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.

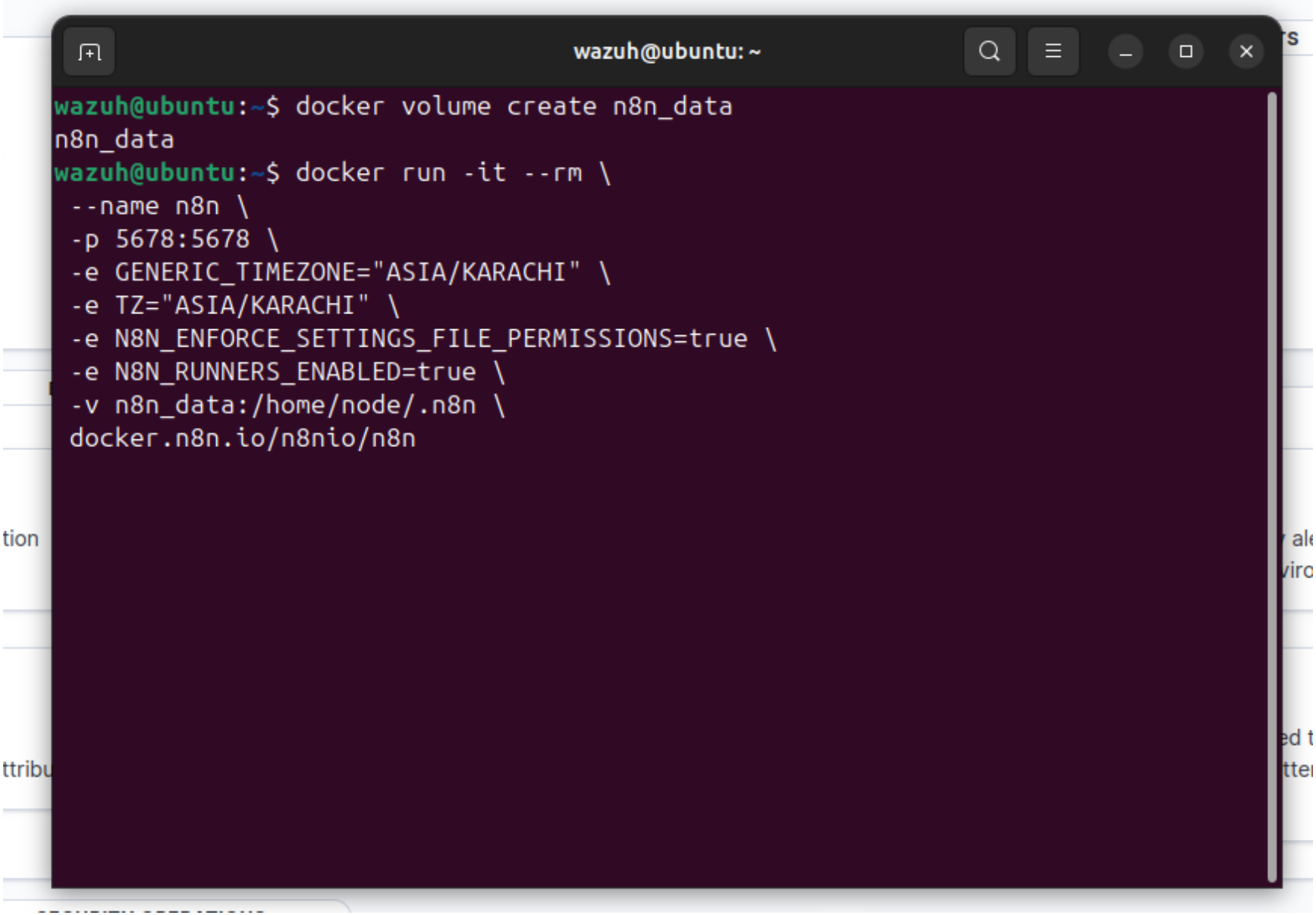
Installing n8n

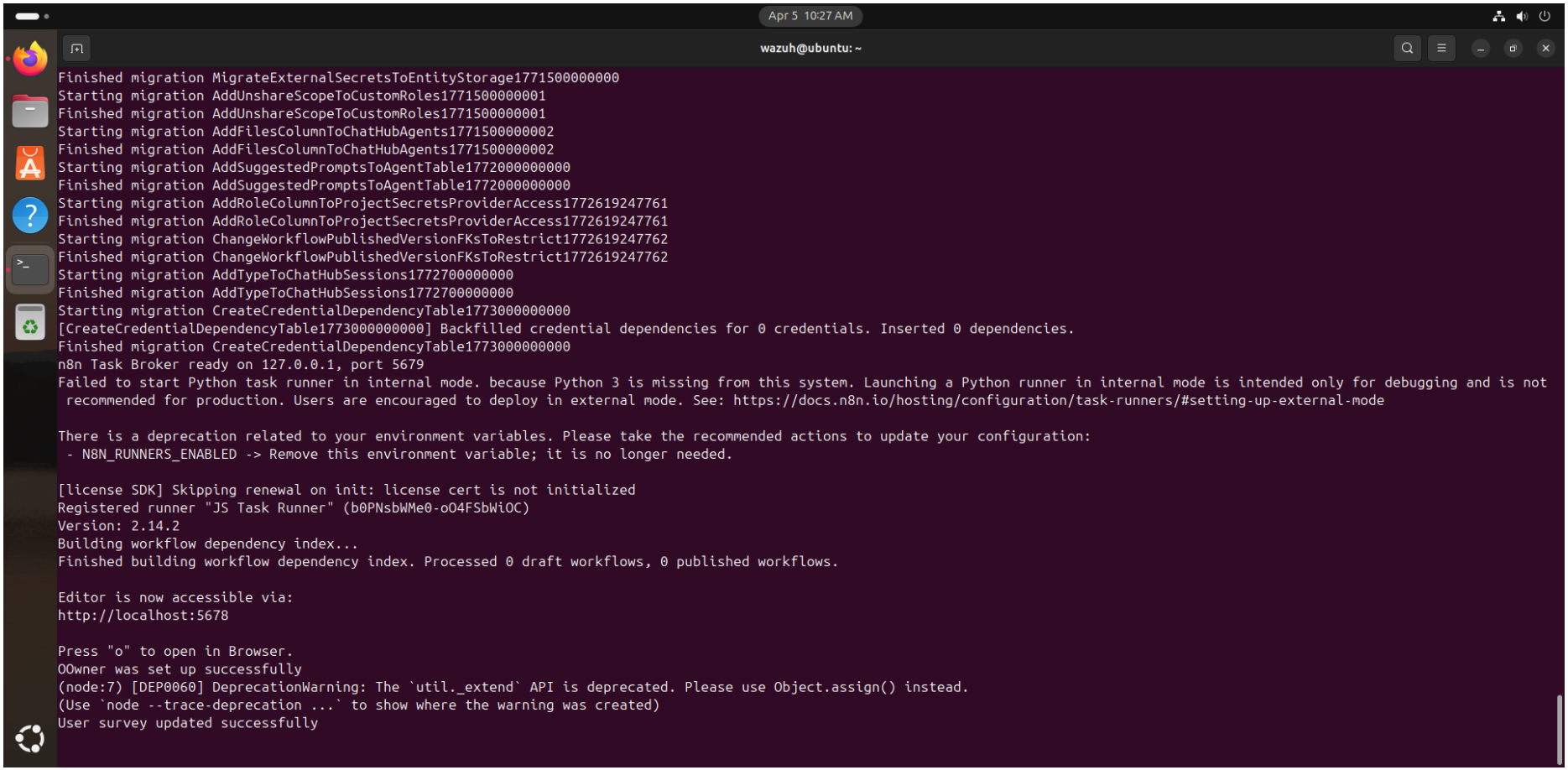
To install n8n, we first need to have docker installed.

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:

