## Introduction

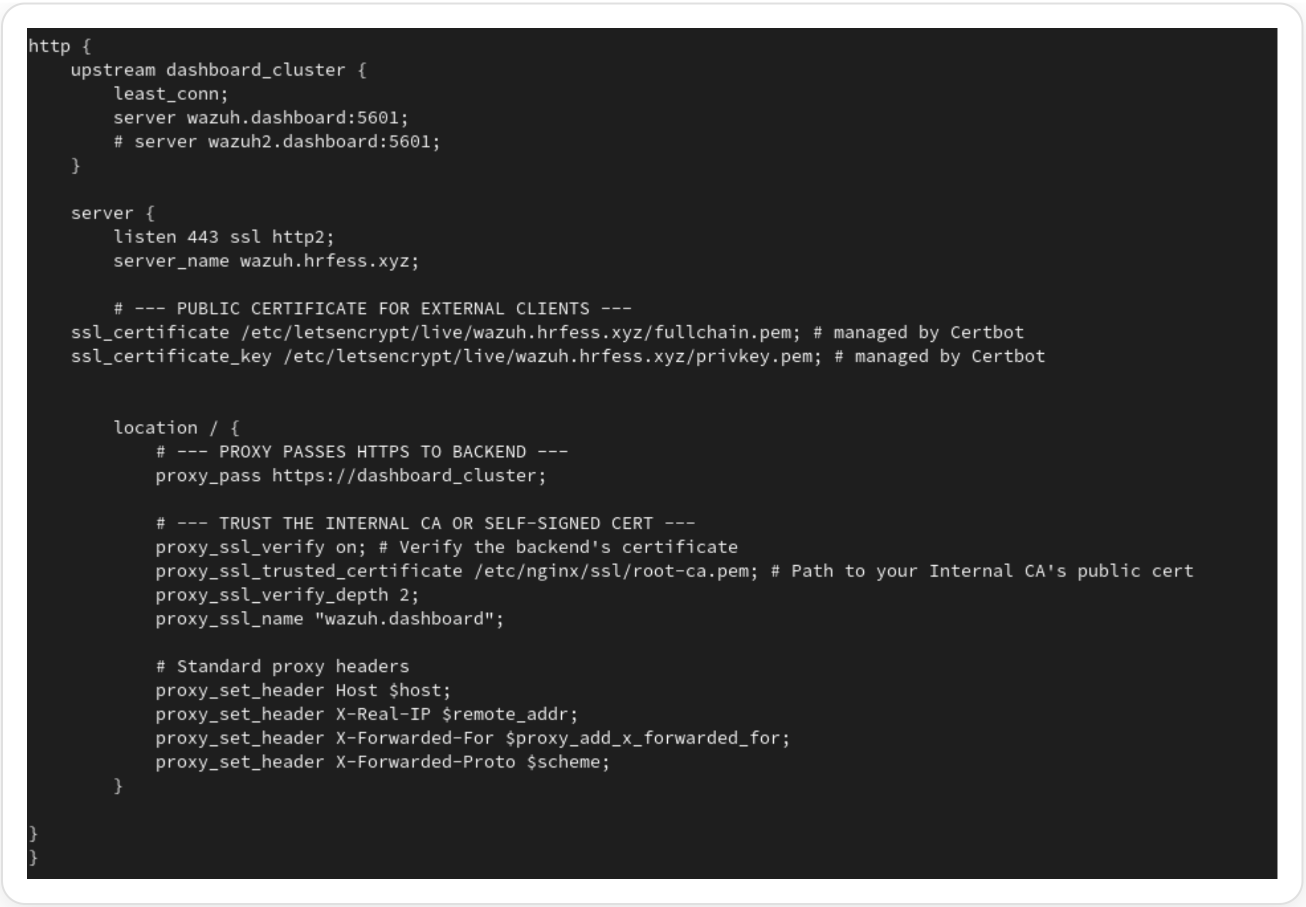
I started by deploying a docker compose version of the SIEM.
I was interested first in making my host reachable via internet though my internet subscription doesn't allow that. This is important to let let's encrypt verify my ownership of the domain name that would be bound to the SSL certificate. Then, I configured a multi-node version of docker-based wazuh deployment. I moved to define the specification of the application in a docker-compose-file. Finally, I ended up by switching to swarm mode and setting a CI/CD pipeline for continuous and collaborative enhancements.

## HTTPS Termination

In my stack, a component called reverse-proxy is responsible of:

1. Load-balancing agents' enrollment and communications.
2. Load-balancing dashboards' cluster.
3. Terminating the HTTPS upstream of the dashboard cluster with letsencrypt signed ssl certificate, so, the analyst won't be prompted to trust the internal self generated ssl cert.
   This implementation was with goals in mind, including:

- Once started, the container of the reverse proxy should auto renew the ssl certificate with no downtime and no intervention.
- The first run of the container would handle the generation of let's encrypt ssl cert: authentication, and placement of the cert.
  I specified that the container puts the generated ssl cert among other files on `/configs/nginx/letsencrypt` at first deployment. This happens when the environment variable `FIRST_RUN` at `/configs/nginx/.env` is set to true.
  The container runs an entrypoint script at `/security/nginx/entrypoint.sh` that verifies FIRST_RUN. This script, however, sets two crontab jobs for automatic renewal. And, it is based on a nginx image where certbot utility installs.
  The configuration part of nginx that terminates the internal https endpoints is at figure 1.



