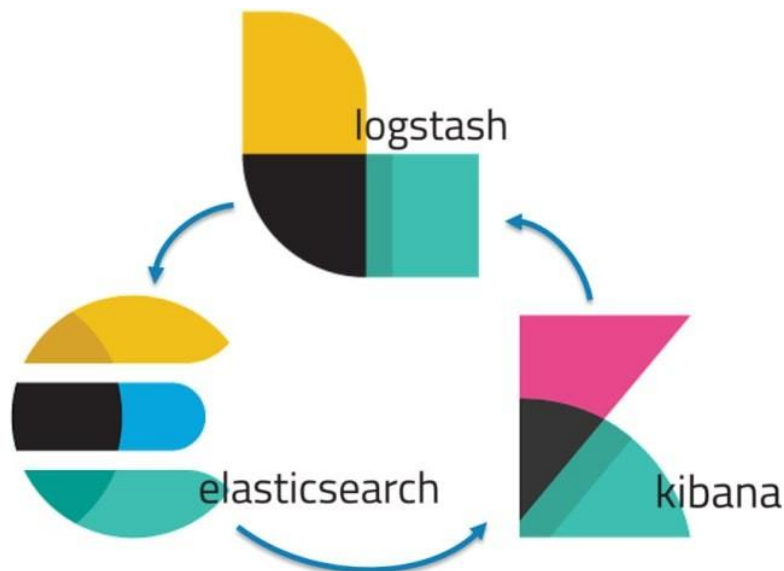


## Déploiement d'une Architecture DevSecOps avec SIEM : Comparaison VM vs Docker pour la Surveillance des Logs et la Détection d'Incidents



**Préparé par:**

Wissal BOUTAYEB

Fatima BOUYARMANE

**Encadré par:**

M. AMAMOU Ahmed

# PLAN

## 1. Introduction

- Contexte du projet
- Objectifs des TP
- Technologies utilisées (GitLab CI/CD, ELK, DHCP, Docker, Docker Conteneur etc.)

## 2. Architecture

- Schéma global de l'architecture (pour VM et Docker)

## 3. Implémentation TP1 - Architecture avec VMs

- Configuration des différentes VMs
- Pipeline CI/CD GitLab (avec fichier .gitlab-ci.yml)
- Configuration de Filebeat pour la collecte des logs
- Mise en place d'Elasticsearch et Kibana
- Simulation d'incidents de sécurité

## 4. Implémentation TP2 - Architecture avec Docker

- Configuration du réseau Docker
- Conteneurisation des services
- Adaptation du pipeline CI/CD
- Configuration de Filebeat en environnement conteneurisé
- Gestion des volumes et persistance des données

## Résultats et analyse

- Dashboards Kibana créés
- Incidents détectés (captures d'écran)
- Analyse des logs collectés
- Comparaison des performances VM vs Docker

## 6. Sécurité et bonnes pratiques

- Mesures de sécurité implémentées dans le pipeline CI/CD
- Résultats des outils SAST et Container scanning

## 1. Introduction:

Ce TP explore la mise en place d'une architecture DevSecOps intégrant un SIEM (Elasticsearch/Kibana) DevSecOps veiller les logs et détecter des incidents de sécurité.

### L'objectif :

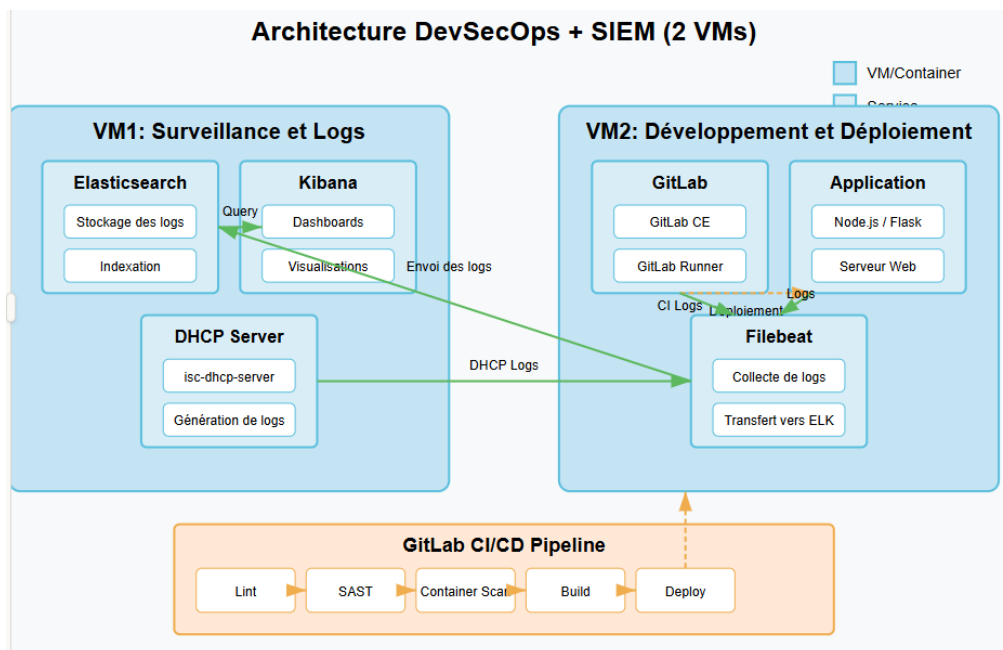
Déployer l'infrastructure sur des machines virtuelles (VMs) pour une isolation forte, simulant un environnement traditionnel.

Reproduire le même système avec des conteneurs Docker, afin de comparer flexibilité et reproductibilité.

Les outils comme Elasticsearch, Kibana, et Filebeat seront utilisés pour centraliser et analyser les logs, tandis que GitLab CI assurera une chaîne CI/CD sécurisée.

