

Behavioral micro-graph

Live traffic → SPAN Dest (Zeek VM) - - > filebeat for live file transfer (also Zeek VM) → filebeat Dest (Logstash VM) - - > ETL (Extract Transform Load data) using Logstash config: live.conf - - > sends the data to neo4j database using HTTP Queries → neo4j database modelling the entire architecture based on the live traffic

This is an end-to-end system designed to do everything from raw packet capture on a single subnet to updating the graph database with information from zeek logs (specifically conn.log, dns.log, and http.log)

As of now this graph databases uses a “Connection as-edge” architecture, but it is really easy to change that by simply changing the “live.conf” file for Logstash config

The software used in this project is:

Docker

```
amir@ubuntu-server:~$ docker --version
Docker version 27.5.1, build 27.5.1-0ubuntu3~24.04.2
```

Zeek (used inside a docker container) -version 7.0.8

```
amir@ubuntu-server:~$ docker run --rm -it \
> --network host \
> --cap-add=NET_ADMIN \
> --cap-add=NET_RAW \
> zeek/zeek:lts \
> zeek --version
zeek version 7.0.8
```

Logstash for ubuntu version 7.10.0

```
amir@ubuntu-server:~$ /usr/share/logstash/bin/logstash --version
Using bundled JDK: /usr/share/logstash/jdk
logstash 7.10.0
```

Filebeat for ubuntu version 7.10.2

```

amir@ubuntu-server:~$ filebeat version
filebeat version 7.10.2 (amd64), libbeat 7.10.2 [aacf9ecd9c494aa0908f61fbca82c906b16562a8 built 2021-01-12 23:11:24 +0000 UTC]

```

Neo4j Community edition (linux) -version 2025.05.0

```

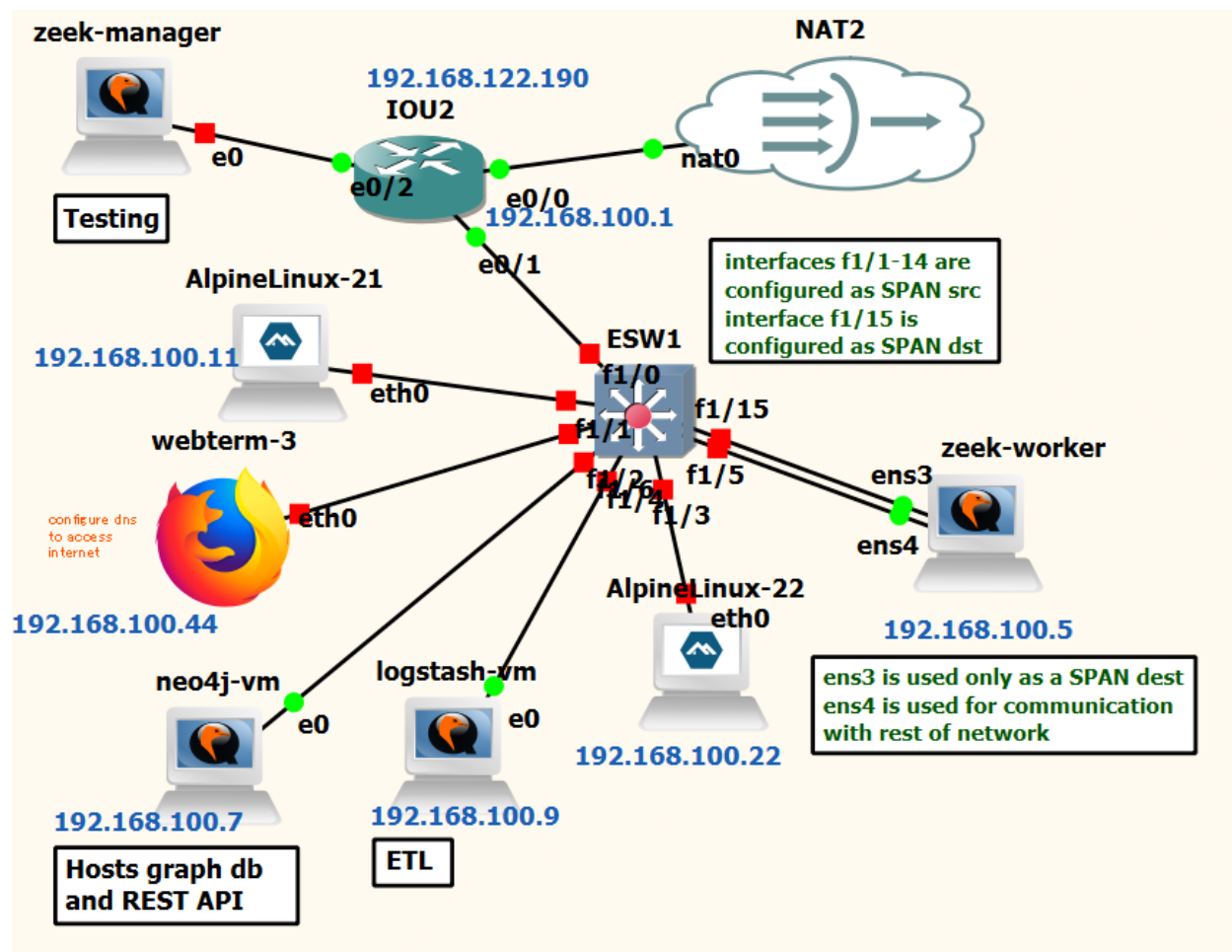
amir@ubuntu-server:~$ neo4j --version
2025.05.0

