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

📥 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 \;
```

🐳 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
```

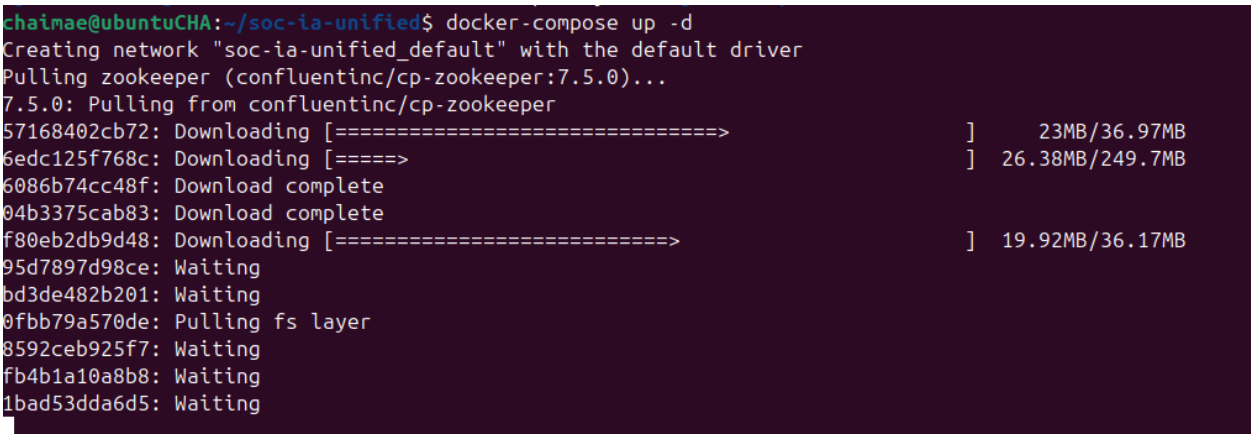
```
  kafka:
    image: confluentinc/cp-kafka:7.5.0
    ports: ["9092:9092", "9093:9093"]
    depends_on: [zookeeper]
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAIN
TEXT_HOST://localhost:9093
      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
```

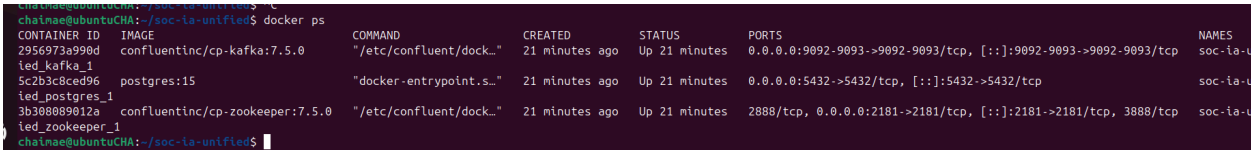
```
chaimae@ubuntuCHA:~/soc-ia-unified$ cat > docker-compose.yml << 'EOFDOCKER'
version: '3.8'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.5.0
    ports: ["2181:2181"]
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181

  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
chaimae@ubuntuCHA:~/soc-ia-unified$ ls
agents  config  dashboard  data  docker-compose.yml  logs  scripts  tests
```

```
      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
```

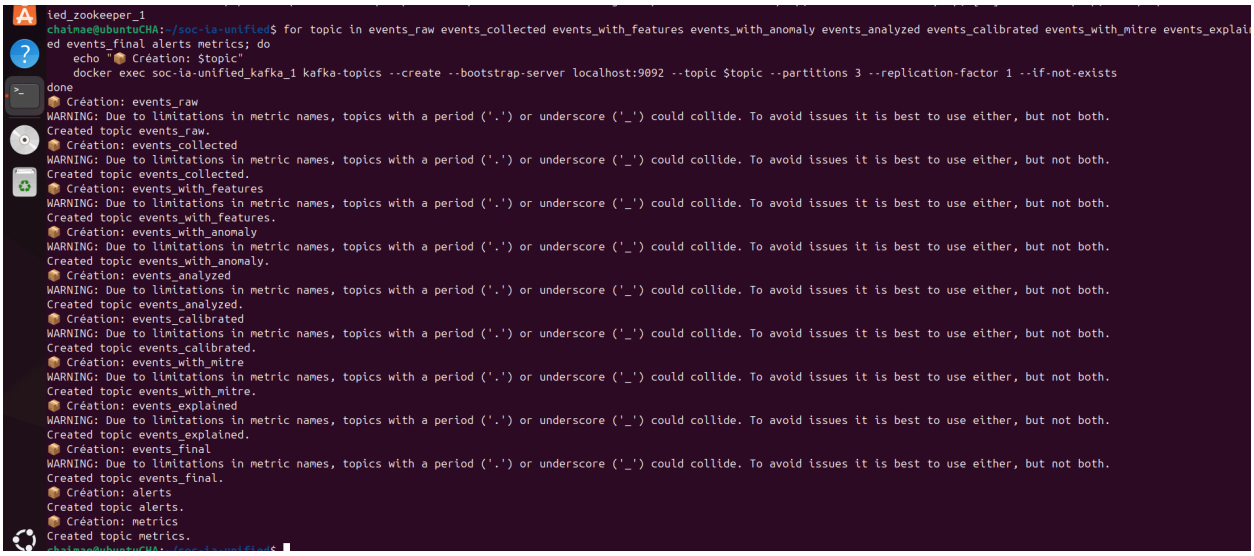




./LM-Studio-0.3.31-7-x64.AppImage --no-sandbox

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
```



Topic	Status
events_raw	✅ Created
events_collected	✅ Created
events_with_features	✅ Created
events_with_anomaly	✅ Created
events_analyzed	✅ Created
events_calibrated	✅ Created
events_with_mitre	✅ Created
events_explained	✅ Created
events_final	✅ Created
alerts	✅ Created
metrics	✅ Created

```
Created topic metrics.
chaimae@ubuntuCHA:~/soc-ia-unified$ docker exec soc-ia-unified_kafka_1 kafka-topics --list --bootstrap-server localhost:909
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
```

```
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)

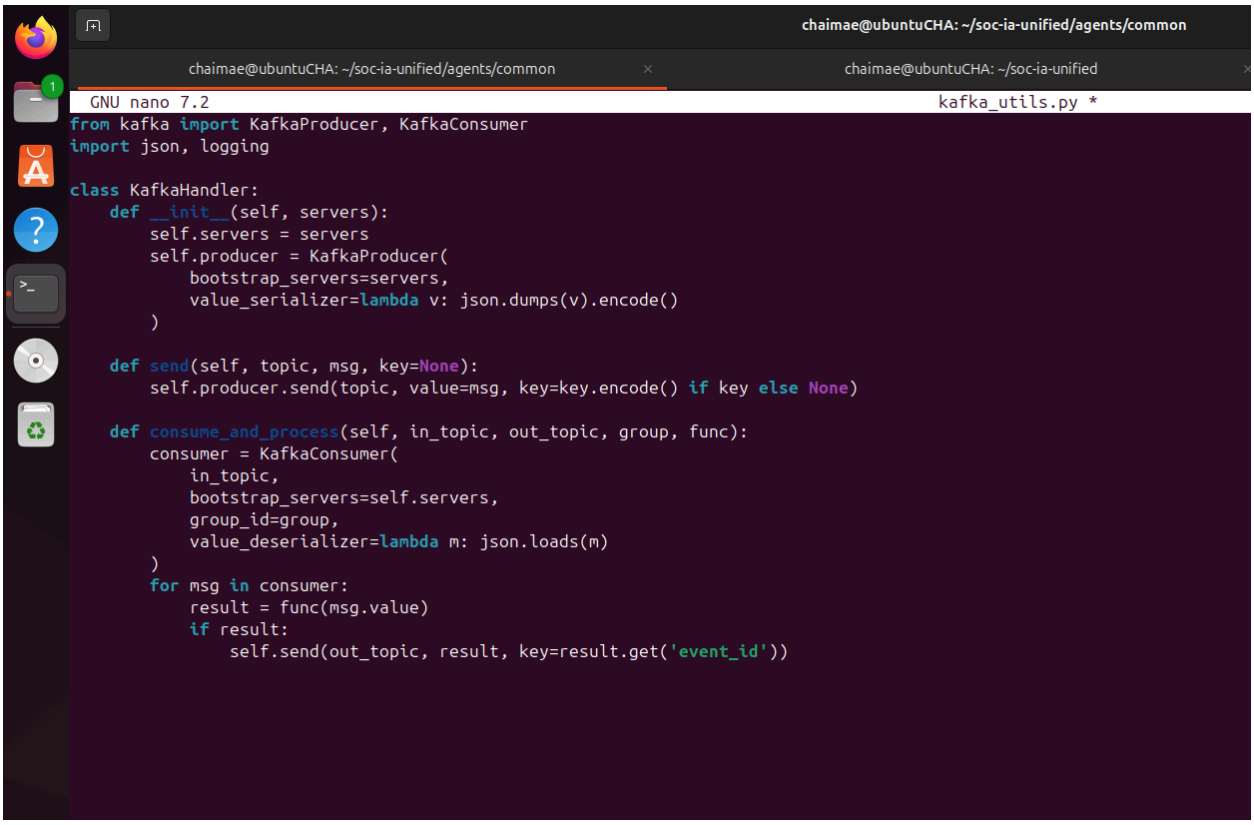
```
kafka:
  bootstrap_servers: "localhost:9093"

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'))
```



