Full-Stack Secure Network & Application Infrastructure with SIEM Monitoring

By: Jared Lowe

F25 Senior Capstone

August 13th - December 5th

TOC

1. Project Overview

This final report documents my final status for the semester-long project. The overall goal of this project is to design and implement a secure, enterprise-like IT environment that mirrors the operational complexity and challenges of a small business network. The project includes deploying servers, databases, and a web application, surrounded by layers of network security, monitoring, and logging, to simulate what a modern IT or cybersecurity professional might encounter in the field. Over the course of the build, each system was configured, secured, and integrated into the SIEM for real-time monitoring and analysis. After I will evaluate the findings that I found and do/state what I would do to keep it a secure enterprise network.

This project allows me to integrate several disciplines: networking, systems administration, cybersecurity, and web development into a single, complex, and unified environment. By the end of the semester, I have learned and built a functioning network system capable of not only hosting services securely but also detecting and monitoring potential security events through a Security Information and Event Management (SIEM) system.

This report documents every component of the environment, including system configurations, firewall routing, application deployment, and SIEM integration. Each screenshot included contains a clear description and explains the impact on the overall project.

This report focuses on the entirety of the project, which includes:

- Setting up the virtual infrastructure and network topology
- Deploying and configuring the pfSense firewall
- Installing and hardening Linux and Windows virtual machines
- Building and securing a PostgreSQL database
- Developing and hosting a Ruby on Rails web application
- Enabling basic application-level logging for future SIEM monitoring

2. Project Purpose

The purpose is to simulate a real-world enterprise IT infrastructure that aims to represent how a modern organization might create a combination of web applications, databases, and firewalls under the supervision of a centralized monitoring system:

- Securely segmented — isolating internal networks from external threats
- Application-driven — featuring a working, database-backed web app
- Monitoring-ready — structured to integrate with a SIEM for visibility and alerting

By completing this build, I'm gaining not only technical experience but also the ability to think like both a system administrator. SOC analyst, and a cybersecurity analyst.

3. Objectives

1. Implemented network segmentation, routing, and security policies using pfSense.
2. Deployed and configured a Linux-based web application and PostgreSQL database.
3. Built a functional monitoring environment capable of SIEM-level logging, analysis, and alerting.
4. Simulated a realistic enterprise structure by integrating servers, workstations, and network appliances across multiple isolated VLANs and firewall zones.
5. Developed hands-on proficiency in system administration, networking, Linux hardening, and secure configuration practices.
6. Applied cybersecurity principles through controlled offensive testing and log correlation.
7. Documented the entire implementation process with clear, structured, and professional-level detail.

4. Scope of Work

This project focused on building the infrastructure, including network segmentation, firewall architecture, virtual machines, database systems, logging pipelines, and security monitoring. After validating system connectivity and log ingestion, the environment was expanded to include Pi-hole, Filebeat agents, and staged cyberattacks to test SIEM visibility.

The table below outlines each major task and its completion status:

Steps	Time	Completed
VMWare virtual networks + machines	Week 1	Yes
pfSense firewall setup	Week 2	Yes
Linux & Windows VM setup	Week 3	Yes
PostgreSQL database creation	Week 3-5	Yes
Ruby on Rails app deployment	Week 5-7	Yes
Siem configuration and alerting	Week 7-8	yes
PiHole	Week 8-9	Yes
Cyber attacks and logging	Week 9-13	Yes
Constructing stronger networking after attacks	Week 12-14	Yes
Documentation and Final	Week 14	Yes

5. Costs and Risks

I have not and will not buy any software used or code to reach this point in my project. I have been loaned an SSD by Dr.Wimmer for my project. All resources and software were found for free online.
The overall risk exposure is low. Most technical risks are mitigated through documentation, version control, and the use of VMware snapshots.Since I also used backups everyday I did not run into any problems.

6. Issues

Throughout the project, I faced several challenges, primarily related to network connectivity and firewall configuration. Early on, devices were unable to communicate properly because pfSense rules or interfaces were misconfigured. Troubleshooting these issues involved trial, error, and learning, mostly in understanding how VMware's virtual networks communicate with pfSense.

I received helpful guidance from Dr. Wimmer, who walked me through several key troubleshooting steps and explained underlying networking principles. While these setbacks delayed progress slightly, they significantly deepened my understanding of both networking fundamentals and pfSense's architecture.

Now, all systems communicate cleanly, firewall rules are properly segmented, and the environment runs smoothly. These early difficulties turned into valuable learning experiences that strengthened my troubleshooting skills and confidence with networked environments.

With Wimmers help I got the systems online and communicated and the rest of the process went smoothly as I kept regular backups and snapshots.

7. Lessons Learned

This project provided significant hands-on experience in both technical execution and structured project management. Key lessons learned include:

- Documentation is essential. Maintaining detailed notes on IP addresses, passwords, keys, firewall rules, and system configurations prevented confusion and helped quickly resolve issues especially during pfSense and filebeat troubleshooting.

- Planning network segmentation early avoids later problems. The importance of designing VLANs, interfaces, and firewall policies before deployment became clear once systems began interacting across different networks. Static Ips for pfsense worked faster.

- Infrastructure tasks are interconnected. Fixing one issue often required understanding several layers: routing, DNS, firewalls, OS configuration, and log forwarding mirroring real enterprise environments. Doing one task meant checking the other machines and pfsense.

- Cybersecurity visibility matters. The SIEM component reinforced how logs, alerts, and detections form the first line of defense for SOC analysts.

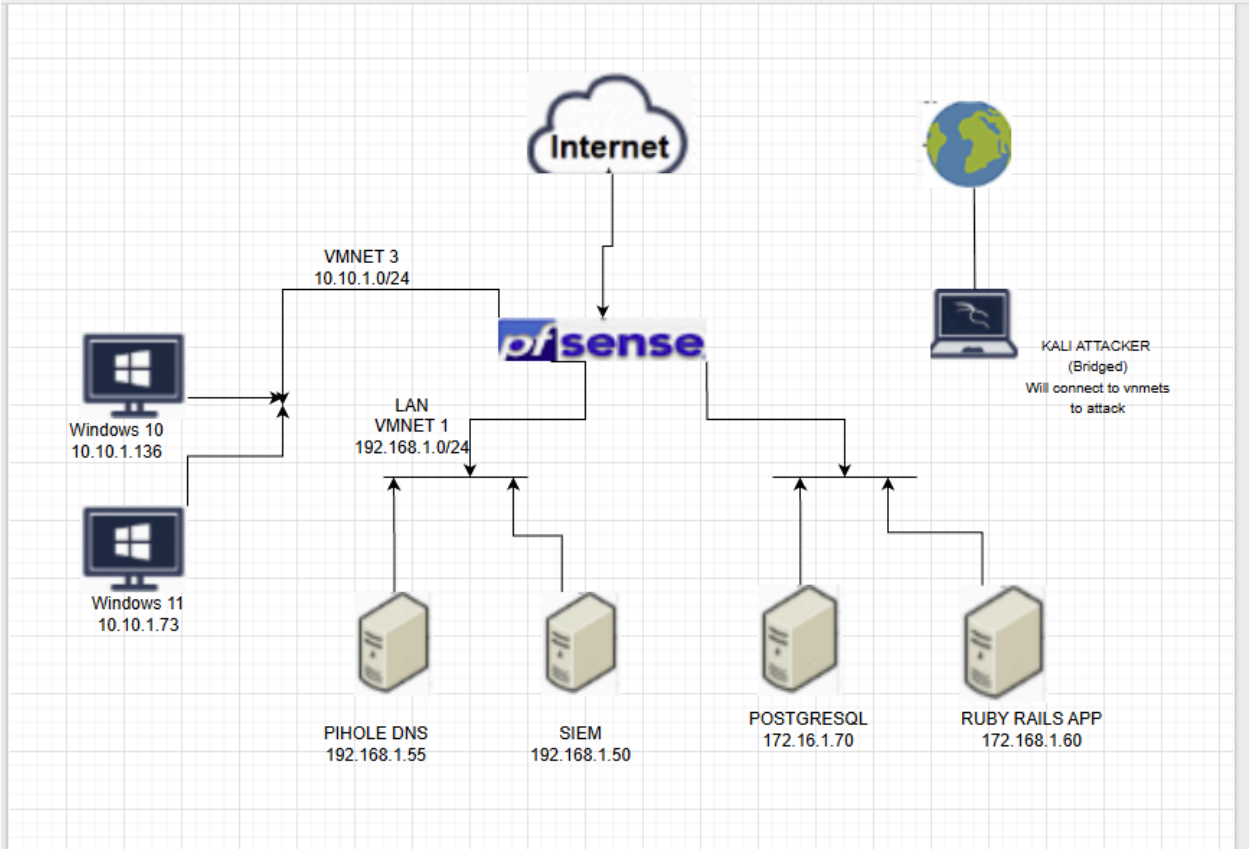
- Hands-on testing builds stronger understanding than theory in class. Running attacks from Kali and observing their traces in Elasticsearch provided real-world insight into how intrusion attempts appear from a defender’s perspective.

Overall, this project significantly enhanced my technical confidence and prepared me for real-world roles in networking, systems administration, and cybersecurity operations.

8. Implementation

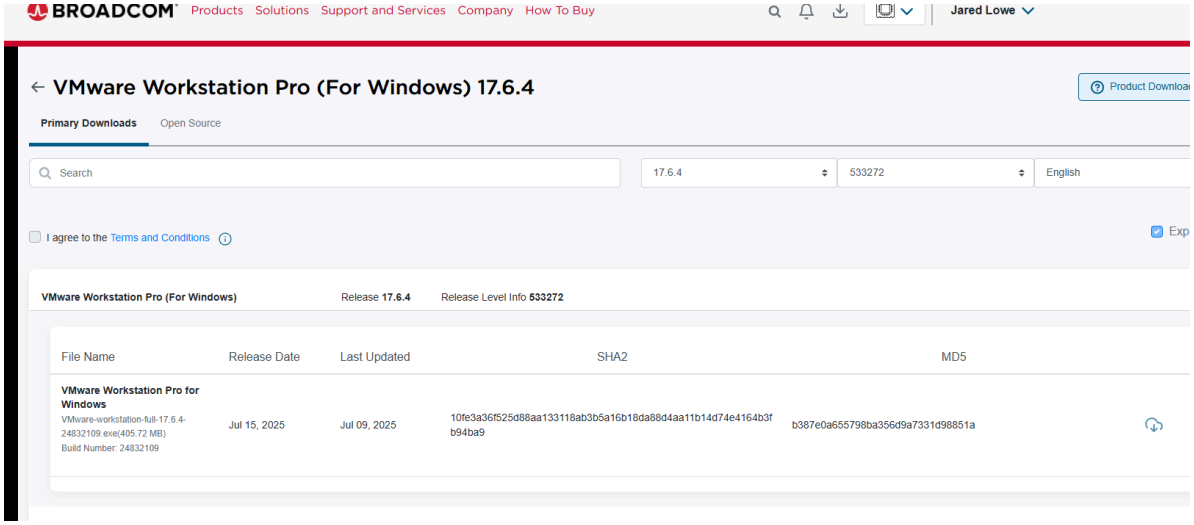
1. Network Architecture and Topology

2. VMnet	3. Subnet	4. Purpose
5. VMnet8	6. NAT / Bridged	7. Internet access (WAN)
8. VMnet1	9. 192.168.10.0/24	10. Internal LAN & Management
11. VMnet3	12. 10.10.1.0/24	13. Workers sim
14. VMnet4	15. 172.16.1.0/24	16. Application services

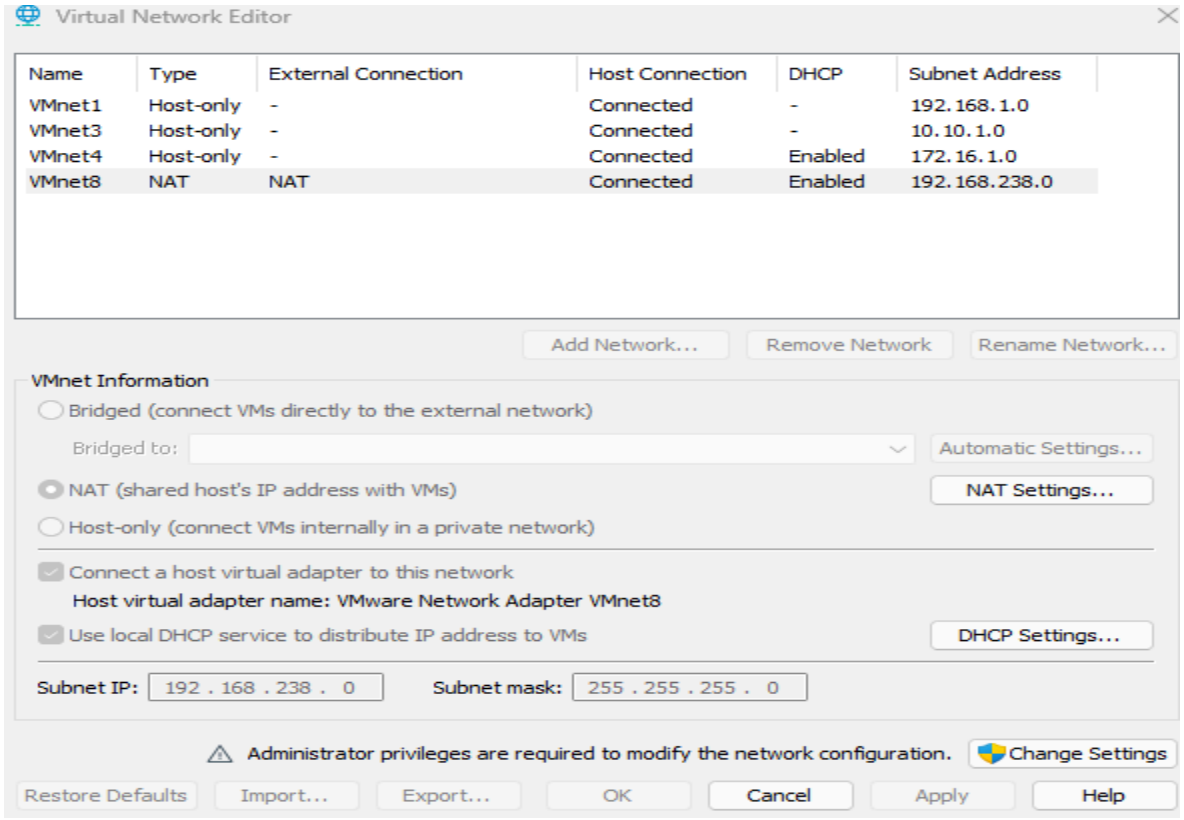


2. Created a VMware virtual network:

Here I set up vmware workstation pro to utilize all vms and use the virtual network editor.



Here is where I used the network editor to configure our network map to distribute different networks to each vm.



3. Created the remaining VMs and attached them to the correct VMnets. Also, power on VMs and perform initial OS installs/updates on each VM where i set the static ips that correlated with the range.