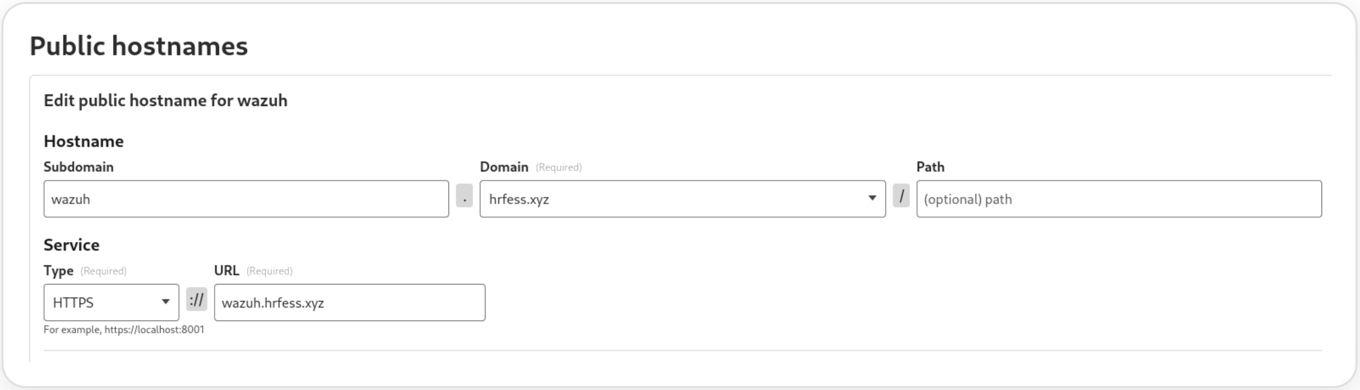
```
http {
    upstream dashboard_cluster {
        least_conn;
        server wazuh.dashboard:5601;
        # server wazuh2.dashboard:5601;
    }

    server {
        listen 443 ssl http2;
        server_name wazuh.hrfess.xyz;

        # --- PUBLIC CERTIFICATE FOR EXTERNAL CLIENTS ---
    ssl_certificate /etc/letsencrypt/live/wazuh.hrfess.xyz/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/wazuh.hrfess.xyz/privkey.pem; # managed by Certbot


        location / {
            # --- PROXY PASSES HTTPS TO BACKEND ---
            proxy_pass https://dashboard_cluster;

            # --- TRUST THE INTERNAL CA OR SELF-SIGNED CERT ---
            proxy_ssl_verify on; # Verify the backend's certificate
            proxy_ssl_trusted_certificate /etc/nginx/ssl/root-ca.pem; # Path to your Internal CA's public cert
            proxy_ssl_verify_depth 2;
            proxy_ssl_name "wazuh.dashboard";

            # Standard proxy headers
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

}
}
```

Figure 1: nginx.conf file

The domain name in the configuration file should resolve to the host running this container.

## Internet facing host

To bypass the restriction of not having a static IP address, I used cloudflare zero trust tunnels.
At the first launch of the container I used an HTTP tunnel. This is because certbot authenticates and grabs the certs form letsencrypt using http.
At production and test environment swarm mode/compose mode, https tunnel has been used.
Figure 2 shows the cloudflare settings at this stage.



Public hostnames

Edit public hostname for wazuh

Hostname

| | | |
|---|---|---|
| Subdomain: wazuh | Domain (Required): hrfess.xyz | Path: (optional) path |

Service

| | |
|---|---|
| Type (Required): HTTPS | URL (Required): wazuh.hrfess.xyz |

For example, https://localhost:8001

Cloudflare https tunnel

When https tunnel is in use, the local dns resolver `/etc/hosts` must have the entry:

```
127.0.0.1 wazuh.hrfess.xyz
```

So, nginx can validate the subject name.

## Docker compose mode

To simulate an initial prototype of swarm cluster, I started by a docker-compose.yml specification of the app. This version edited the docker-compose.yml that of wazuh multi-node: master/worker.
It used 4 overlay networks with bridged adapters: frontend, backend, indexer and management. Services were assigned to these networks as necessary. And, used local volumes.

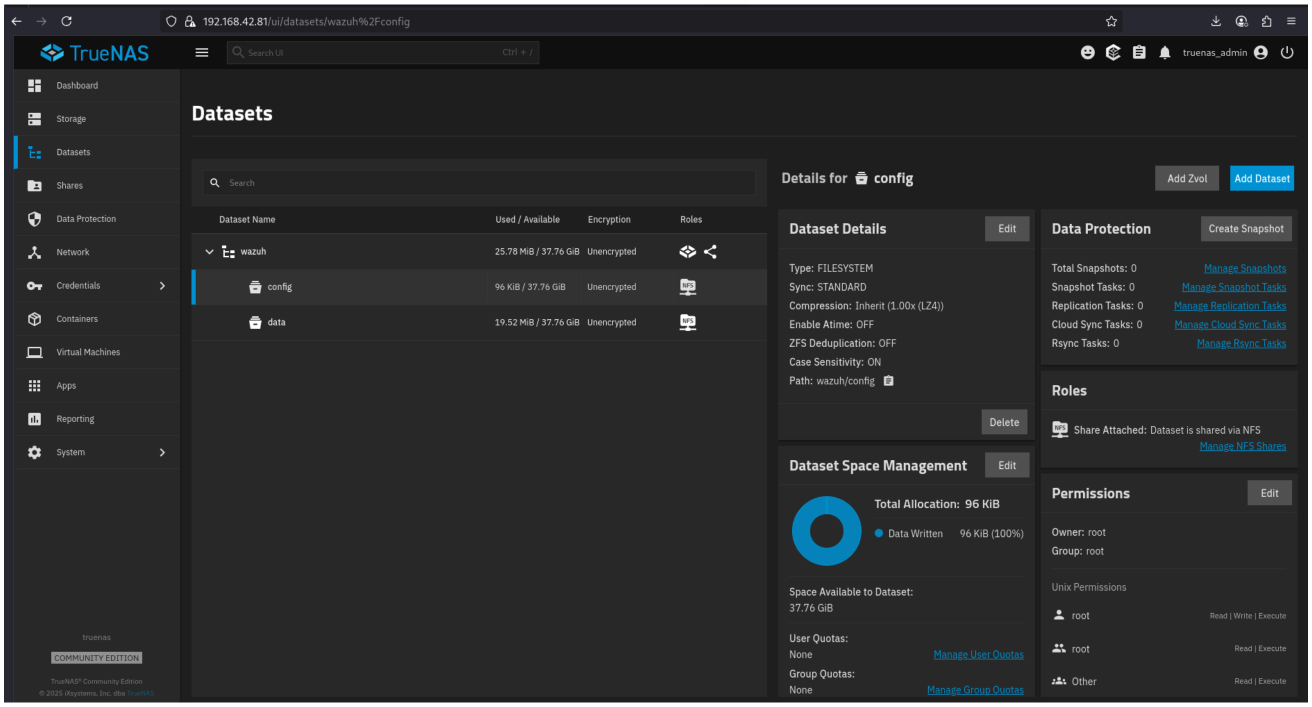
## Swarm mode

Prior to passing to the swarm mode, the docker compose file has been adjusted to take care of the following:

- Overlay networks for cross-nodes communication.
- Minimal exposition of ports to the host.
- NFS based volumes for cross-nodes data and config storage persistence.
- Update and restart policy. (to do)
- Possibility to add constraints on where to run the container. (to do)

### NFS share

I started by making TrueNAS up and running, and setting two datasets (see figure 2).



Datasets

| Dataset Name | Used / Available | Encryption | Roles |
|---|---|---|---|
| wazuh | 25.78 MiB / 37.76 GiB | Unencrypted | |
| config | 96 KiB / 37.76 GiB | Unencrypted | NFS |
| data | 19.52 MiB / 37.76 GiB | Unencrypted | NFS |

Details for config

Dataset Details

Type: FILESYSTEM
Sync: STANDARD
Compression: Inherit (1.00x (LZ4))
Enable Atime: OFF
ZFS Deduplication: OFF
Case Sensitivity: ON
Path: wazuh/config

Data Protection

Total Snapshots: 0 Manage Snapshots
Snapshot Tasks: 0 Manage Snapshot Tasks
Replication Tasks: 0 Manage Replication Tasks
Cloud Sync Tasks: 0 Manage Cloud Sync Tasks
Rsync Tasks: 0 Manage Rsync Tasks

Roles

Share Attached: Dataset is shared via NFS
Manage NFS Shares

Dataset Space Management

Total Allocation: 96 KiB
Data Written 96 KiB (100%)
Space Available to Dataset: 37.76 GiB
User Quotas: None Manage User Quotas
Group Quotas: None Manage Group Quotas