```

This document outline steps taken to create this pipeline and issues faced with solutions, as well open issues / next steps to be fixed

Setting up SPAN

In our topology:



The first thing was to setup a router connected to NAT – in order to have internet access.

This router gets connected to an EtherSwitch device (ESW1) – this is the only switch available in GNS3 that allows port mirroring (SPAN) capabilities. The image of the switch was found online: *c3725-adventerprisek9-mz.124-15.T14.image*

The switch was configured so that interfaces f1/1 – f1/14 are SPAN **sources** – meaning that all traffic coming **in and out** of these interfaces will get mirrored and sent to the SPAN **destination/s** – which in the case was only set up to be interface f1/15 – the connected to *zeek-worker*'s interface **ens3**.

So, in short, any traffic coming in and out of interfaces f1/1 to f1/14 will be **copied** and sent to the *zeek-worker* vm via interface **ens3**.

Zeek-worker was given another interface – ens4 – which is used for communication with any devices in the outside world.

The reason for this being that, once an interface is being used as a SPAN dest (in this case, ens3) it is unable to be used for any form of communication – the interface is practically idle, with its only purpose being to receive traffic as a SPAN dest. For this reason, a new interface (ens4) was added for external communication and the *zeek-worker* was assigned an ip address on that interface.

About IP Addresses

DHCP was disabled on all devices in this network and static ips were assigned to all the devices – this is mainly for ease of use. If DHCP is enabled, each device will have a different IP address each day, or even change within one day, which makes it hard to keep track of which device (which ip) is being used to run the zeek log captures, the logstash, and the neo4j database. The ips should remain static so that, when referenced in things like config files (such as live.conf – which references the IP address of the neo4j server), they are reliable and don't subject to change.

The Zeek VM

The zeek vm (*zeek-worker*) was already talked about in the SPAN section, but only in the context of it being the SPAN dest. This is logical because the zeek vm will be used to monitor the live network traffic (through the SPAN interface – ens3) and use this traffic to create log files (conn.log, dns.log, http.log, ssl.log, files.log, etc..)

Zeek was installed with the version mentioned in the beginning of this document and was ran inside of a docker container using the following command:

```
docker run --rm -it \
--network host \
--cap-add=NET_ADMIN \
--cap-add=NET_RAW \
-v ~/zeek-logs:/zeek \
zeek/zeek:its \
bash -c "cd /zeek && zeek -i ens3"
```

This command does the following (ChatGPT-generated explanation of the command in red):

◆ docker run --rm -it

run: Launches a new container

--rm: Automatically deletes the container after it exits (no leftovers)

-it: Interactive terminal so you can see and stop it with Ctrl+C

◆ --network host

Shares the host's network stack with the container

This allows Zeek inside the container to see actual traffic on your host's interfaces like ens3

◆ --cap-add=NET_ADMIN --cap-add=NET_RAW

Grants the container low-level network privileges required for packet capture (like tcpdump or pcap-based tools)

These are not granted by default for security reasons

◆ -v ~/zeek-logs:/zeek

Mounts your host directory ~/zeek-logs into the container at /zeek

So any logs written by Zeek inside the container are saved on your real system, persistently

◆ `zeek/zeek:its`

This is the official Zeek Docker image, preconfigured with everything needed

`:its` is the Long-Term Support version tag

◆ `bash -c "cd /zeek && zeek -i ens3"`

Runs a bash shell inside the container that:

Changes to the `/zeek` directory (your mounted log folder)

Starts Zeek on interface `ens3`

Zeek writes logs directly into `/zeek` (which is `~/zeek-logs` on your host)*

It launches zeek to listen on `ens3` and saves logs inside `~/zeek-logs/`

Filebeat for live log transfer

Filebeat is a software downloaded on the Zeek vm, used to transfer the logs created by zeek, in real time, to the logstash vm. As of now, filebeat is only configured to transfer the following logs: `conn.log`, `dns.log`, and `http.log`, but it can easily be configured to include more of the zeek logs.

