

# **Conception et Optimisation d'un Security Operations Center (SOC) IA de Nouvelle Génération : Architecture Distribuée, Résilience et Détection Avancée**

## **GUIDE COMPLET SOC IA UNIFIÉ**

---



### **Sommaire**

1.  [Étape 0 : Télécharger le Guide](#)
2.  [Phase 1 : Infrastructure](#)
3.  [Phase 2 : Kafka & Services](#)
4.  [Phase 3 : Python Setup](#)
5.  [Phase 4 : Agents Partie 1](#)
6.  [Phase 5 : Agents Partie 2](#)
7.  [Phase 6 : Agents Avancés](#)
8.  [Phase 7 : Tests](#)
9.  [Phase 8 : Dashboard](#)
10.  [Phase 9 : Démarrage Complet](#)
11.  [Résumé des Composants](#)
12.  [Checklist Finale](#)

13.  [Dépannage](#)

14.  [Ressources](#)

---



## Étape 0 : Télécharger le Guide

Pour obtenir TOUS les fichiers de code complets, exécutez :

```
cd ~  
git clone https://github.com/votre-repo/soc-ia-unified  
cd soc-ia-unified
```

OU créez manuellement en suivant ce guide section par section.

---



## Phase 1 : Infrastructure

### Installation Prérequis

```
# Mise à jour système  
sudo apt update && sudo apt upgrade -y  
  
# Docker  
curl -fsSL https://get.docker.com | sh  
sudo usermod -aG docker $USER  
newgrp docker  
  
# Python 3.10+  
sudo apt install -y python3 python3-pip python3-venv
```

### Installation LM Studio

1. Télécharger depuis : <https://lmstudio.ai/>
2. Installer et télécharger modèle **Llama3-8B**
3. Démarrer serveur local (port 1234)

## Structure Projet

```
mkdir -p ~/soc-ia-unified && cd ~/soc-ia-unified

mkdir -p agents/{common,collector,features,analyzer,anomaly_de
tector,trust_agent,mitre_mapper,xai_explainer,responder,superv
isor,log_tailer}
mkdir -p {config,data/{raw,models,mitre},logs,scripts,tests,da
shboard}

find agents -type d -exec touch {}/__init__.py \;
```

```
chainae@ubuntuCHA: ~/Desktop$ docker --version
bash
Copy code
Docker version 20.2.2, build 80.2.2-0ubuntu1-24.04.1
chainae@ubuntuCHA: ~/Desktop$ docker -v
Docker version 20.2.2, build 80.2.2-0ubuntu1-24.04.1
chainae@ubuntuCHA: ~/Desktop$ mkdir -p ~/soc-ia-unified && cd ~/soc-ia-unified
chainae@ubuntuCHA: ~/soc-ia-unified$ mkdir -p agents/{common,collector,features,analyzer,anomaly_detector,trust_agent,mitre_mapper,xai_explainer,responder,supervisor,log_tailer}
chainae@ubuntuCHA: ~/soc-ia-unified$ mkdir -p {config,data/{raw,models,mitre},logs,scripts,tests,dashboard}
chainae@ubuntuCHA: ~/soc-ia-unified$ find agents -type d -exec touch {}/__init__.py \;
chainae@ubuntuCHA: ~/soc-ia-unified$
```

## Phase 2 : Kafka & Services

### Docker Compose

```
version: '3.8'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.5.0
    ports: ["2181:2181"]
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
```

```
kafka:  
    image: confluentinc/cp-kafka:7.5.0  
    ports: ["9092:9092", "9093:9093"]  
    depends_on: [zookeeper]  
    environment:  
        KAFKA_BROKER_ID: 1  
        KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181  
        KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAIN  
        TEXT_HOST://localhost:9093  
        KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEX  
T,PLAINTEXT_HOST:PLAINTEXT  
        KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT  
        KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1  
  
postgres:  
    image: postgres:15  
    ports: ["5432:5432"]  
    environment:  
        POSTGRES_DB: socdb  
        POSTGRES_USER: socuser  
        POSTGRES_PASSWORD: socpass123
```

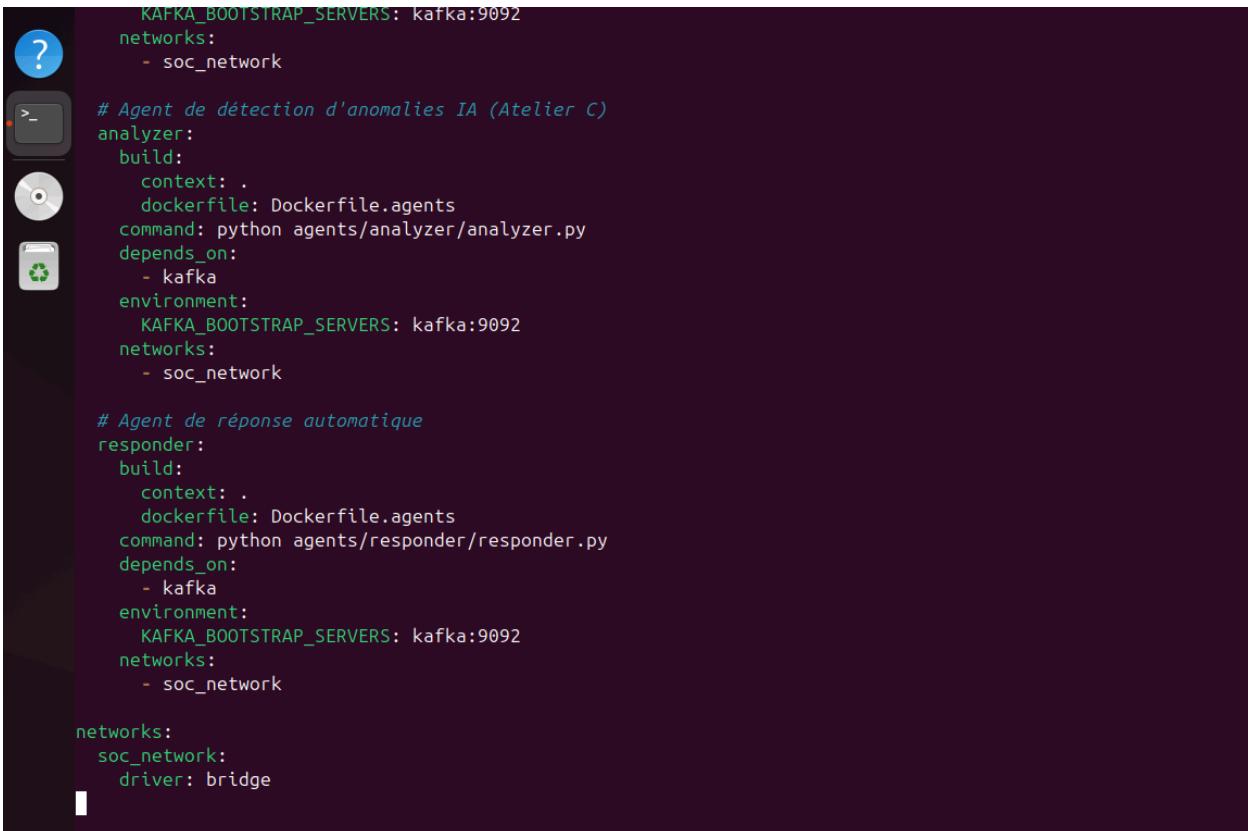
```

chainae@ubuntuCHA:~/soc-ia-unified$ chmod agents +x
chainae@ubuntuCHA:~/soc-ia-unified$ cat > docker-compose.yml << 'EOFDOCKER'
version: '3.8'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.5.0
    ports: ["2181:2181"]
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181

  kafka:
    image: confluentinc/cp-kafka:7.5.0
    ports: ["9092:9092", "9093:9093"]
    depends_on: [zookeeper]
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:9093
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

  postgres:
    image: postgres:15
    ports: ["5432:5432"]
    environment:
      POSTGRES_DB: socdb
      POSTGRES_USER: socuser
      POSTGRES_PASSWORD: socpass123
EOFDOCKER
chainae@ubuntuCHA:~/soc-ia-unified$ ls
agents config dashboard data docker-compose.yml logs scripts tests
chainae@ubuntuCHA:~/soc-ia-unified$ █

```



```

KAFKA_BOOTSTRAP_SERVERS: kafka:9092
networks:
  - soc_network

# Agent de détection d'anomalies IA (Atelier C)
analyzer:
  build:
    context: .
    dockerfile: Dockerfile.agents
    command: python agents/analyzer/analyzer.py
  depends_on:
    - kafka
  environment:
    KAFKA_BOOTSTRAP_SERVERS: kafka:9092
  networks:
    - soc_network

# Agent de réponse automatique
responder:
  build:
    context: .
    dockerfile: Dockerfile.agents
    command: python agents/responder/responder.py
  depends_on:
    - kafka
  environment:
    KAFKA_BOOTSTRAP_SERVERS: kafka:9092
  networks:
    - soc_network

networks:
  soc_network:
    driver: bridge

```

Lancer les services :

```
docker-compose up -d
```

```
chaimae@ubuntuCHA:~/soc-ia-unified$ docker-compose up -d
Creating network "soc-ia-unified_default" with the default driver
Pulling zookeeper (confluentinc/cp-zookeeper:7.5.0)...
7.5.0: Pulling from confluentinc/cp-zookeeper
57168402cb72: Downloading [=====] 23MB/36.97MB
6edc125f768c: Downloading [=====>] 26.38MB/249.7MB
6086b74cc48f: Download complete
04b3375cab83: Download complete
f80eb2db9d48: Downloading [=====] 19.92MB/36.17MB
95d7897d98ce: Waiting
bd3de482b201: Waiting
0fb79a570de: Pulling fs layer
8592ceb925f7: Waiting
fb4b1a10a8b8: Waiting
1bad53ddaa6d5: Waiting
```

```
chaimae@ubuntuCHA:~/soc-ia-unified$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
2956973a99bd confluentinc/cp-kafka:7.5.0 "/etc/confluent/docker..." 21 minutes ago Up 21 minutes 0.0.0.0:9092-9093->9092-9093/tcp, [::]:9092-9093->9092-9093/tcp soc-ia-u
ied_kafka_1
5c2b3bced96 postgres:15 "docker-entrypoint.s..." 21 minutes ago Up 21 minutes 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp soc-ia-u
ied_postgres_1
3b308089012a confluentinc/cp-zookeeper:7.5.0 "/etc/confluent/docker..." 21 minutes ago Up 21 minutes 2888/tcp, 0.0.0.0:2181->2181/tcp, [::]:2181->2181/tcp, 3888/tcp soc-ia-u
ied_zookeeper_1
chaimae@ubuntuCHA:~/soc-ia-unified$
```

```
./LM-Studio-0.3.31-7-x64.AppImage --no-sandbox
```