Permissions

Owner: root
Group: root

Unix Permissions

root Read | Write | Execute
root Read | Execute
Other Read | Execute

Figure 2: TrueNAS dashboard

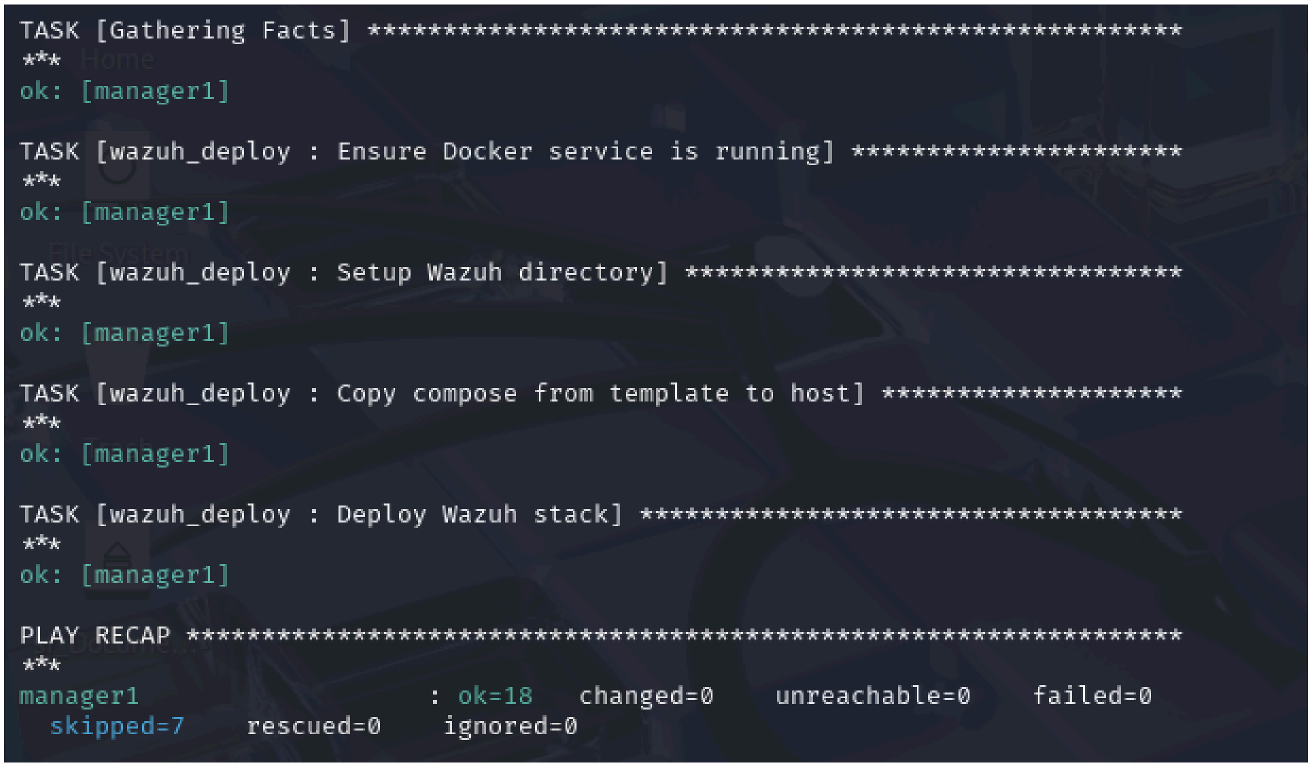
Mounting the NFS server and preparing it for wazuh is one of ansible's roles, see `/ansible`. The NFS server used in this project comes pre installed in **TrueNAS**: a NAS solution that was chosed for the following reseans:

- Truenas with ZFS filesystem enables fast and small in size (differential snapshots ) snapshots if we want to rollback to a storage state. (to do)
- We can use 2 mirrored disks in one ZFS pool. Hence, gaining doubled number of I/O operation per seconds. (to do)
- It integrates well with cloud storage services for remote backup (to do).
  Its worth noting that certificate generation - either by certbot or the wazuh-ssl-generator - ends up by a strict file access policy (400). Hence, the following commands should be run inside the NAS to avoid errors.

```
sudo find /mnt/wazuh/config/configs/nginx/nginx/ssl -type f -exec chmod 444 {} \;
sudo find /mnt/wazuh/config/configs/nginx/letsencrypt -type d -exec chmod 755 {} \;
sudo find /mnt/wazuh/config/configs/wazuh_indexer_ssl_certs -type f -exec chown root:root {} \;
sudo find /mnt/wazuh/config/configs/wazuh_indexer_ssl_certs -type f -exec chmod 755 {} \; # To do: Access granted to only the uid:gid that wazuh_master acts as.
```

## Workflow

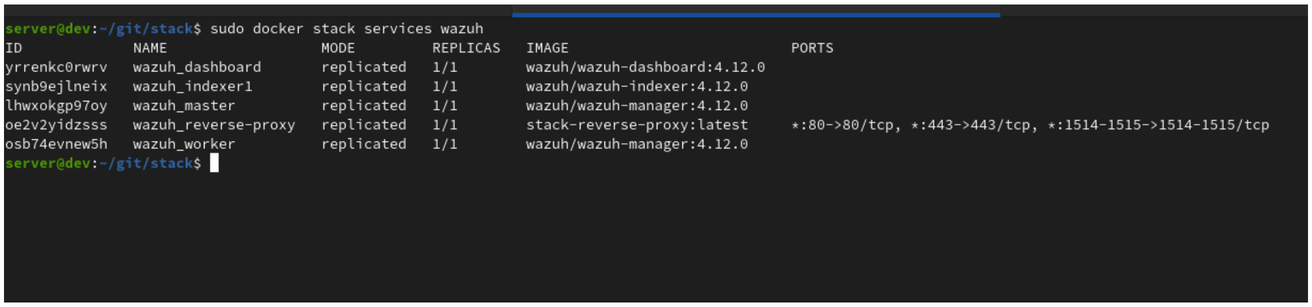
Its time to deploy the cluster, ansible has automated this. I ran: `ansible-playbook site.yaml --ask-pass --ask-become-pass` from within `/ansible` .
Playbooks were executed successfully (see figure 3).



```
TASK [Gathering Facts] *****************************************************
***
ok: [manager1]

TASK [wazuh_deploy : Ensure Docker service is running] *********************
***
ok: [manager1]

TASK [wazuh_deploy : Setup Wazuh directory] ********************************
***
ok: [manager1]

TASK [wazuh_deploy : Copy compose from template to host] *******************
***
ok: [manager1]

TASK [wazuh_deploy : Deploy Wazuh stack] ***********************************
***
ok: [manager1]

PLAY RECAP *****************************************************************
***
manager1                   : ok=18   changed=0    unreachable=0    failed=0
   skipped=7    rescued=0    ignored=0
```