The config file (`filebeat.yml`) is described as follows:

the config file is located in `/etc/filebeat/filebeat.yml`

It was configured as follows:

The filebeat Input section looks like this:

```
# ===== Filebeat inputs =====

filebeat.inputs:

# Each - is an input. Most options can be set at the input level
# you can use different inputs for various configurations
# Below are the input specific configurations.

- type: log

  enabled: true

  paths:
    - /home/amir/zeek-logs/conn.log

  fields:
    log_type: conn

- type: log

  enabled: true

  paths:
    - /home/amir/zeek-logs/dns.log

  fields:
    log_type: dns

- type: log

  enabled: true

  paths:
    - /home/amir/zeek-logs/http.log

  fields:
    log_type: http
```

And the Elasticsearch Output section was modified to look like this:

```
# ----- Elasticsearch Output -----  
output.logstash:  
  # Array of hosts to connect to.  
  hosts: ["192.168.100.9:5044"]  
  
  # Protocol - either `http` (default) or `https`.  
  #protocol: "https"  
  
  # Authentication credentials - either API key or username/password.  
  #api_key: "id:api_key"  
  #username: "elastic"  
  #password: "changeme"
```

These were the only changes made to the filebeat.yml file. Everything else was kept the same.

To run filebeat use the following commands:

```
sudo systemctl enable filebeat
```

```
sudo systemctl start filebeat
```

and to check the status use:

```
sudo systemctl status filebeat
```

...

Now, you may have noticed that in the output section I included an IP address and a port – namely ip 192.168.100.9 and port 5044. This is mapped to the ip address of the logstash server and, as for the port, it still needs to be set up on the logstash server side in order to get the pipeline to work. So, let's go to the logstash server and talk about how that was configured.

One thing to note – for debugging purposes, to see the logs generated by filebeat, use the following command inside the zeek vm (the vm also running filebeat):

```
sudo journalctl -u filebeat -f
```

Logstash server

The logstash server is used as a middle man between the zeek vm and the neo4j vm. It takes the zeek logs from the zeek vm in real time via filebeat and transforms these log files into cypher queries that are to be used via HTTP queries to populate the neo4j database. All of this is done using a **single** config file – which I’ve named live.conf.

Logstash was installed on the vm with the versio mentioned at the start of the document.

Here’s a step by step guide on installation (ChatGPT generated):

“logstash setup:

”Here’s a step-by-step guide to ****download and install Logstash version 7.10**** on Ubuntu.

 Step 1: Add the Elastic GPG Key

```bash

wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -

...

---

###  Step 2: Add the Elastic 7.x APT repository

Create the file:

```bash

sudo nano /etc/apt/sources.list.d/elastic-7.x.list

...

Add this line:

```
```bash
deb https://artifacts.elastic.co/packages/7.x/apt stable main
```
```

Save and exit (Ctrl+O, Enter, Ctrl+X)

 Step 3: Update APT cache

```
```bash
sudo apt update
```
```

 Step 4: Install **Logstash 7.10**

You must **explicitly specify the version**:

```
```bash
sudo apt install logstash=1:7.10.0-1
```
```

>  The version number includes `1:` as part of the Debian package versioning.

 Step 5: Pin the version (optional but recommended)

This prevents automatic upgrades:

```
```bash
sudo apt-mark hold logstash
...

```

###  Step 6: Start and enable Logstash

```
```bash
sudo systemctl enable logstash
sudo systemctl start logstash
...

```

To check if it's running:

```
```bash
sudo systemctl status logstash
...

```

---

”

To run logstash using a config file (say, example.conf), the following command is used:

```
sudo /usr/share/logstash/bin/logstash -f example.conf
```

Our config file for this setup is called live.conf

The full config file is attached to this document.

## Neo4j Setup

Since the community version of neo4j only allows for one database, I created two databases by setting up neo4j on two different machines: the neo4j vm and the logstash vm. The database on the neo4j vm uses a connection as-node architecture while the database on the logstash vm uses an as-edge architecture.

For both databases, the access for the database is as follows:

[REDACTED]

[REDACTED]

To view the neo4j database using a web-based platform you can use the webterm vm. now if you go on webterm and go to <http://192.168.100.9:7474> (or <http://192.168.100.7:7474> for the as-node architecture) then you can access the neo4j database.

(Note about the webterm VM – you must manually set up a DNS server every time you open it by going to `/etc/resolv.conf` and writing this “nameserver 8.8.8.8” – this is only if you want internet access)