## 2. Architecture

### Schéma global de notre architecture déployée



Dans cette architecture, nous avons utilisé deux machines virtuelles distinctes, **VM1** et **VM2**. Sur la machine VM1, nous avons configuré Elasticsearch, Kibana et un serveur DHCP. Elasticsearch est utilisé pour l'indexation des données, Kibana pour leur visualisation, tandis que le serveur DHCP permet de générer des logs réseau. Sur la machine VM2, nous avons installé Filebeat, chargé de collecter les logs et de les transférer vers Elasticsearch. Par la suite, nous avons

déployé notre application à l'aide d'un pipeline CI/CD sécurisé, en utilisant GitLab et GitLab Runner, qui assurent respectivement la gestion du code et l'exécution automatisée du pipeline de déploiement.

### 3. Implémentation TP1 - Architecture avec VMs

#### VM 1:

On commence tout d'abord par installer **filebeat** pour la collecte des logs :

```
fera@fera-ubuntu:~$ curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch.gpg
fera@fera-ubuntu:~$ echo "deb [signed-by=/usr/share/keyrings/elasticsearch.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list
deb [signed-by=/usr/share/keyrings/elasticsearch.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main

fera@fera-ubuntu:~$ sudo apt update
sudo apt install filebeat -y
Hit:1 http://ma.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:3 http://ma.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 https://artifacts.elastic.co/packages/8.x/apt stable InRelease [3,248 B]
Get:5 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 Packages [74.7 kB]
Hit:6 http://ma.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:7 https://artifacts.elastic.co/packages/8.x/apt stable/main i386 Packages [3,396 B]
Fetched 338 kB in 3s (105 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
239 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  filebeat
0 upgraded, 1 newly installed, 0 to remove and 239 not upgraded.
Need to get 63.9 MB of archives.
After this operation, 239 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 filebeat amd64 8.18.1 [63.9 MB]
Fetched 63.9 MB in 43s (1,485 kB/s)
Selecting previously unselected package filebeat.
(Reading database ... 201762 files and directories currently installed.)
Preparing to unpack .../filebeat_8.18.1_amd64.deb ...
Unpacking filebeat (8.18.1) ...
Setting up filebeat (8.18.1) ...
```

Ensuite, on configure le fichier **filebeat.yml** :

Ce fichier de configuration définit comment Filebeat doit fonctionner pour envoyer des données à Elasticsearch.

#### Inputs (Entrées) :

Collecte les logs de GitLab Runner (**/var/log/gitlab-runner/\*.log**)

Collecte les logs DHCP depuis **/var/log/syslog** avec gestion des logs multilignes

#### Modules :

Charge les modules Filebeat (configurés dans modules.d/), bien que le rechargement automatique soit désactivé

#### Sortie (Output) :

Envoie les données à Elasticsearch sur **http://192.168.199.147:9200** (sur VM2)

## Journalisation (Logging) :

Enregistre les logs de Filebeat dans `/var/log/filebeat/filebeat.log`

## Kibana :

Configure la connexion à Kibana sur **192.168.199.147:5601**(sur VM2)

```
GNU nano 6.2 /etc/filebeat/filebeat.yml *
# ===== Filebeat inputs =====
filebeat.inputs:

# Logs GitLab Runner
- type: log
  enabled: true
  paths:
    - /var/log/gitlab-runner/*.log
  fields:
    service: gitlab-runner
    environment: dev
# Logs DHCP (syslog)
- type: log
  enabled: true
  paths:
    - /var/log/syslog
  multiline.pattern: '^[A-Za-z]{3} [ 0-9]{2} '
  multiline.negate: true
  multiline.match: after

  fields:
    service: dhcp
    environment: dev
# ===== Filebeat modules =====
filebeat.config.modules:
  path: ${path.config}/modules.d/*.yml
  reload.enabled: false
# ===== Elasticsearch output =====
output.elasticsearch:
  hosts: ["http://192.168.199.147:9200"]
# ===== Logging =====
logging.level: info
logging.to_files: true
logging.files:
  path: /var/log/filebeat
  name: filebeat.log
  keepfiles: 7
  permissions: 0644
# ===== Kibana (optional) =====
setup.kibana:
  host: "192.168.199.147:5601"
```

Après avoir activer filebeat, on voit qu'il est running :

```
19:33:15 16 ملى fera-ubuntu rtlebeat[16685]: {"log_level": "info", "@timestamp": "2025-05-16T19:33:15.376+0100", "lo
19:33:15 16 ملى fera-ubuntu filebeat[16685]: {"log_level": "info", "@timestamp": "2025-05-16T19:33:15.376+0100", "lo
19:33:15 16 ملى fera-ubuntu filebeat[16685]: {"log_level": "info", "@timestamp": "2025-05-16T19:33:15.377+0100", "lo
19:33:15 16 ملى fera-ubuntu filebeat[16685]: {"log_level": "info", "@timestamp": "2025-05-16T19:33:15.380+0100", "lo
19:33:15 16 ملى fera-ubuntu filebeat[16685]: {"log_level": "info", "@timestamp": "2025-05-16T19:33:15.381+0100", "lo
19:33:15 16 ملى fera-ubuntu filebeat[16685]: {"log_level": "info", "@timestamp": "2025-05-16T19:33:15.383+0100", "lo
```

```

feraf@feraf-ubuntu:~$ sudo gitlab-ctl status
[sudo] password for fera:
Sorry, try again.
[sudo] password for fera:
run: alertmanager: (pid 53032) 190s; run: log: (pid 52158) 325s
run: gitally: (pid 53048) 189s; run: log: (pid 49233) 751s
run: gitlab-exporter: (pid 52965) 195s; run: log: (pid 51914) 358s
down: gitlab-kas: 0s, normally up, want up; run: log: (pid 49465) 714s
run: gitlab-workhorse: (pid 52927) 197s; run: log: (pid 51640) 391s
run: logrotate: (pid 49157) 775s; run: log: (pid 49166) 771s
run: nginx: (pid 52938) 196s; run: log: (pid 51741) 380s
run: node-exporter: (pid 52952) 196s; run: log: (pid 51825) 371s
run: postgres-exporter: (pid 53041) 189s; run: log: (pid 52228) 315s
run: postgresql: (pid 49286) 731s; run: log: (pid 49299) 728s
run: prometheus: (pid 52986) 194s; run: log: (pid 52059) 338s
run: puma: (pid 51434) 418s; run: log: (pid 51463) 414s
run: redis: (pid 49193) 764s; run: log: (pid 49205) 762s
run: redis-exporter: (pid 52970) 195s; run: log: (pid 51984) 349s
run: sidekiq: (pid 51517) 407s; run: log: (pid 51541) 403s

```

On a par la suite **reconfiguré GitLab** pour .