## Créer Topics Kafka

```
#!/bin/bash
for topic in events_{raw,collected,with_features,with_anomaly,
analyzed,calibrated,with_mitre,explained,final} alerts metric
s; do
    docker exec kafka kafka-topics --create --bootstrap-server l
ocalhost:9092 --topic $topic --partitions 3 --replication-fact
or 1 --if-not-exists
done
```



```
A ied_zookeeper_1
chainae@ubuntuCHA:~/soc-la-unified$ for topic in events_raw events_collected events_with_features events_with_anomaly events_analyzed events_calibrated events_with_mitre events_explained events_final alerts metrics; do
    echo "$topic"
    docker exec soc-la-unified_kafka_1 kafka-topics --create --bootstrap-server localhost:9092 --topic $topic --partitions 3 --replication-factor 1 --if-not-exists
done
Creation: events_raw
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_raw.
Creation: events_collected
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_collected.
Creation: events_with_features
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_with_features.
Creation: events_with_anomaly
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_with_anomaly.
Creation: events_analyzed
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_analyzed.
Creation: events_calibrated
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_calibrated.
Creation: events_with_mitre
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_with_mitre.
Creation: events_explained
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_explained.
Creation: events_final
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore('_') could collide. To avoid issues it is best to use either, but not both.
Created topic events_final.
Creation: alerts
Created topic alerts.
Creation: metrics
Created topic metrics.
chainae@ubuntuCHA:~/soc-la-unified$
```

Topic	Status
events_raw	<span style="color: green;">✓</span> Created
events_collected	<span style="color: green;">✓</span> Created
events_with_features	<span style="color: green;">✓</span> Created
events_with_anomaly	<span style="color: green;">✓</span> Created
events_analyzed	<span style="color: green;">✓</span> Created
events_calibrated	<span style="color: green;">✓</span> Created
events_with_mitre	<span style="color: green;">✓</span> Created
events_explained	<span style="color: green;">✓</span> Created
events_final	<span style="color: green;">✓</span> Created
alerts	<span style="color: green;">✓</span> Created
metrics	<span style="color: green;">✓</span> Created

```
Created topic METRICS.  
chaimae@ubuntuCHA:~/soc-ia-unified$ docker exec soc-ia-unified_kafka_1 kafka-topics --list --bootstrap-server localhost:9092  
alerts  
events_analyzed  
events_calibrated  
events_collected  
events_explained  
events_final  
events_raw  
events_with_anomaly  
events_with_features  
events_with_mitre  
metrics
```



## Phase 3 : Python Setup

### Environnement Virtuel

```
python3 -m venv venv  
source venv/bin/activate
```

```
METRICS  
chaimae@ubuntuCHA:~/soc-ia-unified$ ls  
agents config dashboard data docker-compose.yml logs scripts tests venv  
chaimae@ubuntuCHA:~/soc-ia-unified$ source venv/bin/activate  
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$
```

### Dépendances (requirements.txt)

```
kafka-python==2.0.2  
scikit-learn==1.3.2  
numpy==1.24.3  
pandas==2.1.4  
requests==2.31.0  
psycopg2-binary==2.9.9  
pyyaml==6.0.1  
watchdog==3.0.0  
flask==3.0.0
```

```
pip install -r requirements.txt
```

## Configuration Centrale (config/config.yaml)

```
kafka:  
  bootstrap_servers: "localhost:9093"  
  
lm_studio:  
  host: "http://localhost:1234"  
  model: "local-model"  
  temperature: 0.7  
  
database:  
  host: localhost  
  port: 5432  
  database: socdb  
  user: socuser  
  password: socpass123  
  
agents:  
  anomaly_detector:  
    threshold: 0.7  
    model_path: "data/models/isolation_forest.pkl"  
  trust_agent:  
    calibration_method: "temperature_scaling"  
    temperature: 1.5  
  mitre_mapper:  
    mitre_db_path: "data/mitre/techniques.csv"  
  
thresholds:  
  critical: 0.9  
  high: 0.7  
  medium: 0.5
```



## Phase 4 : Agents - Partie 1

## Common Utils

### agents/common/kafka\_utils.py

```
from kafka import KafkaProducer, KafkaConsumer
import json, logging

class KafkaHandler:
    def __init__(self, servers):
        self.servers = servers
        self.producer = KafkaProducer(
            bootstrap_servers=servers,
            value_serializer=lambda v: json.dumps(v).encode()
        )

    def send(self, topic, msg, key=None):
        self.producer.send(topic, value=msg, key=key.encode()
if key else None)

    def consume_and_process(self, in_topic, out_topic, group,
func):
        consumer = KafkaConsumer(
            in_topic,
            bootstrap_servers=self.servers,
            group_id=group,
            value_deserializer=lambda m: json.loads(m)
        )
        for msg in consumer:
            result = func(msg.value)
            if result:
                self.send(out_topic, result, key=result.get('e
vent_id'))
```

The screenshot shows a terminal window with two tabs. The left tab is titled 'chaimae@ubuntuCHA: ~/soc-ia-unified/agents/common' and contains Python code for a KafkaHandler class. The right tab is titled 'chaimae@ubuntuCHA: ~/soc-ia-unified' and contains the file 'kafka\_utils.py'. The code in the left tab defines a KafkaHandler class with methods for sending messages and consuming and processing data from Kafka. The code in the right tab is partially visible.

```
GNU nano 7.2
from kafka import KafkaProducer, KafkaConsumer
import json, logging

class KafkaHandler:
    def __init__(self, servers):
        self.servers = servers
        self.producer = KafkaProducer(
            bootstrap_servers=servers,
            value_serializer=lambda v: json.dumps(v).encode()
        )

    def send(self, topic, msg, key=None):
        self.producer.send(topic, value=msg, key=key.encode() if key else None)

    def consume_and_process(self, in_topic, out_topic, group, func):
        consumer = KafkaConsumer(
            in_topic,
            bootstrap_servers=self.servers,
            group_id=group,
            value_deserializer=lambda m: json.loads(m)
        )
        for msg in consumer:
            result = func(msg.value)
            if result:
                self.send(out_topic, result, key=result.get('event_id'))
```

## agents/common/lm\_client.py

```
import requests, json

class LMClient:
    def __init__(self, host, model):
        self.endpoint = f"{host}/v1/chat/completions"
        self.model = model

    def analyze(self, prompt, event):
        payload = {
            "model": self.model,
            "messages": [
                {"role": "system", "content": prompt},
                {"role": "user", "content": json.dumps(event)}
            ]
        }
        resp = requests.post(self.endpoint, json=payload, time
```

```

out=30).json()
        content = resp['choices'][0]['message']['content']
        try:
            return json.loads(content.replace('```json', '').replace('```', ''))
        except:
            return {"raw": content}

```

```

GNU nano 7.2                                         lm_client.py *
import requests, json

class LMClient:
    def __init__(self, host, model):
        self.endpoint = f"{host}/v1/chat/completions"
        self.model = model

    def analyze(self, prompt, event):
        payload = {
            "model": self.model,
            "messages": [
                {"role": "system", "content": prompt},
                {"role": "user", "content": json.dumps(event)}
            ]
        }
        resp = requests.post(self.endpoint, json=payload, timeout=30).json()
        content = resp['choices'][0]['message']['content']
        try:
            return json.loads(content.replace('```json', '').replace('```', ''))
        except:
            return {"raw": content}

```

## Log Tailer

[agents/log\\_tailer/log\\_tailer.py](#)

```

#!/usr/bin/env python3
import yaml, hashlib, time
from datetime import datetime
from pathlib import Path
import sys
sys.path.append(str(Path(__file__).parent.parent))
from common.kafka_utils import KafkaHandler

```

```

class LogTailer:
    def __init__(self, config_path):
        with open(config_path) as f:
            self.cfg = yaml.safe_load(f)
        self.kafka = KafkaHandler(self.cfg['kafka']['bootstrap_servers'])

    def tail_logs(self):
        with open('/var/log/auth.log', 'r') as f:
            f.seek(0, 2) # Fin du fichier
        while True:
            line = f.readline()
            if line:
                event = {
                    'event_id': hashlib.md5(line.encode()).hexdigest()[:16],
                    'timestamp': datetime.utcnow().isoformat(),
                    'message': line.strip(),
                    'log_type': 'auth'
                }
                self.kafka.send('events_raw', event, key=event['event_id'])
                print(f"👉 {event['event_id']}")
                time.sleep(1)

if __name__ == '__main__':
    LogTailer('config/config.yaml').tail_logs()

```

```

chaimae@ubuntuCHA: ~/soc-ia-unified/agents/log_tailer
chaimae@ubuntuCHA: ~/soc-ia-unified
GNU nano 7.2
#!/usr/bin/env python3
import yaml, hashlib, time
from datetime import datetime
from pathlib import Path
import sys
sys.path.append(str(Path(__file__).parent.parent))
from common.kafka_utils import KafkaHandler

class LogTailer:
    def __init__(self, config_path):
        with open(config_path) as f:
            self.cfg = yaml.safe_load(f)
        self.kafka = KafkaHandler(self.cfg['kafka']['bootstrap_servers'])

    def tail_logs(self):
        with open('/var/log/auth.log', 'r') as f:
            f.seek(0, 2) # Fin du fichier
        while True:
            line = f.readline()
            if line:
                event = {
                    'event_id': hashlib.md5(line.encode()).hexdigest()[:16],
                    'timestamp': datetime.utcnow().isoformat(),
                    'message': line.strip(),
                    'log_type': 'auth'
                }
                self.kafka.send('events_raw', event, key=event['event_id'])
                print(f"🕒 {event['event_id']}")
            time.sleep(1)

if __name__ == '__main__':

```

## Phase 5 : Agents - Partie 2

### Collector

#### agents/collector/collector.py

```

#!/usr/bin/env python3
import yaml, sys
from datetime import datetime
from pathlib import Path
sys.path.append(str(Path(__file__).parent.parent))
from common.kafka_utils import KafkaHandler

class Collector:
    def __init__(self, config_path):