4.pfsense→ attached all VMNETSs to it

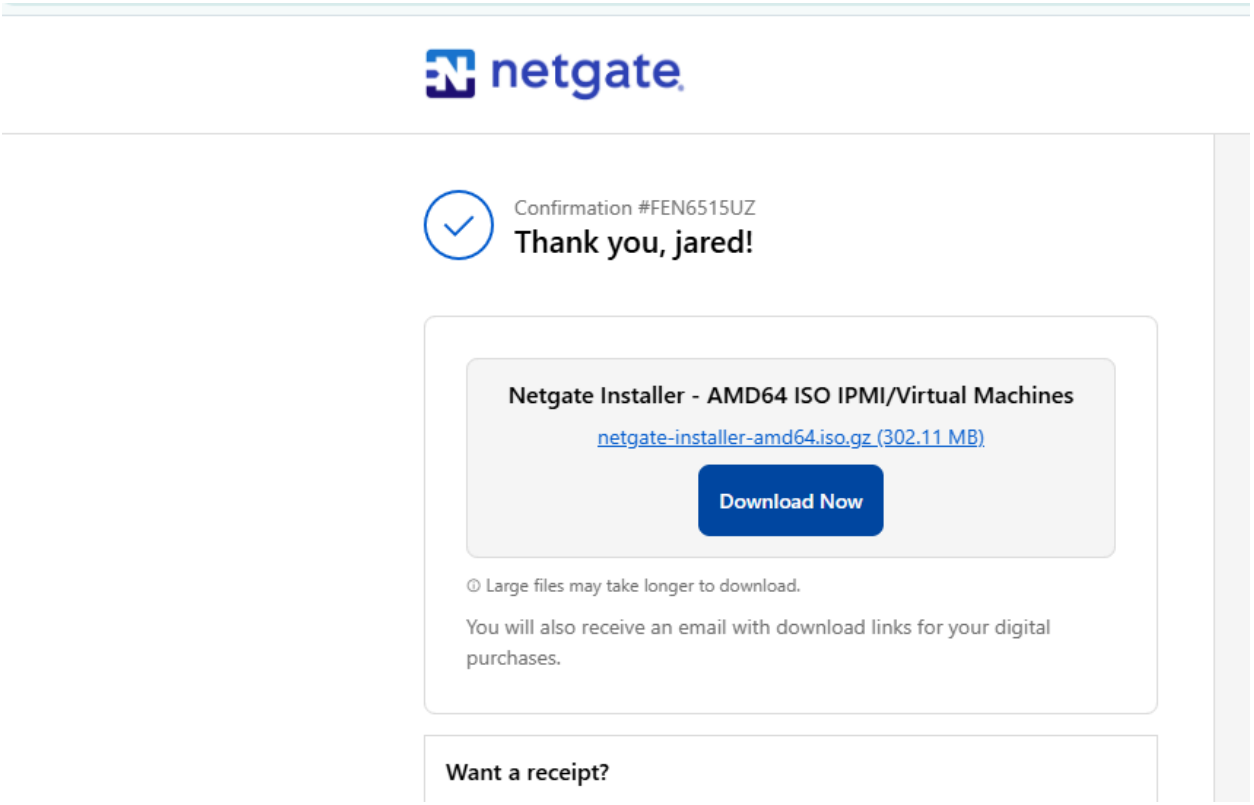


figure 4.1

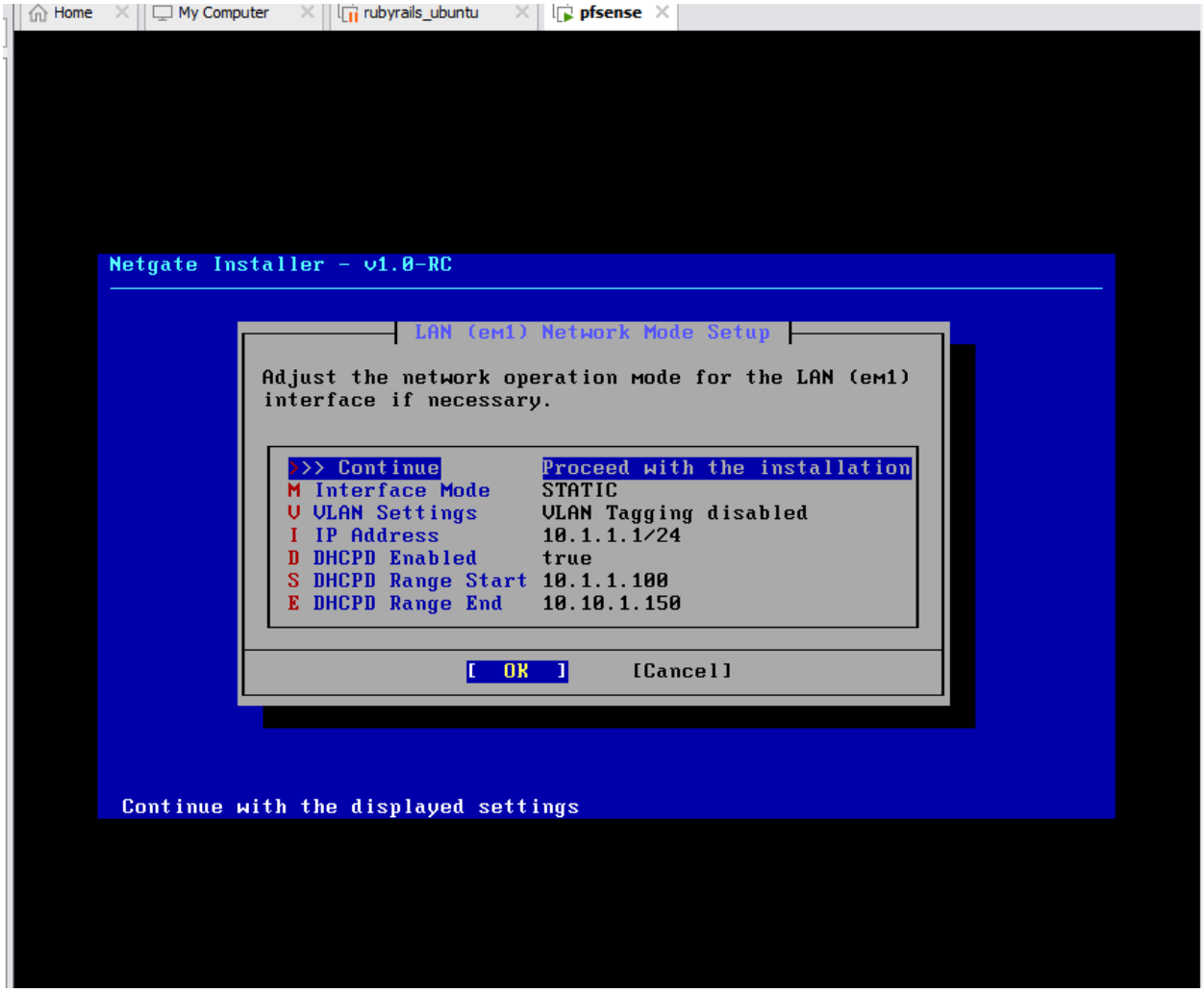


figure 4.2

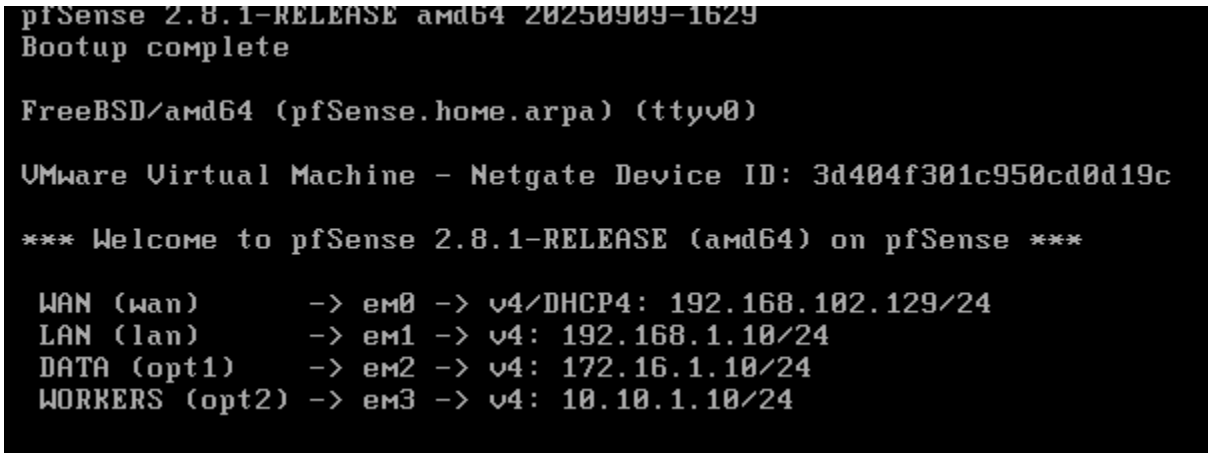


figure 4.3

5.ubuntu-ruby-rails → vmnet4

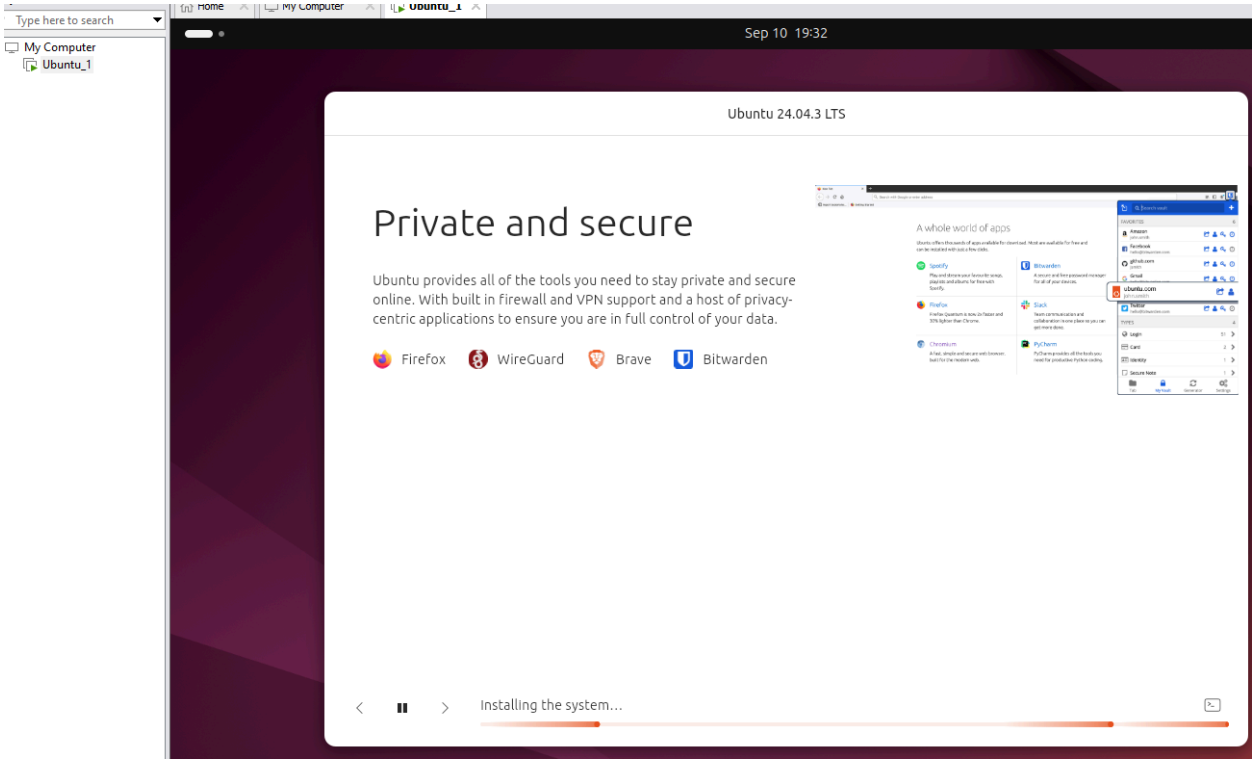


figure 5.1

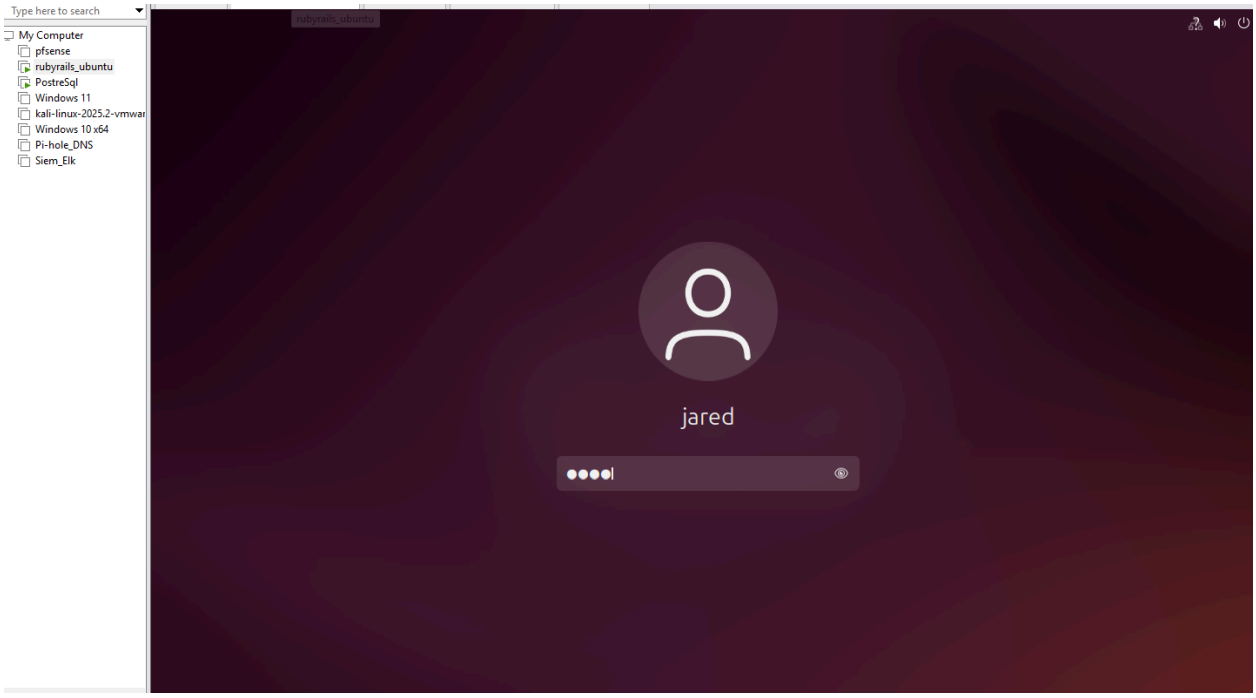


figure 5.2

6. ubuntu-postgresql → vmnet4

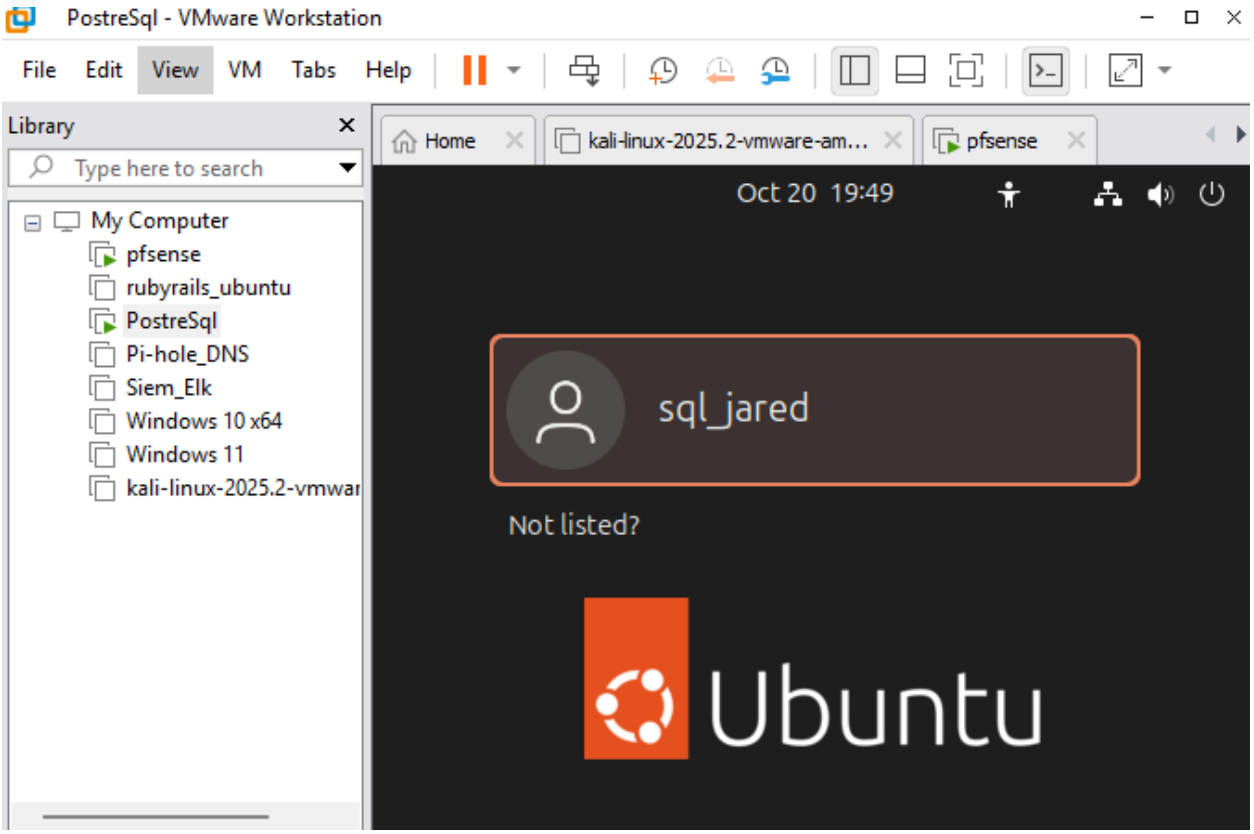


figure 6.1

7.ubuntu-siem-elk →vmnet1

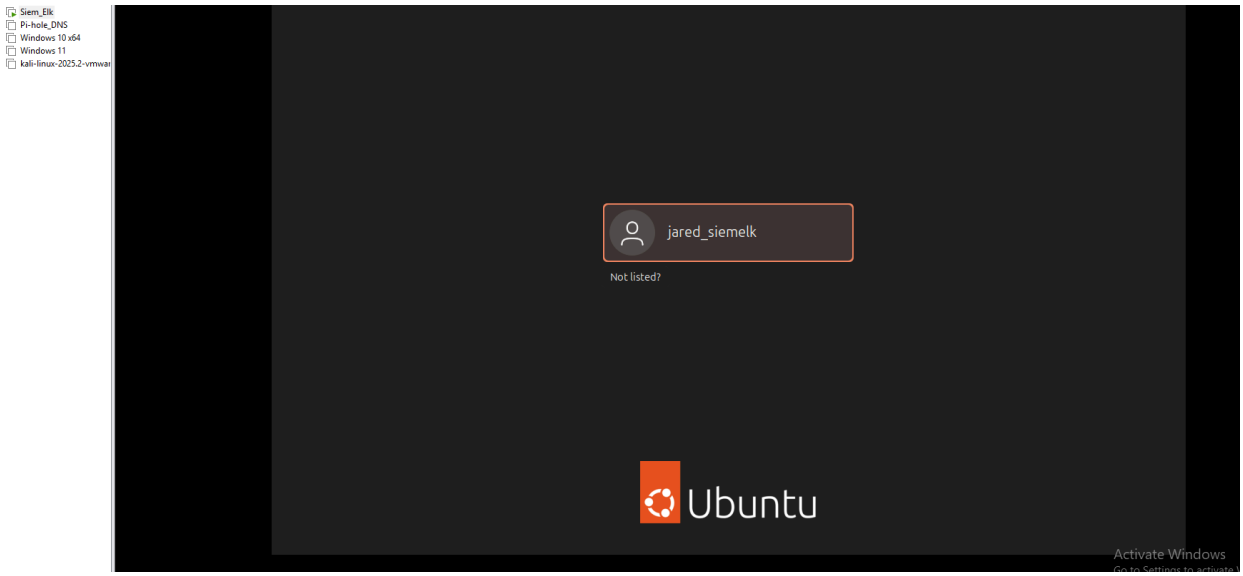


figure 7.1

8. Windows 11 → vmnet3

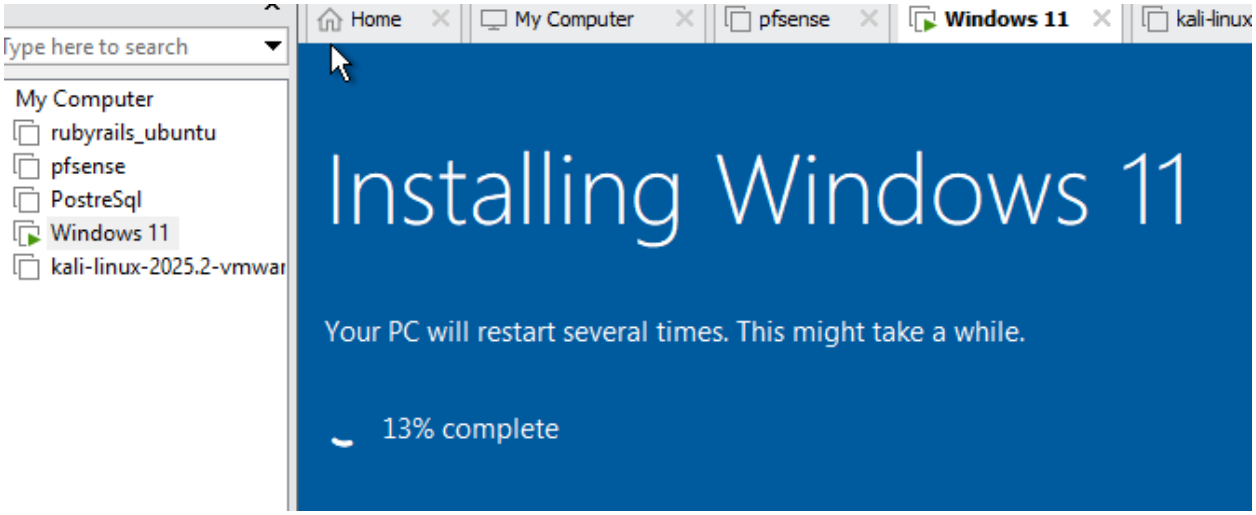


figure 8.1

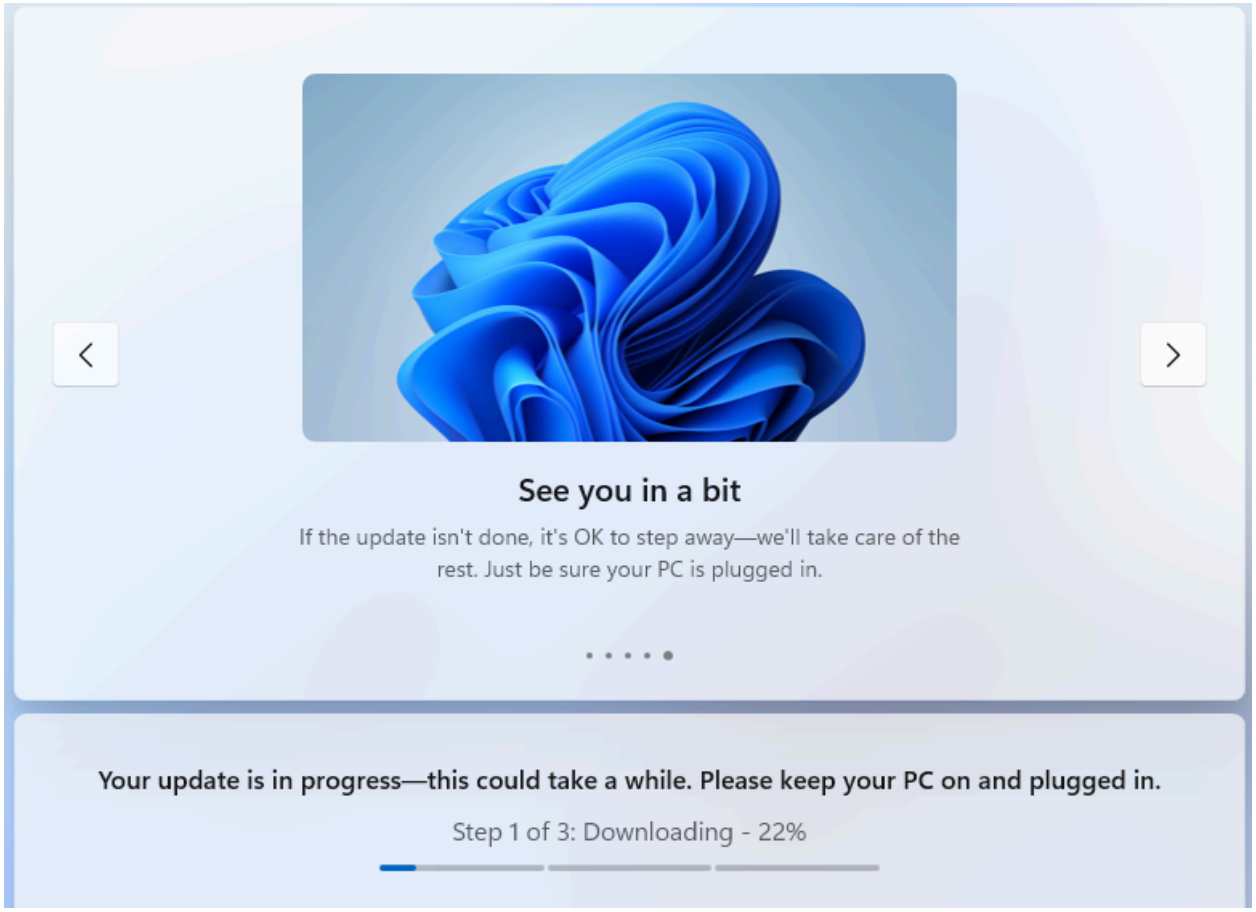


figure 8.2

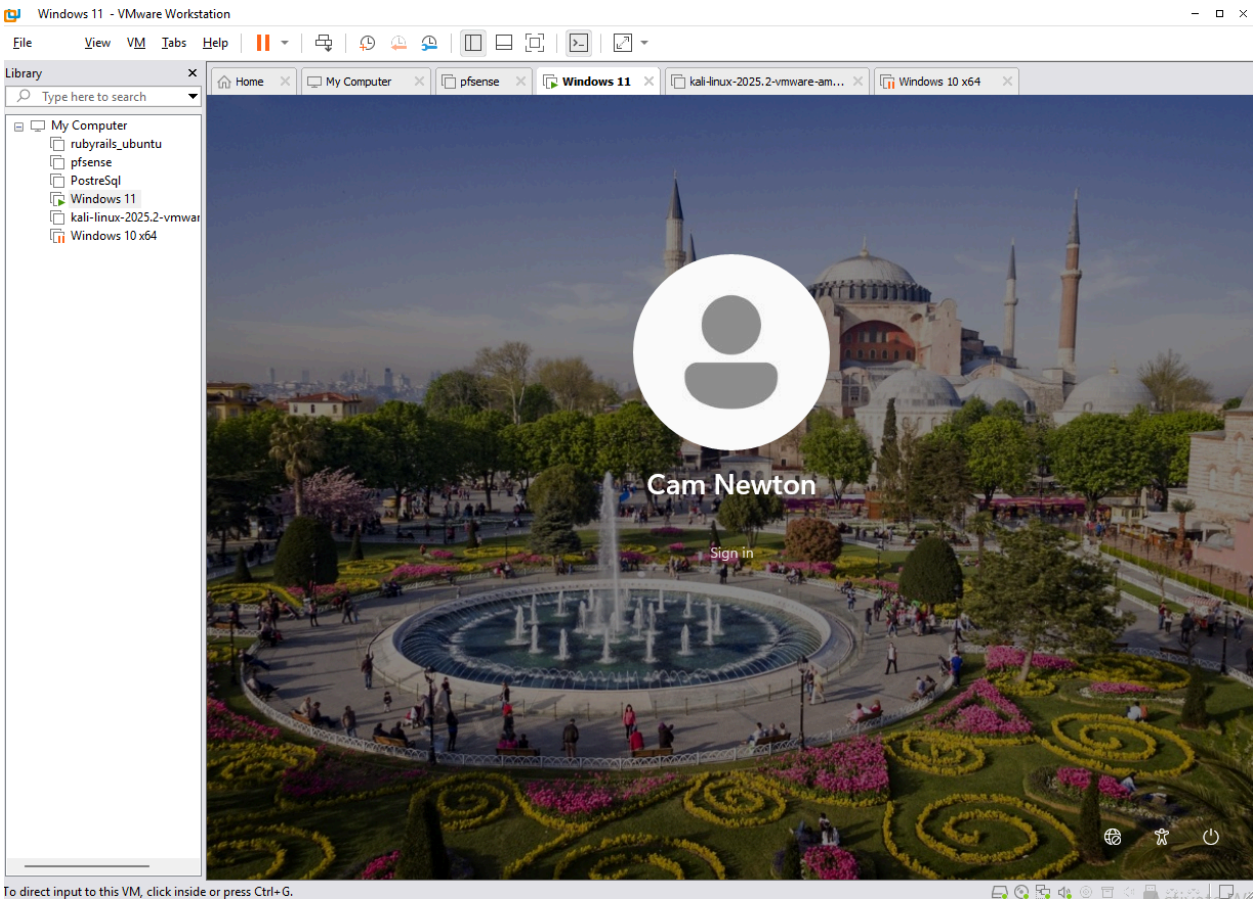


figure 8.3

9.windows10 → vmnet3

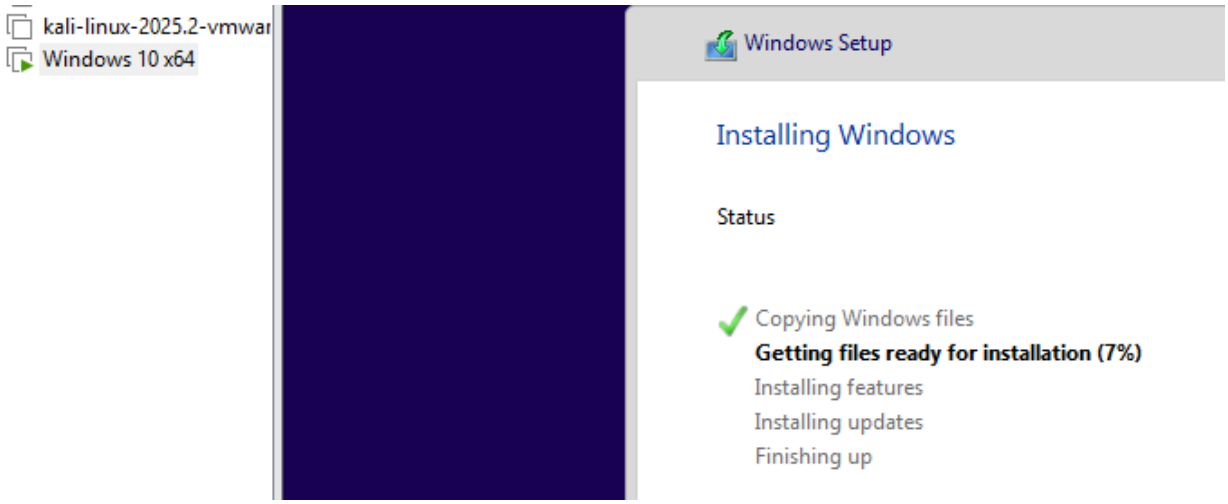


figure 9.1

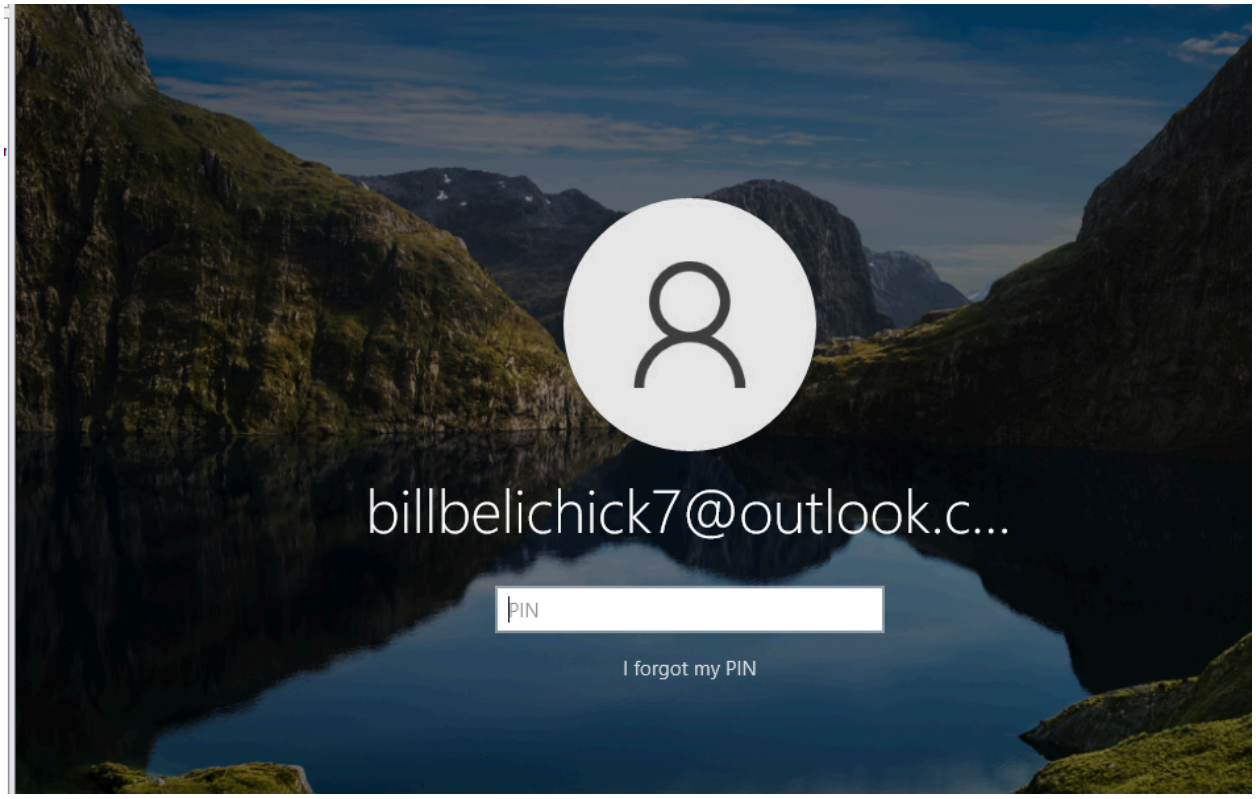


figure 9.2

10. kali-linux-attacker →bridged to outside(will connect to other vmnets after set up

