

Wazuh n8n Integration

Talal Ahmed | Team Lambda

Pre-requisites

- Wazuh Manager and Agent setup
- A hosted instance of n8n

I already have Wazuh set up on Ubuntu 24.04.

Installing n8n

To install n8n, we first need to have docker installed.

Install docker with the following commands:

```
sudo apt update
sudo apt install docker.io -y
sudo usermod -aG docker $USER
newgrp docker
```

After that, we can install and run n8n in a container with the following commands:

```
wazuh@ubuntu: ~  
wazuh@ubuntu:~$ docker volume create n8n_data  
n8n_data  
wazuh@ubuntu:~$ docker run -it --rm \  
  --name n8n \  
  -p 5678:5678 \  
  -e GENERIC_TIMEZONE="ASIA/KARACHI" \  
  -e TZ="ASIA/KARACHI" \  
  -e N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true \  
  -e N8N_RUNNERS_ENABLED=true \  
  -v n8n_data:/home/node/.n8n \  
  docker.n8n.io/n8nio/n8n
```

```

Apr 5 10:27 AM
wazuh@ubuntu: ~
Finished migration MigrateExternalSecretsToEntityStorage1771500000000
Starting migration AddUnshareScopeToCustomRoles1771500000001
Finished migration AddUnshareScopeToCustomRoles1771500000001
Starting migration AddFilesColumnToChatHubAgents1771500000002
Finished migration AddFilesColumnToChatHubAgents1771500000002
Starting migration AddSuggestedPromptsToAgentTable1772000000000
Finished migration AddSuggestedPromptsToAgentTable1772000000000
Starting migration AddRoleColumnToProjectSecretsProviderAccess1772619247761
Finished migration AddRoleColumnToProjectSecretsProviderAccess1772619247761
Starting migration ChangeWorkflowPublishedVersionFKsToRestrict1772619247762
Finished migration ChangeWorkflowPublishedVersionFKsToRestrict1772619247762
