Kubernetes Security Fundamentals

Audit Logging

Audit logging in Kubernetes is a powerful tool for tracking and monitoring activities within the cluster. It provides a detailed record of operations, allowing administrators to detect unauthorized access, monitor compliance, and perform forensic analysis in case of security incidents.

Audit logs capture information about all requests made to the Kubernetes API Server. This includes the request details, the user making the request, the resource being accessed, and the outcome of the request. Configuring audit logging involves setting up audit policies and ensuring logs are securely stored and monitored.

RealLife Example:

Imagine a well-managed kingdom with meticulous record-keeping of all significant events. Just like these detailed logs provide valuable insights into the kingdom's operations and help identify any irregularities, audit logging in Kubernetes allows you to track activity within your cluster and detect potential security threats.

Why is Audit Logging Essential in Kubernetes?

- **Security Investigations:** Audit logs provide a historical record of all user actions, API requests, and cluster events. These logs are invaluable for investigating security incidents, understanding the root cause of problems, and identifying suspicious activity. Imagine the king's advisors analyzing the royal records to investigate a security breach or identify any suspicious events within the kingdom.
- **Compliance Requirements:** Many security regulations mandate the logging and auditing of user activity within IT systems. Robust audit logs help demonstrate compliance with these regulations and provide evidence of your security posture. Imagine the records acting as proof of the kingdom's adherence to strict security protocols and proper management.
- **Troubleshooting Issues:** Audit logs can be instrumental in troubleshooting operational issues within your cluster. By analyzing logs, you can identify configuration errors, resource bottlenecks, or unexpected

application behavior. Think of the records helping the royal engineers diagnose problems like malfunctioning infrastructure or inefficient resource allocation.

What Does Kubernetes Audit Logging Record?
Kubernetes audit logging captures a comprehensive record of various activities, including:

- **API requests and responses:** Tracks all interactions with the Kubernetes API Server, including user actions and application requests. Imagine recording every audience granted by the king, along with the visitor and the purpose of the visit.
- **Authentication and authorization events:** Logs user login attempts, successful authentications, and authorization decisions made by RBAC. Imagine documenting who entered the kingdom, how they gained access (credentials), and what permissions they were granted.
- **Cluster configuration changes:** Tracks modifications to cluster resources like deployments, pods, or namespaces. Imagine recording any changes made to the kingdom's infrastructure, laws, or district boundaries.

Best Practices for Audit Logging in Kubernetes:

- **Enable Audit Logging:** Ensure audit logging is enabled in your Kubernetes cluster configuration. Imagine the king mandating the scribes to diligently record all important events within the kingdom.
- **Define Logging Level:** Choose an appropriate logging level, balancing detail with log volume. Too much detail can overwhelm analysis, while too little might miss crucial information. Imagine the records capturing essential events without becoming excessively verbose.
- **Centralize Log Collection:** Aggregate logs from all cluster components for easier analysis and storage. Imagine all the royal records being collected and stored in a central archive for easy access.
- **Log Retention Policy:** Establish a policy for how long to retain audit logs, considering compliance requirements and storage capacity. Imagine the kingdom archiving records for a specific duration based on regulations and available space.
- **Utilize Log Analysis Tools:** Leverage tools to analyze and visualize audit logs for efficient identification of security incidents or trends. Imagine

the king's advisors employing sophisticated tools to analyze the records and identify patterns or suspicious activity.

Link to documentation:
https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/

Lab Exercise: Configuring and Managing Kubernetes Audit Logging
Objective
In this lab, you will learn how to configure and manage audit logging in Kubernetes. You will set up audit policies, enable audit logging, and integrate logs with a centralized logging solution.

Prerequisites
- A running Kubernetes cluster
- kubectl configured to interact with your cluster

Step-by-Step Instructions
Step 1: Create an Audit Policy
1. **Define an Audit Policy**
   - Create a YAML file for the audit policy.

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

```
        users:
["system:serviceaccount:default:my-service-account"]
        verbs: ["create", "update", "delete"]
        resources:
        - group: ""
          resources: ["secrets"]
```

2. **Save the Audit Policy**
   - Save the policy file as audit-policy.yaml.

Step 2: Configure the API Server to Use the Audit Policy

1. **Edit the API Server Manifest**
   - Edit the API Server manifest (usually located in /etc/kubernetes/manifests/kube-apiserver.yaml) to include audit logging flags.

```
      -
--audit-policy-file=/etc/kubernetes/audit-policy.yaml
      - --audit-log-path=/var/log/kubernetes/audit.log
      - --audit-log-maxage=30
      - --audit-log-maxbackup=10
      - --audit-log-maxsize=100
```

2. **Save and Apply Changes**
   - Save the changes to the API Server manifest. The API Server will automatically restart and apply the new configuration.

Step 3: Verify Audit Logging Configuration

1. **Create and Apply a Pod**
   - Create a pod to generate audit logs.

```
      apiVersion: v1
      kind: Pod
      metadata:
        name: audit-log-test
```

```
  spec:
    containers:
    - name: nginx
      image: nginx
```

- Apply the pod configuration.

```
kubectl apply -f audit-log-test.yaml
```

2. Check the Audit Logs
   - View the audit logs to verify the configuration.

```
cat /var/log/kubernetes/audit.log
```