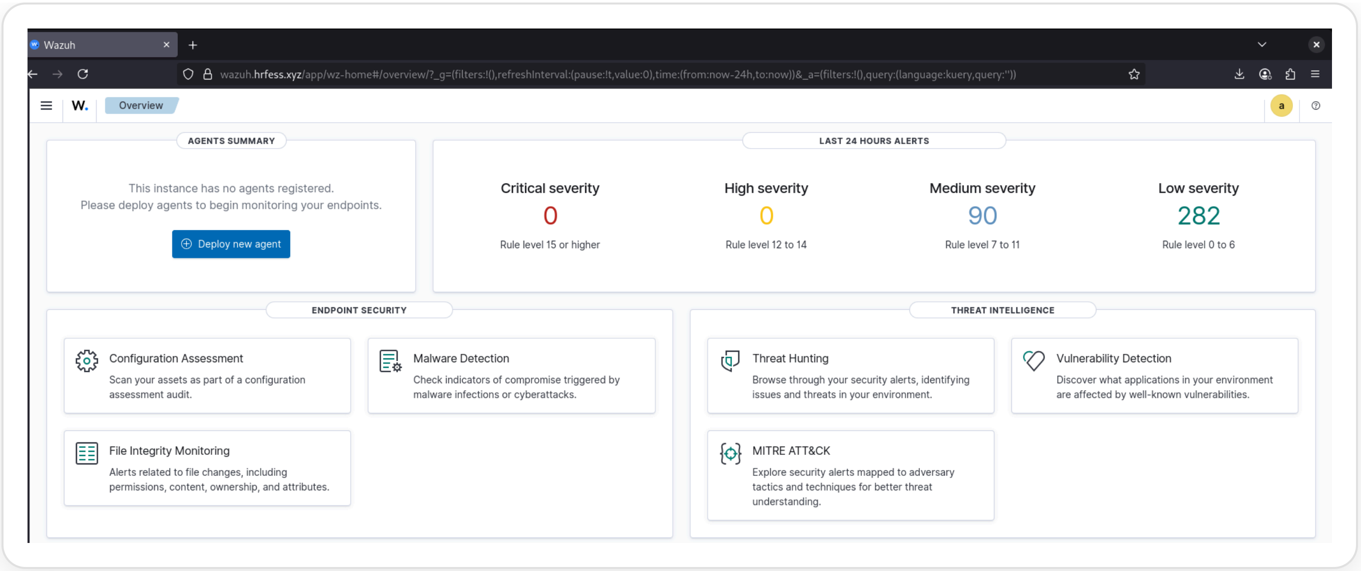
Figure 3: Ansible run

After few a while, I got a fully extensible multi-node version of Wazuh, reachable from the internet. (see figures 4 and 5).



```
server@dev:~/git/stack$ sudo docker stack services wazuh
ID             NAME                  MODE         REPLICAS   IMAGE                             PORTS
yrrenkc0rwrv   wazuh_dashboard       replicated   1/1        wazuh/wazuh-dashboard:4.12.0
synb9ejlneix   wazuh_indexer1        replicated   1/1        wazuh/wazuh-indexer:4.12.0
lhwxokgp97oy   wazuh_master          replicated   1/1        wazuh/wazuh-manager:4.12.0
oe2v2yidzsss   wazuh_reverse-proxy   replicated   1/1        stack-reverse-proxy:latest        *:80->80/tcp, *:443->443/tcp, *:1514-1515->1514-1515/tcp
osb74evnew5h   wazuh_worker          replicated   1/1        wazuh/wazuh-manager:4.12.0
server@dev:~/git/stack$ 
```

Figure 4: Docker services


Wazuh
wazuh.hrfess.xyz/app/wz-home#/overview/?_g=(filters:!(),refreshInterval:(pause:!t,value:0),time:(from:now-24h,to:now))&_a=(filters:!(),query:(language:kuery,query:''))
W.
Overview
a
AGENTS SUMMARY
Deploy new agent
LAST 24 HOURS ALERTS
ENDPOINT SECURITY
THREAT INTELLIGENCE

This instance has no agents registered.
Please deploy agents to begin monitoring your endpoints.

| Critical severity | High severity | Medium severity | Low severity |
|---|---|---|---|
| 0 | 0 | 90 | 282 |
| Rule level 15 or higher | Rule level 12 to 14 | Rule level 7 to 11 | Rule level 0 to 6 |

Configuration Assessment
Scan your assets as part of a configuration assessment audit.

Malware Detection
Check indicators of compromise triggered by malware infections or cyberattacks.

File Integrity Monitoring
Alerts related to file changes, including permissions, content, ownership, and attributes.

Threat Hunting
Browse through your security alerts, identifying issues and threats in your environment.

Vulnerability Detection
Discover what applications in your environment are affected by well-known vulnerabilities.