- La Détection de la plateforme Linux.
- Chargement et synchronisation des **cookbooks** nécessaires (comme gitlab, redis, nginx, postgresql, logrotate, gitlab-kas, etc.).

```
fera@fera-ubuntu:~$ sudo gitlab-ctl reconfigure
[2025-05-18T12:53:03+01:00] INFO: Started Cinc Zero at chefzero://localhost:1 with repository at /opt/gitlab/embedded
(One version per cookbook)
Cinc Client, version 18.3.0
Patents: https://www.chef.io/patents
Infra Phase starting
[2025-05-18T12:53:03+01:00] INFO: *** Cinc Client 18.3.0 ***
[2025-05-18T12:53:03+01:00] INFO: Platform: x86_64-linux
[2025-05-18T12:53:03+01:00] INFO: Cinc-client pid: 31326
/opt/gitlab/embedded/lib/ruby/gems/3.2.0/gems/ffi-yajl-2.6.0/lib/ffi_yajl/encoder.rb:42: warning: undefining the alloc
ator of T_DATA class FFI_Yajl::Ext::Encoder::YajlGen
[2025-05-18T12:53:06+01:00] INFO: Setting the run_list to ["recipe[gitlab]"] from CLI options
[2025-05-18T12:53:06+01:00] INFO: Run List is [recipe[gitlab]]
[2025-05-18T12:53:06+01:00] INFO: Run List expands to [gitlab]
[2025-05-18T12:53:06+01:00] INFO: Starting Cinc Client Run for fera-ubuntu
[2025-05-18T12:53:06+01:00] INFO: Running start handlers
[2025-05-18T12:53:06+01:00] INFO: Start handlers complete.
Resolving cookbooks for run list: ["gitlab"]
[2025-05-18T12:53:08+01:00] INFO: Loading cookbooks [gitlab@0.0.1, package@0.1.0, logrotate@0.1.0, postgresql@0.1.0, r
edis@0.1.0, monitoring@0.1.0, registry@0.1.0, mattermost@0.1.0, consul@0.1.0, gitally@0.1.0, praefect@0.1.0, gitlab-kas
@0.1.0, gitlab-pages@0.1.0, letsencrypt@0.1.0, nginx@0.1.0, runit@5.1.7, acme@4.1.6, crond@0.1.0]
Synchronizing cookbooks:
- gitlab (0.0.1)
- redis (0.1.0)
- package (0.1.0)
- postgresql (0.1.0)
- logrotate (0.1.0)
- monitoring (0.1.0)
- registry (0.1.0)
- mattermost (0.1.0)
- gitally (0.1.0)
- gitlab-kas (0.1.0)
- letsencrypt (0.1.0)
- nginx (0.1.0)
- runit (5.1.7)
- acme (4.1.6)
- crond (0.1.0)
```

```
fera@fera-ubuntu:~$ sudo gitlab-ctl restart
ok: run: alertmanager: (pid 41191) 0s
ok: run: gitally: (pid 41215) 0s
ok: run: gitlab-exporter: (pid 41232) 1s
ok: run: gitlab-kas: (pid 41240) 0s
ok: run: gitlab-workhorse: (pid 41252) 0s
ok: run: logrotate: (pid 41264) 1s
ok: run: nginx: (pid 41272) 0s
ok: run: node-exporter: (pid 41283) 1s
ok: run: postgres-exporter: (pid 41290) 0s
ok: run: postgresql: (pid 41298) 1s
ok: run: prometheus: (pid 41305) 0s
ok: run: puma: (pid 41317) 0s
ok: run: redis: (pid 41328) 0s
ok: run: redis-exporter: (pid 41337) 0s
ok: run: sidekiq: (pid 41351) 1s
```

- Par la suite on a Télécharger la dernière version du binaire **GitLab Runner** et on a enregistré dans **/usr/local/bin/** (emplacement standard pour les exécutables système).
- Par la suite on a Lancer l'enregistrement du Runner pour le connecter à notre projet GitLab.

```

fera@fera-ubuntu:~$ cd TaskTimer/
fera@fera-ubuntu:~/TaskTimer$ sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64
[sudo] password for fera:
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 106M 100 106M 0 0 166k 0 0:10:54 0:10:54 --:--:-- 154k^[[C^[[
fera@fera-ubuntu:~/TaskTimer$ sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64
[sudo] password for fera:
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
2 106M 2 2633k 0 0 170k 0 0:10:37 0:00:15 0:10:22 241k^C
fera@fera-ubuntu:~/TaskTimer$ sudo chmod +x /usr/local/bin/gitlab-runner
fera@fera-ubuntu:~/TaskTimer$
fera@fera-ubuntu:~/TaskTimer$ sudo gitlab-runner register
Segmentation fault (core dumped)
fera@fera-ubuntu:~/TaskTimer$
fera@fera-ubuntu:~/TaskTimer$
fera@fera-ubuntu:~/TaskTimer$ 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 Time Current
Dload Upload Total Spent Left Speed
100 106M 100 106M 0 0 194k 0 0:09:19 0:09:19 --:--:-- 199k
fera@fera-ubuntu:~/TaskTimer$ sudo chmod +x /usr/local/bin/gitlab-runner