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
```

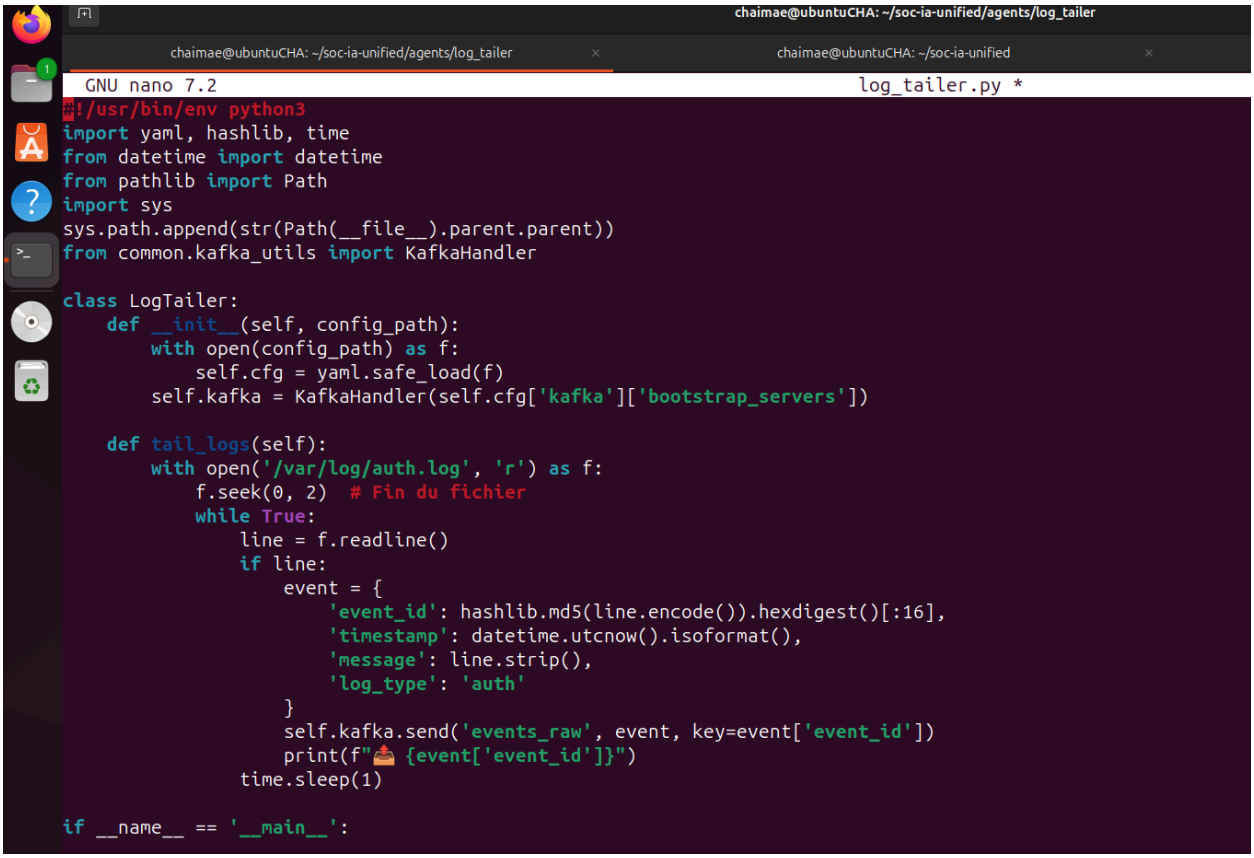
```
out=30).json()
        content = resp['choices'][0]['message']['content']
        try:
            return json.loads(content.replace('```json', '').r
eplace('```', ''))
        except:
            return {"raw": content}
```

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().isoform
at(),
                        'message': line.strip(),
                        'log_type': 'auth'
                    }
                    self.kafka.send('events_raw', event, key=e
vent['event_id'])
                    print(f"📤 {event['event_id']}")
                time.sleep(1)

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



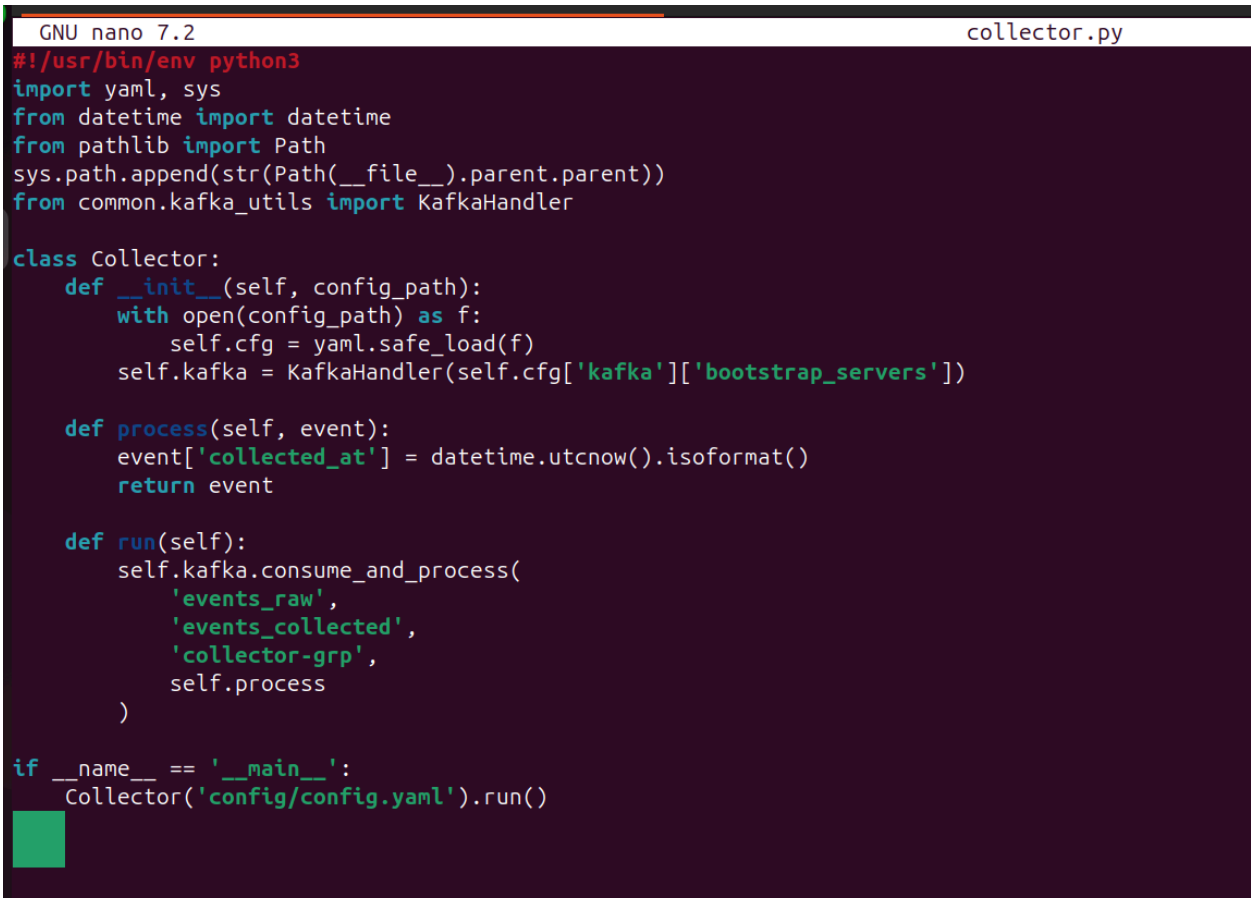
🤖 Phase 5 : Agents - Partie 2

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()
```