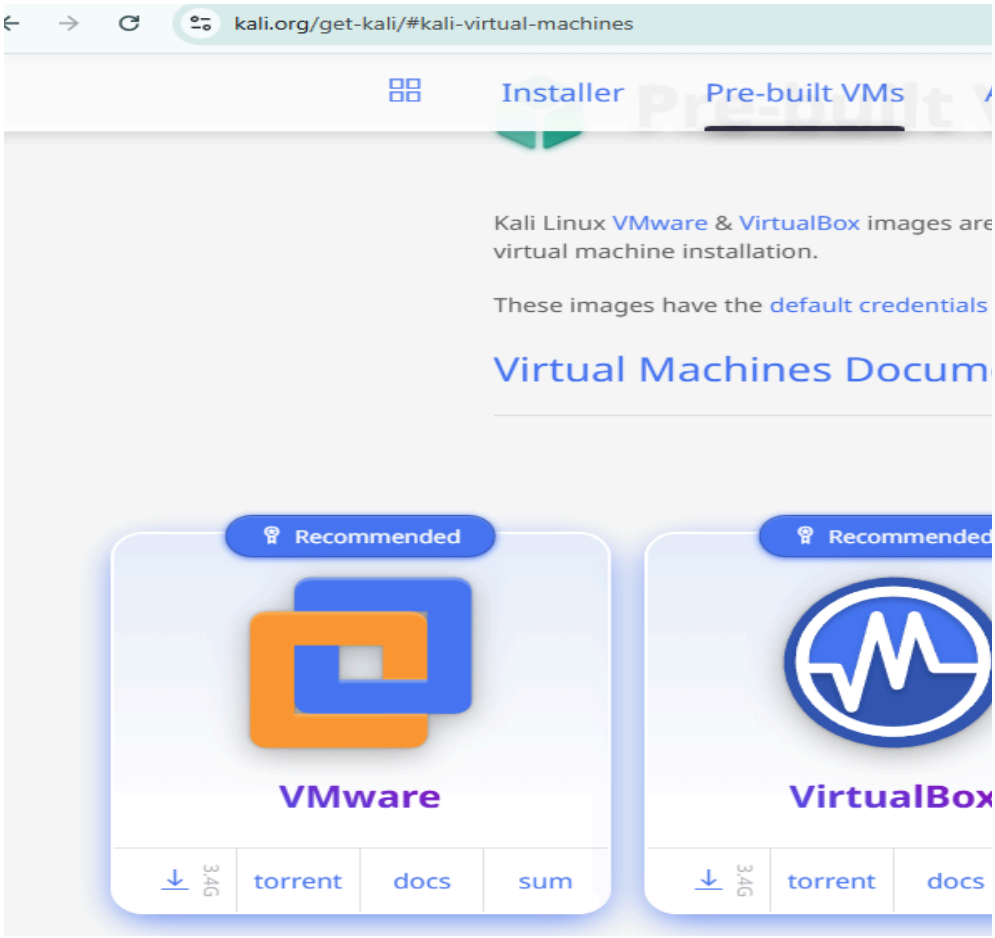

kali.org/get-kali/#kali-virtual-machines
Installer
Pre-built VMs
Kali Linux VMware & VirtualBox images are
virtual machine installation.
These images have the default credentials
Virtual Machines Docum
Recommended
VMware
torrent
docs
sum
Recommended
VirtualBox
torrent
docs

figure 10.1

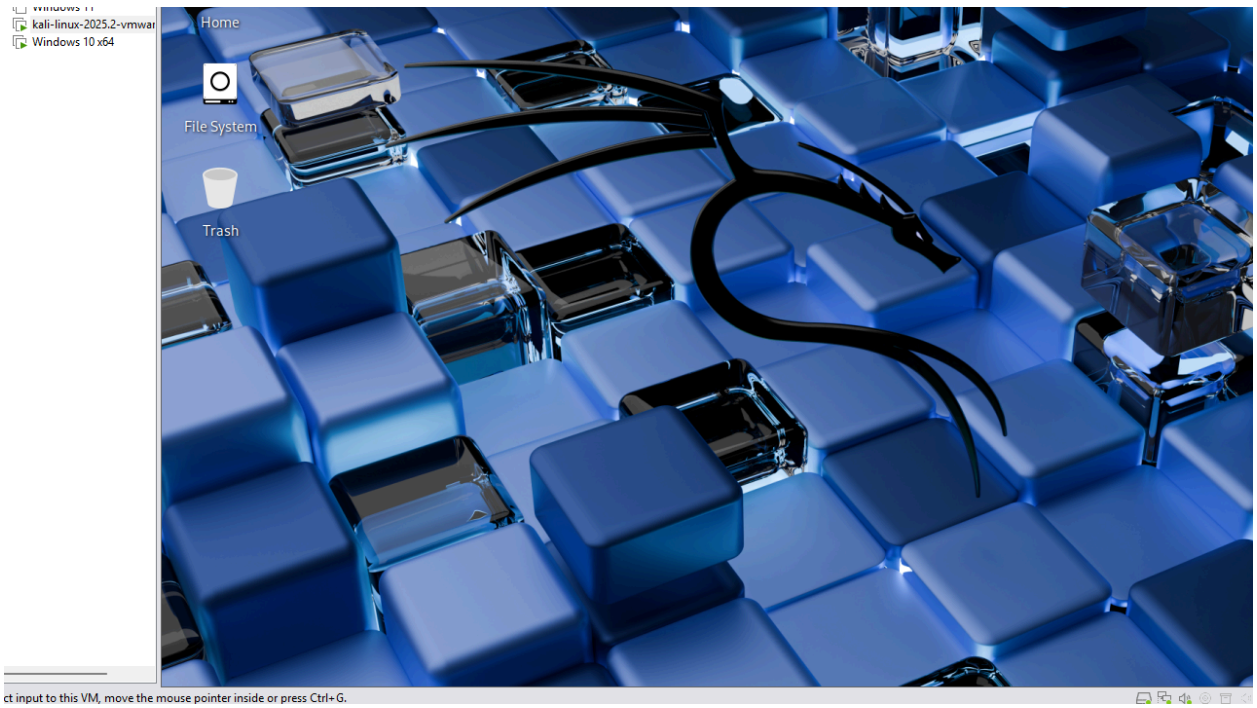

kali-linux-2025.2-vmwar
Windows 10 x64
Home
File System
Trash
ct input to this VM, move the mouse pointer inside or press Ctrl+G.

figure 10.2

11. Boot pfSense, assign interfaces, and verify pfSense WebGUI access

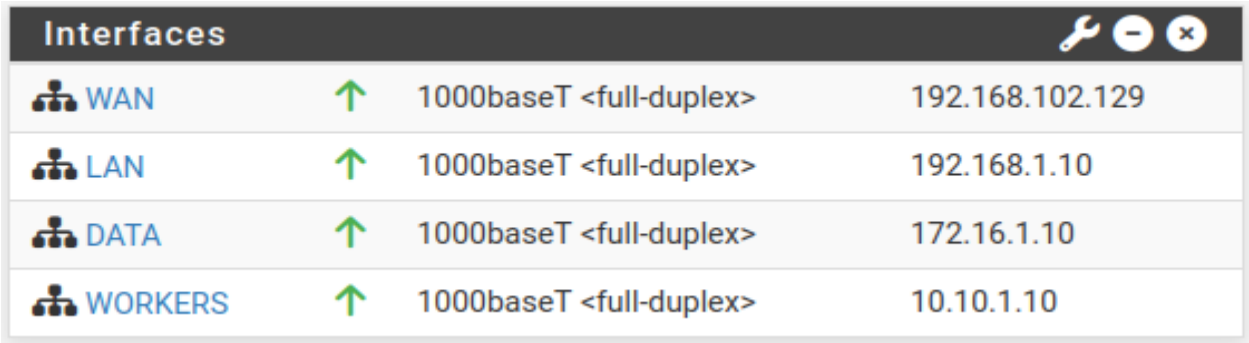


figure 11.1

12. Configure pfSense firewall rules with first being the NAT outbound rules so it can access internet:

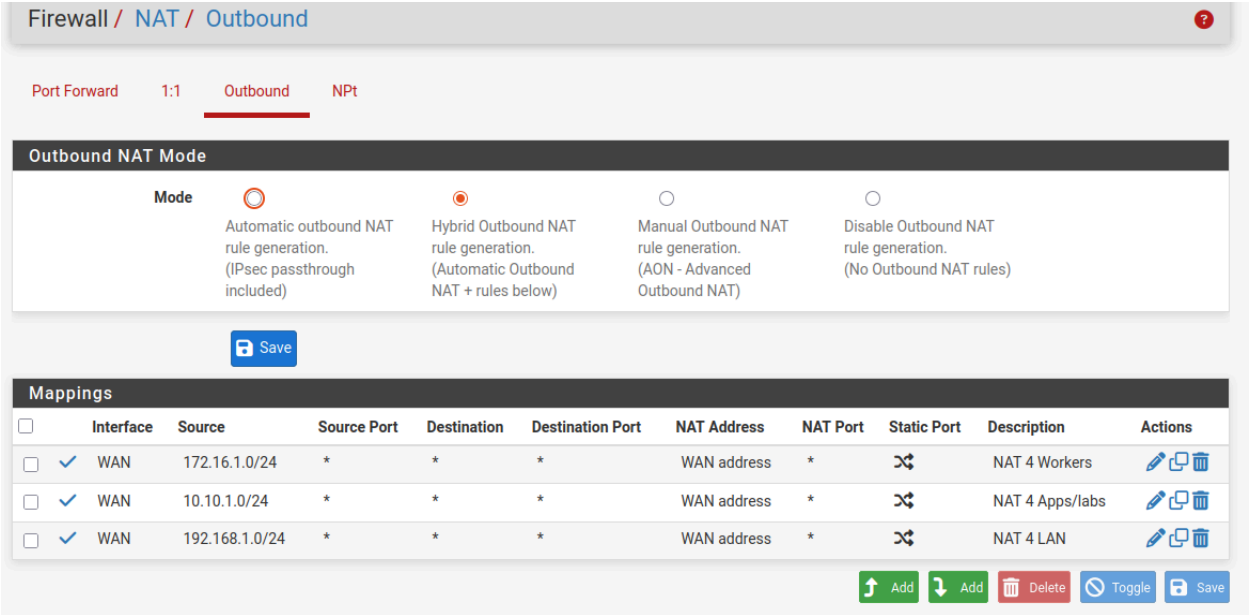


Figure 12.1

13. Validate IP addressing and reachability: each VM can ping its gateway and pfSense LAN IP with firewall access. Doing this with firewall rules.

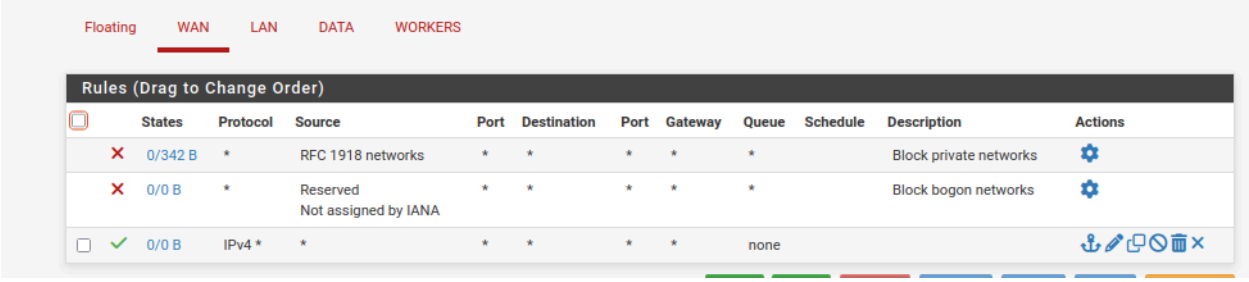
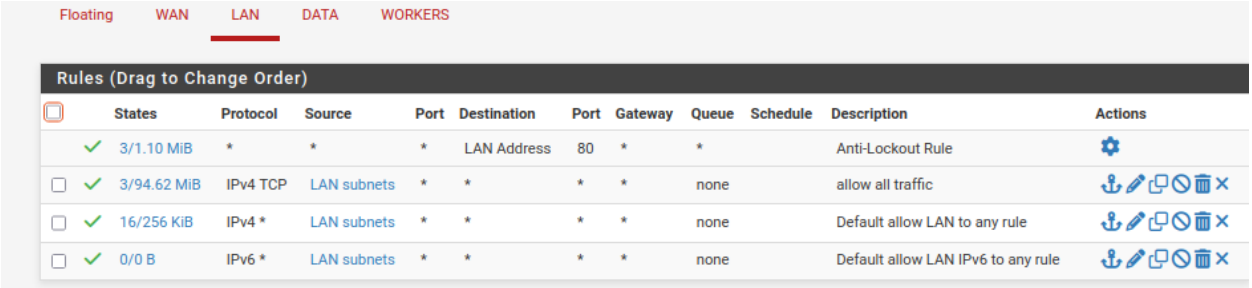
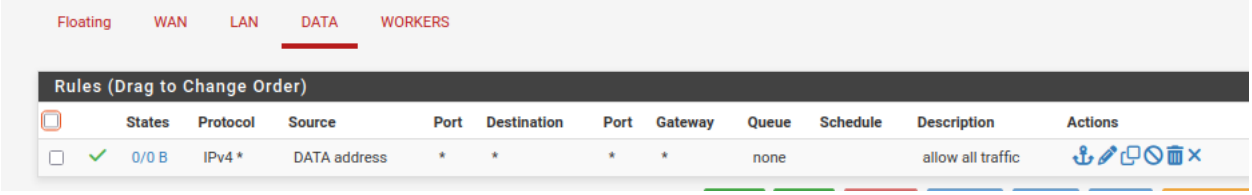


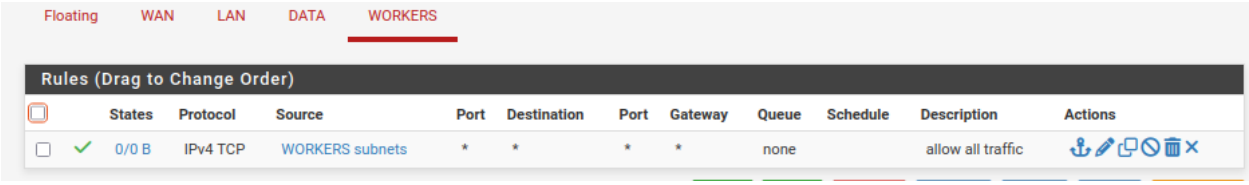
Figure 13.1



, figure 13.2



, figure 13.3



, figure 13.4

14. Install PostgreSQL on ubuntu-postgresql and create the application database and user. With these commands:

Running sudo apt update

sudo apt install -y build-essential libssl-dev zlib1g-dev libreadline-dev \

libyaml-dev libxml2-dev libxslt1-dev libcurl4-openssl-dev libffi-dev libncurses5-dev libgdbm-dev libdb-dev uuid-dev

Here, I enabled and started my PostgreSQL.

```
sql-jared@sql-jared-VMware-Virtual-Platform:~$ sudo systemctl enable postgresql
[sudo] password for sql-jared:
Synchronizing state of postgresql.service with SysV service script with /usr/lib
/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable postgresql
sql-jared@sql-jared-VMware-Virtual-Platform:~$ sudo systemctl start postgresql
sql-jared@sql-jared-VMware-Virtual-Platform:~$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled; prese>
     Active: active (exited) since Tue 2025-10-21 19:05:23 EDT; 3min 45s ago
   Main PID: 1973 (code=exited, status=0/SUCCESS)
        CPU: 1ms

Oct 21 19:05:23 sql-jared-VMware-Virtual-Platform systemd[1]: Starting postgres>
Oct 21 19:05:23 sql-jared-VMware-Virtual-Platform systemd[1]: Finished postgres>
lines 1-8/8 (END)
```

figure 14.1

15. Now I entered my psql and set my databases, roles

CREATE DATABASE myapp_development;

CREATE USER myapp_user WITH PASSWORD 'pass';

ALTER ROLE myapp_user SET client_encoding TO 'utf8';

ALTER ROLE myapp_user SET default_transaction_isolation TO 'read committed';

ALTER ROLE myapp_user SET timezone TO 'UTC';

GRANT ALL PRIVILEGES ON DATABASE myapp_development TO myapp_user;

GRANT ALL ON SCHEMA public TO testuser;

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO testuser;

```
postgres@sql-jared-VMware-Virtual-Platform:~$ psql
psql (16.10 (Ubuntu 16.10-0ubuntu0.24.04.1))
Type "help" for help.

postgres=# \du
                             List of roles
 Role name  |                         Attributes
------------+------------------------------------------------------------
 jared      | Create DB
 myapp_user |
 postgres   | Superuser, Create role, Create DB, Replication, Bypass RLS
 testuser   | Create DB

postgres=#
```

figure 15.1

16. Install Ruby, Rails, and app prerequisites on ubuntu-ruby-rails; deploying the Rails app.

```
jared@jared-VMware-Virtual-Platform:~$ ruby -v
ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-linux]
jared@jared-VMware-Virtual-Platform:~$ bundler -v
Bundler version 2.4.10
jared@jared-VMware-Virtual-Platform:~$
```

figure 16.1

17. Configure Rails database.yml to point to ubuntu-postgresql and run migrations. I also tested to see if my Ruby Rails application was working.

cd ~testapp

Nano config/database.yml

```
  GNU nano 7.2                                                conf
default: &default
  adapter: postgresql
  encoding: unicode
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
  username: testuser
  password: pass
  host: 172.16.1.70
  port: 5432

  timeout: 5000

development:
  <<: *default
  database: testdb
```

figure 17.1

```
test:
  <<: *default
  database: testdb_test


# Store production database in the storage/ directory, which by default
# is mounted as a persistent Docker volume in config/deploy.yml.
production:
  primary:
    <<: *default
    database: testdb_test
  cache:
    <<: *default
    database: storage/production_cache.sqlite3
    migrations_paths: db/cache_migrate
  queue:
    <<: *default
    database: storage/production_queue.sqlite3
    migrations_paths: db/queue_migrate
  cable:
    <<: *default
    database: storage/production_cable.sqlite3
    migrations_paths: db/cable_migrate
```

figure 17.2

```
jared@jared-VMware-Virtual-Platform:~$ cd ~/testapp
jared@jared-VMware-Virtual-Platform:~/testapp$ rails db:create
Database 'testdb' already exists
Database 'testdb_test' already exists
jared@jared-VMware-Virtual-Platform:~/testapp$ rails db:migrate
jared@jared-VMware-Virtual-Platform:~/testapp$ rails server -b 172.16.1.60
=> Booting Puma
=> Rails 8.0.3 application starting in development
=> Run `bin/rails server --help` for more startup options
Puma starting in single mode...
* Puma version: 7.0.4 ("Romantic Warrior")
* Ruby version: ruby 3.2.2 (2023-03-30 revision e51014f9c0) [x86_64-linux]
*  Min threads: 3
*  Max threads: 3
*  Environment: development
*          PID: 3003
* Listening on http://172.16.1.60:3000
Use Ctrl-C to stop
Started GET "/" for 172.16.1.60 at 2025-10-23 11:32:23 -0400
Cannot render console from 172.16.1.60! Allowed networks: 127.0.0.0/127.255.255.255, ::1
  ActiveRecord::SchemaMigration Load (0.8ms)  SELECT "schema_migrations"."version" FROM "schem
a_migrations" ORDER BY "schema_migrations"."version" ASC /*application='Testapp'*/
Processing by Rails::WelcomeController#index as HTML
  Rendering /home/jared/.rbenv/versions/3.2.2/lib/ruby/gems/3.2.0/gems/railties-8.0.3/lib/rail
s/templates/rails/welcome/index.html.erb
  Rendered /home/jared/.rbenv/versions/3.2.2/lib/ruby/gems/3.2.0/gems/railties-8.0.3/lib/rails
/templates/rails/welcome/index.html.erb (Duration: 0.6ms | GC: 0.0ms)
Completed 200 OK in 33ms (Views: 4.3ms | ActiveRecord: 0.0ms (0 queries, 0 cached) | GC: 0.8ms
)
```

figure 17.3

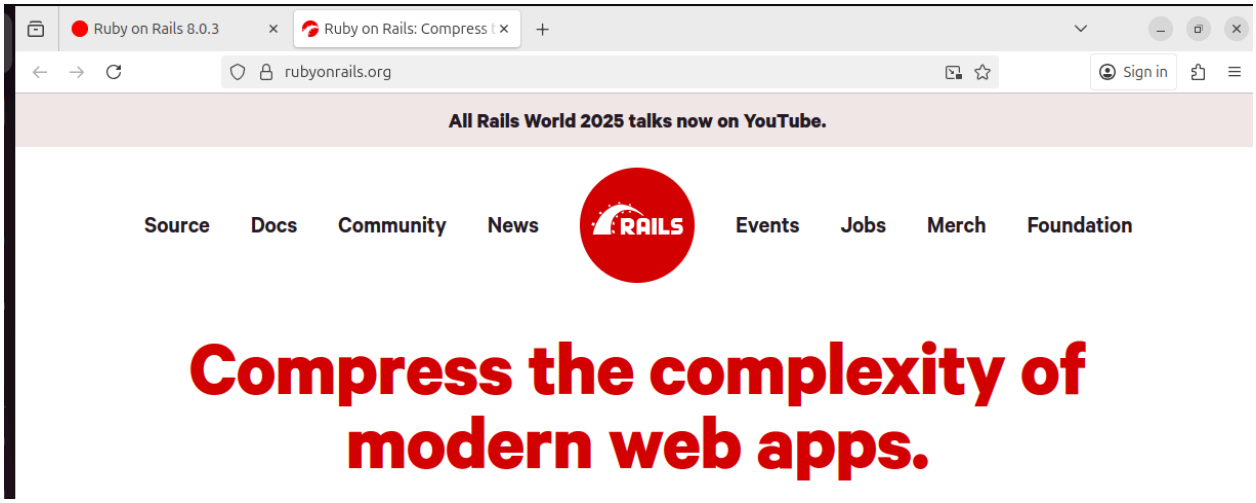


figure 17.4

18. Here, I integrated more of my PostgreSQL and Rails by setting up the logs to show on the Rails app. This gives it an interactive and cleaner look to the app and also see the entry logs.

nano log_entries_controller.rb

```
class LogEntriesController < ApplicationController

 def index

        @logs = LogEntry.all  end end
```

cd ../views

mkdir log_entries

cd log_entries

nano index.html.erb

```
        <h1>System Logs</h1><ul>  <% @logs.each do |log| %> <li><%= log.level %>: <%=
log.message %></li> <% end %></ul>
```

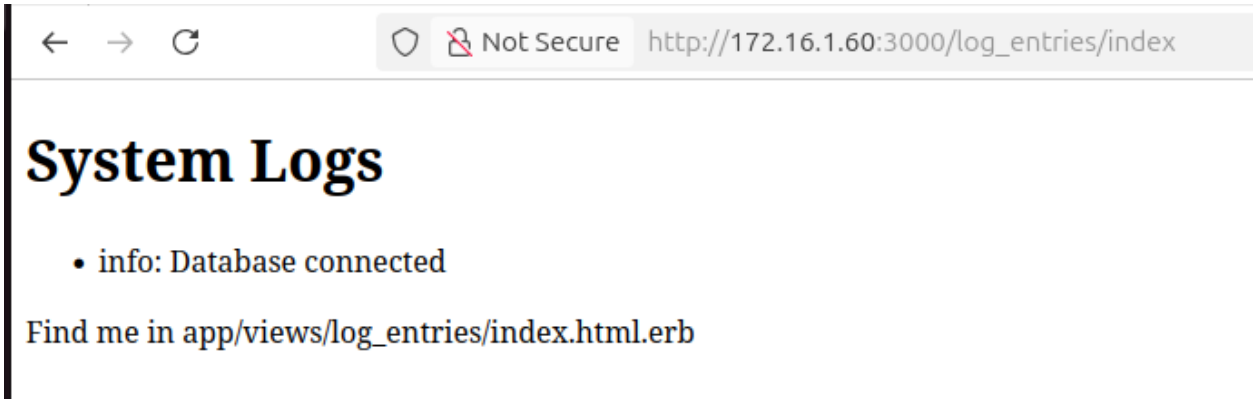


figure 18.1

```
Started GET "/log_entries/index" for 172.16.1.60 at 2025-10-23
12:18:38 -0400
Processing by LogEntriesController#index as HTML
  Rendering layout layouts/application.html.erb
  Rendering log_entries/index.html.erb within layouts/application
  [1m[36mLogEntry Load (2.2ms)[0m  [1m[34mSELECT
"log_entries".* FROM "log_entries" /
*action='index',application='Testapp',controller='log_entries'*/[0m
[0m
 ↳ app/views/log_entries/index.html.erb:3
  Rendered log_entries/index.html.erb within layouts/application
(Duration: 4.7ms | GC: 0.0ms)
```

figure 18.2

Made it look nicer

<!DOCTYPE html>

<html>

<head>

<title>JARED’S Secure IT Dashboard</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

