FAQ (Frequently Asked Questions)

Detection Engine not working

Issue/Introduction: What to do when the detection engine is not working as expected?

Environment:

ShieldOps Server: Version 1

Resolution:

> **Prerequisites:** you must have shell access to the host, sufficient privileges (run commands with `sudo` if required), Docker and Docker Compose available, and know the absolute path to the `ShieldOps` project directory.

Check the systemd journal for the worker service

Run:

```
journalctl -u django_worker -e --no-pager
```

1. *What to look for:* recent error messages, stack traces, or repeated restart attempts. Note timestamps and any clear failure reasons (permission denied, missing files, configuration errors, etc.).

Change to the project directory

Move to the folder that contains the `docker-compose.yml` (or `compose` files) for `ShieldOps`:

```
cd ShieldOps
```

2. Make sure the compose file defines a `worker` service (the commands below assume the service is called `worker`).

Bring the worker container down

Stop and remove the worker container to ensure a clean restart:

```
docker compose down worker
```

3. *Notes:* If your environment uses the old single- word binary, the equivalent is `docker-compose down worker`.

Restart the systemd service that manages the worker

Restart the systemd unit that is responsible for the detection engine:

```
sudo systemctl restart django_worker
```

Wait a few seconds, then check its status:

```
sudo systemctl status <backend_service>_worker --no-pager
```

4. *What to expect:* the unit should show `Active: active (running)` or at least `exited` with no recent errors. If the unit fails immediately, review the unit's logs again with `journalctl`.

Bring the worker containers back up (rebuild if required)

Start the worker container(s) with a fresh build:

```
docker compose up -d --build worker
```

Confirm containers are running:

```
docker compose ps
```

Verify logs after restart

Re-check the journal to ensure the worker is behaving correctly:

```
journalctl -u django_worker -e --no-pager
```

And to view Docker container logs:

```
docker logs shieldops-worker-(1-8)
```

Confirm detection engine functionality

- Validate that alerts or detection outputs resume as expected.
- If available, run a small test that triggers the detection pipeline and confirm the expected output.

The Docker services are unhealthy

Issue/Introduction: How to resolve if the docker services are unhealthy?

Environment:

ShieldOps Server: Version 1

Resolution (Step-by-Step):

1. **Check the status of your Docker services.**
    - First, navigate to the project directory where your `docker-compose.yml` file is located.

      ```
      cd ShieldOps
      ```

    - Run the command `docker ps`. This will display a list of all running containers, along with their current status, health, and ports.
2. **Inspect the service logs.**
    - If a service appears unhealthy, use the `docker logs <service_name>` command to get more information.
    - The logs will often contain error messages or other clues that can help you identify the root cause of the problem.
3. **Attempt to restart the service.**
    - If the logs don't immediately reveal the issue, a simple restart can often fix transient problems.
    - Run: `systemctl restart <service_name>`
    - After the restart, check the logs again with `docker logs <service_name>` to see if the service is now behaving correctly.
4. **Stop and restart the service with Docker Compose.**
    - If the service is still unhealthy, you can try stopping and restarting it using Docker Compose to ensure a clean state.
    - First, stop and remove the unhealthy container: `docker compose down <service_name>`
    - Then, start the container again: `docker compose up -d <service_name>`
5. **Examine the `docker-compose.yml` file.**
    - If restarting doesn't work, the problem might be in the command the service is trying to run.

- Open your `docker-compose.yml` file and carefully inspect the `command:` section for the service. Make sure it is correct and that the application inside the container is running the correct process to be considered healthy.

6. **Ping the service inside the container.**
   - If the service still isn't working, the problem could be a network or connectivity issue.
   - To test if the service is even active, you can exec into the container and try to ping it.

     ```
     docker exec -it <service_name> /bin/bash
     ```

     ```
     bash-4.2$ ping <service_name>
     ```

   - This helps determine if the container is functional but not communicating, or if the service itself is just not active.

Parser is not working

Issue/Introduction: How to resolve if the parser is not working?

Environment

ShieldOps Server: Version 1

Resolution:

- **Check the parser logs.**
  - Start by inspecting the logs of the parser service itself.
  - Run: `docker logs parser`
  - **What to look for:** Look for error messages or indications that the parsing pipeline has been terminated. If you see a "pipeline terminated" message, the problem is likely with the parser's configuration.
- **Examine the collector logs.**
  - If the parser logs show no errors, but no data is being received, the problem could be with the collector service, which is responsible for gathering the logs.
  - Run: `docker logs collector`
  - **What to look for:** Analyze the collector's logs to see if it is successfully collecting and sending data. This will help you determine if the issue is with the data ingestion point.
- **Inspect Kafka logs and restart the service.**
  - If both the parser and collector logs appear normal, the issue might be with Kafka, which handles the message queue.
  - A simple restart can often resolve a temporary Kafka issue.
  - Try: `systemctl restart kafka`
  - If the problem persists after the restart, you will need to inspect Kafka's internal logs for a more detailed diagnosis. These logs can reveal issues like configuration problems, disk space issues, or other internal errors.
  - Try: `docker exec -it collector /bin/bash`

    followed by `kafka-topics.sh --list --bootstrap-server localhost:9092`

    This should list all the current topics inside kafka, you can further check individually in topics whether they are actively consuming logs or not by the following command:
    `kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic sample_parser --from-beginning --max-messages 5`

500 Server Error on UI

Introduction/Issue: How to resolve if a 500 server error occurs and no resources are shown on the UI?

Environment

ShieldOps Server: Version 1

Resolution:

1. **Check the status of the backend server.**
    - First, inspect the server's logs to see if it's running and to identify any specific error messages.
    - Run: `journalctl -u django -e --no-pager`
    - **What to look for:** Look for specific endpoints that are failing. Note down any error messages or stack traces as they are crucial for diagnosing the issue. If an endpoint is failing, it needs to be resolved by the development team.
2. **Wait for services to load.**
    - If the logs show no errors but indicate that the server is unreachable, it's possible that the services are still loading. Sometimes, a brief waiting period can resolve this. Wait a few moments and refresh the UI.
3. **Restart the backend service.**
    - If the issue persists, the next step is to restart the backend service. This can often resolve temporary hang-ups or resource conflicts.
    - Run: `sudo systemctl restart django`
    - After the restart, inspect the logs again using `journalctl -u django -e --no-pager` to confirm that the service is running without errors.