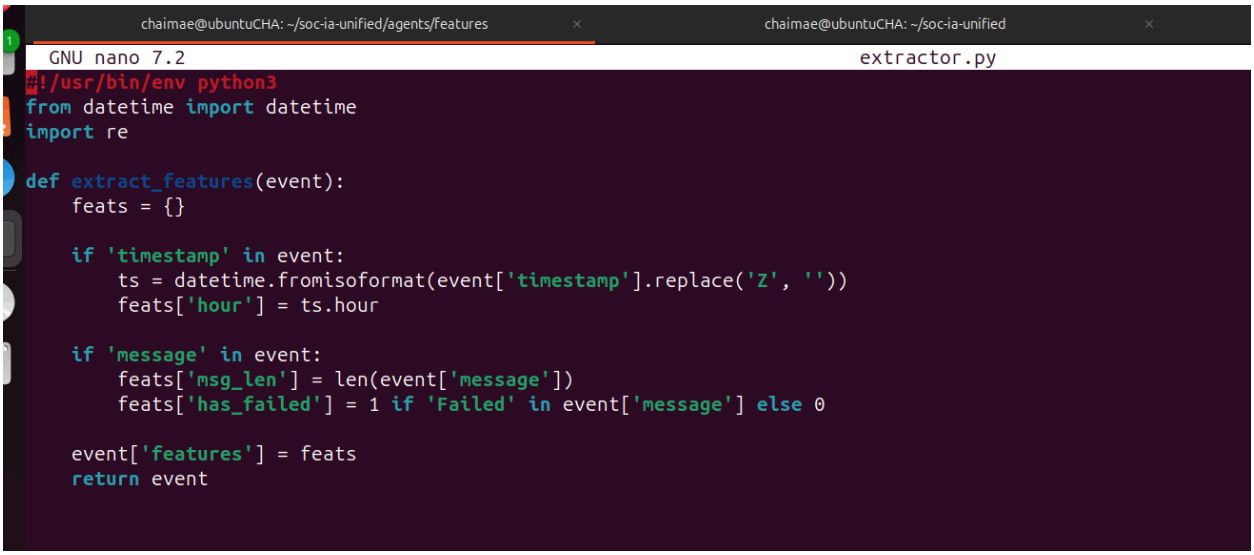


Feature Extractor

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['messag
e'] 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', 'r
b') 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
```

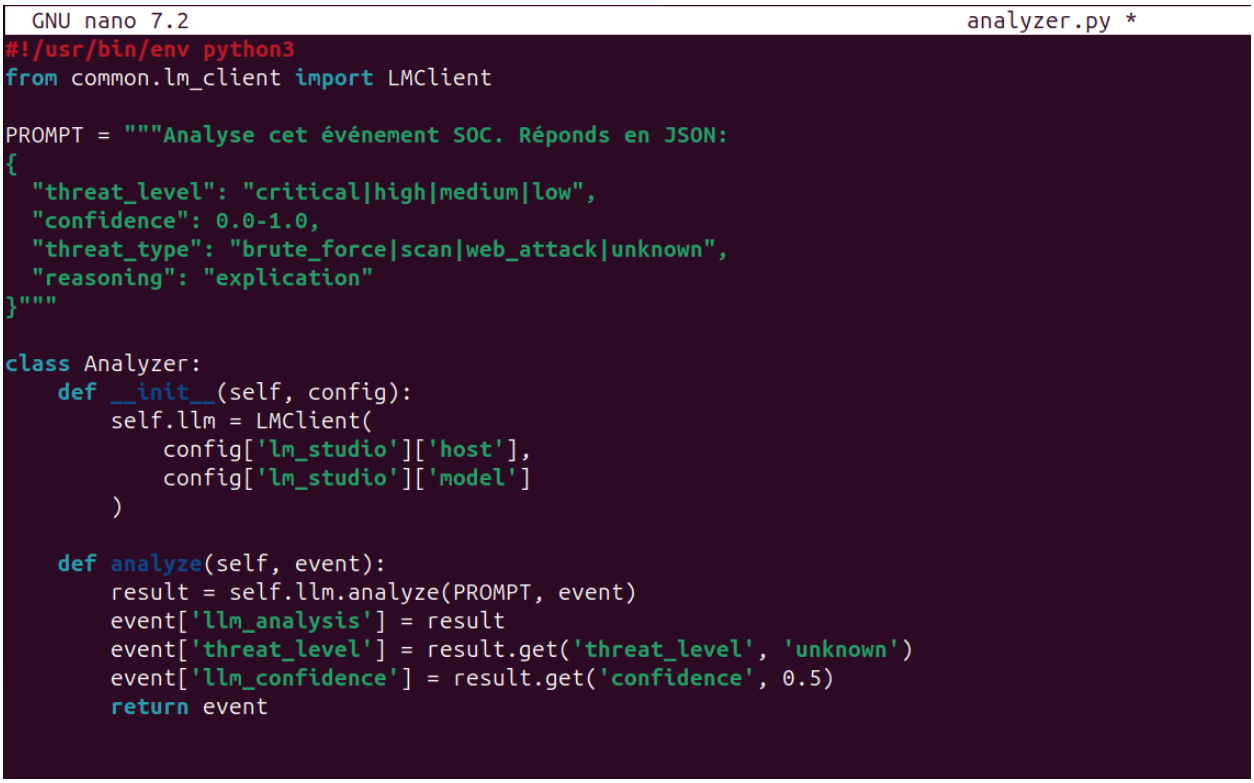
🧠 Phase 6 : Agents Avancés

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', 'un
known')
        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']['te
mperature']

    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

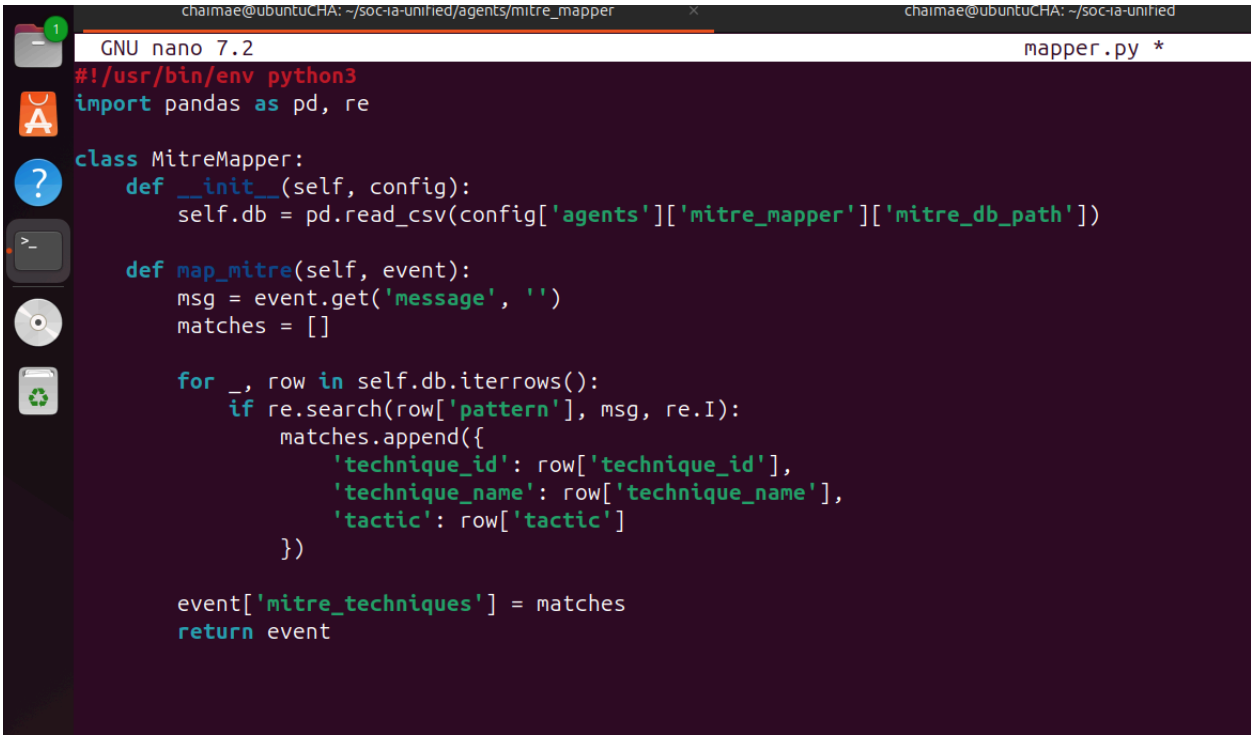
data/mitre/techniques.csv

```
technique_id,technique_name,tactic,pattern
T1110,Brute Force,credential_access,"Failed.*password|authenti
cation failure"
T1046,Network Service Scanning,discovery,"nmap|port scan"
T1190,Exploit Public-Facing Application,initial_access,"404|50
0|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
```

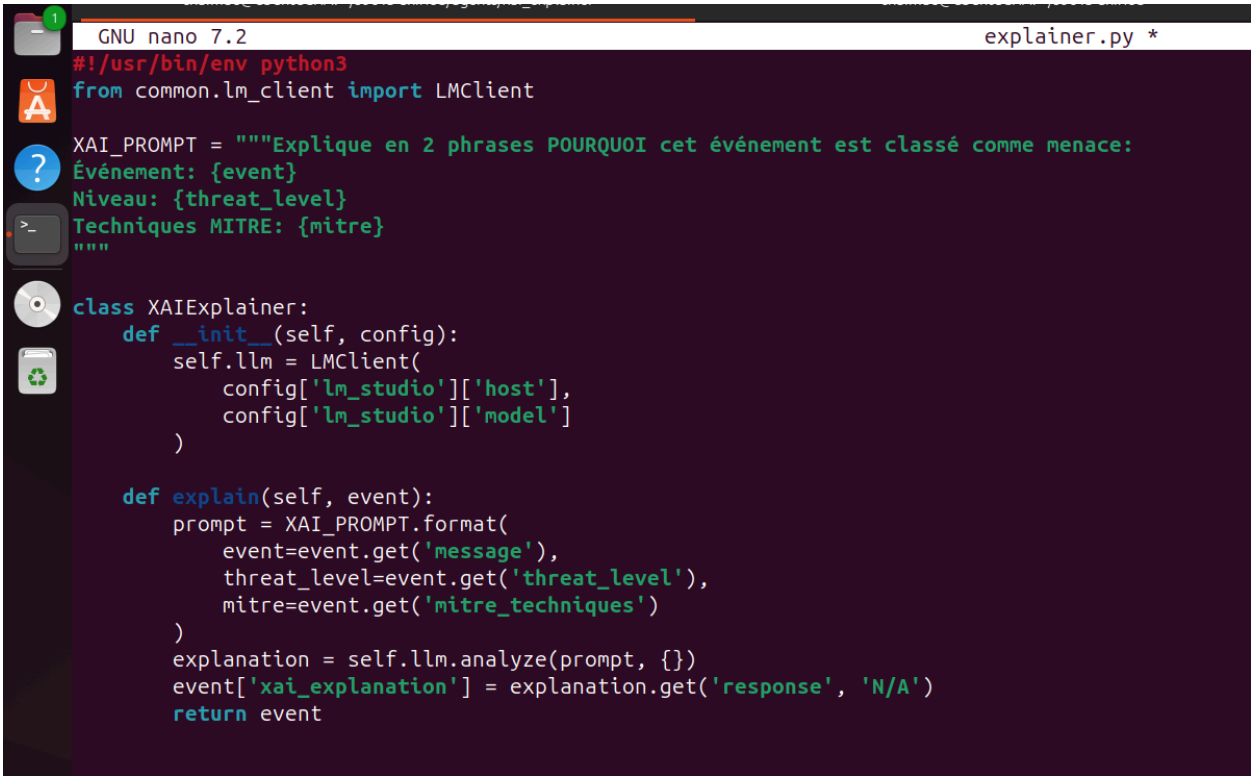


XAI Explainer

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
```



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', i
p, '-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
```

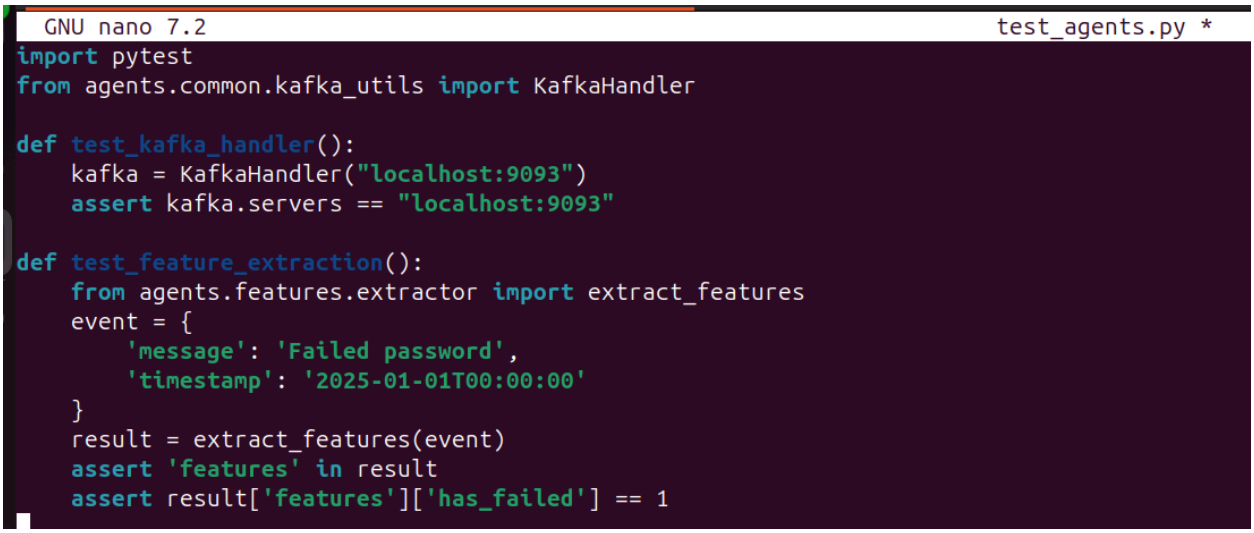
🧪 Phase 7 : Tests (3h)

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
```



Lancer les tests :

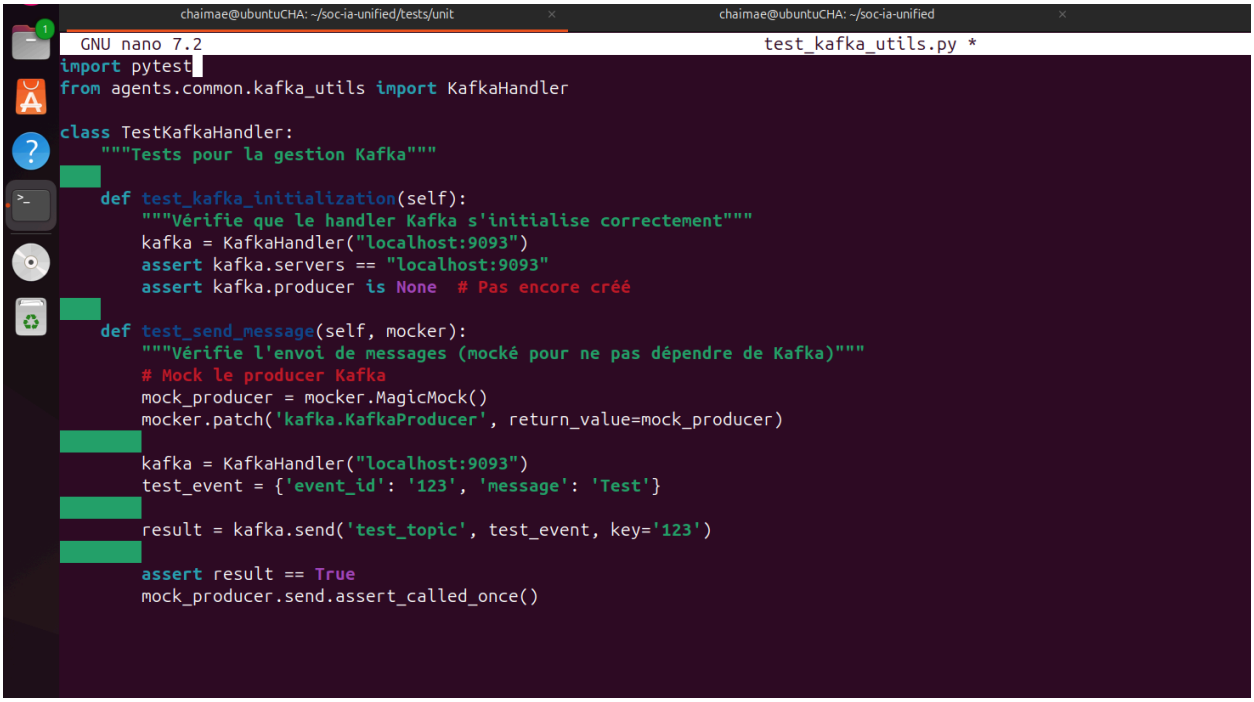
```
pytest tests/
```

```
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$
```

