iu-cf-lab-04-Iskander_Nafikov

- **Name**: Iskander Nafikov
- **E-mail**: i.nafikov@innopolis.university
- **GitHub**: https://github.com/iskanred

CCF Lab 4 - Incident response and log management

Task description

In this lab, you will set up a complete environment for detecting and responding to security incidents using the Wazuh security platform. You will simulate real-world attacks, configure automated responses, and explore best practices for managing logs and system alerts. This hands-on exercise is designed to reinforce key concepts in incident detection, active response, and log retention.

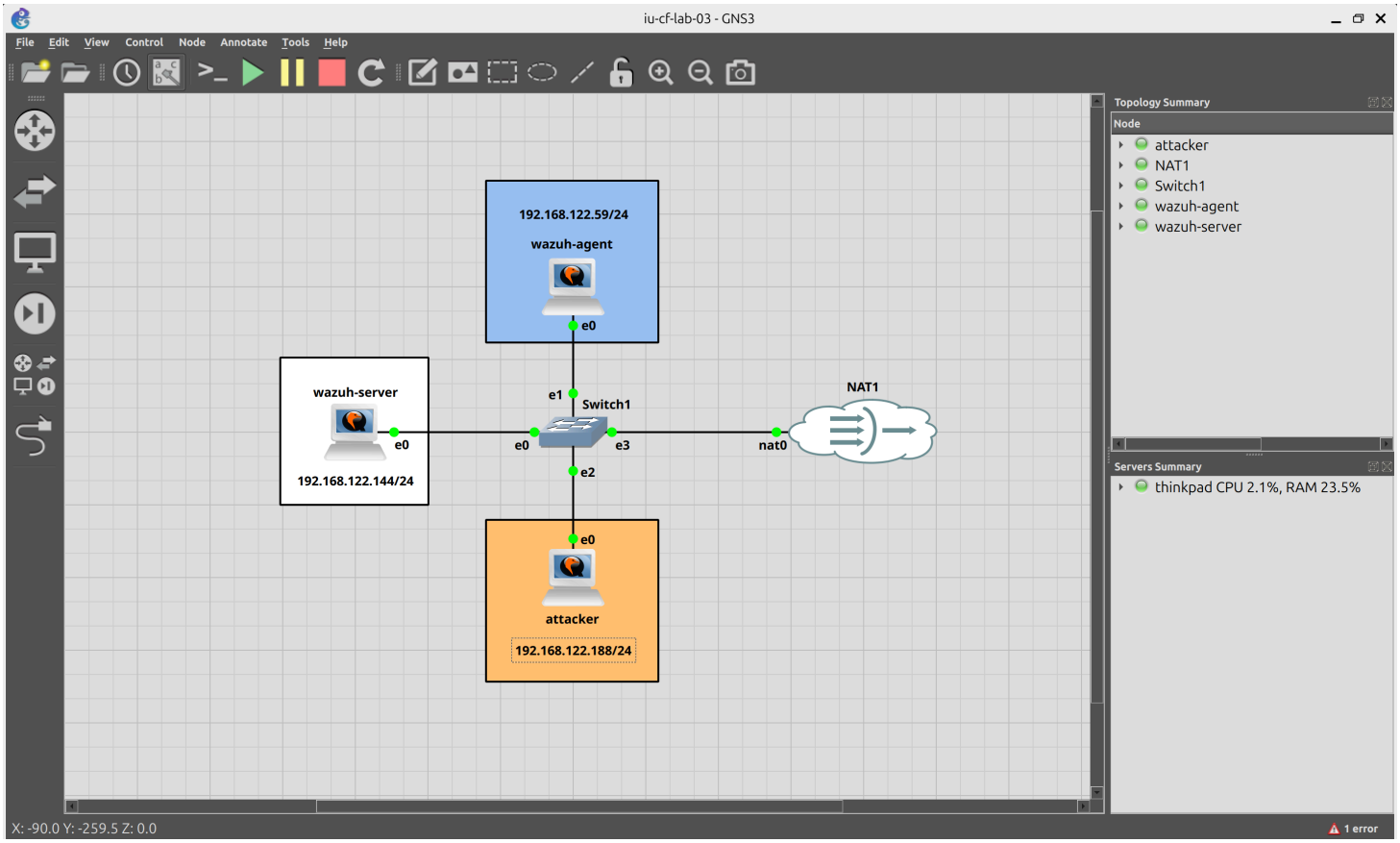
Task 1 - Preparation

1.

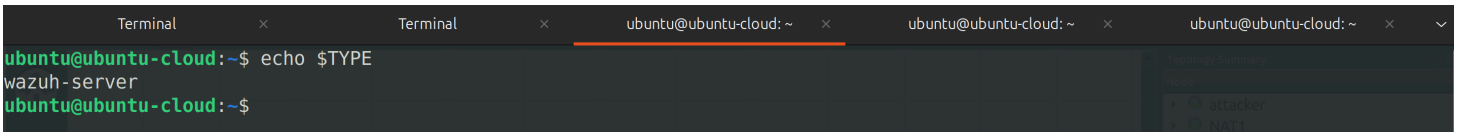
Task description

Deploy Wazuh central components (server, indexer, and dashboard) on your host machine or use an alternate method like Docker.

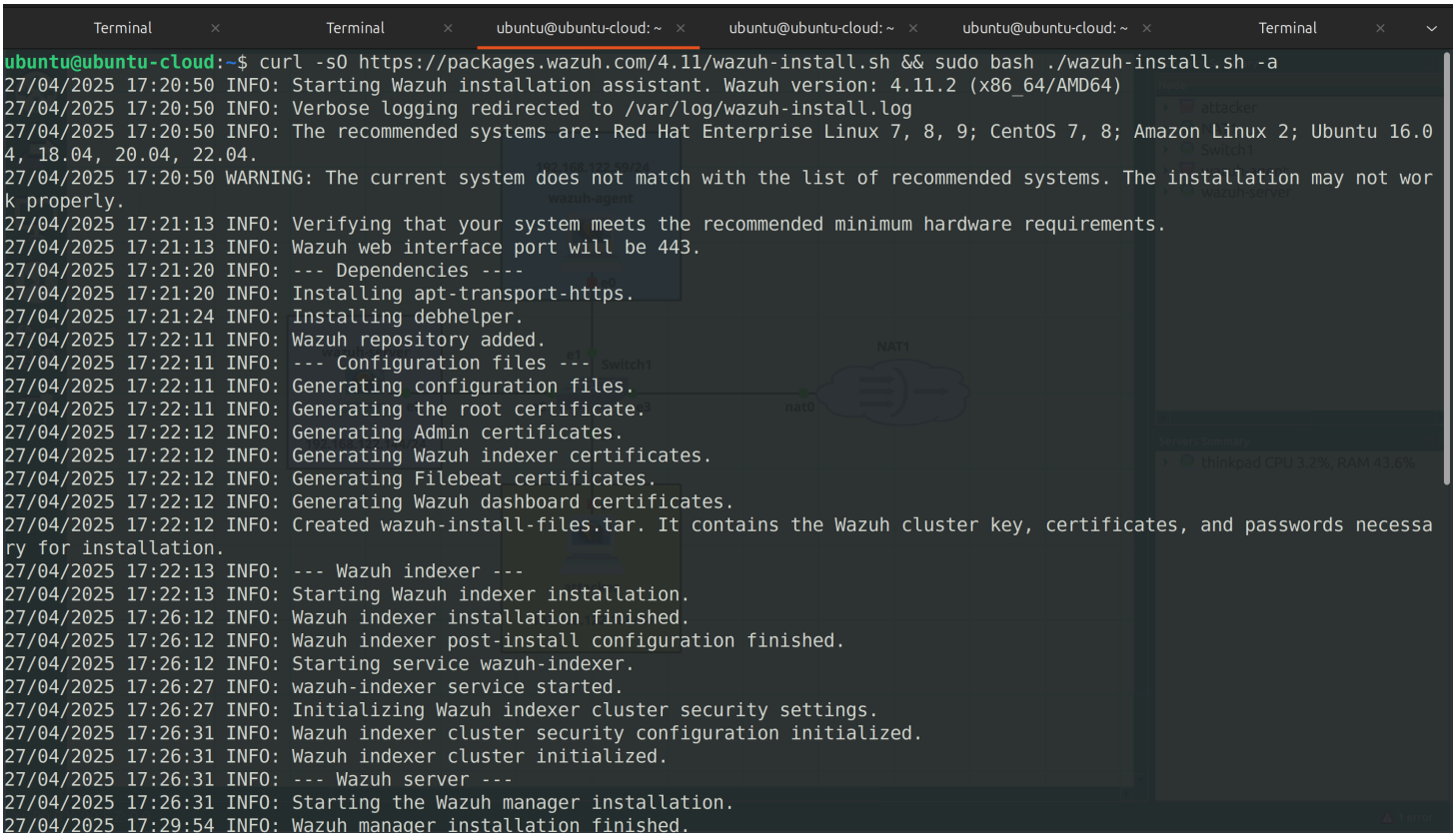
- First, I deployed 3 VMs inside the GNS3



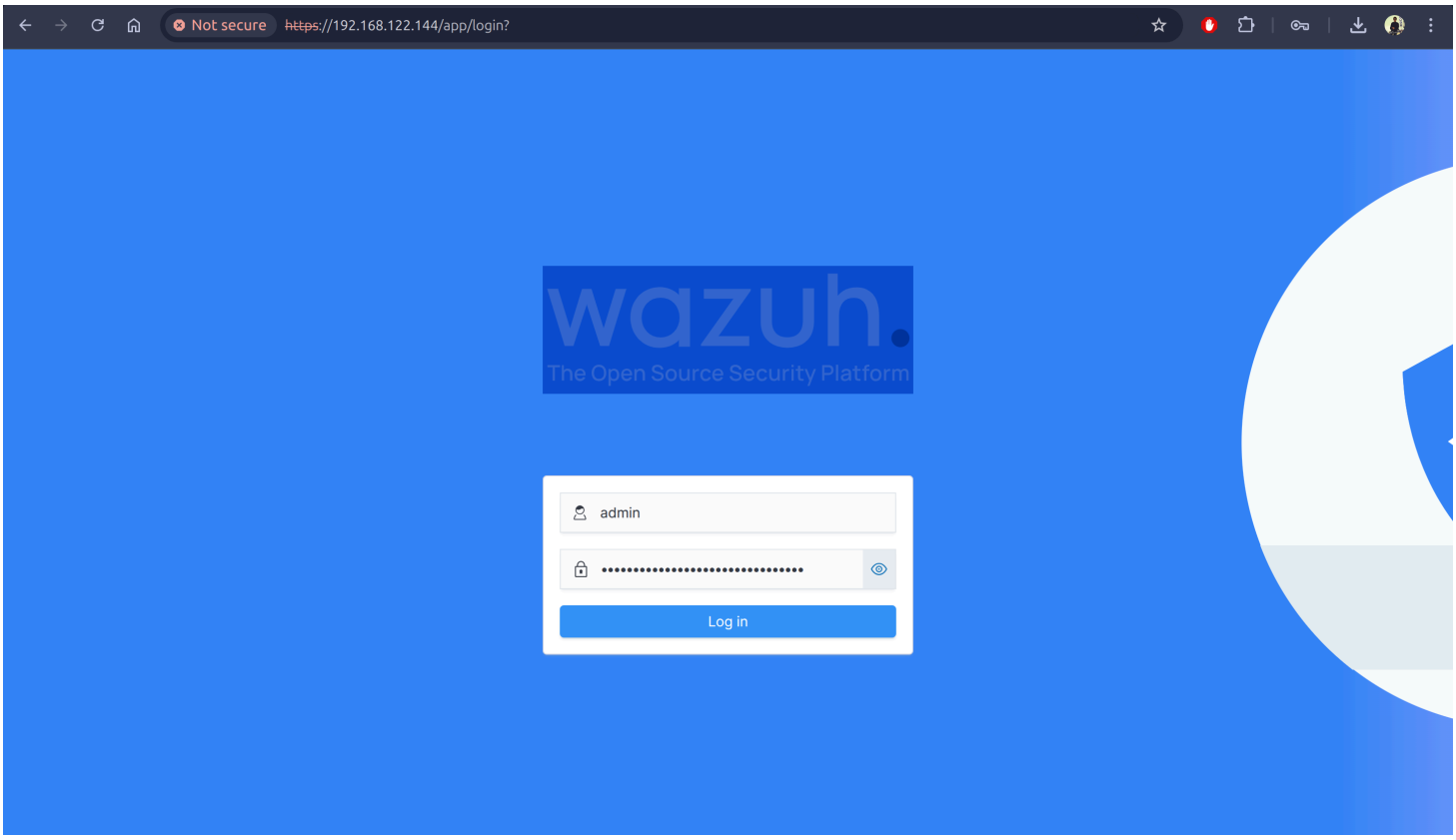
- I entered the Wazuh server (manager + indexer + dashboard) VM and exported `$TYPE` environment variable



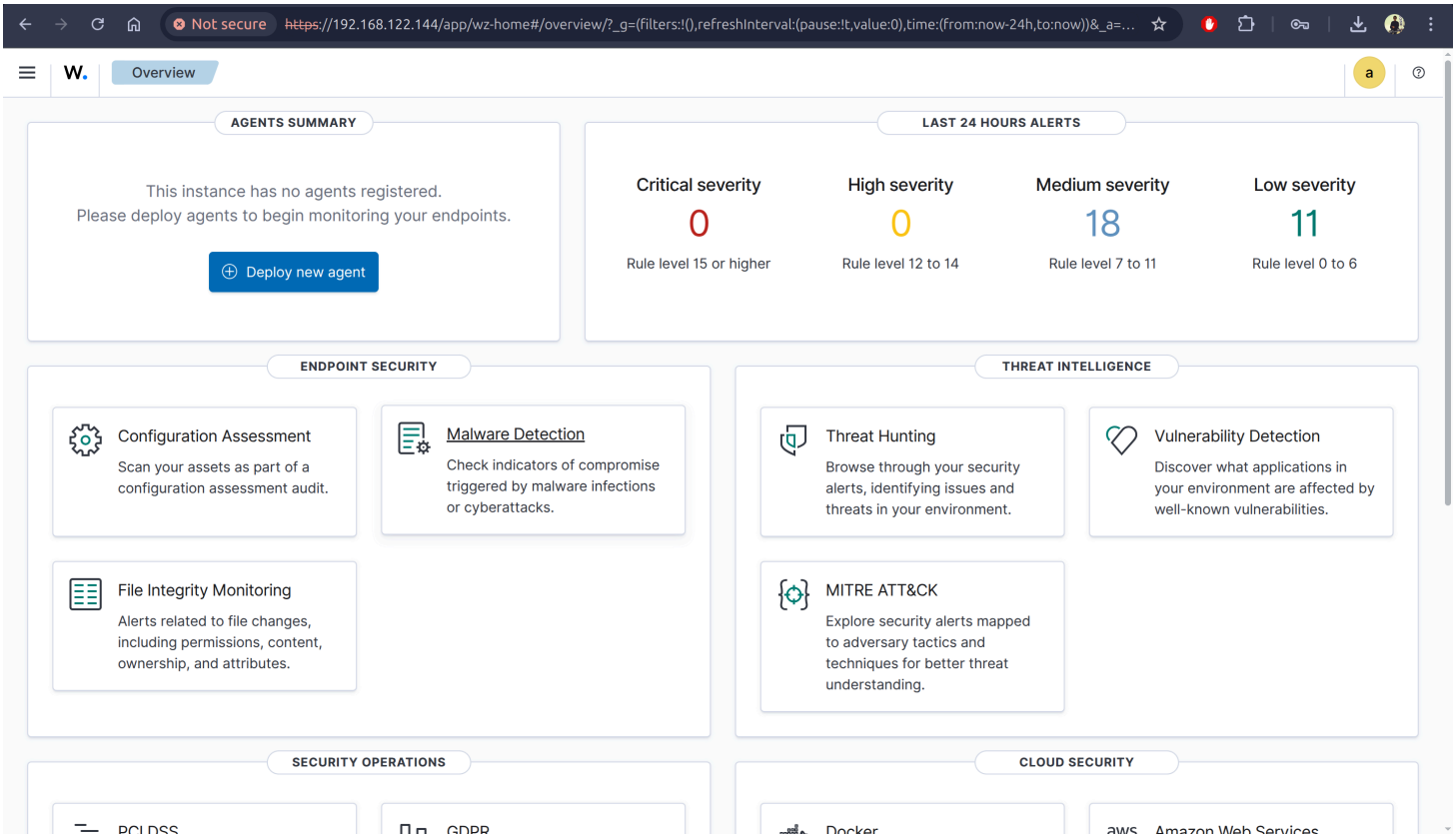
- Then I installed the Wazuh following the Quickstart instruction from the official website



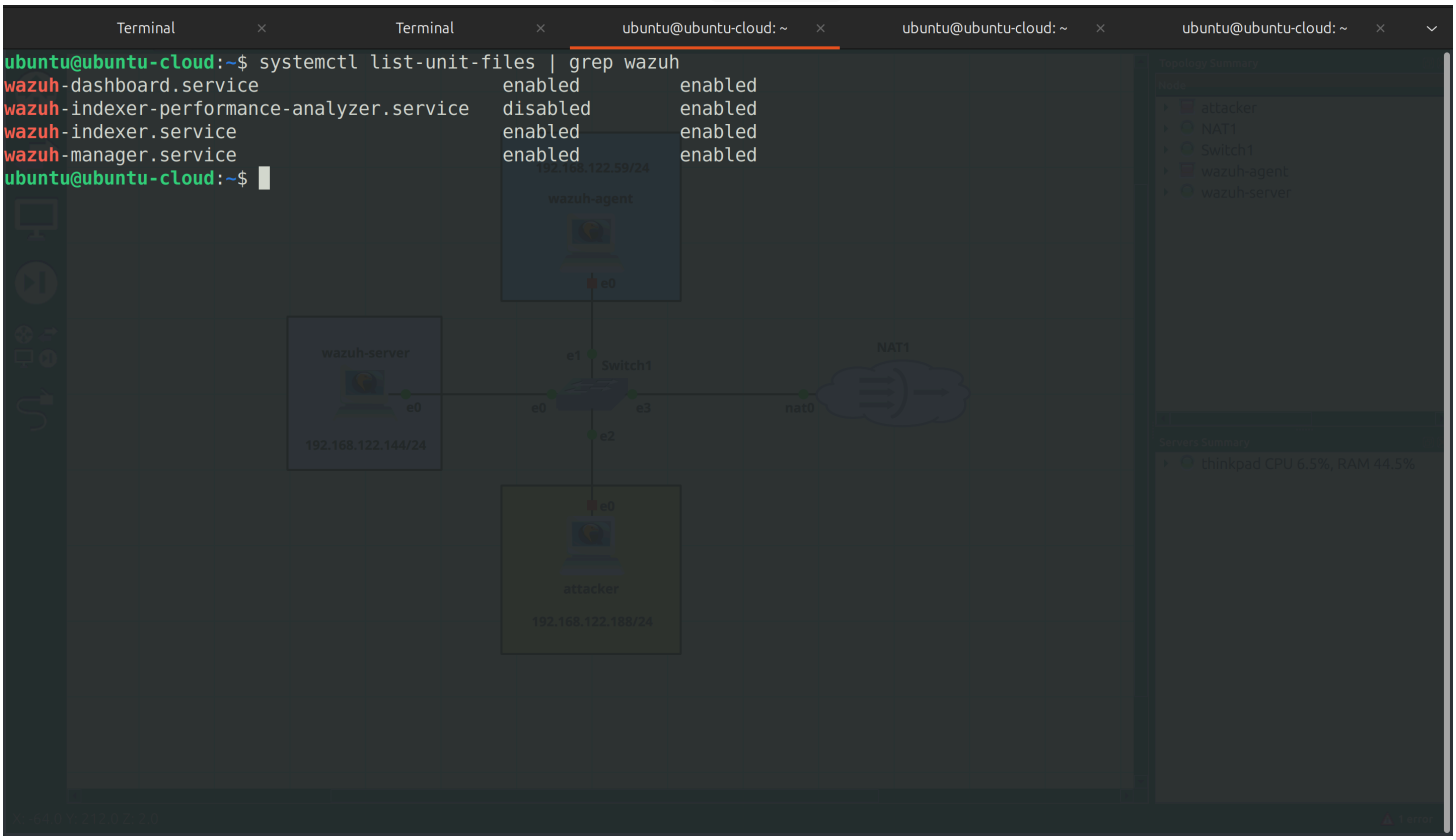
- Finally, I was able to access Wazuh dashboard through HTTPS



- After logging in with generated password I went to the Dashboard's



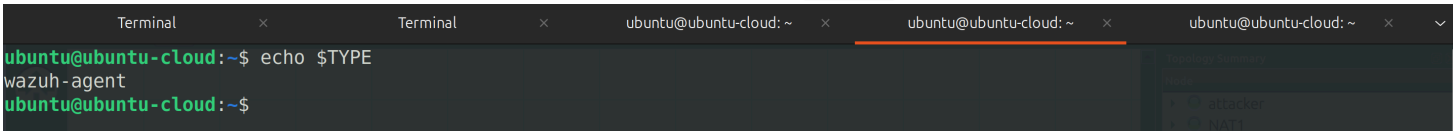
- As we see Wazuh components were deployed as `systemd` services