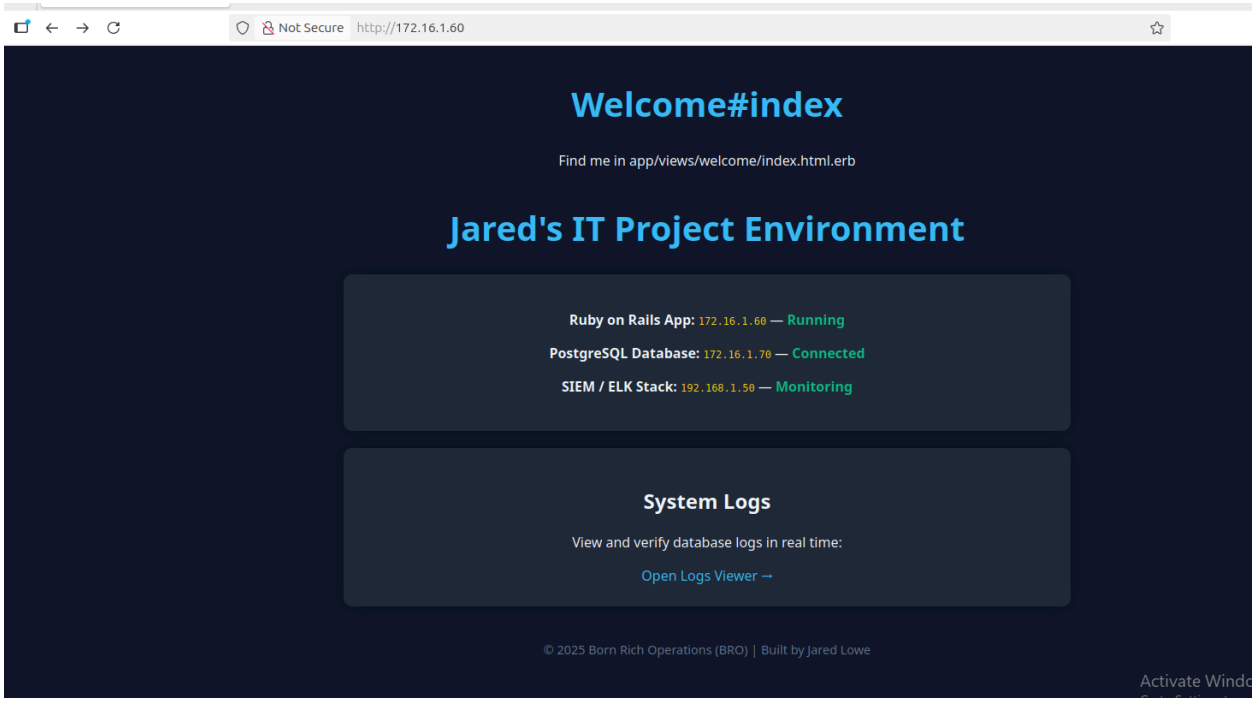
body { background-color:#0f172a; color:#f1f5f9; font-family: 'Segoe UI', sans-serif; text-align:center; margin:0; padding:0; }

h1 { font-size:2.5rem; color:#38bdf8; margin-top:40px; }

.card { background:#1e293b; border-radius:10px; padding:25px; margin:20px auto; width:80%; max-width:800px; box-shadow:0 0 20px rgba(0,0,0,0.4); }

```
    .status { color:#10b981; font-weight:bold; }

    .ipbox { font-family:monospace; color:#facc15; }

    a { color:#38bdf8; text-decoration:none; }

    a:hover { text-decoration:underline; }

  </style>

 </head>



 <body>

  <h1>Jared’s IT Project Environment</h1>

  <div class="card">

    <p><b>Ruby on Rails App:</b> <span class="ipbox">172.16.1.60</span> — <span
class="status">Running</span></p>

    <p><b>PostgreSQL Database:</b> <span class="ipbox">172.16.1.70</span> — <span
class="status">Connected</span></p>

    <p><b>SIEM / ELK Stack:</b> <span class="ipbox">192.168.1.50</span> — <span
class="status">Monitoring</span></p>

  </div>

 <div class="card">

    <h2>System Logs</h2>
```

```
    <p>View and verify database logs in real time:</p>
    <a href="/log_entries">Open Logs Viewer →</a>
  </div>
  <footer style="margin-top:40px;font-size:0.9rem;color:#64748b;">
    &copy; 2025 Born Rich Operations (BRO) | Built by Jared Lowe
  </footer>
 </body>
</html>
```

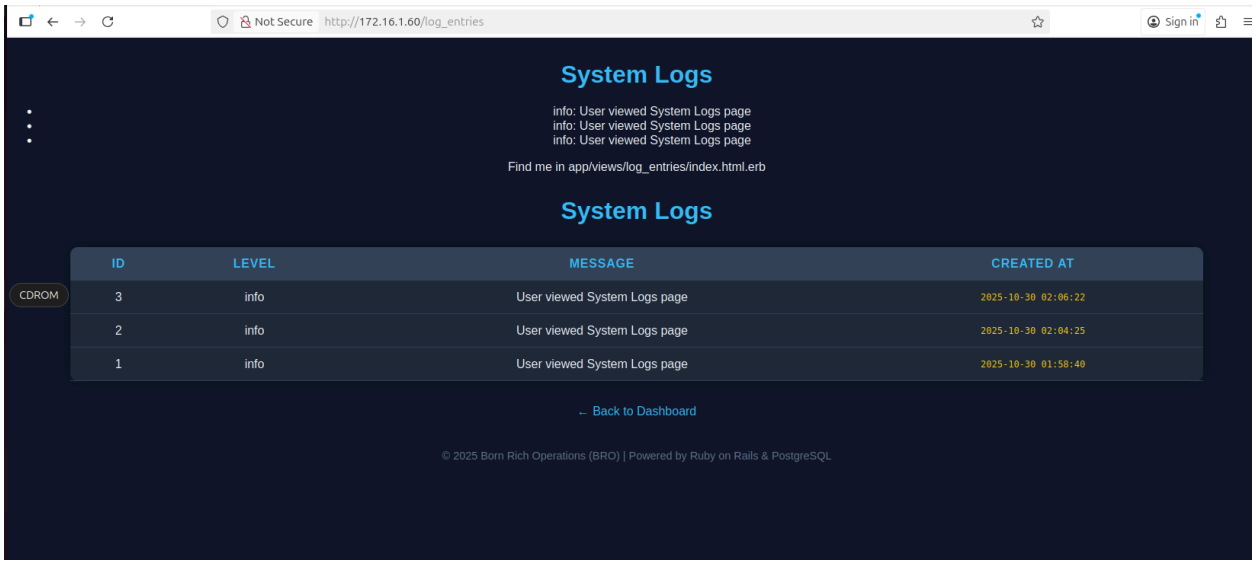



```
<!DOCTYPE html>
```

```
<html>
  <head>
    <title>System Logs</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
      body { background-color:#0f172a; color:#f1f5f9; font-family: 'Segoe UI', sans-serif;
text-align:center; margin:0; padding:0; }
      h1 { color:#38bdf8; margin-top:30px; }
      table { width:90%; margin:30px auto; border-collapse:collapse; background:#1e293b;
border-radius:10px; overflow:hidden; }
      th, td { padding:12px 15px; border-bottom:1px solid #334155; }
      th { background:#334155; color:#38bdf8; }
      tr:hover { background:#1e40af; }
      .timestamp { color:#facc15; font-family:monospace; }
      a { color:#38bdf8; text-decoration:none; }
      a:hover { text-decoration:underline; }
    </style>
  </head>
```

```
<body>
  <h1>System Logs</h1>
  <table>
    <tr>
      <th>ID</th>
      <th>Level</th>
      <th>Message</th>
      <th>Created At</th>
    </tr>
    <% @logs.each do |log| %>
      <tr>
        <td><%= log.id %></td>
        <td><%= log.level %></td>
        <td><%= log.message %></td>
        <td class="timestamp"><%= log.created_at.strftime("%Y-%m-%d %H:%M:%S") if log.created_at %></td>
      </tr>
    <% end %>
```

```
</table>


<p><a href="/">← Back to Dashboard</a></p>

</body>

</html>
```




```
  GNU nano 7.2                                  /home/jared/testapp/app/controllers/application_controller.rb
class ApplicationController < ActionController::Base
  # Only allow modern browsers supporting webp images, web push, badges, import maps, CSS nesting, and CSS :has.
  allow_browser versions: :modern
end
def record_log(message, level = 'info')
  LogEntry.create(message: message, level: level)
rescue => e
  Rails.logger.error "Failed to record log: #{e.message}"
end
```

```
  GNU nano 7.2                                  /home/jared/testapp/app/controllers/log_entries_controller.rb
class LogEntriesController < ApplicationController
  def index
    record_log("User viewed System Logs page", "info")
    @logs = LogEntry.all.order(created_at: :desc)
  end

  def api
    render json: LogEntry.all
  end
end
```

I also added it to be readable in JSON for my future Siem tool(api), figure 18.1.1

19. Install Puma and nginx and create a systemd service for the Rails app.

figure 19.1

20. Add Elastic repository & installed Elasticsearch

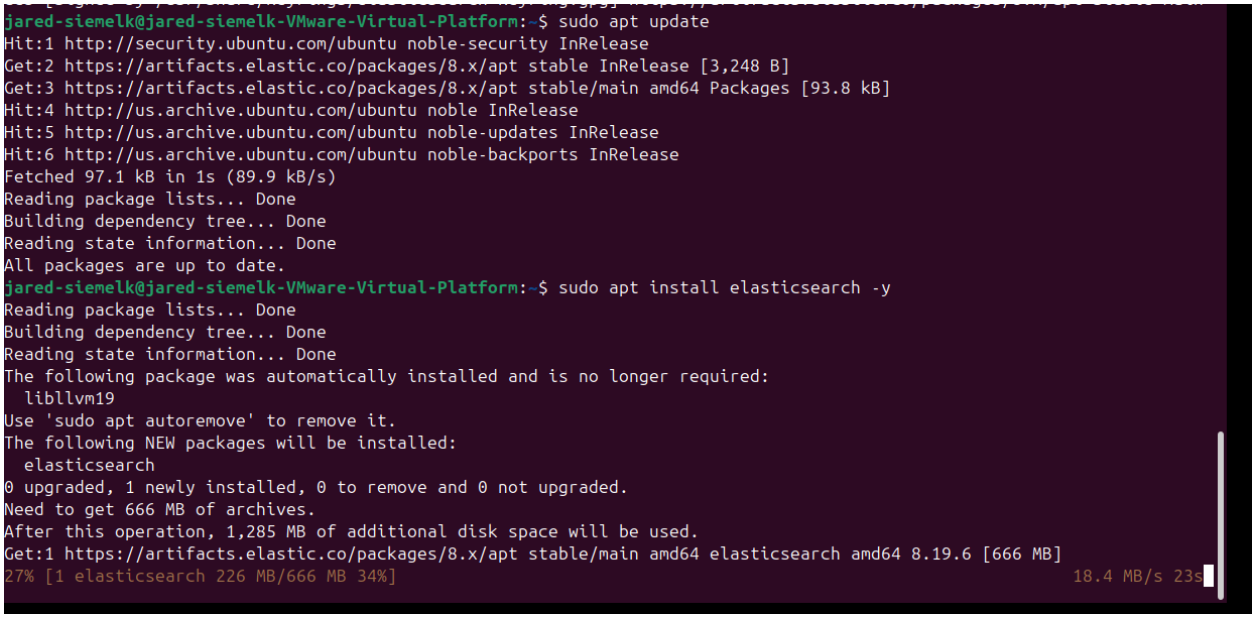


figure 20.1

sudo apt update

sudo apt install wget curl gnupg -y

Imports a GPG key and saved it:

curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg

Adds a repository:

echo "deb [signed-by=/usr/share/keyrings/elasticsearch-keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.list

sudo apt install elasticsearch -y

Then I configure it:

Edit /etc/elasticsearch/elasticsearch.yml:

network.host: 192.168.1.50

http.port: 9200

discovery.type: single-node

Then enable & start:

sudo systemctl daemon-reload

sudo systemctl enable elasticsearch

sudo systemctl start elasticsearch

Test:

curl http://192.168.1.50:9200

```
Password for the [elastic] user successfully reset.
New value: VjMX=O=ZAeuMW4JXSyrc
jared-siemelk@jared-siemelk-VMware-Virtual-Platform:~$ ^C
jared-siemelk@jared-siemelk-VMware-Virtual-Platform:~$ curl -u elastic https://192.168.1.50:9200 -k
Enter host password for user 'elastic':
{
  "name" : "jared-siemelk-VMware-Virtual-Platform",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "t-cYdFg3RcertYol87-ehg",
  "version" : {
    "number" : "8.19.6",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "d2c42d91a1eb9e14b1a37c4d87eb2533ec859e2b",
    "build_date" : "2025-10-21T22:05:27.062491219Z",
    "build_snapshot" : false,
    "lucene_version" : "9.12.2",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
jared-siemelk@jared-siemelk-VMware-Virtual-Platform:~$
```

figure 20.2

```
jared-siemelk@jared-siemelk-VMware-Virtual-Platform:~$ sudo systemctl status elasticsearch --no-pager
sudo systemctl status kibana --no-pager
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; preset: enabled)
    Drop-In: /etc/systemd/system/elasticsearch.service.d
             └─override.conf
     Active: active (running) since Mon 2025-11-03 17:45:53 EST; 54min ago
       Docs: https://www.elastic.co
   Main PID: 1604 (java)
      Tasks: 97 (limit: 9374)
     Memory: 2.7G (peak: 2.8G)
        CPU: 3min 30.970s
     CGroup: /system.slice/elasticsearch.service
             ├─1604 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.scrip…
             ├─2272 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.ne…
             └─2303 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

Nov 03 17:44:30 jared-siemelk-VMware-Virtual-Platform systemd[1]: Starting elasticsearch.service - Elasticsearch...
Nov 03 17:44:30 jared-siemelk-VMware-Virtual-Platform systemd-entrypoint[1604]: warning: ignoring JAVA_HOME=/usr/sh… JDK
Nov 03 17:45:53 jared-siemelk-VMware-Virtual-Platform systemd[1]: Started elasticsearch.service - Elasticsearch.
```

figure 20.3

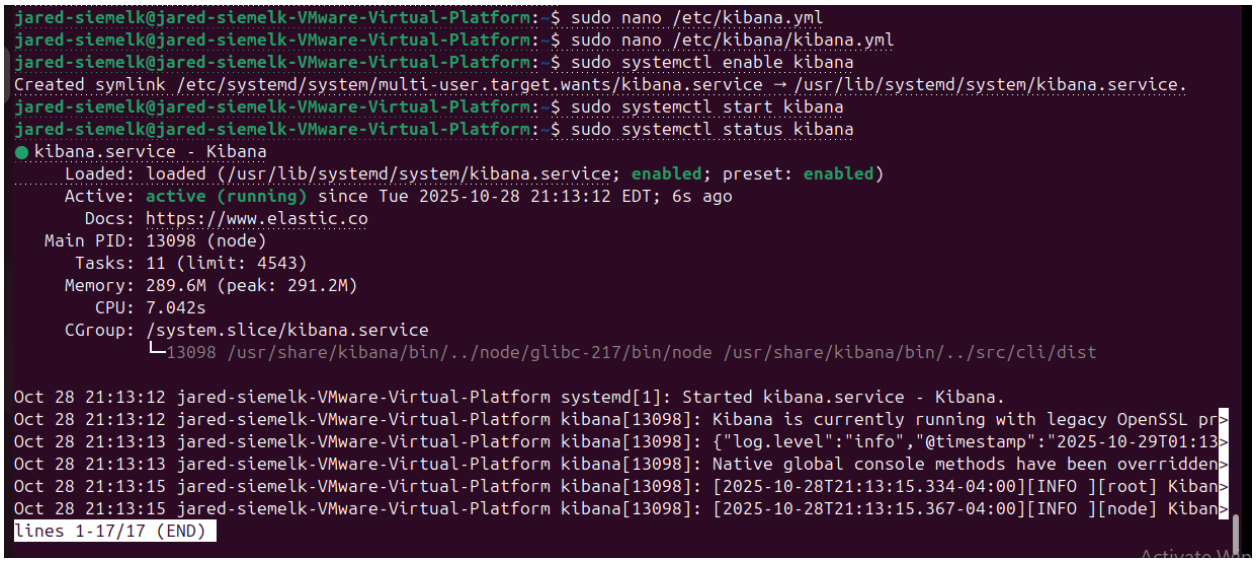


figure 20.4

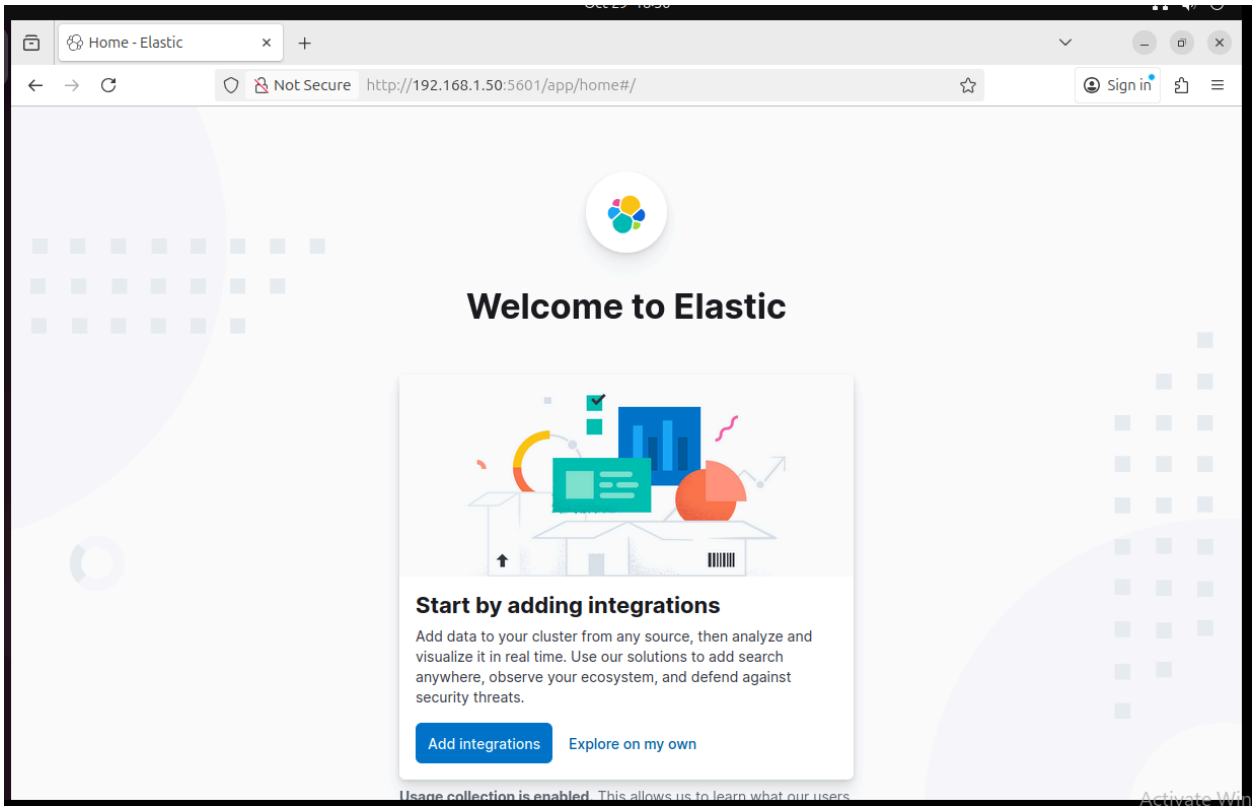


figure 20.5

21. Now since we got Elastic on and working I set all vm to use filebeat to forward logs to elastic so we can create alerts. Filebeat will read PostgreSQL log files and forward them securely to

Elasticsearch.

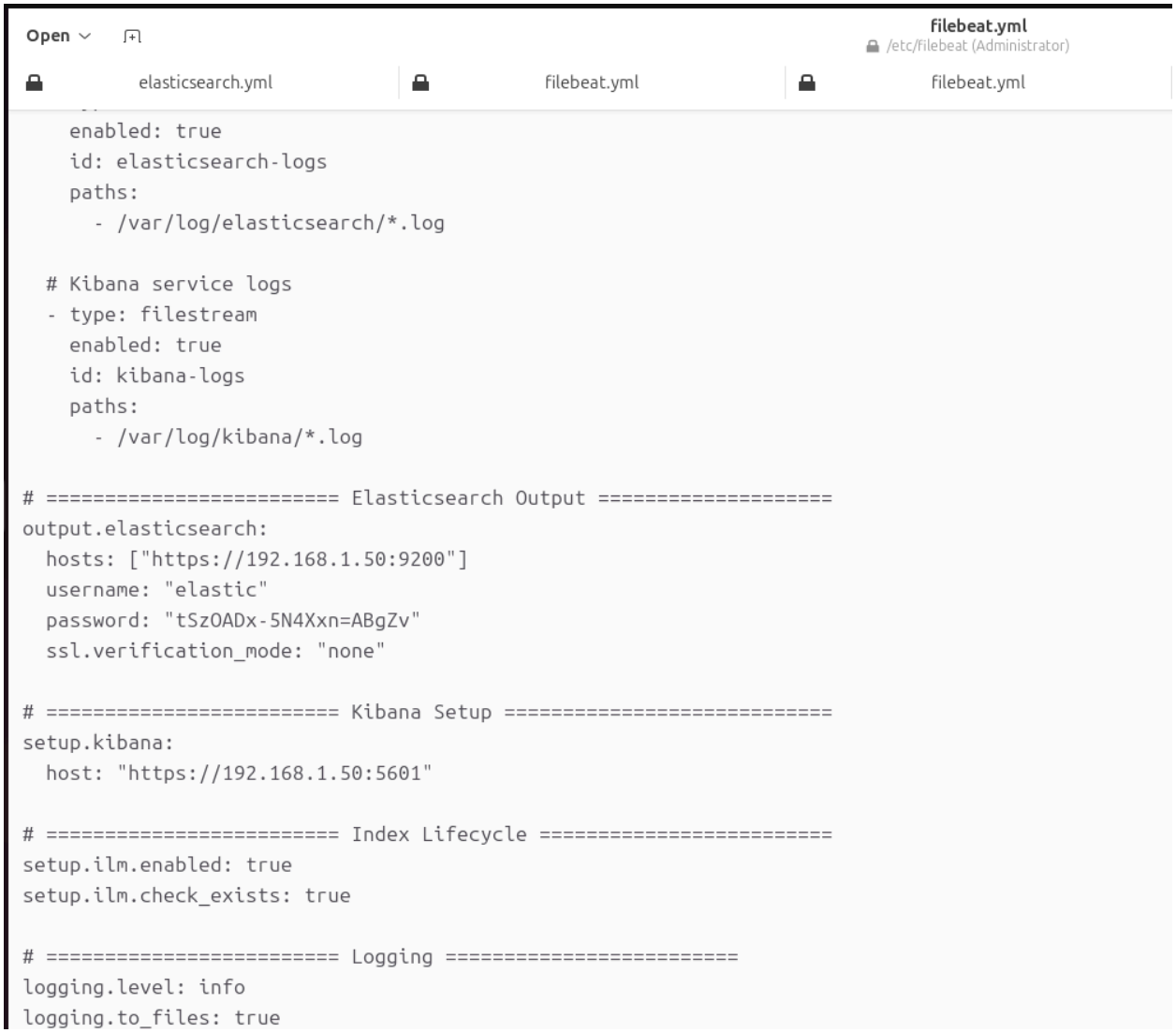


figure 21.1

Here we have the filebeats.yml for ruby rails to show that it is working and sending.

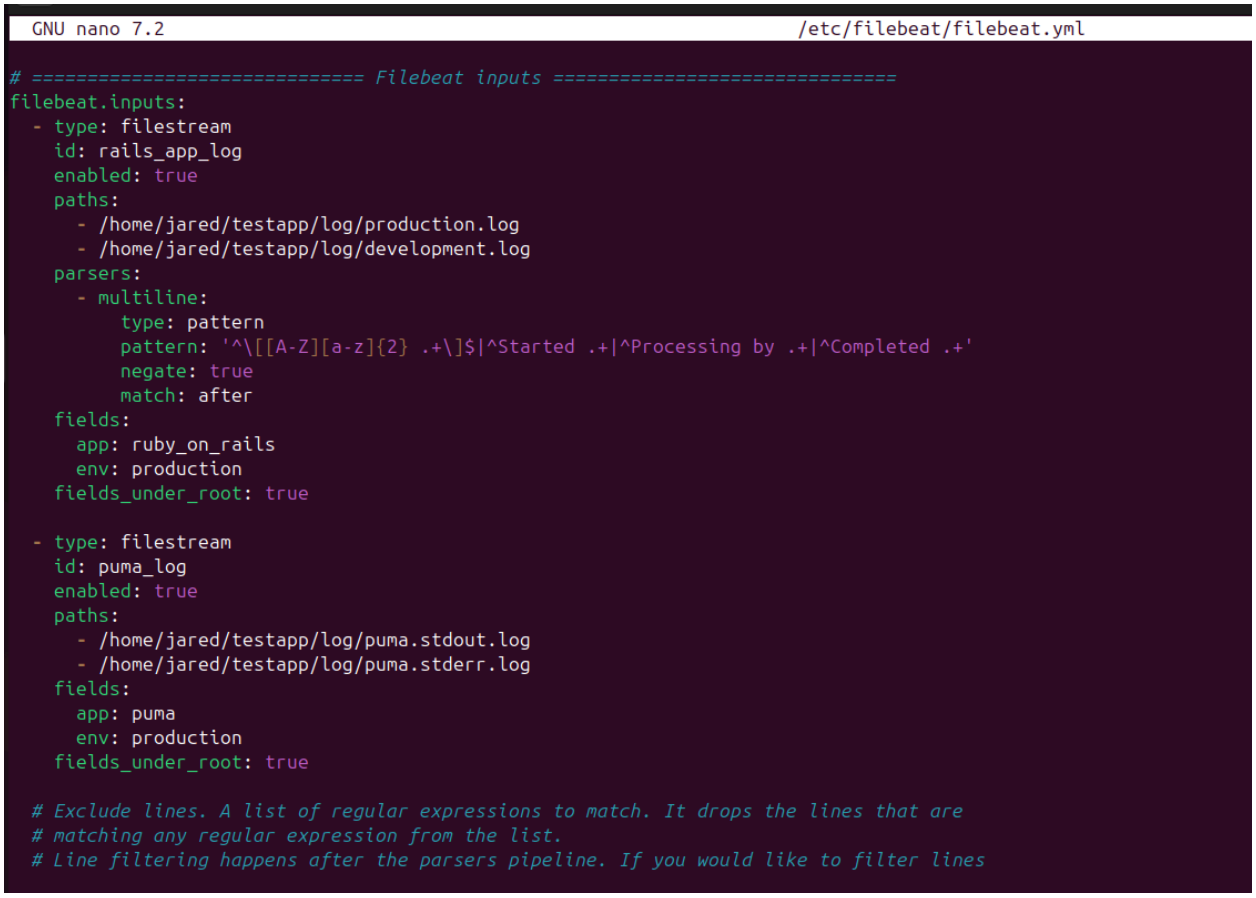


figure 21.2

Permissions was a big issue so here on each machine for simplicity:

sudo chown root:filebeat /etc/filebeat/filebeat.yml

sudo chmod 640 /etc/filebeat/filebeat.yml

sudo chown -R filebeat:filebeat /var/log/filebeat

Enable detailed SQL activity logging for ingestion.

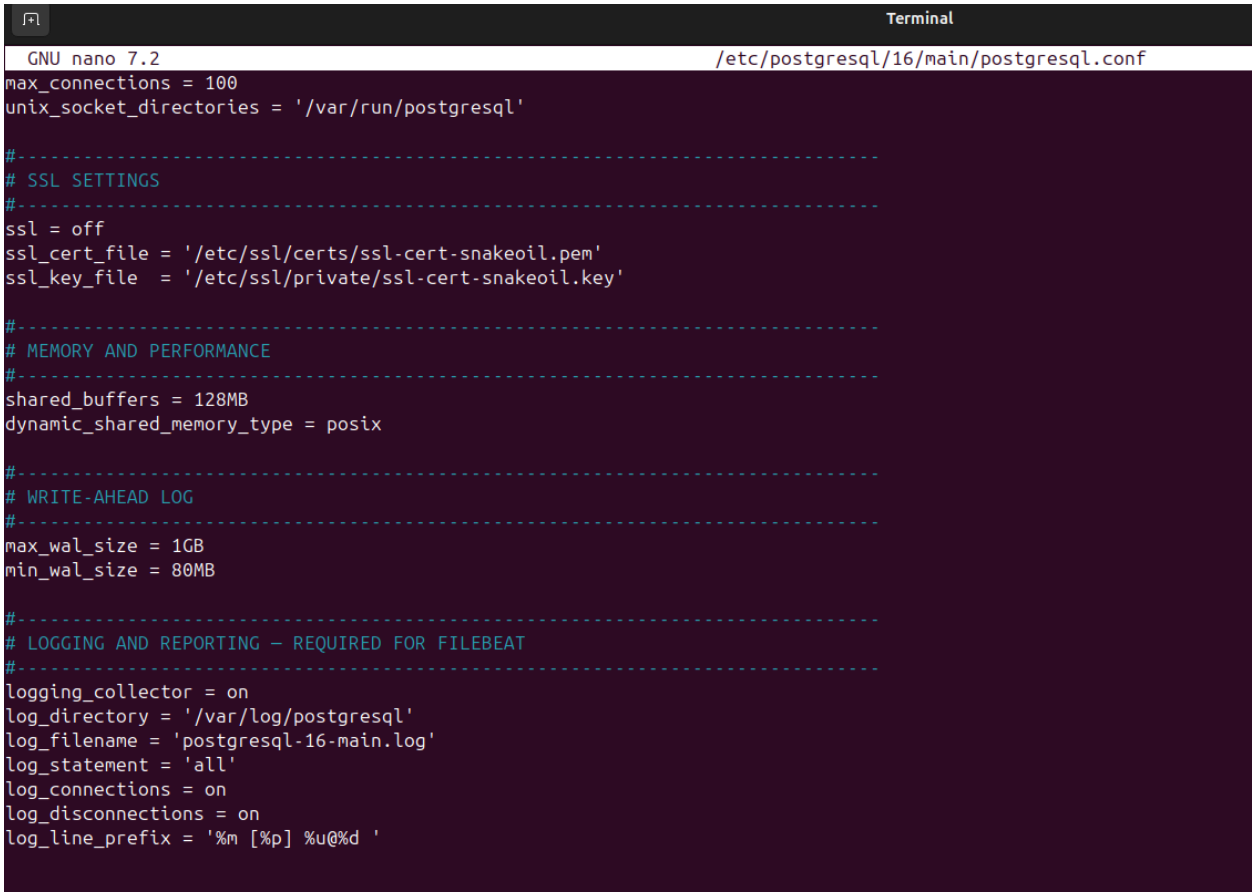


figure 21.2

After Filebeat starts sending logs:

sudo filebeat setup --dashboards \

-E setup.kibana.host=http://192.168.1.50:5601 \

-E output.elasticsearch.username=elastic \

-E output.elasticsearch.password='tSzOADx-5N4Xxn=ABgZv'

Also for simplicity I added all pfsense logs to be forwarded to Kibana where we can read them there.

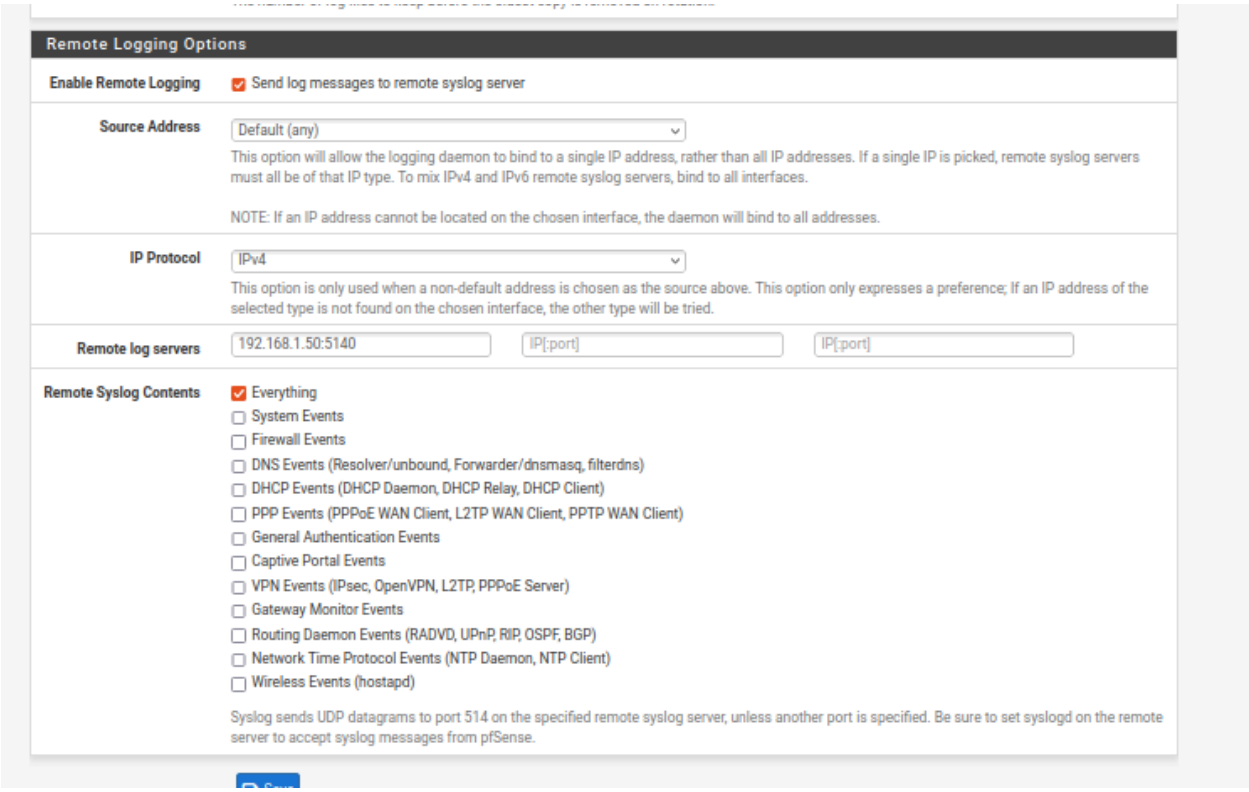


figure 21.3

22. Now that I have all vms to send logs with filebeat I now will take them with kibana and elastic to show on their site to have alerts and read the logs.

This loads prebuilt dashboards such as; which is excellent for this project:

[Filebeat PostgreSQL] Database Logs Overview

[Filebeat PostgreSQL] Query Duration Overview ECS

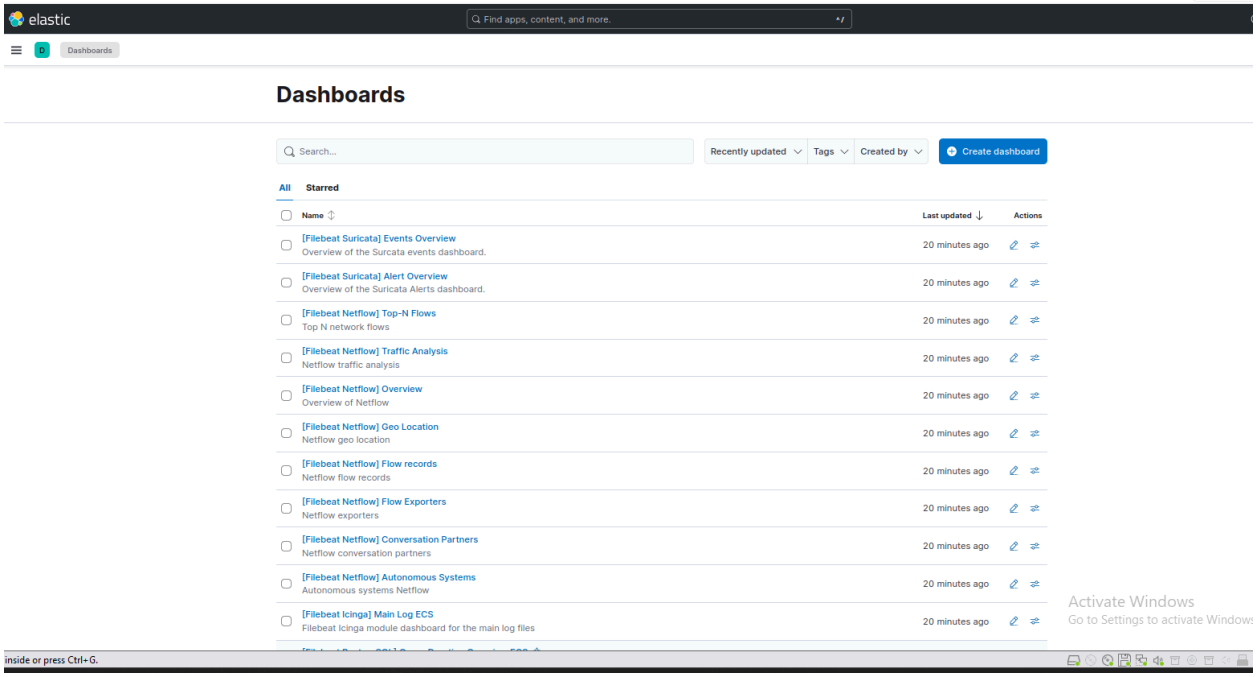


figure 22.1

We will use the vms credentials such as apps and ips to filter machine activity that I would like to see which will help for alerting.

connection authorized: user=postgres

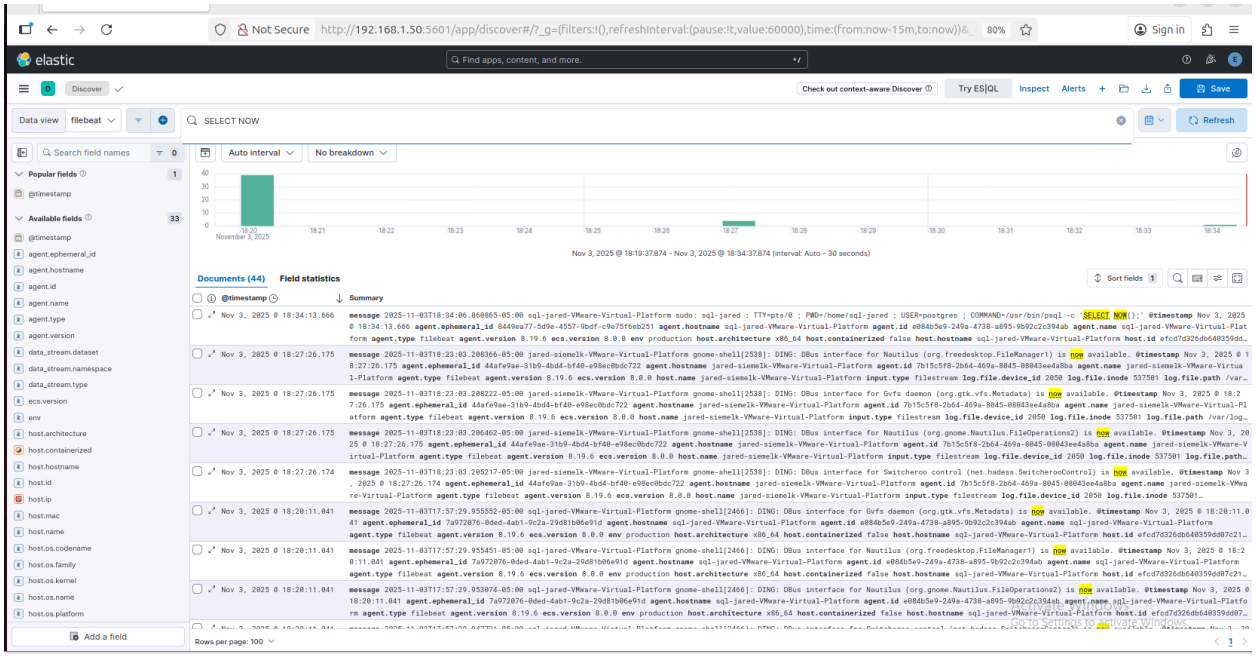


figure 22.2

Example log entries observed in Kibana making sure my ruby appears and right host machine:

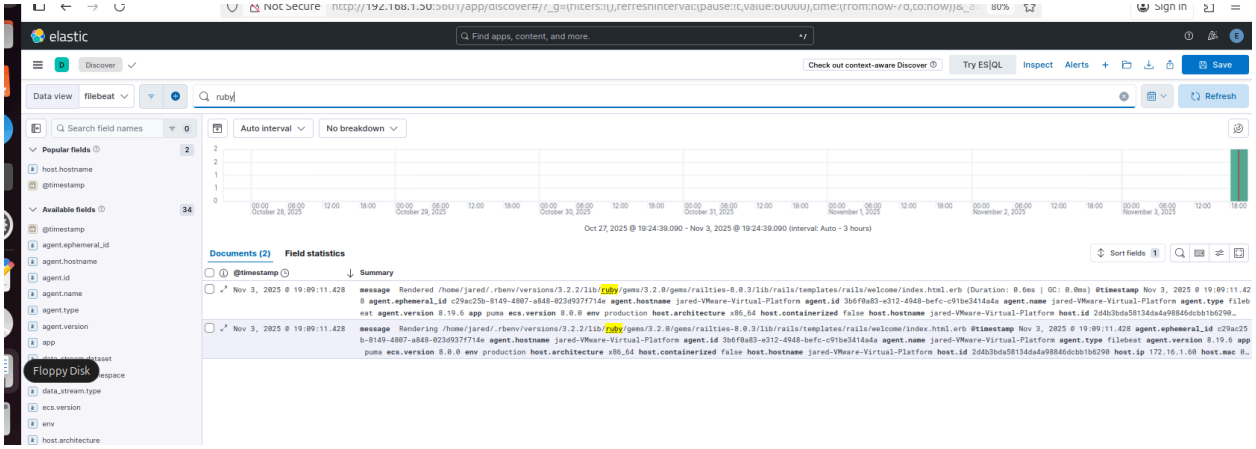


figure 22.3

Now that we have them all working now we set up dashboards in kibana that where we are able to see any errors, logs, and activity on our virtual machines(database and site)

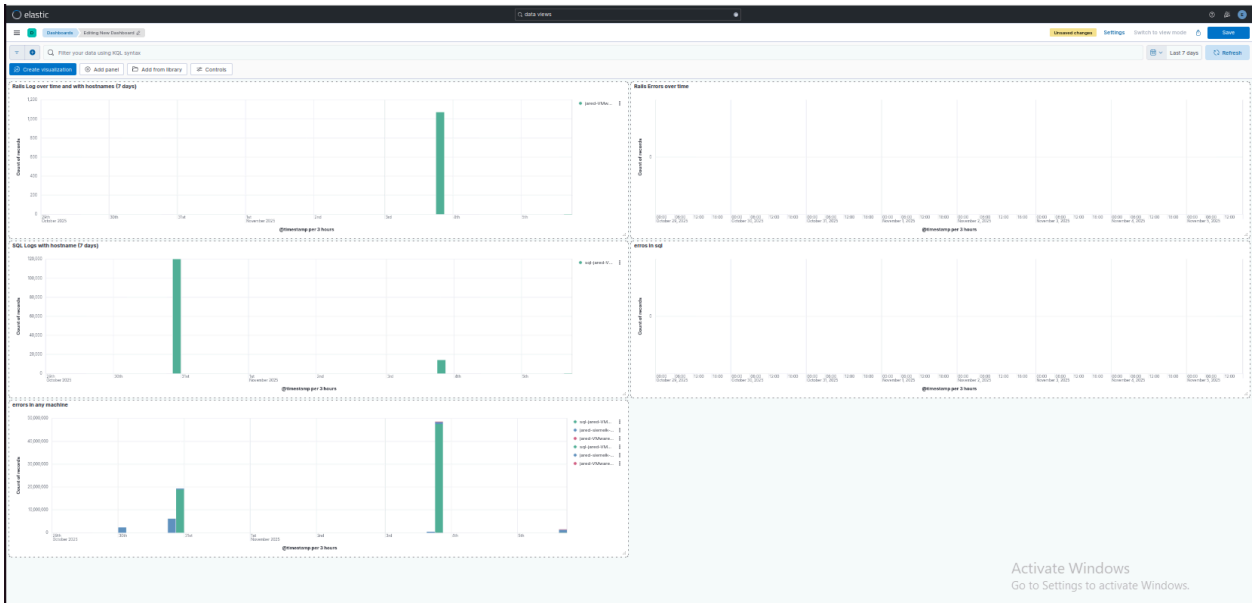


figure 22.4

Here I tested with logging into the db in rails on the site multiple items and then i sent a test db to the test database in psql and got activity!