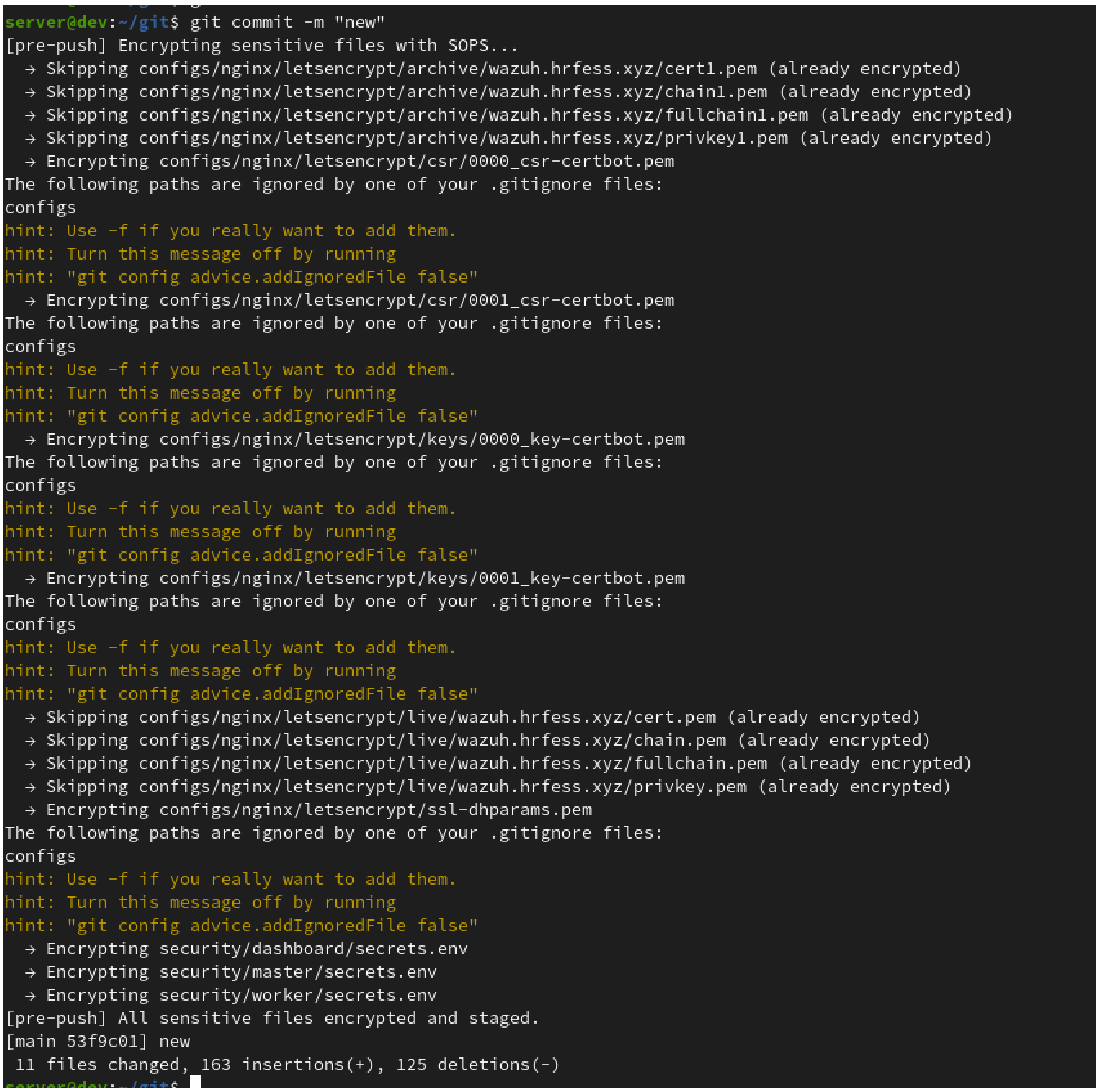
MITRE ATT&CK
Explore security alerts mapped to adversary tactics and techniques for better threat understanding.

Figure 5: Healthy dashboard

## CI/CD

### Pre-push

Before committing changes to the local git's index, SOPS encrypts every project's file that matches the pattern : .pem or secrets.env. This has been achieved via a bash script set as pre-commit hook. Figure 5 shows this operation.