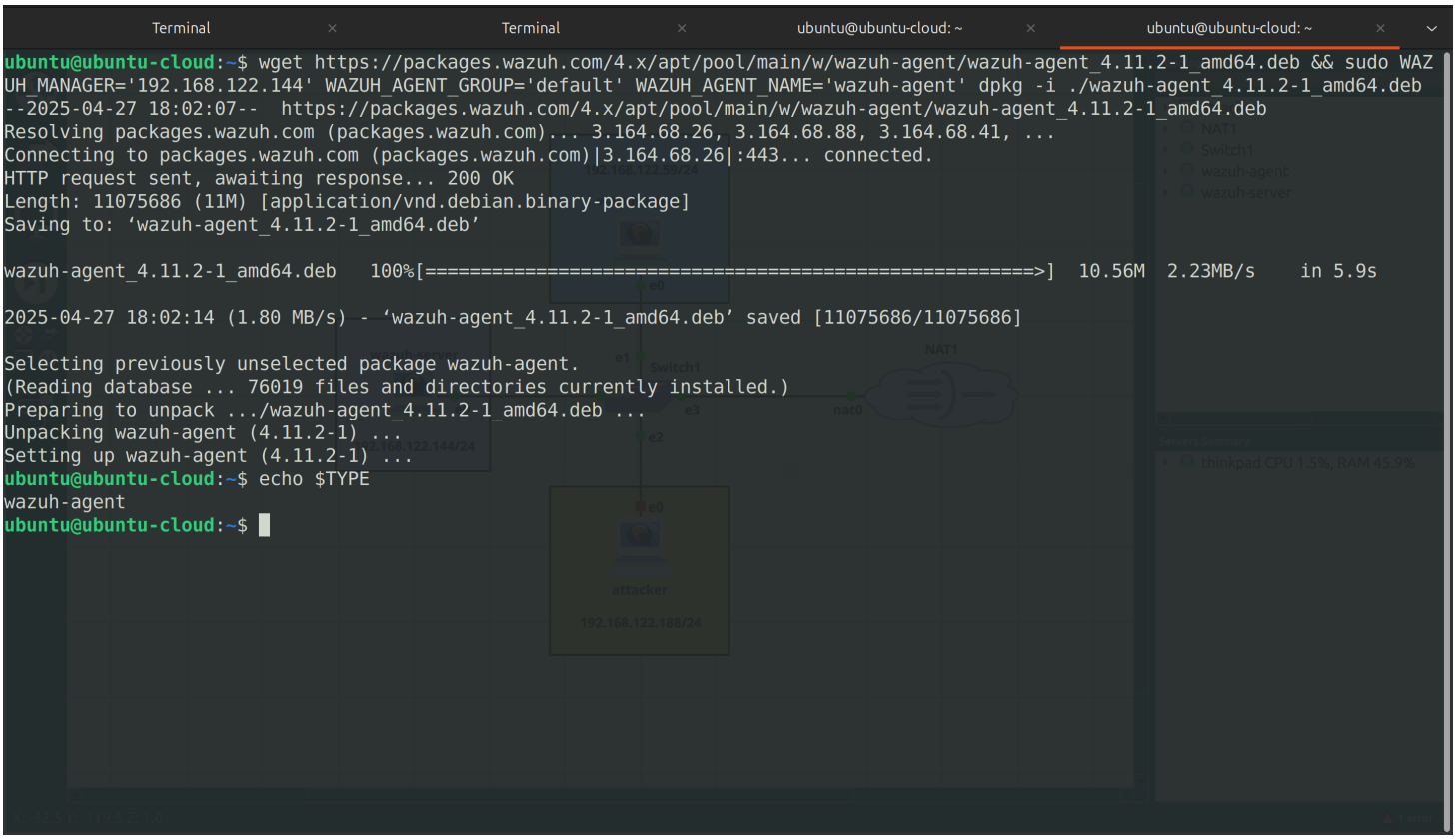
2.

> ### Task description
>
> Set up a second machine with a Unix-like OS, install the Wazuh agent, and enroll it with the Wazuh manager. Enable SSH and create a test user account.

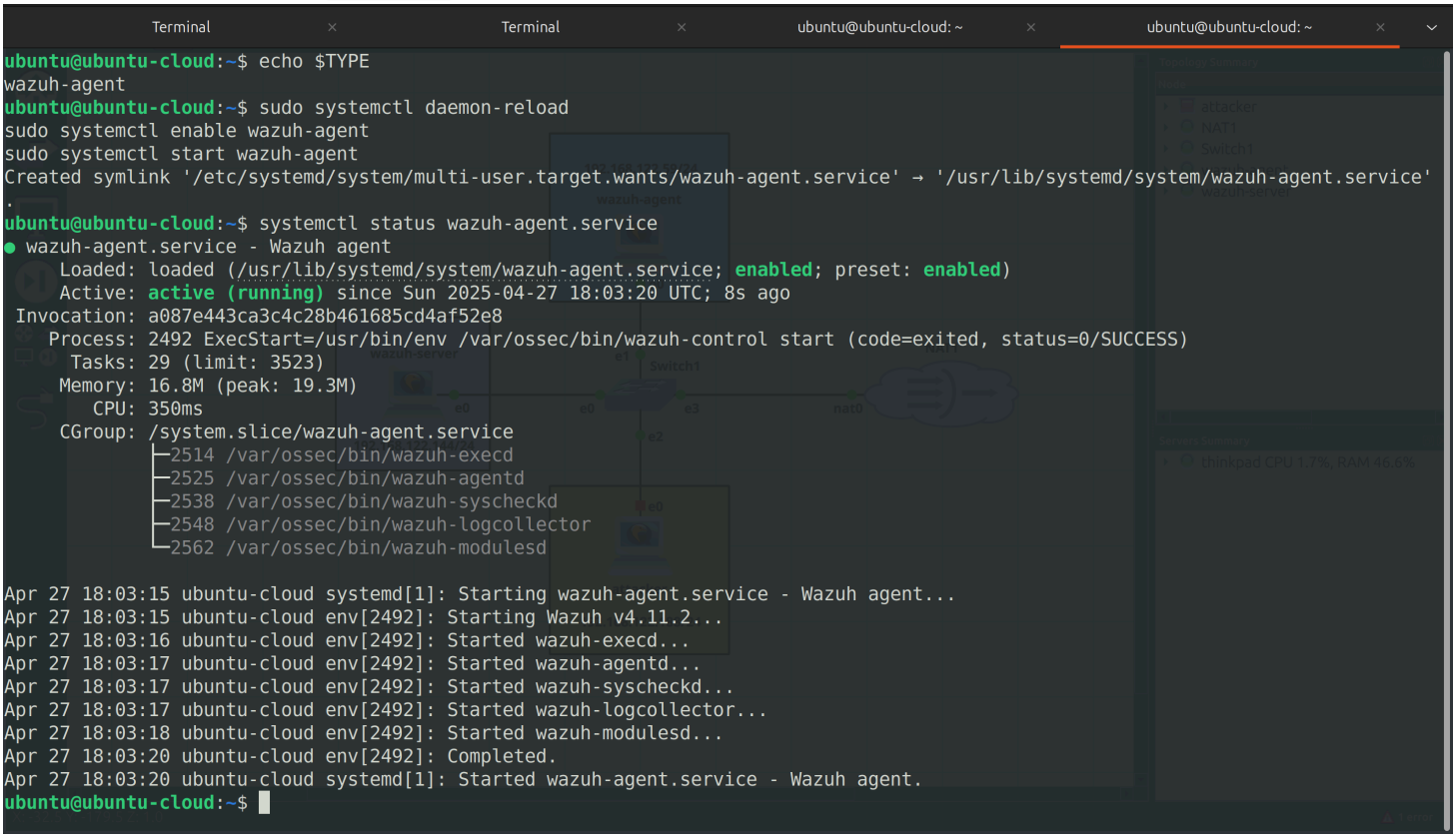
- I entered the Wazuh agent for Ubuntu VM and exported `$TYPE` environment variable



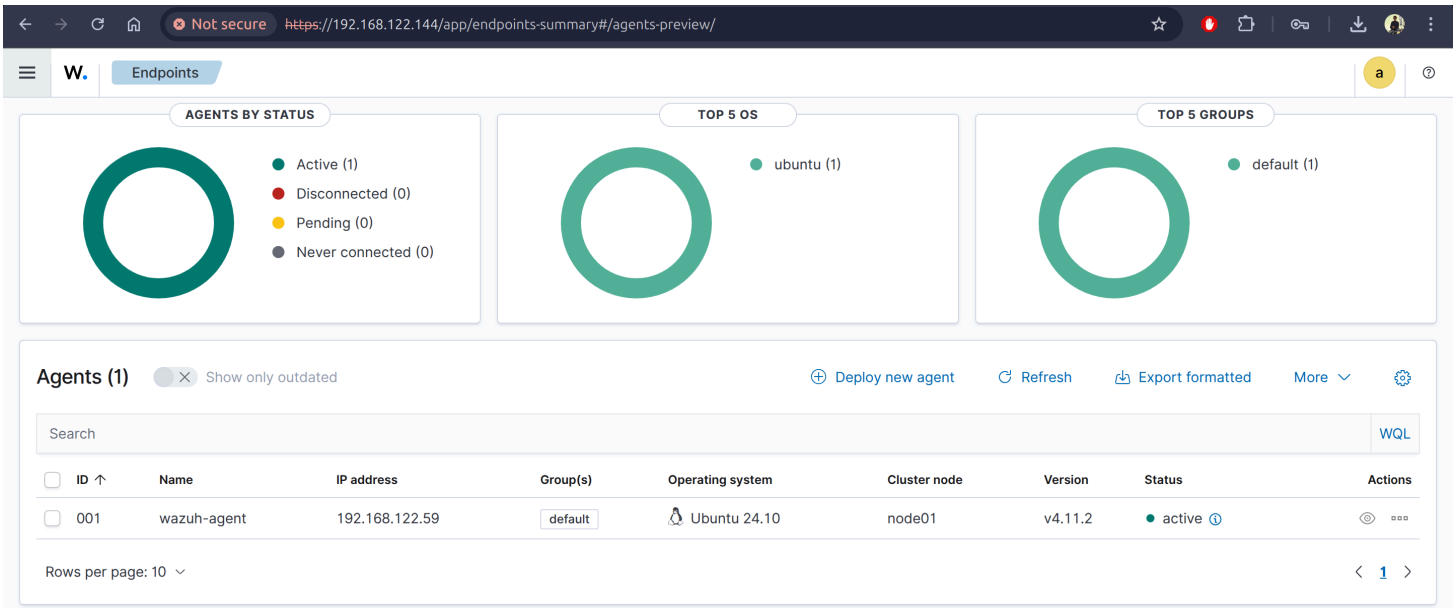
- I deployed an agent using the generated command on the Wazuh Dasbhoard



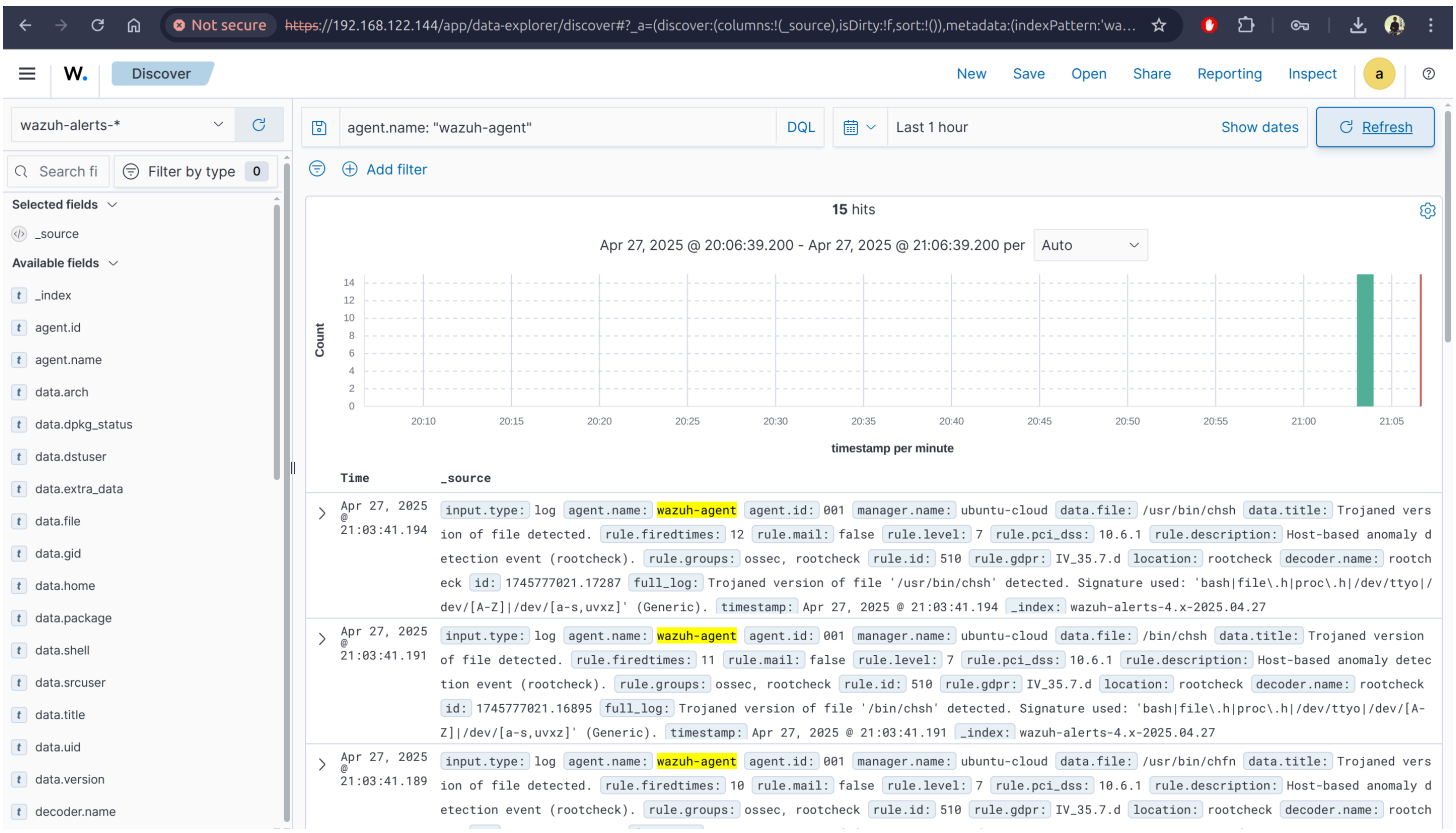
- Finally, I launched the `systemd` service of the Wazuh Agent



- After this I could see this agent on the Wazuh Dasbhoard



- Also, I could see its logs

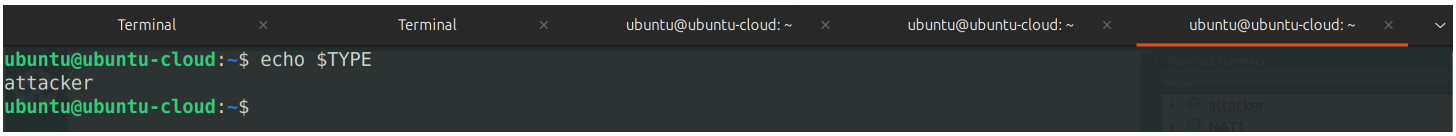


3.

Task description

Prepare a third machine to act as the "attacker" endpoint to simulate cyber attacks. You can collaborate with your colleagues to simulate attacks if you don't have enough hardware resources for three machines.

- I entered the attacker's Ubuntu VM and exported `$TYPE` environment variable



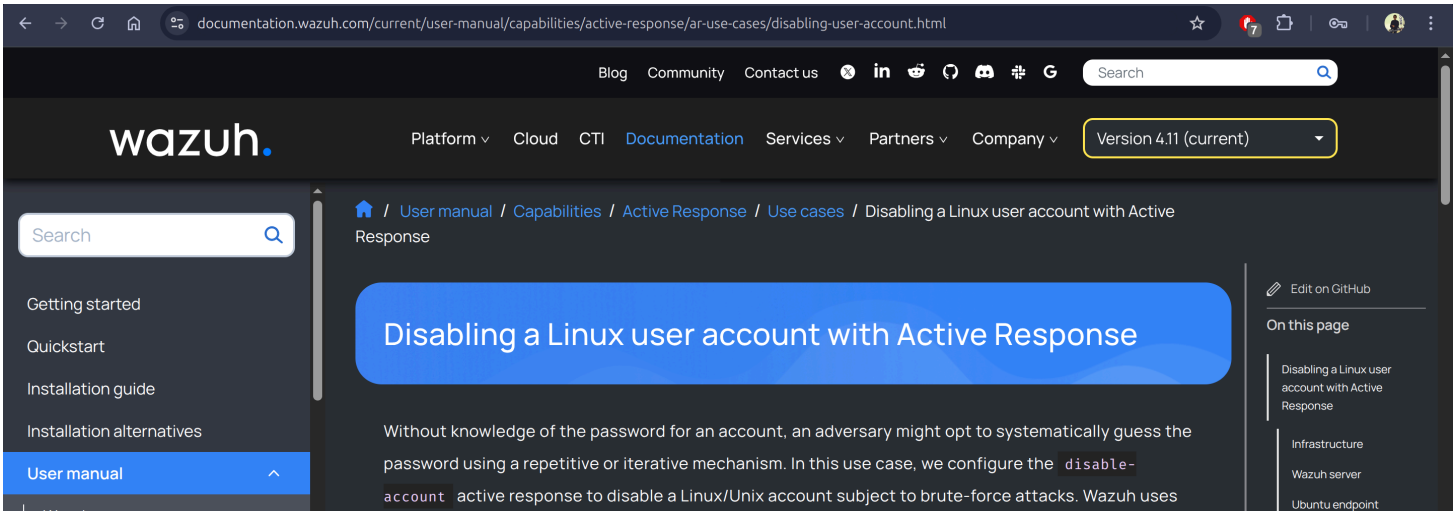
Task 2 - Configure active response

1.

> **Task description**
>
> Enable the Wazuh active response feature to disable a user account for 10 minutes when a brute force attempt is detected.

