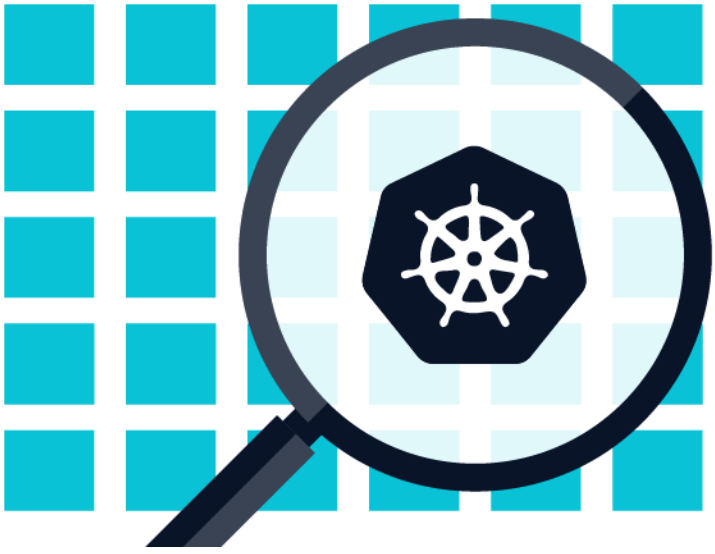
CKA Notes- Monitoring, Logging and Alerting in Kubernetes

Cloud Champ

Hey there, I'm Nasiullha, a remote DevOps engineer, freelancer, and YouTuber at Cloudchamp. With a deep passion for cloud computing and DevOps. As a freelancer, I've helped clients design

https://www.youtube.com/@cloudchamp?sub_confirmation=1

Monitoring



Why Monitoring?

In Kubernetes, monitoring involves tracking various metrics to ensure the health and performance of your cluster and applications. Metrics can include resource usage,

application performance, and other key indicators.

Example:

- Monitoring CPU and memory usage of pods to identify potential performance bottlenecks.

How Monitoring is Done in Kubernetes?

- **Metric Server (Heapster):**
  - Metric Server is a component that collects metrics from the Kubernetes API server and makes them available for querying.
  - Installation steps and commands vary based on the Kubernetes distribution. For example, using Minikube:

```
minikube addons enable metrics-server
```

- **Exposure to Cloud or Open Source Solutions:**
  - Metrics collected by tools like Prometheus can be exposed for visualization and analysis.
- **Adding in Code (Custom Metrics):**
  - Developers can instrument their code to expose custom metrics specific to their applications.
  - Example: Adding custom metrics in a Node.js application.

```
const prometheus = require('prom-client');
prometheus.collectDefaultMetrics();

const httpRequestDurationMicroseconds = new prometheus.Histogram({
  name: 'http_request_duration_seconds',
  help: 'Duration of HTTP requests in seconds',
  labelNames: ['method', 'route'],
  buckets: [0.1, 0.5, 1, 1.5, 2, 2.5, 3],
});
```

- **Service Mesh:**

- Service meshes like Istio provide additional monitoring capabilities, including traffic metrics, error rates, and more.

Installation of Metric Server

Minikube:

```
minikube addons enable metrics-server
```

Helm:

```
helm install stable/metrics-server
```

GitHub:

Refer to the official GitHub repository for installation instructions.

Monitoring Commands

- **List Node Metrics:**

  ```
  kubectl top node
  ```

- **List Pod Metrics:**

  ```
  kubectl top pod
  ```

Alerting

Why Alerting?

Alerting in Kubernetes is crucial for early detection and troubleshooting of issues, allowing for proactive problem resolution.

How Alerting?

- **Using Different Tools:**
  - Prometheus AlertManager
  - Grafana
  - Kubernetes-native tools like kube-state-metrics
- **Tools for Alerts:**
  - Prometheus AlertManager
  - Grafana Alerts
  - External tools like PagerDuty or Slack integrations
- **Alerting Commands:**

```
# Example command to set up a Prometheus alerting rule
kubectl apply -f prometheus-alert-rules.yaml
```

  - Ensure to set alerts for specific conditions to avoid unnecessary noise.

Visualization

To enhance visibility, metrics can be presented visually using charts, graphs, and other forms of data representation. Tools like Grafana are commonly used for this purpose.

Logging

Why Logging

Logging is essential for gaining insights into application behavior, troubleshooting issues, and meeting certification requirements like CKA exams.

How Logging

- **Tools Used:**
  - Fluentd
  - Elasticsearch
  - Kibana
- **Logging Commands:**

```
# Port-forward Elasticsearch for querying logs
kubectl port-forward svc/elasticsearch 9200
```

  - Creating scripts or command line tools for automating common log-related tasks is a best practice.

> 💡 Set alerts for specific events based on log analysis to proactively identify and address potential issues.