Starting migration AddTypeToChatHubSessions1772700000000
Finished migration AddTypeToChatHubSessions1772700000000
Starting migration CreateCredentialDependencyTable1773000000000
[CreateCredentialDependencyTable1773000000000] Backfilled credential dependencies for 0 credentials. Inserted 0 dependencies.
Finished migration CreateCredentialDependencyTable1773000000000
n8n Task Broker ready on 127.0.0.1, port 5679
Failed to start Python task runner in internal mode. because Python 3 is missing from this system. Launching a Python runner in internal mode is intended only for debugging and is not recommended for production. Users are encouraged to deploy in external mode. See: https://docs.n8n.io/hosting/configuration/task-runners/#setting-up-external-mode

There is a deprecation related to your environment variables. Please take the recommended actions to update your configuration:
- N8N_RUNNERS_ENABLED -> Remove this environment variable; it is no longer needed.

[license SDK] Skipping renewal on init: license cert is not initialized
Registered runner "JS Task Runner" (b0PNsbWMe0-o04FSbWlOC)
Version: 2.14.2
Building workflow dependency index...
Finished building workflow dependency index. Processed 0 draft workflows, 0 published workflows.

Editor is now accessible via:
http://localhost:5678

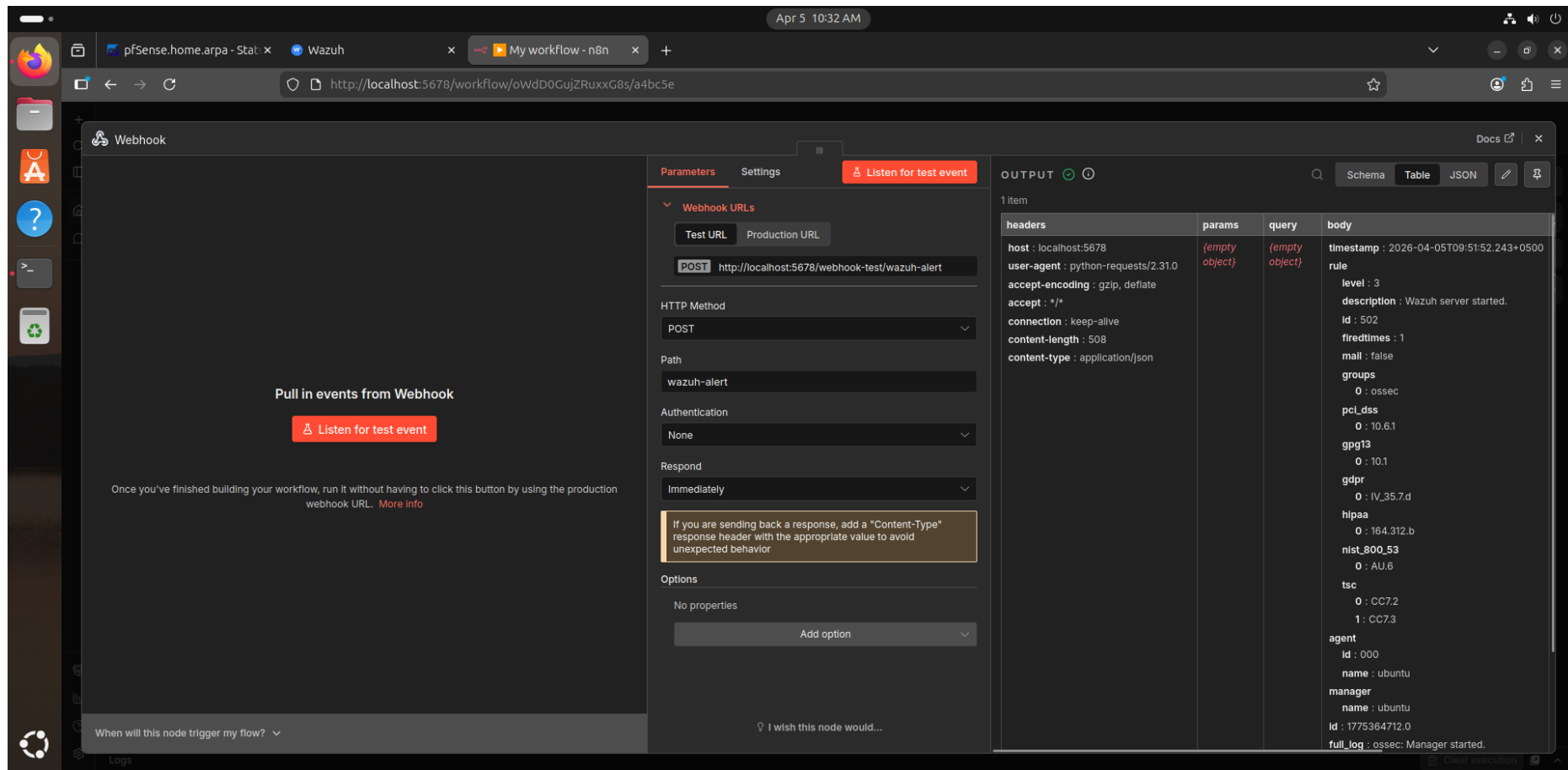
Press "o" to open in Browser.
OOwner was set up successfully
(node:7) [DEP0060] DeprecationWarning: The `util._extend` API is deprecated. Please use Object.assign() instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
User survey updated successfully

