EN

OPNsense Monitoring with Grafana + InfluxDB

Real-Time Dashboards — OPNsense Stack Series #9

Mars 2026 | botum.ca

OPNsense Monitoring with Grafana and InfluxDB

Real-Time Dashboards — Episode 9/9 of the OPNsense Stack Series

Table of Contents

Full article: blog.botum.ca/opnsense-monitoring-grafana-influxdb-guide

EN

OPNsense Monitoring with Grafana + InfluxDB

Real-Time Dashboards — OPNsense Stack Series #9

Mars 2026 | botum.ca

Series hub: blog.botum.ca/opnsense-stack-securite-enterprise-proxmox

1. Why Monitor Your Firewall?

A firewall without monitoring is like driving blindfolded. In my BOTUM infrastructure, I learned the hard way that the most costly network issues are the ones you don't see coming: a silent WAN saturation at 3am, an intrusion attempt slipping through the cracks, an IoT VLAN generating 10x its normal traffic.

Real-time monitoring with Grafana + InfluxDB gives me three essential capabilities:

• **Full visibility:** see in real-time what crosses each interface, VLAN, and VPN tunnel.

• **Anomaly detection:** identify abnormal patterns — traffic spikes, DNS floods, connections to suspicious IPs.

• **SLA compliance:** prove uptime, document network incidents, measure WAN latency.

■ *This is the 9th and final episode of the OPNsense Stack Series. You already have the firewall, VLANs, VPN, WiFi, IDS/IPS protection, NAC, and DNS filtering. Now you get the eyes.*

2. Monitoring Stack Architecture

The stack I use is simple and battle-tested:

```
OPNsense (raw metrics)
     ↓  os-telegraf plugin (collection)
     ↓
Telegraf (metrics agent)
     ↓  HTTP/UDP to port 8086
     ↓
InfluxDB 2.x (time-series database)
     ↓  Grafana data source
     ↓
Grafana (visualization + alerting)
     ↓  notifications
     ↓
Email / Telegram / Slack
```

Each component runs in a Docker container on a dedicated Proxmox VM (or on the same Proxmox host as OPNsense). I allocate 2 vCPUs + 2 GB RAM — more than enough for infrastructure handling up to 1 Gbps of aggregated traffic.

■ *Telegraf is a lightweight agent installed inside OPNsense that collects system and network metrics, then forwards them to InfluxDB via the HTTP API.*

3. Enable InfluxDB Export in OPNsense

3.1 Install the os-telegraf Plugin

```
# In OPNsense: System > Firmware > Plugins
# Search "telegraf" and install: os-telegraf

# Or via SSH in OPNsense:
pkg install os-telegraf
```

3.2 Configure Telegraf in OPNsense

```
# Services > Telegraf > General
# Check: Enable Telegraf
# InfluxDB URL: http://PROXMOX-IP:8086
# Token: (generated in InfluxDB — see section 4)
# Bucket: opnsense
# Organization: botum

# Enable collectors:
# ✓ Interfaces (traffic per interface)
# ✓ Firewall (rules, states)
# ✓ CPU / Memory / Disk
# ✓ Netflow (if enabled)
# ✓ Gateway (WAN latency)
```

3.3 Manual telegraf.conf Configuration (Advanced)

```
# /usr/local/etc/telegraf.conf

[agent]
  interval = "10s"
  flush_interval = "10s"

[[inputs.net]]
  interfaces = ["em0", "vtnet0", "igc0"]

[[inputs.cpu]]
  percpu = false
  totalcpu = true

[[inputs.mem]]

[[inputs.system]]

[[inputs.netstat]]

# OPNsense-specific firewall metrics
[[inputs.exec]]
  commands = ["/usr/local/opnsense/scripts/OPNsense/Diagnostics/interface.php"]
  data_format = "json"
  timeout = "5s"

[[outputs.influxdb_v2]]
  urls = ["http://192.168.1.x:8086"]
  token = "YOUR-INFLUXDB-TOKEN"
  organization = "botum"
  bucket = "opnsense"
  timeout = "5s"
```

4. Install InfluxDB 2.x in Docker on Proxmox

I deploy InfluxDB 2.x on the same Proxmox VM as Grafana to minimize latency and simplify maintenance.

