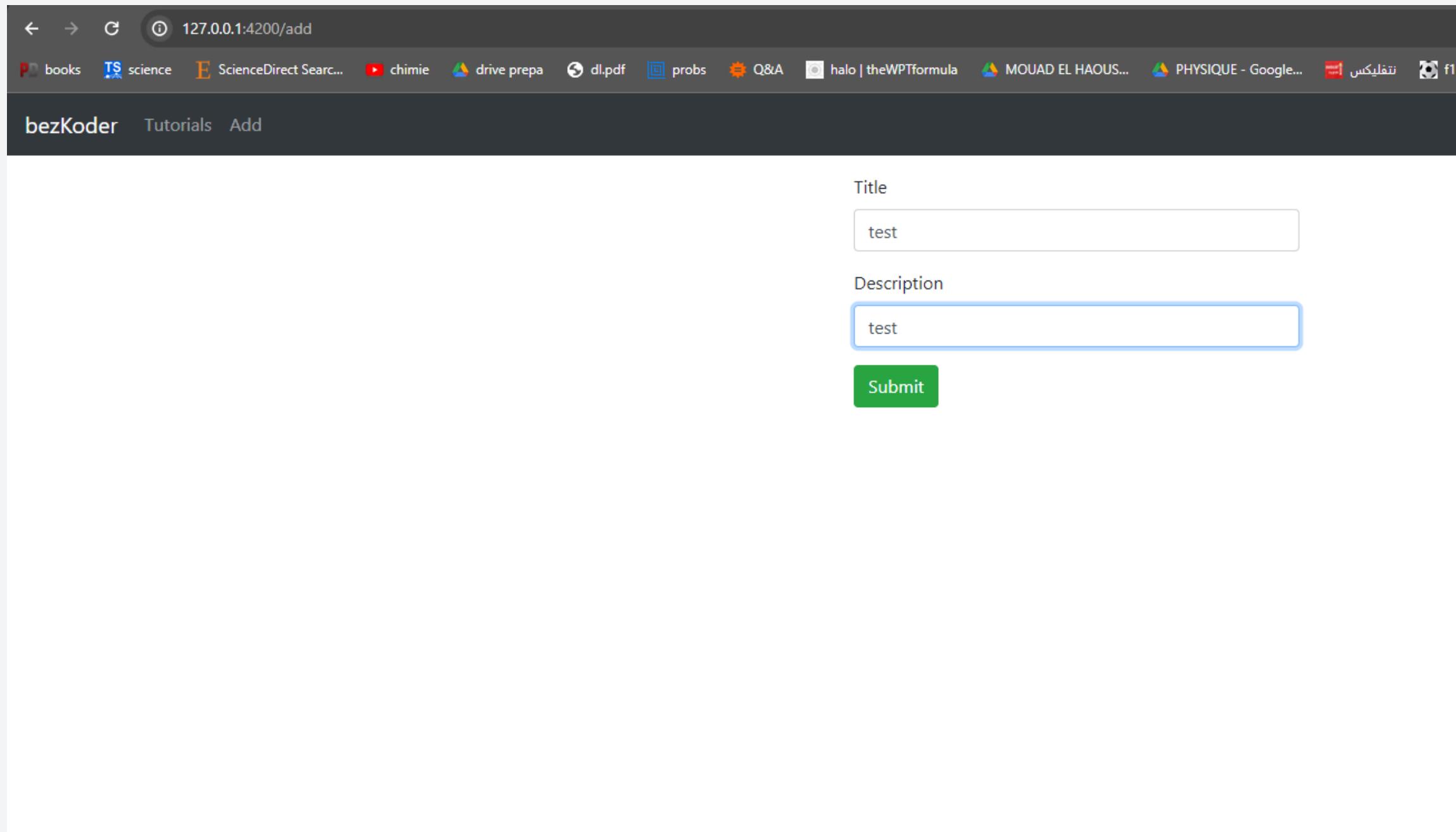
frontend:

```
PS C:\Users\LENOVO\webapp-k8s-config-main> kubectl port-forward service/frontend-service 4200:4200 -n webapp
Forwarding from 127.0.0.1:4200 -> 4200
Forwarding from [::1]:4200 -> 4200
Handling connection for 4200
|
```

the frontend is successfully accessed:



we try to interact with the database:



← → G 127.0.0.1:4200/add

books science ScienceDirect Searc... chimie drive prepa dl.pdf probs Q&A halo | theWPTformula MOUAD EL HAOUS... PHYSIQUE - Google... تفليكس f1

bezKoder Tutorials Add

Title

Description

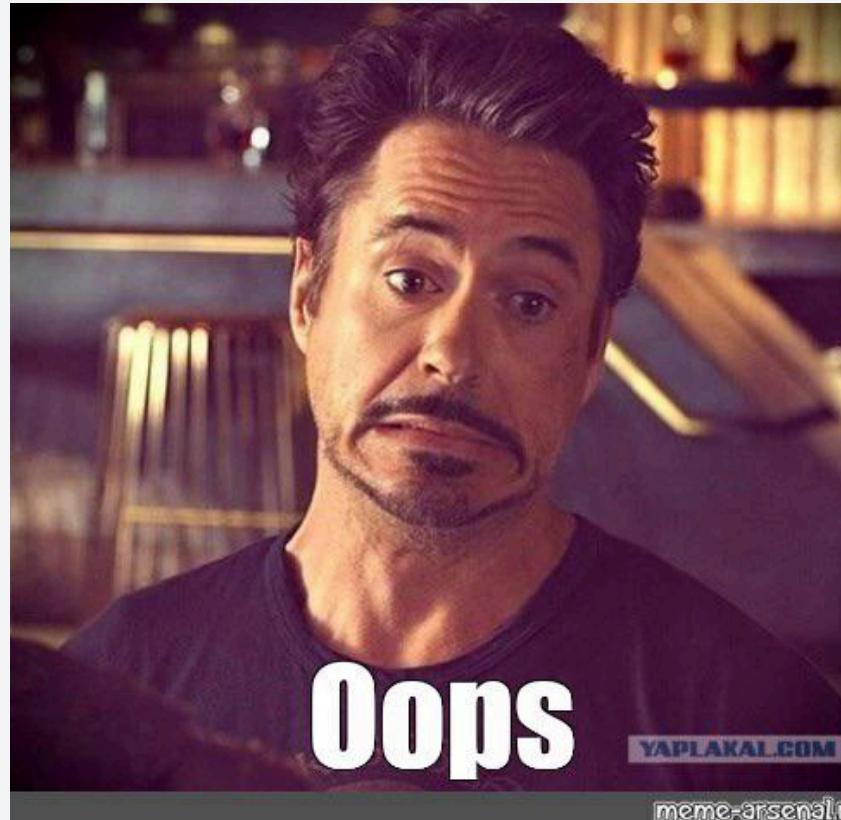
test

test

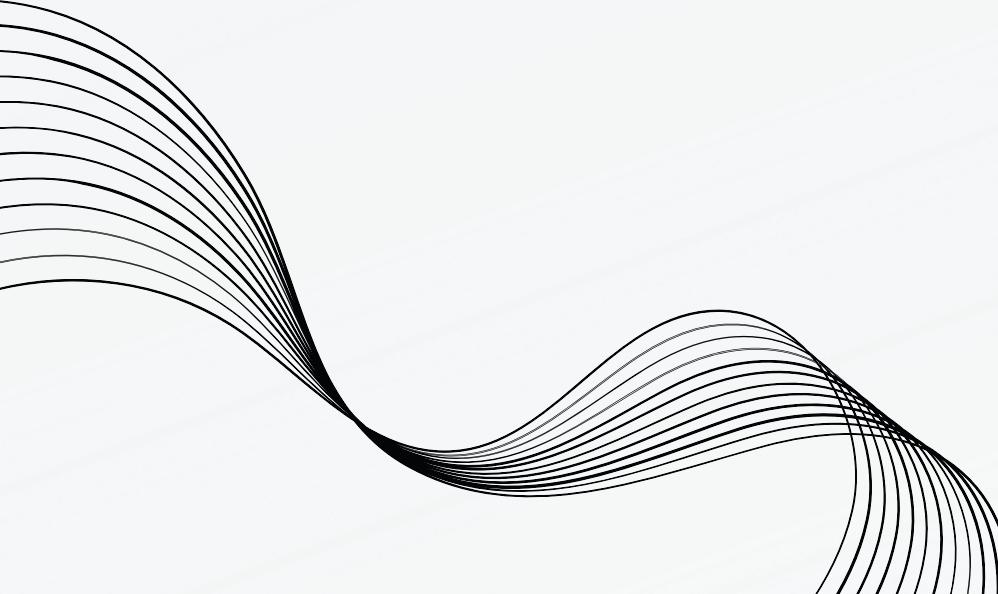
Submit

when we try to interact with the database, the connection with the pod gets lost:

```
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
E0109 23:30:26.490146    21804 portforward.go:409] an error occurred forwarding 8080 -> 8080: error forwarding port 8080 to pod 45efe105776168c7b33044ce4581f888e653ee3f82ecbc58c79655980b00a8a3, uid : exit status 1: 2024/01/09 22:30:26 socat[354564] E connect(5, AF=2 127.0.0.1:8080, 16): Connection refused
error: lost connection to pod
PS C:\Users\LENOVO> |
```



6-REPOSITORIES



The gitlab and dockerhub repositories:

gitlab : https://gitlab.com/iccn_devsecops

dockerhub : <https://hub.docker.com/repositories/4bd3lgh4f0r>