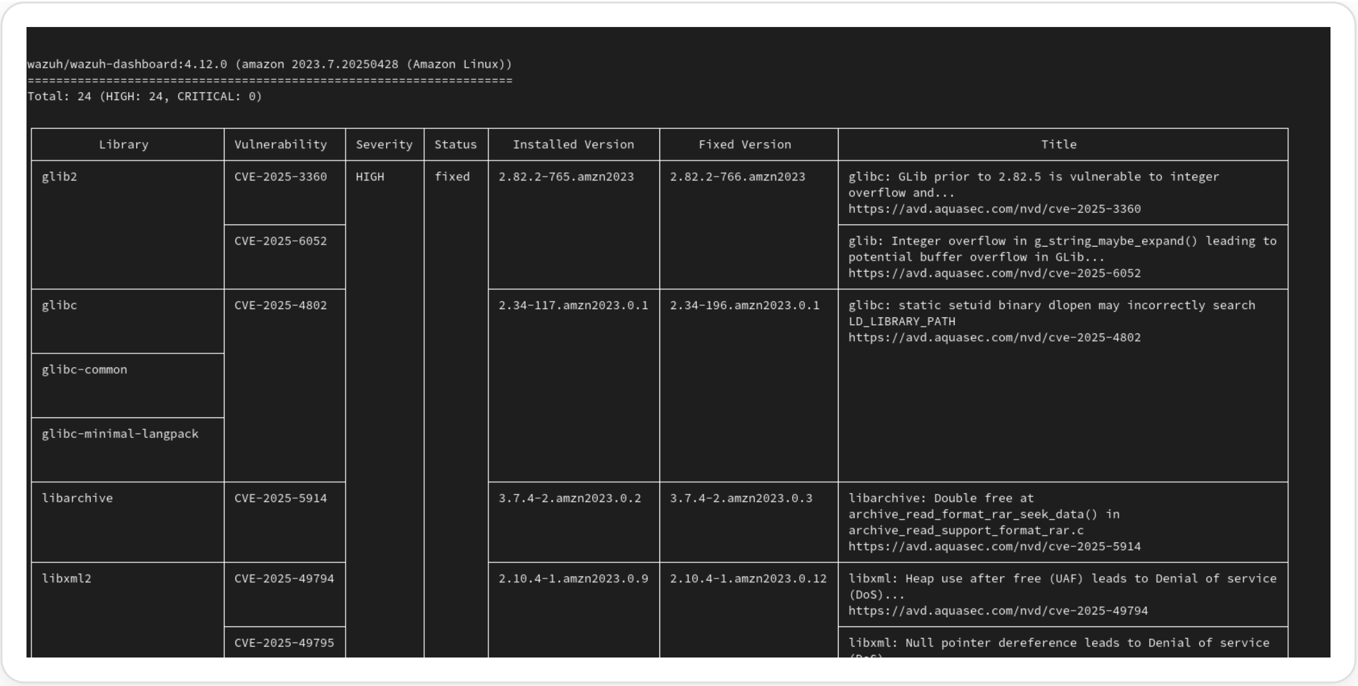


```
server@dev:~/git$ git commit -m "new"
[pre-push] Encrypting sensitive files with SOPS...
  → Skipping configs/nginx/letsencrypt/archive/wazuh.hrfess.xyz/cert1.pem (already encrypted)
  → Skipping configs/nginx/letsencrypt/archive/wazuh.hrfess.xyz/chain1.pem (already encrypted)
  → Skipping configs/nginx/letsencrypt/archive/wazuh.hrfess.xyz/fullchain1.pem (already encrypted)
  → Skipping configs/nginx/letsencrypt/archive/wazuh.hrfess.xyz/privkey1.pem (already encrypted)
  → Encrypting configs/nginx/letsencrypt/csr/0000_csr-certbot.pem
The following paths are ignored by one of your .gitignore files:
configs
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
  → Encrypting configs/nginx/letsencrypt/csr/0001_csr-certbot.pem
The following paths are ignored by one of your .gitignore files:
configs
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
  → Encrypting configs/nginx/letsencrypt/keys/0000_key-certbot.pem
The following paths are ignored by one of your .gitignore files:
configs
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
  → Encrypting configs/nginx/letsencrypt/keys/0001_key-certbot.pem
The following paths are ignored by one of your .gitignore files:
configs
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
  → Skipping configs/nginx/letsencrypt/live/wazuh.hrfess.xyz/cert.pem (already encrypted)
  → Skipping configs/nginx/letsencrypt/live/wazuh.hrfess.xyz/chain.pem (already encrypted)
  → Skipping configs/nginx/letsencrypt/live/wazuh.hrfess.xyz/fullchain.pem (already encrypted)
  → Skipping configs/nginx/letsencrypt/live/wazuh.hrfess.xyz/privkey.pem (already encrypted)
  → Encrypting configs/nginx/letsencrypt/ssl-dhparams.pem
The following paths are ignored by one of your .gitignore files:
configs
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
  → Encrypting security/dashboard/secrets.env
  → Encrypting security/master/secrets.env
  → Encrypting security/worker/secrets.env
[pre-push] All sensitive files encrypted and staged.
[main 53f9c01] new
 11 files changed, 163 insertions(+), 125 deletions(-)
server@dev:~/git$
```