```

Now, we can access our n8n instance from <http://localhost:5678>

Creating a Webhook flow

We first create a webhook node to receive the alert from Wazuh



The screenshot shows the n8n Webhook node configuration in a web browser. The browser tabs include 'pfSense.home.arpa - Stati...', 'Wazuh', and 'My workflow - n8n'. The address bar shows 'http://localhost:5678/workflow/oWdD0GujZRuxxG8s/a4bc5e'.

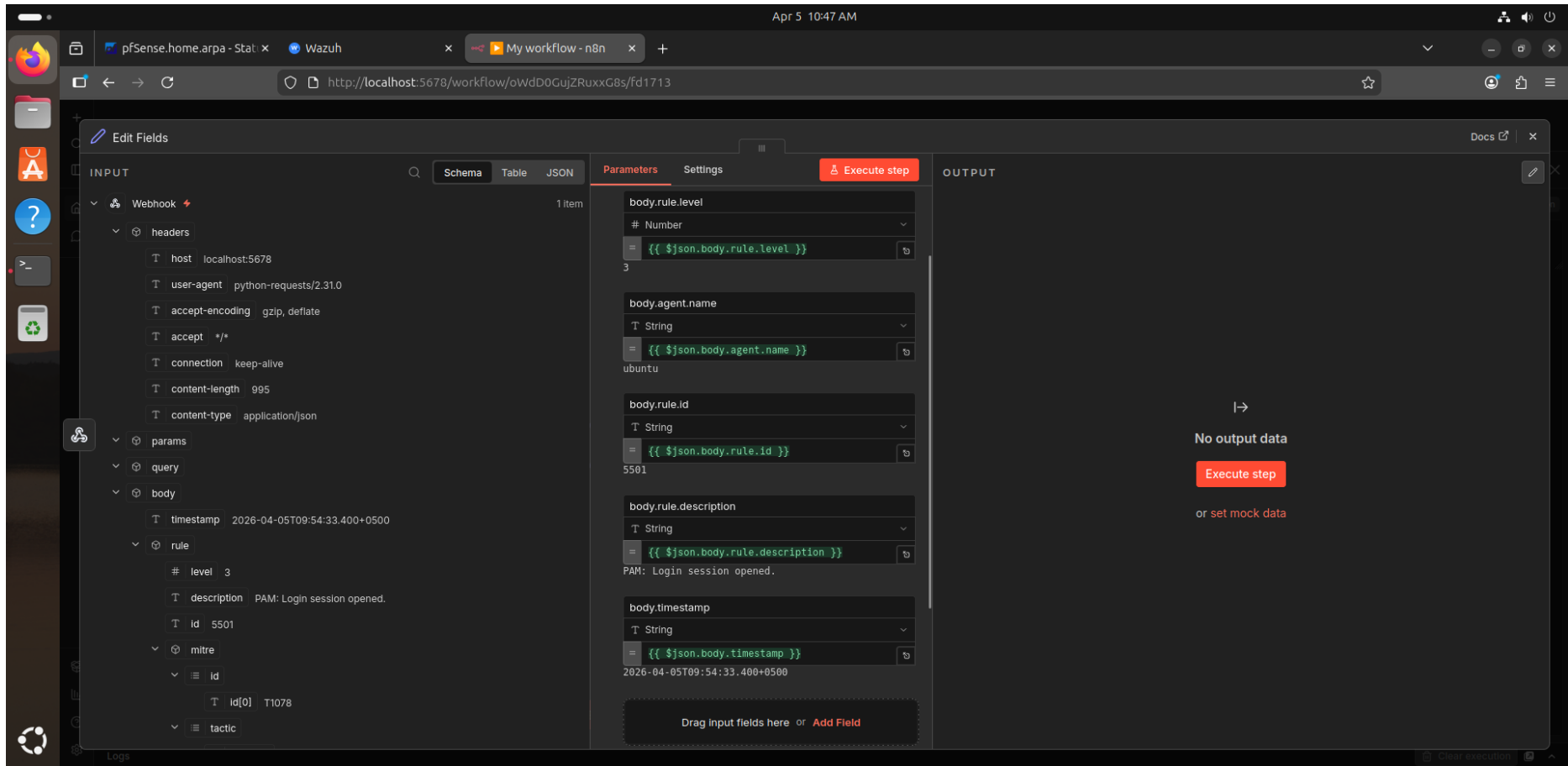
The Webhook node configuration is as follows:

- Webhook URLs:** Test URL: `http://localhost:5678/webhook-test/wazuh-alert`
- HTTP Method:** POST
- Path:** wazuh-alert
- Authentication:** None
- Respond:** Immediately
- Options:** No properties