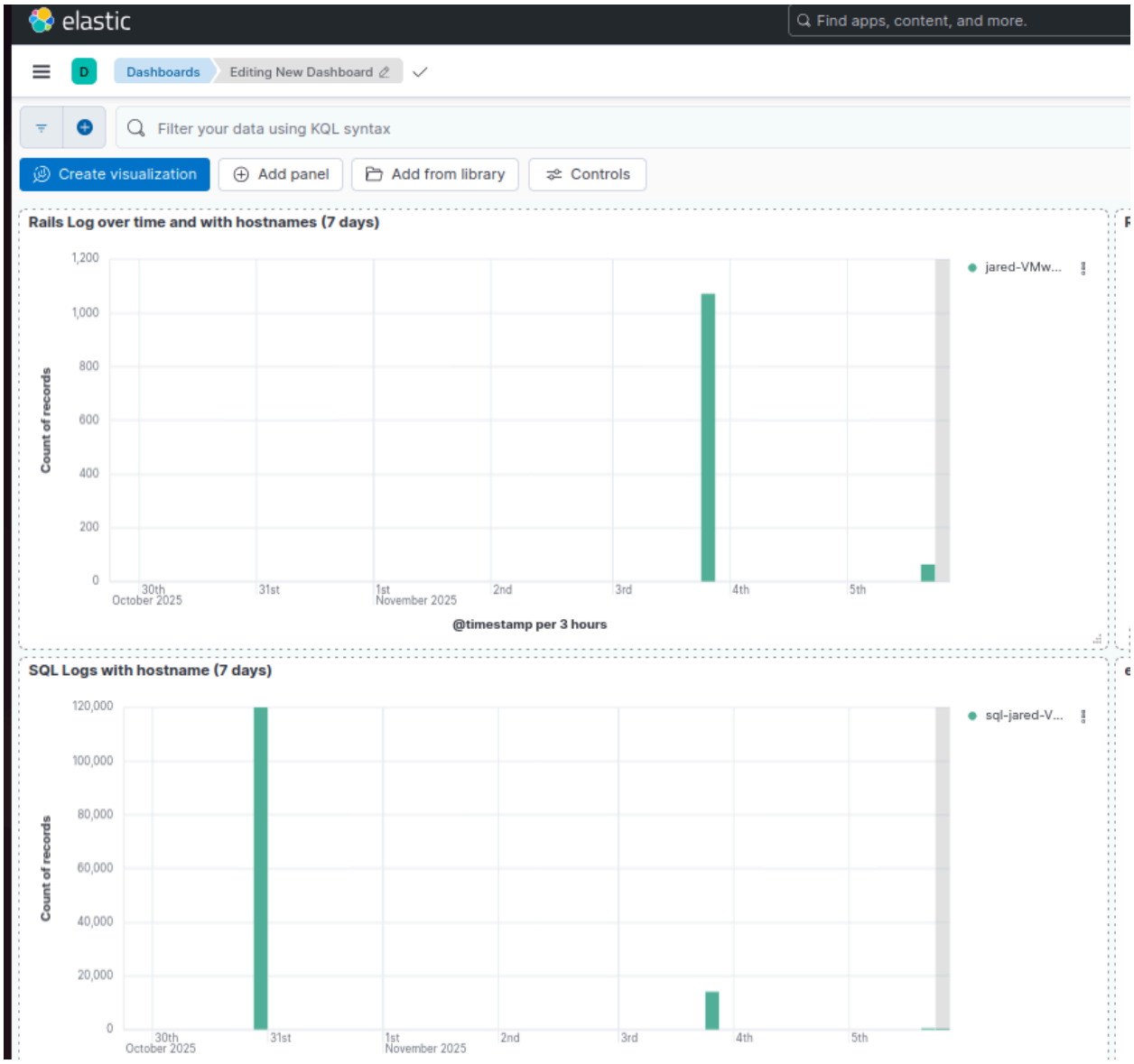


figure 22.5

23. Testing the rules and alerts for the machines to make sure logs are coming in on time and working.

With our first rule/alert any ruby rails that gives out a bad http response code will get alerted:

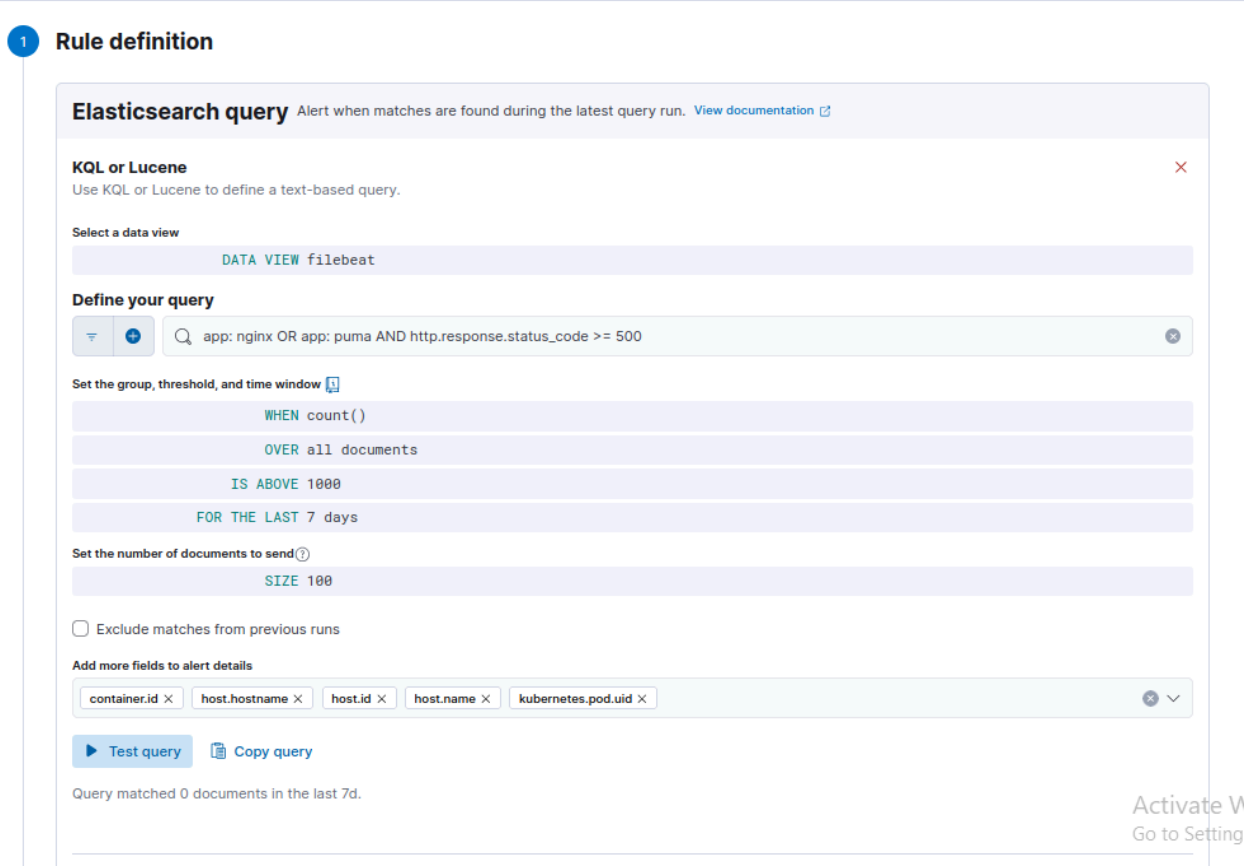


figure 23.1

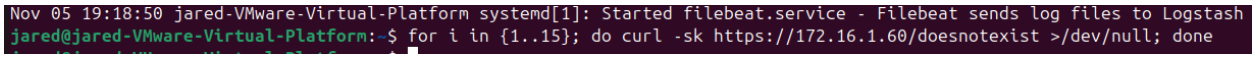


figure 23.1.1

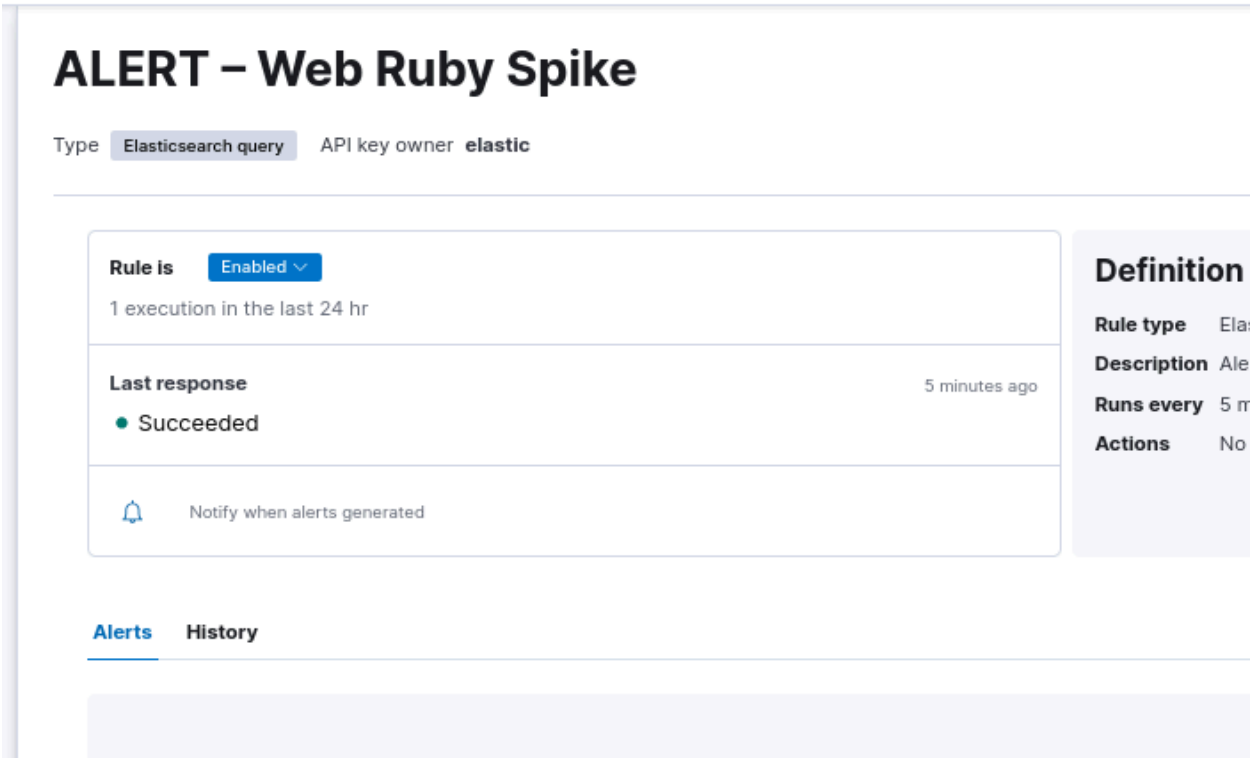


figure 23.1.2

When postgresql gets a bad login testing alerts and rules

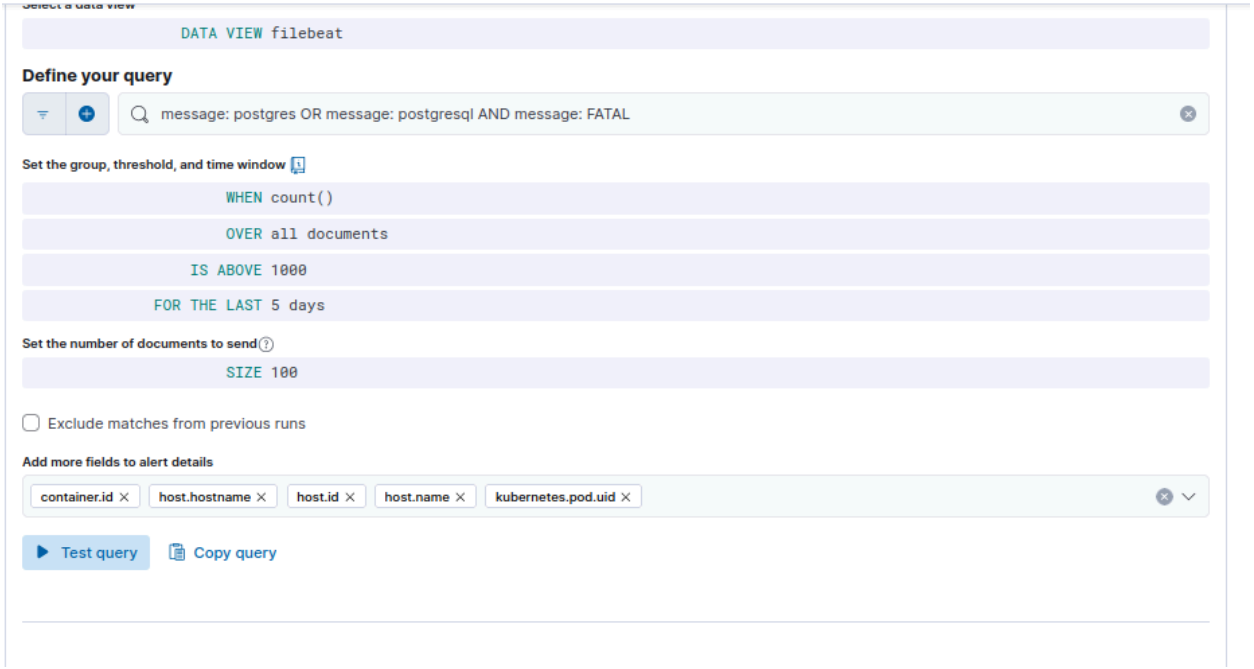


figure 23.2

```
sql-jared@sql-jared-VMware-Virtual-Platform:~$ PGPASSWORD=badpass psql -h 172.16.1.70 -U wrong -d postgres -c "SELECT 1;" || true
psql: error: connection to server at "172.16.1.70", port 5432 failed: FATAL:  password authentication failed for user "wrong"
sql-jared@sql-jared-VMware-Virtual-Platform:~$
```

figure 23.2.1

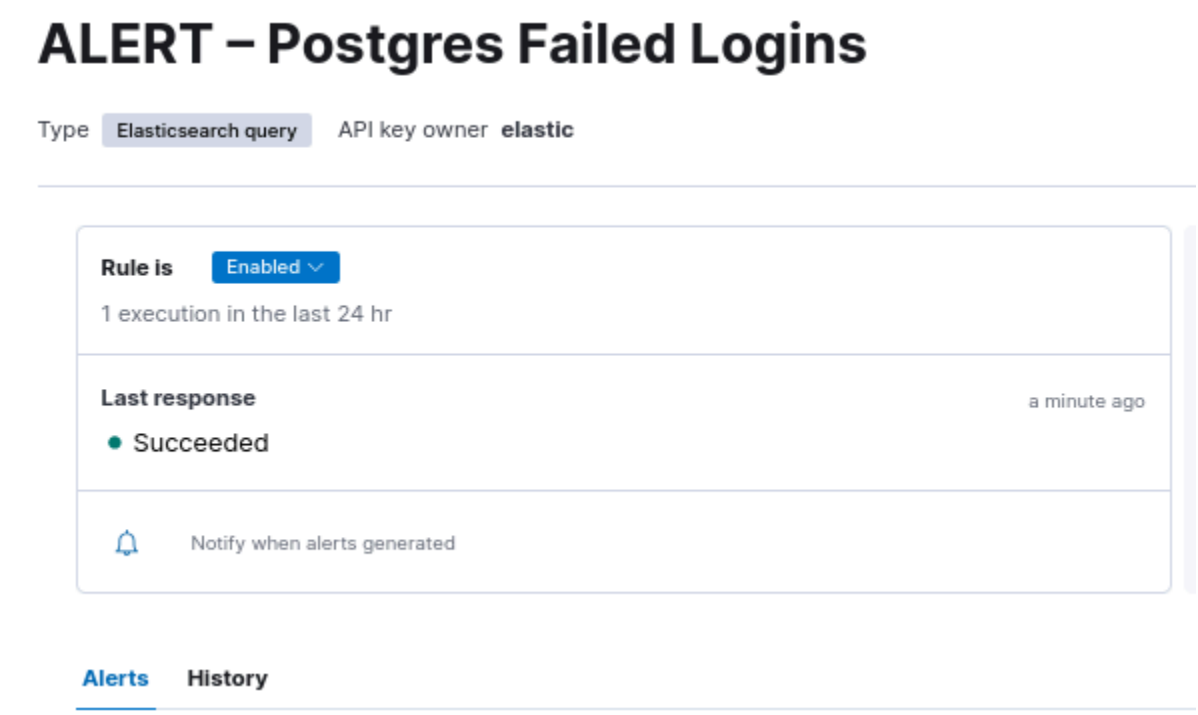


figure 23.2.2

24. Now I will add the pihole DNS and configure it for pfsense to use.

curl -sSL https://install.pi-hole.net | bash

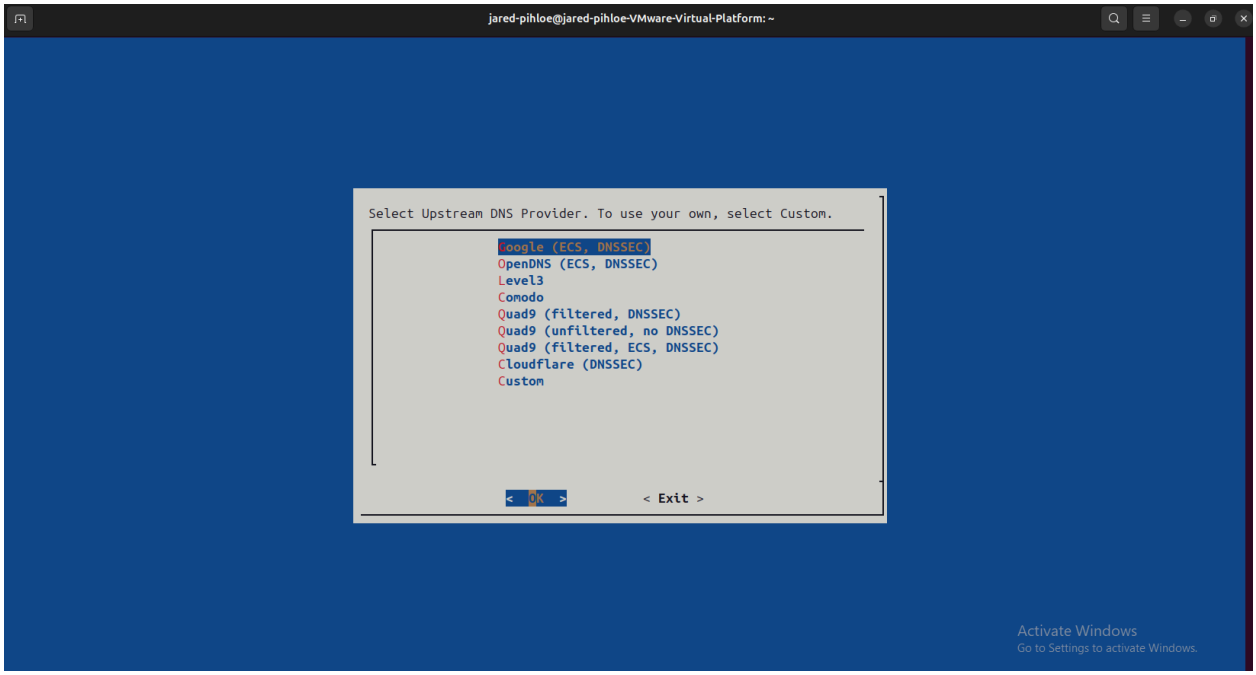


figure 24.1

Here I finished the installation and now we can move to adding it to pfsense.

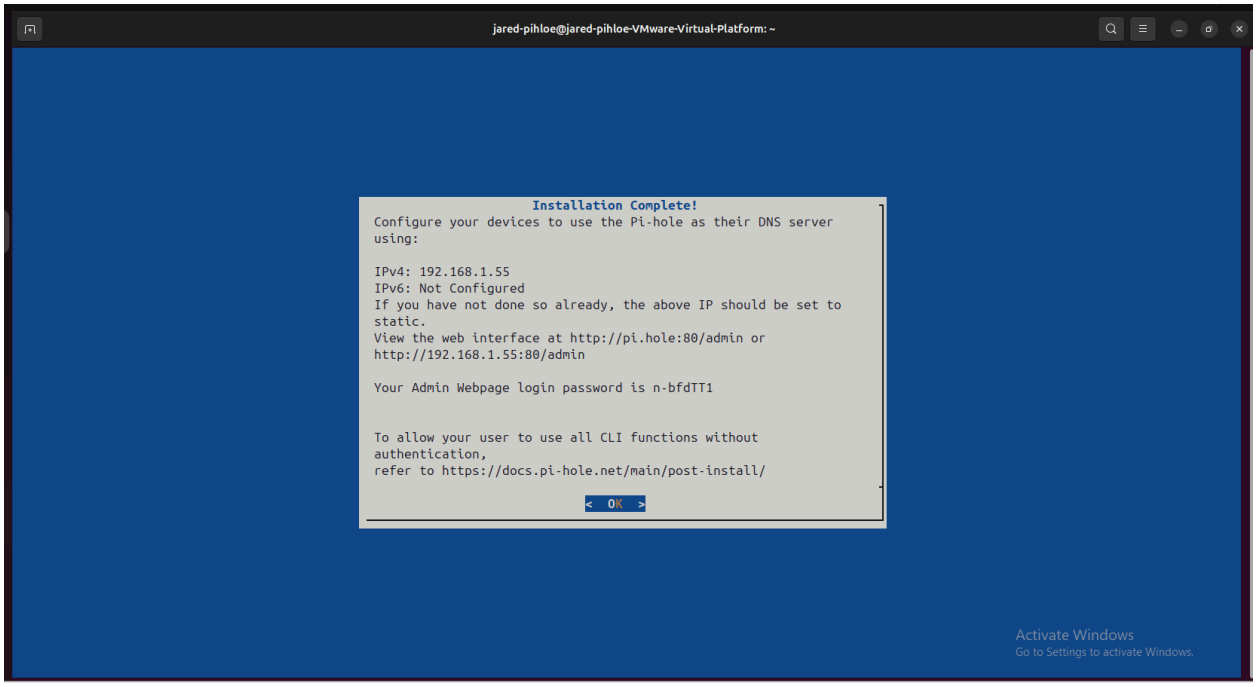


figure 24.2

Here I set it the DNS on the LAN to the ip for the pihole(I did it for each vnmet/interface)

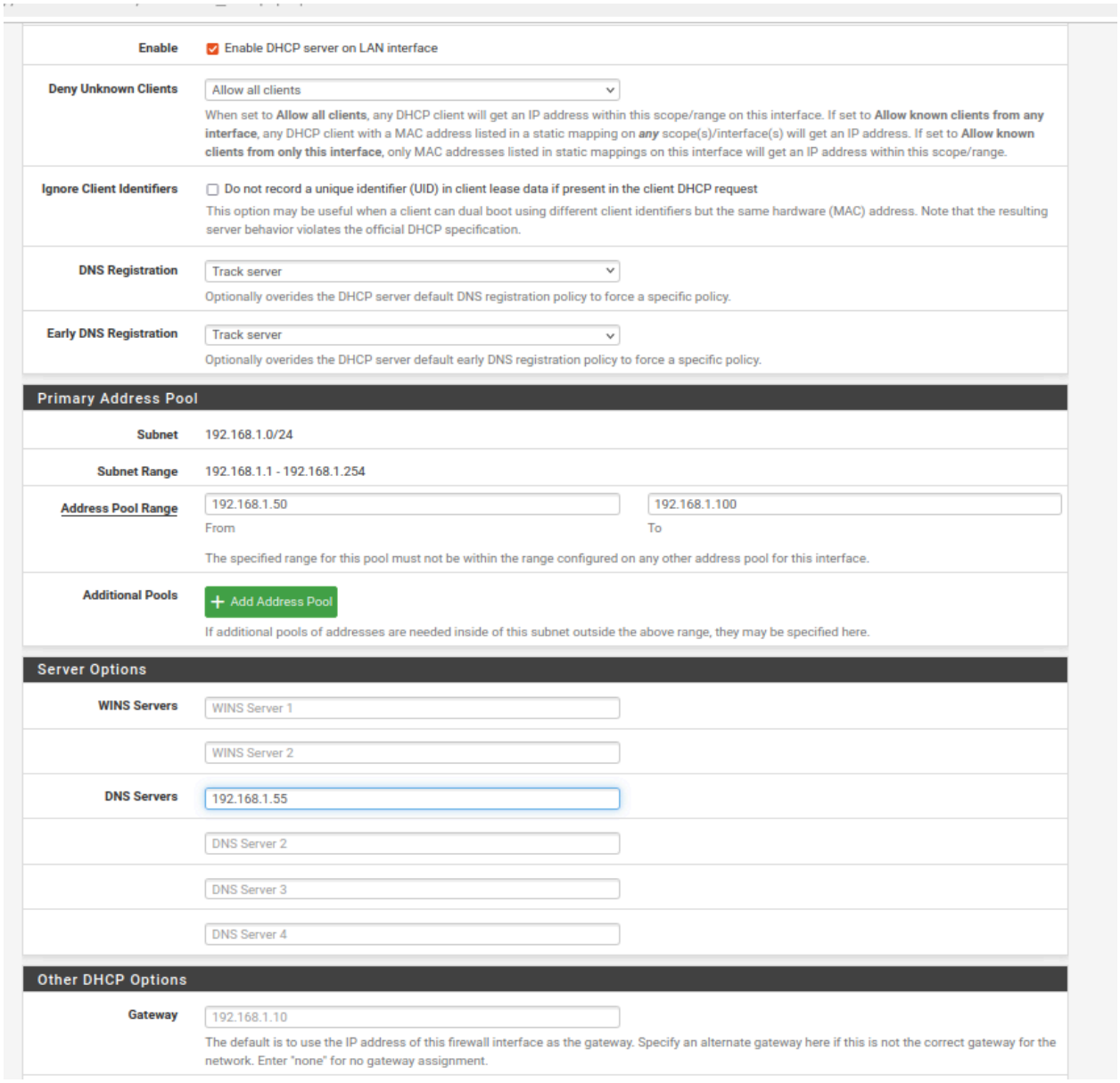


figure 24.2

25.Now I can start using the kali machine to attack the network.

I allowed the kali to be connected and take all vmnets so I can attack machines all at once and use this here to see the ips that are attacking using Kibana.

