Kubernetes Auditing and Monitoring with Falco

# 1. Introduction

Security observability in Kubernetes is vital for detecting threats, enforcing compliance, and maintaining system integrity. Falco, an open-source project by Sysdig and part of the CNCF, provides real-time security monitoring and auditing for Kubernetes workloads.

# 2. What is Falco?

Falco is a cloud-native runtime security tool that detects unexpected behavior in applications, containers, and Kubernetes environments by monitoring system calls.

It acts as an intrusion and anomaly detection system by enforcing rules and alerting on suspicious activity, such as unauthorized file access or privilege escalation.

# 3. How Falco Works

- Falco uses Linux kernel features (like eBPF or a kernel module) to monitor system calls.  
- It processes events and matches them against a set of customizable rules.  
- When a rule is violated, Falco generates an alert in real-time.  
- It integrates well with Kubernetes via audit logs and metadata enrichment.

# 4. Installing Falco in Kubernetes

You can deploy Falco as a DaemonSet using Helm:

helm repo add falcosecurity https://falcosecurity.github.io/charts  
helm repo update  
helm install falco falcosecurity/falco

# 5. Kubernetes Audit Log Integration

Falco can consume Kubernetes audit logs using the `falco-audit-logger` plugin.

Steps:  
- Enable audit logging on the Kubernetes API server.  
- Configure the audit policy file.  
- Use a webhook to send logs to the `falco-audit-logger`.

Sample Audit Policy:

apiVersion: audit.k8s.io/v1  
kind: Policy  
rules:  
 - level: Metadata  
 resources:  
 - group: ""  
 resources: ["pods"]

# 6. Falco Rules

Falco uses rules to define what constitutes suspicious behavior. These rules are written in YAML and can be customized.

Example Rule: Alert on shell in container:

- rule: Shell in Container  
 desc: Detect a shell running inside a container  
 condition: container.id != host and proc.name = bash  
 output: Shell spawned in container (user=%user.name command=%proc.cmdline)  
 priority: WARNING

# 7. Alerting and Integrations

Falco can send alerts via multiple channels:  
- stdout (logs)  
- syslog  
- gRPC output for custom integrations  
- Webhooks (e.g., to Slack, Teams, SIEM systems)  
Tools like Falcosidekick enhance these capabilities by supporting more destinations.

# 8. Best Practices

- Start with default rules, then refine and extend based on your environment.  
- Enable Kubernetes audit log integration for broader visibility.  
- Use Falcosidekick for extended alerting.  
- Regularly test and tune rule sets.  
- Use RBAC to limit sensitive operations in Kubernetes.

# 9. Tools & Visualization

- Falcosidekick UI: Web interface for viewing Falco alerts.  
- Elasticsearch + Kibana: Centralized logging and visualization.  
- Prometheus & Grafana: Metrics and dashboarding integration.

# 10. Summary

Falco offers powerful auditing and runtime threat detection for Kubernetes. It integrates deeply with Kubernetes components, providing visibility into both system and application behaviors.

With Falco, you can enforce runtime security policies and detect potential threats before they escalate.