**Kubernetes Monitoring with Prometheus and Grafana - Notes**

**1. Introduction**

* **Course:** Day 42 of the Complete DevOps Course
* **Topic:** Kubernetes Monitoring with **Prometheus** and **Grafana**
* **GitHub Repository:** Includes installation steps and demo commands.
* **Prerequisite:** A Kubernetes cluster (Minikube, K3s, Kind, or a production setup).

**2. Importance of Monitoring**

* **For Single Clusters:** Manual management is feasible.
* **For Multiple Clusters & Teams:**
  + Difficult to track issues without monitoring.
  + Essential for visibility into deployments, services, and resource usage.
* **For Scaling Environments:**
  + As clusters grow (Dev, Staging, Production), proactive monitoring is critical.

**3. Overview of Prometheus**

**What is Prometheus?**

* Open-source monitoring system initially developed by **SoundCloud**.
* Collects and stores metrics in a **time-series database**.
* Uses **PromQL** (Prometheus Query Language) for querying metrics.

**Prometheus Architecture:**

* **Prometheus Server:** Scrapes metrics from the Kubernetes API server.
* **Time-Series Database:** Stores metrics with timestamps.
* **Alert Manager:** Sends alerts via Slack, Email, etc.
* **Exporters:** Fetch additional metrics (e.g., Node Exporter, kube-state-metrics).

**Default Kubernetes Metrics:**

* The Kubernetes API server exposes basic metrics at the /metrics endpoint.
* **kube-state-metrics** provides deeper insights into deployments, pods, and service statuses.

**4. Overview of Grafana**

**What is Grafana?**

* A visualization tool for monitoring data.
* Supports multiple data sources (Prometheus, Elasticsearch, etc.).
* Provides **dashboards** for better representation of metrics.

**Why Use Grafana?**

* Prometheus provides raw data in JSON format.
* Grafana converts this data into **graphs, charts, and dashboards**.

**5. Setting Up Monitoring on Minikube**

**Step 1: Start Minikube Cluster**

minikube start --memory=4g --driver=hyperkit

*(Use hyperkit for Mac, or docker/virtualbox for other OS.)*

**Step 2: Install Prometheus using Helm**

# Add Prometheus Helm repo

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo update

# Install Prometheus

helm install prometheus prometheus-community/prometheus

* Verify installation:

kubectl get pods

kubectl get svc

**Step 3: Expose Prometheus UI**

kubectl expose service prometheus-server --type=NodePort --name=prometheus-server-ext

* Access Prometheus UI:

minikube ip

# Open browser: http://<minikube-ip>:<NodePort>

**Step 4: Install Grafana using Helm**

# Add Grafana Helm repo

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

# Install Grafana

helm install grafana grafana/grafana

* Retrieve Grafana admin password:

kubectl get secret grafana -o jsonpath="{.data.admin-password}" | base64 --decode

* Expose Grafana:

kubectl expose service grafana --type=NodePort --name=grafana-ext

* Access Grafana UI:

http://<minikube-ip>:<NodePort>

**Step 5: Configure Grafana with Prometheus**

1. Go to **Data Sources** → **Add Prometheus**.
2. Enter Prometheus server URL: http://prometheus-server:80.
3. Click **Save & Test**.

**Step 6: Import a Dashboard**

* Use **Grafana Dashboard ID 3662** (predefined Kubernetes dashboard).
* Navigate to **Dashboards → Import → Enter ID 3662 → Load**.

**Step 7: Expose kube-state-metrics**

kubectl expose service prometheus-kube-state-metrics --type=NodePort --name=kube-state-metrics-ext --target-port=8080

* Access metrics:

http://<minikube-ip>:<NodePort>/metrics

**6. Advanced Monitoring**

**Custom Application Metrics**

* Applications can expose a /metrics endpoint using **Prometheus client libraries**.
* Update Prometheus ConfigMap to scrape custom metrics:

kubectl edit cm prometheus-server

* Add a new job in scrape\_configs:

- job\_name: 'my-app-metrics'

static\_configs:

- targets: ['<app-service>:<port>']

**7. Key Takeaways**

✅ **Prometheus** = Collects and stores metrics.  
✅ **Grafana** = Visualizes metrics with dashboards.  
✅ **kube-state-metrics** = Enhances Kubernetes monitoring.  
✅ **Custom Metrics** = Expose /metrics endpoints in applications.

**8. Next Steps**

* Explore **Alert Manager** for notifications.
* Learn how to create **custom Prometheus exporters**.
* Experiment with **advanced Grafana dashboards**.

📌 **GitHub Repository:** Link (Commands & Installation Steps)

**End of Notes** 🚀  
Let me know if you need any modifications! 😊

# \*\*Kubernetes Monitoring with Prometheus and Grafana - Notes\*\*

## \*\*1. Introduction\*\*

- \*\*Course:\*\* Day 42 of Complete DevOps Course

- \*\*Topic:\*\* Kubernetes Monitoring using \*\*Prometheus\*\* and \*\*Grafana\*\*

- \*\*GitHub Repository:\*\* Contains installation steps and demo commands for practical learning.

- \*\*Prerequisite:\*\* A Kubernetes cluster (Minikube, K3s, Kind, or production cluster).