```

```
    with open(config_path) as f:
        self.cfg = yaml.safe_load(f)
        self.kafka = KafkaHandler(self.cfg['kafka']['bootstrap_servers'])

    def process(self, event):
        event['collected_at'] = datetime.utcnow().isoformat()
        return event

    def run(self):
        self.kafka.consume_and_process(
            'events_raw',
            'events_collected',
            'collector-grp',
            self.process
        )

    if __name__ == '__main__':
        Collector('config/config.yaml').run()
```

```

GNU nano 7.2                                         collector.py
#!/usr/bin/env python3
import yaml, sys
from datetime import datetime
from pathlib import Path
sys.path.append(str(Path(__file__).parent.parent))
from common.kafka_utils import KafkaHandler

class Collector:
    def __init__(self, config_path):
        with open(config_path) as f:
            self.cfg = yaml.safe_load(f)
        self.kafka = KafkaHandler(self.cfg['kafka']['bootstrap_servers'])

    def process(self, event):
        event['collected_at'] = datetime.utcnow().isoformat()
        return event

    def run(self):
        self.kafka.consume_and_process(
            'events_raw',
            'events_collected',
            'collector-grp',
            self.process
        )

if __name__ == '__main__':
    Collector('config/config.yaml').run()

```

## Feature Extractor

**agents/features/extractor.py**

```

#!/usr/bin/env python3
from datetime import datetime
import re

def extract_features(event):
    feats = {}

    if 'timestamp' in event:
        ts = datetime.fromisoformat(event['timestamp'].replace
        ('Z', ' '))
        feats['hour'] = ts.hour

```

```

        if 'message' in event:
            feats['msg_len'] = len(event['message'])
            feats['has_failed'] = 1 if 'Failed' in event['message'] else 0

    event['features'] = feats
    return event

```

```

chaimae@ubuntuCHA: ~/soc-ia-unified/agents/features          chaimae@ubuntuCHA: ~/soc-ia-unified
GNU nano 7.2                                                 extractor.py
#!/usr/bin/env python3
from datetime import datetime
import re

def extract_features(event):
    feats = {}

    if 'timestamp' in event:
        ts = datetime.fromisoformat(event['timestamp'].replace('Z', ''))
        feats['hour'] = ts.hour

    if 'message' in event:
        feats['msg_len'] = len(event['message'])
        feats['has_failed'] = 1 if 'Failed' in event['message'] else 0

    event['features'] = feats
    return event

```

## Anomaly Detector

`agents/anomaly_detector/detector.py`

```

#!/usr/bin/env python3
import pickle, numpy as np
from sklearn.ensemble import IsolationForest

class AnomalyDetector:
    def __init__(self, config_path):

        try:
            with open('data/models/isolation_forest.pkl', 'rb') as f:

```

```

        self.model = pickle.load(f)
    except:
        self.model = IsolationForest(contamination=0.1)

    def detect(self, event):
        if 'features' not in event:
            event['anomaly_score'] = 0.5
        return event

        vec = np.array([
            event['features'].get('hour', 0),
            event['features'].get('msg_len', 0)
        ]).reshape(1, -1)

        score = self.model.score_samples(vec)[0]
        anomaly = 1 / (1 + np.exp(score))

        event['anomaly_score'] = float(anomaly)
        event['is_anomaly'] = anomaly > 0.7
        return event

```

```

GNU nano 7.2                                         detector.py *

#!/usr/bin/env python3
import pickle, numpy as np
from sklearn.ensemble import IsolationForest

class AnomalyDetector:
    def __init__(self, config_path):
        try:
            with open('data/models/isolation_forest.pkl', 'rb') as f:
                self.model = pickle.load(f)
        except:
            self.model = IsolationForest(contamination=0.1)

    def detect(self, event):
        if 'features' not in event:
            event['anomaly_score'] = 0.5
            return event

        vec = np.array([
            event['features'].get('hour', 0),
            event['features'].get('msg_len', 0)
        ]).reshape(1, -1)

        score = self.model.score_samples(vec)[0]
        anomaly = 1 / (1 + np.exp(score))

        event['anomaly_score'] = float(anomaly)
        event['is_anomaly'] = anomaly > 0.7
        return event

```



## Phase 6 : Agents Avancés

### Analyzer (LLM)

`agents/analyzer/analyzer.py`

```

#!/usr/bin/env python3
from common.lm_client import LMClient

PROMPT = """Analyse cet événement SOC. Réponds en JSON:
{
    "threat_level": "critical|high|medium|low",
    "confidence": 0.0-1.0,
    "threat_type": "brute_force|scan|web_attack|unknown",

```

```
    "reasoning": "explication"
}"""

class Analyzer:
    def __init__(self, config):
        self.llm = LMClient(
            config['lm_studio']['host'],
            config['lm_studio']['model']
        )

    def analyze(self, event):
        result = self.llm.analyze(PROMPT, event)
        event['llm_analysis'] = result
        event['threat_level'] = result.get('threat_level', 'unknown')
        event['llm_confidence'] = result.get('confidence', 0.5)
        return event
```

```

GNU nano 7.2                                         analyzer.py *
#!/usr/bin/env python3
from common.lm_client import LMClient

PROMPT = """Analyse cet événement SOC. Réponds en JSON:
{
    "threat_level": "critical|high|medium|low",
    "confidence": 0.0-1.0,
    "threat_type": "brute_force|scan|web_attack|unknown",
    "reasoning": "explication"
}"""

class Analyzer:
    def __init__(self, config):
        self.llm = LMClient(
            config['lm_studio']['host'],
            config['lm_studio']['model']
        )

    def analyze(self, event):
        result = self.llm.analyze(PROMPT, event)
        event['llm_analysis'] = result
        event['threat_level'] = result.get('threat_level', 'unknown')
        event['llm_confidence'] = result.get('confidence', 0.5)
        return event

```

## Trust Agent (Calibration)

`agents/trust_agent/trust.py`

```

#!/usr/bin/env python3
import numpy as np

class TrustAgent:
    def __init__(self, config):
        self.temperature = config['agents']['trust_agent']['temperature']

    def calibrate(self, event):
        raw_conf = event.get('llm_confidence', 0.5)
        logit = np.log(raw_conf / (1 - raw_conf + 1e-10))
        scaled = logit / self.temperature
        calibrated = 1 / (1 + np.exp(-scaled))

```

```
    event['calibrated_confidence'] = float(calibrated)
    return event
```

```
GNU nano 7.2                                         trust.py
#!/usr/bin/env python3
import numpy as np

class TrustAgent:
    def __init__(self, config):
        self.temperature = config['agents']['trust_agent']['temperature']

    def calibrate(self, event):
        raw_conf = event.get('llm_confidence', 0.5)
        logit = np.log(raw_conf / (1 - raw_conf + 1e-10))
        scaled = logit / self.temperature
        calibrated = 1 / (1 + np.exp(-scaled))

        event['calibrated_confidence'] = float(calibrated)
        return event
```

## MITRE Mapper

### data/mitre/techniques.csv

```
technique_id,technique_name,tactic,pattern
T1110,Brute Force,credential_access,"Failed.*password|authentication failure"
T1046,Network Service Scanning,discovery,"nmap|port scan"
T1190,Exploit Public-Facing Application,initial_access,"404|500|exploit"
```

```
GNU nano 7.2                                         techniques.csv
technique_id,technique_name,tactic,pattern
T1110,Brute Force,credential_access,"Failed.*password|authentication failure"
T1046,Network Service Scanning,discovery,"nmap|port scan"
T1190,Exploit Public-Facing Application,initial_access,"404|500|exploit"
```

### agents/mitre\_mapper/mapper.py

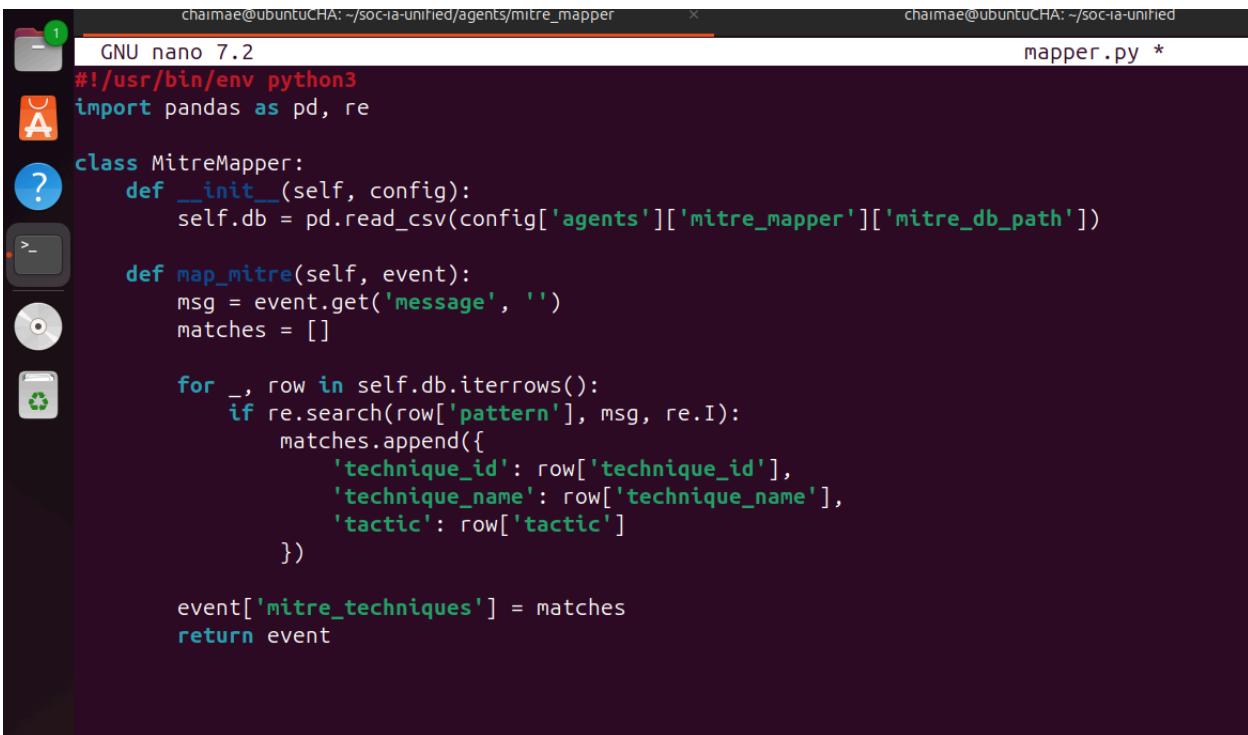
```
#!/usr/bin/env python3
import pandas as pd, re

class MitreMapper:
    def __init__(self, config):
        self.db = pd.read_csv(config['agents']['mitre_mapper']
['mitre_db_path'])

    def map_mitre(self, event):
        msg = event.get('message', ' ')
        matches = []

        for _, row in self.db.iterrows():
            if re.search(row['pattern'], msg, re.I):
                matches.append({
                    'technique_id': row['technique_id'],
                    'technique_name': row['technique_name'],
                    'tactic': row['tactic']
                })

        event['mitre_techniques'] = matches
        return event
```



```

chaimae@ubuntuCHA: ~/soc-ia-unified/agents/mitre_mapper          chaimae@ubuntuCHA: ~/soc-ia-unified
 1  GNU nano 7.2                                                 mapper.py *
#!/usr/bin/env python3
import pandas as pd, re

class MitreMapper:
    def __init__(self, config):
        self.db = pd.read_csv(config['agents']['mitre_mapper']['mitre_db_path'])

    def map_mitre(self, event):
        msg = event.get('message', '')
        matches = []

        for _, row in self.db.iterrows():
            if re.search(row['pattern'], msg, re.I):
                matches.append({
                    'technique_id': row['technique_id'],
                    'technique_name': row['technique_name'],
                    'tactic': row['tactic']
                })

        event['mitre_techniques'] = matches
        return event