4.1 Docker Compose — Full Monitoring Stack

```
mkdir -p /opt/monitoring/{influxdb,grafana}
cd /opt/monitoring

cat > docker-compose.yml << 'EOF'
version: '3.8'

services:
  influxdb:
    image: influxdb:2.7
    container_name: influxdb
    restart: unless-stopped
    ports:
      - "8086:8086"
    volumes:
      - ./influxdb/data:/var/lib/influxdb2
      - ./influxdb/config:/etc/influxdb2
    environment:
      DOCKER_INFLUXDB_INIT_MODE: setup
      DOCKER_INFLUXDB_INIT_USERNAME: admin
      DOCKER_INFLUXDB_INIT_PASSWORD: BotumSecure2026!
      DOCKER_INFLUXDB_INIT_ORG: botum
      DOCKER_INFLUXDB_INIT_BUCKET: opnsense
      DOCKER_INFLUXDB_INIT_RETENTION: 90d

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    restart: unless-stopped
    ports:
      - "3000:3000"
    volumes:
      - ./grafana:/var/lib/grafana
    environment:
      GF_SECURITY_ADMIN_PASSWORD: BotumSecure2026!
      GF_INSTALL_PLUGINS: grafana-clock-panel,grafana-worldmap-panel
    depends_on:
      - influxdb

EOF

docker compose up -d
echo "Monitoring stack started!"
echo "InfluxDB: http://PROXMOX-IP:8086"
echo "Grafana:  http://PROXMOX-IP:3000"
```

4.2 Generate InfluxDB Token

```
# Via InfluxDB web UI (http://IP:8086)
# 1. Log in with admin / BotumSecure2026!
# 2. Load Data > API Tokens > Generate API Token
# 3. Select: All Access Token
# 4. Copy token (88-char string)
# 5. Paste into OPNsense Telegraf config

# Or via CLI:
docker exec influxdb influx auth create   --org botum   --all-access   --description "Telegraf OP

# Token is shown ONCE — save it in your vault!
```

■ *Store the InfluxDB token in your secrets manager (Vaultwarden / 1Password). Once generated, it won't be displayed again in plaintext.*

5. Configure Grafana with InfluxDB

5.1 Add InfluxDB Data Source

```
# Grafana → Configuration > Data Sources > Add Data Source
# Select: InfluxDB

# Query Language: Flux (recommended for InfluxDB 2.x)
# URL: http://influxdb:8086  (or http://PROXMOX-IP:8086)

# InfluxDB Details:
# Organization: botum
# Token: YOUR-TOKEN
# Default Bucket: opnsense

# Click: Save & Test
# ✓ "datasource is working. 1 buckets found"
```

5.2 Basic Flux Query — WAN Traffic

```
// Flux query — real-time WAN traffic
from(bucket: "opnsense")
  |> range(start: -1h)
  |> filter(fn: (r) => r._measurement == "net")
  |> filter(fn: (r) => r.interface == "em0")
  |> filter(fn: (r) => r._field == "bytes_recv" or r._field == "bytes_sent")
  |> derivative(unit: 1s, nonNegative: true)
  |> map(fn: (r) => ({ r with _value: r._value * 8.0 }))
  |> aggregateWindow(every: 30s, fn: mean, createEmpty: false)
```

6. Import OPNsense Dashboards

The Grafana community offers excellent pre-built dashboards for OPNsense. Here are the ones I use:

- **OPNsense Overview (ID: 13713)** : WAN traffic, CPU, memory, firewall states, gateways
- **OPNsense Interfaces (ID: 14966)** : Throughput per interface, errors, packets/sec
- **OPNsense VPN WireGuard (custom)** : Connected peers, traffic per tunnel, latency
- **VLAN Traffic (custom)** : Traffic per VLAN, top clients, anomalies
- **OPNsense Firewall Rules (ID: 15413)** : Rule hits, blocked connections, IP geolocation

```
# Import via Grafana UI:
# Dashboards > Import > Dashboard ID or JSON

# Quick import from grafana.com:
# Dashboard ID 13713 → OPNsense Overview
# Select: Data Source = InfluxDB (opnsense)
# Import

# Customize dashboard variables:
# Bucket: opnsense
# WAN interfaces: em0, vtnet0 (per your config)
# LAN/VLAN interfaces: vtnet1, vtnet2, vtnet3...
```

■ *Community dashboard IDs may change over time. If an ID is unavailable, search "OPNsense" on grafana.com/grafana/dashboards.*

7. Grafana Alerting: Bandwidth and Suspicious Connections

7.1 Configure a Contact Point (Notification Channel)

```
# Grafana → Alerting > Contact Points > Add Contact Point
# Type: Telegram (recommended for mobile alerts)

# Bot Token: your Telegram Bot Token
# Chat ID: your Chat ID (or group)

# Test: "Test" — you should receive a Telegram message
# Save the contact point

# Available types: Email, Slack, PagerDuty, OpsGenie, Webhook...
```

7.2 Create Alert — WAN Bandwidth Saturation

```
# Grafana → Alerting > Alert Rules > New Alert Rule

# 1. Query:
from(bucket: "opnsense")
  |> range(start: -5m)
  |> filter(fn: (r) => r._measurement == "net" and r.interface == "em0")
  |> filter(fn: (r) => r._field == "bytes_sent" or r._field == "bytes_recv")
  |> derivative(unit: 1s, nonNegative: true)
  |> map(fn: (r) => ({ r with _value: r._value * 8.0 }))
  |> sum()

# 2. Condition:
# WHEN avg() IS ABOVE 900000000  (900 Mbps)
# FOR 2 minutes

# 3. Annotations:
# Summary: WAN saturation detected
# Description: WAN traffic > 900 Mbps for {{ $values.A }} seconds

# 4. Notification policy → Telegram
```