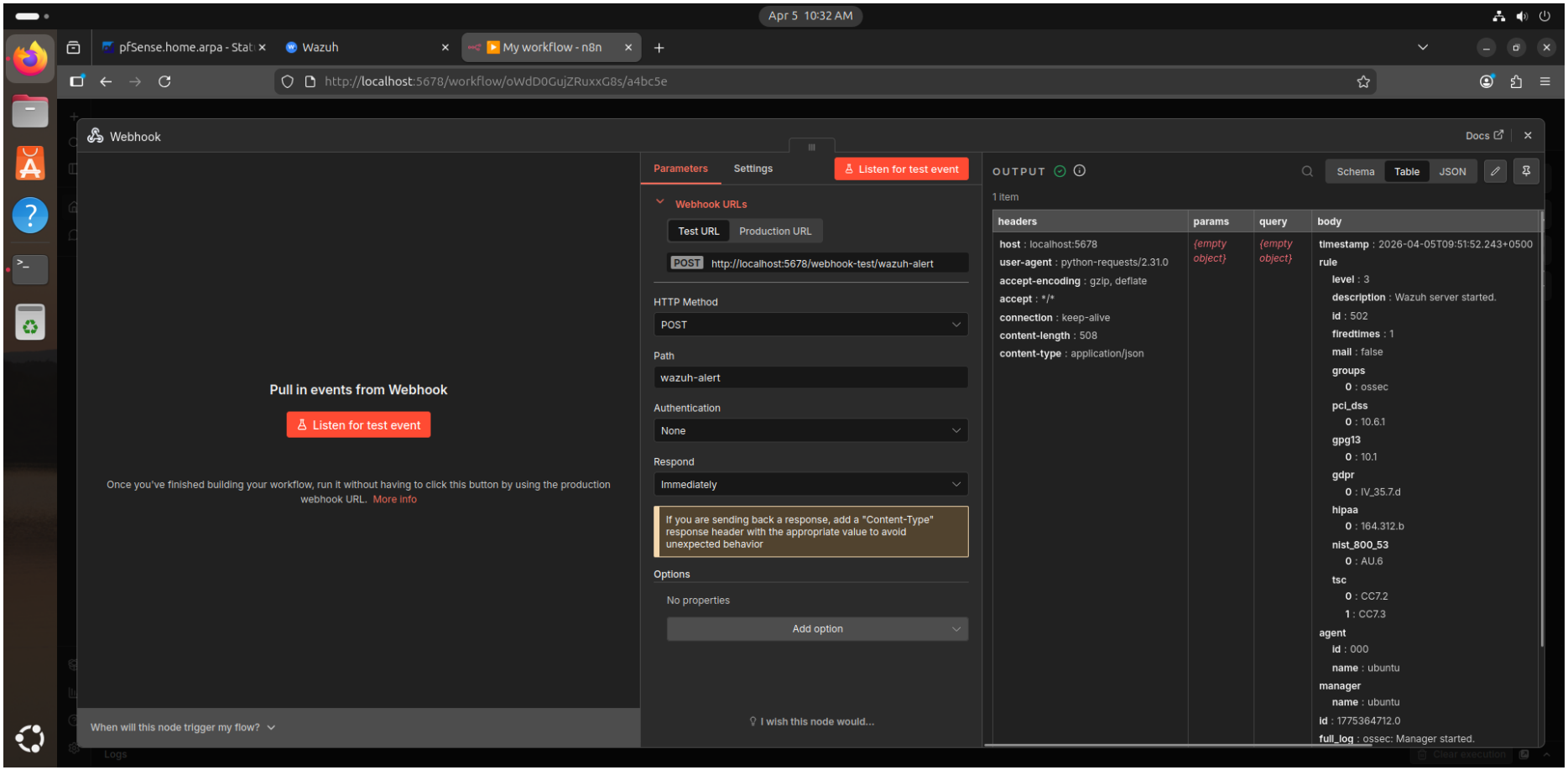

wazuh@ubuntu: ~
wazuh@ubuntu:~$ docker volume create n8n_data
n8n_data
wazuh@ubuntu:~$ docker run -it --rm \
--name n8n \
-p 5678:5678 \
-e GENERIC_TIMEZONE="ASIA/KARACHI" \
-e TZ="ASIA/KARACHI" \
-e N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true \
-e N8N_RUNNERS_ENABLED=true \
-v n8n_data:/home/node/.n8n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