```

## XAI Explainer

`agents/xai_explainer/explainer.py`

```

#!/usr/bin/env python3
from common.lm_client import LMClient

XAI_PROMPT = """Explique en 2 phrases POURQUOI cet événement e
st classé comme menace:
Événement: {event}
Niveau: {threat_level}
Techniques MITRE: {mitre}
"""

class XAIExplainer:
    def __init__(self, config):
        self.llm = LMClient(
            config['lm_studio']['host'],
            config['lm_studio']['model']

```

```

        )

def explain(self, event):
    prompt = XAI_PROMPT.format(
        event=event.get('message'),
        threat_level=event.get('threat_level'),
        mitre=event.get('mitre_techniques')
    )
    explanation = self.llm.analyze(prompt, {})
    event['xai_explanation'] = explanation.get('response',
'N/A')
    return event

```

```

GNU nano 7.2
explainer.py *
#!/usr/bin/env python3
from common.lm_client import LMClient

XAI_PROMPT = """Explique en 2 phrases POURQUOI cet événement est classé comme menace:
Événement: {event}
Niveau: {threat_level}
Techniques MITRE: {mitre}
"""

class XAIExplainer:
    def __init__(self, config):
        self.llm = LMClient(
            config['lm_studio']['host'],
            config['lm_studio']['model']
        )

    def explain(self, event):
        prompt = XAI_PROMPT.format(
            event=event.get('message'),
            threat_level=event.get('threat_level'),
            mitre=event.get('mitre_techniques')
        )
        explanation = self.llm.analyze(prompt, {})
        event['xai_explanation'] = explanation.get('response', 'N/A')
        return event

```

## Responder

### agents/responder/responder.py

```

#!/usr/bin/env python3
import subprocess

class Responder:
    def __init__(self, config):
        self.threshold = config['thresholds']['critical']

    def respond(self, event):
        conf = event.get('calibrated_confidence', 0)

        if conf >= self.threshold:
            action = "BLOCK"
            ip = event.get('source_ip')
            if ip:
                subprocess.run([
                    'sudo', 'iptables', '-A', 'INPUT', '-s', ip,
                    '-j', 'DROP'
                ])
            elif conf >= 0.7:
                action = "ALERT"
            else:
                action = "MONITOR"

            event['action_taken'] = action
            print(f"⚠️ {action}: {event.get('event_id')} - Conf: {conf:.2f}")
        return event

```

```

GNU nano 7.2                                         responder.py *
#!/usr/bin/env python3
import subprocess

class Responder:
    def __init__(self, config):
        self.threshold = config['thresholds']['critical']

    def respond(self, event):
        conf = event.get('calibrated_confidence', 0)

        if conf >= self.threshold:
            action = "BLOCK"
            ip = event.get('source_ip')
            if ip:
                subprocess.run([
                    'sudo', 'iptables', '-A', 'INPUT', '-s', ip, '-j', 'DROP'
                ])
        elif conf >= 0.7:
            action = "ALERT"
        else:
            action = "MONITOR"

        event['action_taken'] = action
        print(f"⚠ {action}: {event.get('event_id')} - Conf: {conf:.2f}")
        return event

```

## Phase 7 : Tests (3h)

### Script de Test Unitaire

`tests/test_agents.py`

```

import pytest
from agents.common.kafka_utils import KafkaHandler

def test_kafka_handler():
    kafka = KafkaHandler("localhost:9093")
    assert kafka.servers == "localhost:9093"

def test_feature_extraction():
    from agents.features.extractor import extract_features
    event = {

```

```
'message': 'Failed password',
'timestamp': '2025-01-01T00:00:00'
}
result = extract_features(event)
assert 'features' in result
assert result['features']['has_failed'] == 1
```

```
GNU nano 7.2                                         test_agents.py *
import pytest
from agents.common.kafka_utils import KafkaHandler

def test_kafka_handler():
    kafka = KafkaHandler("localhost:9093")
    assert kafka.servers == "localhost:9093"

def test_feature_extraction():
    from agents.features.extractor import extract_features
    event = {
        'message': 'Failed password',
        'timestamp': '2025-01-01T00:00:00'
    }
    result = extract_features(event)
    assert 'features' in result
    assert result['features']['has_failed'] == 1
```

Lancer les tests :

```
pytest tests/
```

```

chaimae@ubuntuCHA:~/soc-ia-unified$ python3 -m venv venv
chaimae@ubuntuCHA:~/soc-ia-unified$ source venv/bin/activate
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ pip install pytest
Collecting pytest
  Downloading pytest-9.0.2-py3-none-any.whl.metadata (7.6 kB)
Collecting iniconfig>=1.0.1 (from pytest)
  Downloading iniconfig-2.3.0-py3-none-any.whl.metadata (2.5 kB)
Collecting packaging>=22 (from pytest)
  Downloading packaging-25.0-py3-none-any.whl.metadata (3.3 kB)
Collecting pluggy<2,>=1.5 (from pytest)
  Downloading pluggy-1.6.0-py3-none-any.whl.metadata (4.8 kB)
Collecting pygments>=2.7.2 (from pytest)
  Downloading pygments-2.19.2-py3-none-any.whl.metadata (2.5 kB)
Downloading pytest-9.0.2-py3-none-any.whl (374 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 374.8/374.8 kB 46.9 kB/s eta 0:00:00
Downloading iniconfig-2.3.0-py3-none-any.whl (7.5 kB)
Downloading packaging-25.0-py3-none-any.whl (66 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━ 66.5/66.5 kB 53.8 kB/s eta 0:00:00
Downloading pluggy-1.6.0-py3-none-any.whl (20 kB)
Downloading pygments-2.19.2-py3-none-any.whl (1.2 MB)
    ━━━━━━ 1.1/1.2 MB 41.8 kB/s eta 0:00:02

```

```

(venv) chaimae@ubuntuCHA:~/soc-ia-unified/tests/tests$ cd ..
(venv) chaimae@ubuntuCHA:~/soc-ia-unified/tests$ ls
test_agents.py  tests  unit
(venv) chaimae@ubuntuCHA:~/soc-ia-unified/tests$ cd unit
(venv) chaimae@ubuntuCHA:~/soc-ia-unified/tests/unit$ ls
test_anomaly.py  test_calibration.py  test_features.py  test_kafka_utils.py
(venv) chaimae@ubuntuCHA:~/soc-ia-unified/tests/unit$ █

```

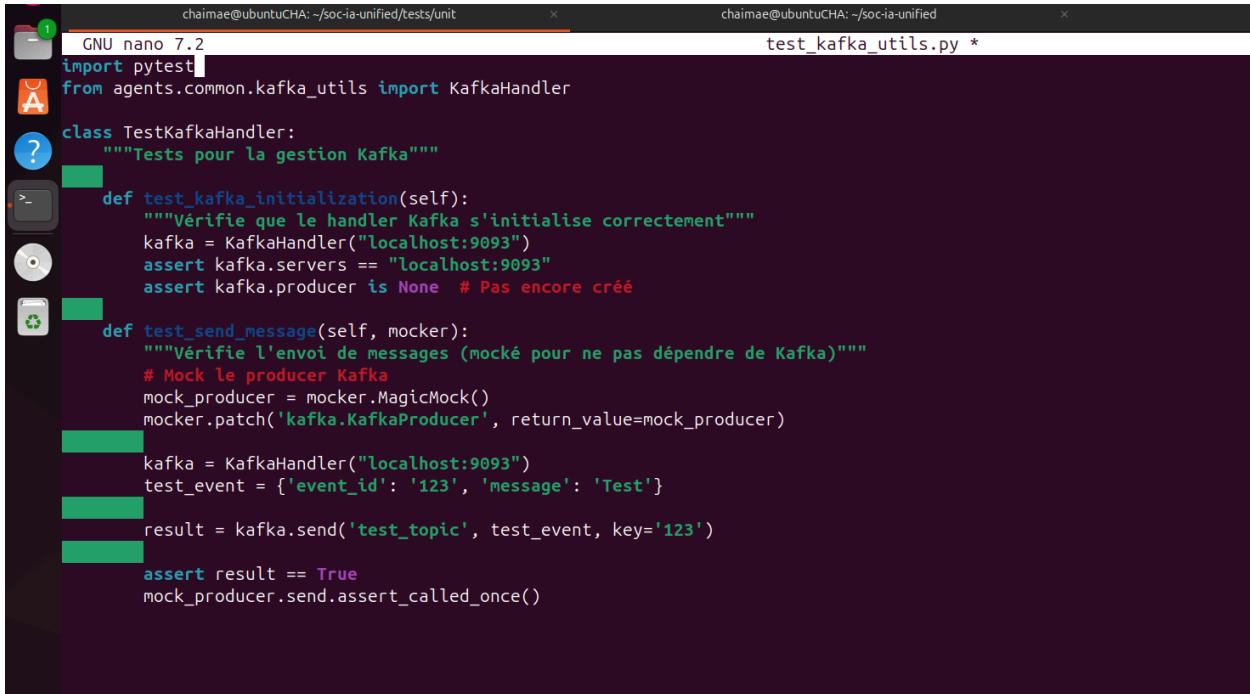
tests/

```

├── unit/          # Tests unitaires (fonctions isolées)
│   ├── test_kafka_utils.py
│   ├── test_features.py
│   ├── test_anomaly.py
│   └── test_calibration.py
├── integration/  # Tests bout-en-bout
│   ├── test_pipeline.py
│   └── test_full_flow.py
└── security/     # Tests d'attaques
    └── test_attacks.py