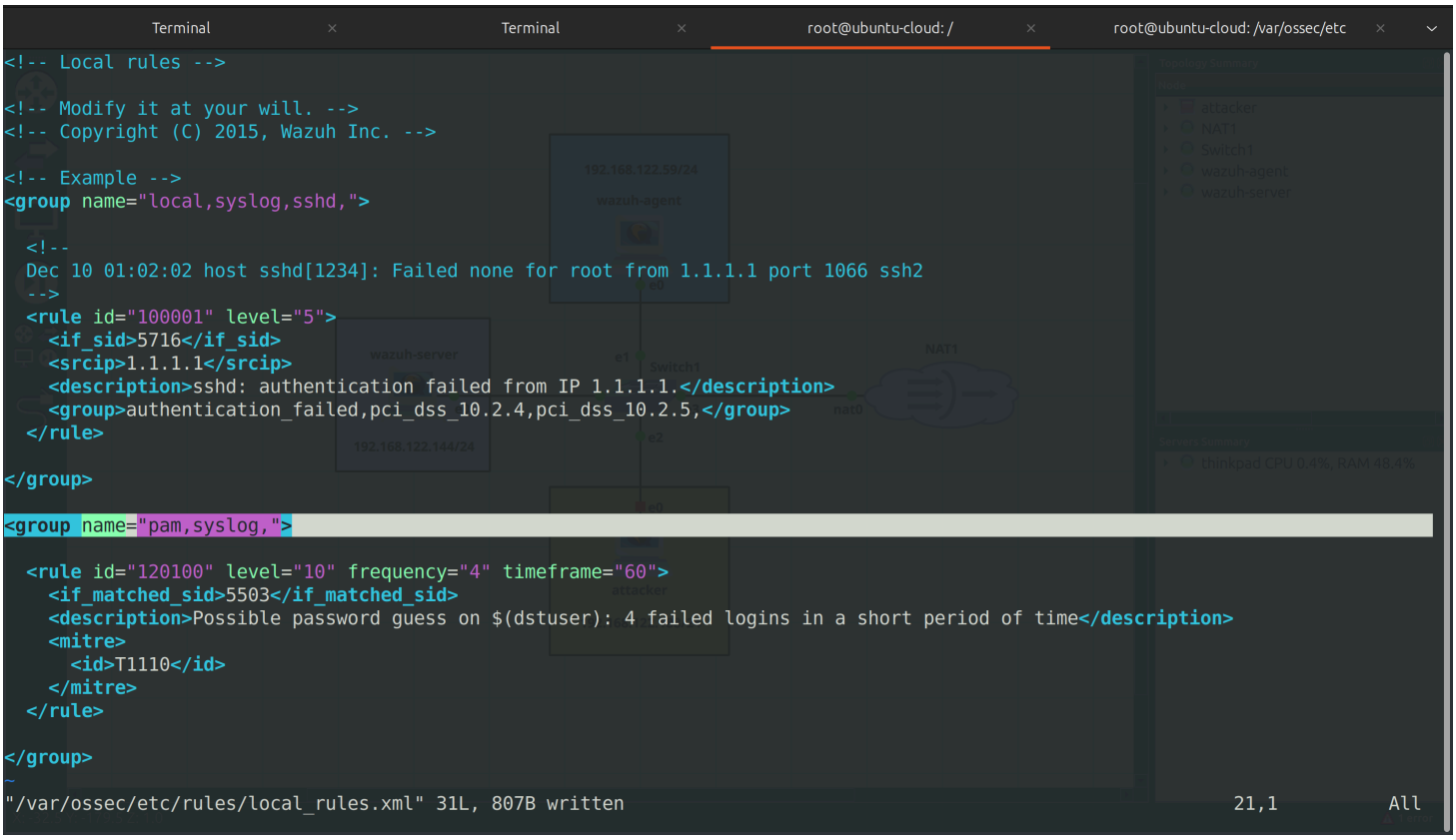
Configuration

- Firstly, I found an instruction how to disable Linux user account in case of brute force attack on the Wazuh official website

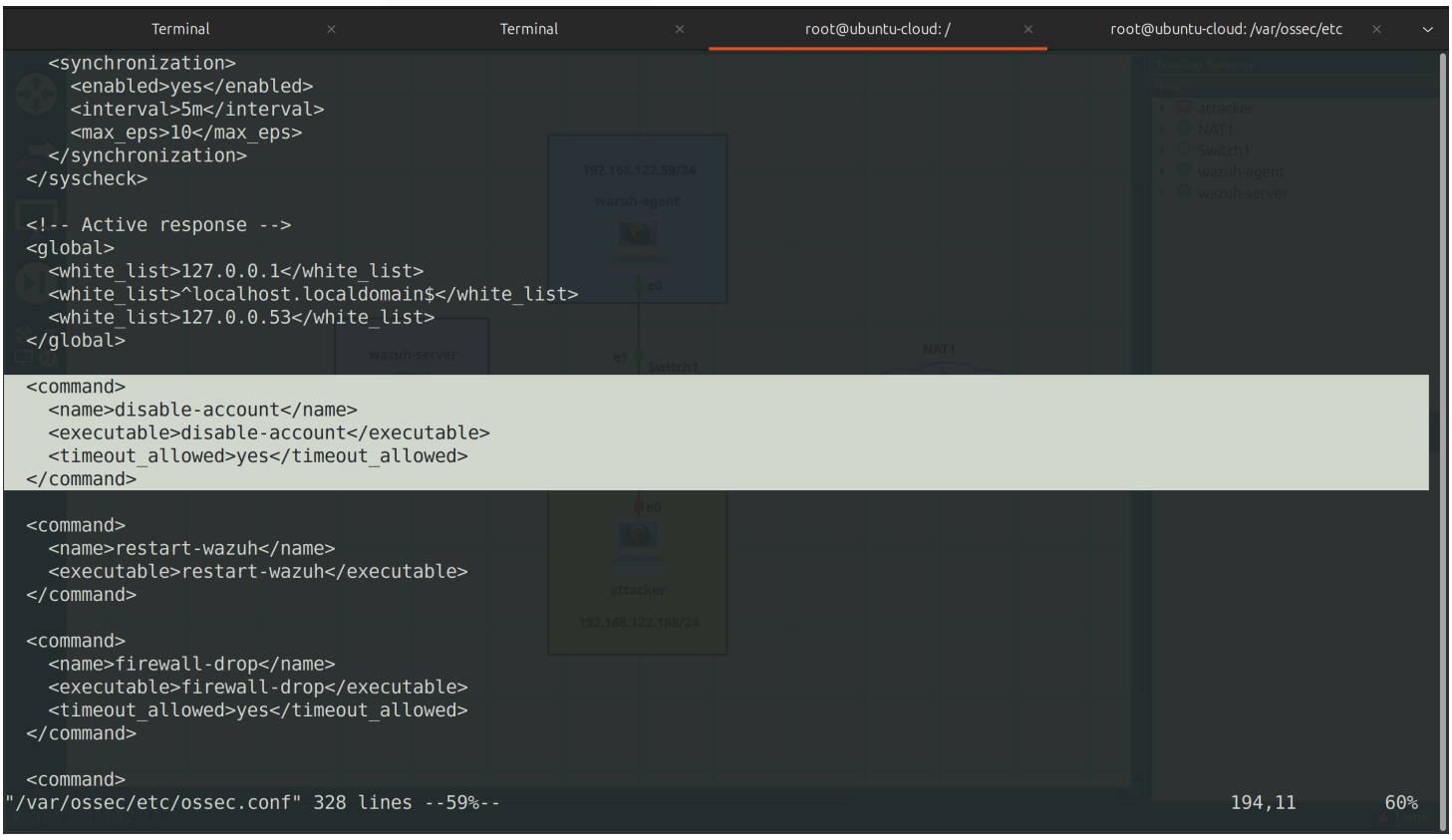


- Following the instruction I added a new alert group and a rule inside it that detects "possible password guess" for a specific user on the Wazuh Server. I made the rule triggered after 4 failed login tries in

one minute. Also, I assigned the MIT&RE attack ID $= T1110$ which is a Brute Force.

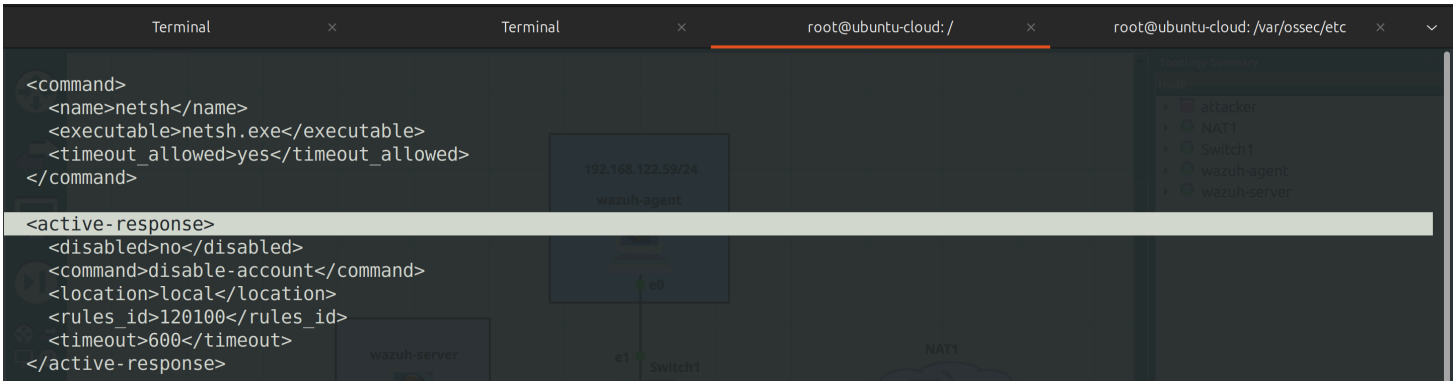


- I made sure that there is a `disable-account` command listed inside the `/var/ossec/etc/ossec.conf

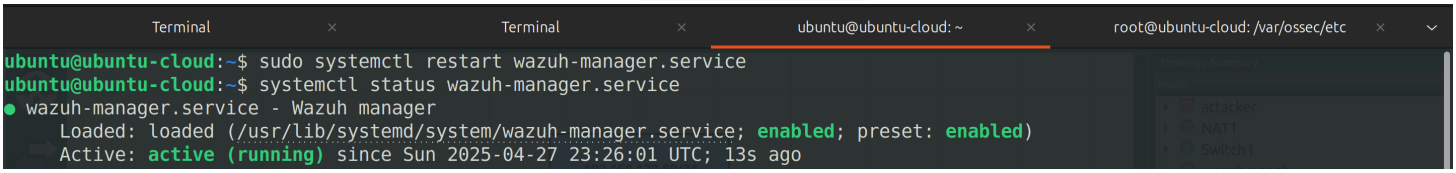


- Then I added a new active response action inside the `/var/ossec/etc/ossec.conf`. This action executes `disable-account` command whenever the rule with ID $= 120100$ (which is a rule I defined to detect a brute force attack) is triggered. Also, we that this action has a required timeout for 10 minutes (600 seconds). After that period, the Active Response reverts its action and re-enables the

account.



- Finally, to apply new changes I restarted the `wazuh-manager` daemon

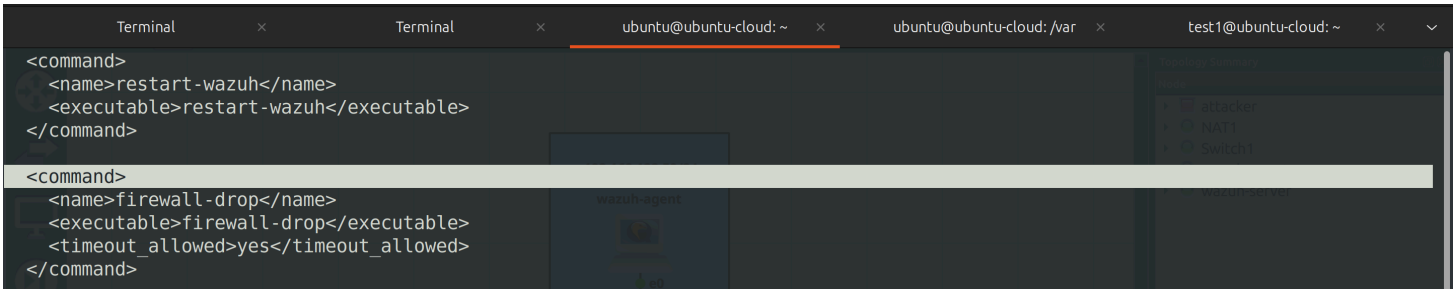


2.

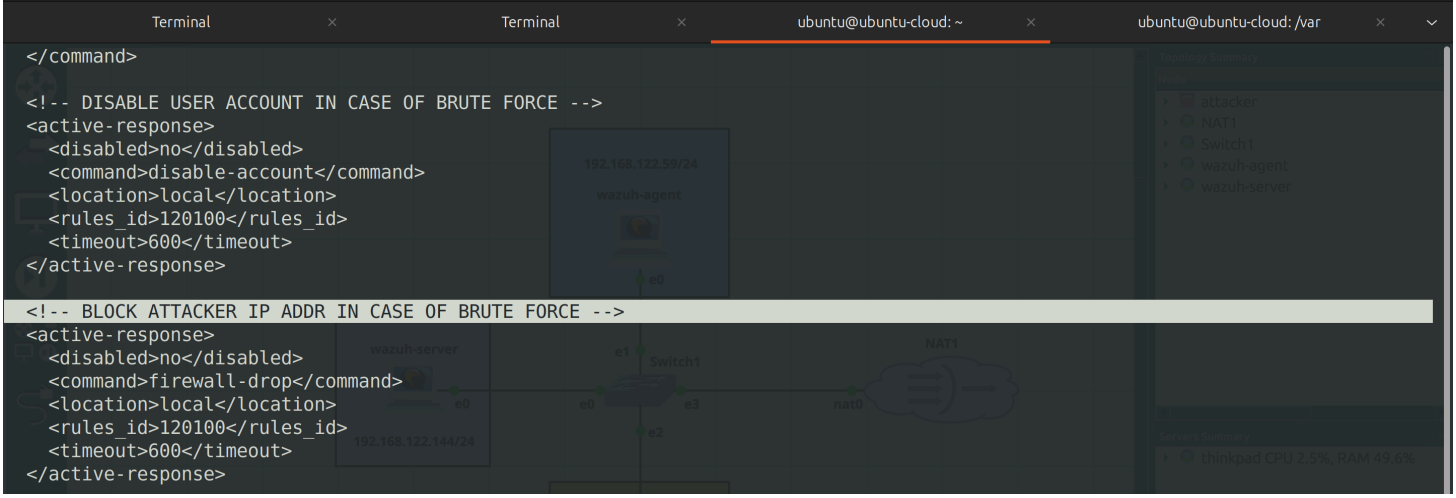
Task description

Also configure it to block the attacker's IP address for 10 minutes.

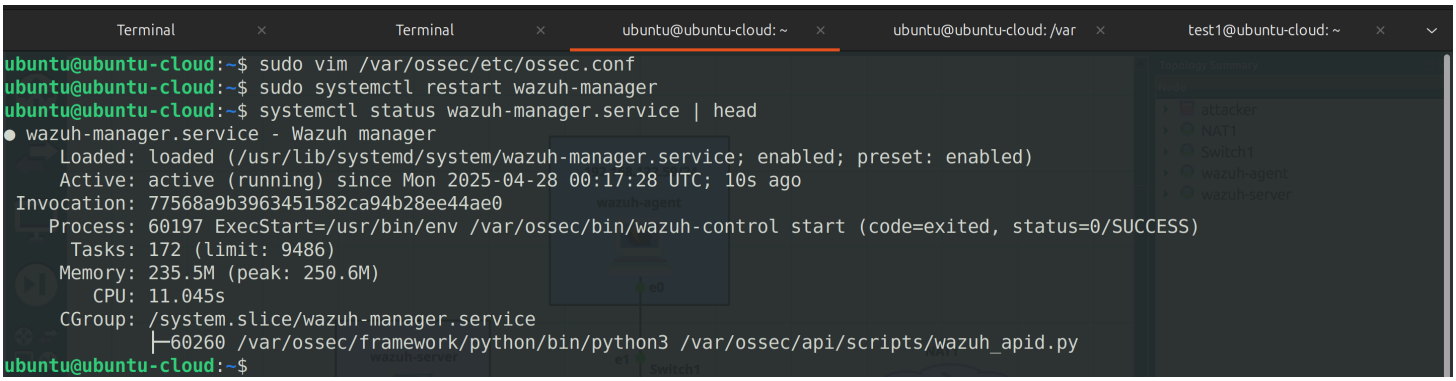
- Now I need to perform similar action. The packets from attacker's IP must be dropped for 10 minutes if the brute force is detected. I found an instruction that is pretty similar to the previous one on the Wazuh official website.
- First, let's start with making sure that the `firewall-drop` command exists inside the Wazuh Manager config



- Then I added a new active response action that drops attacker's packets if the same rule with ID $= 120100$, which I defined in the previous sub-task, is triggered



- Finally, I again restarted the Wazuh Manager daemon

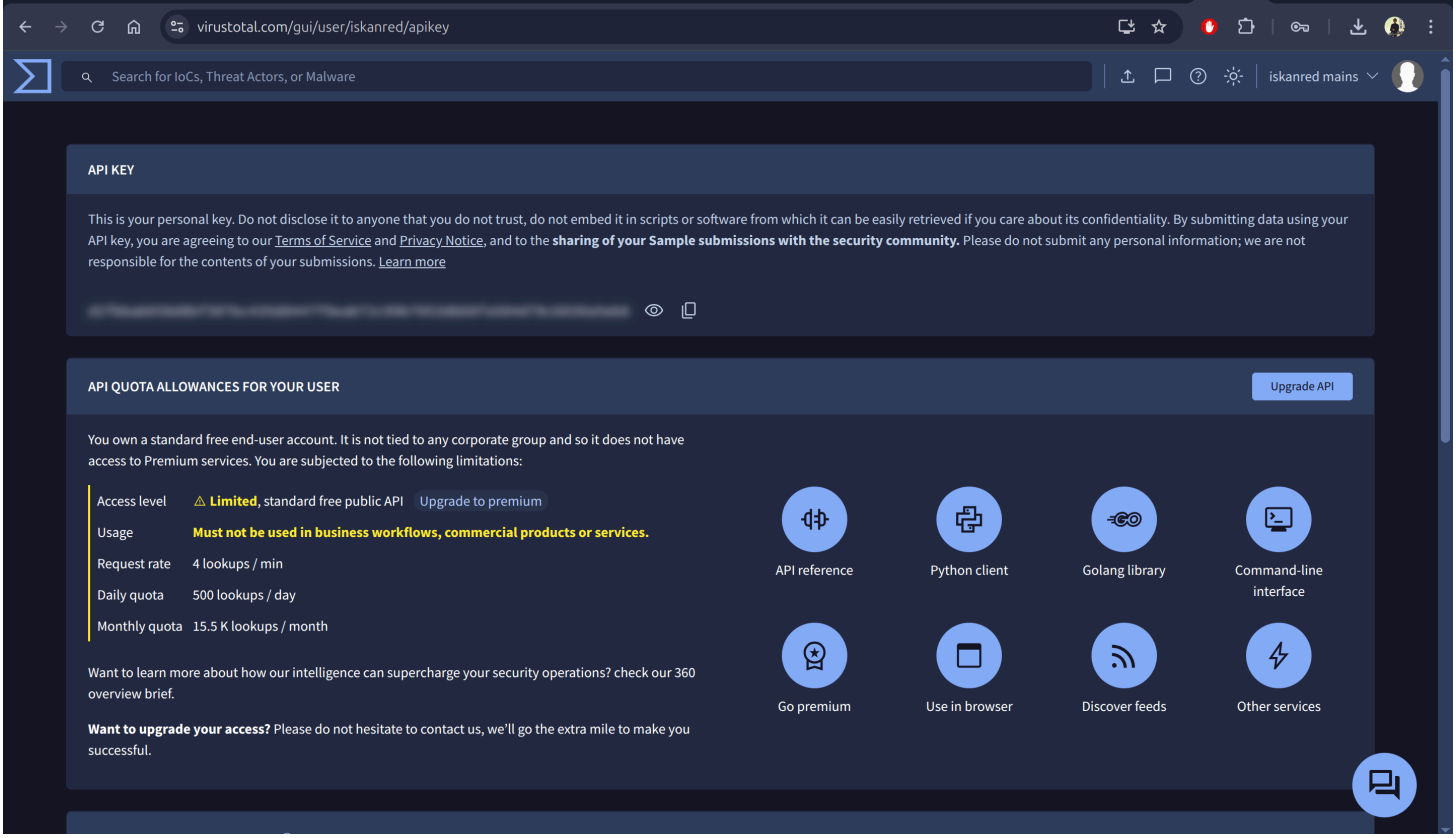


3.

Task description

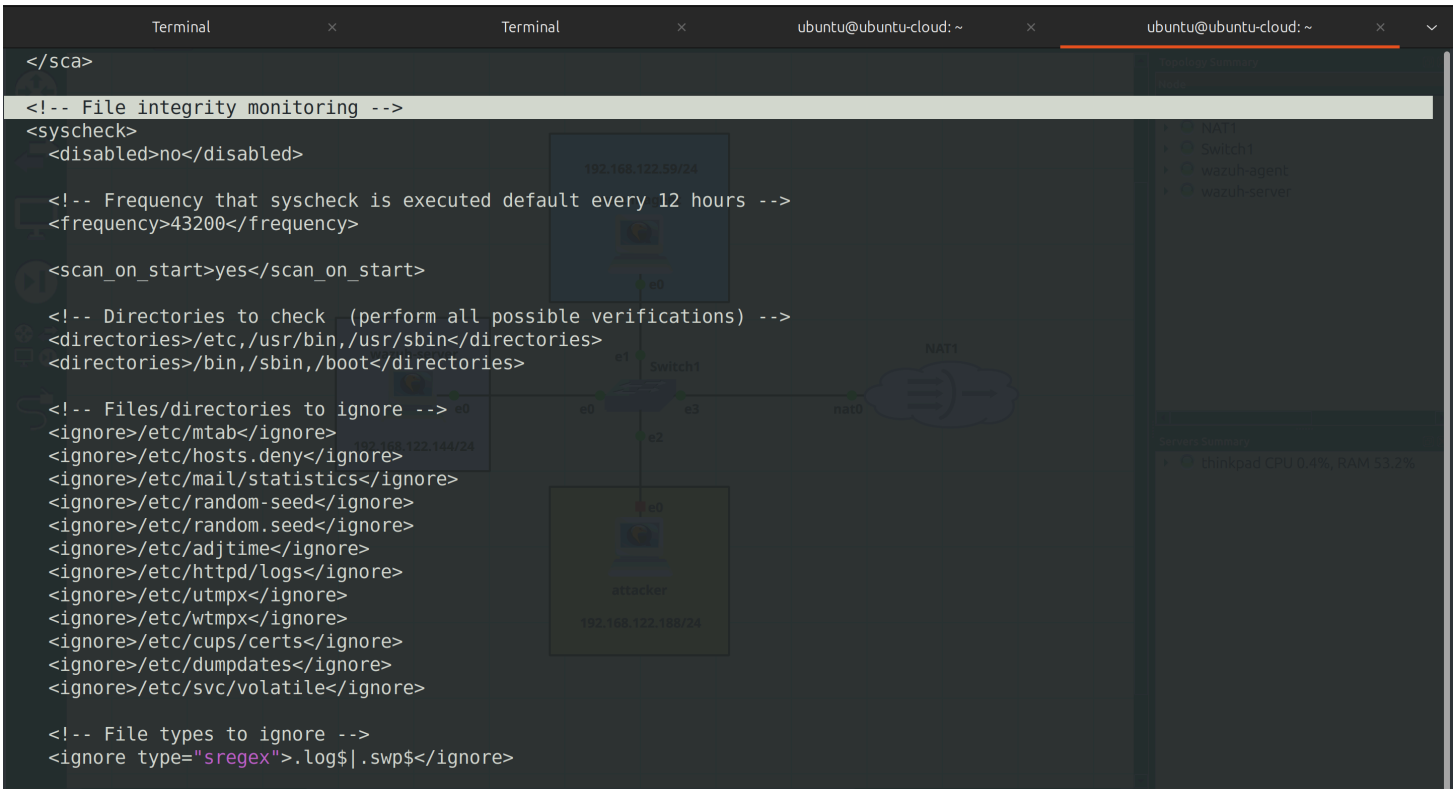
Create `/home/<USERNAME>/malwarefiles/` on the monitored endpoint. Integrate malware detection (e.g., VirusTotal or YARA), and monitor this directory for malware. Configure Wazuh to automatically delete detected malware files.

- Again, I found a useful instruction to detect and delete malware files using VirusTotal on the Wazuh official website.
- First, I signed up VirusTotal to obtain an API key in order to allow Wazuh to make requests for checking a file.

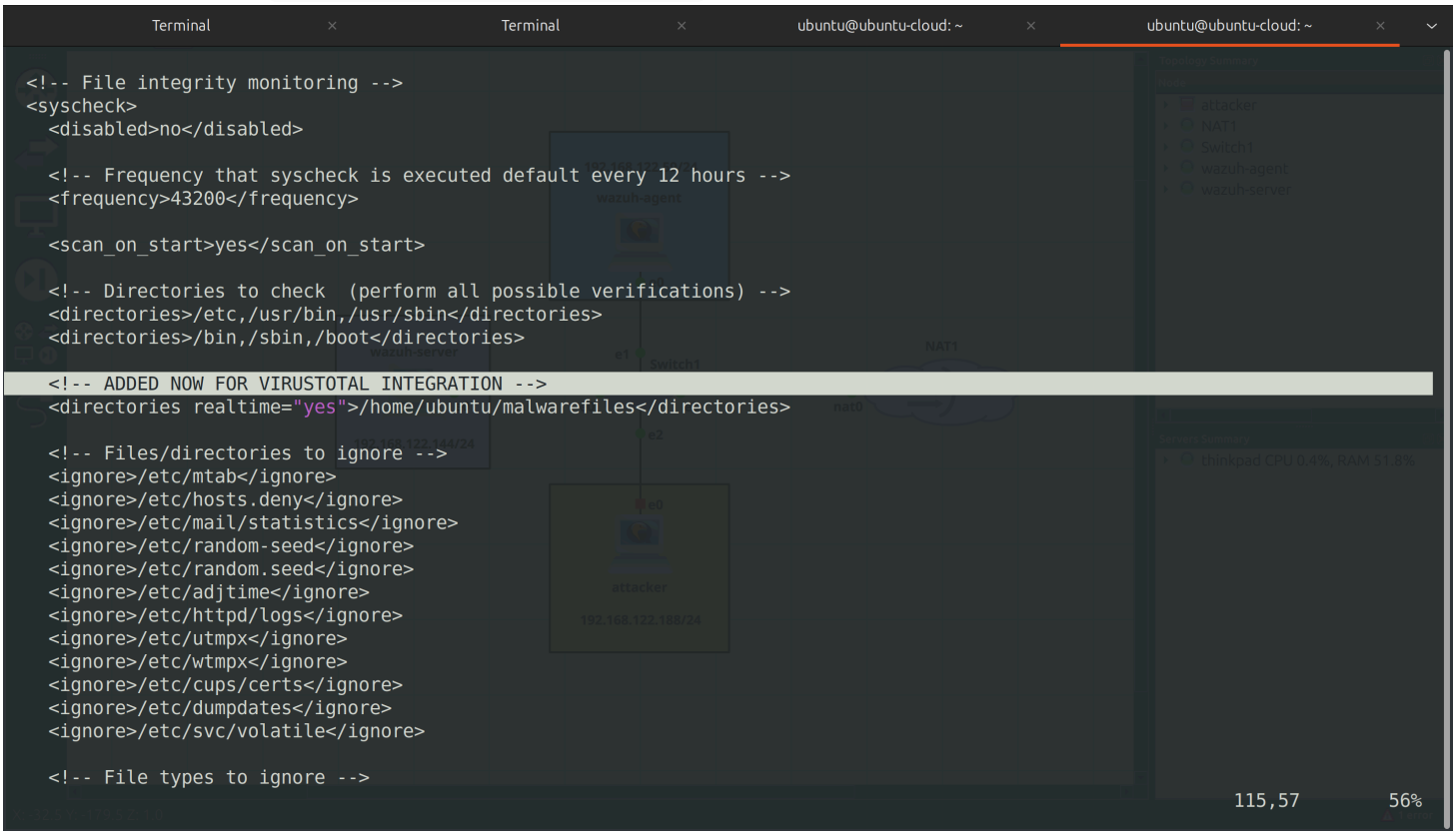


Wazuh Agent

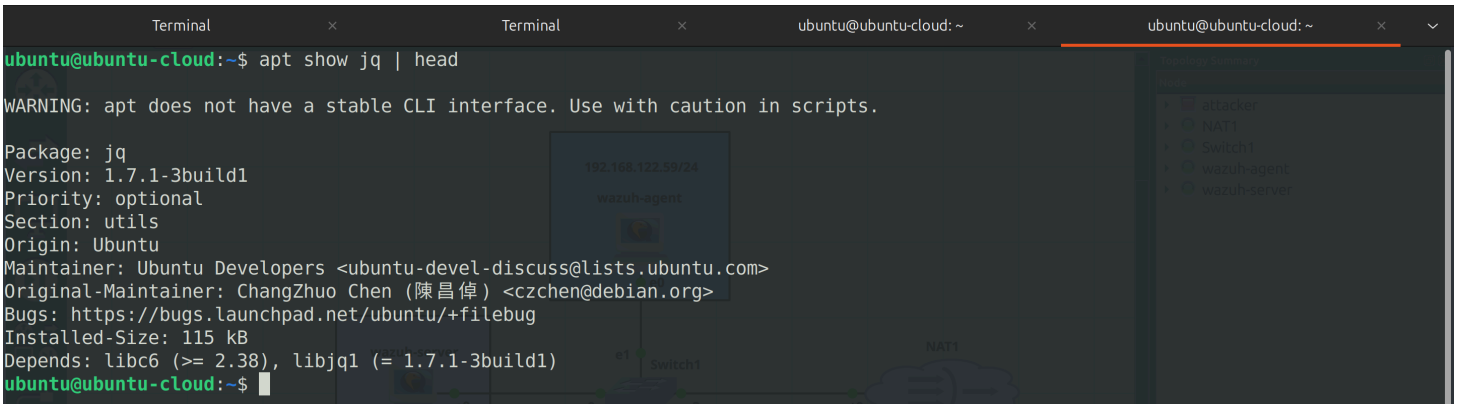
- Then I made sure that the FIM is enabled on the `wazuh-agent` inside the `/var/ossec/etc/ossec.conf` config. 1. Wazuh FIM looks for any file addition, change, or deletion on the monitored folders. This module has the hash of these files stored and triggers alerts when it detects any changes.



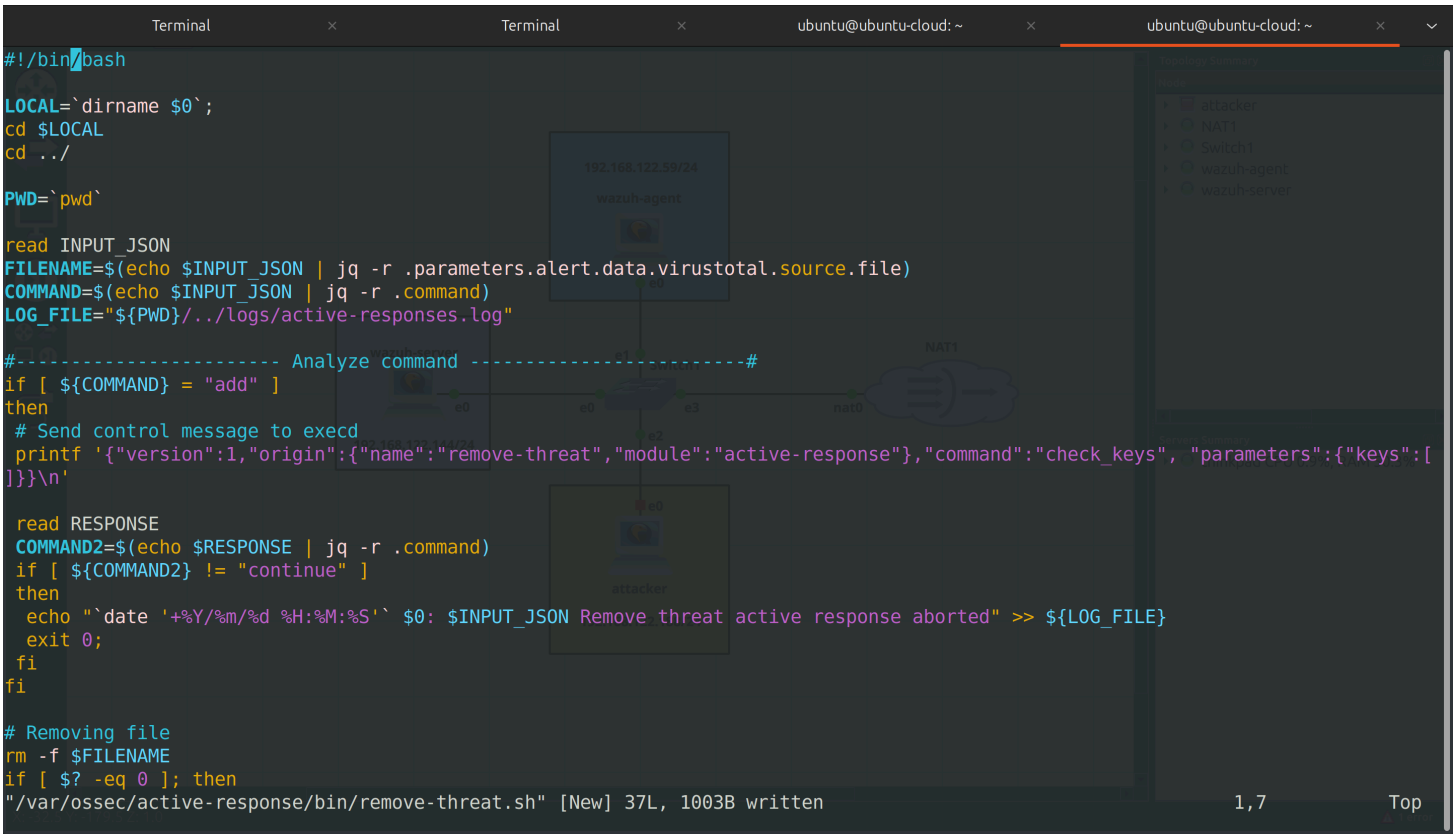
- Then I also added `/home/ubuntu/malwarefiles` to be monitored in near real-time.



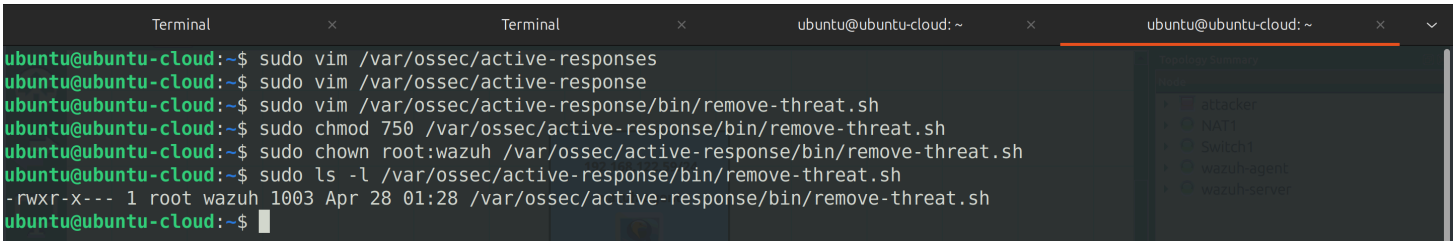
- Afterwards, I installed `jq` tool which will help us to read a log easily in order to make proper active response actions



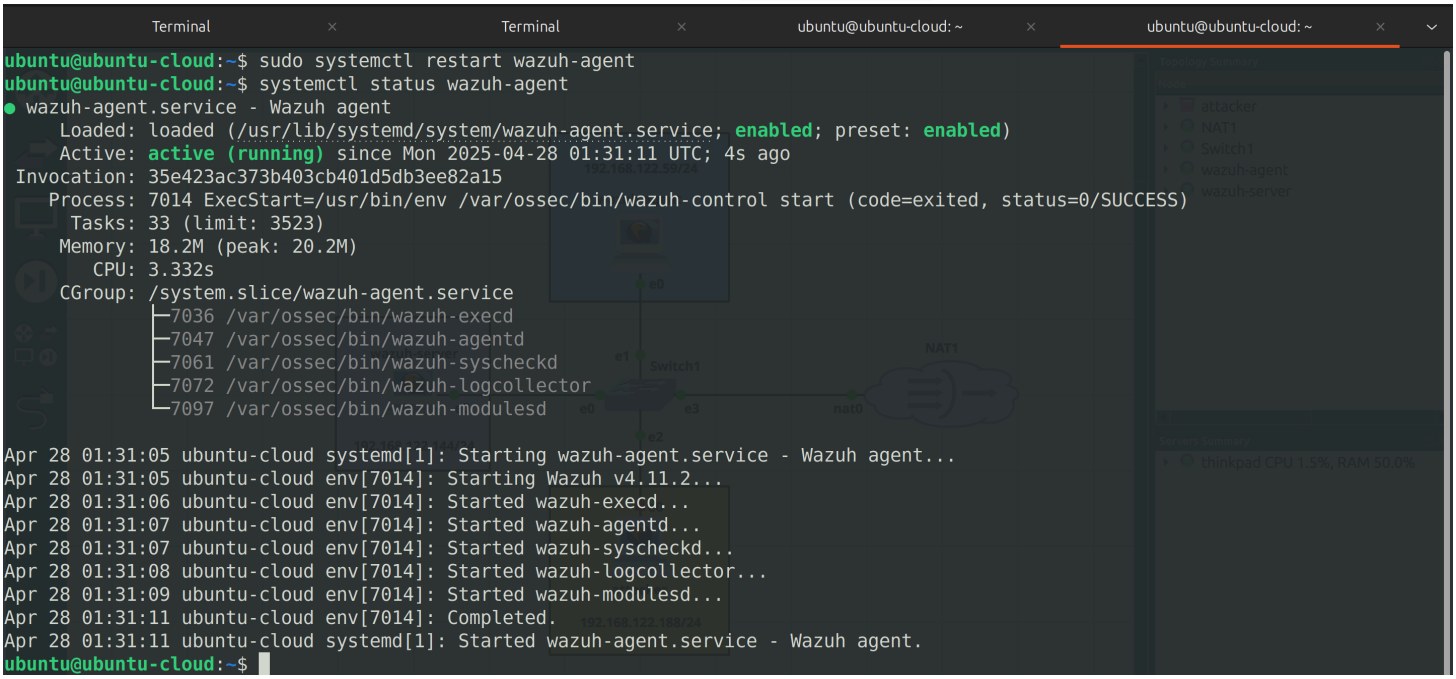
- Then I copied an active response script that removes malware file.



- Moving further, I changed the permissions and file ownership for this script in order to allow only Wazuh Agent to run it.



- Finally, I restarted the Wazuh Agent daemon on my `wazuh-agent` endpoint

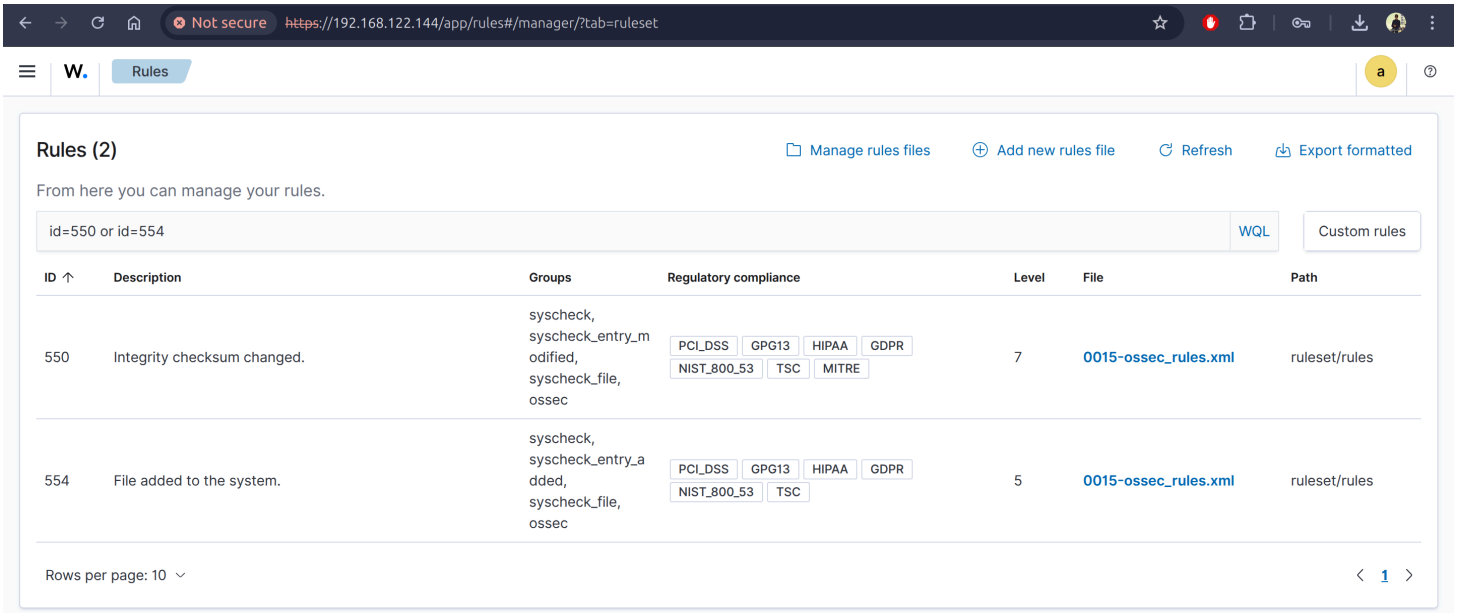


Wazuh Server

- I added two new rules that are triggered whenever a file was added or modified inside the `/home/ubuntu/malwarefiles` directory

```
Terminal  Terminal  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~

    <group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
  </rule>

</group>

<group name="pam,syslog,">

  <rule id="120100" level="10" frequency="4" timeframe="60">
    <if_matched_sid>5503</if_matched_sid>
    <description>Possible password guess on $(dstuser): 4 failed logins in a short period of time</description>
    <mitre>
      <id>T1110</id>
    </mitre>
  </rule>

</group>

<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">

    <rule id="100200" level="7">
        <if_sid>550</if_sid>
        <field name="file">/home/ubuntu/malwarefiles</field>
        <description>File modified in /home/ubuntu/malwarefiles directory.</description>
    </rule>

    <rule id="100201" level="7">
        <if_sid>554</if_sid>
        <field name="file">/home/ubuntu/malwarefiles</field>
        <description>File added to /home/ubuntu/malwarefiles directory.</description>
    </rule>

</group>
"/var/ossec/etc/rules/local_rules.xml" 47L, 1292B written                    42,28         Bot
```

- These rules are triggered when FIM basic rules are triggered



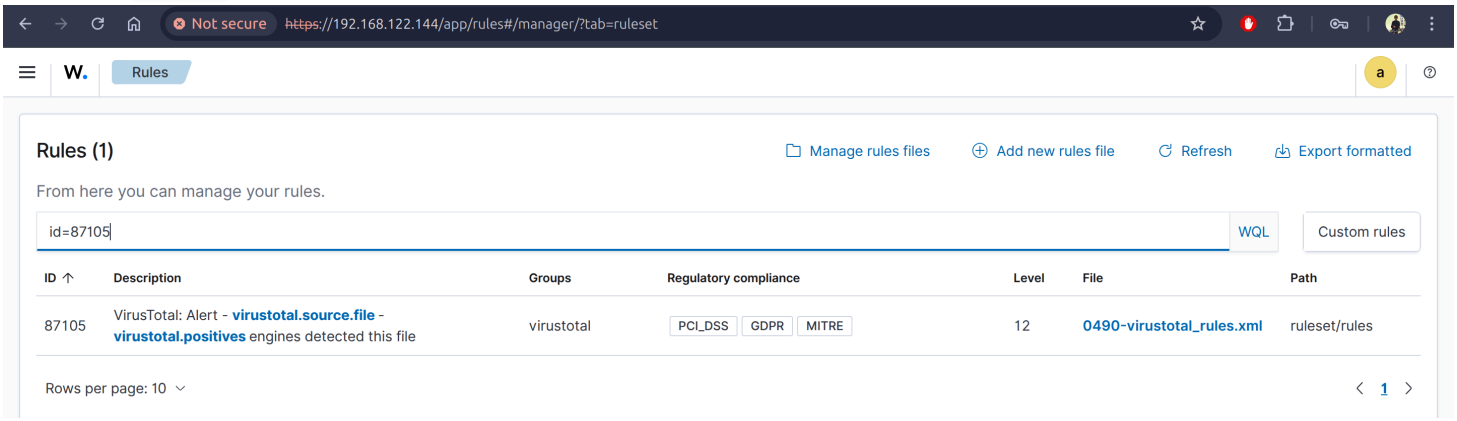
- Then I added VirusTotal integration inside the Wazuh Manager config. My API Key is masked. This allows to trigger a VirusTotal query whenever any of the rules `100200` and `100201` are triggered.

```
Terminal  Terminal  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~

</ossec_config>

<ossec_config>
  <localfile>
    <log_format>journald</log_format>
    <location>journald</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/ossec/logs/active-responses.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/dpkg.log</location>
  </localfile>

</ossec_config>

<!-- VIRUSTOTAL INTEGRATION CONFIG -->
<ossec_config>

  <integration>
    <name>virustotal</name>
    <api_key>**************************</api_key>
    <rule_id>100200,100201</rule_id>
    <alert_format>json</alert_format>
  </integration>

</ossec_config>
"/var/ossec/etc/ossec.conf" 352L, 9769B written                    338,14        Bot
```

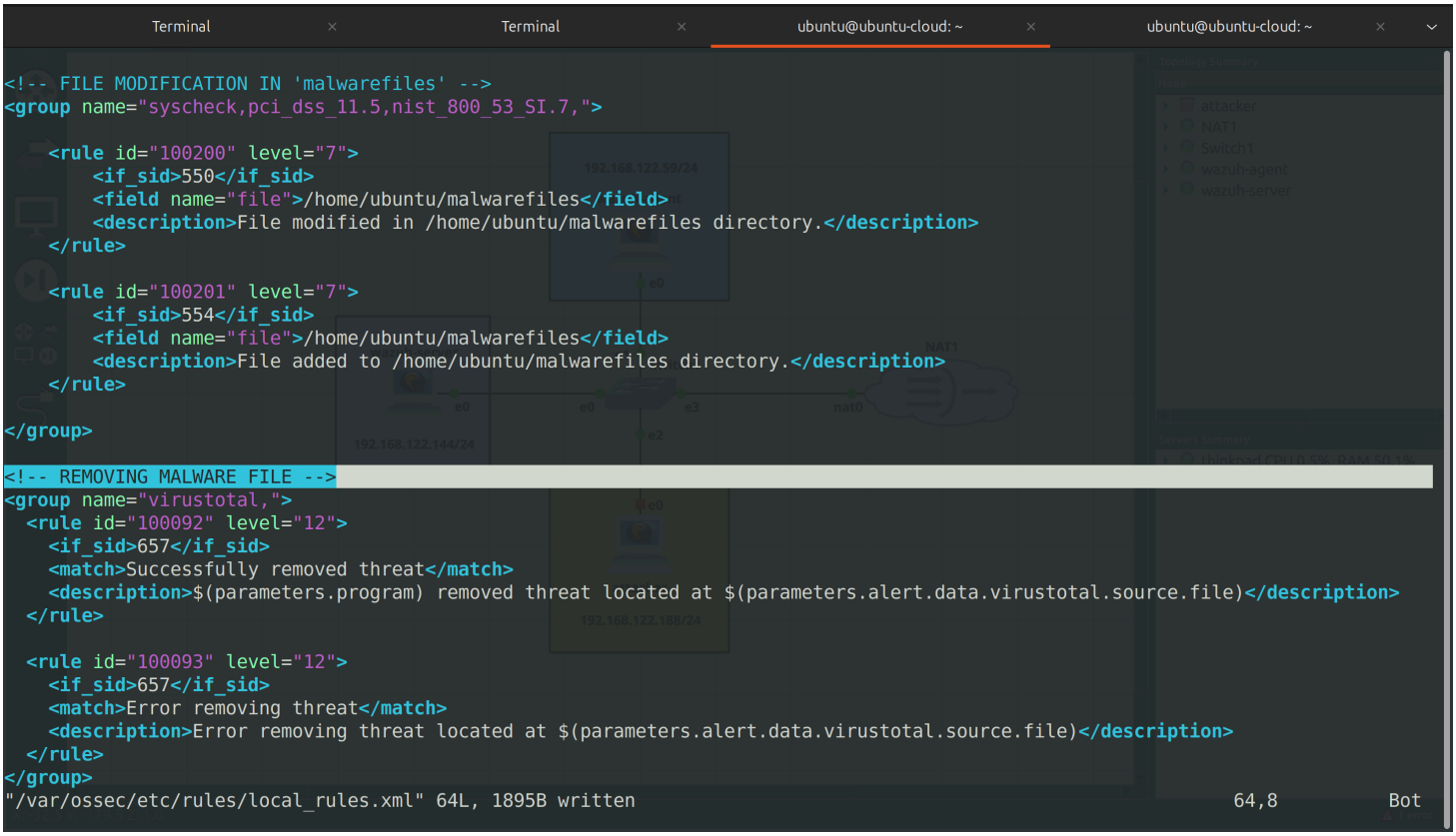
- Also, I added a `remove-threat` command that executes my `remove-threat.sh` script created above and `remove-thread` active response action that uses this command whenever a rule with ID $= 87105$ is triggered.

```
Terminal  Terminal  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/dpkg.log</location>
  </localfile>

</ossec_config>

<!-- VIRUSTOTAL INTEGRATION CONFIG -->
<ossec_config>

  <integration>
    <name>virustotal</name>
    <api_key>d2fbbab658d8bf307bc435d8447f9eab72c99b7053db68fa504d78c6830a5eb8</api_key>
    <rule_id>100200,100201</rule_id> <!-- FILE INSIDE 'malwarefiles' MODIFIED OR ADDED -->
    <alert_format>json</alert_format>
  </integration>

  <!-- ACTIVE RESPONSE -->
  <command>
    <name>remove-threat</name>
    <executable>remove-threat.sh</executable>
    <timeout_allowed>no</timeout_allowed> <!-- NO REVERT AFTER TIME -->
  </command>

  <active-response>
    <disabled>no</disabled>
    <command>remove-threat</command>
    <location>local</location>
    <rules_id>87105</rules_id> <!-- MALWARE DETECTED -->
  </active-response>

"/var/ossec/etc/ossec.conf" 366L, 10258B written                    344,0-1       99%
```

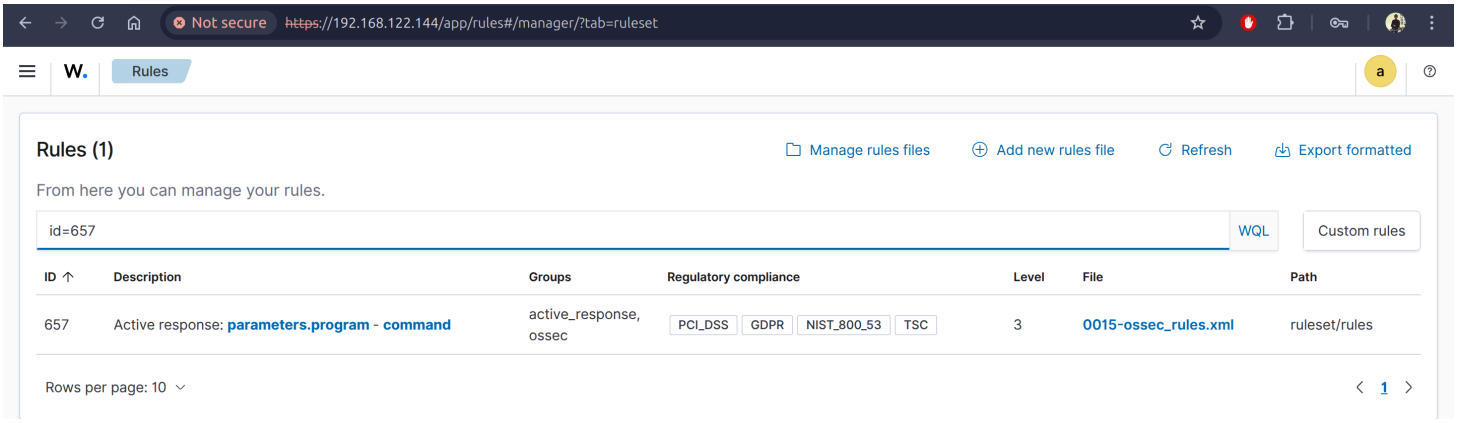
- The rule `87105` is triggered when VirusTotal detects a malware inside the file sent to it.



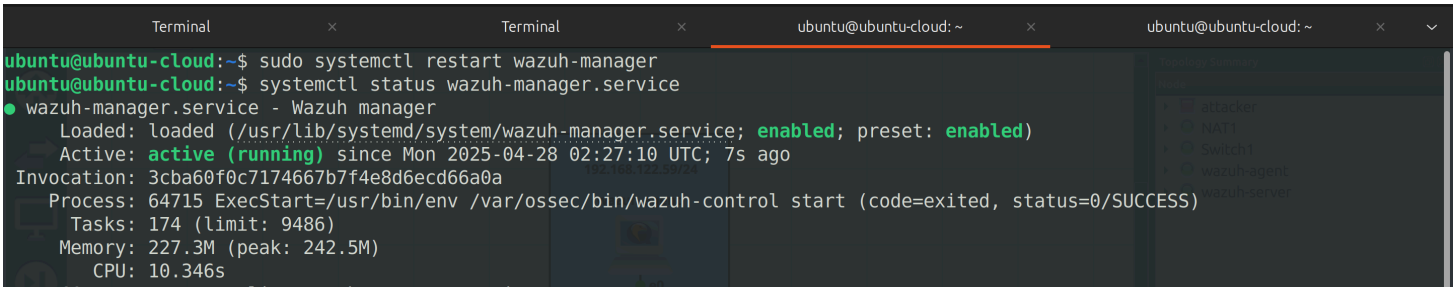
- Then I added two more rules that says if removing a malware file was successful or not



- The rule `657` is triggered whenever an active response action is done



- Finally, I restarted Wazuh Manager daemon



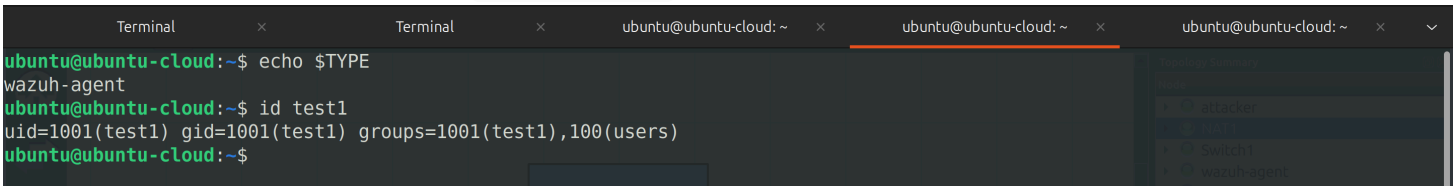
- To sum up the steps to detect, remove, and notify are the following:
  1. FIM detects a file added or modified inside the `/home/ubuntu/malwarefiles`
  2. The rules about file addition modification is triggered
  3. Modified or added files are sent to the VirusTotal
  4. After VirusTotal returns the result Wazuh checks if files are malware
  5. If so, the active response is started: the files are removed.
  6. If removal is successful, the rule is triggered
  7. If not, another rule is triggered

Task 3 - Simulate attacks

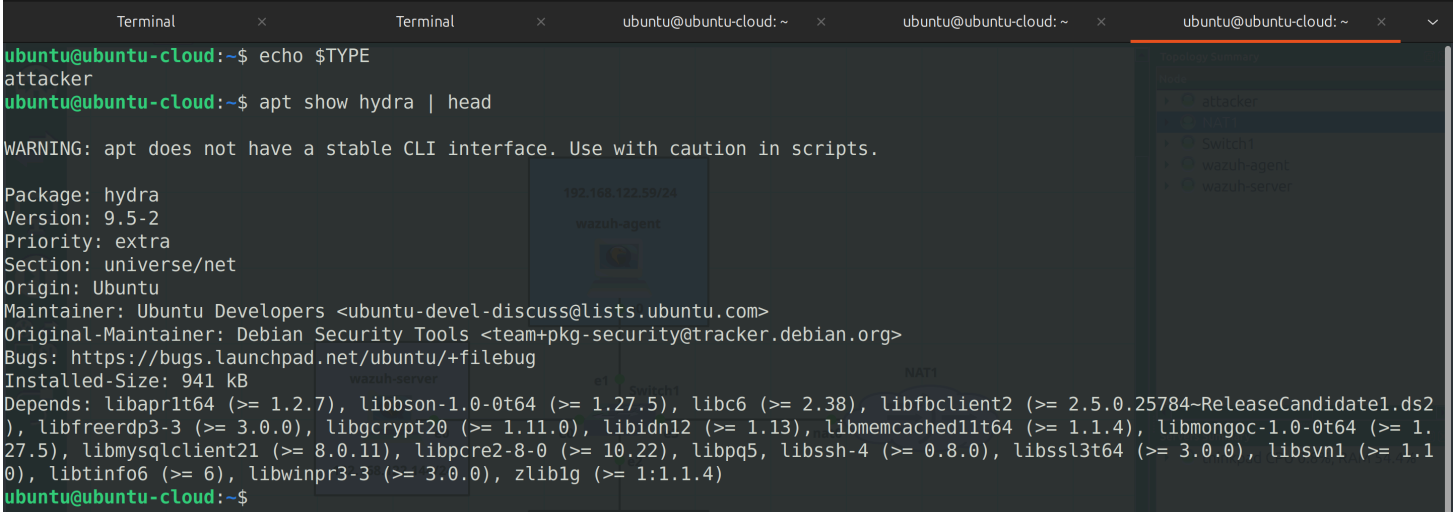
1.

> **Task description**
>
> Launch an SSH brute force attack from the attacker endpoint against the test user

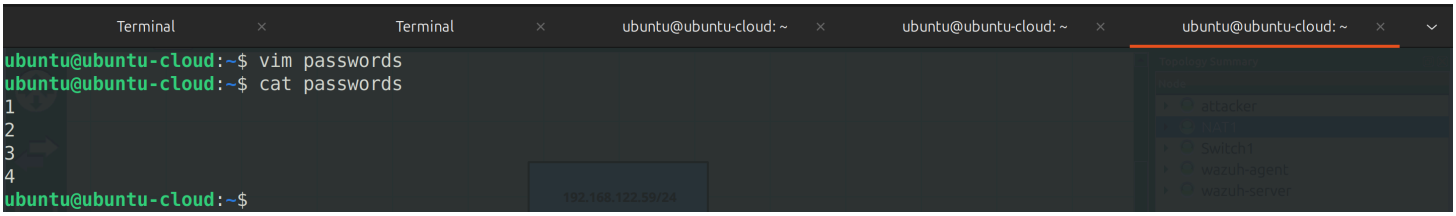
- First, I created a test user on the `wazuh-agent`



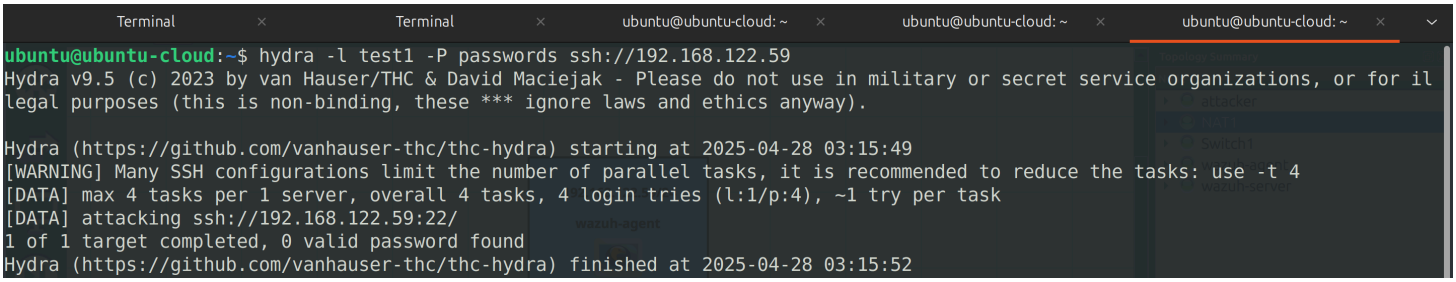
- Also, I installed hydra tool on the `attacker` machine. This tool allows to make brute force attacks easily.



- I created 4 password to make 4 login tries through SSH



- I started a brute force attack which was completed quickly with exactly 4 login tries

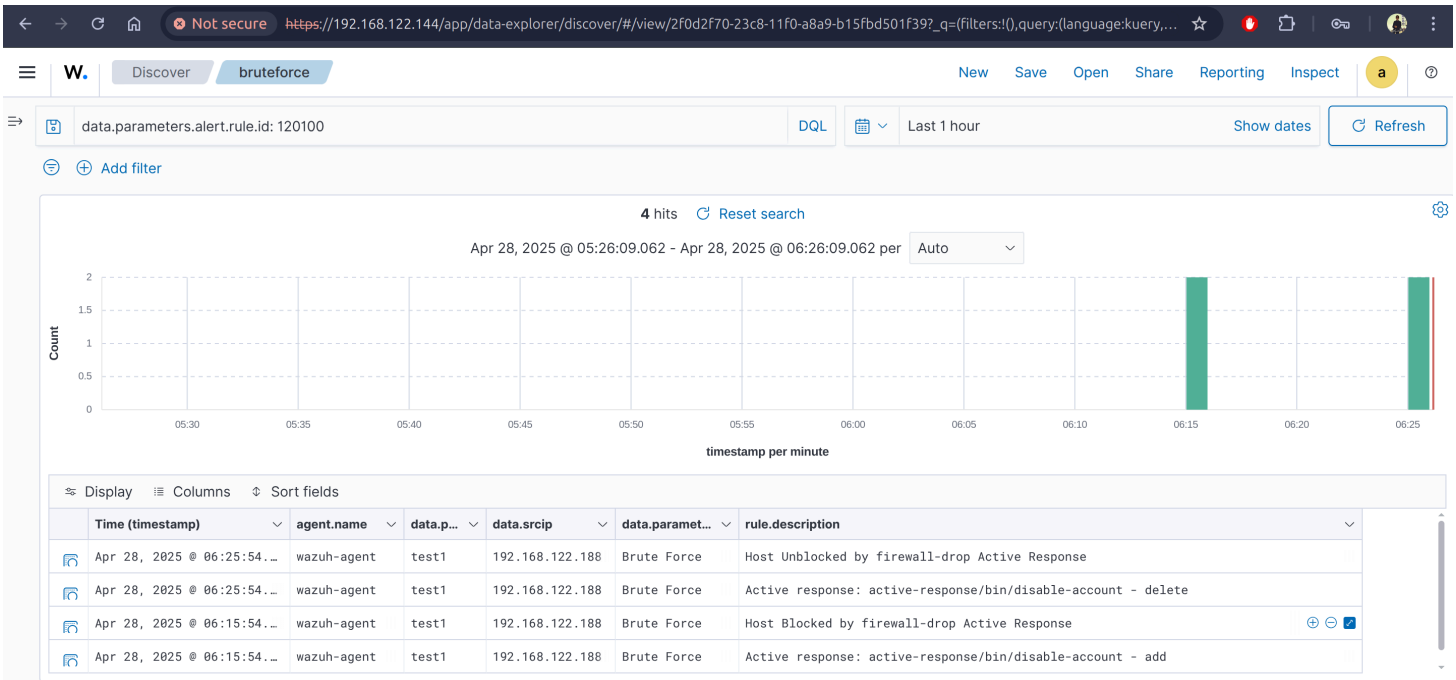


2.

Task description

Show relevant alerts on the Wazuh dashboard.

- Below we see that the alerts actually appeared on the dashboard



- From them we can see that the MIT&RE attack type was successfully recognized (Brute Force). We see the attacker's IP address and rules description.
  - `Host Blocked`
  - `disable-account - add`
- Also, we see alerts that also notifies us about disabling applied active response actions after exactly 10 minutes
  - `Host Unblocked`
  - `disable account - delete`

3.