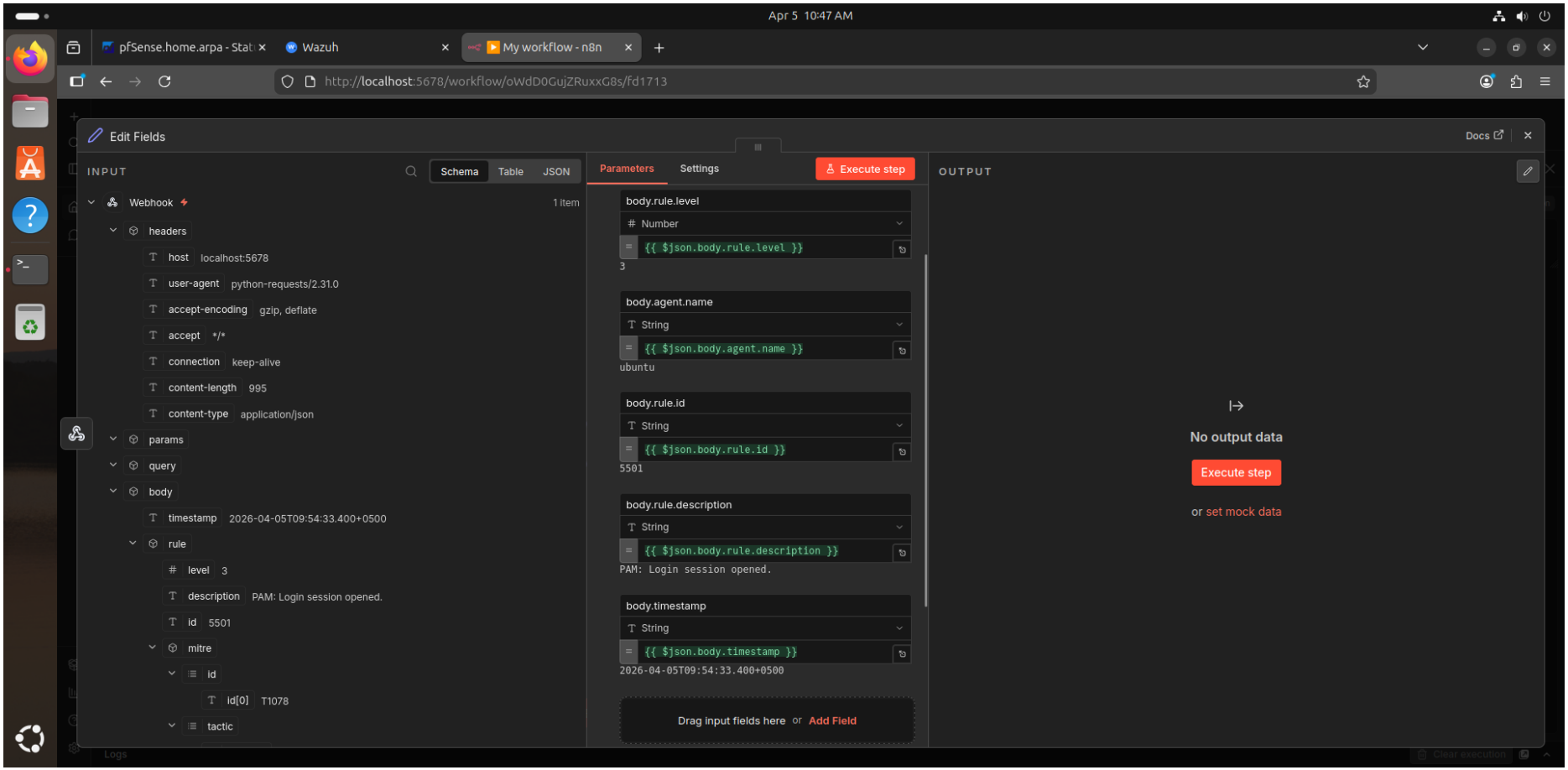
Now, we can access our n8n instance from https://localhost:5678