Figure 5: Encryption of secrets with SOPS

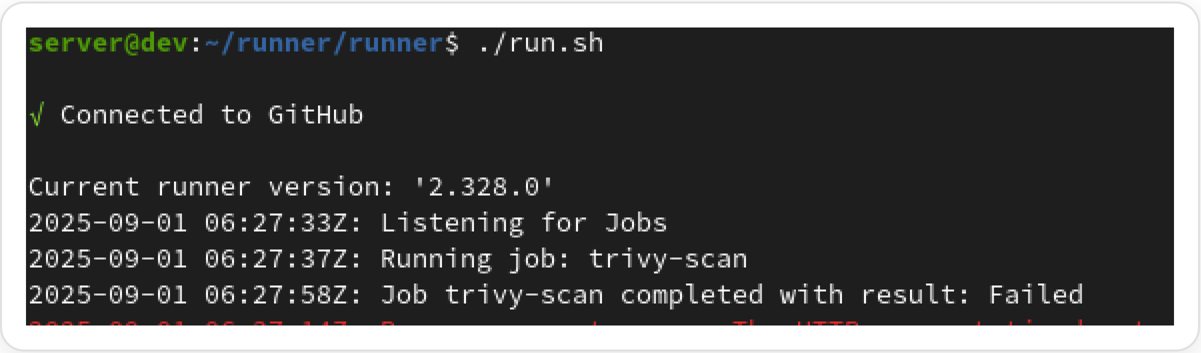
## Post-push

Whenever a push has been performed to the main branch, 2 github actions would be scheduled to a self-hosted runner. The first detects syntax errors in yml file by invoking yamllint utility. The second scans all the images defined on stack.yml for high/critical vulnerabilities. It does so using trivy. Its worth noting that the dependency of the latest image of wazuh-dashboard to Amazon Linux, makes it prone to high vulnerabilities. The vulnerabilities present in core libraries of the OS (see figure 6).



```
wazuh/wazuh-dashboard:4.12.0 (amazon 2023.7.20250428 (Amazon Linux))
=====================================================================
Total: 24 (HIGH: 24, CRITICAL: 0)

┌───────────────────────┬────────────────┬──────────┬────────┬─────────────────────┬──────────────────────┬────────────────────────────────────────────────────────────┐
│        Library        │ Vulnerability  │ Severity │ Status │  Installed Version  │    Fixed Version     │                           Title                            │
├───────────────────────┼────────────────┼──────────┼────────┼─────────────────────┼──────────────────────┼────────────────────────────────────────────────────────────┤
│ glib2                 │ CVE-2025-3360  │ HIGH     │ fixed  │ 2.82.2-765.amzn2023 │ 2.82.2-766.amzn2023  │ glibc: GLib prior to 2.82.5 is vulnerable to integer       │
│                       │                │          │        │                     │                      │ overflow and...                                            │
│                       │                │          │        │                     │                      │ https://avd.aquasec.com/nvd/cve-2025-3360                  │
│                       ├────────────────┤          │        │                     │                      ├────────────────────────────────────────────────────────────┤
│                       │ CVE-2025-6052  │          │        │                     │                      │ glib: Integer overflow in g_string_maybe_expand() leading to │
│                       │                │          │        │                     │                      │ potential buffer overflow in GLib...                       │
│                       │                │          │        │                     │                      │ https://avd.aquasec.com/nvd/cve-2025-6052                  │
├───────────────────────┼────────────────┤          │        ├─────────────────────┼──────────────────────┼────────────────────────────────────────────────────────────┤
│ glibc                 │ CVE-2025-4802  │          │        │ 2.34-117.amzn2023.0.1 │ 2.34-196.amzn2023.0.1 │ glibc: static setuid binary dlopen may incorrectly search │
│                       │                │          │        │                     │                      │ LD_LIBRARY_PATH                                            │
│                       │                │          │        │                     │                      │ https://avd.aquasec.com/nvd/cve-2025-4802                  │
├───────────────────────┤                │          │        │                     │                      │                                                            │
│ glibc-common          │                │          │        │                     │                      │                                                            │
│                       │                │          │        │                     │                      │                                                            │
├───────────────────────┤                │          │        │                     │                      │                                                            │
│ glibc-minimal-langpack │               │          │        │                     │                      │                                                            │
│                       │                │          │        │                     │                      │                                                            │
├───────────────────────┼────────────────┤          │        ├─────────────────────┼──────────────────────┼────────────────────────────────────────────────────────────┤
│ libarchive            │ CVE-2025-5914  │          │        │ 3.7.4-2.amzn2023.0.2 │ 3.7.4-2.amzn2023.0.3 │ libarchive: Double free at                                │
│                       │                │          │        │                     │                      │ archive_read_format_rar_seek_data() in                     │
│                       │                │          │        │                     │                      │ archive_read_support_format_rar.c                          │
│                       │                │          │        │                     │                      │ https://avd.aquasec.com/nvd/cve-2025-5914                  │
├───────────────────────┼────────────────┤          │        ├─────────────────────┼──────────────────────┼────────────────────────────────────────────────────────────┤
│ libxml2               │ CVE-2025-49794 │          │        │ 2.10.4-1.amzn2023.0.9 │ 2.10.4-1.amzn2023.0.12 │ libxml: Heap use after free (UAF) leads to Denial of service │
│                       │                │          │        │                     │                      │ (DoS)...                                                   │
│                       │                │          │        │                     │                      │ https://avd.aquasec.com/nvd/cve-2025-49794                 │
│                       ├────────────────┤          │        │                     │                      ├────────────────────────────────────────────────────────────┤
│                       │ CVE-2025-49795 │          │        │                     │                      │ libxml: Null pointer dereference leads to Denial of service │
```

Figure 6:Highs in the dashboard



```
server@dev:~/runner/runner$ ./run.sh

√ Connected to GitHub

Current runner version: '2.328.0'
2025-09-01 06:27:33Z: Listening for Jobs
2025-09-01 06:27:37Z: Running job: trivy-scan
2025-09-01 06:27:58Z: Job trivy-scan completed with result: Failed
```

## Ansible

On a host with python3.8 installed, I started by running:
`ansible-galaxy collection install -r collections/requirements.yaml`
This would install external requirements for ansible. The defined workflows handle both update of stack.yml and first deployment.
If it is first deployment:

1. Set `first_deployment` of `Setup NFS server task` inside site.yml to true.
2. Put `configs` and `security` inside `ansible/roles/mount_nfs/templates`.