7.3 Alert — Suspicious Connections Spike

```
# Flux query — firewall connections per minute
from(bucket: "opnsense")
  |> range(start: -10m)
  |> filter(fn: (r) => r._measurement == "pf_state")
  |> filter(fn: (r) => r._field == "count")
  |> derivative(unit: 1m, nonNegative: true)
  |> mean()

# Condition: WHEN mean() IS ABOVE 5000 (connections/min)
# This threshold must be calibrated to your normal traffic
# Observe for 1 week before setting thresholds
```

8. Integrate Suricata and AdGuard Home into Grafana

8.1 Suricata Logs via Loki (Recommended)

```
# Install Grafana Loki for log aggregation (complement to InfluxDB)
# Add to docker-compose.yml:

  loki:
    image: grafana/loki:latest
    container_name: loki
    restart: unless-stopped
    ports:
      - "3100:3100"
    volumes:
      - ./loki:/loki

  promtail:
    image: grafana/promtail:latest
    container_name: promtail
    restart: unless-stopped
    volumes:
      - /var/log:/var/log:ro
      - ./promtail-config.yml:/etc/promtail/config.yml

# In OPNsense: configure Suricata to write EVE logs as JSON
# Intrusion Detection > Administration > Log to syslog: ON
# Format: EVE JSON

# In Proxmox: forward OPNsense syslog to Promtail
# /etc/promtail-config.yml:
scrape_configs:
  - job_name: suricata
    static_configs:
      - targets: [localhost]
        labels:
          job: suricata
          __path__: /var/log/suricata/*.json
```

8.2 Suricata Dashboard in Grafana

```
# Grafana → Dashboards → Import
# Community dashboard: "Suricata EVE Logs" (ID: 14972)
# Data Source: Loki

# Useful panels:
# - Top 10 triggered rules
# - Alerts by severity (HIGH/MEDIUM/LOW)
# - Attack timeline
# - Geo map of source IPs (WorldMap plugin)
# - Flow: src_ip → dst_ip → proto
```

8.3 AdGuard Home Metrics via REST API

```
# AdGuard Home exposes stats via REST API
# Integration with Telegraf:

[[inputs.http]]
  urls = ["http://192.168.1.x:3000/control/stats"]
  method = "GET"
  username = "admin"
  password = "YOUR-PASSWORD"
  data_format = "json"
  name_suffix = "_adguard"
  [inputs.http.tags]
    source = "adguard-home"

# Collected metrics:
# dns_queries, blocked_filters, blocked_safebrowsing
# replaced_safesearch, blocked_parental
# avg_processing_time

# In Grafana → Panel → Query InfluxDB:
from(bucket: "opnsense")
  |> range(start: -24h)
  |> filter(fn: (r) => r._measurement == "http_adguard")
  |> filter(fn: (r) => r._field == "dns_queries" or r._field == "blocked_filters")
```

With Suricata (episode 7) and AdGuard Home (episode 8) integrated into Grafana, you have a unified security view: intrusion detection + DNS filtering, all in one dashboard.

9. Conclusion: Full Series Recap

We've reached the end of 9 episodes building a complete enterprise-grade network infrastructure on OPNsense + Proxmox. Here's what you have:

- **Episode 1 — OPNsense on Proxmox:** High-performance firewall VM, virtual network interfaces
- **Episode 2 — Zero-Trust VLANs:** Network segmentation: LAN, IoT, DMZ, Guest — full isolation
- **Episode 3 — WireGuard VPN + LTE SD-WAN:** Site-to-site VPN, remote workers, 4G/LTE failover
- **Episode 4 — WiFi + UniFi AP:** WiFi AP management, wireless VLAN isolation, captive portal
- **Episode 5 — CrowdSec + Fail2ban:** Collaborative protection, dynamic blocklists, IP banning
- **Episode 6 — NAC FreeRADIUS 802.1X:** Network authentication, certificates, identity-based access policy
- **Episode 7 — Suricata IDS/IPS + DPI:** Intrusion detection, inline prevention, deep packet inspection
- **Episode 8 — AdGuard Home + DoH:** DNS filtering, blocklists, DNS over HTTPS — privacy and security
- **Episode 9 — Grafana + InfluxDB:** Real-time monitoring, dashboards, alerting — total visibility

This stack gives you enterprise-grade network infrastructure that most $50K/year network-budget SMEs don't have — for under $500 in hardware and $0 in software licenses.

Next up: the OpenClaw Series — infrastructure automation, intelligent agents, self-healing networks. Stay tuned.

Full article: blog.botum.ca/opnsense-monitoring-grafana-influxdb-guide

Series hub: blog.botum.ca/opnsense-stack-securite-enterprise-proxmox

Website: www.botum.ca • contact@botum.ca