Creating a Webhook flow

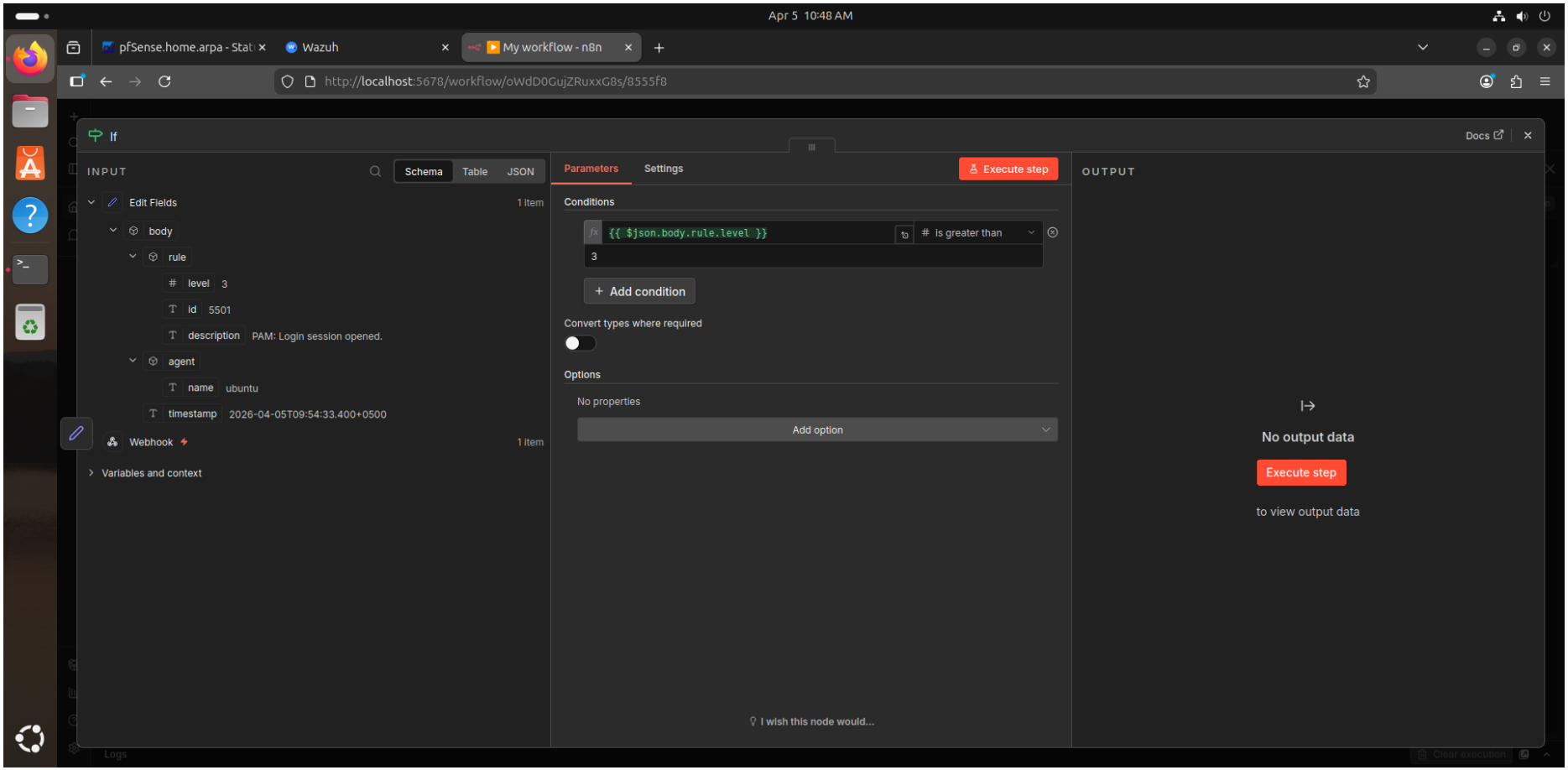
We first create a webhook node to receive the alert from Wazuh



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.



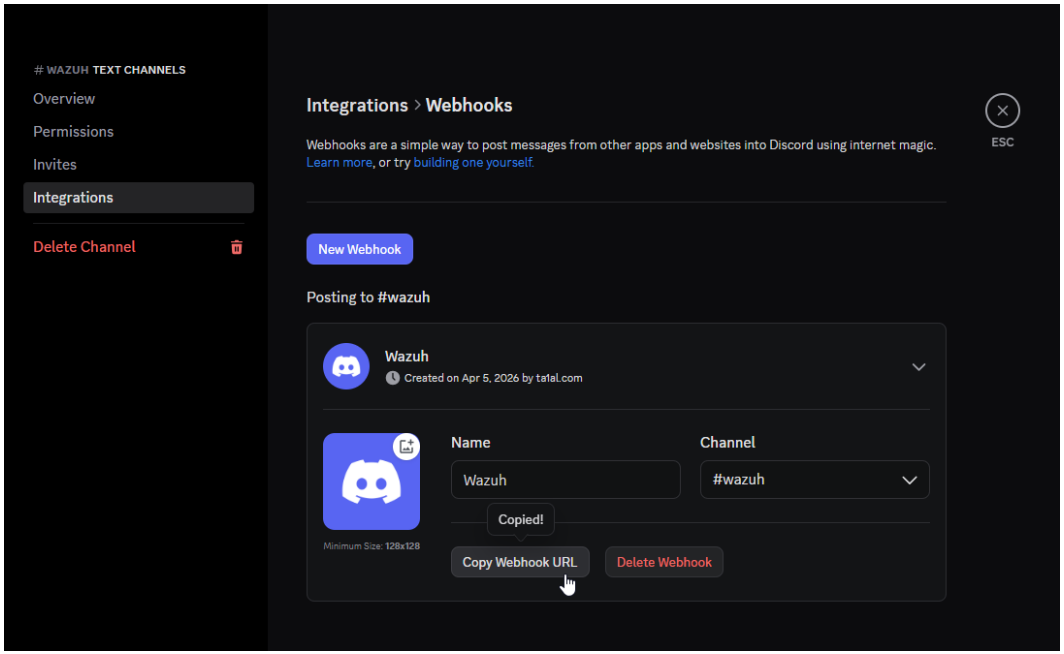
Next, we will filter for alert with level higher than 3.



