

ELK Stack SIEM Implementation

A hands-on SOC lab built on Ubuntu — Elasticsearch · Kibana · Logstash · Nginx

Project Overview

I built this project to get hands-on experience with the ELK Stack by setting up a home lab that collects, processes, and visualises real system data. The stack runs on a local Ubuntu VM and includes Elasticsearch for storage and indexing, Kibana for visualisation, Logstash as the data processing pipeline, Auditbeat to collect system events, and Nginx as a reverse proxy with authentication in front of Kibana. The goal was to go through the full setup from scratch and end up with actual data flowing through the pipeline — not just installed services, but something working end to end.

Virtual Machine Setup

I ran everything inside a dedicated VM to keep the setup isolated from my host machine. Elasticsearch is particularly memory-hungry, so I wanted to give it a clean environment with enough resources to actually run without constantly swapping. I used VMware Fusion with the following specs:

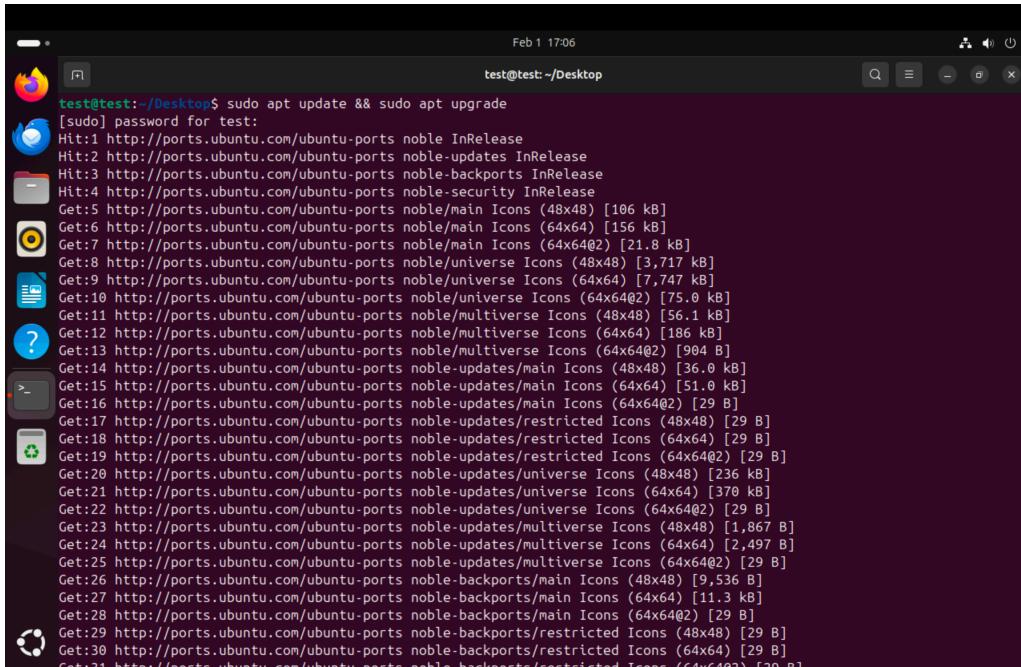
- **Platform:** VMware Fusion
- **OS:** Ubuntu 22.04 LTS
- **Storage:** 50 GB — Elasticsearch index files grow quickly, so you need the space
- **RAM:** 4 GB — the minimum to run Elasticsearch and Kibana together without issues

I went with Ubuntu 22.04 LTS because it's stable, well-supported, and Elastic maintains an official apt repository for it, which makes installing and updating the stack much cleaner than doing it manually.

Step 1 — System Update

Before installing anything, I updated the system to make sure all existing packages were current. This matters because outdated packages can cause dependency conflicts when you start adding third-party repositories like Elastic's, and it also keeps the base system clean before layering anything on top.

```
sudo apt update && sudo apt upgrade
```



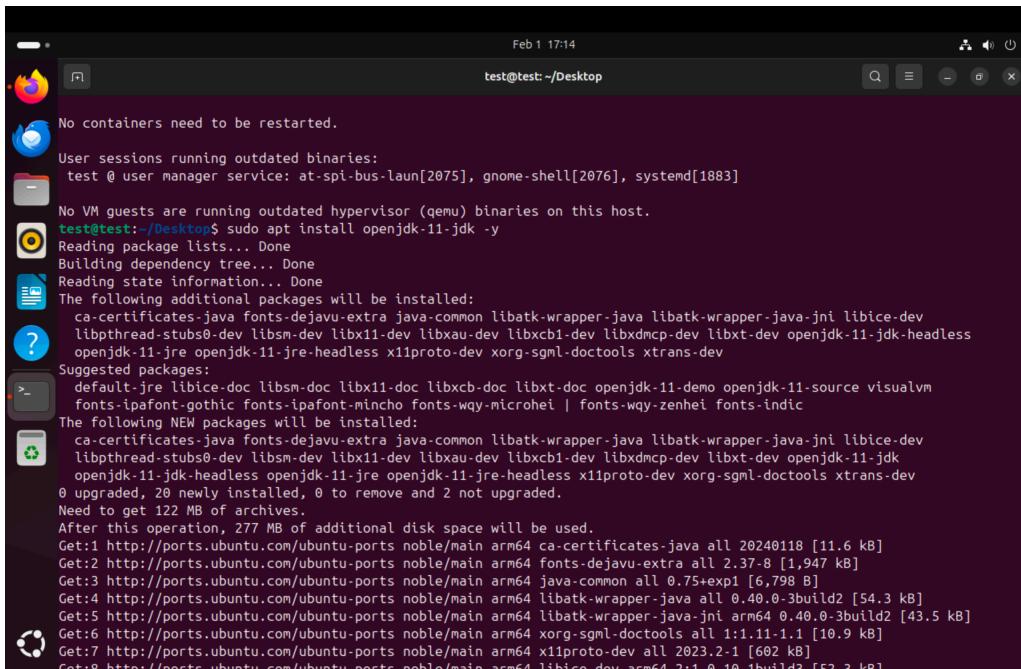
```
Feb 1 17:06
test@test:~/Desktop$ sudo apt update && sudo apt upgrade
[sudo] password for test:
Hit:1 http://ports.ubuntu.com/ubuntu-ports noble InRelease
Hit:2 http://ports.ubuntu.com/ubuntu-ports noble-updates InRelease
Hit:3 http://ports.ubuntu.com/ubuntu-ports noble-backports InRelease
Hit:4 http://ports.ubuntu.com/ubuntu-ports noble/main Icons (48x48) [106 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports noble/main Icons (64x64) [156 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports noble/main Icons (64x64@2) [21.8 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports noble/main Icons (48x48@2) [3,717 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports noble/universe Icons (48x48) [3,717 kB]
Get:9 http://ports.ubuntu.com/ubuntu-ports noble/universe Icons (64x64) [7,747 kB]
Get:10 http://ports.ubuntu.com/ubuntu-ports noble/universe Icons (64x64@2) [75.0 kB]
Get:11 http://ports.ubuntu.com/ubuntu-ports noble/multiverse Icons (48x48) [56.1 kB]
Get:12 http://ports.ubuntu.com/ubuntu-ports noble/multiverse Icons (64x64) [186 kB]
Get:13 http://ports.ubuntu.com/ubuntu-ports noble/multiverse Icons (64x64@2) [904 kB]
Get:14 http://ports.ubuntu.com/ubuntu-ports noble-updates/main Icons (48x48) [36.0 kB]
Get:15 http://ports.ubuntu.com/ubuntu-ports noble-updates/main Icons (64x64) [51.0 kB]
Get:16 http://ports.ubuntu.com/ubuntu-ports noble-updates/main Icons (64x64@2) [29 kB]
Get:17 http://ports.ubuntu.com/ubuntu-ports noble-updates/restricted Icons (48x48) [29 kB]
Get:18 http://ports.ubuntu.com/ubuntu-ports noble-updates/restricted Icons (64x64) [29 kB]
Get:19 http://ports.ubuntu.com/ubuntu-ports noble-updates/restricted Icons (64x64@2) [29 kB]
Get:20 http://ports.ubuntu.com/ubuntu-ports noble-updates/universe Icons (48x48) [236 kB]
Get:21 http://ports.ubuntu.com/ubuntu-ports noble-updates/universe Icons (64x64) [370 kB]
Get:22 http://ports.ubuntu.com/ubuntu-ports noble-updates/universe Icons (64x64@2) [29 kB]
Get:23 http://ports.ubuntu.com/ubuntu-ports noble-updates/multiverse Icons (48x48) [1,867 kB]
Get:24 http://ports.ubuntu.com/ubuntu-ports noble-updates/multiverse Icons (64x64) [2,497 kB]
Get:25 http://ports.ubuntu.com/ubuntu-ports noble-updates/multiverse Icons (64x64@2) [29 kB]
Get:26 http://ports.ubuntu.com/ubuntu-ports noble-backports/main Icons (48x48) [9,536 kB]
Get:27 http://ports.ubuntu.com/ubuntu-ports noble-backports/main Icons (64x64) [11.3 kB]
Get:28 http://ports.ubuntu.com/ubuntu-ports noble-backports/main Icons (64x64@2) [29 kB]
Get:29 http://ports.ubuntu.com/ubuntu-ports noble-backports/restricted Icons (48x48) [29 kB]
Get:30 http://ports.ubuntu.com/ubuntu-ports noble-backports/restricted Icons (64x64) [29 kB]
Get:31 http://ports.ubuntu.com/ubuntu-ports noble-backports/restricted Icons (64x64@2) [29 kB]
```

System update running — fetching latest packages

Step 2 — Install Java

Elasticsearch runs on the JVM, so Java needs to be installed first. The version matters here — Elasticsearch 8.x works with OpenJDK 11, so that's what I installed. Using a different version can cause the service to fail on startup without a very obvious error message, so it's worth pinning to the right one from the start.

```
sudo apt install openjdk-11-jdk -y
```



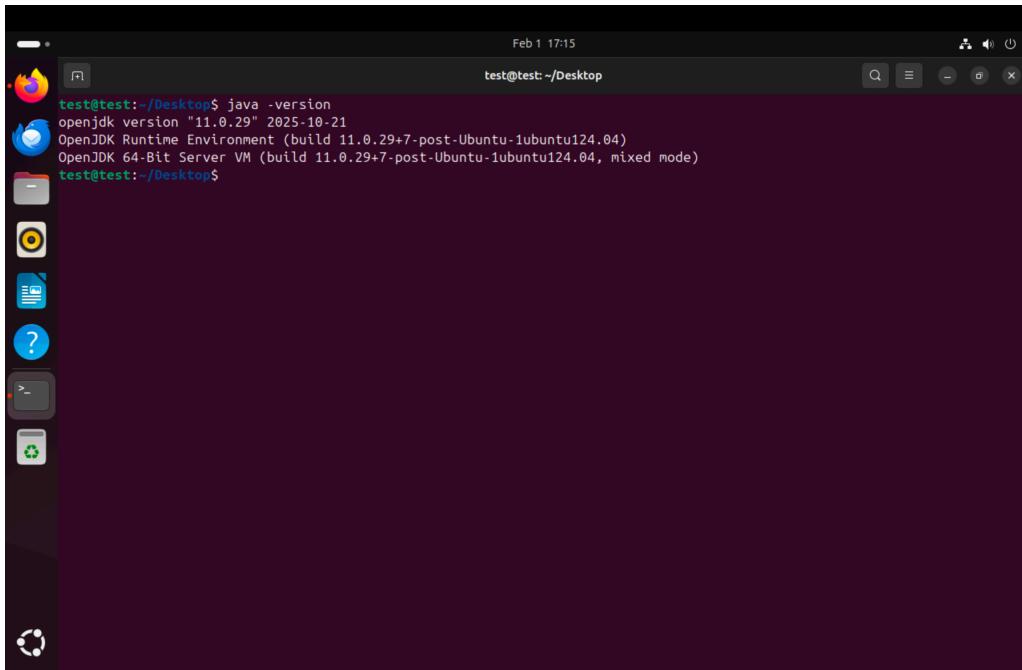
The screenshot shows a terminal window titled "test@test: ~/Desktop". The terminal output is as follows:

```
Feb 1 17:14
test@test:~/Desktop
No containers need to be restarted.
User sessions running outdated binaries:
  test @ user manager service: at-spi-bus-laun[2075], gnome-shell[2076], systemd[1883]
No VM guests are running outdated hypervisor (qemu) binaries on this host.
test@test:~/Desktop$ sudo apt install openjdk-11-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libice-dev
  libpthread-stubs0-dev libsm-dev libxi1-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk-headless
  openjdk-11-jre openjdk-11-jre-headless x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  default-jre libice-doc libsm-doc libxi1-doc libxcb-doc libxt-doc openjdk-11-demo openjdk-11-source visualvm
  fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni libice-dev
  libpthread-stubs0-dev libsm-dev libxi1-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11proto-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 20 newly installed, 0 to remove and 2 not upgraded.
Need to get 122 MB of archives.
After this operation, 277 MB of additional disk space will be used.
Get:1 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 ca-certificates-java all 20240118 [11.6 kB]
Get:2 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 fonts-dejavu-extra all 2.37-8 [1,947 kB]
Get:3 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 java-common all 0.75+exp1 [6,798 B]
Get:4 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 libatk-wrapper-java all 0.40.0-3build2 [54.3 kB]
Get:5 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 libatk-wrapper-java-jni arm64 0.40.0-3build2 [43.5 kB]
Get:6 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 xorg-sgml-doctools all 1:1.11-1.1 [10.9 kB]
Get:7 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 x11proto-dev all 2023.2-1 [602 kB]
Get:8 http://ports.ubuntu.com/ubuntu-ports noble/main arm64 libice-dev arm64 2.1.0-10.1build3 [52.3 kB]
```

OpenJDK 11 installing

After the install, I verified it was picked up correctly. If you have multiple Java versions on the system this is especially important, since the wrong one being active will cause Elasticsearch to fail silently.

```
java -version
```

A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and white text. It displays the command 'java -version' and its output, which shows OpenJDK version 11.0.29. The desktop interface includes a dock with icons for various applications like a browser, file manager, and terminal, and a panel at the top with system status indicators.

OpenJDK 11.0.29 — correct version confirmed

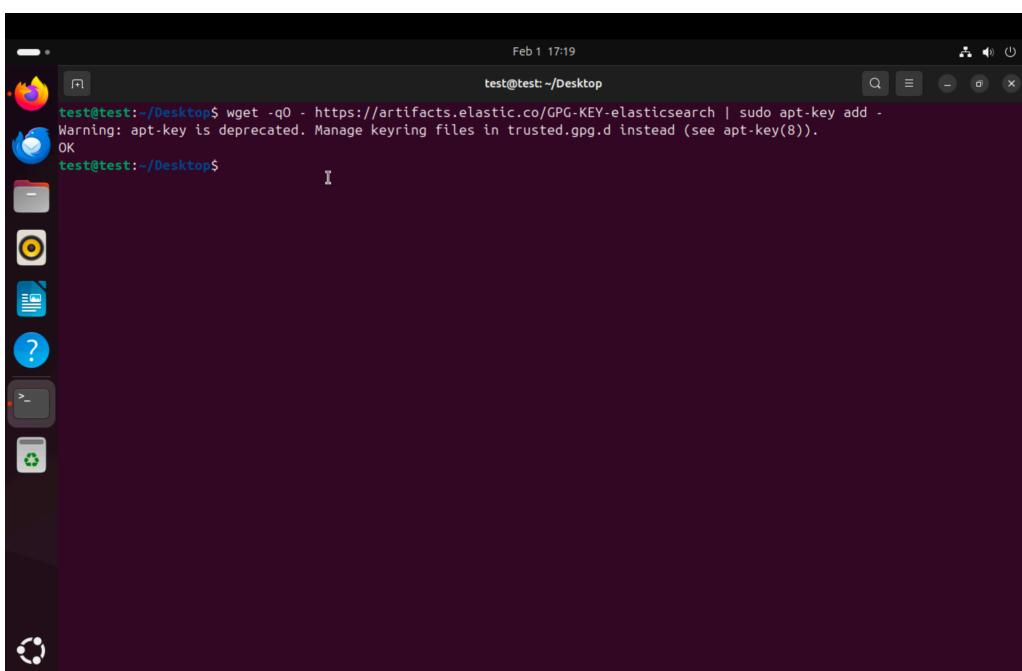
Step 3 — Install Elasticsearch

Elasticsearch is the core of the whole setup, it's where all the data lives. Everything else either pushes data into it or reads data out of it. Rather than downloading a package manually, I added Elastic's official apt repository so I could install it properly and get updates through the normal package manager going forward.

Add the Elastic repository

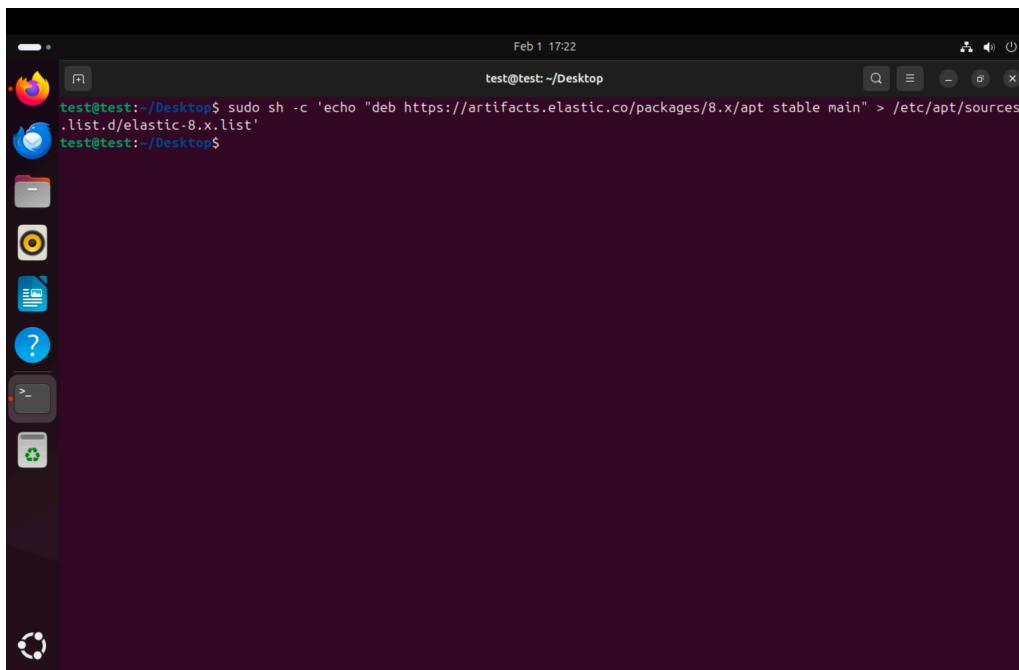
First I added the GPG key so apt can verify the packages actually come from Elastic, then registered the repository itself. Without the GPG key, apt will refuse to install from that source.

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

A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and white text. It displays the command 'wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -' and its output, which includes a warning about the deprecation of apt-key and the command 'OK'. The desktop interface includes a dock with icons for various applications like a browser, file manager, and terminal, and a panel at the top with system status indicators.

GPG key added — OK

```
sudo sh -c 'echo "deb https://artifacts.elastic.co/packages/8.x/apt stable main" > /etc/apt/sources.list.d/elastic-8.x.list'
```

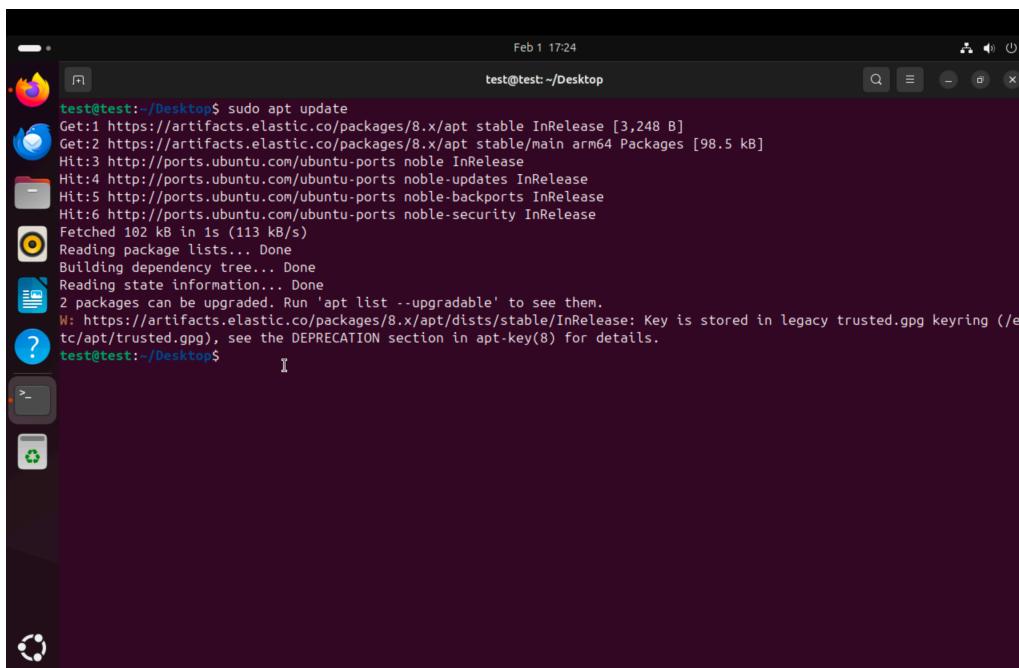


A screenshot of a terminal window titled "test@test:~/Desktop". The window shows the command "sudo sh -c 'echo \"deb https://artifacts.elastic.co/packages/8.x/apt stable main\" > /etc/apt/sources.list.d/elastic-8.x.list'" being run. The terminal is dark-themed with white text. The desktop environment includes icons for a file manager, terminal, and system settings.

Elastic 8.x repository added to apt sources

Then refresh the package list so apt picks up what's available from the new repository:

```
sudo apt update
```

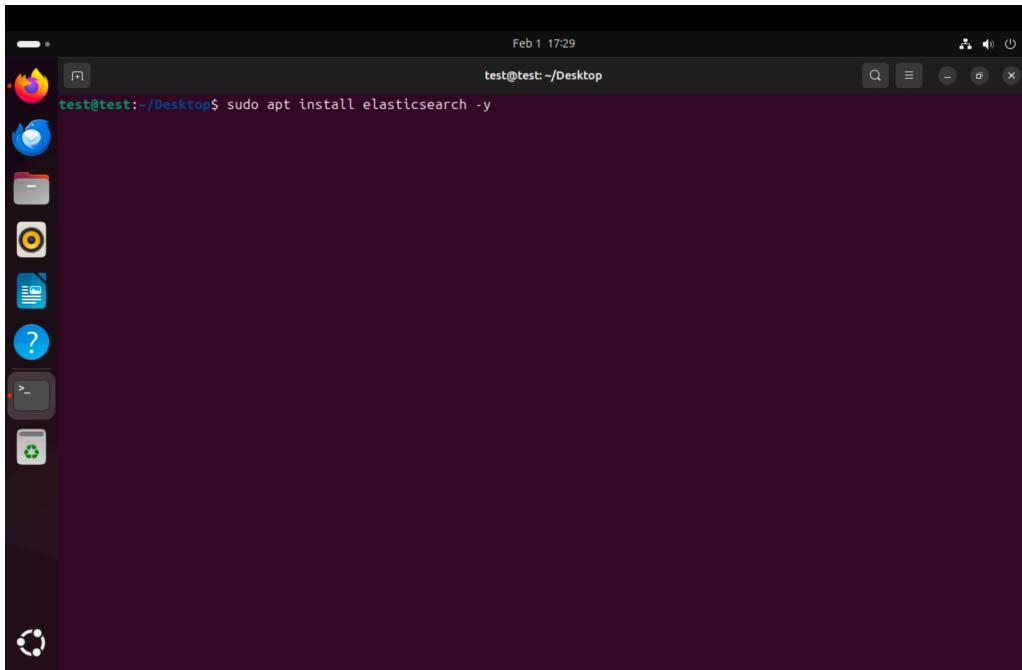


A screenshot of a terminal window titled "test@test:~/Desktop". The window shows the output of the "sudo apt update" command. It lists several package sources being checked and updated, including "https://artifacts.elastic.co/packages/8.x/apt", "http://ports.ubuntu.com/ubuntu-ports", and "http://ports.ubuntu.com/ubuntu-ports noble-backports". The terminal is dark-themed with white text. The desktop environment includes icons for a file manager, terminal, and system settings.

apt now sees packages from artifacts.elastic.co

Install

```
sudo apt install elasticsearch -y
```



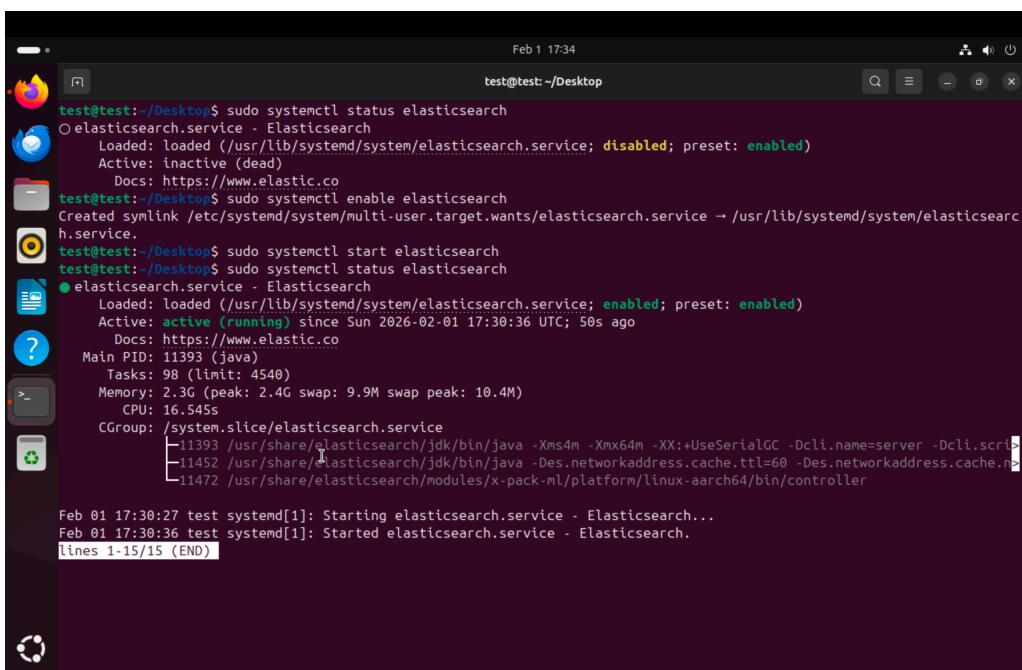
```
test@test:~/Desktop$ sudo apt install elasticsearch -y
```

Elasticsearch downloading and installing

Enable and start

Elasticsearch doesn't start automatically after install, which is intentional, you're expected to configure it first. I enabled it so it starts on reboot, then started it manually, and checked the status to make sure it came up cleanly.

```
sudo systemctl enable elasticsearch  
sudo systemctl start elasticsearch  
sudo systemctl status elasticsearch
```



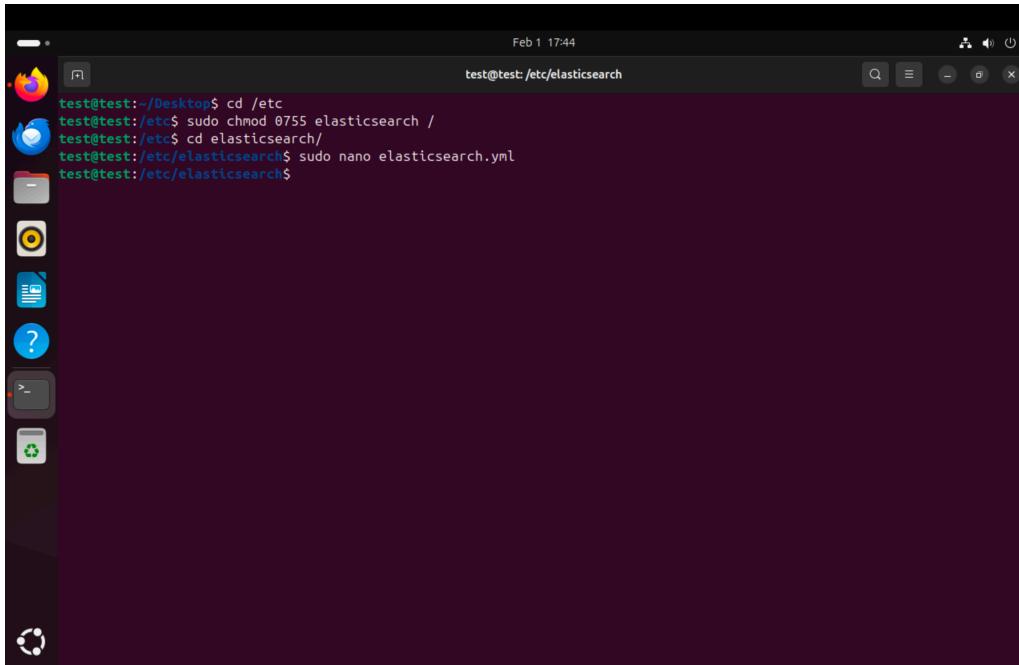
```
test@test:~/Desktop$ sudo systemctl status elasticsearch  
● elasticsearch.service - Elasticsearch  
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; disabled; preset: enabled)  
     Active: inactive (dead)  
       Docs: https://www.elastic.co  
test@test:~/Desktop$ sudo systemctl enable elasticsearch  
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service → /usr/lib/systemd/system/elasticsearch.service.  
test@test:~/Desktop$ sudo systemctl start elasticsearch  
test@test:~/Desktop$ sudo systemctl status elasticsearch  
● elasticsearch.service - Elasticsearch  
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; preset: enabled)  
     Active: active (running) since Sun 2026-02-01 17:30:36 UTC; 50s ago  
       Docs: https://www.elastic.co  
         Main PID: 11393 (java)  
            Tasks: 98 (limit: 4540)  
           Memory: 2.3G (peak: 2.4G swap: 9.9M swap peak: 10.4M)  
             CPU: 16.545s  
            CGroup: /system.slice/elasticsearch.service  
                      └─11393 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.script=true  
                         ├─11452 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.negative.ttl=10  
                         ├─11472 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-aarch64/bin/controller  
  
Feb 01 17:30:27 test systemd[1]: Starting elasticsearch.service - Elasticsearch...  
Feb 01 17:30:36 test systemd[1]: Started elasticsearch.service - Elasticsearch.  
lines 1-15/15 (END)
```

Elasticsearch active and running

Configure — elasticsearch.yml

The default config is bare-bones and won't get the cluster to form properly. I edited the config file to set the cluster name, node name, and the network settings Elasticsearch needs to bootstrap itself as a single-node cluster.

```
cd /etc && sudo chmod 0755 elasticsearch/
cd elasticsearch/
sudo nano elasticsearch.yml
```



A screenshot of a terminal window titled "test@test:/etc/elasticsearch". The window shows the command history:

```
test@test:~/Desktop$ cd /etc
test@test:/etc$ sudo chmod 0755 elasticsearch /
test@test:/etc$ cd elasticsearch/
test@test:/etc/elasticsearch$ sudo nano elasticsearch.yml
test@test:/etc/elasticsearch$
```

Opening elasticsearch.yml

Here's what I changed and why each setting is needed:

- **cluster.name: elkstack** : a name for the cluster; nodes only join if the cluster name matches
- **node.name: test** : a name for this specific node, which shows up in logs and the API response
- **discovery.seed_hosts: ["192.168.160.133"]** : where to look for other nodes on startup; in a single-node setup this just points to itself
- **cluster.initial_master_nodes: ["test"]** : required for the first bootstrap; without it, Elasticsearch won't elect a master and the cluster stays stuck
- **http.port: 9200** : the port for REST API requests; this is the default and what Kibana will connect to

```
GNU nano 7.2         .elasticsearch.yml *
# ===== Elasticsearch Configuration =====#
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please consult the documentation for further information on configuration options:
# https://www.elastic.co/guide/en/elasticsearch/reference/index.html
#
# -----
# Cluster
#
# Use a descriptive name for your cluster:
#
cluster.name: elkstack
#
# -----
# Node
#
# Use a descriptive name for the node:
#
node.name: test
#
# Add custom attributes to the node:
#
```

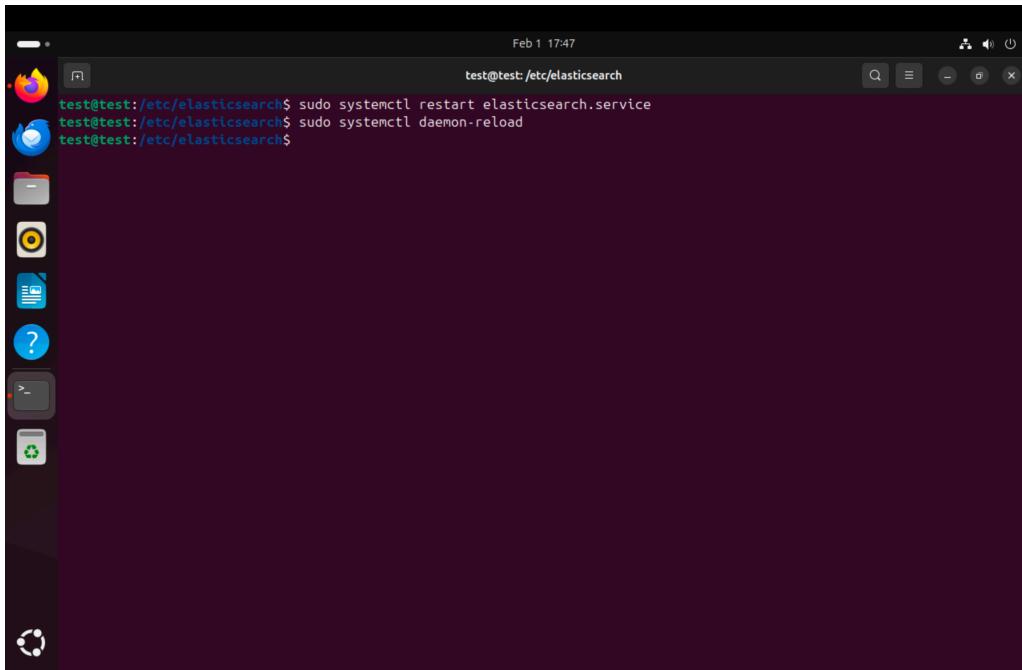
elasticsearch.yml — cluster.name and node.name set

```
GNU nano 7.2         .elasticsearch.yml *
http.port: 9200
#
# For more information, consult the network module documentation.
#
# -----
# Discovery
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.seed_hosts: ["192.168.160.133"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
cluster.initial_master_nodes: ["test"]
#
# For more information, consult the discovery and cluster formation module documentation.
#
# -----
# Various
#
# Allow wildcard deletion of indices:
#
#action.destructive_requires_name: false
#
#
#xpack.security.enabled : false
```

elasticsearch.yml — discovery and master node settings

After saving, I restarted the service to apply the changes:

```
sudo systemctl restart elasticsearch.service
sudo systemctl daemon-reload
```

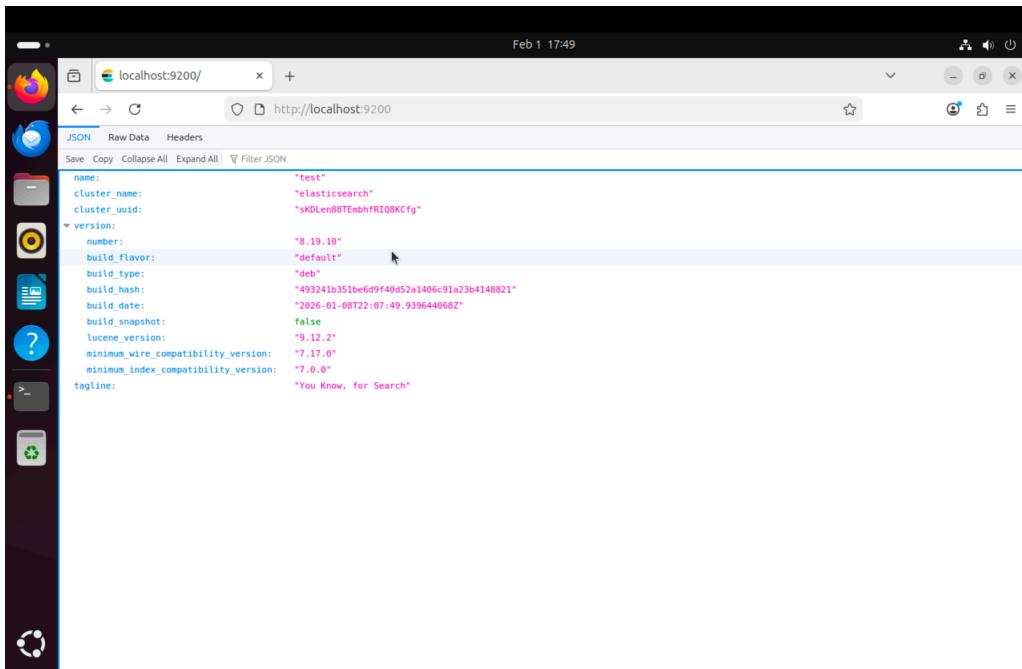


```
test@test:/etc/elasticsearch$ sudo systemctl restart elasticsearch.service
test@test:/etc/elasticsearch$ sudo systemctl daemon-reload
test@test:/etc/elasticsearch$
```

Elasticsearch restarted after config change

Verify

I opened localhost:9200 in the browser to confirm Elasticsearch was up and responding. If you get a JSON response with the cluster name and version, the node formed correctly. If the page doesn't load, something in the config is off — check the service logs with journalctl -u elasticsearch.



The screenshot shows a browser window with the URL `http://localhost:9200`. The page displays a JSON response from the Elasticsearch cluster. The response includes the cluster name, UUID, and detailed version information:

```
name: "test"
cluster_name: "elasticsearch"
cluster_uuid: "skDLen88TembhfRI08KCfg"
version:
  number: "8.19.10"
  build_flavor: "default"
  build_type: "deb"
  build_hash: "493241b351be6df40d52a1406c91a23b4148821"
  build_date: "2026-01-08T22:07:49.939644068Z"
  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"
```

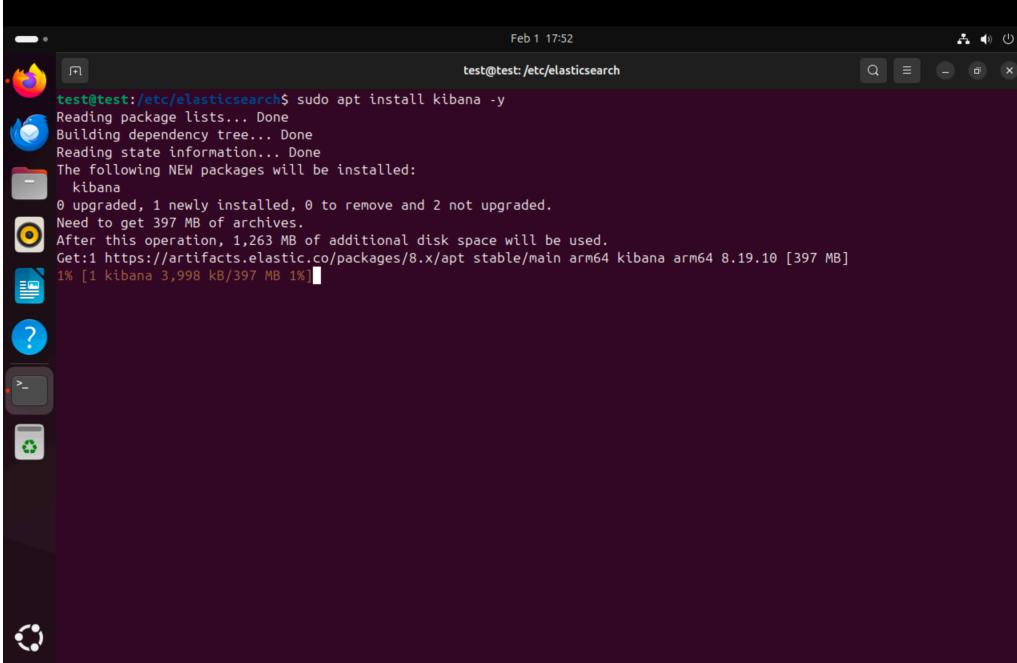
Elasticsearch 8.19.10 responding at localhost:9200

Step 4 — Install Kibana

Kibana is the web interface for the stack, it connects to Elasticsearch and gives you a browser-based UI to search and visualise all the data that's been indexed. Since I already added the Elastic repository, installing Kibana is one command. The important thing is to make sure it's the same version as Elasticsearch — mismatched versions cause connection errors that aren't always obvious.

Install

```
sudo apt install kibana -y
```



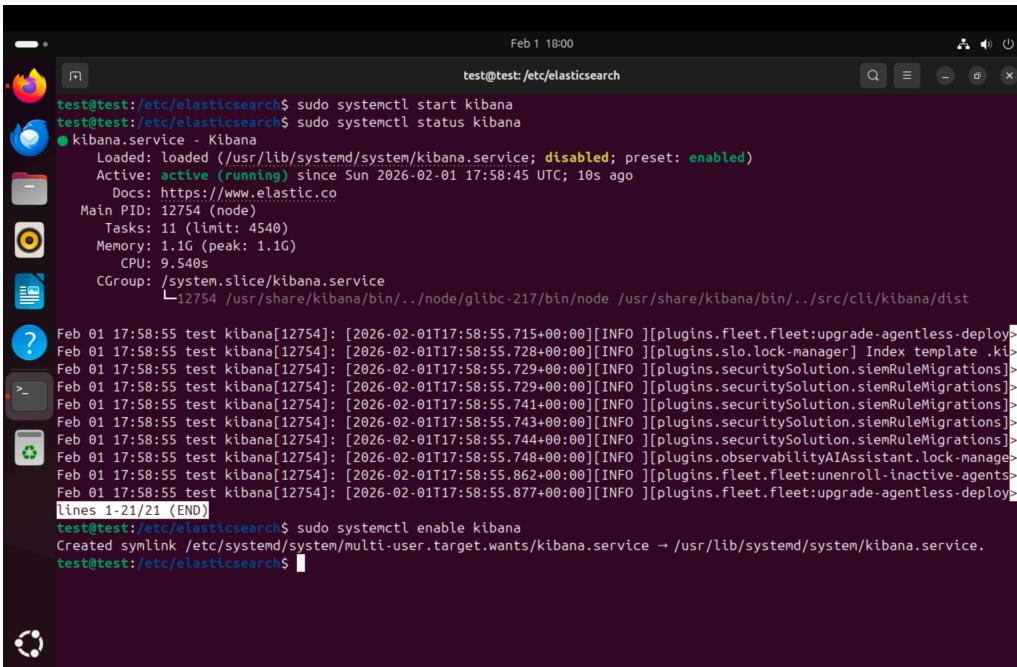
```
test@test:/etc/elasticsearch$ sudo apt install kibana -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  kibana
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 397 MB of archives.
After this operation, 1,263 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt/stable/main arm64 kibana arm64 8.19.10 [397 MB]
1% [1 kibana 3,998 kB/397 MB 1%]
```

Kibana 8.19.10 downloading — matching Elasticsearch version

Start and enable

Kibana is a Node.js app and takes a bit longer to start than most services because it needs to connect to Elasticsearch and set up its own internal indices. Give it a minute or two before trying to open it in the browser.

```
sudo systemctl start kibana
sudo systemctl status kibana
sudo systemctl enable kibana
```



```
test@test:/etc/elasticsearch$ sudo systemctl start kibana
test@test:/etc/elasticsearch$ sudo systemctl status kibana
● kibana.service - Kibana
   Loaded: loaded (/usr/lib/systemd/system/kibana.service; disabled; preset: enabled)
   Active: active (running) since Sun 2026-02-01 17:58:45 UTC; 10s ago
     Docs: https://www.elastic.co
          >Main PID: 12754 (node)
             Tasks: 11 (limit: 4540)
           Memory: 1.1G (peak: 1.1G)
            CPU: 9.540s
          CGroup: /system.slice/kibana.service
                  └─12754 /usr/share/kibana/bin/..../node/glibc-217/bin/node /usr/share/kibana/bin/..../src/cli/kibana/dist

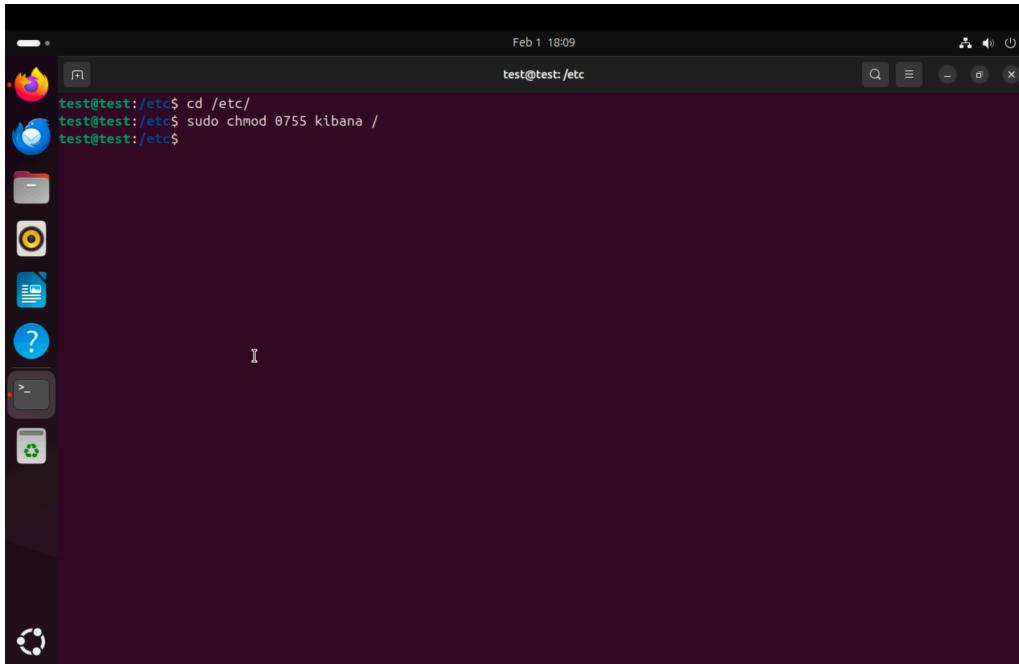
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.715+00:00][INFO ][plugins.fleet.fleetupgrade-agentless-deploy]>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.728+00:00][INFO ][plugins.slo.lock-manager] Index template .ki>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.729+00:00][INFO ][plugins.securitysolution.siemrulemigrations]>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.729+00:00][INFO ][plugins.securitysolution.siemrulemigrations]>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.741+00:00][INFO ][plugins.securitysolution.siemrulemigrations]>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.743+00:00][INFO ][plugins.securitysolution.siemrulemigrations]>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.744+00:00][INFO ][plugins.securitysolution.siemrulemigrations]>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.748+00:00][INFO ][plugins.observabilityaiassistant.lock-manage>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.862+00:00][INFO ][plugins.fleet.fleetunenroll-inactive-agents]>
Feb 01 17:58:55 test kibana[12754]: [2026-02-01T17:58:55.877+00:00][INFO ][plugins.fleet.fleetupgrade-agentless-deploy]>
lines 1-21/21 (END)
test@test:/etc/elasticsearch$ sudo systemctl enable kibana
Created symlink /etc/systemd/system/multi-user.target.wants/kibana.service → /usr/lib/systemd/system/kibana.service.
test@test:/etc/elasticsearch$
```

Kibana active and running

Configure — kibana.yml

By default Kibana only listens on localhost, which means the browser on my host machine can't reach it. I edited the config to bind it to all interfaces and set the public URL, so links and redirects generated by Kibana point to the right address.

```
cd /etc/ && sudo chmod 0755 kibana/
```

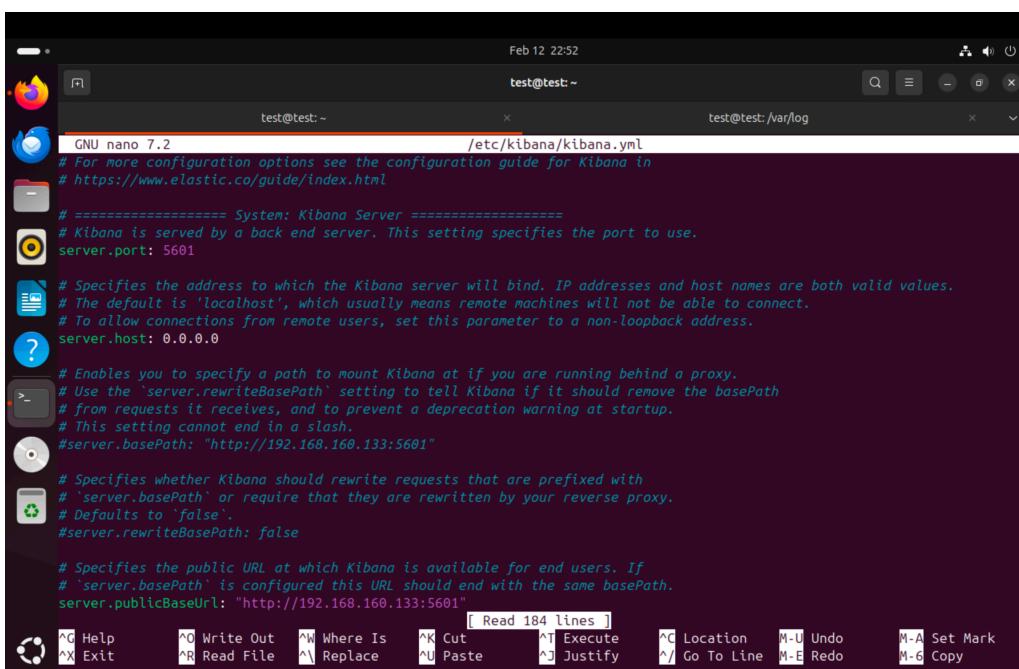


```
test@test:/etc$ cd /etc/
test@test:/etc$ sudo chmod 0755 kibana /
test@test:/etc$
```

Setting permissions on /etc/kibana

The settings I changed:

- **server.port: 5601** : default Kibana port
- **server.host: 0.0.0.0** : without this, Kibana only listens on localhost and you can't reach it from another machine
- **server.name: test** : display name for this instance
- **server.publicBaseUrl: http://192.168.160.133:5601** : the external URL; needed so Kibana generates correct links once Nginx is in front



```
GNU nano 7.2          /etc/kibana/kibana.yml
# For more configuration options see the configuration guide for Kibana in
# https://www.elastic.co/guide/index.html

# ===== System: Kibana Server =====
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

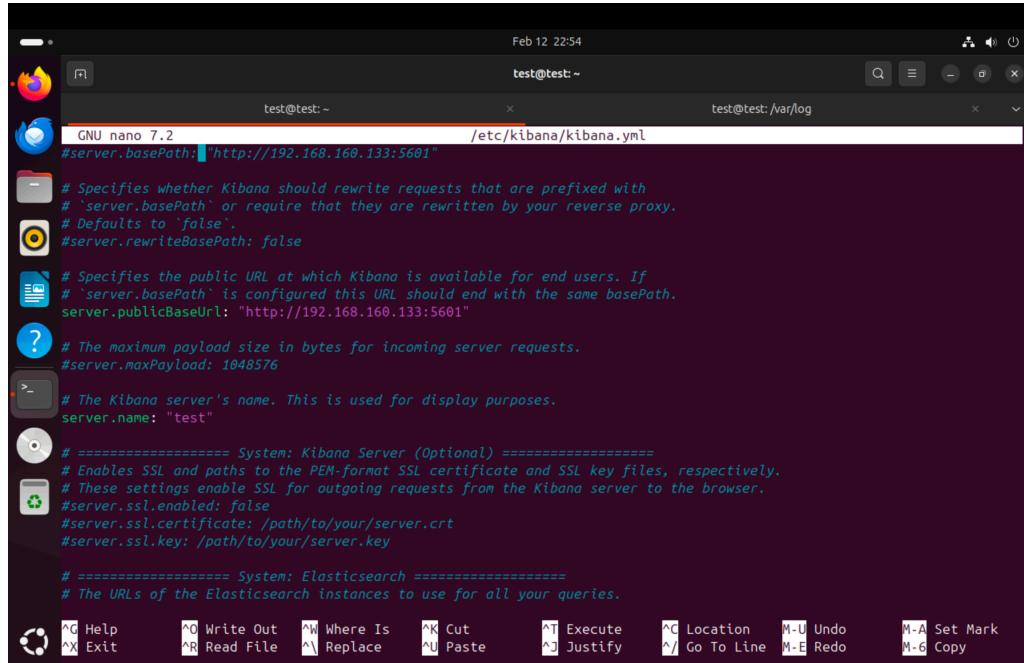
# Specifies the address to which the Kibana server will bind. IP addresses and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not be able to connect.
# To allow connections from remote users, set this parameter to a non-loopback address.
server.host: 0.0.0.0

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the 'server.rewriteBasePath' setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
#serverbasePath: "http://192.168.160.133:5601"

# Specifies whether Kibana should rewrite requests that are prefixed with
# 'serverbasePath' or require that they are rewritten by your reverse proxy.
# Defaults to 'false'.
#server.rewriteBasePath: false

# Specifies the public URL at which Kibana is available for end users. If
# 'serverbasePath' is configured this URL should end with the same basePath.
server.publicBaseUrl: "http://192.168.160.133:5601"
```

kibana.yml — port and host configured



```
Feb 12 22:54
test@test:~ test@test:/var/log
GNU nano 7.2 /etc/kibana/kibana.yml
#serverbasePath: "http://192.168.160.133:5601"
# Specifies whether Kibana should rewrite requests that are prefixed with
# 'server.basePath' or require that they are rewritten by your reverse proxy.
# Defaults to 'false'.
#server.rewriteBasePath: false

# Specifies the public URL at which Kibana is available for end users. If
# 'server.basePath' is configured this URL should end with the same basePath.
server.publicBaseUrl: "http://192.168.160.133:5601"

# The maximum payload size in bytes for incoming server requests.
#server.maxPayload: 1048576

# The Kibana server's name. This is used for display purposes.
server.name: "test"

# ===== System: Kibana Server (Optional) =====
# Enables SSL and paths to the PEM-format SSL certificate and SSL key files, respectively.
# These settings enable SSL for outgoing requests from the Kibana server to the browser.
#server.ssl.enabled: false
#server.ssl.certificate: /path/to/your/server.crt
#server.ssl.key: /path/to/your/server.key

# ===== System: Elasticsearch =====
# The URLs of the Elasticsearch instances to use for all your queries.

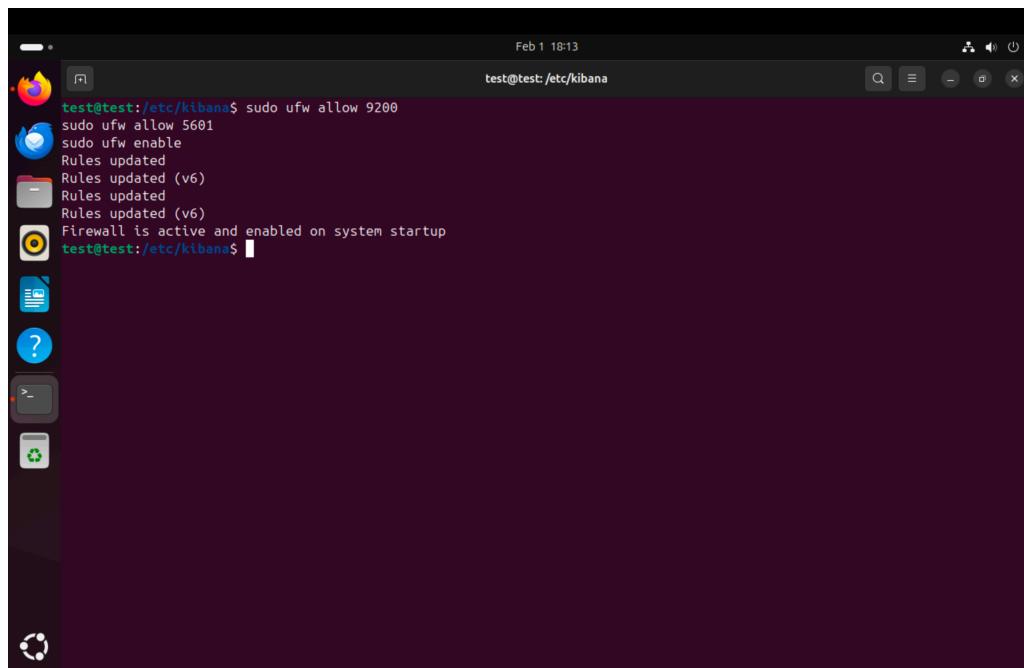
FG Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
FX Exit      ^R Read File   ^A Replace    ^U Paste      ^J Justify    ^G Go To Line M-E Redo
M-A Set Mark M-6 Copy
```

kibana.yml — server name and publicBaseUrl set

Open firewall ports

Ubuntu's firewall (UFW) blocks inbound traffic by default. I opened ports 9200 and 5601 so I could reach Elasticsearch and Kibana from the host machine. If you skip this step and wonder why nothing connects — this is usually why.

```
sudo ufw allow 9200
sudo ufw allow 5601
sudo ufw enable
```



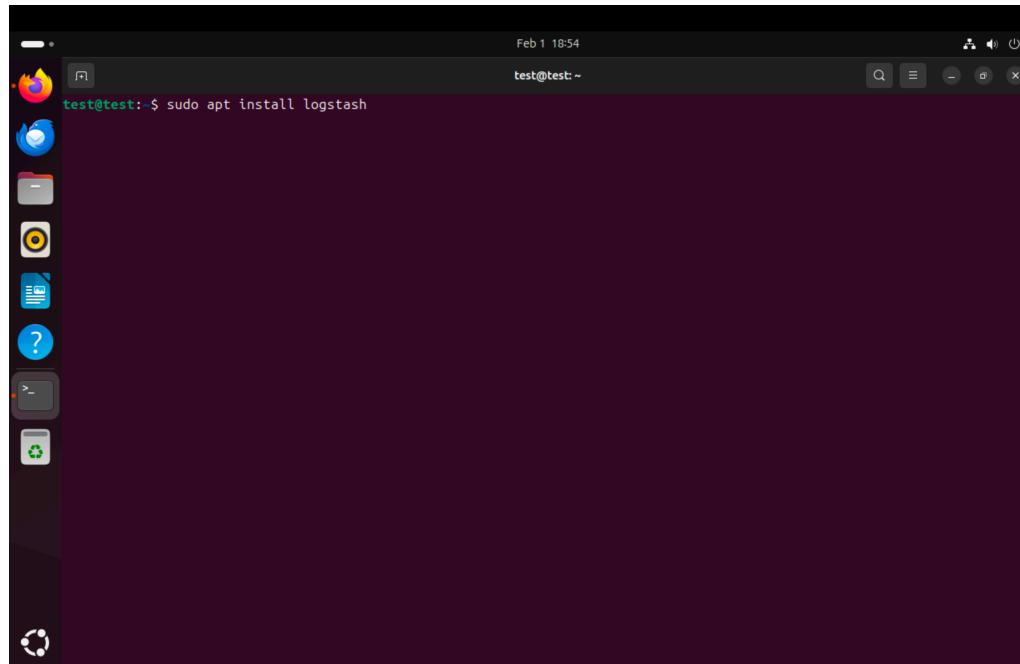
```
Feb 1 18:13
test@test:/etc/kibana$ sudo ufw allow 9200
sudo ufw allow 5601
sudo ufw enable
Rules updated
Rules updated (v6)
Rules updated (v6)
Firewall is active and enabled on system startup
test@test:/etc/kibana$
```

UFW rules added and firewall enabled

Step 5 — Install and Configure Logstash

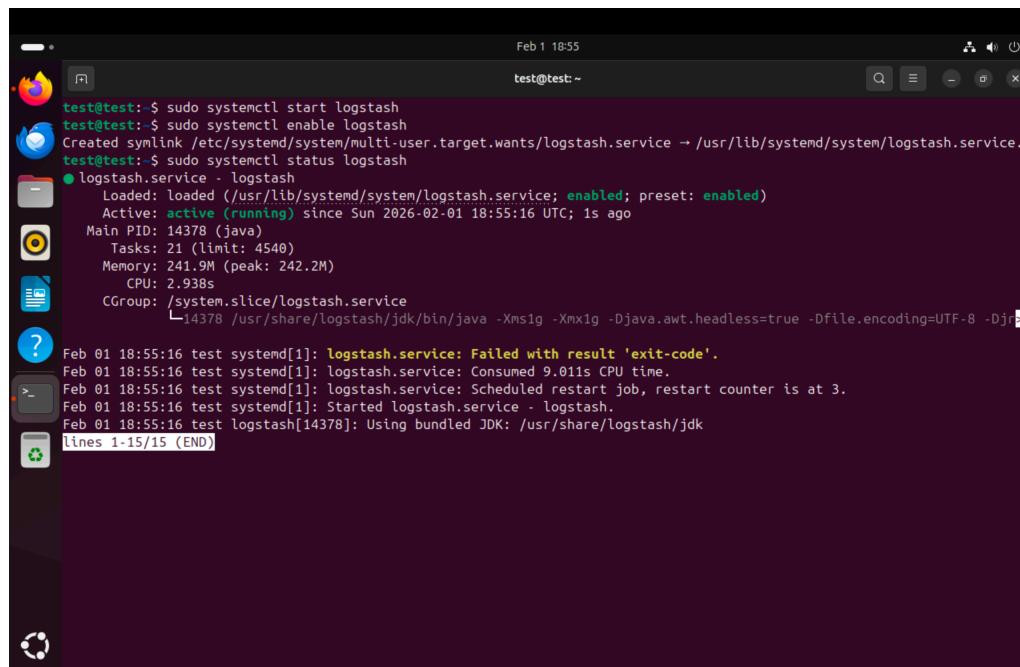
Logstash is the data pipeline, it receives logs from various sources, processes and transforms them, and forwards them to Elasticsearch. I installed it and then created two config files: one that tells Logstash where to listen for incoming data, and one that tells it where to send the output.

```
sudo apt install logstash
```



Logstash installing

```
sudo systemctl start logstash  
sudo systemctl enable logstash  
sudo systemctl status logstash
```

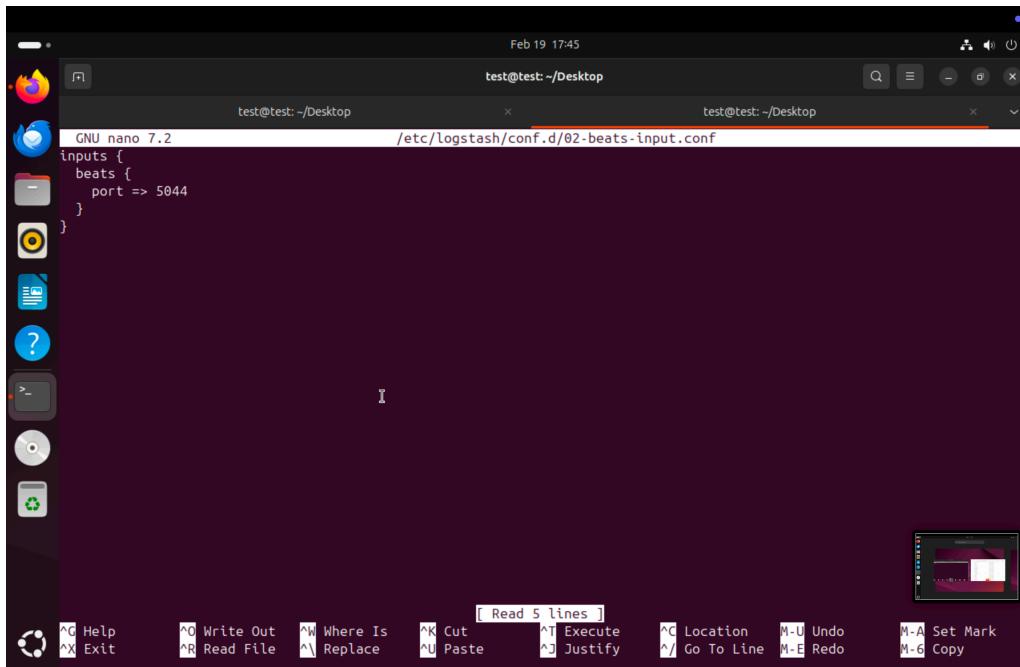


Logstash active and running

Configure the pipeline

Logstash reads pipeline config from .conf files placed in /etc/logstash/conf.d/. I created two files to define the input and output.

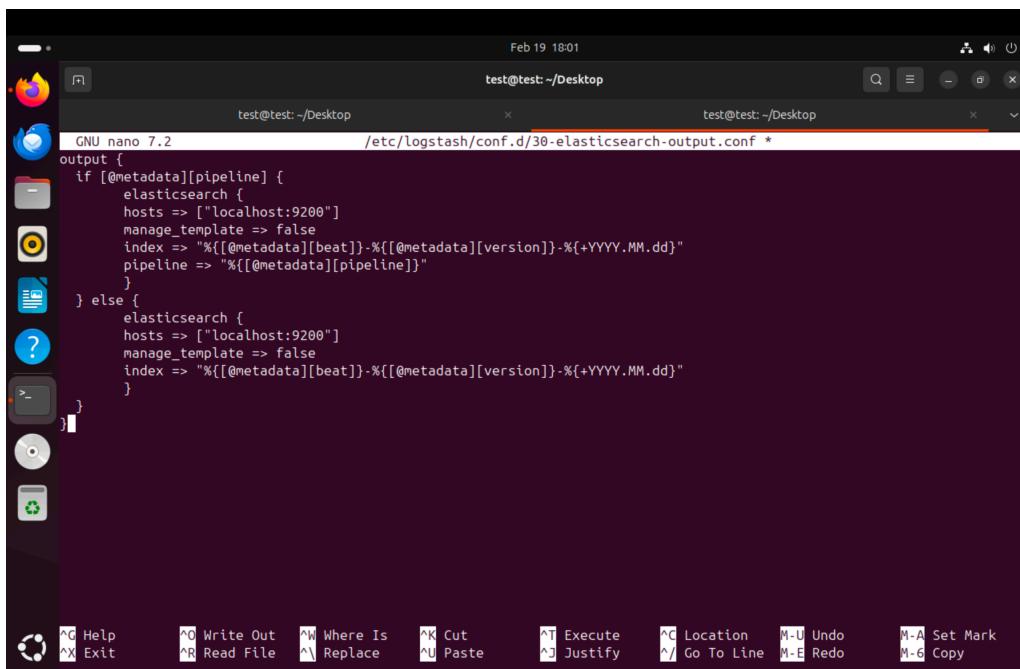
The input file tells Logstash to listen on port 5044 for connections from Beat agents (Auditbeat, Filebeat, etc.). Port 5044 is the default for the Beats protocol, so any Beat you configure will try to send there by default.



```
GNU nano 7.2
inputs {
  beats {
    port => 5044
  }
}
```

02-beats-input.conf — listening on port 5044 for Beat agents

The output file routes the processed events to Elasticsearch and dynamically names the index based on the Beat that sent the data, so Auditbeat events go into auditbeat-* indices, Filebeat events into filebeat-* and so on. This keeps the data organised and makes it easy to target the right index pattern in Kibana later.



```
output {
  if [@metadata][pipeline] {
    elasticsearch {
      hosts => ["localhost:9200"]
      manage_template => false
      index => "%{@metadata}[beat]-%{@metadata}[version]-%{+YYYY.MM.dd}"
      pipeline => "%{@metadata}[pipeline]"
    }
  } else {
    elasticsearch {
      hosts => ["localhost:9200"]
      manage_template => false
      index => "%{@metadata}[beat]-%{@metadata}[version]-%{+YYYY.MM.dd}"
    }
  }
}
```

30-elasticsearch-output.conf — forwarding to Elasticsearch with dynamic index names

Test the pipeline config

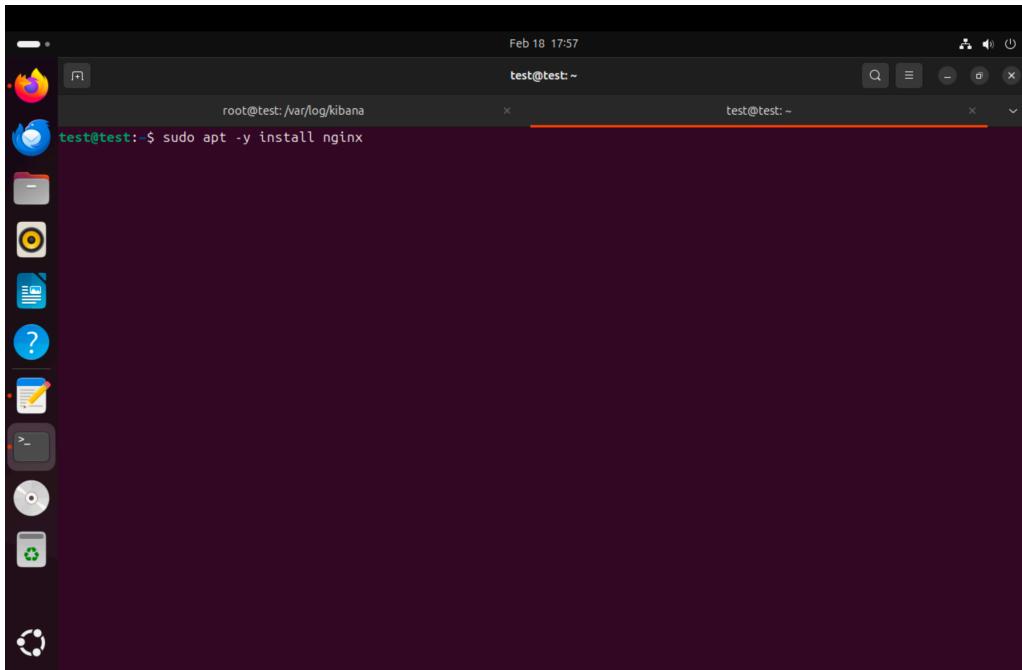
I always test new Logstash configs before restarting the service. A syntax error in a pipeline file will crash Logstash on restart, so this validation step saves a lot of headache.

Logstash config test — "Configuration OK"

Step 6 — Set Up Nginx as Reverse Proxy

Kibana was accessible on port 5601 at this point, but I wanted two things: a cleaner URL on port 80, and a login prompt so the dashboard isn't just open to anyone who can reach the VM. I set up Nginx as a reverse proxy that sits in front of Kibana and handles both.

Install Nginx



```
Feb 18 17:57
root@test:/var/log/kibana
test@test:~
```

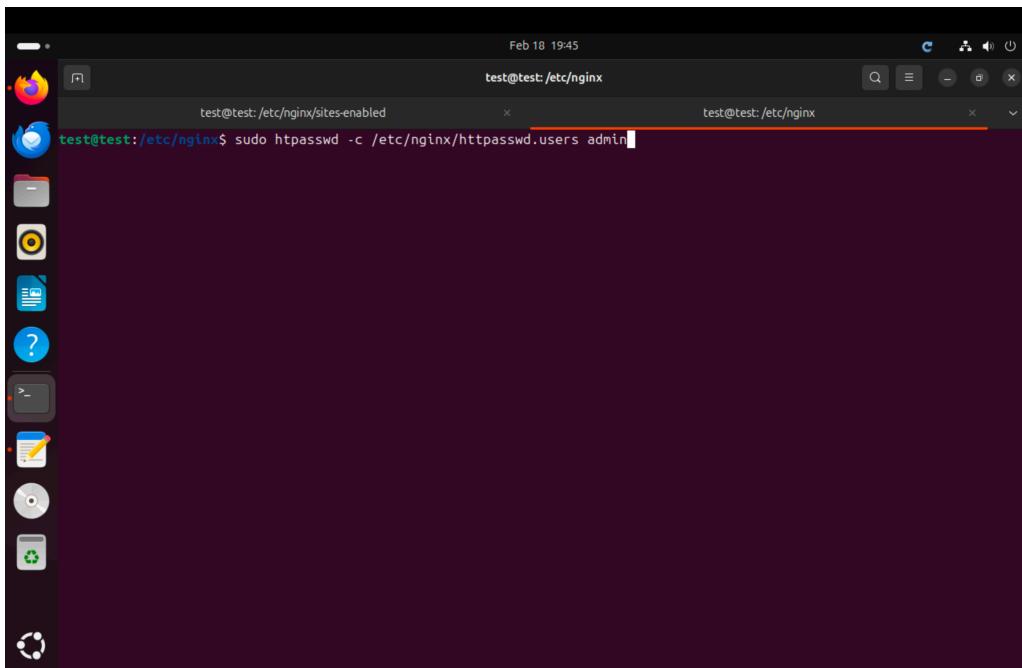
```
test@test:~$ sudo apt -y install nginx
```

Nginx installing

Create a password file

Nginx's Basic Auth reads from an `htpasswd` file, a list of usernames with their hashed passwords. The `-c` flag creates the file. If you run it again with `-c` later it will wipe the existing one, so only use `-c` the first time.

```
sudo htpasswd -c /etc/nginx/.htpasswd admin
```



```
Feb 18 19:45
test@test:/etc/nginx/sites-enabled
test@test:/etc/nginx
```

```
test@test:/etc/nginx$ sudo htpasswd -c /etc/nginx/.htpasswd admin
```

htpasswd file created with admin user

Configure the virtual host

I created a server block that listens on port 80, checks credentials against the `htpasswd` file, and proxies authenticated requests to Kibana on `localhost:5601`. The proxy headers for `Upgrade` and `Connection` are important — Kibana uses WebSockets, and without those headers the connection gets dropped by the proxy.

A screenshot of a terminal window titled "GNU nano 7.2 /etc/nginx/sites-enabled/server.conf". The window shows the configuration file content:

```
server {
    listen 80;

    server_name 192.168.160.133;

    auth_basic "Restricted Access";
    auth_basic_user_file /etc/nginx/htpasswd.users;

    location / {
        proxy_pass http://localhost:5601;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

The terminal has a dark theme and includes a standard set of keyboard shortcuts at the bottom.

/etc/nginx/sites-enabled/server.conf — the reverse proxy config

Test and enable

I ran `nginx -t` to validate the config before restarting. A syntax error here takes the whole Nginx server down, so it's always worth checking first.

```
sudo nginx -t
```

A screenshot of a terminal window titled "test@test:/etc/nginx/sites-enabled". The window shows the command `sudo nginx -t` being run and its output:

```
test@test:/etc/nginx/sites-enabled$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
test@test:/etc/nginx/sites-enabled$
```

Config test passes — syntax is OK

```
sudo systemctl enable nginx
sudo systemctl start nginx
sudo systemctl status nginx
```

The screenshot shows a terminal window titled "test@test:/etc/nginx/sites-enabled". The terminal output is as follows:

```
Feb 18 18:25
test@test:/etc/nginx/sites-enabled$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
test@test:/etc/nginx/sites-enabled$ sudo systemctl start nginx
test@test:/etc/nginx/sites-enabled$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
    Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
    Active: active (running) since Sat 2026-02-14 00:16:52 UTC; 4 days ago
      Docs: man:nginx(8)
      Main PID: 67520 (nginx)
         Tasks: 3 (limit: 4540)
        Memory: 372.0K (peak: 4.1M swap: 2.1M swap peak: 2.1M)
          CPU: 9ms
        CGroup: /system.slice/nginx.service
                ├─67520 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
                ├─68857 "nginx: worker process"
                └─68858 "nginx: worker process"

Feb 14 00:16:52 test systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
Feb 14 00:16:52 test systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
Feb 14 00:18:50 test systemd[1]: Reloading nginx.service - A high performance web server and a reverse proxy server...
Feb 14 00:18:50 test nginx[68855]: 2026/02/14 00:18:50 [notice] 68855#68855: signal process started
Feb 14 00:18:50 test systemd[1]: Reloaded nginx.service - A high performance web server and a reverse proxy server.
test@test:/etc/nginx/sites-enabled$
```

Nginx running — reverse proxy is live on port 80

Step 7 — Install and Configure Auditbeat

With the pipeline in place, I needed something to actually generate data. I chose Auditbeat because it hooks directly into the Linux audit subsystem and captures real system-level events (logins, process starts, network connections, user changes) rather than just reading log files. It sends everything as structured JSON, which means it's immediately queryable in Kibana without any parsing work.

Install

```
sudo apt install auditbeat -y
```

Configure — auditbeat.yml

Auditbeat's system module is what I wanted. It monitors several datasets and I enabled all of them to get as much visibility as possible:

- **host** : basic system info: hostname, IPs, OS version, uptime
- **login** : login and logout events, including SSH sessions and failed attempts
- **process** : every process that starts or stops on the system
- **socket** : network connections being opened and closed
- **user** : changes to user accounts on the system
- **package** : packages being installed, updated, or removed

```
Feb 23 01:16
test@test:~ /etc/auditbeat/auditbeat.yml *

GNU nano 7.2
- module: system
  datasets:
    - package # Installed, updated, and removed packages
  period: 2m # The frequency at which the datasets check for changes

- module: system
  datasets:
    - host # General host information, e.g. uptime, IPs
    - login # User logins, logouts, and system boots.
    - process # Started and stopped processes
    - socket # Opened and closed sockets
    - user # User information

  # How often datasets send state updates with the current state of the system (e.g. all currently running processes, all open sockets).
  state.period: 12h

  # Enabled by default. Auditbeat will read password fields in # /etc/passwd and /etc/shadow and store a hash locally to # detect any changes.
  user.detect_password_changes: true

  # File patterns of the login record files.
  login.wtmp_file_pattern: /var/log/wtmp*
  login.btmp_file_pattern: /var/log/btmp*

^G Help      ^O Write Out   ^W Where Is   ^K Cut        ^T Execute   ^C Location   M-U Undo   M-A Set Mark
^X Exit      ^R Read File   ^M Replace    ^U Paste     ^J Justify   ^L Go To Line M-E Redo   M-G Copy
```

auditbeat.yml — system module with all datasets enabled

For the output I pointed Auditbeat directly at Elasticsearch on port 9200. I could have routed it through Logstash on port 5044 instead, but since Auditbeat already sends clean structured JSON there's nothing to transform, so going direct is simpler.

```
Feb 23 01:16
test@test:~ /etc/auditbeat/auditbeat.yml *

# The cloud.id setting overwrites the `output.elasticsearch.hosts` and
# `setup.kibana.host` options.
# You can find the `cloud.id` in the Elastic Cloud web UI.
#cloud.id:

# The cloud.auth setting overwrites the `output.elasticsearch.username` and
# `output.elasticsearch.password` settings. The format is '<user>:<pass>'.
#cloud.auth:

# ===== Outputs =====
# Configure what output to use when sending the data collected by the beat.
# -----
# Elasticsearch Output -----
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["localhost:9200"]

  # 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"

# ----- Logstash Output -----
#output.logstash:
```

auditbeat.yml — output.elasticsearch set to localhost:9200

Load dashboards, start, and enable

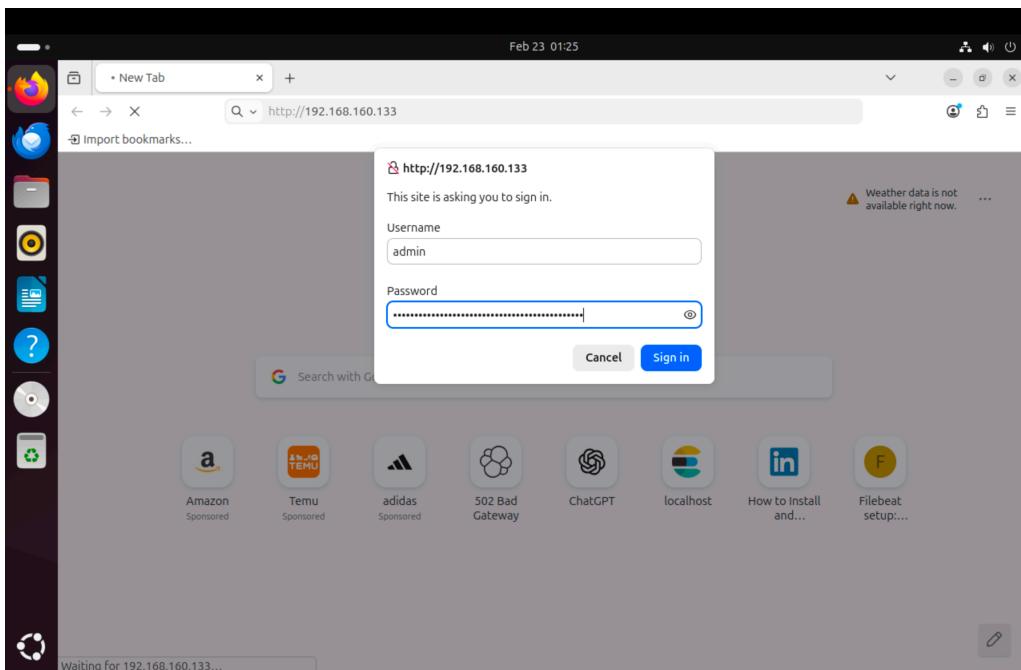
Auditbeat can push its own pre-built dashboards into Kibana automatically. I ran the setup command first so the dashboards were ready as soon as data started flowing. Then I started the service and enabled it so it runs on reboot.

```
sudo auditbeat setup -e
sudo systemctl start auditbeat
sudo systemctl enable auditbeat
```

Auditbeat running — dashboards loaded into Kibana

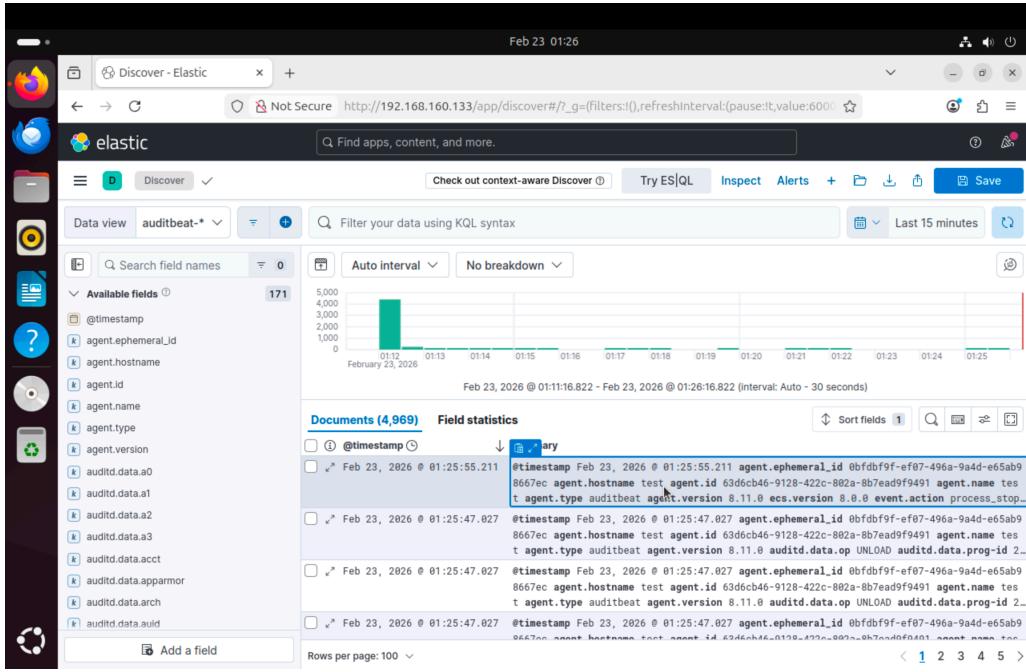
Result — Data Flowing Through the Pipeline

With everything running, I navigated to `http://192.168.160.133` and got the Nginx Basic Auth prompt, which confirmed that the reverse proxy was working and the dashboard wasn't just openly accessible.



Basic Auth prompt from Nginx before reaching Kibana

After logging in, I opened the Discover view in Kibana and selected the auditbeat-* index pattern. The data was already there – nearly 5,000 documents indexed in the first 15 minutes, with 171 available fields covering everything from process names to network socket details. Every event had a timestamp, the hostname it came from, and full context about what happened.



Kibana Discover — 4,969 auditbeat documents already indexed

The full pipeline was working: Auditbeat collecting events at the kernel level, Logstash receiving and forwarding them, Elasticsearch indexing and storing everything, Kibana making it all searchable, and Nginx handling the authentication layer in front. From here the next step is building custom dashboards and writing KQL queries to search through the data and spot anything interesting.

What's Next

This project gave me a solid foundation — a working SIEM pipeline that collects real system data, indexes it, and makes it searchable. But there's a lot more I want to build on top of it, and I'll be documenting everything the same way as I go.

The next writeup will cover detection rules and how to actually use the Security app in Kibana — creating custom rules, triggering alerts by simulating real attack techniques, and walking through what an alert investigation looks like from start to finish. It's the part that turns this from a monitoring setup into something that can actively detect threats.

After that I want to integrate Wazuh, which is an open-source EDR and SIEM platform that pairs well with the ELK Stack. Wazuh adds host-based intrusion detection, file integrity monitoring, vulnerability detection, and a dedicated agent that goes much deeper than Auditbeat alone. Setting it up alongside what's already here would make this a genuinely complete security lab.

Further down the line: adding a second VM as a target machine to monitor, simulating attack scenarios with tools like Hydra or Metasploit, and building Kibana dashboards that visualise the results in a way that tells a clear story. Everything will be documented step by step and added to this repository as it gets done.