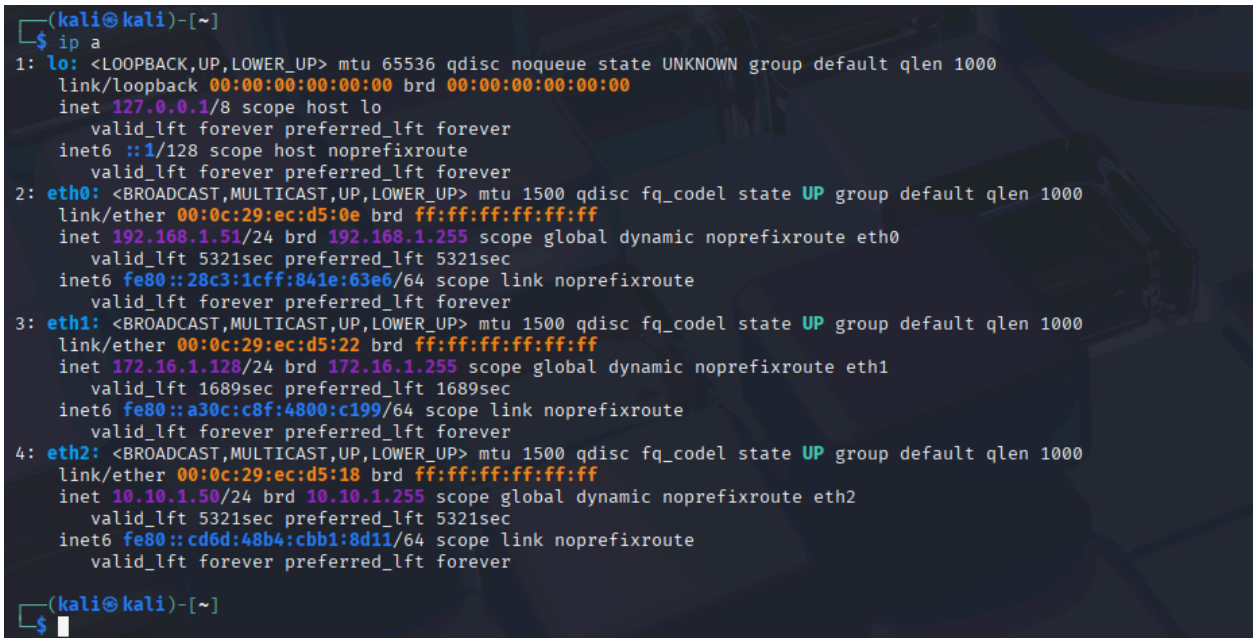
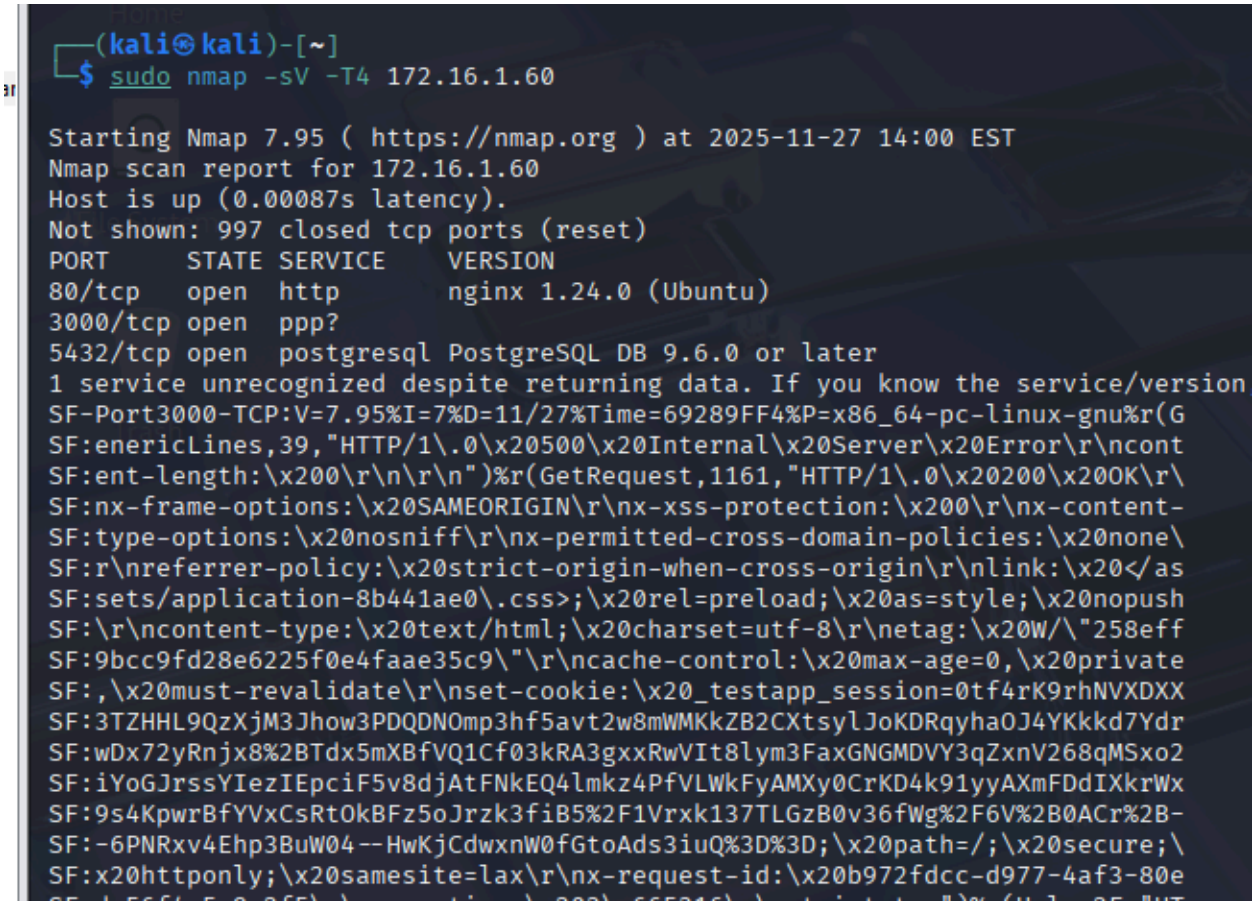


figure 25.1

I then installed metasploit and gobuster where i will use them to attack.

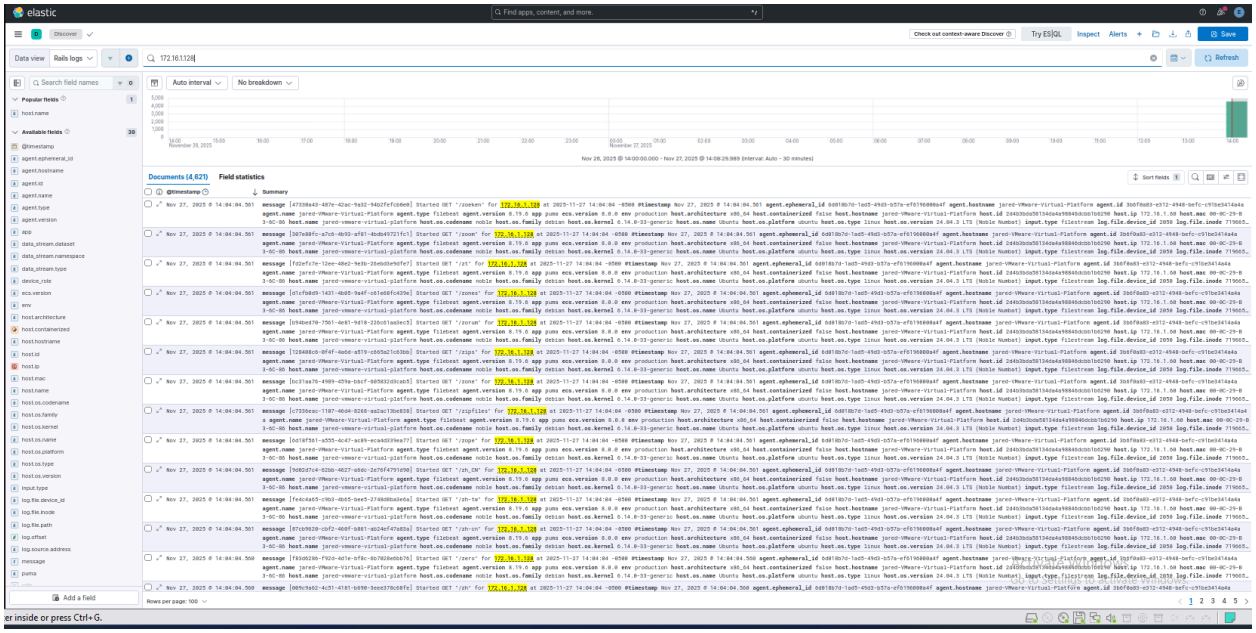
26. Ruby Rails Attacks



simple nmap scan to find vulnerabilities.

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir \
  -u http://172.16.1.60:3000 \
  -w /usr/share/wordlists/dirb/common.txt

===============================================================
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://172.16.1.60:3000
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.8
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/404                 (Status: 200) [Size: 4836]
/500                 (Status: 200) [Size: 7918]
/400                 (Status: 200) [Size: 6685]
/robots.txt          (Status: 200) [Size: 99]
/up                  (Status: 200) [Size: 73]
Progress: 4613 / 4613 (100.00%)
===============================================================
Finished
===============================================================

┌──(kali㉿kali)-[~]
```

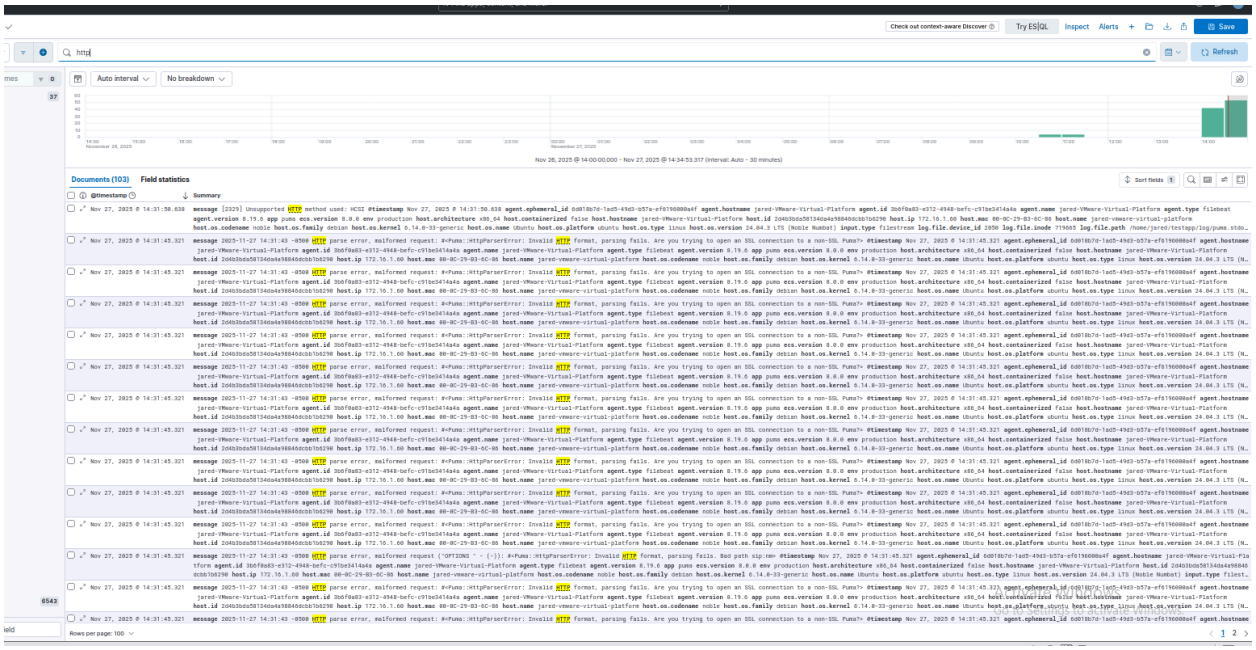
Here I use gobuster scan on the machine to discover several accessibilities.



Alot of /get that kali used for this test and we see this on kibana.

```
┌──(kali㉿kali)-[~]
└─$ msfconsole -q -x "use auxiliary/scanner/http/http_version; set RHOSTS 172.16.1.60; set RPORT 3000; run; exit"
RHOSTS ⇒ 172.16.1.60
RPORT ⇒ 3000
[+] 172.16.1.60:3000
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

┌──(kali㉿kali)-[~]
```

Used the metaspoit command to attack our ruby rails site that I made.

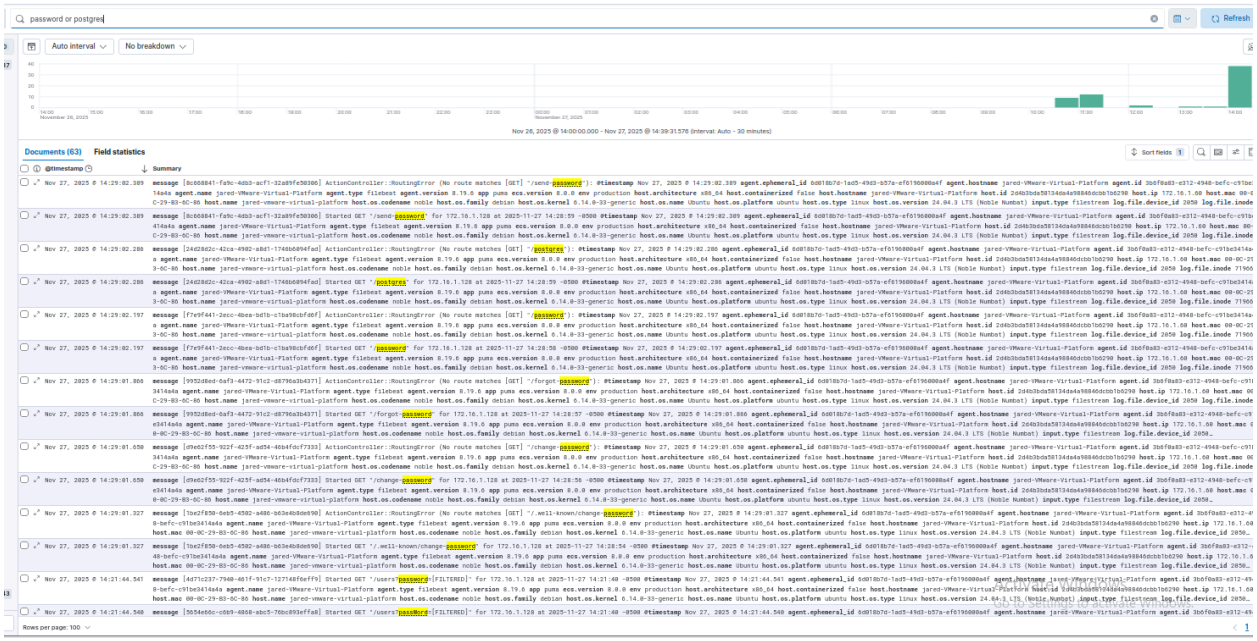


using HTTP search we saw all the attacks.

27. Postgresql Attacks

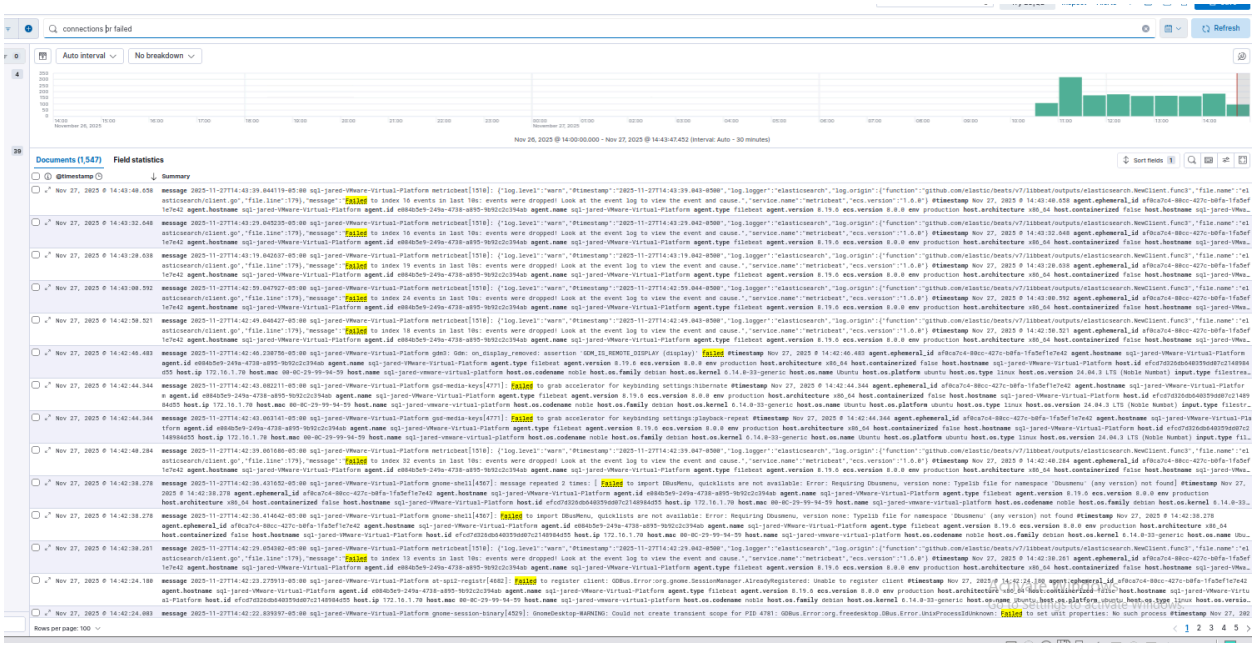
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sS -sV -p 5432 172.16.1.70
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-27 14:36 EST
Nmap scan report for 172.16.1.70
Host is up (0.00074s latency).

PORT     STATE SERVICE    VERSION
5432/tcp open  postgresql PostgreSQL DB 9.6.0 or later
MAC Address: 00:0C:29:99:94:59 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.90 seconds

┌──(kali㉿kali)-[~]
```

using nmap to find vulnerabilities.



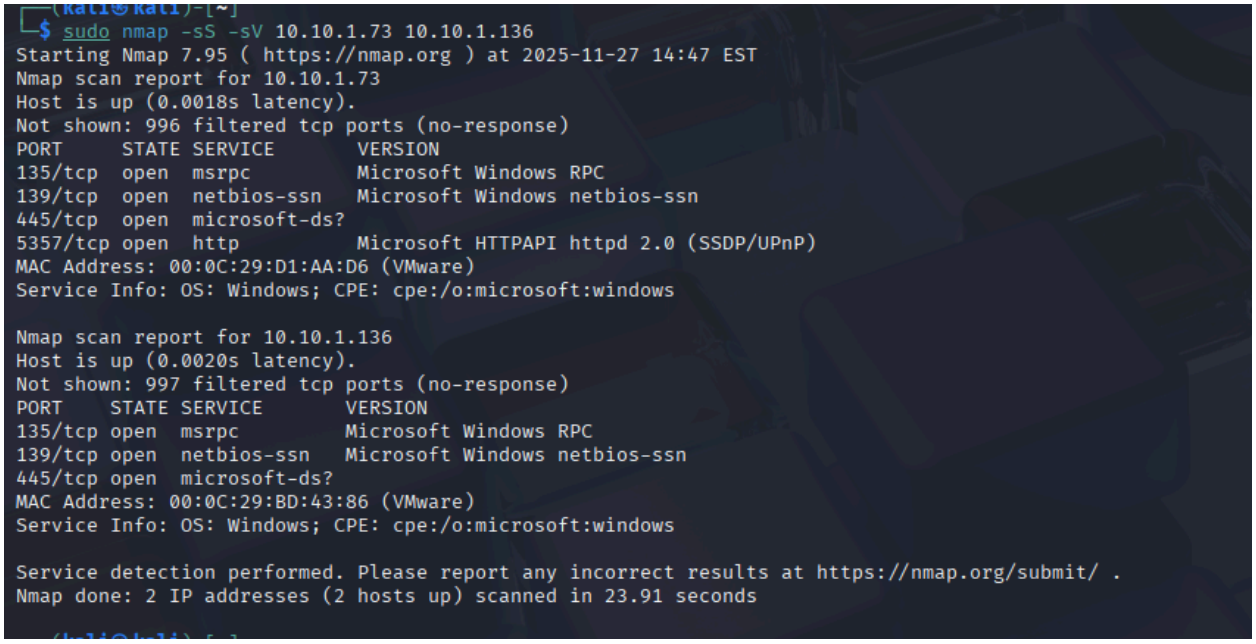
```
┌──(kali㉿kali)-[~]
└─$ for i in {1..50}; do PGPASSWORD=wrongpass psql -h 172.16.1.70 -U postgres -c "SELECT 1;" 2>/dev/null & done
[2] 101359
[3] 101360
[4] 101361
[5] 101362
[6] 101363
[7] 101364
[8] 101365
[9] 101366
[10] 101367
[11] 101368
[12] 101369
[13] 101370
[14] 101371
[15] 101372
[16] 101373
[17] 101374
[18] 101375
[19] 101376
[20] 101377
[21] 101378
[22] 101379
[23] 101380
[24] 101381
[25] 101382
[26] 101383
[27] 101384
[28] 101385
[29] 101386
[30] 101387
[31] 101388
```

Using an attack to send bad login attempts into postgres.

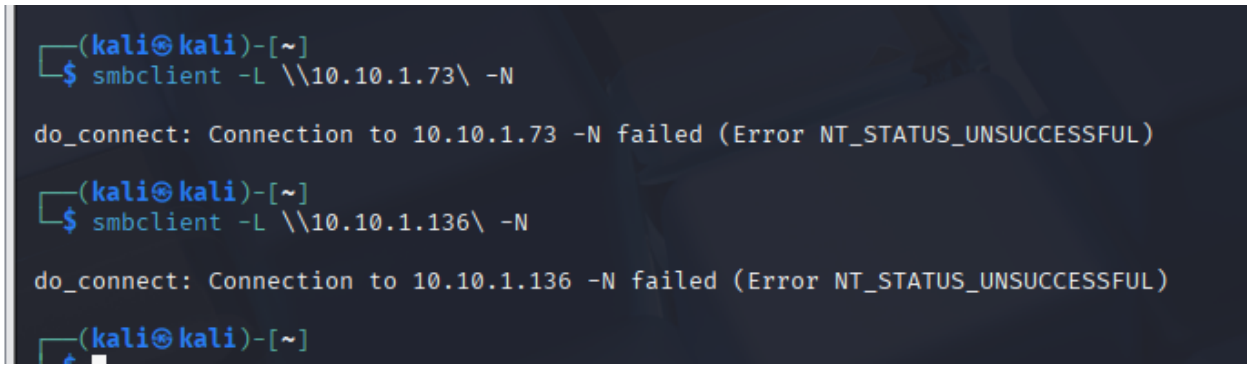


We saw those failures here in the logs

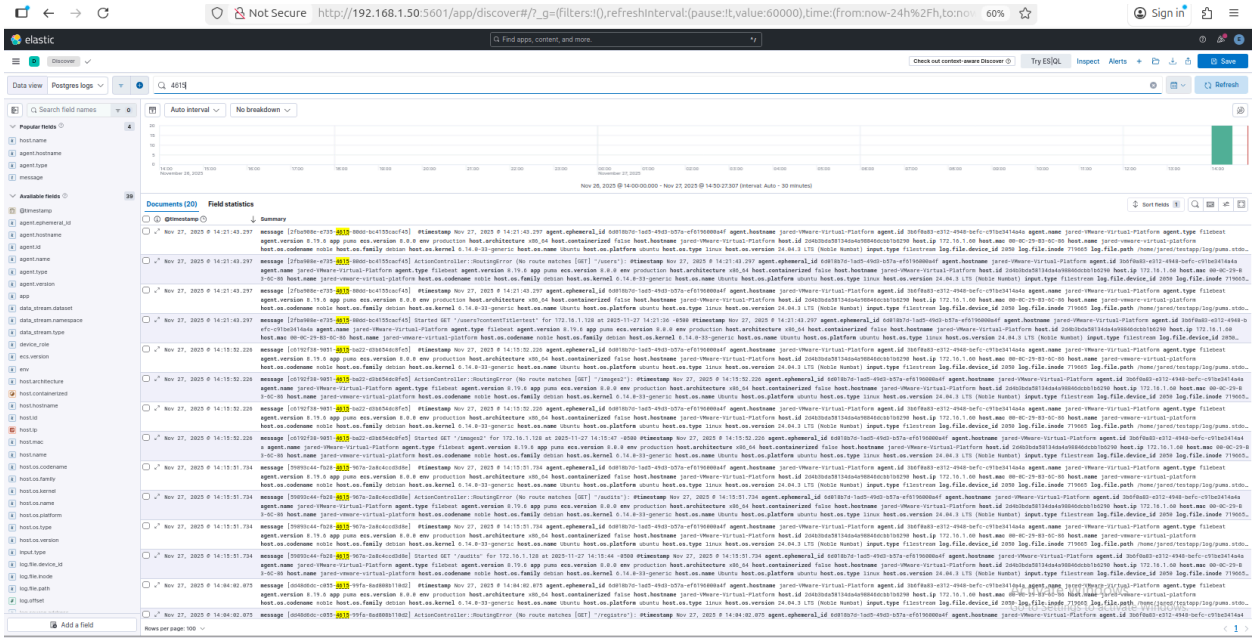
28. Win 10 and 11 Attacks



Nmap scan to see vulnerabilities in the vmnet 3 machines.