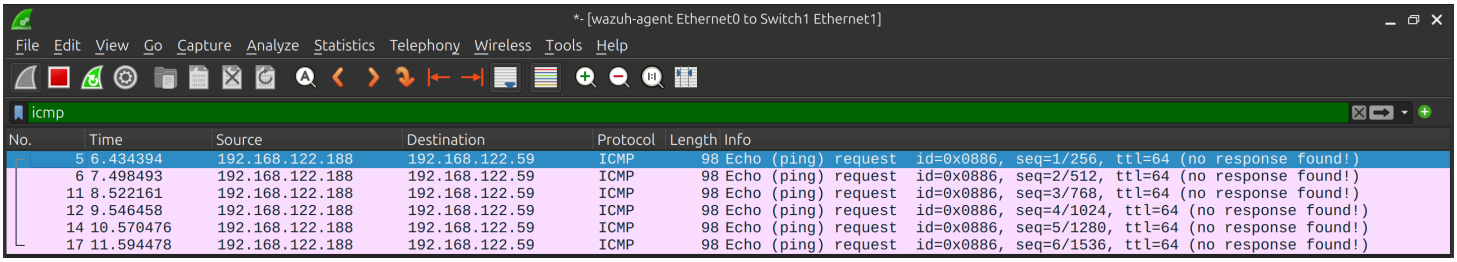
Task description

Provide evidence that the user account was disabled and the attacker IP blocked. Include dashboard alerts and screenshots from the endpoint involved.

- We see that the attacker's **IP address was blocked** since an attacker cannot successfully ping the `wazuh-agent` endpoint

```
Terminal  Terminal  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~
ubuntu@ubuntu-cloud:~$ hydra -l test1 -P passwords ssh://192.168.122.59
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for il
legal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-28 03:15:49
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 4 tasks per 1 server, overall 4 tasks, 4 login tries (l:1/p:4), ~1 try per task
[DATA] attacking ssh://192.168.122.59:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-04-28 03:15:52
ubuntu@ubuntu-cloud:~$ ping 192.168.122.59
PING 192.168.122.59 (192.168.122.59) 56(84) bytes of data.
^C
--- 192.168.122.59 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2062ms

ubuntu@ubuntu-cloud:~$
```

- Meanwhile, ping requests were sent successfully but with no reply



- Also, we see that the `test1` **user account was disabled**

```
Terminal  Terminal  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~
ubuntu@ubuntu-cloud:~$ sudo passwd --status test1
test1 L 2025-04-27 0 99999 7 -1
ubuntu@ubuntu-cloud:~$
```

```
-S, --status
    Display account status information. The status information
    consists of 7 fields. The first field is the user's login
    name. The second field indicates if the user account has a
    locked password (L), has no password (NP), or has a usable
    password (P). The third field gives the date of the last
    password change. The next four fields are the minimum age,
    maximum age, warning period, and inactivity period for the
    password. These ages are expressed in days.
```

- And after 10 minutes everything works again

```
Terminal  Terminal  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~  ubuntu@ubuntu-cloud: ~
ubuntu@ubuntu-cloud:~$ hydra -l test1 -P passwords ssh://192.168.122.59
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for il
legal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-28 03:15:49
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 4 tasks per 1 server, overall 4 tasks, 4 login tries (l:1/p:4), ~1 try per task
[DATA] attacking ssh://192.168.122.59:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-04-28 03:15:52
ubuntu@ubuntu-cloud:~$ ping 192.168.122.59
PING 192.168.122.59 (192.168.122.59) 56(84) bytes of data.
^C
--- 192.168.122.59 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2062ms

ubuntu@ubuntu-cloud:~$ ping 192.168.122.59
PING 192.168.122.59 (192.168.122.59) 56(84) bytes of data.
^C
--- 192.168.122.59 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5160ms

ubuntu@ubuntu-cloud:~$ ping 192.168.122.59
PING 192.168.122.59 (192.168.122.59) 56(84) bytes of data.
64 bytes from 192.168.122.59: icmp_seq=1 ttl=64 time=1.29 ms
64 bytes from 192.168.122.59: icmp_seq=2 ttl=64 time=1.15 ms
^C
--- 192.168.122.59 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.151/1.220/1.290/0.069 ms
ubuntu@ubuntu-cloud:~$
```

```
Terminal × Terminal × ubuntu@ubuntu-cloud: ~ × ubuntu@ubuntu-cloud: ~ × ubuntu@ubuntu-cloud: ~ ×
ubuntu@ubuntu-cloud:~$ sudo passwd --status test1
test1 P 2025-04-27 0 99999 7 -1
ubuntu@ubuntu-cloud:~$
```

4.

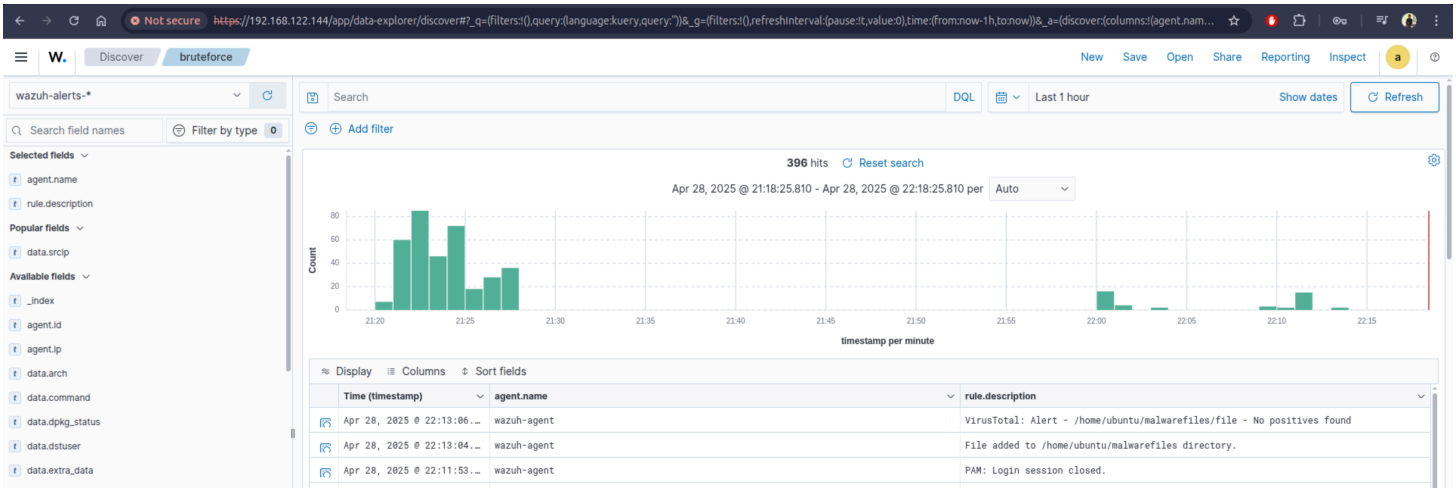
Task description

Download a malware sample into the `/home/<USERNAME>/malwarefiles/` directory. Confirm detection and show that it was automatically removed.

- First, let me check if our rules work correctly
- I added a new file to the `/home/ubuntu/malwarefiels/` directory which is a safe empty file

```
Terminal × Terminal × ubuntu@ubuntu-cloud: ~ × ubuntu@ubuntu-cloud: ~/malwarefiles ×
ubuntu@ubuntu-cloud:~/malwarefiles$ touch file
ubuntu@ubuntu-cloud:~/malwarefiles$
```

- And we can instantly see the alerts fired on Wazuh Dashboard



- We see that this file was not malware. Hence, it was not removed

```
Terminal × Terminal × ubuntu@ubuntu-cloud: ~ × ubuntu@ubuntu-cloud: ~/malwarefiles ×
ubuntu@ubuntu-cloud:~/malwarefiles$ ls -l
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 Apr 28 19:13 file
```

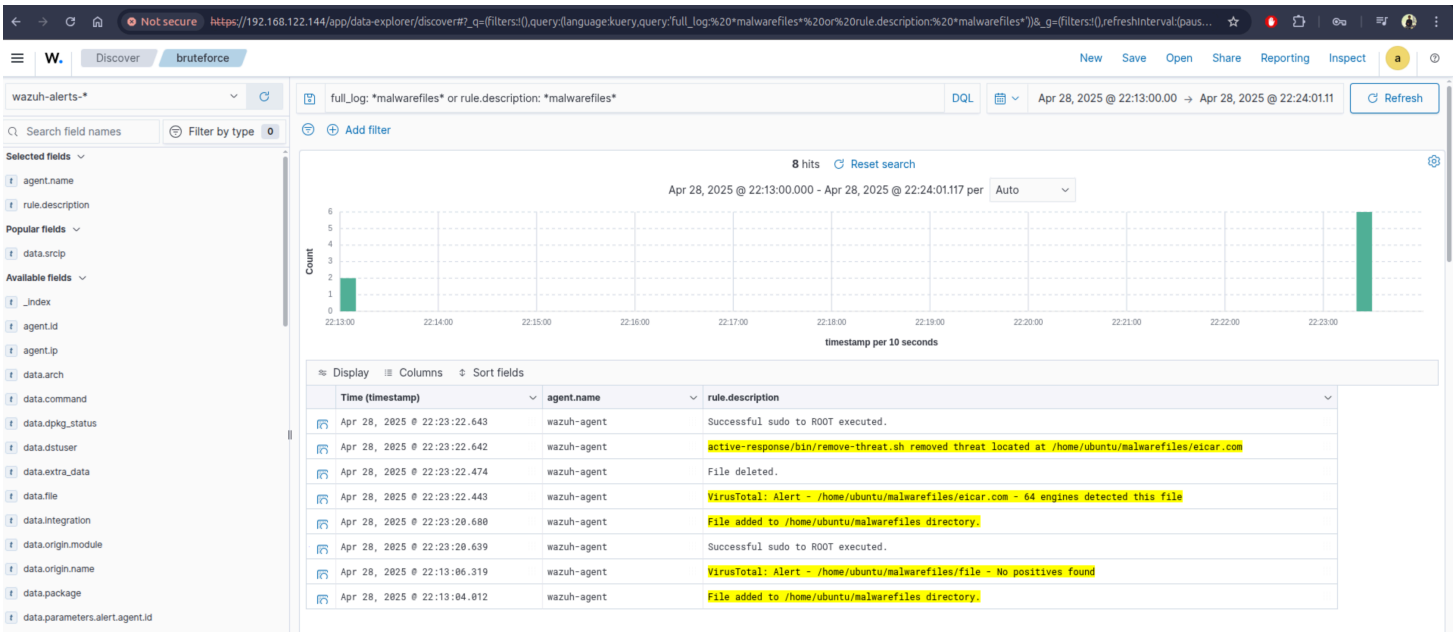
- Now let's check a malware file
- I downloaded a file that is used to check if antivirus software works: eicar.com

```
Terminal × Terminal × ubuntu@ubuntu-cloud: ~ × ubuntu@ubuntu-cloud: ~/malwarefiles ×
ubuntu@ubuntu-cloud:~/malwarefiles$ sudo curl -Lo ./eicar.com https://secure.eicar.org/eicar.com && sudo ls -lah ./eicar.com
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    68  100    68    0     0     81      0 --:--:-- --:--:-- --:--:--    81
-rw-r--r-- 1 root root 68 Apr 28 19:23 ./eicar.com
ubuntu@ubuntu-cloud:~/malwarefiles$
```

- As we see the file was actually created
- However, it had been quickly deleted

```
Terminal × Terminal × ubuntu@ubuntu-cloud: ~ × ubuntu@ubuntu-cloud: ~/malwarefiles ×
ubuntu@ubuntu-cloud:~/malwarefiles$ ls -l
total 0
-rw-rw-r-- 1 ubuntu ubuntu 0 Apr 28 19:13 file
ubuntu@ubuntu-cloud:~/malwarefiles$
```

- And on the dashboard we can see that the files was actually sent to VirusTotal and deleted after



Task 4 - Log management

1.

Task description

Define what a log retention policy is.

- A **log retention policy** is a defined set of guidelines that specifies how long log records should be retained, stored, and eventually deleted. This policy typically considers factors such as:
  - Compliance requirements: Meeting legal or regulatory mandates concerning data retention.
  - Storage limitations: Managing disk space and storage costs by determining how long logs should be kept.
  - Operational needs: Establishing a timeframe for which logs are useful for troubleshooting, auditing, or performance monitoring.
- A well-defined log retention policy helps organizations balance the need for retaining logs for analysis and compliance against the costs and risks associated with storing large volumes of data.

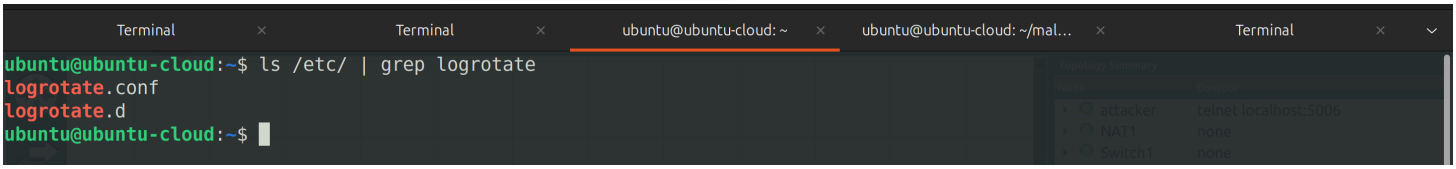
2.

Task description

Explain how logs are rotated on Linux and how disk space is managed in relation to logs.

How logs are rotated

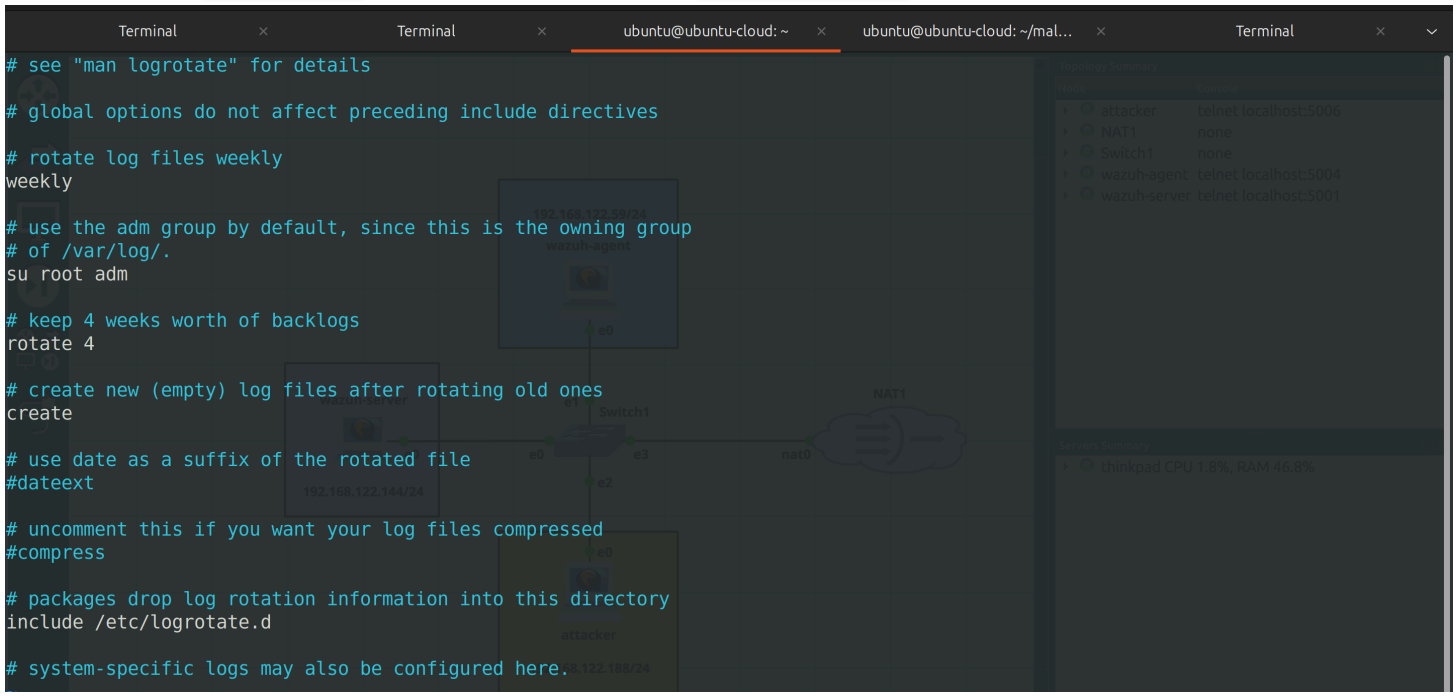
- **Log rotation** is the process of renaming, archiving, and managing log files to prevent them from consuming excessive disk space while maintaining a manageable history of log entries for analysis and auditing. In Linux, log rotation is typically handled by tools and services such as **`logrotate`**. It is configured through a set of configuration files (commonly located in `/etc/logrotate.conf` and individual configuration files in `/etc/logrotate.d/`).



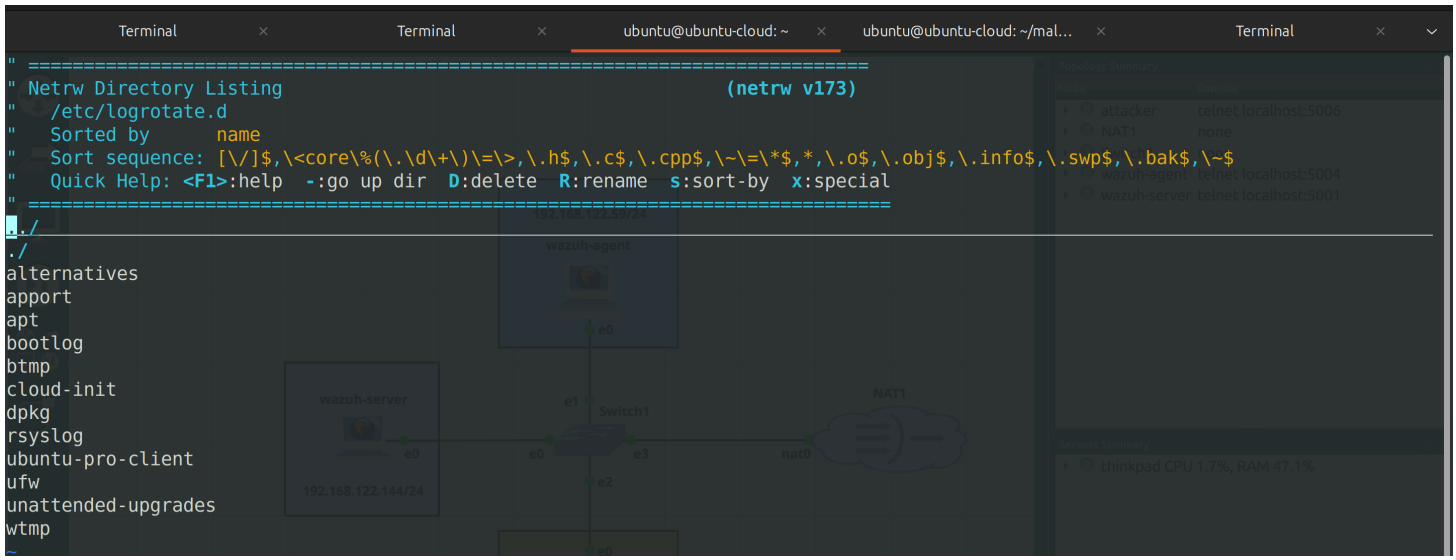
- `logrotate` allows customization of rotation policies, including:
  - **Frequency**: Logs can be rotated daily, weekly, monthly, or based on size.
  - **Retention**: Determine how many old log files to keep (e.g., keep the last 4 rotated logs).
  - **Compression**: Old logs can be compressed (e.g., using gzip) to save disk space.
  - **Post-rotation actions**: Commands can be specified to run after rotation, such as restarting services (e.g., systemd services or rsyslog).
- Example of a `logrotate` configuration:

```
  /var/log/example.log {
    daily                  # Rotate daily
    missingok              # Don't throw an error if the log file is missing
    rotate 7               # Keep the last 7 logs
    compress               # Compress old logs
      delaycompress          # Delay compression until the next rotation
      notifempty             # Do not rotate the log if it is empty
      create 0640 root adm   # Create a new log file with given permissions and
ownership
      sharedscripts          # Run post-rotate scripts only once for multiple logs
      postrotate
        systemctl reload example.service
    endscript
  }
```

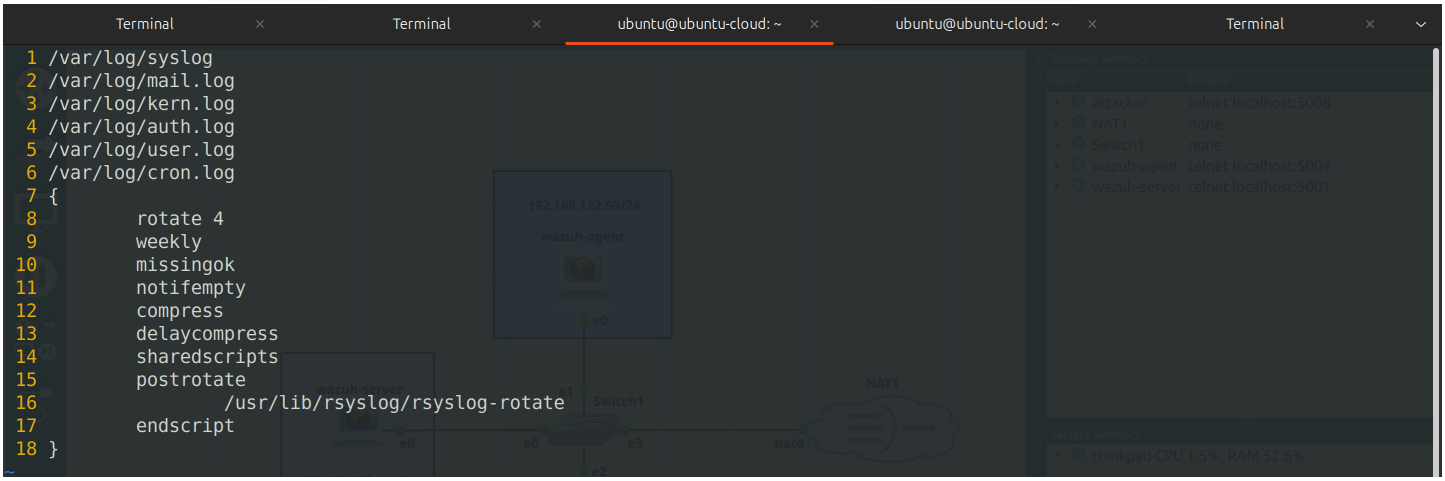
- Example of a `logrotate` configuration from the `wazuh-server` machine



- Program-specific configs



- Example of a `rsyslog` logrotate config which I found on the `wazuh-server`



How disk space is managed in relation to logs

- **Space Consumption**. Logs can grow rapidly, depending on the amount of data generated by applications and services. Without proper management, logs can fill up disk space, leading to system performance issues and potential service interruptions.
- **Preventing Disk Full Conditions**. Using log rotation helps prevent disk full conditions by periodically cleaning up old log files. By specifying retention periods and ensuring that logs are compressed, organizations can effectively manage disk space usage.
- **Monitoring Disk Usage**. It's important to monitor disk usage statistics regularly to ensure that log rotation is effectively reducing space usage. Tools like `du`, `df`, and logging monitoring frameworks can help track how much space is consumed by logs.
- **Archiving and Offloading**. In some cases, organizations may need to archive old logs for future reference or compliance. This can involve transferring old log files to secondary storage or a centralized log management solution, such as using services like ELK stack, Splunk, or SageDB.

3.

Task description

Create a configuration to automatically delete Wazuh alert log files older than 90 days

- I have read that actually "By default, the Wazuh server retains logs and does not delete them automatically. However, you can choose when to manually or automatically delete these logs according to your legal and regulatory requirements." So that's why we actually may need to configure log retention!
- I explored that alert log files are stored on Wazuh Manager or `wazuh-server` in my case inside the `/var/ossec/logs/alerts/` directory

- Both `alerts.log` and `alerts.json` keep the same logs in a different format, while `2025` directory stores only compressed log files and their checksum that are signed daily

```
root@ubuntu-cloud:/var/ossec/logs/alerts/2025/Apr# ls -al
total 816
drwxr-x--- 2 wazuh wazuh   4096 Apr 28 21:17 .
drwxr-x--- 3 wazuh wazuh   4096 Apr 28 21:17 ..
-rw-r----- 1 wazuh wazuh   7820 Apr 28 00:00 ossec-alerts-27.json.gz
-rw-r----- 1 wazuh wazuh    499 Apr 28 00:00 ossec-alerts-27.json.sum
-rw-r----- 1 wazuh wazuh   5610 Apr 28 00:00 ossec-alerts-27.log.gz
-rw-r----- 1 wazuh wazuh    496 Apr 28 00:00 ossec-alerts-27.log.sum
-rw-r----- 2 wazuh wazuh 482305 Apr 28 21:32 ossec-alerts-28.json
-rw-r----- 2 wazuh wazuh 308128 Apr 28 21:32 ossec-alerts-28.log
root@ubuntu-cloud:/var/ossec/logs/alerts/2025/Apr#
```

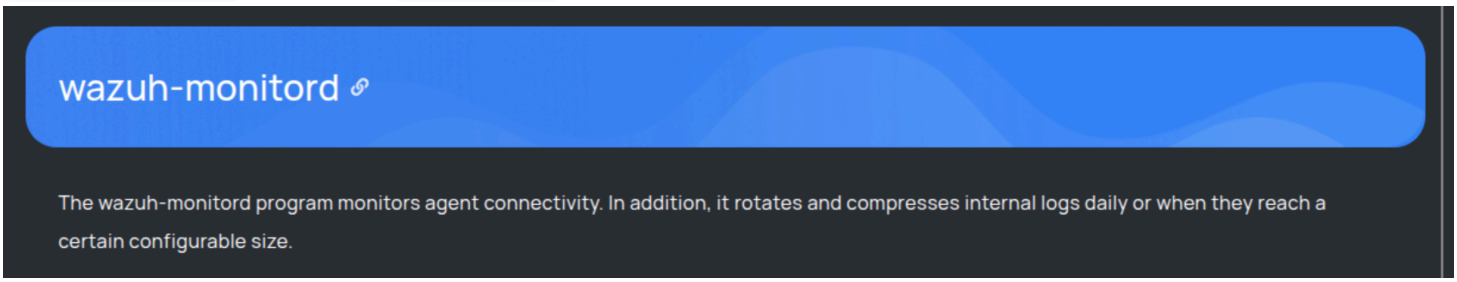
- As we see both files have no information about `Apr 27` logs

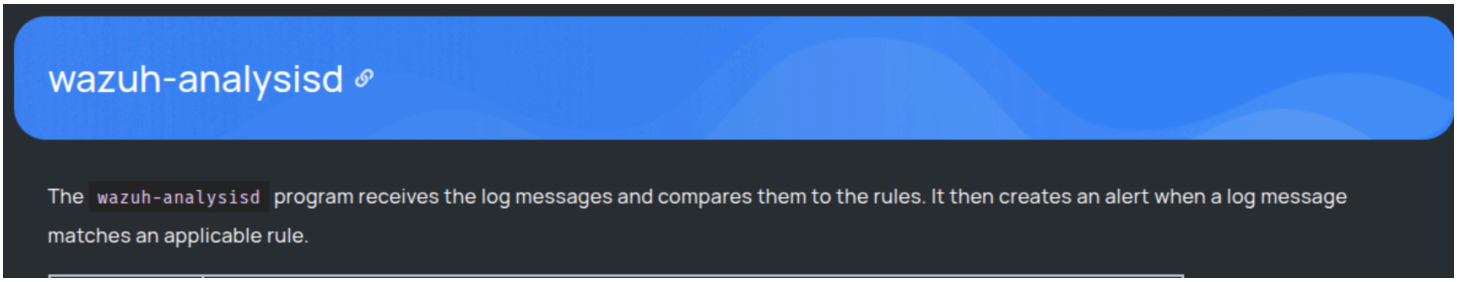
```
ubuntu@ubuntu-cloud:~$ sudo grep "Apr 27 " /var/ossec/logs/alerts/alerts.json; sudo grep "Apr 27 " /var/ossec/logs/alerts/alerts.
log
ubuntu@ubuntu-cloud:~$
```

- While for `Apr 28` (today) there are lots of log records

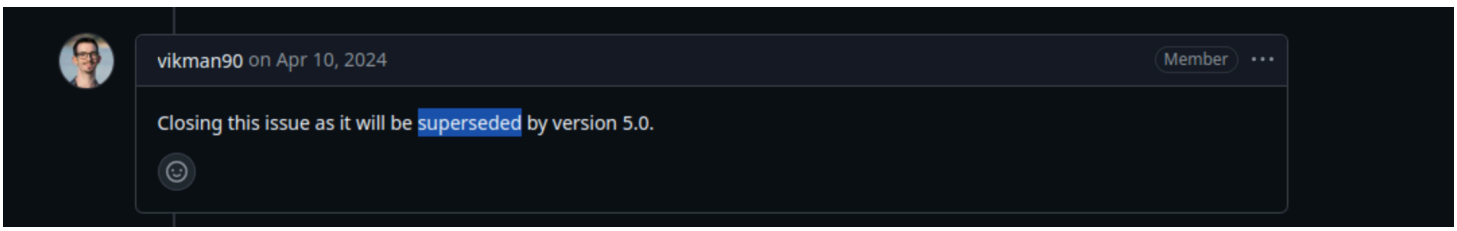
```
ubuntu@ubuntu-cloud:~$ sudo grep "Apr 28 " /var/ossec/logs/alerts/alerts.json | head -1
{"timestamp":"2025-04-28T00:05:46.449+0000","rule":{"level":3,"description":"Successful sudo to ROOT executed.","id":"5402","mitr
e":{"id":["T1548.003"],"tactic":["Privilege Escalation","Defense Evasion"],"technique":["Sudo and Sudo Caching"]},"firedtimes":1,
"mail":false,"groups":["syslog","sudo"],"pci_dss":["10.2.5","10.2.2"],"gpg13":["7.6","7.8","7.13"],"gdpr":["IV_32.2"],"hipaa":["1
64.312.b"],"nist_800_53":["AU.14","AC.7","AC.6"],"tsc":["CC6.8","CC7.2","CC7.3"]},"agent":{"id":"000","name":"ubuntu-cloud"},"man
ager":{"name":"ubuntu-cloud"},"id":"1745798746.0","full_log":"Apr 28 00:05:45 ubuntu-cloud sudo[60001]:  ubuntu : TTY=pts/0 ; PW
D=/home/ubuntu ; USER=root ; COMMAND=/usr/bin/vim /var/ossec/etc/ossec.conf","predecoder":{"program_name":"sudo","timestamp":"Apr
 28 00:05:45","hostname":"ubuntu-cloud"},"decoder":{"parent":"sudo","name":"sudo","ftscomment":"First time user executed the sudo
 command"},"data":{"srcuser":"ubuntu","dstuser":"root","tty":"pts/0","pwd":"/home/ubuntu","command":"/usr/bin/vim /var/ossec/etc/
ossec.conf"},"location":"journald"}
ubuntu@ubuntu-cloud:~$ sudo grep "Apr 28 " /var/ossec/logs/alerts/alerts.log | head -1
2025 Apr 28 00:05:46 ubuntu-cloud->journald
ubuntu@ubuntu-cloud:~$
```

- From all this I inferred that Wazuh Server already rotates logs daily! Then I found proofs of this inference. However, as far as I got this works only together **`monitord`** component which is responsible for "product" logs (`ossec.log` , `logs/wazuh` , not "event" (`logs/alerts` , `logs/archives` , etc.). But **`analysisd`** component is responsible for event logs.





- After I realized it, I tried to find some setting for Wazuh Server configuration to enable log retention for event logs. However, after a small research I found an official issue that states that log rotation mechanism for event logs will become configurable only in version 5.0 which is not released yet 😞



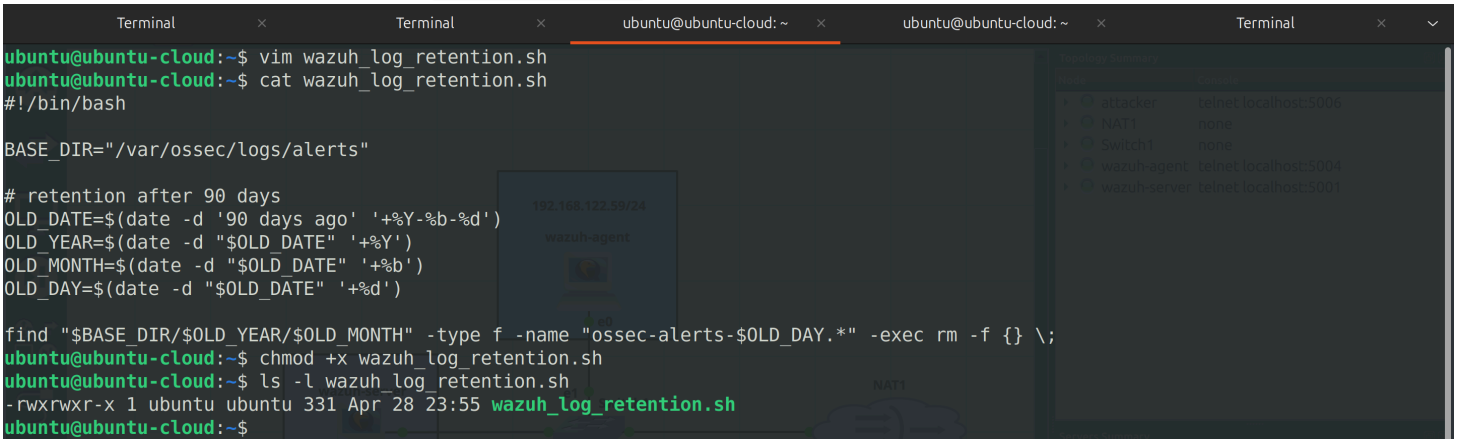
- Also, I haven't found a possibility to disable default log rotation policy. There were some closed issues only. Therefore, I cannot inject my own log rotation and retention mechanism such as **logrotate**
- Hence, the only option to make log retention I had is to write my own script making it a cron job.
- Below is the my script

```
#!/bin/bash

BASE_DIR="/var/ossec/logs/alerts"

# retention after 90 days
OLD_DATE=$(date -d '90 days ago' '+%Y-%b-%d')
OLD_YEAR=$(date -d "$OLD_DATE" '+%Y')
OLD_MONTH=$(date -d "$OLD_DATE" '+%b')
OLD_DAY=$(date -d "$OLD_DATE" '+%d')

find "$BASE_DIR/$OLD_YEAR/$OLD_MONTH" -type f -name "ossec-alerts-$OLD_DAY.*" -
exec rm -f {} \;
```

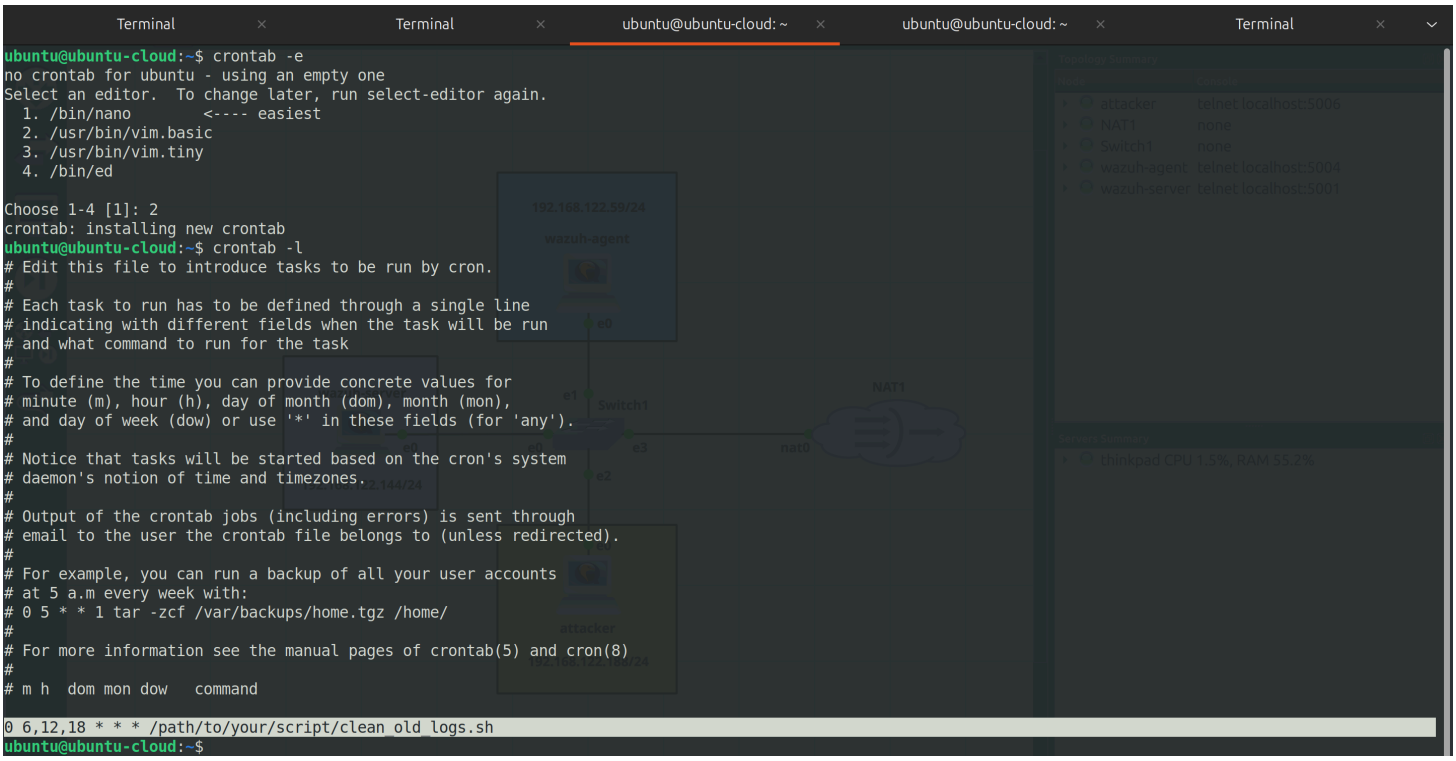
- And here is the configuration of a cronjob that runs my script 3 times a day (at 6am, 12pm, 6pm) every day.

```
0 6,12,18 * * * /path/to/your/script/clean_old_logs.sh
```

- So, I created my script on the `wazuh-server`



- And created a cronjob



Bonus

1.

> **Task description**
>
> What are indices, and how do they differ from log files?

Definition

- An **index** in Elasticsearch or similar log database (such as Wazuh Indexer) is a collection of documents that share similar characteristics. It is akin to a database in traditional relational database management systems. Each index is essentially a **data structure optimized for fast search and retrieval operations**. Indices hold documents in a format that Elasticsearch can efficiently process. Each document represents a discrete piece of data (like a JSON object) and can contain various fields with associated values.

Comparison to log files

- **Purpose**:
  - **Log Files**: Serve as raw data sources where events and logs are generated. They are the initial form of data storage before any processing.

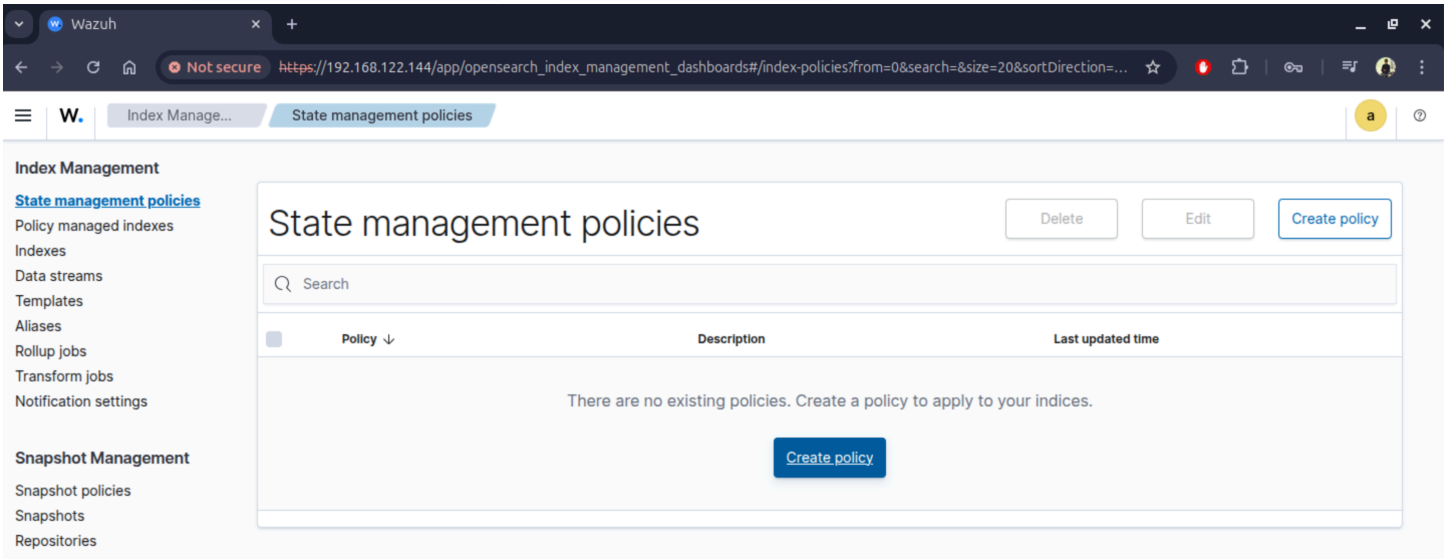
    - **Indices**: Serve as organized collections of processed data that enable fast querying and analysis. They allow for efficient searching and aggregations based on structured data.
- **Structure**:
    - **Log Files**: Typically unstructured or semi-structured text files, which can be difficult to query without processing.
    - **Indices**: Structured data storage with defined mappings and fields, optimized for rapid searches and analyses based on those fields.
- **Lifecycle**:
    - **Log Files**: May be stored on disk in a particular location or directory for a specified duration, governed by log rotation policies.
    - **Indices**: Managed and retained according to index lifecycle policies in a database, which can include retention, aliasing, and deletion strategies.
- **Querying**:
    - **Log Files**: Queries against raw log files require additional tools or scripts to search through plain text data.
    - **Indices**: Queries against indices can leverage Elasticsearch's powerful search capabilities, allowing for precise queries and visualizations.

2.

Task description

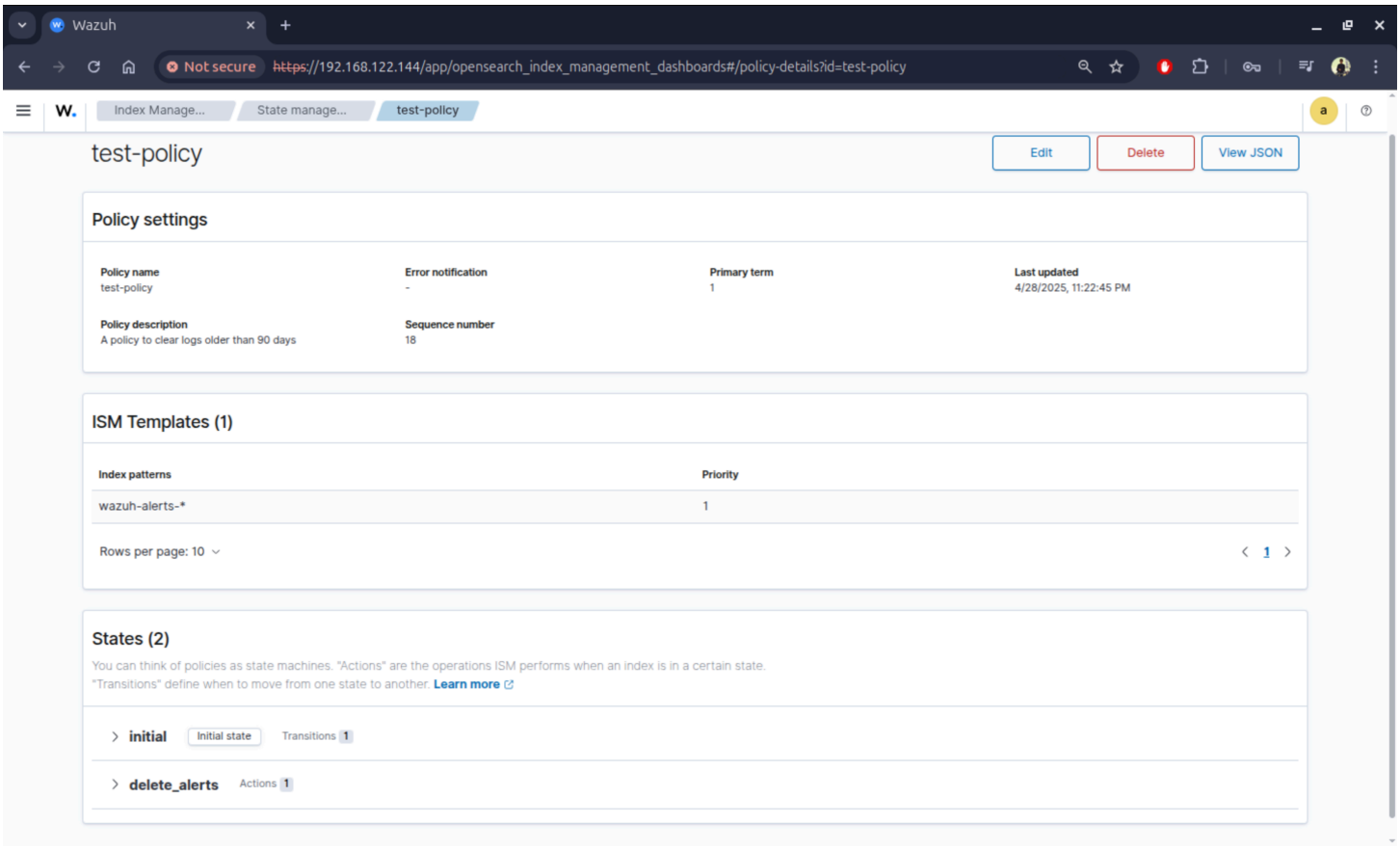
Create an index retention policy to delete Wazuh alert indices after 90 days.

- As always I found an instruction on the Wazuh official website
- I went to the `Indexer Management` -> `Index Management`

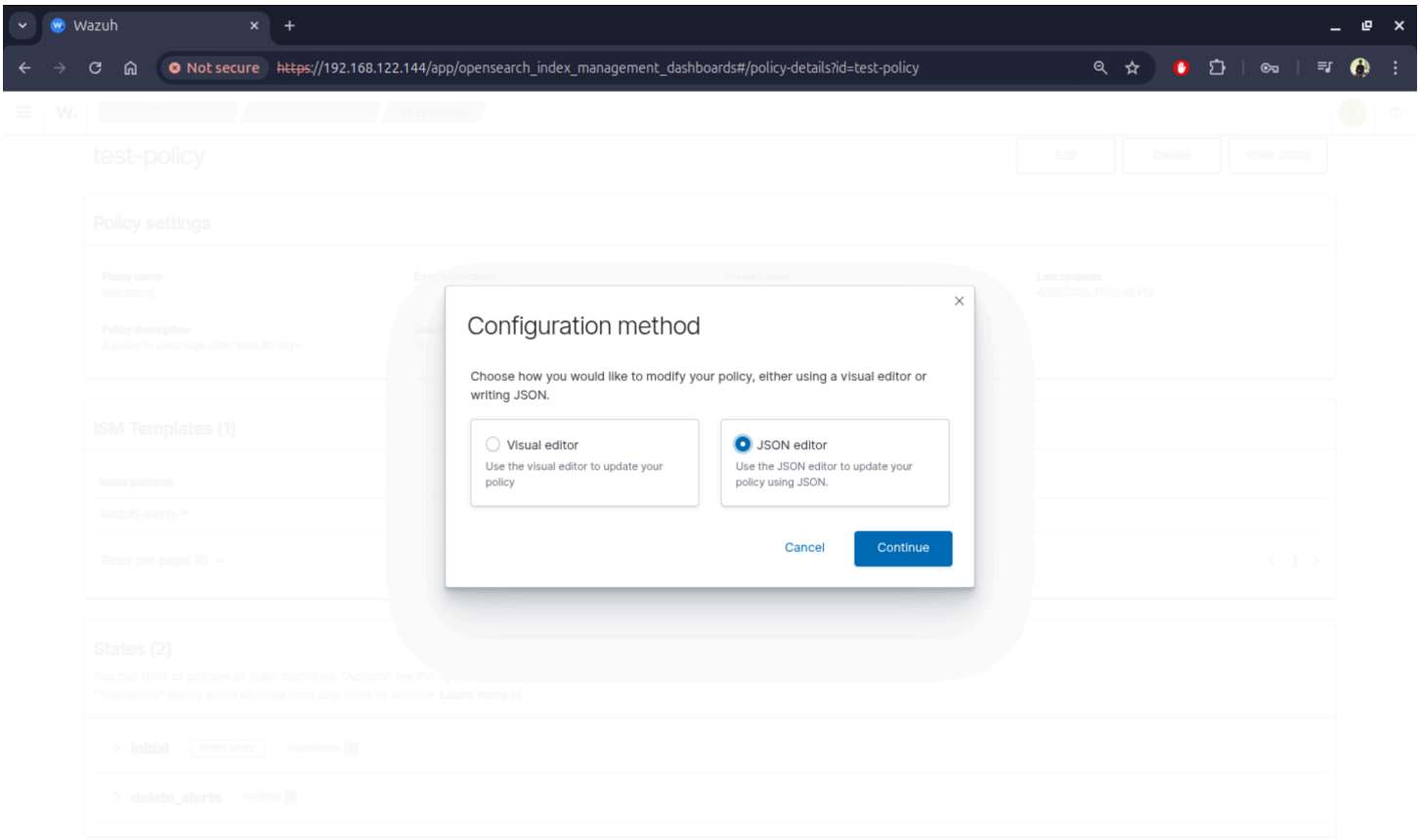


- I was suggested to select a method to create a new policy:
    - Visual editor in Web UI
    - Or in JSON editor

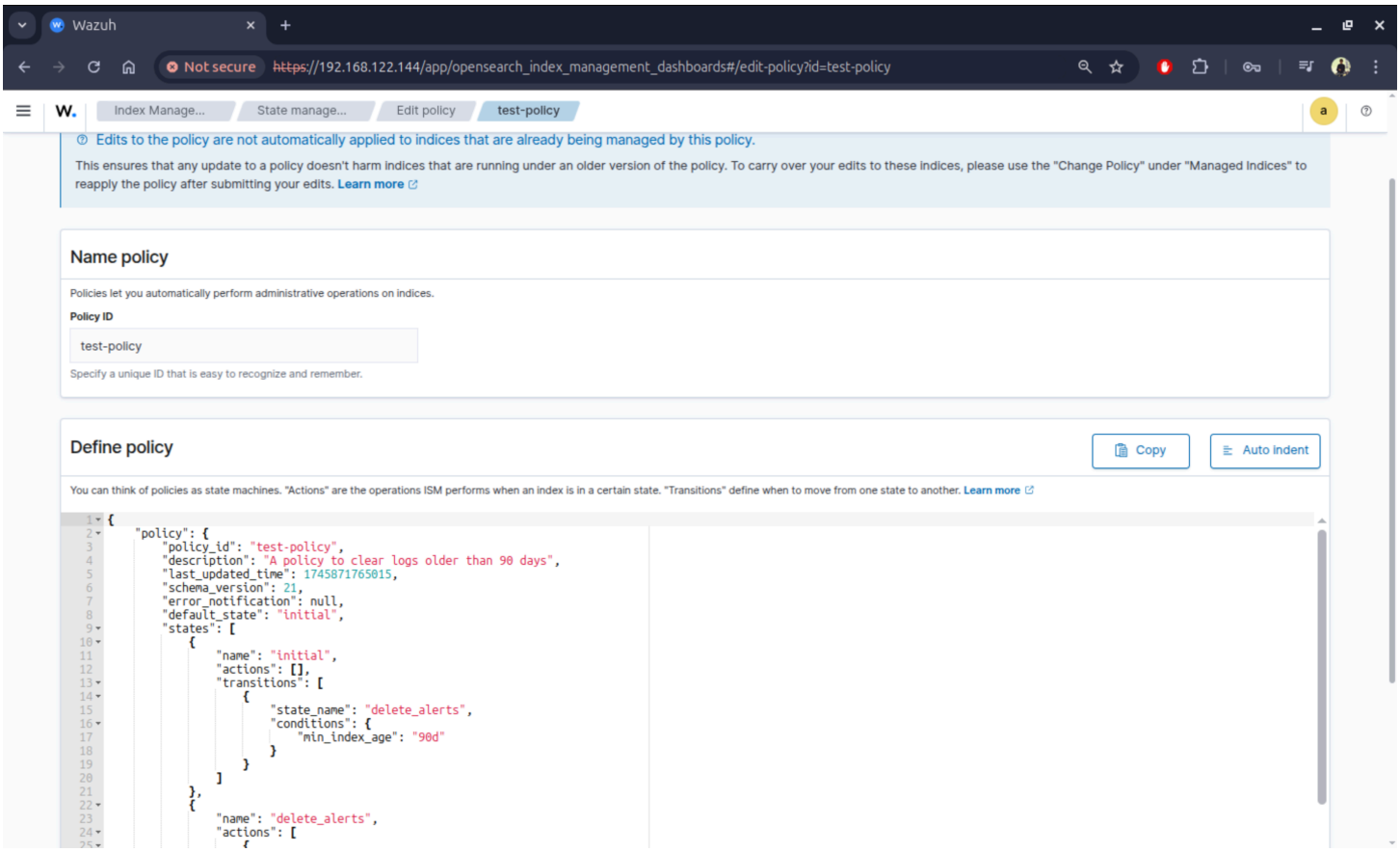
- Using a visual editor I created a new `test-policy` policy and configured two states for `wazuh-alerts-*` indices:
  - `inital` : logs are initially in this state
  - `delete` : after 90 days logs are transited to this state
- After a transition to `delete` state logs are immediately deleted
- The result is below



- Then I simply selected to modify the policy this time using the JSON editor



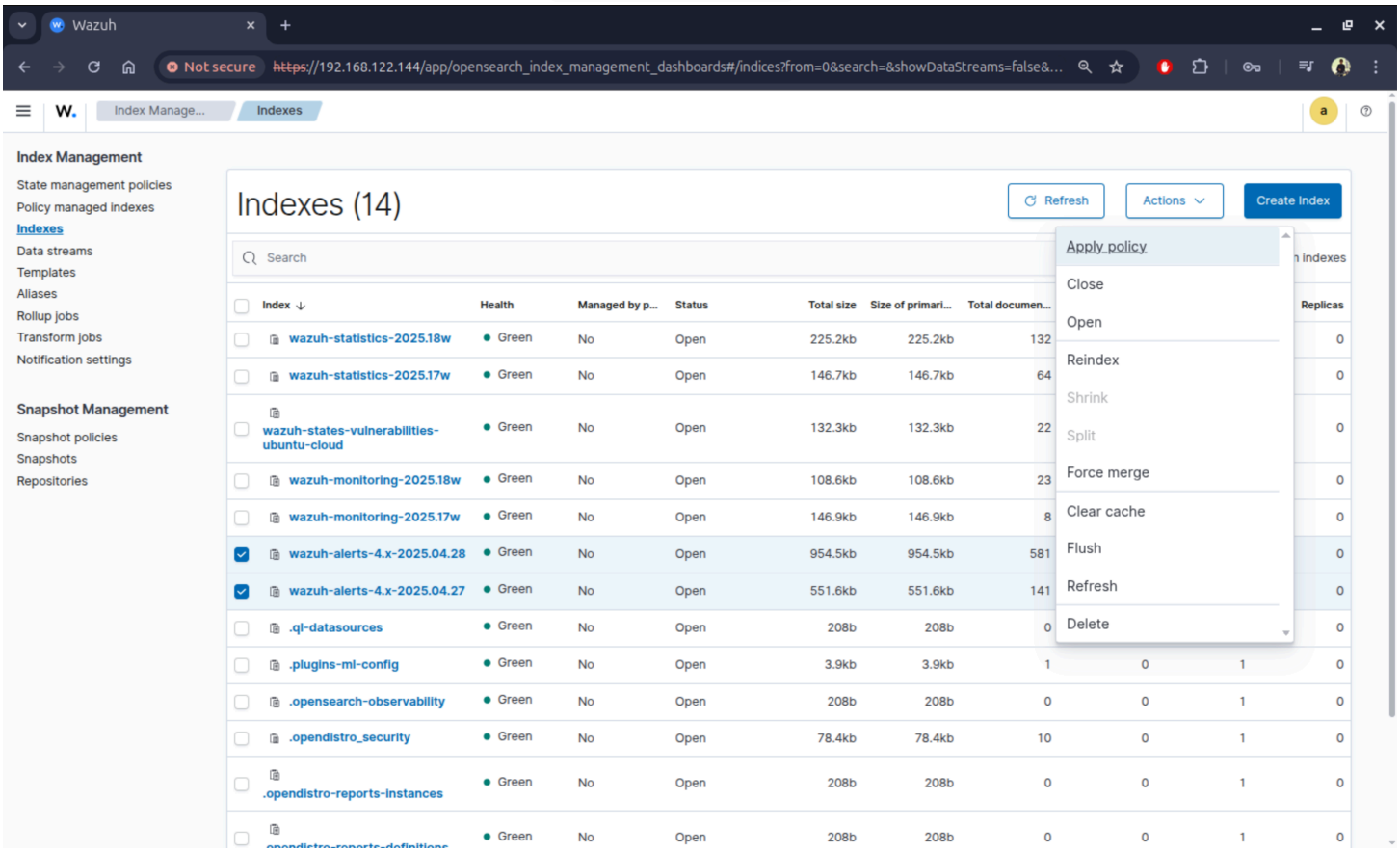
- The JSON editor was appeared and I could see the configuration of my log retention policy in a JSON format

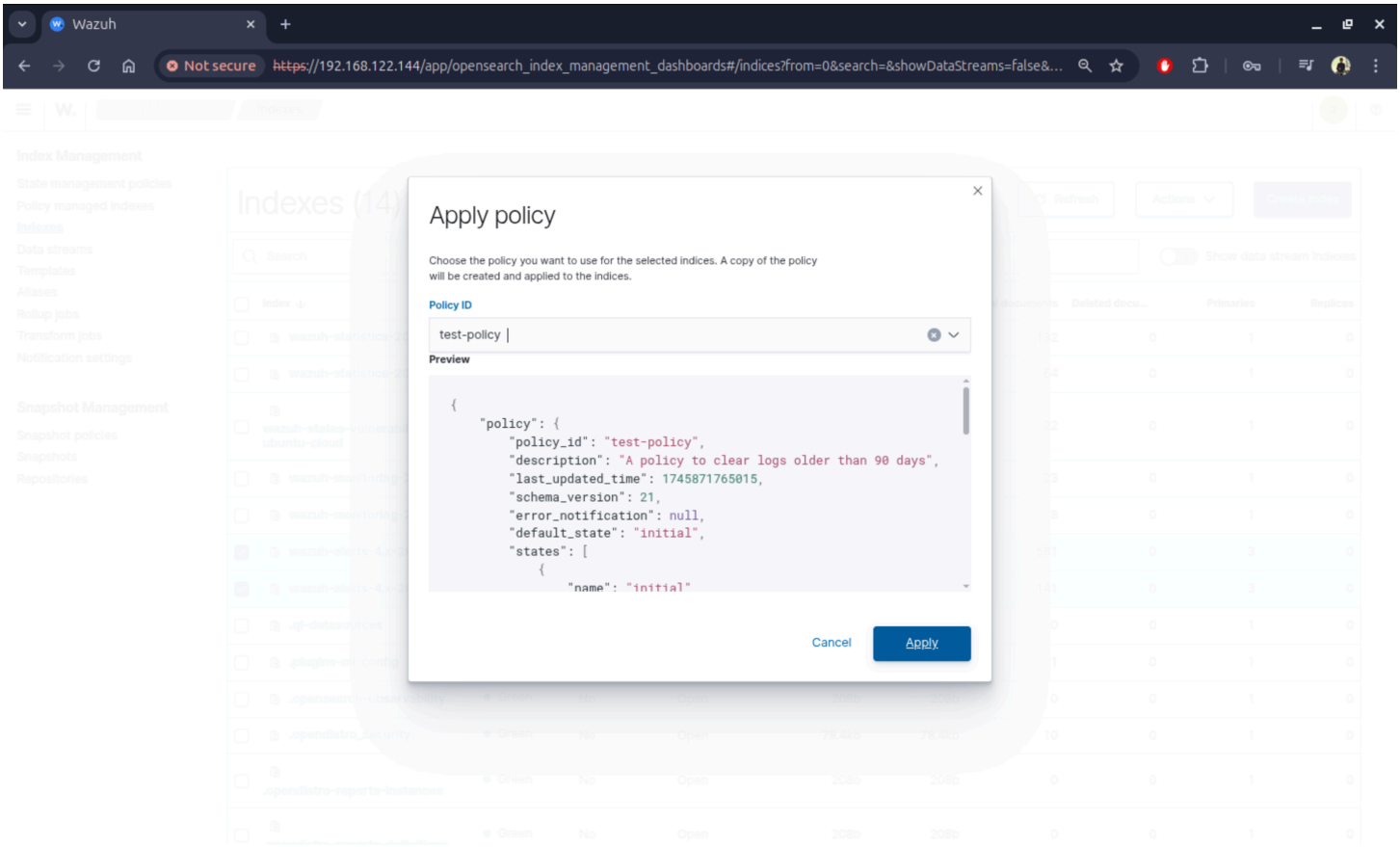


- So, below is the configuration of my log retention policy

```
{
  "policy": {
    "policy_id": "test-policy",
    "description": "A policy to clear logs older than 90 days",
    "last_updated_time": 1745871765015,
    "schema_version": 21,
    "error_notification": null,
    "default_state": "initial",
    "states": [
      {
        "name": "initial",
        "actions": [],
        "transitions": [
          {
            "state_name": "delete_alerts",
            "conditions": {
              "min_index_age": "90d"
            }
          }
        ]
      },
      {
        "name": "delete_alerts",
        "actions": [
          {
            "retry": {
              "count": 3,
              "backoff": "exponential",
              "delay": "1m"
            },
            "delete": {}
          }
        ],
        "transitions": []
      }
    ],
    "ism_template": [
      {
        "index_patterns": [
          "wazuh-alerts-*"
        ],
        "priority": 1,
        "last_updated_time": 1745871200545
      }
    ]
  }
}
```

- Then I applied this policy to the existing `wazuh-alerts-*` indices manually





- Finally, I could see that my policy was applied to the `wazuh-alerts` indices