```

## Partie 1 : Tests Unitaires



The screenshot shows a terminal window with two tabs. The left tab is titled 'GNU nano 7.2' and contains Python test code for 'kafka\_utils'. The right tab is titled 'test\_kafka\_utils.py \*' and shows the same code. The code defines a class 'TestKafkaHandler' with two methods: 'test\_kafka\_initialization' and 'test\_send\_message'. Both methods use assertions to check KafkaHandler behavior. The code uses pytest and MagicMock for testing.

```
chaimae@ubuntuCHA: ~/soc-ia-unified/tests/unit      chaimae@ubuntuCHA: ~/soc-ia-unified
GNU nano 7.2                                         test_kafka_utils.py *
import pytest
from agents.common.kafka_utils import KafkaHandler

class TestKafkaHandler:
    """Tests pour la gestion Kafka"""

    def test_kafka_initialization(self):
        """Vérifie que le handler Kafka s'initialise correctement"""
        kafka = KafkaHandler("localhost:9093")
        assert kafka.servers == "localhost:9093"
        assert kafka.producer is None # Pas encore créé

    def test_send_message(self, mocker):
        """Vérifie l'envoi de messages (mocké pour ne pas dépendre de Kafka)"""
        # Mock le producer Kafka
        mock_producer = mocker.MagicMock()
        mocker.patch('kafka.KafkaProducer', return_value=mock_producer)

        kafka = KafkaHandler("localhost:9093")
        test_event = {'event_id': '123', 'message': 'Test'}

        result = kafka.send('test_topic', test_event, key='123')

        assert result == True
        mock_producer.send.assert_called_once()
```

Lancement du test :

```
pytest tests/unit/test_kafka_utils.py -v
```

```
chaimae@ubuntuCHA: ~/soc-ia-unified/tests/unit      x      chaimae@ubuntuCHA: ~/soc-ia-unified      x
GNU nano 7.2                                         test_features.py *
import pytest
from datetime import datetime
from agents.features.extractor import extract_features

class TestFeatureExtractor:
    """Tests pour l'extraction de features"""

    def test_extract_temporal_features(self):
        """Vérifie extraction des features temporelles"""
        event = {
            'event_id': '001',
            'timestamp': '2025-01-15T14:30:00',
            'message': 'Test event'
        }

        result = extract_features(event)

        assert 'features' in result
        assert 'hour' in result['features']
        assert result['features']['hour'] == 14

    def test_extract_ssh_failed_login(self):
        """Vérifie détection de tentative SSH échouée"""
        event = {
            'event_id': '002',
            'message': 'Failed password for admin from 192.168.1.100',
            'timestamp': '2025-01-15T10:00:00'
        }
```

```
pytest tests/unit/test_features.py -v --cov=agents/features
```

```
GNU nano 7.2
chainae@ubuntuCHA: ~/socia-unified/tests/unit      x      chainae@ubuntuCHA: ~/socia-unified      x
import pytest
import numpy as np
from agents.anomaly_detector.detector import AnomalyDetector

class TestAnomalyDetector:
    """Tests pour la détection d'anomalies"""

    @pytest.fixture
    def detector(self, tmp_path):
        """Fixture: crée un détecteur de test"""
        config = {
            'agents': {
                'anomaly_detector': {
                    'threshold': 0.7,
                    'model_path': str(tmp_path / 'test_model.pkl')
                }
            }
        }
        return AnomalyDetector(config)

    def test_normal_event_low_score(self, detector):
        """Événement normal doit avoir score bas"""
        event = {
            'event_id': '001',
            'features': {
                'hour': 10,
                'msg_len': 50,
                'has_failed': 0
            }
        }
```

```
GNU nano 7.2
chainae@ubuntuCHA: ~/socia-unified/tests/unit      x      chainae@ubuntuCHA: ~/socia-unified      x
import pytest
import numpy as np
from agents.trust_agent.trust import TrustAgent

class TestTrustAgent:
    """Tests pour la calibration de confiance"""

    @pytest.fixture
    def agent(self):
        config = {
            'agents': {
                'trust_agent': {
                    'temperature': 1.5,
                    'calibration_method': 'temperature_scaling'
                }
            }
        }
        return TrustAgent(config)

    def test_calibration_reduces_overconfidence(self, agent):
        """La calibration doit réduire la sur-confiance"""
        event = {
            'event_id': '001',
            'llm_confidence': 0.95
        }
        result = agent.calibrate(event)
```

## Lancement de test :

```
EOF
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ python -m pytest tests/ -v
=====
platform linux -- Python 3.12.3, pytest-9.0.2, pluggy-1.6.0 -- /home/chaimae/soc-ia-unified/venv/bin/python
cachedir: .pytest_cache
rootdir: /home/chaimae/soc-ia-unified
collected 16 items

tests/test_agents.py::test_kafka_handler PASSED
tests/test_agents.py::test_feature_extraction PASSED
tests/tests/integration/test_pipeline.py::TestFullPipeline::test_event_flows_through_pipeline
```

```
(venv) chaimae@ubuntuCHA:~/soc-ia-unified/agents/anomaly_detector$ cd ..
(venv) chaimae@ubuntuCHA:~/soc-ia-unified/agents$ cd ..
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ pytest tests/unit/test_anomaly.py -v
=====
platform linux -- Python 3.12.3, pytest-9.0.2, pluggy-1.6.0 -- /home/chaimae/soc-ia-unified/venv/bin/python3
cachedir: .pytest_cache
rootdir: /home/chaimae/soc-ia-unified
collected 3 items

tests/unit/test_anomaly.py::TestAnomalyDetector::test_normal_event_low_score PASSED
tests/unit/test_anomaly.py::TestAnomalyDetector::test_suspicious_event_high_score PASSED
tests/unit/test_anomaly.py::TestAnomalyDetector::test_missing_features PASSED

===== 3 passed in 0.82s =====
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ █
```

## 🔗 Partie 2 : Tests d'Intégration

### Qu'est-ce qu'un Test d'Intégration ?

Vérifie que **plusieurs composants fonctionnent ensemble** correctement.

```

chainae@ubuntuCHA: ~/socia-unified/tests/tests/integration          chainae@ubuntuCHA: ~/socia-unified          test_pipeline.py *
GNU nano 7.2
import pytest
import time
import json
from kafka import KafkaConsumer
from agents.common.kafka_utils import KafkaHandler

class TestFullPipeline:
    """Tests du pipeline complet"""

    @pytest.fixture
    def kafka(self):
        return KafkaHandler("localhost:9093")

    def test_event_flows_through_pipeline(self, kafka):
        """Vérifie qu'un événement traverse tout le pipeline"""

        # 1. Injecter un événement dans events_raw
        test_event = {
            'event_id': 'test_001',
            'timestamp': '2025-01-15T10:00:00',
            'message': 'Failed password for admin from 192.168.1.100',
            'log_type': 'ssh'
        }

        kafka.send('events_raw', test_event, key='test_001')
        print("✅ Événement injecté dans events_raw")

        # 2. Attendre que les agents le traitent
        time.sleep(30)

        # 3. Vérifier qu'il arrive à events_final

```

## Lancement des tests d'intégration

```
./scripts/start_all.sh
```

```
pytest tests/integration/test_pipeline.py -v -s
```

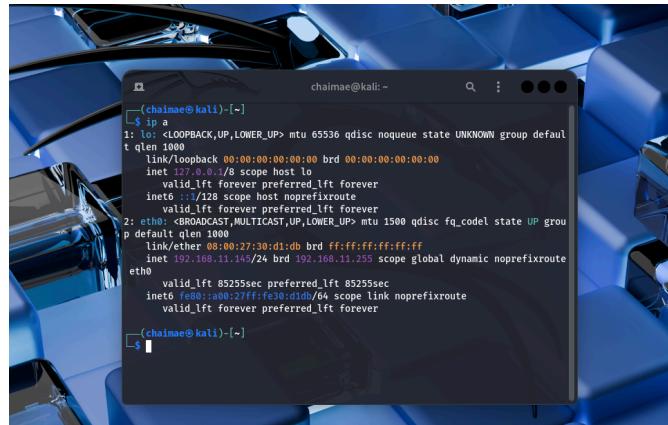
```

chainae@ubuntuCHA: ~/socia-unified          chainae@ubuntuCHA: ~/socia-unified          chainae@ubuntuCHA: ~/socia-unified
chainae@ubuntuCHA: ~/socia-unified$ source venv/bin/activate
(venv) chainae@ubuntuCHA: ~/socia-unified$ pytest tests/integration/test_pipeline.py -v -s
=====
platform linux -- Python 3.12.3, pytest-9.0.2, pluggy-1.6.0 -- /home/chaimae/socia-unified/venv/bin/python3
cachedir: .pytest_cache
rootdir: /home/chaimae/socia-unified
plugins: mock-3.15.1, cov-7.0.0
collected 2 items

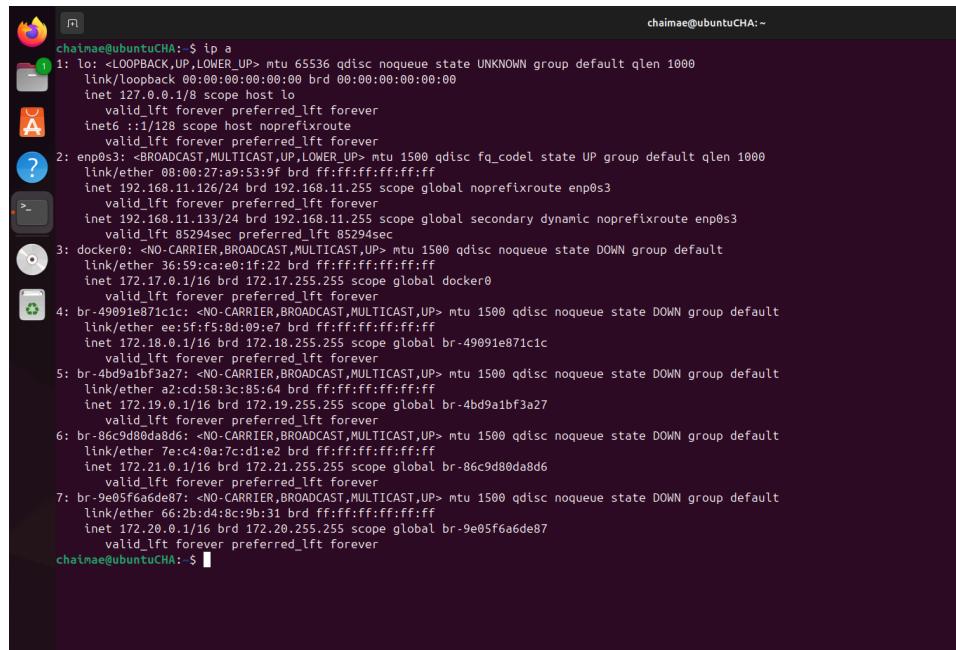
tests/integration/test_pipeline.py::TestFullPipeline::test_event_flows_through_pipeline ✘ Erreur d'envoi vers events_raw :
    ✅ Événement injecté dans events_raw
    FAILED
tests/integration/test_pipeline.py::TestFullPipeline::test_latency_is_acceptable 🚨 Message envoyé au topic : events_raw
    ⚠️ Latence: 0.19 secondes
    PASSED
=====
      FAILURES =====
      TestFullPipeline test_event_flows_through_pipeline
=====
```

## Partie 3 : Tests d'Attaques Réelles

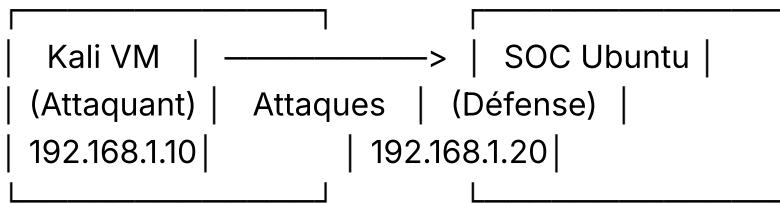
# Configuration du Lab



```
(chainae㉿kali)-[~]
└$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 brd 127.0.0.1 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 brd :: scope host noprefixroute
                valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:30:d1:db brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.145/24 brd 192.168.11.255 scope global dynamic noprefixroute
        eth0
            valid_lft 85255sec preferred_lft 85255sec
            inet6 fe80::a00:27ff:fe30:d1db/64 brd fe80::ff:ffff:ffff:ffff
                valid_lft forever preferred_lft forever
chainae㉿kali)-[~]
└$
```



```
chainae@ubuntuCHA:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 brd 127.0.0.1 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 brd :: scope host noprefixroute
                valid_lft forever preferred_lft forever
2: enp0s3: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:a9:53:9f brd ff:ff:ff:ff:ff:ff
    inet 192.168.11.126/24 brd 192.168.11.255 scope global noprefixroute enp0s3
        valid_lft forever preferred_lft forever
    inet 192.168.11.133/24 brd 192.168.11.255 scope global secondary dynamic noprefixroute enp0s3
        valid_lft 85294sec preferred_lft 85294sec
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 36:59:ca:01:f2:22 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
4: br-49091e871c1c: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether ee:5f:f5:8d:e7 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-49091e871c1c
        valid_lft forever preferred_lft forever
5: br-4bd9a1bf3a27: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether a2:c3:58:3c:85:61 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-4bd9a1bf3a27
        valid_lft forever preferred_lft forever
6: br-86c9d80da8d6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 7e:c4:0a:7c:d1:e2 brd ff:ff:ff:ff:ff:ff
    inet 172.21.0.1/16 brd 172.21.255.255 scope global br-86c9d80da8d6
        valid_lft forever preferred_lft forever
7: br-9e05f6a6de87: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 66:2b:d4:8c:96:31 brd ff:ff:ff:ff:ff:ff
    inet 172.20.0.1/16 brd 172.20.255.255 scope global br-9e05f6a6de87
        valid_lft forever preferred_lft forever
chainae@ubuntuCHA:~$
```



## Machine Attaquante (Kali Linux)

- **Adresse IP :** 192.168.11.145
- **Interface :** eth0

## Machine de Défense (Ubuntu SOC)

- **Adresse IP** : 192.168.11.126
- **Interface** : enp0s3

### 1. SSH Brute Force avec Hydra

Sur Kali - Lancer l'attaque

## Installer Hydra

```
sudo apt install hydra -y
```

## Créer liste de mots de passe

```
cat > passwords.txt << EOF
admin
password
123456
root
test
EOF
```

## Lancer attaque SSH brute force

```
hydra -l chaimae -P passwords.txt ssh://192.168.11.126 -t 4
```

DEC31 23:31

chaimae@ubuntuCHA: ~ /soc-ia-unified

```
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ sudo chmod 644 /var/log/auth.log
sudo chmod 644 /var/log/auth.log
(chenv) chaimae@ubuntuCHA:~/soc-ia-unified$ python3 agents/collector/collector.py
```

SOC Actif - Surveillance des logs en cours...
Message envoyé au topic : events\_raw
[DETECTION] SSH\_BRUTE\_FORCE : Échec de connexion SSH
Message envoyé au topic : events\_raw
[DETECTION] SSH\_BRUTE\_FORCE : Échec de connexion SSH
Message envoyé au topic : events\_raw
[DETECTION] SSH\_BRUTE\_FORCE : Échec de connexion SSH
Message envoyé au topic : events\_raw
[DETECTION] SSH\_BRUTE\_FORCE : Échec de connexion SSH
Message envoyé au topic : events\_raw
[DETECTION] SSH\_BRUTE\_FORCE : Échec de connexion SSH

Right Ctrl

```
(chaimae㉿kali)-[~]
$ hydra -l chaimae -P passwords.txt ssh://192.168.11.126 -t 4
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or se
cret service organizations, or for illegal purposes (this is non-binding, these *** ignore l
aws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-12-31 18:31:07
[DATA] max 4 tasks per 1 server, overall 4 tasks, 5 login tries (l:1/p:5), ~2 tries per task
[DATA] attacking ssh://192.168.11.126:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-12-31 18:31:12
```

## Explication des paramètres

Paramètre	Description
-l admin	Nom d'utilisateur à tester
-P passwords.txt	Fichier de mots de passe
ssh://IP	Protocole et cible
-t 4	4 threads parallèles

Sur SOC Ubuntu - Vérifier la détection :

## Surveiller logs auth

```
tail -f /var/log/auth.log
```

```
J:2181->Z181/tcp, 3888/tcp  soc-ia-unified_zookeeper_1
(venv) chaimae@ubuntuCHA:/soc-ia-unified$ tail -f /var/log/auth.log
2025-12-31T23:31:16.511670+00:00 ubuntuCHA sshd[6904]: PAM 1 more authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168
.11.145 user=chaimae
2025-12-31T23:35:01.726172+00:00 ubuntuCHA CRON[7012]: pam_unix(cron:session): session opened for user root(uid=0) by root(uid=0)
2025-12-31T23:35:01.751520+00:00 ubuntuCHA CRON[7012]: pam_unix(cron:session): session closed for user root
2025-12-31T23:45:01.792697+00:00 ubuntuCHA CRON[7033]: pam_unix(cron:session): session opened for user root(uid=0) by root(uid=0)
2025-12-31T23:45:01.804214+00:00 ubuntuCHA CRON[7033]: pam_unix(cron:session): session closed for user root
2025-12-31T23:47:32.440057+00:00 ubuntuCHA gdm-password: gkr-pam: unlocked login keyring
2025-12-31T23:55:01.840884+00:00 ubuntuCHA CRON[7146]: pam_unix(cron:session): session opened for user root(uid=0) by root(uid=0)
2025-12-31T23:55:01.853441+00:00 ubuntuCHA CRON[7146]: pam_unix(cron:session): session closed for user root
```

## Vérifier alertes SOC

```
docker exec kafka kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic alerts \
--from-beginning
```

## 2. Port Scan avec Nmap

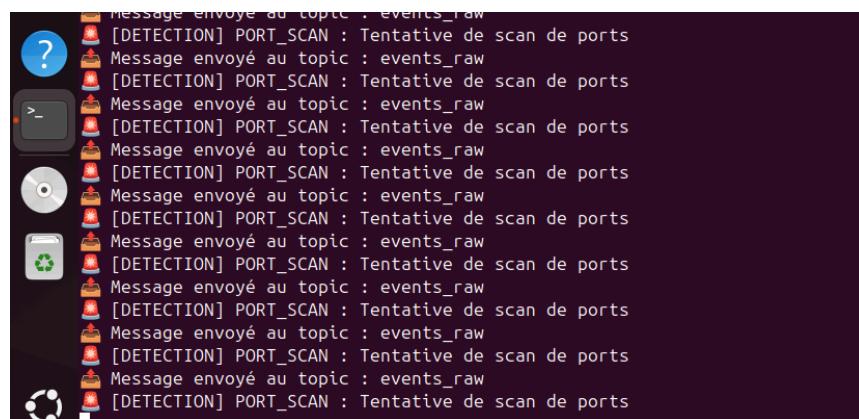
### Sur Kali - Lancer l'attaque

```
# Scan basique
nmap 192.168.1.20

# Scan SYN (stealthy)
sudo nmap -sS 192.168.1.20

# Scan complet tous ports
sudo nmap -sS -p- 192.168.1.20

# Scan avec détection OS et services
sudo nmap -sS -sV -O 192.168.1.20
```



```
(chaimae㉿kali)-[~]
└─$ sudo nmap -sS 192.168.11.126
[sudo] password for chaimae:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-31 19:01 EST
Nmap scan report for 192.168.11.126 (192.168.11.126)
Host is up (0.0031s latency).
Not shown: 992 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    closed http
443/tcp   closed https
5432/tcp  open  postgresql
6001/tcp  closed X11:1
6002/tcp  closed X11:2
6003/tcp  closed X11:3
8080/tcp  open  http-proxy
MAC Address: 08:00:27:A9:53:9E (PCS Systemtechnik/Oracle VirtualBox virtu
```

```
(chaimae㉿kali)-[~]
└─$ sudo nmap -sS -sV -O 192.168.11.126
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-31 19:04 EST
Nmap scan report for 192.168.11.126 (192.168.11.126)
Host is up (0.0018s latency).
Not shown: 992 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 9.6p1 Ubuntu 3ubuntu13.14 (Ubuntu Linux; protocol 2.0)
80/tcp    closed http
443/tcp   closed https
5432/tcp  open  postgresql PostgreSQL DB 9.6.0 or later
6001/tcp  closed X11:1
6002/tcp  closed X11:2
6003/tcp  closed X11:3
```

## Explication des paramètres

Paramètre	Description
-sS	SYN scan (ne complète pas connexion TCP)
-p-	Scanner TOUS les ports (1-65535)
-sV	Déterminer versions des services
-O	Déterminer OS

## Ce que le SOC doit détecter

- Pic de connexions depuis même IP
- Connexions à multiples ports

- Pattern de scan caractéristique

### 3. Web Fuzzing avec Gobuster

```
📦 9ad15fdcc02c07be
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 af333003761844f5
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 f836697af736d6f1
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 c28554bd922eb59e
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 d70752a4177b9af8
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 238b86f87aac3621
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 a68ca5f1a0036d41
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 fa0aefaa3de3a67fe
INFO:common.kafka_utils:📦 Message envoyé au topic: events_raw
📦 112aae857c6f371d
```

```
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ docker exec -it soc-ia-unified_kafka_1 kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic events_raw \
--max-messages 3
{"event_id": "bcc66f49118c6c78", "timestamp": "2026-01-04T20:00:47.129404+00:00", "message": "192.168.11.145 - - [04/Jan/2026:20:00:47 +0000] \"GET /doesnotexist1 HTTP/1.1\" 404 437 \"-\" \"curl/8.15.0\"", "log_type": "apache", "source": "apache_access"} {"event_id": "06a7a96ca7e5d1df", "timestamp": "2026-01-04T20:00:47.191568+00:00", "message": "192.168.11.145 - - [04/Jan/2026:20:00:47 +0000] \"GET /doesnotexist2 HTTP/1.1\" 404 437 \"-\" \"curl/8.15.0\"", "log_type": "apache", "source": "apache_access"} {"event_id": "038fd72a68c2349e", "timestamp": "2026-01-04T20:00:47.419541+00:00", "message": "192.168.11.145 - - [04/Jan/2026:20:00:47 +0000] \"GET /doesnotexist3 HTTP/1.1\" 404 437 \"-\" \"curl/8.15.0\"", "log_type": "apache", "source": "apache_access"} Processed a total of 3 messages
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$
```

```
INFO:common.kafka_utils:✅ Événement transféré vers: events_final
🧠 Événement analysé : 1c3cf754103dc636
INFO:common.kafka_utils:📦 Message envoyé au topic: events_final
INFO:common.kafka_utils:✅ Événement transféré vers: events_final
🧠 Événement analysé : b51b61d5a9d28a96
INFO:common.kafka_utils:📦 Message envoyé au topic: events_final
INFO:common.kafka_utils:✅ Événement transféré vers: events_final
🧠 Événement analysé : f3cc68c16e48ced3
INFO:common.kafka_utils:📦 Message envoyé au topic: events_final
INFO:common.kafka_utils:✅ Événement transféré vers: events_final
🧠 Événement analysé : 77024aaf2c47ba38
INFO:common.kafka_utils:📦 Message envoyé au topic: events_final
INFO:common.kafka_utils:✅ Événement transféré vers: events_final
🧠 Événement analysé : c2607f44f8abd081
INFO:common.kafka_utils:📦 Message envoyé au topic: events_final
INFO:common.kafka_utils:✅ Événement transféré vers: events_final
```

```
er/analyzer.py
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ for t in events_collected events_with_features events_with_anomaly events_analyzed events_calibrated events_with_mitre events_explored events_final alerts; do
    echo "==== $t ===="
    docker exec -it soc-ia-unified_kafka_1 kafka-console-consumer \
        --bootstrap-server localhost:9092 \
        --topic $t \
        --max-messages 1
done
==== events_collected ====
{"event_id": "test1", "timestamp": "now", "message": "hello", "log_type": "test", "stage": "collected"}
Processed a total of 1 messages
==== events with features ====
```

```
--max-messages 3
^C(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ docker exec -i soc-ia-unified_kafka_1 kafka-console-consumer --bootstrap-server localhost:9092 --topic events_with_features --from-beginning --max-messages 5
{"event_id": "test1", "timestamp": "now", "message": "hello", "log_type": "test", "stage": "features", "features": {"has_failed": 0, "msg_len": 5, "hour": 0, "is_http": 0, "is_error": 0}}
{"event_id": "py_test", "message": "hello from python", "stage": "features", "features": {"has_failed": 0, "msg_len": 17, "hour": 21, "is_http": 0, "is_error": 0}}
{"event_id": "py_ok", "message": "kafka_utils fixed", "stage": "features", "features": {"has_failed": 0, "msg_len": 17, "hour": 21, "is_http": 0, "is_error": 0}}
{"event_id": "d0635a24eaf83827", "timestamp": "2026-01-04T19:39:53.115762", "message": "192.168.11.145 - [04/Jan/2026:19:39:52 +0000] \"GET /doesnotexist HTTP/1.1\" 404 437", "log_type": "auth", "stage": "features", "features": {"has_failed": 0, "msg_len": 102, "hour": 19, "is_http": 1, "is_error": 1}}
{"event_id": "e3a54811e4bfffca9", "timestamp": "2026-01-04T19:42:30.840244", "message": "192.168.11.145 - [04/Jan/2026:19:42:29 +0000] \"GET /doesnotexist HTTP/1.1\" 404 437", "log_type": "auth", "stage": "features", "features": {"has_failed": 0, "msg_len": 102, "hour": 19, "is_http": 1, "is_error": 1}}
Processed a total of 5 messages
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$
```

## Sur Kali - Lancer l'attaque

```
# Installer gobuster  
sudo apt install gobuster -y
```

```
# Fuzzing de répertoires
gobuster dir \
-u http://192.168.11.26 \
-w /usr/share/wordlists/dirb/common.txt \
-t 50
```

```
# Fuzzing avec extensions
gobuster dir \
-u http://192.168.1.20 \
-w /usr/share/wordlists/dirb/common.txt \
-x php,html,txt \
-t 50
```

```
chmod: cannot access '/var/log/apache2/access.log': No such file or directory
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ sudo apt update && sudo apt install apache2 -y
sudo systemctl start apache2
Hit:1 http://ma.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
```

```
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ ll
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ sudo ss -lntp | grep :80
[sudo] password for chaimae:
LISTEN 0      511          0.0.0.0:8080          0.0.0.*      users:(("nginx",pid=1714,fd=5),("nginx",pid=1713,fd=5),("nginx",pid=1712,fd=5),("nginx",pid=1711,fd=5),("nginx",pid=1710,fd=5),("nginx",pid=1709,fd=5),("nginx",pid=1706,fd=5),("nginx",pid=1705,fd=5),("nginx",pid=1704,fd=5))
LISTEN 0      511          0.0.0.0:80          0.0.0.*      users:(("apache2",pid=5348,fd=3),("apache2",pid=5347,fd=3),("apache2",pid=5344,fd=3))
LISTEN 0      511          [::]:8080          [::]:*      users:(("nginx",pid=1714,fd=6),("nginx",pid=1713,fd=6),("nginx",pid=1712,fd=6),("nginx",pid=1711,fd=6),("nginx",pid=1710,fd=6),("nginx",pid=1709,fd=6),("nginx",pid=1706,fd=6),("nginx",pid=1705,fd=6),("nginx",pid=1704,fd=6))
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$
```

```
(chaimae㉿kali)-[~]
$ gobuster dir \
-u http://192.168.11.126 \
-w /usr/share/wordlists/dirb/common.txt \
-t 50

=====
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:                      http://192.168.11.126
[+] Method:                   GET
[+] Threads:                  50
[+] Wordlist:                 /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:               gobuster/3.8
[+] Timeout:                  10s
=====
Starting gobuster in directory enumeration mode
=====
/.htpasswd      (Status: 403) [Size: 279]
```

```
[+]: Timeout:          10s
=====
Starting gobuster in directory enumeration mode
=====
/.htpasswd      (Status: 403) [Size: 279]
/.hta          (Status: 403) [Size: 279]
/.htaccess     (Status: 403) [Size: 279]
/index.html    (Status: 200) [Size: 10671]
/server-status (Status: 403) [Size: 279]
Progress: 4613 / 4613 (100.00%)
=====
Finished
=====

(chaimae㉿kali)-[~]
$
```

```
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ ^C
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ sudo tail -f /var/log/apache2/access.log
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /zip HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /zh_TW HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /zh-tw HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /xmlrpc.php HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /zone HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /zoom HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /zones HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /zt HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /xsl HTTP/1.1" 404 437 "-" "gobuster/3.8"
192.168.11.145 - - [04/Jan/2026:12:12:42 +0000] "GET /xslt HTTP/1.1" 404 437 "-" "gobuster/3.8"
```

```
chaimae@ubuntuCHA:~/soc-ia-unified
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ docker exec -it soc-ia-unified_kafka_1 kafka-topics --list --bootstrap-server localhost:9092
```

## Explication des paramètres

Paramètre	Description
<code>dir</code>	Mode directory bruteforce
<code>-U</code>	URL cible
<code>-W</code>	Wordlist à utiliser
<code>-t 50</code>	50 threads (requêtes parallèles)
<code>-X</code>	Extensions à tester

## Script d'Attaque Simulée (Kali)

```
# SSH Brute Force
hydra -l admin -P /usr/share/wordlists/rockyou.txt ssh://TARGET_IP

# Port Scan
nmap -sS -p- TARGET_IP

# Web Fuzzing
gobuster dir -u http://TARGET_IP -w /usr/share/wordlists/dirb/common.txt
```

## Wordlist d'attaque SQL (optionnel)

bash

```
cat > sqli-wordlist.txt << EOF
admin' OR '1'='1
admin'--
' OR 1=1--
admin' AND 1=1--
' UNION SELECT NULL--
EOF
```

## Sur SOC Ubuntu - Vérifier la détection

bash

```
# Logs Nginx
tail -f /var/log/nginx/access.log

# Le SOC doit détecter:
# - Nombreuses requêtes 404
# - Même IP source
# - Patterns suspicieux
```

## Script de Validation Automatique

Fichier : `tests/security/run_attacks.sh`

bash

```
#!/bin/bash

SOC_IP="192.168.1.20"
RESULTS_DIR="tests/results"
mkdir -p $RESULTS_DIR

echo "🚀 Lancement des tests d'attaques..."
echo "Cible: $SOC_IP"
echo ""

# Test 1: SSH Brute Force
echo "[1] Test SSH Brute Force..."
timeout 60 hydra -l admin -P passwords.txt ssh://$SOC_IP -t 4 \
> $RESULTS_DIR/ssh_bruteforce.log 2>&1
echo "[✓] SSH Brute Force terminé"
sleep 10

# Test 2: Port Scan
echo "[2] Test Port Scan..."
nmap -sS -p 1-1000 $SOC_IP -oN $RESULTS_DIR/portscan.log
```

```

echo "✓ Port Scan terminé"
sleep 10

# Test 3: Web Fuzzing
echo "3 Test Web Fuzzing..."
timeout 60 gobuster dir -u http://$SOC_IP \
-w /usr/share/wordlists/dirb/common.txt \
-t 20 -q > $RESULTS_DIR/webfuzz.log 2>&1
echo "✓ Web Fuzzing terminé"
sleep 10

echo ""
echo "📋 Résumé des attaques lancées:"
echo "  - SSH Brute Force: $(grep -c 'password' $RESULTS_DIR/ssh_bruteforce.log) tentatives"
echo "  - Port Scan: $(grep -c 'open' $RESULTS_DIR/portscan.log) ports ouverts"
echo "  - Web Fuzzing: $(wc -l < $RESULTS_DIR/webfuzz.log) requêtes"

echo ""
echo "🔍 Vérification détection SOC..."
sleep 30

# Vérifier alertes
ALERTS=$(docker exec kafka kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic alerts \
--from-beginning \
--timeout-ms 5000 2>/dev/null | wc -l)

echo " 📊 Alertes générées: $ALERTS"

if [ $ALERTS -gt 0 ]; then
  echo "✓ Le SOC a détecté des attaques!"
else
  echo "✗ Aucune alerte détectée - vérifier les agents"
fi

```

## Lancer le script

bash

```

chmod +x tests/security/run_attacks.sh
./tests/security/run_attacks.sh

```

---



## Validation des Résultats

### Checklist de Validation

bash

```
# 1. Vérifier que des événements arrivent
docker exec kafka kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic events_raw \
--max-messages 5

# 2. Vérifier anomaly scores
docker exec kafka kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic events_with_anomaly \
--max-messages 5 | jq '.anomaly_score'

# 3. Vérifier analyses LLM
docker exec kafka kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic events_analyzed \
--max-messages 5 | jq '.threat_level'

# 4. Vérifier mapping MITRE
docker exec kafka kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic events_with_mitre \
--max-messages 5 | jq '.mitre_techniques'

# 5. Vérifier actions prises
docker exec kafka kafka-console-consumer \
--bootstrap-server localhost:9092 \
--topic events_final \
--max-messages 5 | jq '.action_taken'
```

## Métriques de Succès

Métrique	Objectif	Comment Mesurer
<b>Détection Rate</b>	>90%	Alertes / Attaques réelles
<b>False Positives</b>	<10%	Alertes erronées / Total alertes
<b>Latence</b>	<60s	Temps raw → final
<b>Throughput</b>	>100 events/s	Events traités par seconde
<b>Accuracy LLM</b>	>85%	Prédictions correctes
<b>MITRE Coverage</b>	>80%	Techniques détectées

# Résumé Phase 7

## Tests Réalisés

Catégorie	Tests	Status
Tests Unitaires	Kafka Handler, Feature Extraction, Anomaly Detection, Calibration	<input type="checkbox"/>
Tests Intégration	Pipeline complet, Latence	<input type="checkbox"/>
Tests Sécurité	SSH Brute Force, Port Scan, Web Fuzzing	<input type="checkbox"/>

## Commandes Essentielles

bash

```
# Lancer tous les tests
pytest tests/ -v

# Tests avec couverture
pytest tests/ --cov=agents --cov-report=html

# Tests d'attaques
./tests/security/run_attacks.sh

# Rapport HTML
open htmlcov/index.html
```

## Phase 8 : Dashboard

### Dashboard Flask

dashboard/app.py

```
from flask import Flask, render_template, jsonify
import psycopg2

app = Flask(__name__)

def get_db():
```

```

        return psycopg2.connect(
            "host=localhost dbname=socdb user=socuser password=soc
pass123"
        )

@app.route('/')
def index():
    return render_template('dashboard.html')

@app.route('/api/stats')
def stats():
    with get_db() as conn:
        with conn.cursor() as cur:
            cur.execute(
                "SELECT threat_level, COUNT(*) FROM events GRO
UP BY threat_level"
            )
            data = dict(cur.fetchall())
    return jsonify(data)

if __name__ == '__main__':
    app.run(port=5000)

```

```
GNU nano 7.2
from flask import Flask, render_template, jsonify
import psycopg2

app = Flask(__name__)

def get_db():
    return psycopg2.connect(
        "host=localhost dbname=socdb user=socuser password=socpass123"
    )

@app.route('/')
def index():
    return render_template('dashboard.html')

@app.route('/api/stats')
def stats():
    with get_db() as conn:
        with conn.cursor() as cur:
            cur.execute(
                "SELECT threat_level, COUNT(*) FROM events GROUP BY threat_level"
            )
            data = dict(cur.fetchall())
    return jsonify(data)

if __name__ == '__main__':
    app.run()
```

## dashboard/templates/dashboard.html

```
<!DOCTYPE html>
<html>
<head>
    <title>SOC Dashboard</title>
</head>
<body>
    <h1>SOC IA Unified Dashboard</h1>
    <div id="stats"></div>
    <script>
        fetch('/api/stats')
            .then(r => r.json())
            .then(data => {
                document.getElementById('stats').innerHTML =
                    JSON.stringify(data);
            });
    </script>
</body>
```

```
</html>
```



The screenshot shows a terminal window with two tabs. The left tab is titled "GNU nano 7.2" and contains the source code for "dashboard.html". The right tab is titled "chaimae@ubuntuCHA: ~/soc-ia-unified" and shows the file "dashboard.html \*". The code in the nano editor is as follows:

```
<!DOCTYPE html>
<html>
<head>
    <title>SOC Dashboard</title>
</head>
<body>
    <h1>SOC IA Unified Dashboard</h1>
    <div id="stats"></div>
    <script>
        fetch('/api/stats')
            .then(r => r.json())
            .then(data => {
                document.getElementById('stats').innerHTML = JSON.stringify(data);
            });
    </script>
</body>
</html>
```

## 🚀 Phase 9 : Démarrage Complet

### Script de Lancement

**scripts/start\_all.sh**

```
#!/bin/bash
source venv/bin/activate

# Démarrer agents en background
python agents/log_tailer/log_tailer.py &
python agents/collector/collector.py &
python agents/features/extractor.py &
python agents/anomaly_detector/detector.py &
python agents/analyzer/analyzer.py &
python agents/trust_agent/trust.py &
python agents/mitre_mapper/mapper.py &
```

```

python agents/xai_explainer/explainer.py &
python agents/responder/responder.py &

# Dashboard
python dashboard/app.py &

echo "✅ Tous les agents démarrés!"

```

```

chmod +x scripts/start_all.sh
./scripts/start_all.sh

```

## Vérification

```

# Vérifier Kafka
docker exec kafka kafka-topics --list --bootstrap-server local
host:9092

# Vérifier logs
tail -f logs/soc-ia.log

# Dashboard
open http://localhost:5000

```



## Résumé des Composants

Agent	Input Topic	Output Topic	Fonction
<b>Log Tailer</b>	-	events_raw	Lit logs système
<b>Collector</b>	events_raw	events_collected	Normalise
<b>Features</b>	events_collected	events_with_features	Extrait features ML
<b>Anomaly</b>	events_with_features	events_with_anomaly	Score anomalie
<b>Analyzer</b>	events_with_anomaly	events_analyzed	Analyse LLM

Agent	Input Topic	Output Topic	Fonction
Trust	events_analyzed	events_calibrated	Calibration confiance
MITRE	events_calibrated	events_with_mitre	Mapping tactiques
XAI	events_with_mitre	events_explained	Explications
Responder	events_explained	events_final	Actions

## Checklist Finale

- Docker Compose running
- Tous les topics Kafka créés
- LM Studio actif (localhost:1234)
- Base PostgreSQL initialisée
- Environnement Python actif
- Tous les agents démarrés
- Dashboard accessible
- Tests passent
- Logs générés

## Dépannage

### Kafka ne démarre pas

```
docker-compose down -v
docker-compose up -d
```

### LLM timeout

- Vérifier LM Studio actif
- Augmenter timeout dans config.yaml

## Pas d'événements

```
# Générer activité  
sudo tail -f /var/log/auth.log  
  
# Vérifier log_tailer actif  
ps aux | grep log_tailer
```



## Ressources

Ressource	Lien
Documentation Kafka	<a href="https://kafka.apache.org/documentation/">https://kafka.apache.org/documentation/</a>
LM Studio Guide	<a href="https://lmstudio.ai/docs">https://lmstudio.ai/docs</a>
MITRE ATT&CK	<a href="https://attack.mitre.org/">https://attack.mitre.org/</a>
Scikit-learn Isolation Forest	<a href="https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html">https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html</a>



Félicitations! Votre SOC IA Unifié est opérationnel!

```
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ docker exec -it soc-ia-unified_kafka_1 kafka  
-console-consumer \  
--bootstrap-server localhost:9092 \  
--topic events_raw \  
--max-messages 3  
{"event_id": "bcc66f49118c6c78", "timestamp": "2026-01-04T20:00:47.129404+00:00", "mess  
age": "192.168.11.145 - - [04/Jan/2026:20:00:47 +0000] \\"GET /doesnotexist1 HTTP/1.1\\"  
404 437 \"-\" \"curl/8.15.0\"", "log_type": "apache", "source": "apache_access"}  
{"event_id": "06a7a96ca7e5d1df", "timestamp": "2026-01-04T20:00:47.191568+00:00", "mess  
age": "192.168.11.145 - - [04/Jan/2026:20:00:47 +0000] \\"GET /doesnotexist2 HTTP/1.1\\"  
404 437 \"-\" \"curl/8.15.0\"", "log_type": "apache", "source": "apache_access"}  
{"event_id": "038fd72a68c2349e", "timestamp": "2026-01-04T20:00:47.419541+00:00", "mess  
age": "192.168.11.145 - - [04/Jan/2026:20:00:47 +0000] \\"GET /doesnotexist3 HTTP/1.1\\"  
404 437 \"-\" \"curl/8.15.0\"", "log_type": "apache", "source": "apache_access"}  
Processed a total of 3 messages  
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$
```