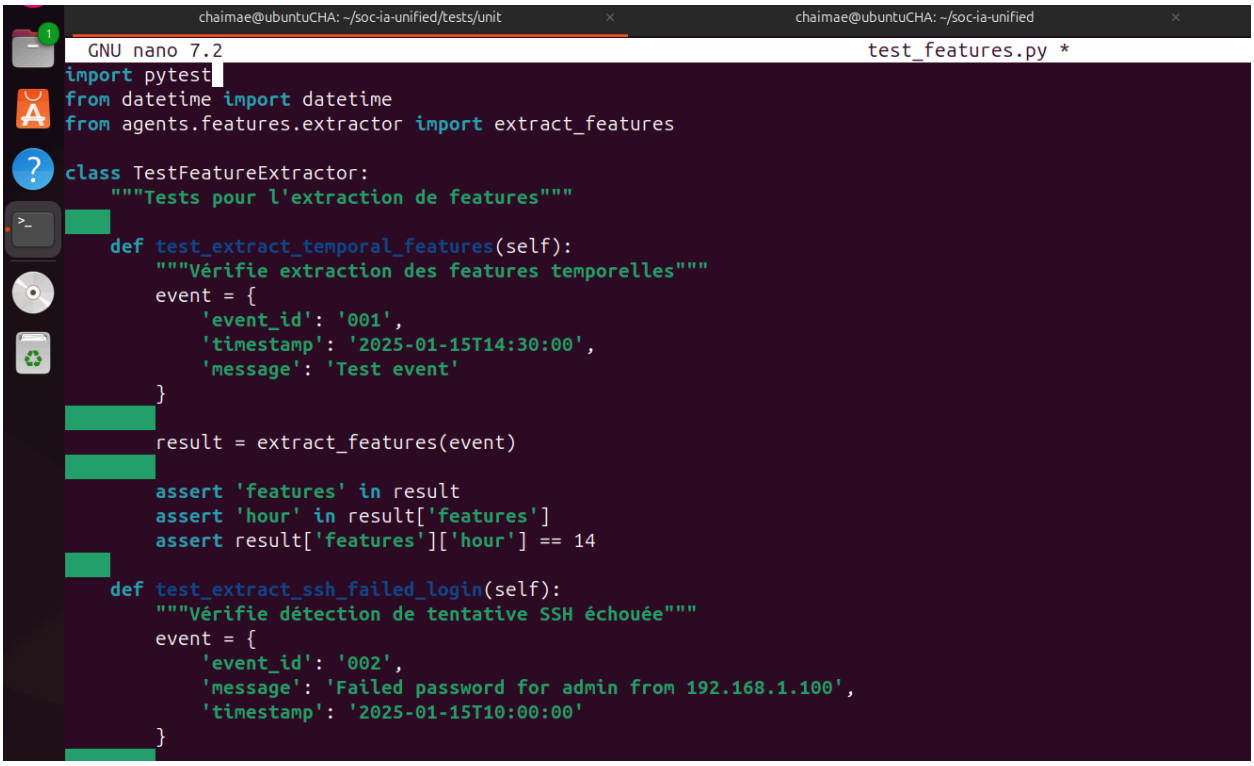
```
tests/
├── unit/                # Tests unitaires (fonctions isolées)
│   ├── test_kafka_utils.py
│   ├── test_features.py
│   ├── test_anomaly.py
│   └── test_calibration.py
├── integration/         # Tests bout-en-bout
│   ├── test_pipeline.py
│   └── test_full_flow.py
└── security/           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

🔬 Partie 1 : Tests Unitaires



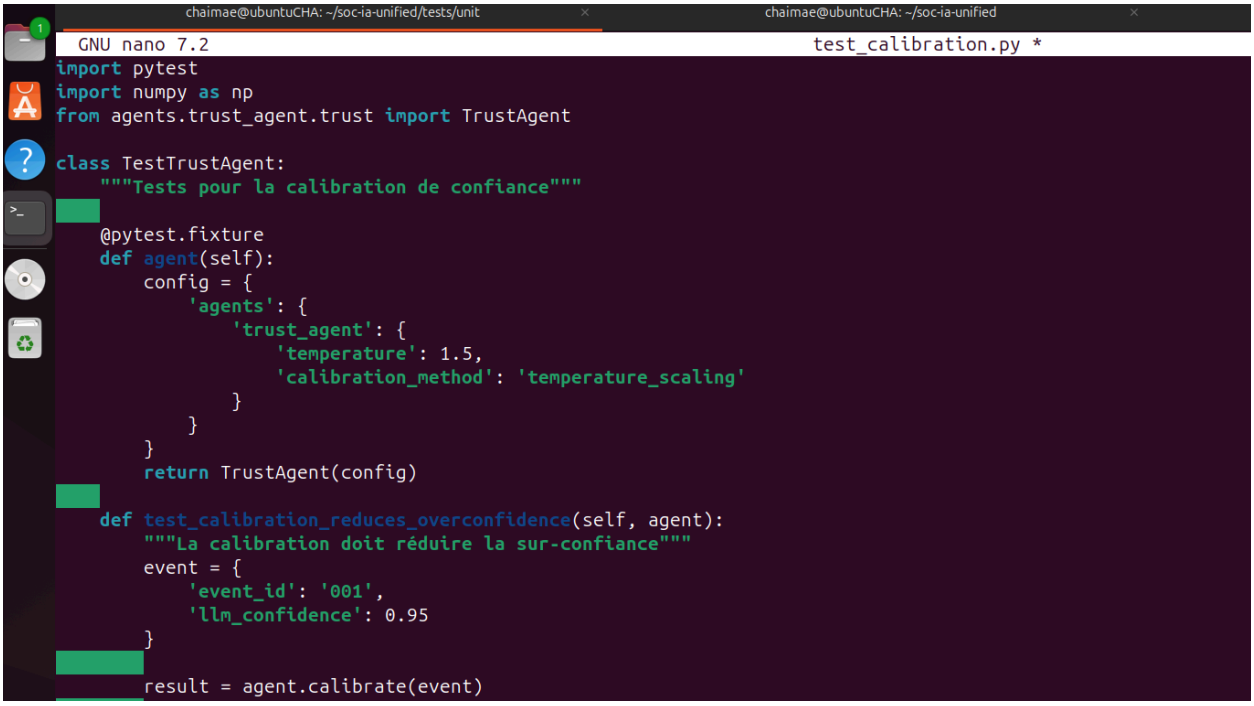
Lancement du test :

pytest tests/unit/test_kafka_utils.py -v




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


chaimae@ubuntuCHA: ~/soc-ia-unified/tests/unit
chaimae@ubuntuCHA: ~/soc-ia-unified
GNU nano 7.2
test_anomaly.py *
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
}
}


chaimae@ubuntuCHA: ~/soc-ia-unified/tests/unit
chaimae@ubuntuCHA: ~/soc-ia-unified
GNU nano 7.2
test_calibration.py *
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 :

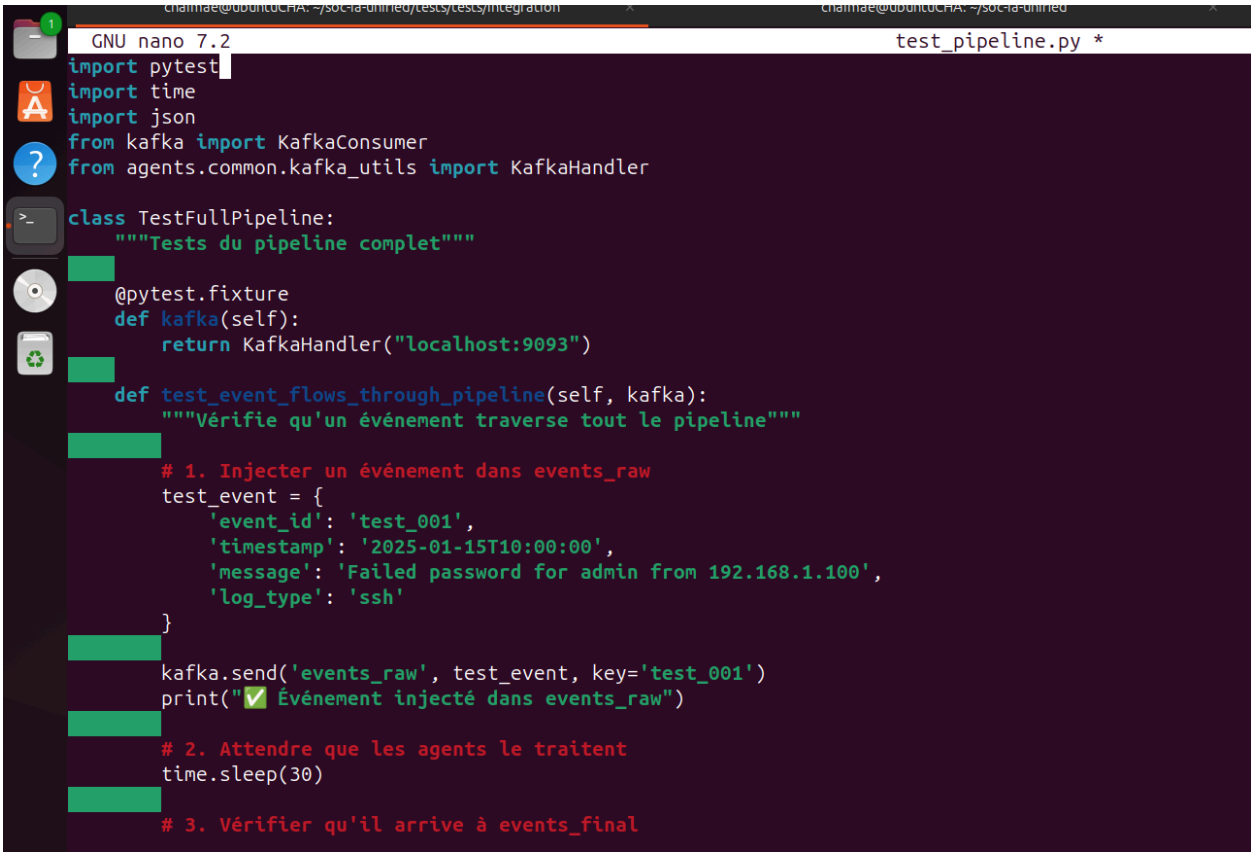
```
EOF
(venv) chaimae@ubuntuCHA:~/soc-ia-unified$ python -m pytest tests/ -v
============================= test session starts ==============================
platform linux -- Python 3.12.3, pytest-9.0.2, pluggy-1.6.0 -- /home/chaimae/soc-ia-unified/venv/bin/python
cachedir: .pytest_cache
rootdir: /home/chaimae/soc-ia-unified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

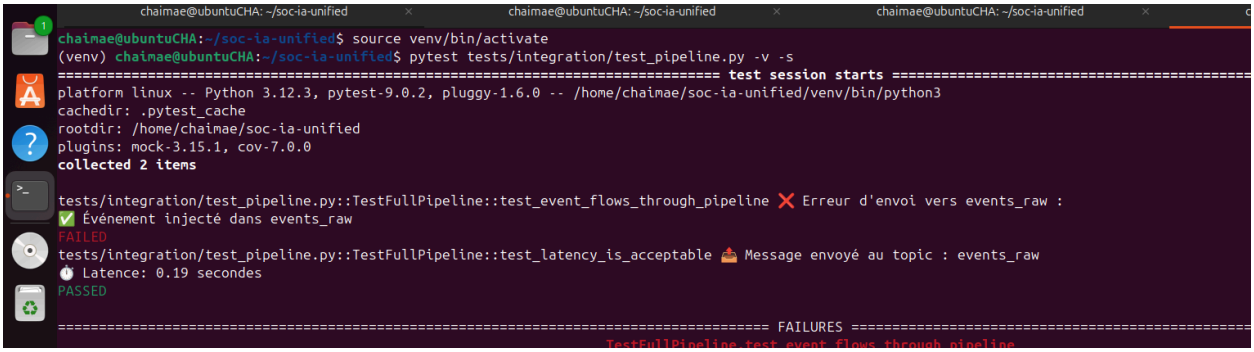
Vérifie que **plusieurs composants fonctionnent ensemble** correctement.



Lancement des tests d'intégration

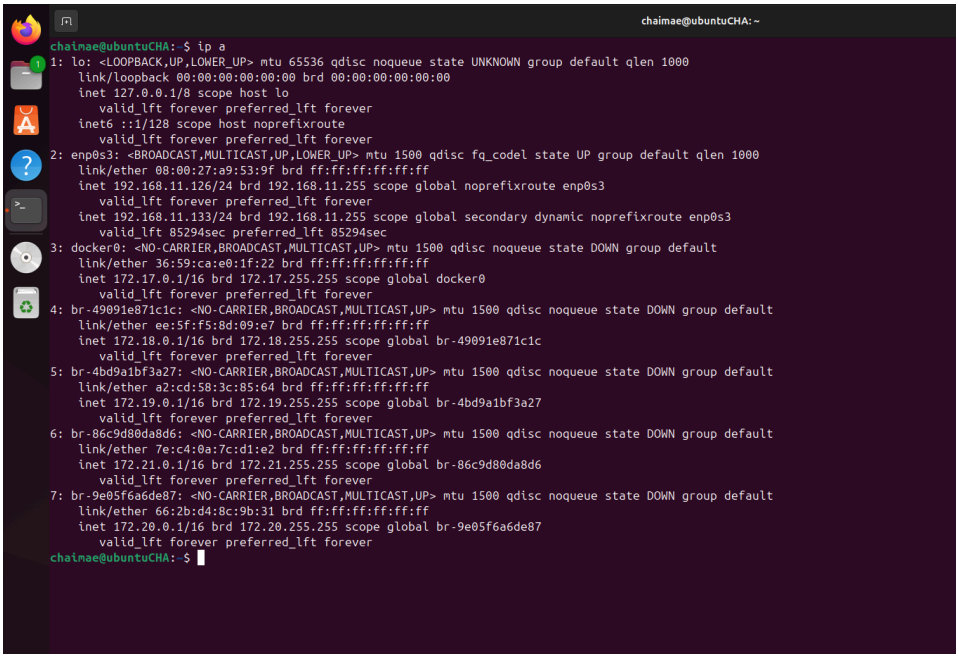
`./scripts/start_all.sh`

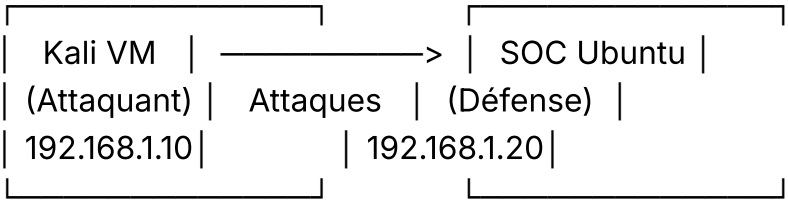
`pytest tests/integration/test_pipeline.py -v -s`



🎯 Partie 3 : Tests d'Attaques Réelles

Configuration du Lab





🕵️ 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

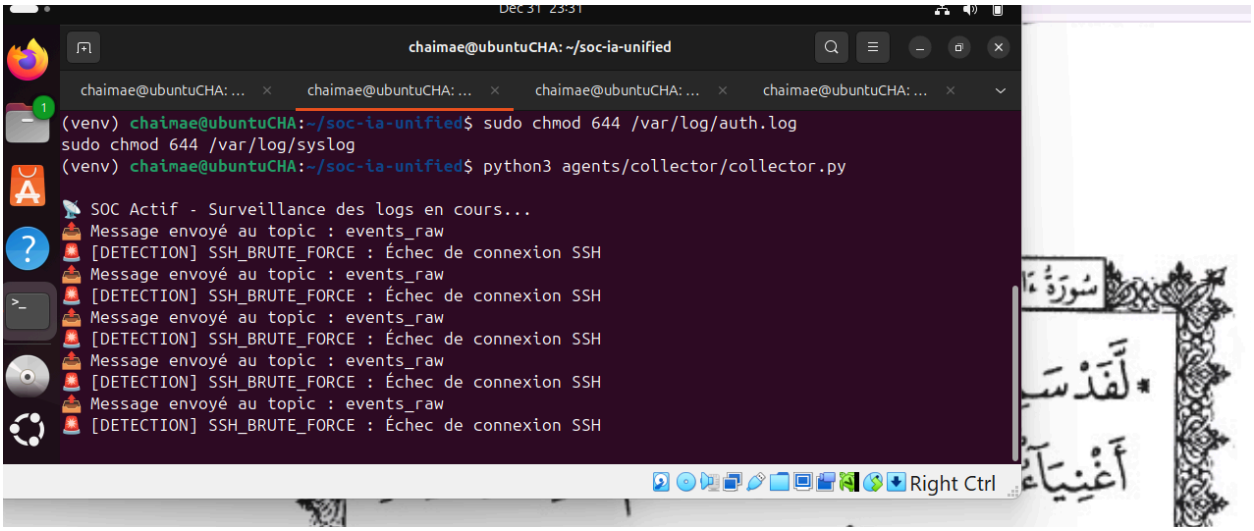
sudo apt install hydra -y

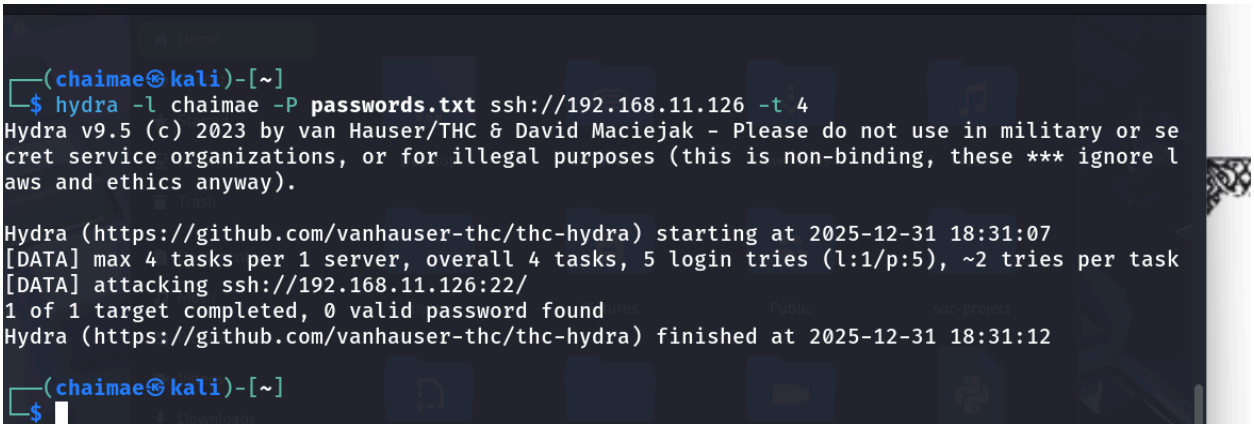
Créer liste de mots de passe

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
```





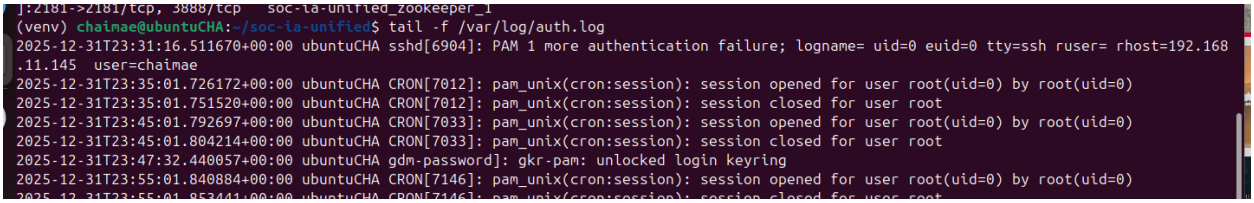
Explication des paramètres

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



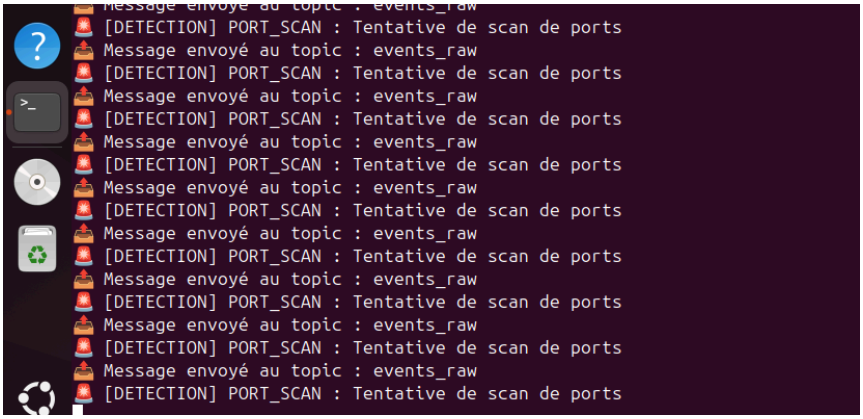
Vérifier alertes SOC

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:A9:53:9F (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

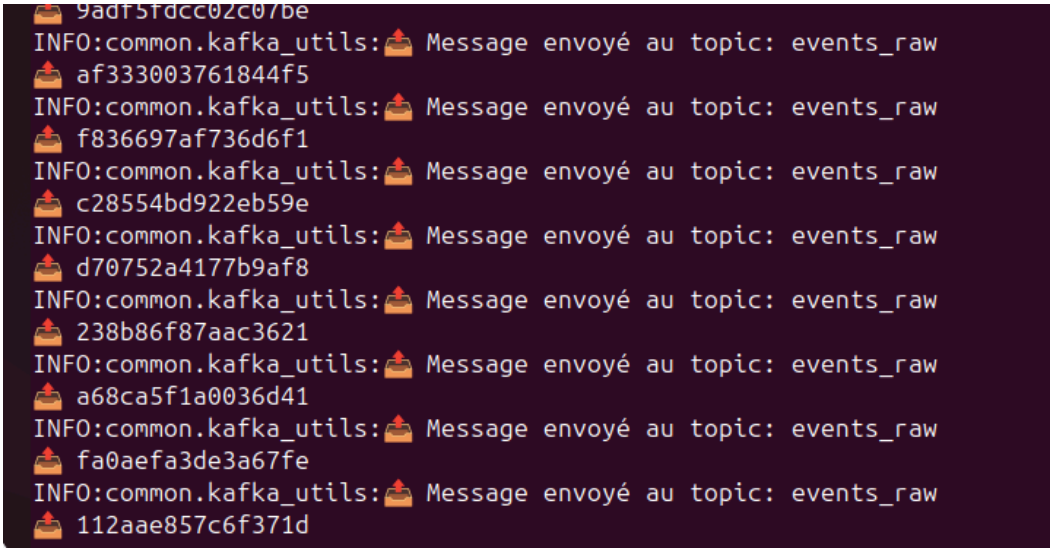
Paramètre	Description
`-sS`	SYN scan (ne complète pas connexion TCP)
`-p-`	Scanner TOUS les ports (1-65535)
`-sV`	Détecter versions des services
`-O`	Détecter OS

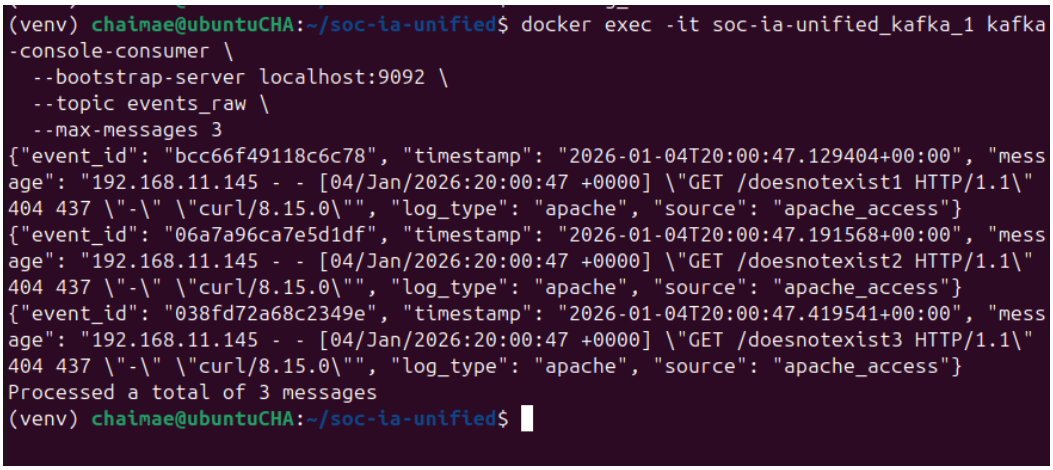
Ce que le SOC doit détecter

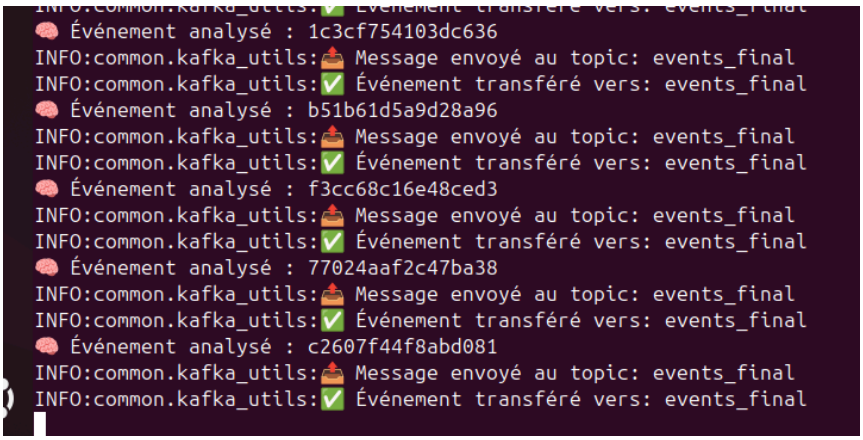
- Pic de connexions depuis même IP
- Connexions à multiples ports

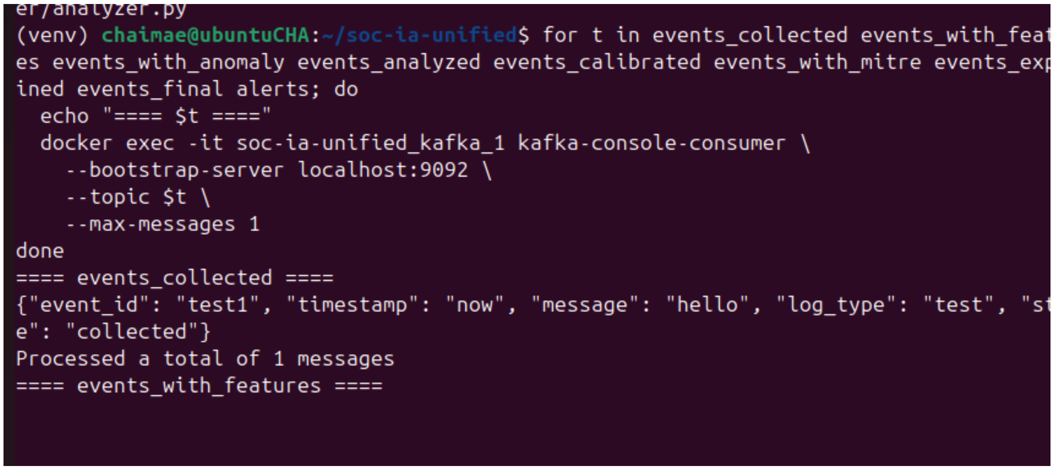
- Pattern de scan caractéristique

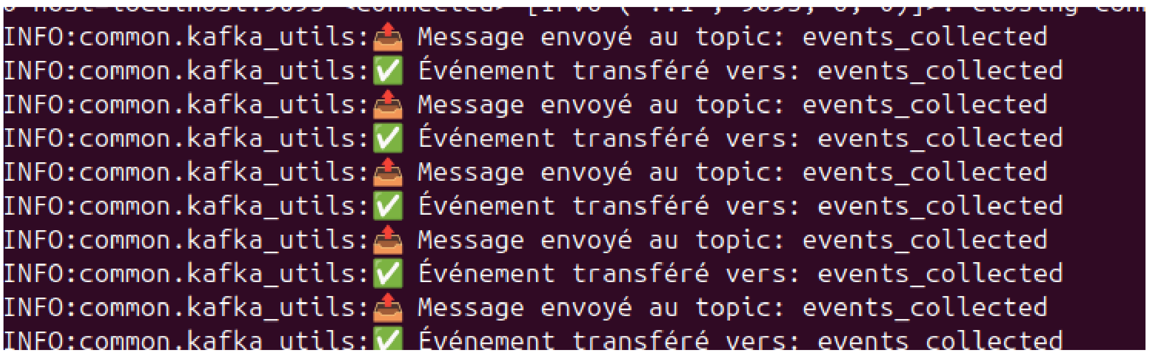
3. Web Fuzzing avec Gobuster

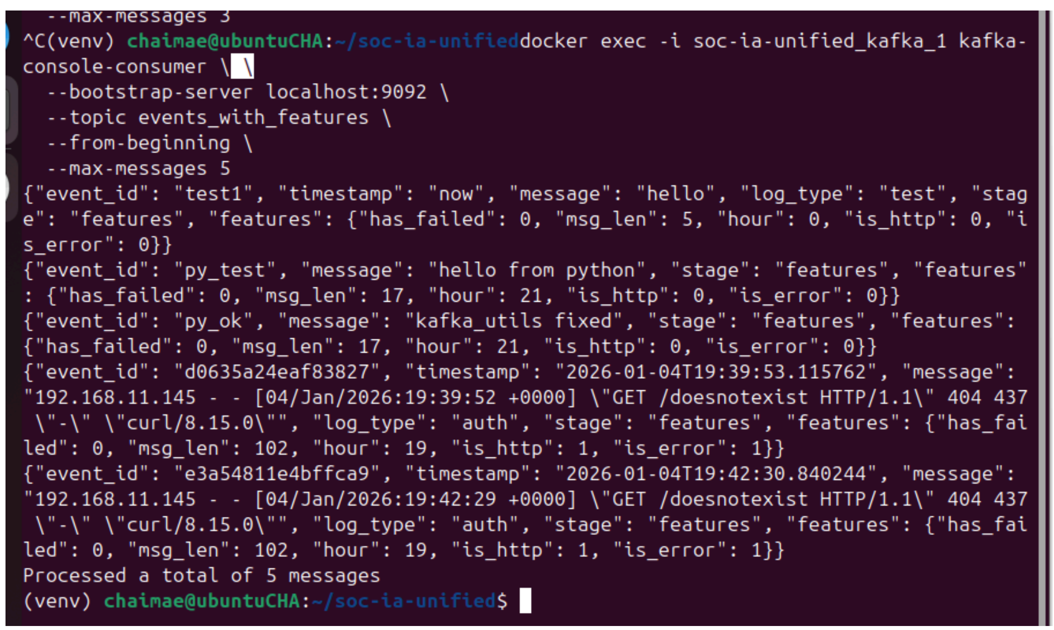








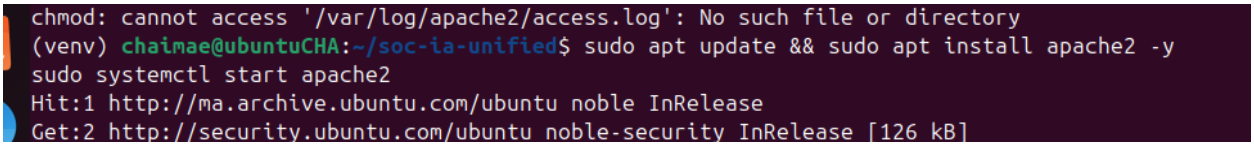




Sur Kali - Lancer l'attaque

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

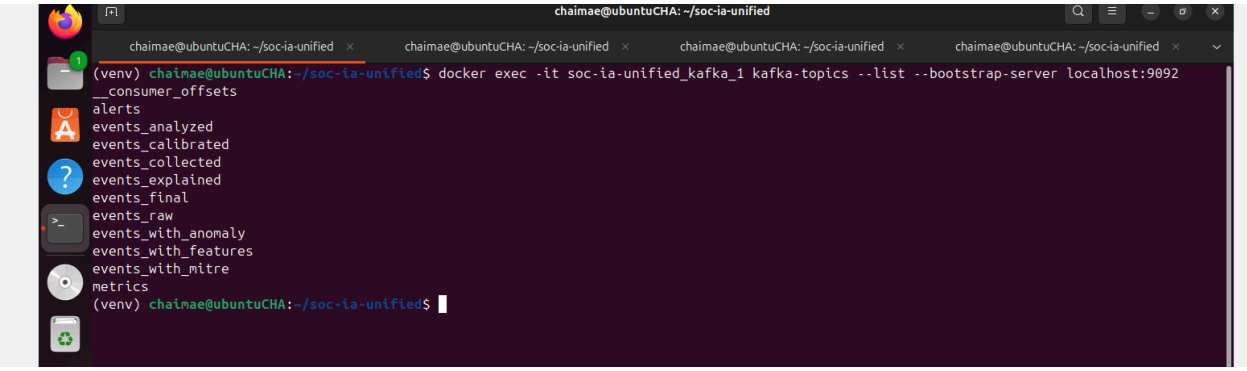
```
# Fuzzing de répertoires
gobuster dir \
  -u http://192.168.11.26 \
  -w /usr/share/wordlists/dirb/common.txt \
  -t 50