---

## \*\*2. Why Monitoring is Required?\*\*

- \*\*Single Cluster:\*\* Easy to manage manually.

- \*\*Multiple Clusters & Teams:\*\*

- Difficult to track issues without monitoring.

- Need visibility into deployments, services, and resource usage.

- \*\*Scaling:\*\* As clusters grow (Dev, Staging, Production), monitoring becomes essential.

---

## \*\*3. Prometheus Overview\*\*

- \*\*What is Prometheus?\*\*

- Open-source monitoring tool (initially developed by SoundCloud).

- Collects and stores metrics in a \*\*time-series database\*\*.

- Uses \*\*PromQL\*\* (Prometheus Query Language) for querying metrics.

- \*\*Architecture:\*\*

- \*\*Prometheus Server:\*\* Scrapes metrics from Kubernetes API server.

- \*\*Time-Series Database:\*\* Stores metrics with timestamps.

- \*\*Alert Manager:\*\* Sends alerts (Slack, Email, etc.) based on rules.

- \*\*Exporters:\*\* Fetch additional metrics (e.g., Node Exporter, kube-state-metrics).

- \*\*Default Kubernetes Metrics:\*\*

- Kubernetes API server exposes basic metrics (`/metrics` endpoint).

- For deeper insights, use \*\*kube-state-metrics\*\* (exposes deployment, pod, service status).

---

## \*\*4. Grafana Overview\*\*

- \*\*What is Grafana?\*\*

- Visualization tool for monitoring data.

- Supports multiple data sources (Prometheus, Elasticsearch, etc.).

- Provides \*\*dashboards\*\* for better representation of metrics.

- \*\*Why Grafana?\*\*

- Prometheus provides raw data (JSON-like format).

- Grafana converts this into \*\*charts, graphs, and dashboards\*\*.

---

## \*\*5. Demo: Setting Up Monitoring on Minikube\*\*

### \*\*Step 1: Start Minikube Cluster\*\*

```sh

minikube start --memory=4g --driver=hyperkit

```

(Use `hyperkit` for better networking on Mac, or `docker`/`virtualbox` for other OS).

### \*\*Step 2: Install Prometheus using Helm\*\*

```sh

# Add Prometheus Helm repo

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo update

# Install Prometheus

helm install prometheus prometheus-community/prometheus

```

- Verify installation:

```sh

kubectl get pods

kubectl get svc

```

### \*\*Step 3: Expose Prometheus UI\*\*

```sh

kubectl expose service prometheus-server --type=NodePort --name=prometheus-server-ext

```

- Access Prometheus UI:

```sh

minikube ip

# Open browser: http://<minikube-ip>:<NodePort>

```

### \*\*Step 4: Install Grafana using Helm\*\*

```sh

# Add Grafana Helm repo

helm repo add grafana https://grafana.github.io/helm-charts

helm repo update

# Install Grafana

helm install grafana grafana/grafana

```

- Get Grafana admin password:

```sh

kubectl get secret grafana -o jsonpath="{.data.admin-password}" | base64 --decode

```

- Expose Grafana:

```sh

kubectl expose service grafana --type=NodePort --name=grafana-ext

```

- Access Grafana UI:

```sh

http://<minikube-ip>:<NodePort>

```

### \*\*Step 5: Configure Grafana with Prometheus\*\*

1. Go to \*\*Data Sources\*\* → \*\*Add Prometheus\*\*.

2. Enter Prometheus server URL (`http://prometheus-server:80`).

3. Save & Test.

### \*\*Step 6: Import a Dashboard\*\*

- Use \*\*Grafana Dashboard ID `3662`\*\* (predefined Kubernetes dashboard).

- Go to \*\*Dashboards → Import → Enter ID `3662` → Load\*\*.

### \*\*Step 7: Expose kube-state-metrics\*\*

```sh

kubectl expose service prometheus-kube-state-metrics --type=NodePort --name=kube-state-metrics-ext --target-port=8080

```

- Access metrics:

```sh

http://<minikube-ip>:<NodePort>/metrics

```

---

## \*\*6. Advanced Monitoring\*\*

### \*\*Custom Application Metrics\*\*

- Developers should expose a `/metrics` endpoint using \*\*Prometheus client libraries\*\*.

- Update Prometheus ConfigMap to scrape custom metrics:

```sh

kubectl edit cm prometheus-server

```

- Add a new job in `scrape\_configs`:

```yaml

- job\_name: 'my-app-metrics'

static\_configs:

- targets: ['<app-service>:<port>']

```

---

## \*\*7. Key Takeaways\*\*

✅ \*\*Prometheus\*\* = Metrics collection & alerting.

✅ \*\*Grafana\*\* = Visualization (dashboards).

✅ \*\*kube-state-metrics\*\* = Extends Kubernetes monitoring beyond API server.

✅ \*\*Custom Metrics\*\* = Developers expose `/metrics` endpoint for app-specific monitoring.

---

## \*\*8. Next Steps\*\*

- Explore \*\*Alert Manager\*\* for notifications.

- Learn \*\*writing custom Prometheus exporters\*\*.

- Try \*\*advanced Grafana dashboards\*\*.

📌 \*\*GitHub Repo:\*\* [Link](#) (Commands & Installation Steps)

---

\*\*End of Notes\*\* 🚀

Hope this helps! Let me know if you need any modifications. 😊