The **OUTPUT** section shows 1 item:

headers	params	query	body
host : localhost:5678 user-agent : python-requests/2.31.0 accept-encoding : gzip, deflate accept : */* connection : keep-alive content-length : 508 content-type : application/json	{empty object}	{empty object}	timestamp : 2026-04-05T09:51:52.243+0500 rule level : 3 description : Wazuh server started. id : 502 firedtimes : 1 mail : false groups 0 : ossec pci_dss 0 : 10.6.1 gpg13 0 : 10.1 gdpr 0 : IV_35.7.d hipaa 0 : 164.312.b nist_800_53 0 : AU.6 tsc 0 : CC7.2 1 : CC7.3 agent id : 000 name : ubuntu manager name : ubuntu id : 1775364712.0 full_log : ossec: Manager started.

Next, we will filter only the values we need by adding a filter node. I filtered the alert level, ID, description, timestamp and the agent.



Apr 5 10:47 AM

pfSense.home.arpa - Stat: x Wazuh x My workflow - n8n x

http://localhost:5678/workflow/oWdD0GujZRuxxGBs/fd1713

Edit Fields

INPUT

Webhook 1 item

headers

host localhost:5678

user-agent python-requests/2.31.0

accept-encoding gzip, deflate

accept */*

connection keep-alive

content-length 995

content-type application/json

params

query

body

timestamp 2026-04-05T09:54:33.400+0500

rule

level 3

description PAM: Login session opened.

id 5501

mitre

id

id[0] T1078

tactic

Parameters

body.rule.level

Number

= {{ \$json.body.rule.level }}

3

body.agent.name

T String

= {{ \$json.body.agent.name }}

ubuntu

body.rule.id

T String

= {{ \$json.body.rule.id }}

5501

body.rule.description

T String

= {{ \$json.body.rule.description }}

PAM: Login session opened.

body.timestamp

T String

= {{ \$json.body.timestamp }}

2026-04-05T09:54:33.400+0500

OUTPUT

No output data

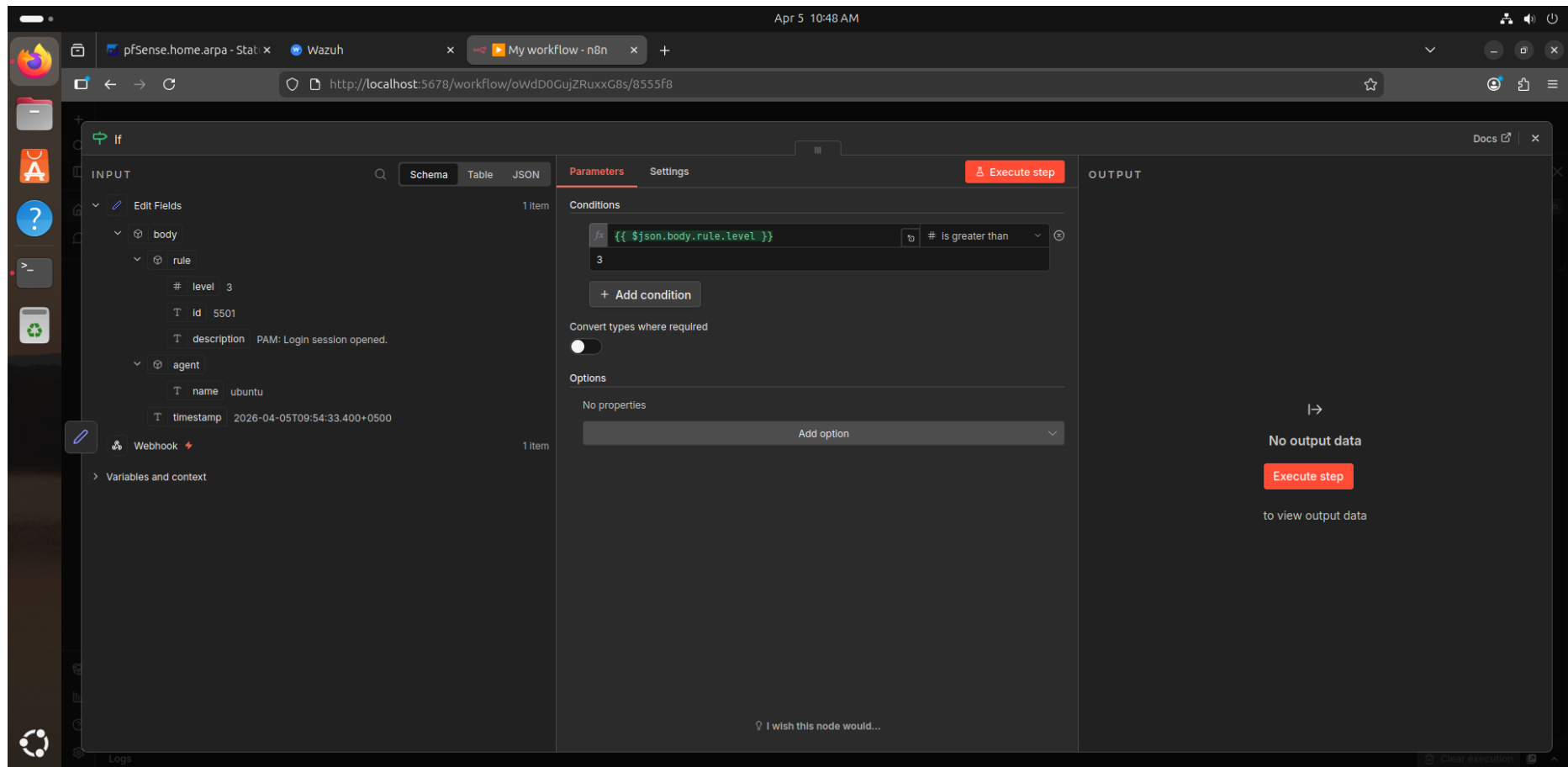
Execute step

or set mock data

Drag input fields here or Add Field

Clear execution

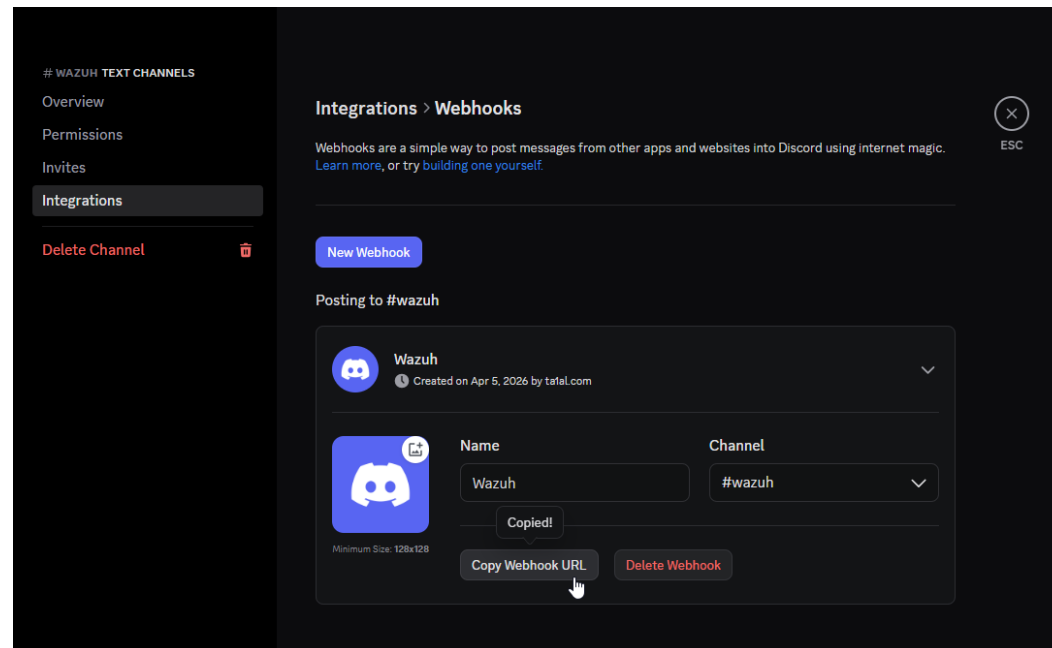
Next, we will filter for alert with level higher than 3.



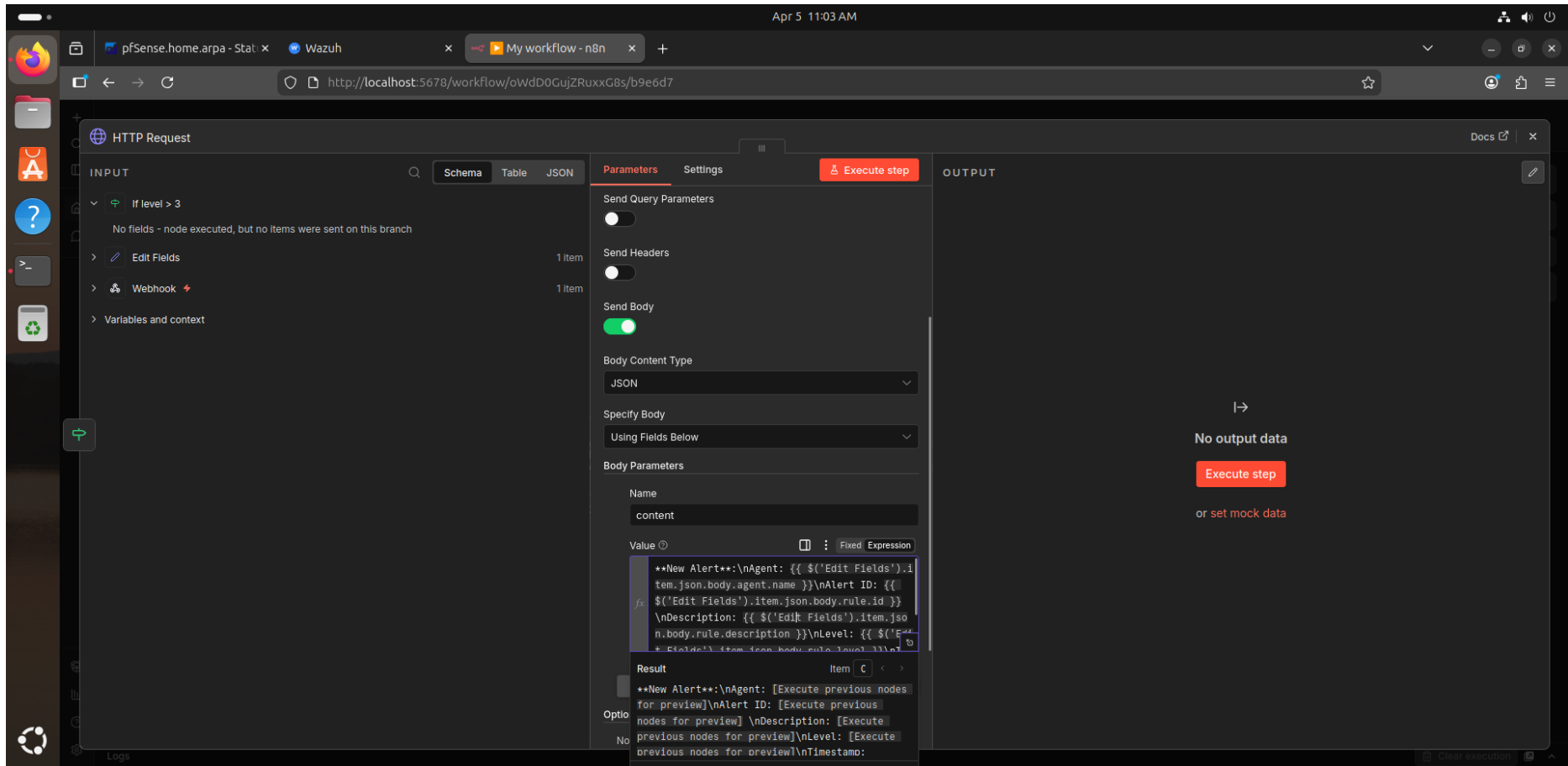
This will significantly reduce the noise.

[Sending Alerts to Discord](#)

Since I will be sending the alerts to Discord. I will create a simple Discord webhook.



Next, I will create a POST request node on n8n. I will also include the content field in the body as per discord webhook requirements.



The screenshot shows the n8n workflow editor interface. The top bar indicates the date and time as 'Apr 5 11:03 AM'. The browser address bar shows the URL 'http://localhost:5678/workflow/oWdD0GujZRuxxG8s/b9e6d7'. The workflow is titled 'My workflow - n8n'.

The main workspace displays an 'HTTP Request' node configuration. The 'Parameters' tab is active, showing the following settings:

- Send Query Parameters:** Disabled (toggle off)
- Send Headers:** Disabled (toggle off)
- Send Body:** Enabled (toggle on)
- Body Content Type:** JSON (dropdown menu)
- Specify Body:** Using Fields Below (dropdown menu)
- Body Parameters:**
 - Name:** content
 - Value:** `**New Alert**:\nAgent: {{ $('Edit Fields').item.json.body.agent.name }}\nAlert ID: {{ $('Edit Fields').item.json.body.rule.id }}\nDescription: {{ $('Edit Fields').item.json.body.rule.description }}\nLevel: {{ $('Edit Fields').item.json.body.rule.level }}`

The 'OUTPUT' panel on the right shows 'No output data' and an 'Execute step' button. Below it, there is a link 'or set mock data'.

The 'INPUT' panel on the left shows the workflow structure:

- If level > 3:** No fields - node executed, but no items were sent on this branch
- Edit Fields:** 1 item
- Webhook:** 1 item
- Variables and context:**

The 'Result' tab at the bottom shows the output of the 'Execute step' node:

```

**New Alert**:\nAgent: [Execute previous nodes for preview]\nAlert ID: [Execute previous nodes for preview] \nDescription: [Execute previous nodes for preview]\nLevel: [Execute previous nodes for preview]\nTimestamp:
  
```

Reading and Forwarding Alerts

In order to forward alerts, we need to read the alerts from `/var/ossec/logs/alerts/alerts.json` and send them to the n8n listener webhook.

To do this, I used the following python script:


```
#!/usr/bin/env python3
import json
import requests
import time
import os

# Path to Wazuh alerts log
ALERT_FILE = "/var/ossec/logs/alerts/alerts.json"

# Your n8n webhook URL
WEBHOOK_URL = "http://localhost:5678/webhook-test/wazuh-alert"

# File to keep track of sent alerts
SENT_ALERTS_FILE = "/var/ossec/logs/alerts/sent_alerts.txt"

# Load already sent alerts
if os.path.exists(SENT_ALERTS_FILE):
    with open(SENT_ALERTS_FILE, "r") as f:
        sent_alerts = set(line.strip() for line in f.readlines())
else:
    sent_alerts = set()

def send_alert(alert):
    try:
        response = requests.post(WEBHOOK_URL, json=alert, timeout=10)
        if response.status_code == 200:
            print(f"Alert sent: {alert.get('rule', '')}.get('description', '')")
            return True
        else:
            print(f"Failed to send alert: {response.status_code} {response.text}")
            return False
    except Exception as e:
        print(f"Error sending alert: {e}")
        return False

while True:
    try:
        with open(ALERT_FILE, "r") as f:
            for line in f:
                line = line.strip()
                if not line:
                    continue
                try:
                    alert = json.loads(line)
                    alert_id = alert.get("id") or alert.get("timestamp") or json.dumps(alert)
                    if alert_id not in sent_alerts:
                        if send_alert(alert):
                            sent_alerts.add(alert_id)
                except json.JSONDecodeError as e:
                    print(f"Skipping invalid JSON line: {e}")

        # Save sent alerts
        with open(SENT_ALERTS_FILE, "w") as f:
            for a in sent_alerts:
                f.write(f"{a}\n")

    except Exception as e:
        print(f"Error reading alerts: {e}")

    time.sleep(30)
```

After I start the workflow and the script. I started receiving alerts on Discord:

