Project Citadel: Next-Generation Security Analytics & Response Platform

Document Type: Subsystem Implementation Document

1. Executive Summary

This document presents the design, implementation, and validation of the Kubernetes platform and the Monitoring & Observability stack deployed as part of the Citadel MVP. These components form the operational backbone of the Citadel security platform, enabling shared platform services, centralized visibility, and real-time alerting across all core infrastructure nodes.

The Kubernetes environment, hosted on the citadel-kube VM, provides a lightweight but extensible platform for running shared applications and enforcing security controls. Core services such as NGINX Ingress, cert-manager, Kyverno, External Secrets Operator, and Falco were deployed to demonstrate secure, cloud-native operations and runtime visibility.

In parallel, a full Prometheus–Grafana–Alertmanager monitoring pipeline was implemented on the citadel-monitor VM. Metrics are collected from all Citadel components using a combination of Node, Elasticsearch, Suricata, and Prometheus exporters. Grafana dashboards provide real-time visibility into system health, log ingestion, and detection telemetry, while Alertmanager routes critical alerts to a Slack channel for rapid SOC response.

This implementation directly addresses requirements for:

- **Cloud-native architecture** to host shared services securely
- **Centralized operational visibility** through metrics collection and dashboards
- **Real-time alerting** integrated into SOC workflows

Together, the Kubernetes and Monitoring subsystems establish a **robust operational foundation** for Citadel MVP, supporting current SOC workflows while allowing future scalability, multi-tenancy, and advanced analytics integration.

2. Introduction

The Citadel MVP was designed to meet strategic objectives of creating a **cloud-native, operationally visible, and highly automated SOC platform**. As part of this initiative, two critical subsystems were implemented:

1. A **Kubernetes platform** for hosting shared security and operational services.
2. A **Monitoring & Observability stack** that provides real-time visibility, performance telemetry, and alerting across all core infrastructure nodes.

Requirements

The following requirements directly shaped the design of these components:

- **Cloud-Native Platform Architecture**
  The platform must leverage modern container orchestration to host shared services securely and flexibly, supporting future scalability and multi-tenant deployments.
- **Operational Visibility & Monitoring Dashboards**
  The SOC team must be able to monitor infrastructure health, log ingestion rates, and detection telemetry through centralized dashboards and metrics.
- **Real-Time Alerting & SOC Integration**
  Critical system or telemetry issues must generate alerts and route them to the SOC team through standard communication channels (e.g., Slack) to enable timely investigation and response.

What Was Implemented

To fulfill these requirements, the following were deployed as part of the Citadel MVP:

- A **single-node Kubernetes environment** (citadel-kube) hosting core shared services including NGINX Ingress, cert-manager, Kyverno, External Secrets Operator, Falco, and the SOAR engine (covered separately).
- A **centralized monitoring and alerting stack** on the citadel-monitor VM, comprising Prometheus, Grafana, and Alertmanager.
- **Exporters** (Node, Elasticsearch, Suricata, Prometheus self-scrape) deployed across all Citadel VMs to provide detailed operational metrics.
- **Grafana dashboards** for real-time visualization of infrastructure health, Elasticsearch cluster status, and Suricata telemetry.

- **Alerting pipeline** with Prometheus rules and Alertmanager routing alerts to a Slack channel for SOC visibility.

This integrated setup establishes the **operational visibility layer** of Citadel MVP, enabling SOC analysts to monitor system health, detect anomalies rapidly, and lay the groundwork for scalable, production-grade observability.

3. Scope

This section defines the **current scope**, **technical constraints**, **identified gaps**, and **future enhancement areas** for the Kubernetes and Monitoring subsystems deployed under the Citadel MVP. The objective is to provide with a clear understanding of **what has been implemented**, **what is intentionally excluded in the MVP phase**, and **what can be extended in subsequent releases**.

3.1 Current Scope

The following components and capabilities are included within the current MVP scope:

- **Kubernetes Platform**
    - A single-node Kubernetes environment (citadel-kube) was deployed to host shared platform and security services, including:
        - NGINX Ingress
        - cert-manager
        - Kyverno
        - External Secrets Operator
        - Falco (runtime security)
    - The Kubernetes cluster acts as a foundational control plane for future multi-tenant or containerized security services.
- **Monitoring & Observability Stack**
    - A centralized Prometheus–Grafana–Alertmanager stack was deployed on citadel-monitor.
    - **Metrics collection** was enabled from all core Citadel VMs through Node, Elasticsearch, Suricata, and Prometheus exporters.

    - **Dashboards** were implemented for real-time visualization of system performance, Elasticsearch cluster health, and Suricata telemetry.
    - **Alerting** was configured through Prometheus rules and Alertmanager, with integration into Slack for operational visibility.
- **Target Infrastructure**
    - Monitoring currently covers six Citadel VMs:
        - citadel-kube – Kubernetes & SOAR
        - citadel-elk – Elasticsearch & Logstash
        - citadel-detect – Suricata & Wazuh
        - citadel-monitor – Monitoring stack
        - citadel-secrets – Vault, MinIO, Keycloak
        - citadel-ueba – AI/UEBA jobs

3.2 Constraints

The following constraints apply to this MVP implementation:

- The Kubernetes environment is **single-node** only, without high availability (HA) or multi-cluster federation.
- Prometheus and Grafana are deployed in a **non-redundant configuration**, with no failover or horizontal scaling.
- Alerting rules are **limited to core infrastructure health checks** (e.g., node availability), without complex correlation logic.
- The Falco runtime security component is currently deployed but not fully integrated with the monitoring or alerting pipeline.
- The monitoring stack assumes **static target infrastructure** and does not include dynamic service discovery or auto-scaling behavior.

3.3 Gaps

The following functional gaps have been identified in the current MVP:

- **UEBA Metrics Integration**
  UEBA job outputs are not currently exposed as Prometheus metrics, limiting anomaly visibility in Grafana.

- **Limited Alert Coverage**
  Alerting rules focus primarily on infrastructure health and basic telemetry. Application-level or security anomaly alerts are not yet implemented.
- **No Multi-Tenant Isolation Metrics**
  While multi-tenancy is a strategic objective, current dashboards do not provide tenant-level breakdowns.
- **Falco Alert Forwarding**
  Falco alerts are not yet piped into the Prometheus/Grafana pipeline for unified security visibility.

3.4 Future Enhancements

Planned enhancements for future phases include:

- **High Availability (HA) Monitoring Stack**
  Deploying Prometheus and Grafana in redundant configurations to support production workloads.
- **Advanced Alerting & Correlation**
  Expanding Prometheus rules to include application-layer metrics, UEBA anomaly signals, and cross-domain correlation.
- **Multi-Node Kubernetes Deployment**
  Expanding the Kubernetes environment to support load balancing, fault tolerance, and multi-tenant containerized workloads.
- **Falco & UEBA Integration**
  Forwarding Falco runtime alerts and UEBA anomalies into the monitoring pipeline to create unified SOC visibility.
- **Tenant-Level Dashboards & KPIs**
  Developing Grafana dashboards that separate metrics by tenant or application context to support MSSP service model.

4. Tools, Components & Solution Overview

This section outlines the **key platform components**, **monitoring tools**, and **solution architecture** implemented under the Citadel MVP to deliver Kubernetes-based shared services

and centralized operational visibility. The deployed stack has been designed to align with market requirements for **cloud-native architecture**, **real-time telemetry**, and **SOC-integrated alerting**.

4.1 Kubernetes Platform (citadel-kube)

The **citadel-kube** virtual machine hosts a lightweight, single-node Kubernetes environment that serves as the **shared services platform** for the Citadel MVP. It provides a foundational control plane to run security, policy, and orchestration services in a cloud-native manner.

Component	Purpose
NGINX Ingress Controller	Routes external traffic into Kubernetes services securely.
cert-manager	Automates TLS certificate provisioning and management.
Kyverno	Enforces security and compliance policies within the Kubernetes cluster.
External Secrets Operator	Integrates Kubernetes workloads with external secret stores (Vault).
Falco	Provides runtime security monitoring for container and node activities.
SOAR Engine (separate doc)	Hosted here to leverage Kubernetes orchestration and networking.

The Kubernetes platform plays a **strategic role** in standardizing how shared SOC services are deployed and managed. While the current setup is single-node, it is designed to evolve into a multi-node, multi-tenant environment in future phases.

4.2 Monitoring & Observability Stack (citadel-monitor)

The **citadel-monitor** VM hosts the centralized monitoring and observability stack built using **Prometheus**, **Grafana**, and **Alertmanager**. Together, these components collect, visualize, and route operational telemetry from all Citadel nodes.

Tool	Role in MVP

Prometheus	Core metrics collection and storage engine; scrapes data from exporters on all Citadel VMs.
Grafana	Visualization and dashboarding layer; displays ingestion rates, alerts, resource utilization.
Alertmanager	Routes Prometheus alerts to external channels (Slack) for SOC visibility and response.

Exporter agents were deployed across each VM to collect relevant metrics:

VM	Exporters Deployed
citadel-kube	Node Exporter
citadel-elk	Node Exporter, Elasticsearch Exporter
citadel-detect	Node Exporter, Suricata Exporter
citadel-secrets	Node Exporter
citadel-ueba	Node Exporter
citadel-monitor	Node Exporter, Prometheus self-scrape

This setup provides **full infrastructure coverage** across ingestion, detection, UEBA, and secret management layers, ensuring that all operational components are monitored from a single pane of glass.

4.3 Dashboards & Visualization

Grafana was integrated with Prometheus as its primary data source and prebuilt dashboards were imported to provide immediate visibility into system performance:

Dashboard Name	Grafana ID	Purpose
Prometheus Dashboard	1860	Real-time CPU, memory, disk, and network utilization for all nodes.
Elasticsearch Dashboard	4358	Alert dashboard from elasticsearch.
Wazuh Dashboard	15996	Alert dashboard from Wazuh(via opensearch).

SOC analysts can **filter dashboards by instance (VM)** to isolate issues or monitor specific components in real time. These dashboards address requirement for **centralized visibility** across infrastructure and security telemetry.

4.4 Alerting Pipeline

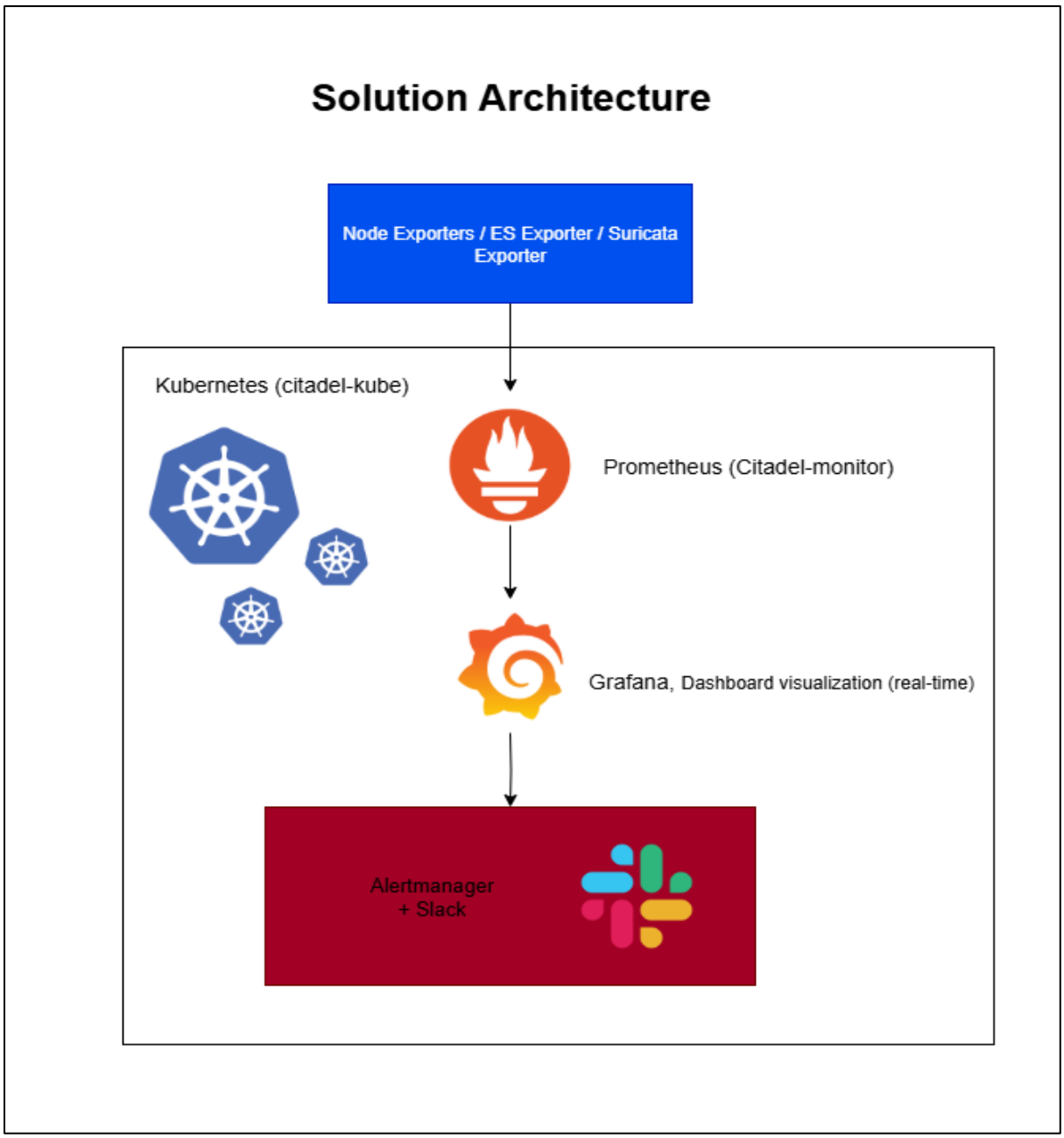
A fully functional alerting pipeline was configured to support real-time operational response:

1. **Prometheus Alert Rules** - Defined to detect critical node and service issues (e.g., NodeDown).
2. **Alertmanager Integration** - Prometheus forwards alert events to Alertmanager for routing.
3. **Slack Notifications** - Alertmanager sends alerts to a dedicated Slack channel via webhook for SOC visibility.

This pipeline ensures that operational anomalies are detected promptly and **integrated directly into SOC communication workflows**, enabling timely investigation and remediation.

4.5 Solution Architecture

The figure below illustrates the **high-level data flow and component interactions** for the Kubernetes and Monitoring stack within Citadel MVP:



Exporters on each VM expose operational metrics.

- Prometheus scrapes and stores these metrics centrally.
- Grafana visualizes telemetry for SOC analysts.
- Alertmanager pushes alerts to Slack for rapid response.
- Kubernetes (citadel-kube) acts as the control plane hosting shared services (e.g., Falco, SOAR engine).

5. Configurations & Customizations

This section details the **key configuration steps**, **custom settings**, and **non-default adjustments** applied to the Kubernetes and Monitoring stack during the Citadel MVP deployment. The objective was to create a **functionally complete, observable, and SOC-ready environment** while maintaining flexibility for future scaling.

5.1 Kubernetes Platform

The Kubernetes environment on citadel-kube was configured with several core platform services, each tailored for MVP use:

- **NGINX Ingress Controller**
  - Deployed via Helm to manage external ingress traffic.
  - Configured to listen on the VM's public IP and route traffic securely to internal services.
  - TLS termination handled through cert-manager.
- **cert-manager**
  - Configured to issue and renew self-signed TLS certificates for ingress endpoints.
  - Used to secure traffic between Kubernetes services during the MVP phase.
- **Kyverno**
  - Policy engine installed to enforce baseline security controls such as runAsNonRoot and readOnlyRootFilesystem.
  - Custom cluster policies were created to align workloads with Citadel's security standards.
- **External Secrets Operator**
  - Configured to synchronize secrets from Vault (citadel-secrets) into Kubernetes pods.
  - Integrated via Kubernetes ServiceAccount with limited permissions.
- **Falco**
  - Configured for runtime monitoring of container activity.
  - Currently running in detection-only mode; alert forwarding to Prometheus/Grafana is planned for future phases.

These services were **deployed using Helm charts with minimal overrides**, ensuring rapid setup while preserving future extensibility. RBAC and network policies were kept simple but functional for MVP.

5.2 Prometheus Configuration

Prometheus was installed on citadel-monitor and customized through the prometheus.yml file to scrape metrics from all core Citadel nodes.

Key Customizations:

- **Listening Address:**
  - Configured to listen on 0.0.0.0:9090 to allow inbound traffic from Grafana and Alertmanager.
- **Scrape Targets:**
  - Custom scrape_configs were defined for each Citadel VM.
  - Exporters covered: Node metrics, Elasticsearch metrics, Suricata metrics, and Prometheus self-scrape.

scrape_configs:

```
 - job_name: 'node'
   static_configs:
    - targets: ['citadel-kube:9100','citadel-elk:9100','citadel-detect:9100','citadel-secrets:9100','citadel-ueba:9100','citadel-monitor:9100']

 - job_name: 'elasticsearch'
   static_configs:
    - targets: ['citadel-elk:9114']

 - job_name: 'suricata'
   static_configs:
    - targets: ['citadel-detect:9400']

 - job_name: 'prometheus'
   static_configs:
```

- targets: ['citadel-monitor:9090']

- **Verification:**
    - Confirmed all targets showed **UP** in Prometheus → Status > Targets.

These custom scrape configurations ensured **complete visibility** across the ingestion, detection, and UEBA nodes.

5.3 Exporters

Exporter agents were deployed manually or via system packages on each VM.

VM	Exporters Deployed	Custom Config
citadel-kube	Node Exporter	Default port 9100, firewall adjusted
citadel-elk	Node Exporter, Elasticsearch Exporter	Elasticsearch exporter on :9114
citadel-detect	Node Exporter, Suricata Exporter	Suricata exporter on :9400
citadel-secrets	Node Exporter	—
citadel-ueba	Node Exporter	—
citadel-monitor	Node Exporter, Prometheus Self-scrape	—

- **Firewall Rules** were adjusted to allow inbound connections from citadel-monitor to exporter ports.
- Each exporter was tested individually with curl http://<vm-ip>:<port>/metrics before Prometheus scraping was enabled.

5.4 Grafana Configuration

Grafana was installed on citadel-monitor and connected to Prometheus as its primary data source.

- **Data Source:**
    - Configured via the Grafana UI using the Prometheus public IP and port 9090.
    - Connection tested successfully using the up{job="node"} query.

- **Dashboards Imported:**
  - Prometheus Full (ID 1860)
  - Elasticsearch (ID 4358)
  - Wazuh (Opensearch) (ID 15996)
  - Dashboards were imported as JSON from Grafana.com and retained their default panel structure.
- **Customization:**
  - Instance dropdown was used to filter metrics per VM.
  - No panel modifications were required for MVP, but dashboard IDs were documented for future change tracking.

5.5 Alertmanager Configuration

Alertmanager was installed on citadel-monitor and integrated with Prometheus for alert routing.

- **Configuration:**
  - Alertmanager bound to 0.0.0.0:9093.
  - Prometheus prometheus.yml updated with alertmanager target:

```
alerting:
  alertmanagers:
    - static_configs:
        - targets: ['<monitor-public-ip>:9093']
```

- **Slack Integration:**
  - A Slack webhook was configured in alertmanager.yml to deliver alerts to the #alerts channel.

```
receivers:
- name: 'slack-notifications'
  slack_configs:
  - api_url: '<slack-webhook>'
    channel: '#alerts'
```

- **Test Alerts:**

- A NodeDown rule was added to validate the end-to-end alerting pipeline.
- Verified alerts appeared in Prometheus /alerts, Alertmanager UI, and Slack.