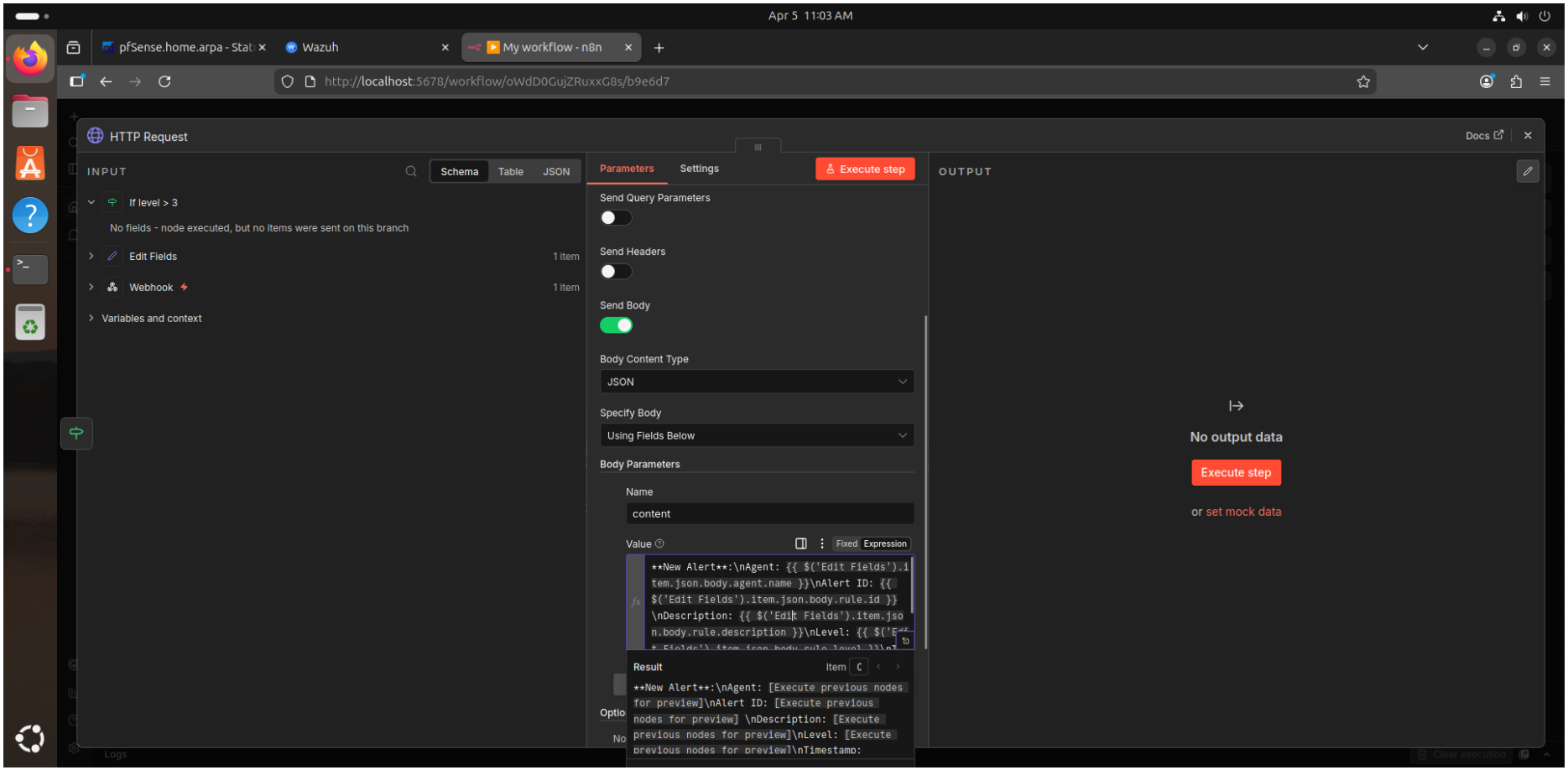
This will significantly reduce the noise.

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.



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:

```python
#!/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: