Setting Up OpenSearch, OpenSearch Dashboard, OpenSearch Exporter, Prometheus, and Grafana using Podman

Table of Contents

Objective:

To set up OpenSearch, OpenSearch Dashboard, Prometheus, and Grafana within a Podman environment for managing, visualizing, and monitoring search-related data.

1. Setting Up the Podman Environment

1.1. Creating a Podman Pod:

Command:

```
podman pod create --name opensearch-pod -p 5601:5601 -p 9114:9114
-p 9200:9200 -p 9090:9090 -p 3000:3000
```

Explanation:

- A pod in Podman is similar to a pod in Kubernetes; it groups multiple containers that share network and storage.
- The command creates a pod named `opensearch-pod` and maps the following ports to the host:
  - **5601**: For OpenSearch Dashboard.
  - **9200**: For OpenSearch.
  - **9114**: For metrics exporter.
  - **9090**: For Prometheus.
  - **3000**: For Grafana.

2. Running OpenSearch in a Podman Container

2.1. Pulling and Running the OpenSearch Container:

Command:

```
podman run -d --name opensearch-node1 \
    --pod opensearch-pod \
    -e "cluster.name=opensearch-cluster" \
    -e "node.name=opensearch-node1" \
    -e "bootstrap.memory_lock=true" \
    -e "OPENSEARCH_JAVA_OPTS=-Xms512m -Xmx512m" \
    -e "discovery.type=single-node" \
    -e "OPENSEARCH_INITIAL_ADMIN_PASSWORD=mrJaat@1222" \
    -v opensearch-data:/usr/share/opensearch/data \
    docker.io/opensearchproject/opensearch:latest
```

Explanation:

- Runs an OpenSearch container inside the previously created pod.
- **Environment variables**:
    - `cluster.name`: Specifies the cluster name.
    - `node.name`: Specifies the node name within the cluster.
    - `OPENSEARCH_JAVA_OPTS`: Sets the JVM options.
    - `OPENSEARCH_INITIAL_ADMIN_PASSWORD`: Required for the security plugin, and provides the initial admin password.
- **Volume Mapping**: The data directory inside the container is mapped to a persistent storage volume `opensearch-data`.

3. Running OpenSearch Dashboard in a Podman Container

3.1. Pulling and Running the OpenSearch Dashboard Container:

Command:

```
podman run -d --name opensearch-dashboards \
    --pod opensearch-pod \
    -e "OPENSEARCH_HOSTS=['https://opensearch-node1:9200']" \
    docker.io/opensearchproject/opensearch-dashboards:latest
```

Explanation:

- Runs the OpenSearch Dashboard container inside the same pod.
- **Environment Variable**:
  - `OPENSEARCH_HOSTS`: Specifies the OpenSearch instances that the dashboard will connect to.

4. Running Prometheus Exporter for OpenSearch in a Podman Container

Option 1: Using Justwatch's Docker Hub Repository (Currently Using this)

Command:

```
podman run -d --name opensearch-exporter \
    --pod opensearch-pod \
    -e "LOG_LEVEL=info" \
    -e "ES_CLUSTER_URI=http://localhost:9200" \
    docker.io/justwatch/elasticsearch_exporter:latest
```

Explanation:

- Image Source: The exporter image is pulled from Justwatch's official Docker Hub repository.
- **Environment Variables:**
  - `LOG_LEVEL`: Sets the log level (e.g., `info`).
  - `ES_CLUSTER_URI`: Connects to the OpenSearch instance running on `localhost` inside the pod.

Use Case:

- This option is simpler and may be suitable for basic setups where only a default subset of metrics is needed.

Option 2: Using Prometheus Community's Quay Repository

Command:

```
podman run -d --name opensearch-exporter \
    --pod opensearch-pod \
    -e LOG_LEVEL=info \
    -e ES_CLUSTER_URI=http://opensearch-node1:9200 \
    -e ES_ALL=true \
    quay.io/prometheuscommunity/elasticsearch-exporter:latest
```

Explanation:

- **Image Source:** The exporter image is pulled from the Prometheus Community's Quay repository.
- **Environment Variables:**
  - `LOG_LEVEL`: Sets the log level (e.g., `info`).
  - `ES_CLUSTER_URI`: Connects to the OpenSearch instance by referring to the container name `opensearch-node1`.
  - `ES_ALL`: Enables the collection of all available metrics from the OpenSearch cluster.

Use Case:

- This option is more feature-rich, allowing comprehensive monitoring by collecting all available metrics from OpenSearch. It is more suitable for production environments or scenarios where in-depth monitoring is required.

5. Running Prometheus in a Podman Container

5.1. Pulling and Running the Prometheus Container:

Command:

```
podman run -d --name prometheus \
    --pod opensearch-pod \
    -v prometheus-data:/prometheus \
    -v ./prometheus.yml:/etc/prometheus/prometheus.yml \
    docker.io/prom/prometheus:latest
```

Explanation:

- **Volume Mapping**:
    - `prometheus-data`: Maps the Prometheus data directory inside the container to a persistent storage volume.
    - `prometheus.yml`: Configuration file for Prometheus, which defines the scraping jobs and targets.

6. Running Grafana in a Podman Container

6.1. Pulling and Running the Grafana Container:

Command:

```
podman run -d --name grafana \
    --pod opensearch-pod \
    -v grafana-data:/var/lib/grafana \
    docker.io/grafana/grafana:latest
```

Explanation:

- **Volume Mapping**:
    - `grafana-data`: Maps the Grafana data directory inside the container to a persistent storage volume.
- Once Grafana is running, you can access it on `http://localhost:3000`. The default login credentials are `admin/admin`.

7. Troubleshooting Common Issues

7.1. Missing Initial Admin Password:

Problem:

- OpenSearch will fail to start if the initial admin password is not set.

Solution:

- Set the `OPENSEARCH_INITIAL_ADMIN_PASSWORD` environment variable as shown in the OpenSearch setup command above.

8. Conclusion

Following the steps in this document, you will have successfully set up OpenSearch, OpenSearch Dashboard, Prometheus, and Grafana using Podman. This setup enables efficient management, visualization, and monitoring of search-related data.

References:

- OpenSearch Official Documentation: opensearch.org
- Podman Official Documentation: podman.io
- Prometheus Official Documentation: prometheus.io
- Grafana Official Documentation: grafana.com