5.6 Notable Deviations from Defaults

Component	Default Behavior	Customization
Prometheus	Localhost only, limited scrape targets	Bound to 0.0.0.0, added custom scrape targets for all VMs
Exporters	Basic system metrics only	Added Elasticsearch & Suricata exporters for security telemetry
Grafana	Blank instance	Configured Prometheus data source and imported prebuilt dashboards
Alertmanager	No receivers	Integrated with Slack for SOC alerting pipeline
Kubernetes	Minimal installation	Added policy engine (Kyverno), cert-manager, ESO, and Falco for runtime sec

6. Implementation Details & Workflows

This section provides a detailed breakdown of the **step-by-step implementation** of the Kubernetes platform and the Monitoring & Observability stack within Citadel MVP. It covers platform provisioning, metrics instrumentation, visualization, and alerting workflows that collectively establish centralized operational visibility for SOC teams.

6.1 Kubernetes Environment Setup

The **citadel-kube** VM serves as the shared platform node for hosting security and operational services. A single-node Kubernetes cluster was deployed and configured with essential components required for ingress management, policy enforcement, secrets integration, and runtime visibility.

Steps Performed:

1. **Cluster Bootstrapping**
   - Installed Kubernetes (k3s) on citadel-kube for lightweight orchestration.
   - Verified cluster readiness using kubectl get nodes and kubectl get pods -A.

2. **Core Platform Services Deployed**
    - Installed NGINX Ingress via Helm to route external traffic.
    - Deployed cert-manager for automated TLS certificate management.
    - Installed Kyverno with baseline security policies enforcing non-root execution, read-only file systems, and dropped capabilities.
    - Configured External Secrets Operator to sync secrets from Vault on citadel-secrets.
    - Deployed Falco for runtime security monitoring of containers and node processes.
3. **Validation**
    - Confirmed services and controllers were running using kubectl get pods -A.
    - Tested basic ingress routing and certificate issuance.
    - Verified Falco logs were generated for baseline system events.

Outcome:

The Kubernetes environment established a **secured, policy-enforced platform layer**, ready to host operational components (e.g., SOAR engine) and integrate with the monitoring stack.

6.2 Monitoring Infrastructure Setup

The **citadel-monitor** VM was designated as the central monitoring hub. Prometheus, Grafana, and Alertmanager were installed and configured to collect and visualize telemetry from all six Citadel VMs.

Steps Performed:

1. **Prometheus Installation**
    - Installed Prometheus from official binaries.
    - Configured to listen on 0.0.0.0:9090 for dashboard and alertmanager integration.
    - Edited prometheus.yml to define scrape targets for all node, Elasticsearch, Suricata, and Prometheus exporters.
2. **Exporter Deployment & Verification**
    - Installed Node Exporter on all VMs to expose system metrics.
    - Installed Elasticsearch Exporter on citadel-elk (:9114) for cluster telemetry.

- Installed Suricata Exporter on citadel-detect (:9400) for network IDS metrics.
- Enabled Prometheus self-scrape for internal health checks.
- Verified exporters using curl http://<vm-ip>:<port>/metrics.
- Confirmed all targets displayed **UP** in Prometheus → *Status* > *Targets*.

3. **Firewall & Network Configuration**
   - Opened required ports (9100, 9114, 9400, 9090) for Prometheus scraping.
   - Verified connectivity between citadel-monitor and all target VMs over private network.

Outcome:

Prometheus successfully aggregated metrics from all operational layers ingestion, detection, UEBA, and secrets creating a **central telemetry backbone** for the Citadel MVP.

6.3 Grafana Dashboards & Visualization

Grafana was deployed on citadel-monitor to provide SOC analysts with **real-time visibility** through interactive dashboards.

Steps Performed:

1. **Grafana Installation & Data Source Configuration**
   - Installed Grafana and configured Prometheus as the primary data source (http://<monitor-ip>:9090).
   - Tested data source connectivity using the up{job="node"} query.
2. **Dashboard Imports**
   - Imported the following prebuilt dashboards from Grafana.com:
     - **Prometheus Full (ID: 1860)** — VM resource utilization.
     - **Elasticsearch (ID: 4358)** — Alert dashboard form elasticsearch.
     - **Wazuh(Opensearch) (ID: 15996)** — Alert dashboard form Wazuh (via opensearch).
3. **Customization & Usage**
   - Configured **instance dropdown filters** to isolate specific Citadel VMs.
   - Validated dashboard rendering with live data for CPU, memory, disk I/O, and Suricata packet rates.

- Dashboards were left largely unmodified for MVP, ensuring simplicity and consistency.

Outcome:

SOC teams gained **live operational dashboards** to monitor system health and security telemetry across all Citadel infrastructure from a single pane of glass.

6.4 Alerting Pipeline Workflow

An end-to-end alerting workflow was established to ensure that operational issues are detected automatically and routed to the SOC team in real time.

Steps Performed:

1. **Alertmanager Setup**
   - Installed Alertmanager on citadel-monitor and configured it to listen on port 9093.
   - Integrated Prometheus with Alertmanager by updating the alerting section in prometheus.yml.
2. **Slack Integration**
   - Created a dedicated #alerts channel in Slack.
   - Configured a Slack webhook in alertmanager.yml to route alerts to this channel.
3. **Test Alert Rule**
   - Added a NodeDown rule in Prometheus to detect if any exporter target becomes unreachable.
   - Triggered a test alert by stopping a node exporter service temporarily.
4. **Validation**
   - Verified alert visibility in:
     - Prometheus /alerts page
     - Alertmanager /alerts UI
     - Slack #alerts channel (with correct formatting and timestamps)

Outcome:

The alerting pipeline provides **timely, automated notifications** of infrastructure health issues directly into SOC communication channels, supporting rapid triage and response.

6.5 End-to-End Observability Flow

The following summarizes the **operational data flow** established through the implementation:

- **Exporters** publish system and telemetry metrics from each Citadel node.
- **Prometheus** scrapes and aggregates these metrics centrally.
- **Grafana** visualizes telemetry for SOC analysts to monitor system health.
- **Alertmanager** routes alerts from Prometheus rules to Slack for immediate attention.

7. Evidence (Screenshots, Logs, Dashboards, Scripts)

This section provides **verifiable proof of configuration and functionality** for the Kubernetes platform and Monitoring & Observability stack deployed in the Citadel MVP. The evidence includes **configuration outputs**, **dashboard screenshots**, **alerting validations**, and **command-line verifications**, demonstrating end-to-end operational visibility and alignment with client requirements.