```

- Par la suite on a Configurer un nouveau Runner pour exécuter des jobs CI/CD depuis un dépôt GitLab.
- Entrées utilisateur :
- URL GitLab : <http://172.16.22.128/> (instance locale/privée).
- Token d'enregistrement : GRI348941VwQFBWew1BgDH8BTyB9.
- Description : fera-ubuntu: shell-runner.
- Tags : vn (pour cibler des jobs spécifiques).
- Executor : shell (exécution locale des commandes).

```

fera@fera-ubuntu:~/TaskTimer$ sudo gitlab-runner unregister --all-runners
[sudo] password for fera:
Runtime platform arch=amd64 os=linux pid=26209 revision=3e653c4e version=18.0.1
Running in system-mode.

WARNING: Unregistering all runners
Unregistering runner manager from GitLab succeeded runner=j6JdAt0-V
Updated /etc/gitlab-runner/config.toml
fera@fera-ubuntu:~/TaskTimer$ sudo gitlab-runner register
Runtime platform arch=amd64 os=linux pid=26461 revision=3e653c4e version=18.0.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com/):
http://172.16.22.128/
Enter the registration token:
GRI348941VwQFBNeVlBgDHsBFlyB9
Enter a description for the runner:
[fera-ubuntu]: shell-runner
Enter tags for the runner (comma-separated):
vn
Enter optional maintenance note for the runner:
shell
WARNING: Support for registration tokens and runner parameters in the 'register' command has been deprecated in GitLab
Runner 15.6 and will be replaced with support for authentication tokens. For more information, see https://docs.gitlab
b.com/ci/runners/new_creation_workflow/
Registering runner... succeeded runner=GR1348941VwQFBNeV
Enter an executor: parallels, virtualbox, docker, docker-windows, docker-autoscaler, custom, shell, kubernetes, instan
ce, ssh, docker+machine:
shell
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"
fera@fera-ubuntu:~/TaskTimer$ sudo gitlab-runner status
Runtime platform arch=amd64 os=linux pid=30879 revision=3e653c4e version=18.0.1
gitlab-runner: Service is running

```




On par la suite développé Cette application qui est un **serveur web minimaliste** développé avec **Flask**

```
app.py 383 B
1 from flask import Flask
2 import logging
3
4 app = Flask(__name__)
5
6 logging.basicConfig(filename='logs/app.log', level=logging.INFO)
7
8 @app.route('/')
9 def hello():
10     app.logger.info("Visited home page")
11     return "Hello from Flask!"
12
13 @app.route('/ping')
14 def ping():
15     app.logger.info("Ping request")
16     return "Pong"
17
18 if __name__ == '__main__':
19     app.run(host='0.0.0.0', port=5000)
```

- **Le fichier .gitlab-ci.yml:** ce fichier contient
- **1 Stages (Étapes) :**  
lint → build → sast → container\_scan → test → deploy  
Chaque stage exécute des tâches spécifiques dans l'ordre défini.
- **2. Jobs (Tâches) Clés**  
**Lint**  
Vérifie la qualité du code Python avec flake8.  
Utilise une image Python 3.11.
- **SAST (Analyse Statique de Sécurité)**  
• Installe des outils de sécurité (semgrep) pour détecter des vulnérabilités dans le code.
- Configure OpenTelemetry pour la supervision.
- **Container Scan**

- Utilise **Trivy** pour scanner l'image Docker à la recherche de vulnérabilités critiques.
- **Build**
- Étape simplifiée (pourrait compiler ou packager l'application).
- **Test**  
Lance les tests Python avec `pytest`.
- **Deploy**
- Déploie uniquement sur la branche `main` (exécution conditionnelle).

```
 .gitlab-ci.yml 1.43 KIB
1  stages:
2    - lint
3    - build
4    - sast
5    - container_scan
6
7    - test
8    - deploy
9
10 lint:
11   stage: lint
12   image: python:3.11
13   tags:
14     - shell
15   before_script:
16     - pip install flake8
17   script:
18     - echo "Running linter..."
19     - flake8 . || true
20
21 sast:
22   stage: sast
23   image: python:3.11
24   tags:
25     - shell
26   before_script:
27     # Update and install required dependencies
28     - apt update && apt install -y git
29
30     # Upgrade pip
31     - pip install --upgrade pip
32
33     # Install required OpenTelemetry dependencies
34     - pip install \
35       opentelemetry-instrumentation==0.54b1 \
36       opentelemetry-instrumentation-requests==0.54b1 \
37       opentelemetry-semantic-conventions==0.54b1 \
38       opentelemetry-util-http==0.54b1
39 --
```


```
40     # Install Semgrep and fix PATH issue
41     - pip install "semgrep==1.121.0" --break-system-packages
42     - export PATH="$HOME/.local/bin:$PATH"
43     script:
44     - semgrep --config=p/ci
45
46     container_scan:
47     stage: container_scan
48     image: aquasec/trivy:latest
49     tags:
50     - shell
51     script:
52     - trivy image --exit-code 1 --severity HIGH,CRITICAL your-image-name || true
53
54     build:
55     stage: build
56     tags:
57     - shell
58     script:
59     - echo "Building the project..."
60
61     test:
62     stage: test
63     image: python:3.11
64     tags:
65     - shell
66     before_script:
67     - pip install pytest
68     script:
69     - echo "Running tests..."
70     - pytest || true
71
72
73     deploy:
74     stage: deploy
75     tags:
76     - shell
77     only:
78     - main
79     script:
80     - echo "Deploying the app..."
```

 Dockerfile  330 B

```
1  # Utilise une image officielle Python
2  FROM python:3.11-slim
3
4  # Définir le répertoire de travail
5  WORKDIR /app
6
7  # Copier les fichiers
8  COPY requirements.txt .
9  RUN pip install --no-cache-dir -r requirements.txt
10
11 COPY . .
12
13 # Exposer le port utilisé par Flask
14 EXPOSE 5000
15
16 # Commande pour lancer l'app Flask
17 CMD ["python", "app.py"]
18
```

Fera was / TaskTimer / Pipelines / #19

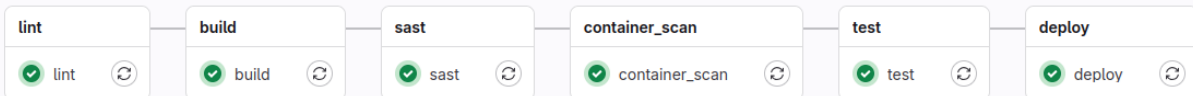
## Update .gitlab-ci.yml file

✓ Passed Fera was created pipeline for commit `6cb6c92f`  4 minutes ago, finished 1 minute ago

For `main`

latest branch 6 jobs 2 minutes 55 seconds, queued for 2 seconds

Pipeline Jobs 6 Tests 0



## Implémentation TP1 - Architecture avec VMs

### VM2

"La première étape consiste à installer **Elasticsearch** (pour l'indexation des données) et **Kibana** (pour leur visualisation).

```
wissal@wissal-virtual-machine:~$  
wissal@wissal-virtual-machine:~$ sudo apt-get -q0 - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -  
sudo apt install apt-transport-https  
echo "deb https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list  
sudo apt update  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).  
OK  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
apt-transport-https is already the newest version (2.4.13).  
0 upgraded, 0 newly installed, 0 to remove and 238 not upgraded.  
deb https://artifacts.elastic.co/packages/8.x/apt stable main  
Hit:1 http://ma.archive.ubuntu.com/ubuntu jammy InRelease  
Get:2 http://ma.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]  
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]  
Get:4 https://artifacts.elastic.co/packages/8.x/apt stable InRelease [3,248 B]  
Get:5 http://ma.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]  
Get:6 https://artifacts.elastic.co/packages/8.x/apt stable/main i386 Packages [3,396 B]  
Get:7 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 Packages [74.7 kB]  
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [54.5 kB]  
Get:9 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 DEP-11 Metadata [208 B]  
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [125 kB]  
Get:11 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 DEP-11 Metadata [208 B]  
Get:12 http://ma.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [114 kB]  
Get:13 http://ma.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 DEP-11 Metadata [212 B]  
Get:14 http://ma.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [359 kB]  
Get:15 http://ma.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]  
Get:16 http://ma.archive.ubuntu.com/ubuntu jammy-backports/main amd64 DEP-11 Metadata [7,080 B]  
Get:17 http://ma.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 DEP-11 Metadata [212 B]  
Get:18 http://ma.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [24.1 kB]  
Get:19 http://ma.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 DEP-11 Metadata [212 B]  
Fetched 1,151 kB in 5s (245 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
238 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

```
wissal@wissal-virtual-machine:~$ sudo apt-get install elasticsearch kibana  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following NEW packages will be installed:  
  elasticsearch kibana  
0 upgraded, 2 newly installed, 0 to remove and 238 not upgraded.  
Need to get 1,009 MB of archives.  
After this operation, 2,336 MB of additional disk space will be used.  
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 elasticsearch amd64 8.18.1 [648 MB]  
Get:2 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 kibana amd64 8.18.1 [361 MB]  
Fetched 1,009 MB in 4min 55s (3,419 kB/s)  
Selecting previously unselected package elasticsearch.  
(Reading database ... 202971 files and directories currently installed.)  
Preparing to unpack .../elasticsearch_8.18.1_amd64.deb ...  
Creating elasticsearch group... OK  
Creating elasticsearch user... OK  
Unpacking elasticsearch (8.18.1) ...  
Selecting previously unselected package kibana.  
Preparing to unpack .../kibana_8.18.1_amd64.deb ...  
Unpacking kibana (8.18.1) ...  
Setting up elasticsearch (8.18.1) ...  
----- Security autoconfiguration information -----  
  
Authentication and authorization are enabled.  
TLS for the transport and HTTP layers is enabled and configured.
```

Ici on voit Une réponse JSON confirmant qu'Elasticsearch est opérationnel :

```
wissal@wissal-virtual-machine:~$ sudo chown -R elasticsearch:elasticsearch /etc/elasticsearch  
sudo chown -R elasticsearch:elasticsearch /var/lib/elasticsearch  
sudo chown -R elasticsearch:elasticsearch /var/log/elasticsearch  
wissal@wissal-virtual-machine:~$ sudo systemctl restart elasticsearch  
wissal@wissal-virtual-machine:~$ curl http://localhost:9200  
{  
  "name" : "fera-ubuntu",  
  "cluster_name" : "my-elasticsearch",  
  "cluster_uuid" : "oeZzShK8Q2mQEnQEJH-EAw",  
  "version" : {  
    "number" : "8.18.1",  
    "build_flavor" : "default",  
    "build_type" : "deb",  
    "build_hash" : "df116ec6455476a07daafc3ded80e2bb1a3385ed",  
    "build_date" : "2025-04-30T10:07:44.026929518Z",  
    "build_snapshot" : false,  
    "lucene_version" : "9.12.1",  
    "minimum_wire_compatibility_version" : "7.17.0",  
    "minimum_index_compatibility_version" : "7.0.0"  
  },  
  "tagline" : "You Know, for Search"  
}
```

```

fera@fera-ubuntu:~$ sudo rm -f /etc/elasticsearch/elasticsearch.keystore
sudo rm -f /etc/elasticsearch/certs/*
fera@fera-ubuntu:~$ sudo chown -R elasticsearch:elasticsearch /etc/elasticsearch
sudo chown -R elasticsearch:elasticsearch /var/lib/elasticsearch
sudo chown -R elasticsearch:elasticsearch /var/log/elasticsearch
fera@fera-ubuntu:~$ sudo systemctl restart elasticsearch
fera@fera-ubuntu:~$ sudo systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2025-05-18 16:07:35 +01; 1min 11s ago
     Docs: https://www.elastic.co
   Main PID: 77650 (java)
    Tasks: 81 (limit: 9382)
   Memory: 4.3G
      CPU: 1min 3.303s
   CGroup: /system.slice/elasticsearch.service
           └─77650 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.sc
           └─77733 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.p
           └─77794 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

16:06:58 18 م.ي fera-ubuntu systemd[1]: Starting Elasticsearch...
16:07:35 18 م.ي fera-ubuntu systemd[1]: Started Elasticsearch.
lines 1-15/15 (END)

```

### Le fichier de configuration elasticsearch.yml :

Dans ce fichier, nous avons défini le nom de notre cluster, le nom du nœud, ainsi que les chemins pour le stockage des données et des logs. Nous avons également configuré le port sur lequel Elasticsearch écoute. Le paramètre `network.host` a été défini pour permettre l'écoute sur toutes les interfaces réseau. De plus, nous avons désactivé certaines options de sécurité afin d'éviter les conflits lors de la communication entre les composants.

```

GNU nano 6.2 /etc/elasticsearch/elasticsearch.yml
cluster.name: my-elasticsearch
node.name: fera-ubuntu
path.data: /var/lib/elasticsearch
path.logs: /var/log/elasticsearch
network.host: 0.0.0.0
http.port: 9200
discovery.type: single-node

# Security - completely disabled
xpack.security.enabled: false
xpack.security.autoconfiguration.enabled: false

```

**Le fichier de configuration de Kibana** contient l'adresse d'Elasticsearch à laquelle Kibana va se connecter, le port sur lequel Kibana écoute, ainsi que le chemin des logs de Kibana.

```
# Adresse Elasticsearch à laquelle Kibana va se connecter
elasticsearch.hosts: ["http://localhost:9200"]

# Bind IP et port où Kibana écoute (optionnel, par défaut localhost:5601)
server.host: "0.0.0.0"
server.port: 5601

# Désactivation de la sécurité côté Kibana
xpack.security.enabled: false
logging:
  appenders:
    file:
      type: file
      fileName: /var/log/kibana/kibana.log
      layout:
        type: json
  root:
    appenders:
      - default
      - file
# policies:

wissal@wissal-virtual-machine: $ sudo systemctl status kibana
[sudo] password for wissal:
● kibana.service - Kibana
   Loaded: loaded (/lib/systemd/system/kibana.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2025-05-18 19:34:31 +01; 1h 1min ago
     Docs: https://www.elastic.co
    Main PID: 13010 (node)
      Tasks: 11 (limit: 8734)
     Memory: 700.5M
        CPU: 5min 48.451s
    CGroup: /system.slice/kibana.service
            └─13010 /usr/share/kibana/bin/../node/glibc-217/bin/node /usr/share/kibana/bin/../src/cli/dist

20:32:24 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:24.958+01:00][INFO ][plugins.fleet.fleet:delete-unenrolled-agents-task:1.0.0] [DeleteUnenrolledAgentsTask] r
20:32:25 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:25.554+01:00][INFO ][plugins.taskManager] Background task node "c89ec093-4760-4cbb-a7e1-7eb12fb2615a" now cl
20:32:26 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:26.308+01:00][WARN ][plugins.securitySolution.telemetry_events.sender] Error pingng telemetry services
20:32:26 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:26.309+01:00][INFO ][plugins.securitySolution.telemetry_events.sender.task] Cannot reach telemetry services
20:32:26 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:26.378+01:00][WARN ][plugins.securitySolution.telemetry_events.sender] Error pingng telemetry services
20:32:26 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:26.379+01:00][INFO ][plugins.securitySolution.telemetry_events.sender.task] Cannot reach telemetry services
20:32:26 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:26.424+01:00][WARN ][plugins.securitySolution.telemetry_events.sender] Error pingng telemetry services
20:32:26 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:26.438+01:00][INFO ][plugins.securitySolution.telemetry_events.sender.task] Cannot reach telemetry services
20:32:27 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:27.877+01:00][INFO ][status] Kibana is now available (was degraded)
20:32:28 18 wissal-virtual-machine kibana[13010]: [2025-05-18T20:32:28.539+01:00][INFO ][status.plugins.taskManager] taskManager plugin is now available: Task Manager is healthy
lines 1-21/21 (END)
```

## Installation et la configuration de Serveur DHCP:

```
wissal@w3:~$ sudo apt install isc-dhcp-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  liblrs-export161 libliscfg-export163
Suggested packages:
  isc-dhcp-server-ldap policycoreutils
The following NEW packages will be installed:
  isc-dhcp-server liblrs-export161 libliscfg-export163
0 upgraded, 3 newly installed, 0 to remove and 69 not upgraded.
Need to get 520 kB of archives.
After this operation, 1,866 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libliscfg-export163 amd64 1:9.11.16+dfsg-3-ubuntu1 [45.9 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 liblrs-export161 amd64 1:9.11.16+dfsg-3-ubuntu1 [18.6 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 isc-dhcp-server amd64 4.4.1-2.1ubuntu5.20.04.5 [455 kB]
Fetched 520 kB in 2s (261 kB/s)
Preconfiguring packages ...
```

## Le fichier de configuration **dhcpd.conf** contient

- **Durées de bail** : default-lease-time de 600 secondes et max-lease-time de 7200 secondes.
- Le serveur est défini comme **autoritaire**.
- Une plage d'adresses IP définie pour le sous-réseau **192.168.199.0/24**, allant de **192.168.199.150 à 192.168.199.200**.
- L'adresse du **routeur** est **192.168.199.1**.
- Le **serveur DNS** utilisé est **8.8.8.8** (Google DNS).

```

GNU nano 4.8 /etc/dhcp/dhcpd.conf
# dhcpd.conf
#
# Sample configuration file for ISC dhcpd
#
# Attention: If /etc/ltsp/dhcpd.conf exists, that will be used as
# configuration file instead of this file.
#
# option definitions common to all supported networks...
default-lease-time 600;
max-lease-time 7200;
authoritative;

subnet 192.168.199.0 netmask 255.255.255.0 {
    range 192.168.199.150 192.168.199.200;
    option routers 192.168.199.1;
    option domain-name-servers 8.8.8.8;
}

```

Dans ce fichier, on définit l'interface sur laquelle le serveur DHCP doit écouter.

```

GNU nano 4.8 /etc/default/isc-dhcp-server
# Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)
#
# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
#DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf
#
# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid
#
# Additional options to start dhcpd with.
# Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""
#
# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
# Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="ens33"
INTERFACESv6=""

```

```

● isc-dhcp-server.service - ISC DHCP IPv4 server
   Loaded: loaded (/lib/systemd/system/isc-dhcp-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2025-05-16 05:38:36 PDT; 9min ago
     Docs: man:dhcpd(8)
    Main PID: 14027 (dhcpd)
      Tasks: 4 (limit: 8160)
     Memory: 7.1M
    CGroup: /system.slice/isc-dhcp-server.service
            └─14027 dhcpd -user dhcpd -group dhcpd -f -4 -pf /run/dhcp-server/dhcpd.pid -cf /etc/dhcp/dhcpd.conf ens33

May 16 05:38:36 w3 dhcpd[14027]: PID file: /run/dhcp-server/dhcpd.pid
May 16 05:38:36 w3 dhcpd[14027]: Wrote 0 leases to leases file.
May 16 05:38:36 w3 dhcpd[14027]: Listening on LPF/ens33/00:0c:29:d8:33:98/192.168.199.0/24
May 16 05:38:36 w3 sh[14027]: Listening on LPF/ens33/00:0c:29:d8:33:98/192.168.199.0/24
May 16 05:38:36 w3 sh[14027]: Sending on LPF/ens33/00:0c:29:d8:33:98/192.168.199.0/24
May 16 05:38:36 w3 sh[14027]: Sending on Socket/fallback/fallback-net
May 16 05:38:36 w3 dhcpd[14027]: Sending on LPF/ens33/00:0c:29:d8:33:98/192.168.199.0/24
May 16 05:38:36 w3 dhcpd[14027]: Sending on Socket/fallback/fallback-net
May 16 05:38:36 w3 dhcpd[14027]: Server starting service.
May 16 05:43:23 w3 dhcpd[14027]: DHCPREQUEST for 192.168.199.147 from 00:0c:29:d8:33:98 via ens33: unknown lease 192.168.199.147.
w1ssal@w3:~$
w1ssal@w3:~$

```

Test de l'attribution dynamique des adresses IP par le serveur :

```

(kali@kali)-[~]
└─$ sudo dhclient -v eth0
Internet Systems Consortium DHCP Client 4.4.3-P1
Copyright 2004-2022 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/eth0/00:0c:29:90:2d:9e
Sending on LPF/eth0/00:0c:29:90:2d:9e
Sending on Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 6
DHCPOFFER of 192.168.199.150 from 192.168.199.147
DHCPREQUEST for 192.168.199.150 on eth0 to 255.255.255.255 port 67
DHCPACK of 192.168.199.150 from 192.168.199.147
bound to 192.168.199.150 -- renewal in 187 seconds.

```



```
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.199.150 netmask 255.255.255.0 broadcast 192.168.199.255
    inet6 fe80::ba0:9c2b:e:2a49 prefixlen 64 scopeid 0<20<link>
    ether 00:0c:29:90:2d:9e txqueuelen 1000 (Ethernet)
    RX packets 325840 bytes 484696653 (462.2 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 17042 bytes 1041240 (1016.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0









(kali@kali)-[~]
$
```

## 4. Implémentation TP2 - Architecture avec Docker

### Objectif :

on Doit reproduire le même scénario mais via des **\*\*conteneurs Docker isolés\*\***, facilitant le déploiement rapide et reproductible (ex. via `docker-compose` ou `Docker + bridge network`).

Voici Architecture comme demandée :

Name	Last commit	Last update
 app	Add new file	41 minutes ago
 dhcp-sim	Add new file	40 minutes ago
 .gitlab-ci.yml	Update .gitlab-ci.yml f...	29 seconds ago
 README.md	Initial commit	46 minutes ago
 docker-compose.y...	Add new file	38 minutes ago
 elasticsearch.yml	Add new file	6 minutes ago
 filebeat.yml	Add new file	39 minutes ago
 kibana.yml	Add new file	5 minutes ago

Voici le fichier **docker-compose.yml**:

**Ce fichier** configure un environnement **DevSecOps** complet avec plusieurs services conteneurisés interconnectés. Voici sa structure globale :

## 1. Services Principaux

**GitLab CE** (latest) Serveur GitLab auto-hébergé Accessible via :Web : 8080 (HTTP) et 443 (HTTPS) et SSH port 22

- Volumes persistants pour : configuration, logs et données
- **Application (Flask/Node)**  
Service personnalisé (construit via **./app/Dockerfile**)  
Expose le port 5000 , Montage du code source en volume (./app)
- **Elasticsearch 8.12.0 :**  
Mode single-node  
Sécurité désactivée (`xpack.security.enabled=false`)  
Port 9200 exposé
- **Kibana 8.12.0 :**  
Interface web sur 5601  
Connecté à Elasticsearch

### **Filebeat 8.12.0 :**

- Collecte les logs de l'application (`/var/log/app`)
- Configuration via `filebeat.yml`

## **. Réseau et Stockage**

- **Réseau dédié** (devsecops) pour isoler les services.

- Volumes nommés pour la persistance des données :
  - gitlab-config, gitlab-data, es-data, etc.

```

1  version: '3.8'
2
3  services:
4
5      #####
6      # GitLab (serveur)
7      #####
8      gitlab:
9          image: gitlab/gitlab-ce:latest
10         hostname: 'gitlab.local'
11         container_name: gitlab
12         ports:
13             - "8080:80"
14             - "443:443"
15             - "22:22"
16         volumes:
17             - gitlab-config:/etc/gitlab
18             - gitlab-logs:/var/log/gitlab
19             - gitlab-data:/var/opt/gitlab
20         networks:
21             - devsecops
22
23         #####
24         # App Flask ou Node
25         #####
26         app:
27             build:
28                 context: ./app
29             container_name: app
30             ports:
31                 - "5000:5000"
32             volumes:
33                 - ./app:/app
34                 - app-logs:/var/log/app
35             networks:
36                 - devsecops
37
38         #####
39         # Elasticsearch
40
41     elasticsearch:
42         image: docker.elastic.co/elasticsearch/elasticsearch:8.12.0
43         container_name: elasticsearch
44         environment:
45             - discovery.type=single-node
46             - xpack.security.enabled=false
47         ports:
48             - "9200:9200"
49         volumes:
50             - es-data:/usr/share/elasticsearch/data
51         networks:
52             - devsecops
53
54         #####
55         # Kibana
56         #####
57         kibana:
58             image: docker.elastic.co/kibana/kibana:8.12.0
59             container_name: kibana
60             ports:
61                 - "5601:5601"
62             depends_on:
63                 - elasticsearch
64             environment:
65                 - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
66             networks:
67                 - devsecops
68
69         #####
70         # Filebeat
71         #####
72         filebeat:
73             image: docker.elastic.co/beats/filebeat:8.12.0
74             container_name: filebeat
75             user: root
76             volumes:
77                 - ./filebeat.yml:/usr/share/filebeat/filebeat.yml:ro
78                 - app-logs:/var/log/app
79             depends_on:
80                 - elasticsearch
81                 - kibana

```

```

82     networks:
83     - devsecops
84
85     #####
86     # Simulateur DHCP / script log
87     #####
88     dhcp-sim:
89     build:
90     context: ./dhcp-sim
91     container_name: dhcp-sim
92     volumes:
93     - ./dhcp-sim:/dhcp
94     networks:
95     - devsecops
96     command: ["python3", "/dhcp/generate_logs.py"]
97
98 volumes:
99     gitlab-config:
100     gitlab-logs:
101     gitlab-data:
102     es-data:
103     app-logs:
104
105 networks:
106     devsecops:
107     driver: bridge

```

La configuration de fichier **filebeat.yml**:

**Entrées (Inputs) Collecte des logs depuis :**

**/logs/app/\*.log** (logs de l'application)

**/logs/dhcp/\*.log** (logs DHCP)

**Sorties (Outputs)**

- **Elasticsearch :**

Envoie les logs à <http://elasticsearch:9200> (service Elasticsearch du réseau Docker).

- **Kibana :**

Configure l'intégration avec Kibana (<http://kibana:5601>).

```
filebeat.yml 281 B
1 filebeat.inputs:
2   - type: log
3     paths:
4       - /logs/app/*.log
5       - /logs/dhcp/*.log
6     multiline.pattern: '^\['
7     multiline.negate: true
8     multiline.match: after
9
10  output.elasticsearch:
11    hosts: ["http://elasticsearch:9200"]
12
13
14  setup.kibana:
15    host: "http://kibana:5601"
16
```

La configuration de fichier **elasticsearch.yml**:

```
elasticsearch.yml 154 B
1 network.host: 0.0.0.0
2 http.port: 9200
3
4 # Désactiver la sécurité pour un environnement de test
5 xpack.security.enabled: false
6 discovery.type: single-node
```

La configuration de fichier **kibana.yml**:

kibana.yml 219 B

```
1 server.name: kibana
2 server.host: "0.0.0.0"
3 elasticsearch.hosts: [ "http://elasticsearch:9200" ]
4
5 # Désactiver l'authentification pour TP
6 xpack.security.enabled: false
7 monitoring.ui.container.elasticsearch.enabled: true
```

Voici le fichier **.gitlab-ci.yml** qui permet d'identifier tous les stages :lint,build ,sast,container\_scan,test et le deploy

.gitlab-ci.yml 1.30 KiB

```
1 stages:
2   - lint
3   - build
4   - sast
5   - container_scan
6   - test
7   - deploy
8
9 variables:
10  DOCKER_DRIVER: overlay2
11  DOCKER_BUILDKIT: 1
12
13
14 # Étape Lint (code quality)
15 lint:
16   stage: lint
17   image: python:3.11
18   tags:
19     - shell
20   before_script:
21     - pip install flake8
22   script:
23     - echo "Running linter..."
24     - flake8 . || true
```

```
25
26 build:
27     stage: build
28     image: docker:latest
29     services:
30         - docker:dind
31     tags:
32         - shell
33     script:
34         - export DOCKER_BUILDKIT=0
35         - docker build -t tp2-app:latest .
36
37 # Analyse de sécurité SAST (Semgrep)
38 sast:
39     stage: sast
40     image: python:3.11
41     tags:
42         - shell
43     before_script:
44         - pip install --upgrade pip
45         - pip install semgrep
46     script:
47         - echo "Running SAST..."
48         - semgrep --config=p/ci || true
49
50 # Scan de la sécurité de l'image Docker
51 container_scan:
52     stage: container_scan
53     image: docker:latest
54     services:
55         - docker:dind
56     tags:
57         - shell
58     script:
59         - echo "Scanning Docker image..."
60         - docker scan tp2-app:latest || true
61
62 # Tests (optionnel selon ton app)
63 test:
64     stage: test
65     image: python:3.11
66     tags:
67         - shell
68     script:
69         - echo "Running tests (placeholder)"
70         - pytest || echo "No tests found"
71
72 # Déploiement avec docker-compose
73 deploy:
74     stage: deploy
75     tags:
76         - shell
77     image: alpine
78     script:
79         - echo "Deploying container..."
80
```

---

# Update .gitlab-ci.yml file

✓ Passed Fera was created pipeline for commit 1a389f5b 44 seconds ago, finished 10 seconds ago

For main

latest branch 60 6 jobs 17 seconds, queued for 4 seconds

Pipeline Jobs 6 Tests 0





Ce projet DevSecOps + SIEM, décliné en deux phases (sur machines virtuelles puis sur conteneurs Docker), nous a permis de mettre en œuvre une **chaîne complète d'intégration et de déploiement continu sécurisée** tout en assurant la **collecte, l'analyse et la visualisation de logs de sécurité**.

Nous avons d'abord déployé manuellement les composants sur des **VMs**, permettant de comprendre les interactions réseau et les configurations système nécessaires. Ensuite, nous avons reproduit la même architecture via **Docker**, ce qui nous a offert une solution plus modulaire, rapide à déployer et plus proche des standards modernes du DevOps.

Les outils utilisés comme **GitLab CI/CD**, **Filebeat**, **Elasticsearch**, **Kibana** et le simulateur de logs réseau nous ont permis de :

- Mettre en place une **pipeline de sécurité automatisée** (linting, SAST, container scanning),
- **Superviser l'activité du système et de l'application**,
- **Détecter des anomalies réseau**, comme des scans ou attaques bruteforce,