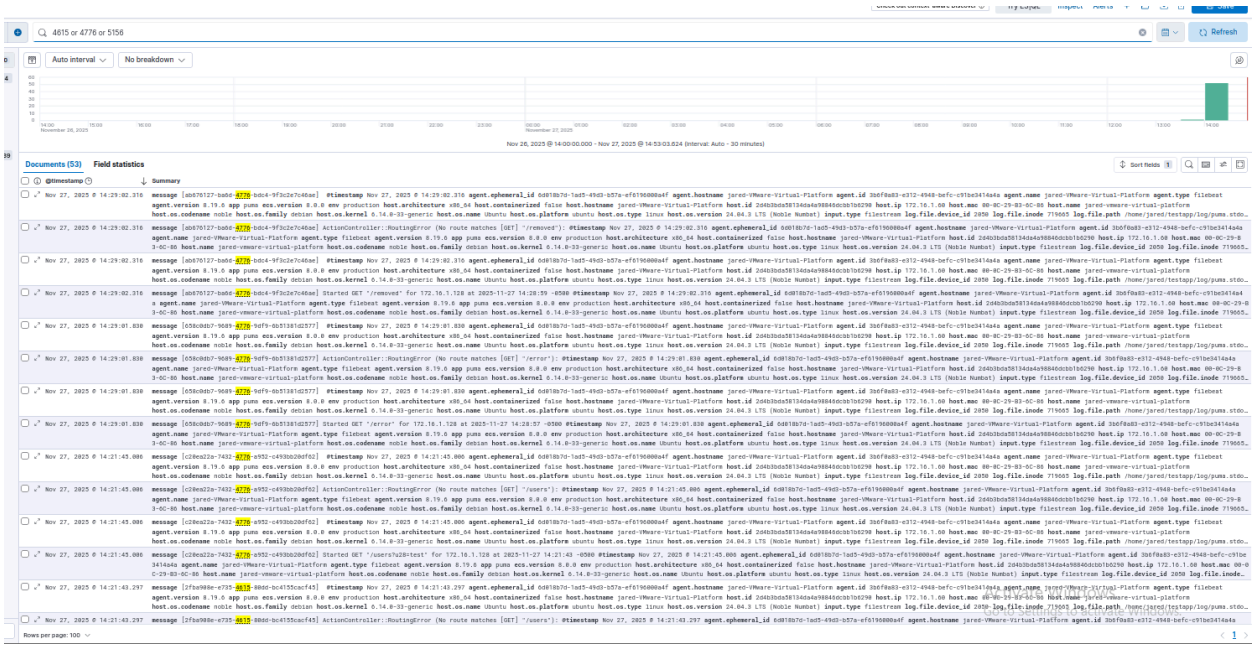
we used a command to try a SMB enumeration attempts that were failed due to windows firewall(good)



Here we see those failed attapmts, it uses port 445

We use these attacks on the windows machines that tries to get as much info as it can from the windows firewall and system.

enum4linux -a 10.10.1.73 and enum4linux -a 10.10.1.136

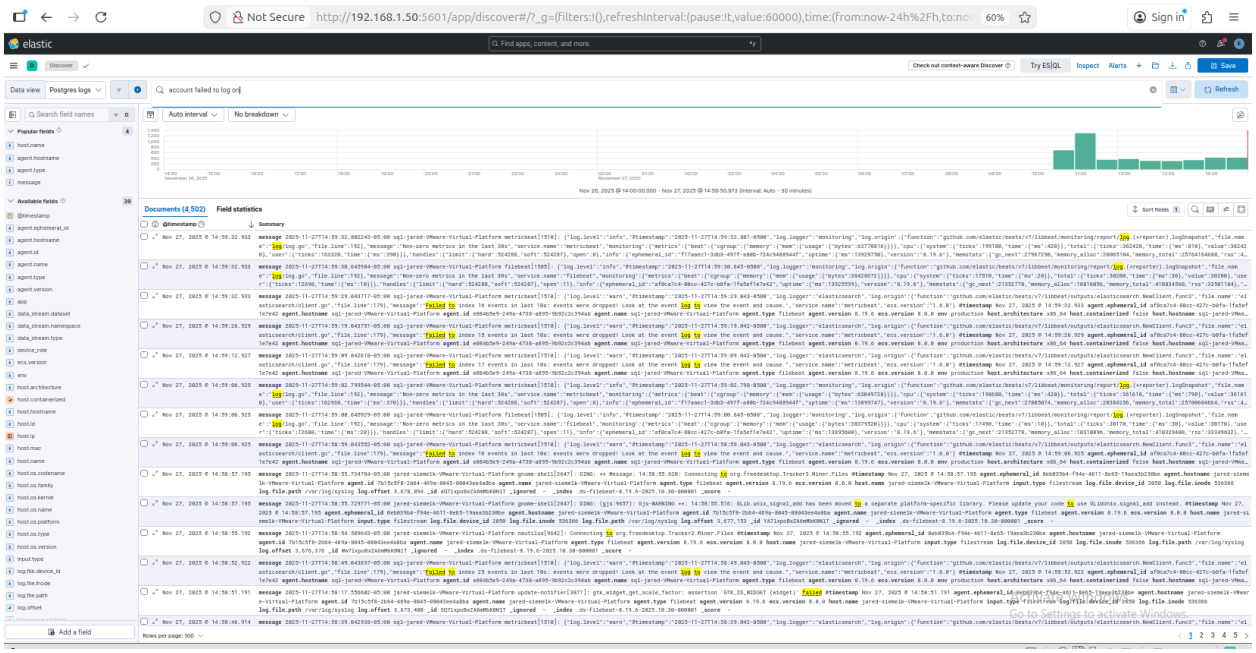


here we saw those in the attempts on ports 445, 4625, 5108, and 139

```
┌──(kali㉿kali)-[~]
└─$ hydra -l administrator -P /usr/share/wordlists/rockyou.txt rdp://10.10.1.73

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizatior

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-27 14:57:40
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connectior
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking rdp://10.10.1.73:3389/
[ERROR] freerdp: The connection failed to establish.
[ERROR] freerdp: The connection failed to establish.
[ERROR] freerdp: The connection failed to establish.
[ERROR] freerdp: The connection failed to establish.
[ERROR] all children were disabled due too many connection errors
0 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-27 14:58:14

┌──(kali㉿kali)-[~]
└─$ hydra -l administrator -P /usr/share/wordlists/rockyou.txt rdp://10.10.1.136
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizatior

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-27 14:58:28
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel connectior
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking rdp://10.10.1.136:3389/
[ERROR] freerdp: The connection failed to establish.
[ERROR] freerdp: The connection failed to establish.
[ERROR] freerdp: The connection failed to establish.
[ERROR] freerdp: The connection failed to establish.
[ERROR] all children were disabled due too many connection errors
0 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-27 14:59:01
```

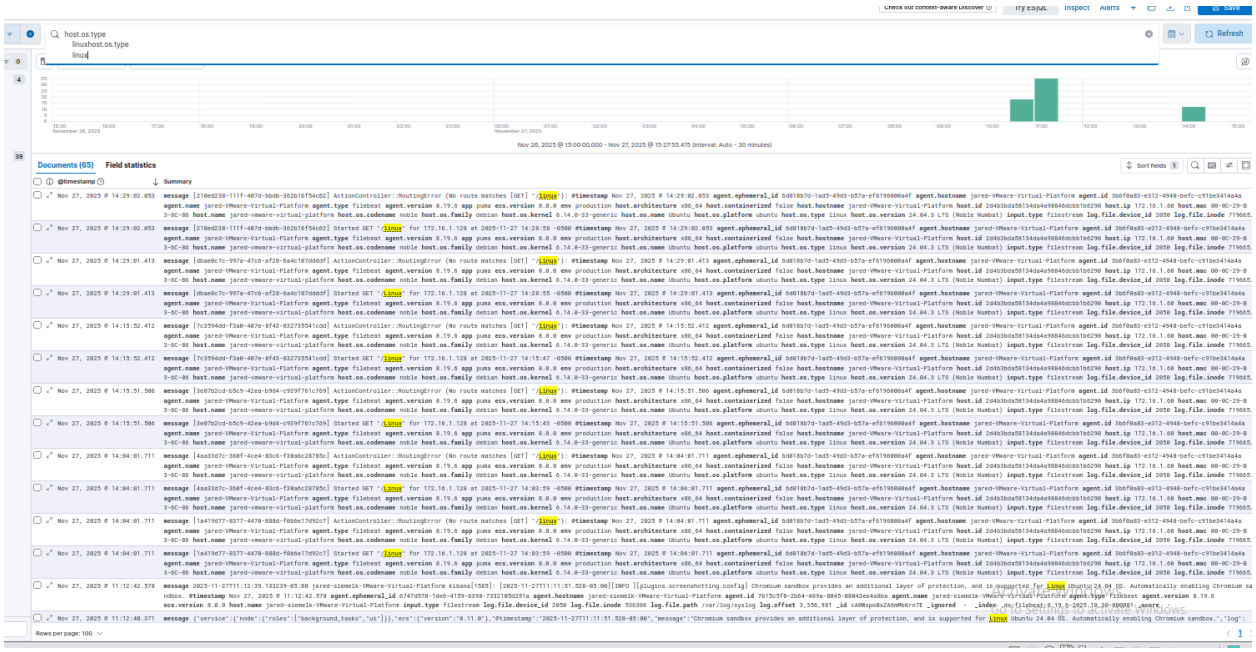
Hydra attempted to brute-force the Windows RDP login service (port 3389) using:

Username: administrator and Password list: rockyou.txt



Here we saw those attempts on the port search.

Using multiple Metasploit auxiliary modules against the two Windows machines. Attempted enumeration of SMB shares which fails

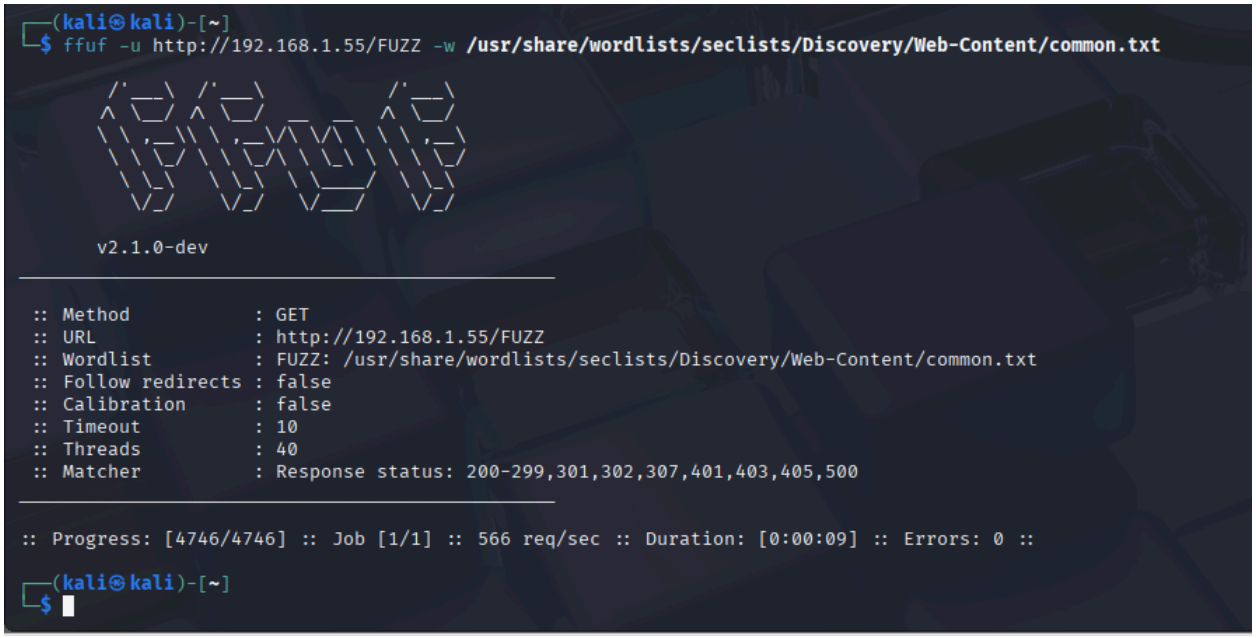


here i search for any linux asking for information.

29. Pihole attacks

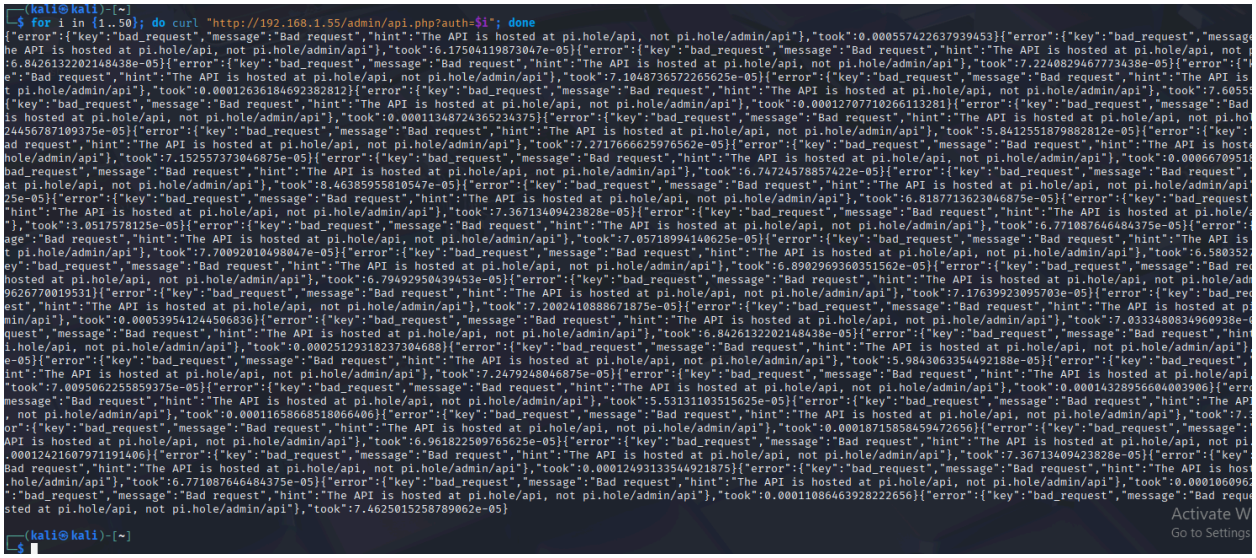
```
[*] exec: sudo nmap -sS -sV -p- 192.168.1.50 192.168.1.55

[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-27 15:29 EST
Nmap scan report for 192.168.1.50
Host is up (0.00060s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE    VERSION
5601/tcp open  http       Elasticsearch Kibana (serverName: jared-siemelk-VMware-Virtual-Platform)
9200/tcp open  ssl/http   Elasticsearch REST API 7.0 or later (Shield plugin; realm: security)
9300/tcp open  ssl/vrace?
MAC Address: 00:0C:29:14:B1:A9 (VMware)

Nmap scan report for 192.168.1.55
Host is up (0.00064s latency).
Not shown: 65532 closed tcp ports (reset)
PORT    STATE SERVICE    VERSION
53/tcp  open  domain     dnsmasq 2.92test21 (pi-hole)
80/tcp  open  webdav
443/tcp open  ssl/webdav
```

nmap on vmnet 1 to see vulnerabilities.



using FFUF which is a fast file brute attacker to discover hidden content where I generated content showing it tried.

In kibana we saw all the failed attempts on the machine.

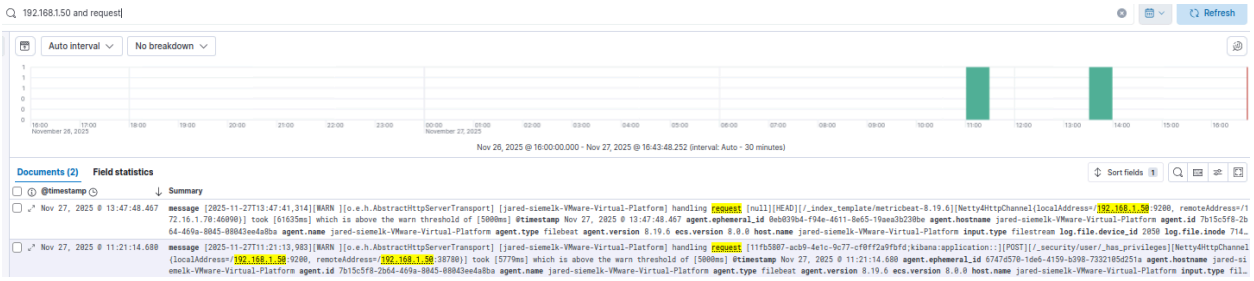


This is an api attack on the pihole machine.

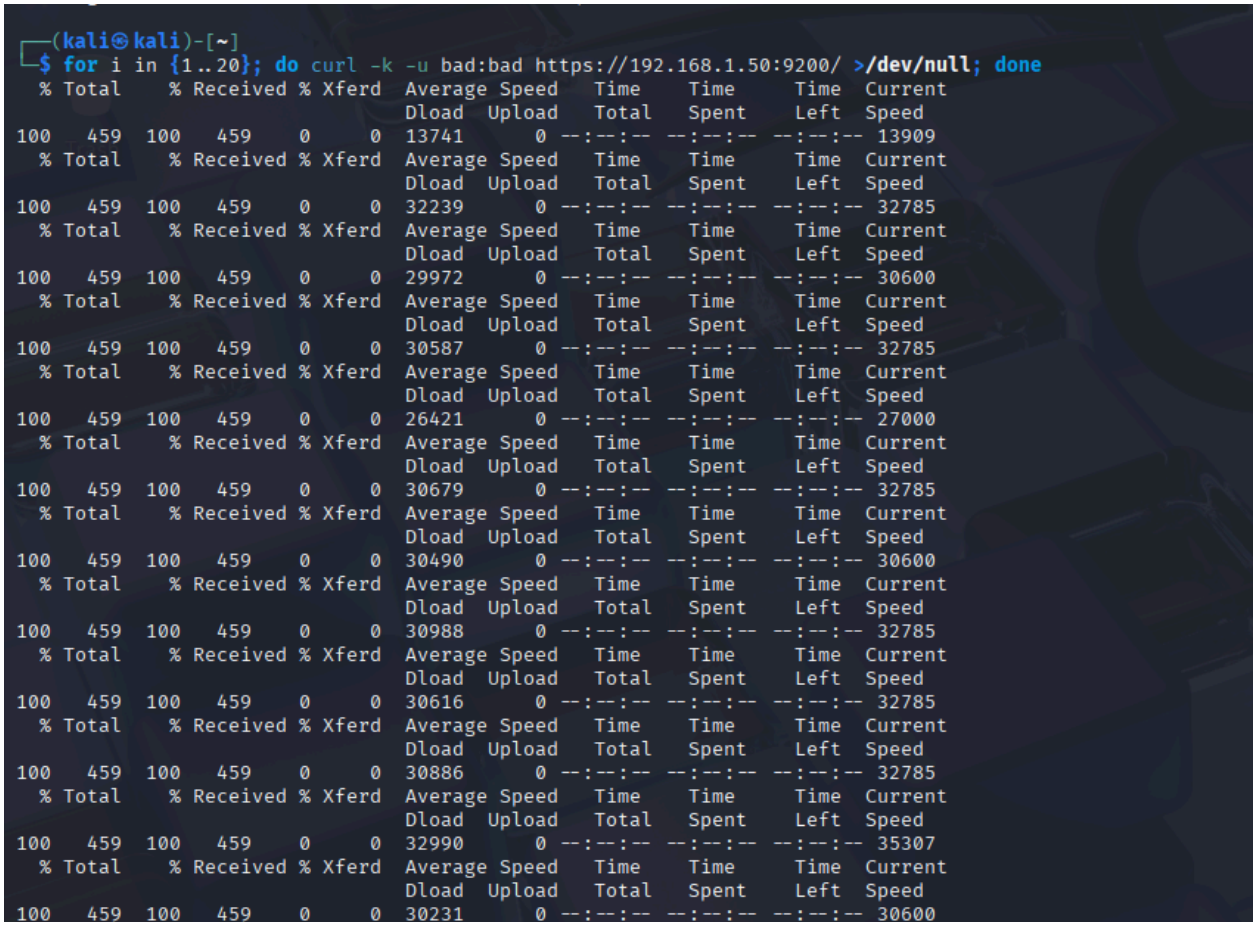
Here we see all those attacks where it seems to get api info.

30. ELK attack

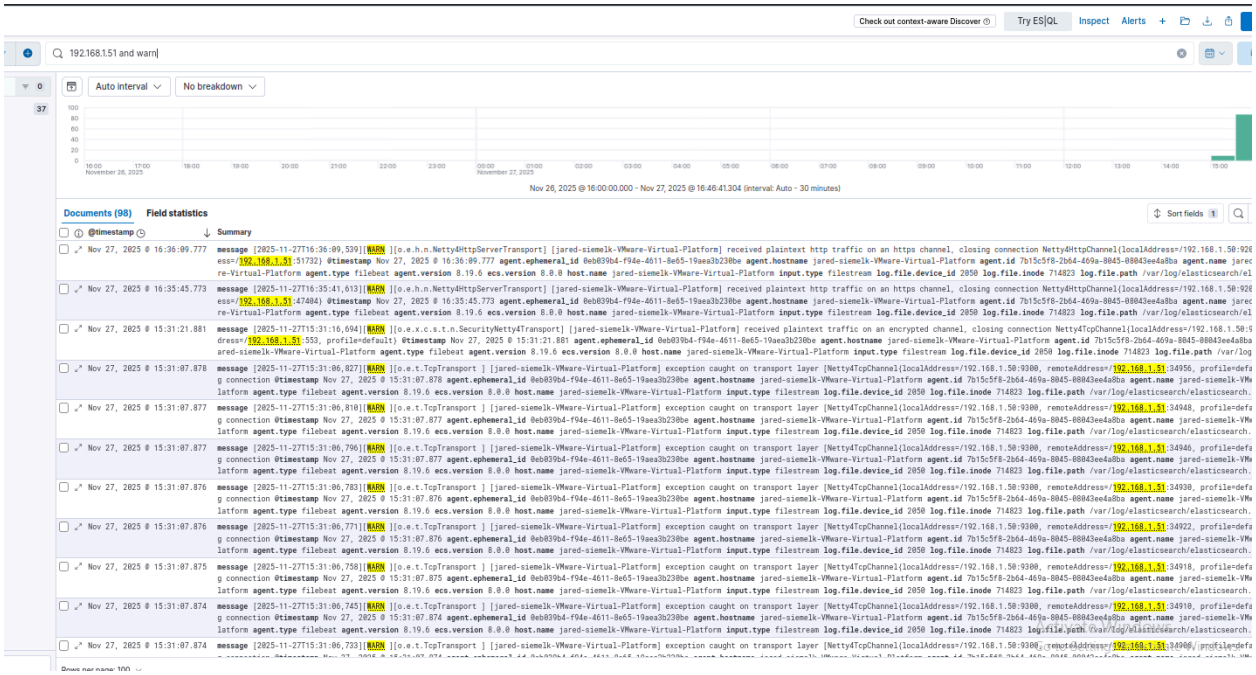
Another FUFF attack on the ELK machine.



Here we saw all the requests on the machine.



I used a repeating credential stuffing(login) attack to the ELK machine



I got to see all these attacks and there were so many logs here.

I used an attack to simulate an attacker trying to enumerate the cluster's indices without valid credentials.

We saw those attacks and even went further to see what they stated when it got them.

9. Conclusions and Findings

Working through the attacks and monitoring everything inside the SIEM taught me how predictable most attacker behavior actually is once logs are seen. Tools like Gobuster, FFUF, Nmap, and SMB scanners created obvious patterns in Kibana. Huge spikes of repeated requests that stood out immediately. Even authentication attacks were easy to spot, because every failed login, bad password, or rejected connection sent tons of logs. Seeing this in real time helped me understand why SIEMs are so important.I also realized that once you know what “normal” traffic looks like in your environment, spotting something unusual becomes a lot easier.

On the blue team/defensive side, the project helped me see several improvements that could make the network stronger. pfSense already separated different parts of the network, but enforcing stricter “deny” rules in pfSense and limiting traffic between only the subnets would make it even harder for an attacker to move around and view things. Adding Suricata would give the firewall the ability not only to filter traffic but to recognize things like port scans or brute force attempts as they happen.

Finally, Kibana gave me the ability to visualize attacks, but refining the dashboards and enabling automated alerts would take the setup to the next level. Tracking authentication failures, unusual network activity, DNS filtering events, and spikes in traffic would allow defenders to react quickly instead of relying on manual monitoring. Overall, this project showed me how important visibility, segmentation, and hardening are to any network. The attacks helped me see how an attacker thinks and the SIEM helped me understand how a defender can stay one step ahead.