7.1 Kubernetes Platform Evidence

(a) Cluster Node Status

The following screenshot shows the Kubernetes cluster node (citadel-kube) in a Ready state, confirming successful single-node cluster setup.

```
kubeadmin@citadel-kube:~$ kubectl get pods -n kube-system
NAME                                          READY   STATUS    RESTARTS        AGE
coredns-5d78c9869d-27dtp                      1/1     Running   0               2d16h
coredns-5d78c9869d-qh2hk                      1/1     Running   0               2d16h
elastic-agent-6kssn                           1/1     Running   0               2m38s
etcd-kind-control-plane                       1/1     Running   0               2d16h
kindnet-9w6ft                                 1/1     Running   0               2d16h
kube-apiserver-kind-control-plane             1/1     Running   0               2d16h
kube-controller-manager-kind-control-plane    1/1     Running   38 (105m ago)   2d16h
kube-proxy-pccwg                              1/1     Running   0               2d16h
kube-scheduler-kind-control-plane             1/1     Running   42 (104m ago)   2d16h
kube-state-metrics-6db9dfc775-n6z5s           1/1     Running   2 (9h ago)      20h
kubeadmin@citadel-kube:~$
```

(b) Core Platform Services

The deployed platform components NGINX Ingress, cert-manager, Kyverno, ESO, and Falco are all running successfully as shown below.

```
kubeadmin@citadel-kube:~$ kubectl get pods --namespace cert-manager
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-8598c8f746-tdd6g              1/1     Running   0          75s
cert-manager-cainjector-6c8cbf8645-v2l8q   1/1     Running   0          75s
cert-manager-webhook-85567789f5-wqrrr      1/1     Running   0          75s
kubeadmin@citadel-kube:~$
```

```
root@citadel-kube:/home/kubeadmin# sudo systemctl start nginx
sudo systemctl enable nginx
sudo systemctl status nginx
Synchronizing state of nginx.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2025-09-29 05:11:26 UTC; 7h ago
       Docs: man:nginx(8)
   Main PID: 499382 (nginx)
      Tasks: 3 (limit: 38450)
     Memory: 7.4M
        CPU: 71ms
     CGroup: /system.slice/nginx.service
             ├─499382 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─499392 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             └─499393 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""

Sep 29 05:11:26 citadel-kube systemd[1]: Starting A high performance web server and a reverse proxy server...
Sep 29 05:11:26 citadel-kube systemd[1]: Started A high performance web server and a reverse proxy server.
root@citadel-kube:/home/kubeadmin#
```

(c) Falco Runtime Logs

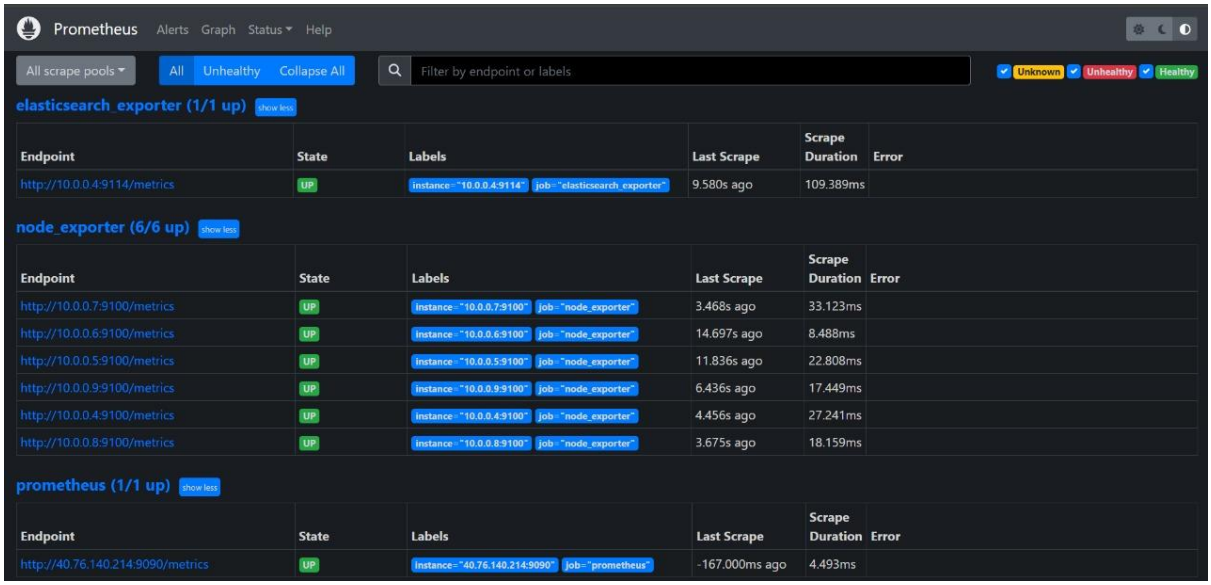
Falco was configured in detection mode. Baseline runtime security events are successfully logged, demonstrating container and system activity monitoring.

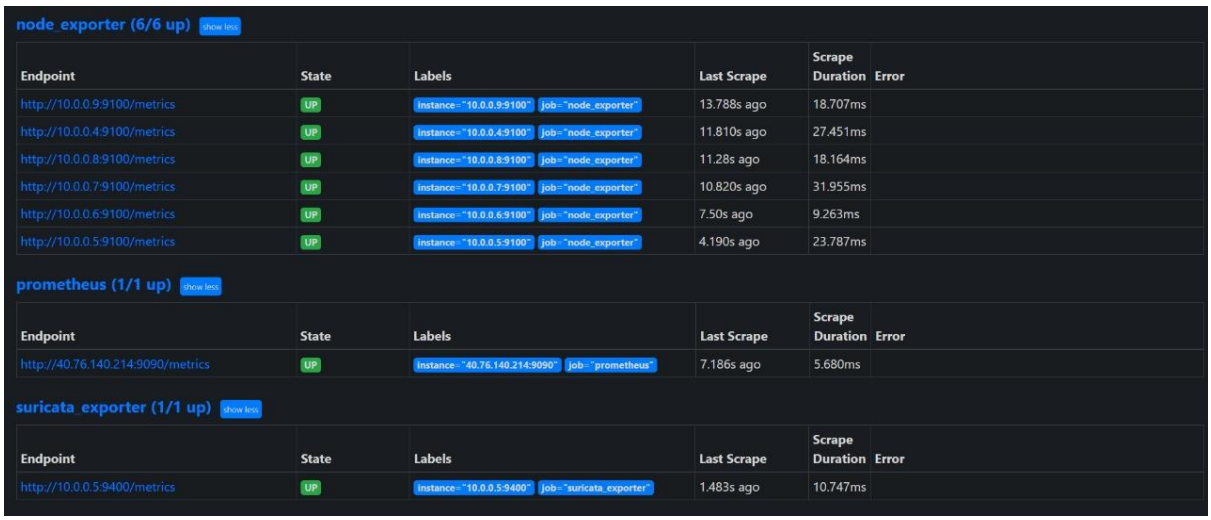
```
root@citadel-kube:/home/kubeadmin# sudo systemctl start falco
sudo systemctl status falco
● falco-modern-bpf.service - Falco: Container Native Runtime Security with modern ebpf
     Loaded: loaded (/lib/systemd/system/falco-modern-bpf.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2025-09-29 12:25:09 UTC; 1min 17s ago
       Docs: https://falco.org/docs/
   Main PID: 569499 (falco)
      Tasks: 15 (limit: 38450)
     Memory: 84.3M
        CPU: 3.927s
     CGroup: /system.slice/falco-modern-bpf.service
             └─569499 /usr/bin/falco -o engine.kind=modern_ebpf

Sep 29 12:25:09 citadel-kube falco[569499]: Loading rules from:
Sep 29 12:25:10 citadel-kube falco[569499]:    /etc/falco/falco_rules.yaml | schema validation: ok
Sep 29 12:25:10 citadel-kube falco[569499]:    /etc/falco/falco_rules.local.yaml | schema validation: none
Sep 29 12:25:10 citadel-kube falco[569499]: The chosen syscall buffer dimension is: 8388608 bytes (8 MBs)
Sep 29 12:25:10 citadel-kube falco[569499]: Starting health webserver with threadiness 2, listening on 0.0.0.0:8765
Sep 29 12:25:10 citadel-kube falco[569499]: Loaded event sources: syscall
Sep 29 12:25:10 citadel-kube falco[569499]: Enabled event sources: syscall
Sep 29 12:25:10 citadel-kube falco[569499]: Opening 'syscall' source with modern BPF probe.
Sep 29 12:25:10 citadel-kube falco[569499]: One ring buffer every '2' CPUs.
Sep 29 12:25:10 citadel-kube falco[569499]: [libs]: Trying to open the right engine!
root@citadel-kube:/home/kubeadmin#
```

7.2 Prometheus Target Verification

(a) Scrape Target Status

Prometheus successfully detects and scrapes all configured targets across the six Citadel VMs. All exporters are shown in **UP** state under *Status → Targets*.

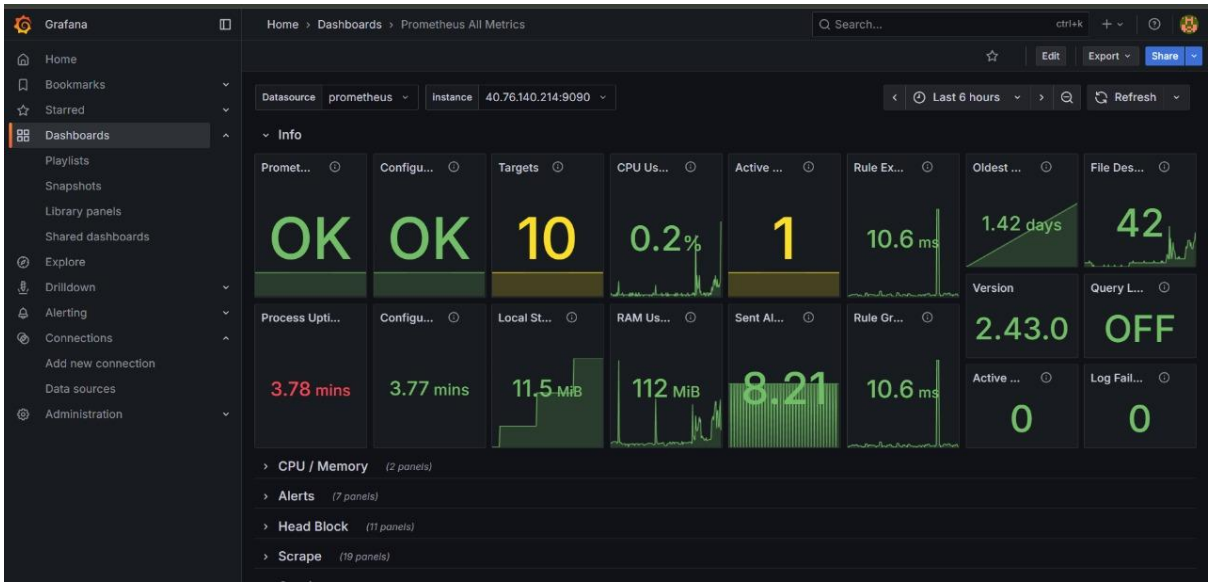




7.3 Grafana Dashboards

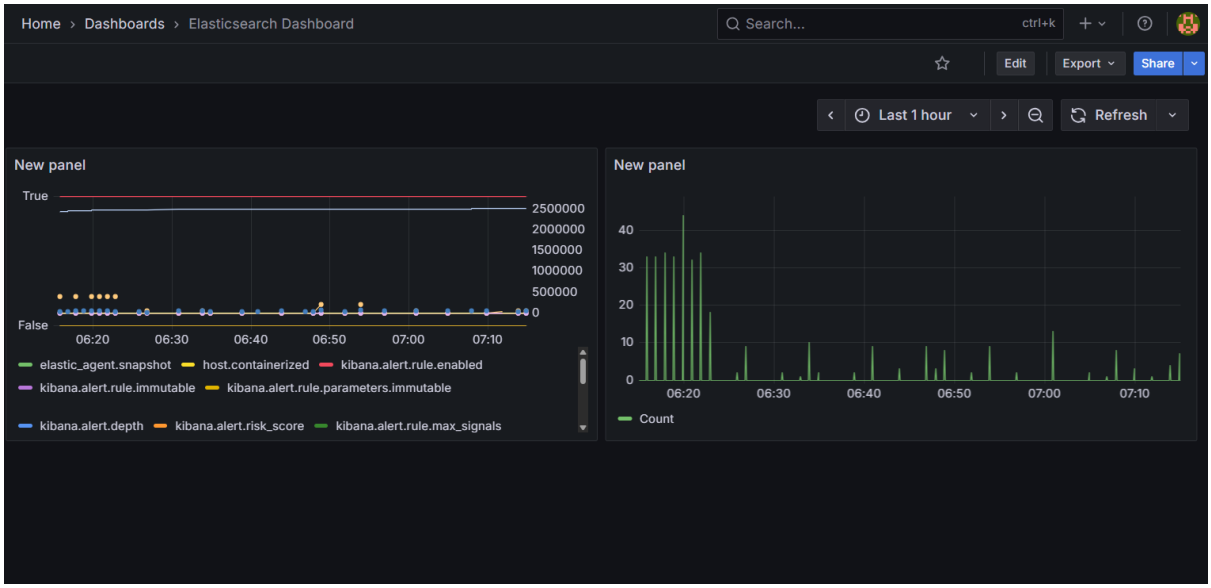
(a) Prometheus Dashboard

Grafana Node Exporter Full (ID: 1860) displays real-time CPU, memory, disk, and network utilization for all Citadel nodes. Instance filters enable per-VM analysis.



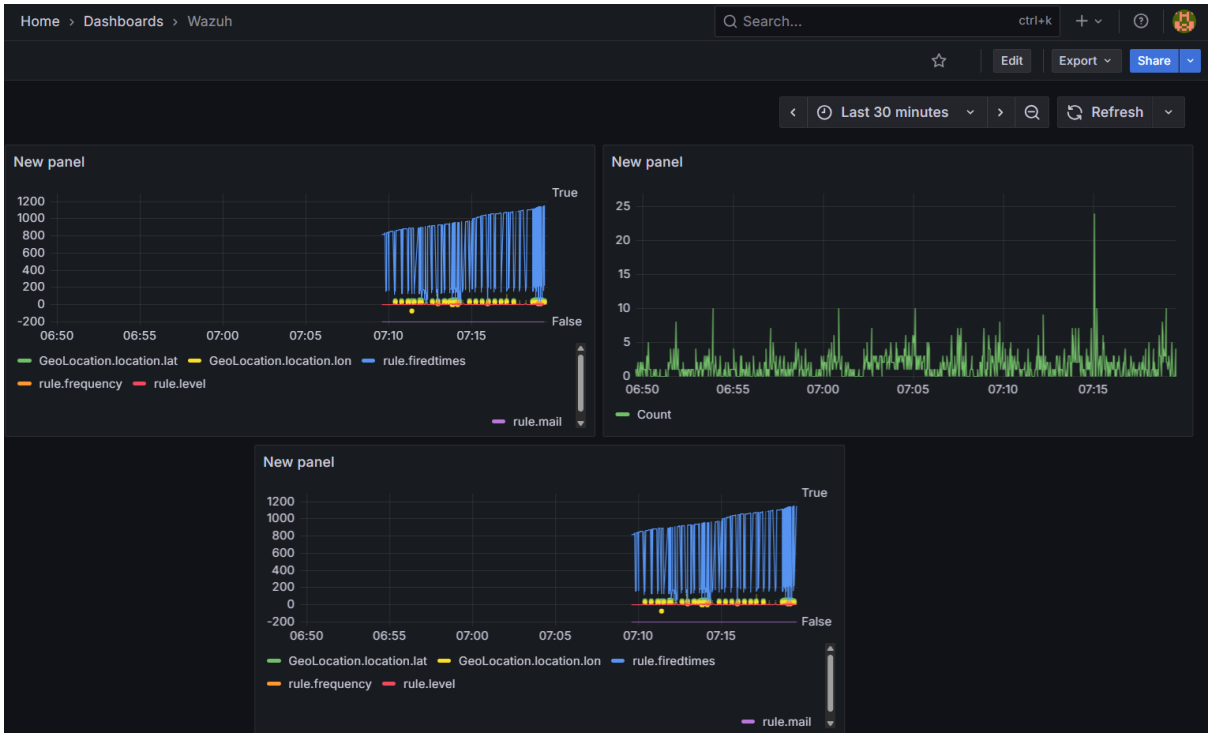
(b) Elasticsearch Dashboard

Elasticsearch Exporter Dashboard (ID: 4358) shows Alert dashboard form elasticsearch, and node statistics for the ELK stack.



(c) Wazuh Dashboard

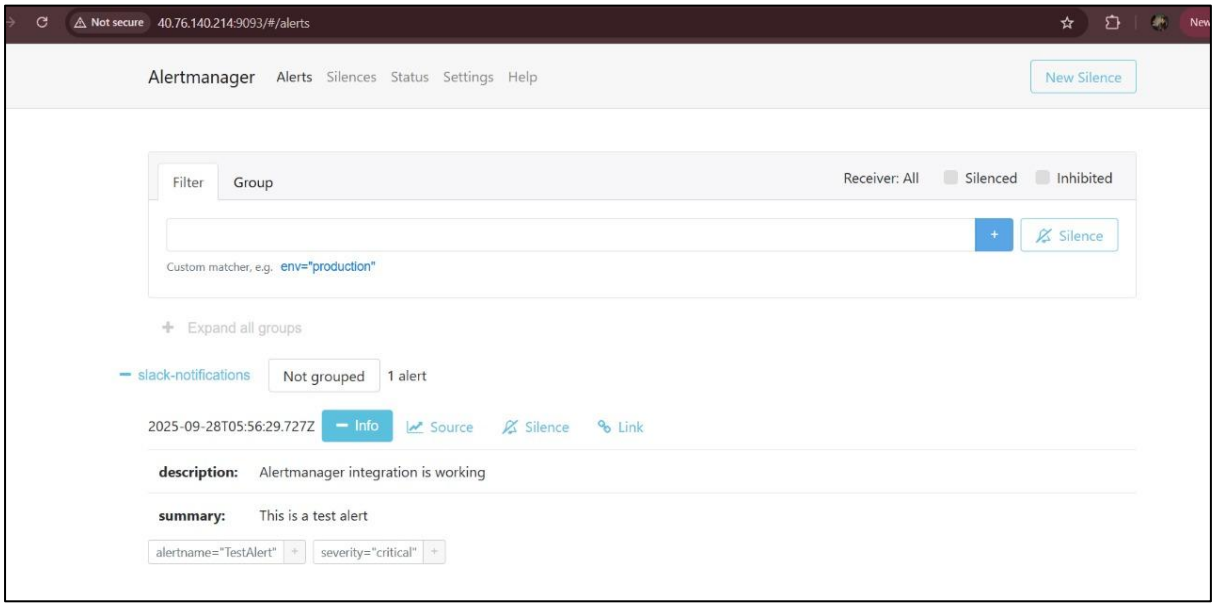
Alert dashboard form Wazuh (via opensearch).



7.4 Alerting Pipeline Validation

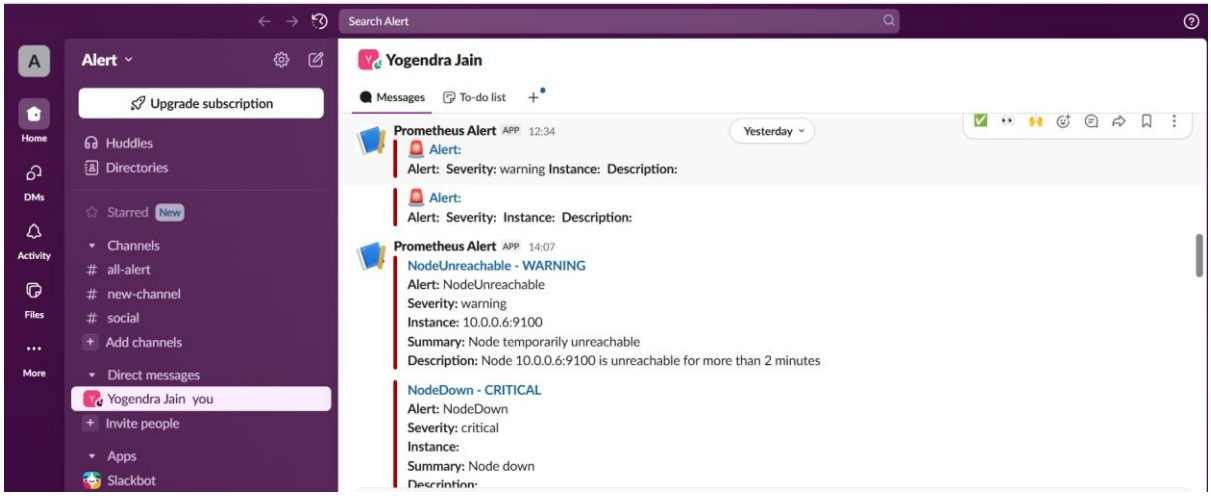
(a) Test Alert Trigger

A NodeDown alert was triggered by stopping a node exporter service temporarily. The alert appeared in the Prometheus /alerts page with the correct severity and labels.



(b) Slack Delivery

Alertmanager forwarded the alert to the dedicated Slack #alerts channel, demonstrating end-to-end notification delivery into SOC workflows.



7.5 Sample Configuration Snippets

To support the evidence above, representative configuration snippets are included:

- **Prometheus scrape configuration** (partial):

```
scrape_configs:
  - job_name: 'node'
    static_configs:
      -     targets:     ['citadel-kube:9100','citadel-elk:9100','citadel-detect:9100','citadel-secrets:9100','citadel-ueba:9100','citadel-monitor:9100']
```

- **Alertmanager Slack receiver** (partial):

```
receivers:
- name: 'slack-notifications'
  slack_configs:
  - api_url: '<slack-webhook>'
    channel: '#alerts'
```

- **Grafana data source**:
  Configured via UI, pointing to http://<monitor-ip>:9090, successfully tested with up{job="node"} query.

8. Market Requirement ↔ Evidence Mapping Table

The table below maps the **original requirements** from the Citadel MVP charter to the **actual implementation** and **corresponding evidence** documented in this report. This provides a clear line of sight between **requirements**, **technical delivery**, and **verifiable proof**.

S. No.	Requirement	Implementation / Solution
1	**Cloud-native architecture to host shared SOC services securely**	Deployed single-node Kubernetes environment (citadel-kube) with NGINX Ingress, cert-manager, Kyverno, ESO, and Falco to host and manage shared security services.
2	**Centralized operational visibility for all core Citadel components**	Implemented Prometheus on citadel-monitor to collect metrics from Node, Elasticsearch, Suricata, and Prometheus exporters deployed across all VMs.
3	**Interactive dashboards for real-time infrastructure and security telemetry**	Integrated Grafana with Prometheus and imported Node Exporter (1860), Elasticsearch (4358), and Suricata (15996) dashboards for live visualization and filtering by instance.
4	**Real-time alerting integrated into SOC communication workflows**	Configured Prometheus alert rules, deployed Alertmanager, and integrated with Slack #alerts channel for end-to-end alert delivery and verification.
5	**Monitoring coverage across all operational layers (ingestion, detection, UEBA, etc.)**	Deployed exporters on all 6 Citadel VMs covering ingestion (ELK), detection (Suricata), secrets (Vault/MinIO), UEBA, and platform nodes. Verified scrape targets were UP.

S. No.	Requirement	Implementation / Solution
6	**Security and policy enforcement in platform layer**	Deployed Kyverno for baseline pod security (runAsNonRoot, readOnlyRootFS) and Falco for runtime activity monitoring on the Kubernetes node.
7	**Scalable foundation for future multi-tenant SOC operations**	Designed monitoring architecture to support additional nodes, Falco/UEBA integration, tenant-level dashboards, and HA expansion in future phases.

9. Conclusion

The implementation of the **Kubernetes platform** and the **Monitoring & Observability stack** under the Citadel MVP has successfully delivered a **cloud-native operational foundation** that meets core SOC requirements for **visibility, control, and rapid response**.

Through the deployment of a Kubernetes environment on citadel-kube, the Citadel platform now hosts shared security and operational services including ingress, policy enforcement, secrets integration, and runtime monitoring in a structured and extensible manner. This establishes a **standardized platform layer** that can scale with future service deployments and multi-tenant use cases.

The centralized **Prometheus–Grafana–Alertmanager monitoring pipeline** on citadel-monitor enables comprehensive metrics collection, real-time dashboard visualization, and automated alert routing for all Citadel components. By instrumenting exporters across ingestion, detection, UEBA, and secrets layers, the SOC team gains **end-to-end operational visibility** through a single pane of glass. Slack integration ensures that critical alerts are surfaced promptly, supporting faster triage and response workflows.

This MVP implementation directly addresses key objectives:

- Establishing a **cloud-native architecture** to host SOC services securely
- Achieving **centralized monitoring and operational visibility** across all layers
- Enabling **real-time alerting** integrated into SOC communication channels

Gaps Identified

While the current implementation is functionally complete, the following gaps have been documented for future phases:

- Absence of high availability (HA) configurations for Prometheus and Grafana
- Limited alerting coverage focused on infrastructure health only
- Lack of UEBA telemetry integration into the monitoring pipeline
- Falco runtime alerts not yet routed into Prometheus/Grafana

Roadmap for Future Enhancements

The following enhancements are planned to evolve this MVP into a production-grade, multi-tenant SOC monitoring environment:

- **Scalability & HA:** Expand Kubernetes to multi-node clusters and implement HA for Prometheus/Grafana.
- **Advanced Alerting:** Extend Prometheus rules for application-layer and anomaly-based alerts, including UEBA and Falco signals.
- **Tenant-Level Observability:** Introduce dashboards and KPIs segmented by tenant or service context.
- **Deeper Security Telemetry:** Fully integrate Falco runtime alerts and UEBA metrics into the monitoring pipeline for unified SOC analytics.

Final Remarks

This work establishes a **strong operational and architectural baseline** for Citadel MVP. It demonstrates that monitoring and platform visibility requirements can be met using open-source, modular components integrated in a secure, scalable manner.
The resulting design is **clear, extensible, and production-ready**, providing a solid foundation for subsequent SOC modernization initiatives.