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



```
┌──(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"
```



Explication des paramètres

Paramètre	Description
`dir`	Mode directory bruteforce
`-u`	URL cible
`-w`	Wordlist à utiliser
`-t 50`	50 threads (requêtes parallèles)
`-x`	Extensions à tester

Script d'Attaque Simulée (Kali)

```
# SSH Brute Force
hydra -l admin -P /usr/share/wordlists/rockyou.txt ssh://TARGE
T_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 "1️⃣  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 "2️⃣  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 "3️⃣  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
Détection Rate	>90%	Alertes / Attaques réelles
False Positives	<10%	Alertes erronées / Total alertes
Latence	<60s	Temps raw → final
Throughput	>100 events/s	Events traités par seconde
Accuracy LLM	>85%	Prédictions correctes
MITRE Coverage	>80%	Techniques détectées

✅ Résumé Phase 7

Tests Réalisés

Catégorie	Tests	Status
Tests Unitaires	Kafka Handler, Feature Extraction, Anomaly Detection, Calibration	⬜
Tests Intégration	Pipeline complet, Latence	⬜
Tests Sécurité	SSH Brute Force, Port Scan, Web Fuzzing	⬜

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
```

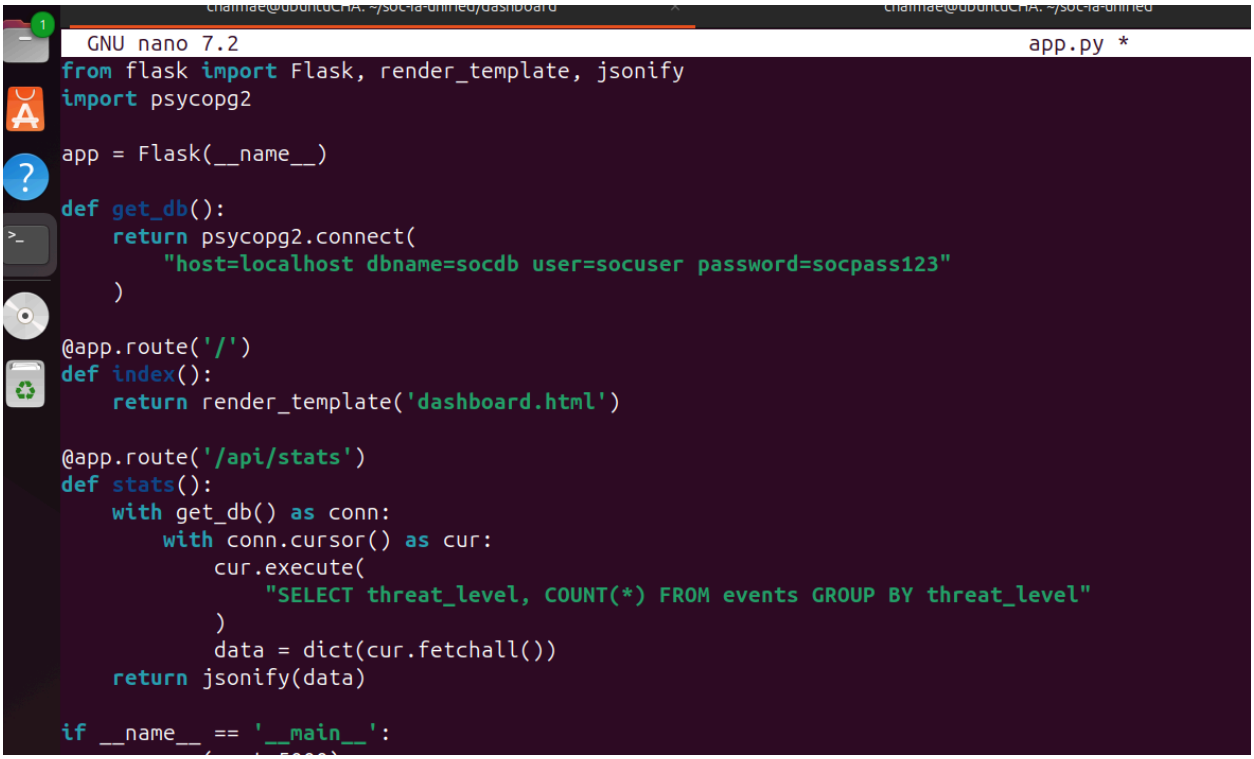
🎨 Phase 8 : Dashboard

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)
```



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 = J
SON.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
Log Tailer	-	events_raw	Lit logs système
Collector	events_raw	events_collected	Normalise
Features	events_collected	events_with_features	Extrait features ML
Anomaly	events_with_features	events_with_anomaly	Score anomalie
Analyzer	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
- [x] ~~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	https://kafka.apache.org/documentation/
LM Studio Guide	https://lmstudio.ai/docs
MITRE ATT&CK	https://attack.mitre.org/
Scikit-learn Isolation Forest	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

> 🎉 Félicitations! Votre SOC IA Unifié est opérationnel!