Monitoring in Kubernetes with Prometheus and Grafana

01:56:48 Exit Lab Complete Lab

2 hours duration Practitioner Rate this lab

VIDEOS GUIDE

Monitoring in Kubernetes with Prometheus and Grafana

Introduction

Monitoring is an essential part of the overall CI/CD picture. To deploy frequently, you need to be able to have confidence that if a deployment breaks something, you will be able to identify the problem and respond quickly to minimize the impact on users. In this activity, you will learn to install and configure Prometheus and Grafana in a Kubernetes cluster, and you will set up some basic Grafana dashboards to give you insight into the performance of the cluster and the applications running in it.

Log in to the Kubernetes master node using the **Kubernetes Master Public IP** provided in the *Credentials* section of the hands-on lab page.

```
ssh cloud_user@&lt;KUBERNETES_MASTER_PUBLIC_IP&gt;
```

Initialize Helm

```
helm init --stable-repo-url=https://charts.helm.sh/stable --wait --
tiller-image ghcr.io/helm/tiller:v2.8.2
```

Install Prometheus in the Kubernetes Cluster

1. To do this, make sure you have cloned the Kubernetes charts repo:

   ```
   cd ~/
   git clone https://github.com/kubernetes/charts
   ```

```
cd charts
git checkout efdcffe0b6973111ec6e5e83136ea74cdbe6527d
cd ../
```

2. Create a *prometheus-values.yml* file:

```
vi prometheus-values.yml
```

3. And paste in this content:

```
alertmanager:
  persistentVolume:
    enabled: false
server:
  persistentVolume:
    enabled: false
```

4. Save and close the file:

```
:wq
```

5. Use helm to install Prometheus with *prometheus-values.yml*:

```
helm install -f ~/prometheus-values.yml ~/charts/stable/prometheus
--name prometheus --namespace prometheus
```

6. We can see which pods are running in this new namespace with the command:

```
kubectl get pods -n prometheus
```

Install Grafana in the Kubernetes Cluster

1. Create a *grafana-values.yml*:

```
vi grafana-values.yml
```

2. And paste in this content (you will use this password to log in to Grafana):

```
adminPassword: password
```

3. Save and close the file:

```
:wq
```

4. Use helm to install Grafana with *grafana-values.yml*:

```
helm install -f ~/grafana-values.yml ~/charts/stable/grafana --name
grafana --namespace grafana
```

5. We can see which pods are running in this new namespace with the command:

```
kubectl get pods -n grafana
```

Deploy a NodePort Service to Provide External Access to Grafana

1. Make a file called *grafana-ext.yml*:

```
vi grafana-ext.yml
```

2. Paste in this content:

```
kind: Service
apiVersion: v1
metadata:
  namespace: grafana
  name: grafana-ext
spec:
  type: NodePort
  selector:
    app: grafana
  ports:
  - protocol: TCP
    port: 3000
    nodePort: 8081
```

3. Save and close the file:

```
:wq
```

4. Deploy the service:

```
kubectl apply -f ~/grafana-ext.yml
```

5. Log in to Grafana using the **Kubernetes Node Public IP** provided on the hands-on lab page:

```
&lt;KUBERNETES_NODE_PUBLIC_IP&gt;:8081
```

6. Log in using the following credentials that were set earlier:
   - **Username**: admin
   - **Password**: password

Create the Monitoring Dashboards

Add a Datasource for Prometheus

1. Click on **Add data source**
   - **Name**: Kubernetes
   - **Type**: `Prometheus`
   - **URL**: `http://prometheus-server.prometheus.svc.cluster.local`
2. Click **Save & Test**

Add the `Kubernetes All Nodes` Community Dashboard

1. Hover your mouse over the **+** in the left sidebar and click on **Import**.
2. In the *Grafana.com Dashboard* field, provide the ID `3131`. Click outside of the field to load information about the dashboard.
3. In the *Options* section:
   - **prometheus**: Kubernetes
4. Click **Import**.
5. Hover your mouse over the **+** in the left sidebar and then click **Dashboard**. Select the **Graph** panel.
6. Hover over the *Panel Title* and click **Edit**.
7. In the **General** tab at the bottom of the screen, set the *Title* to **Requests Per Minute**.
8. In the **Metrics** tab, paste in the following query:

   ```
   sum(rate(http_request_duration_ms_count[2m])) by (service, route,
   method, code)  * 60
   ```

9. With that in place, let's load our `train-schedule` app to give our graph some data:

   ```
   &lt;KUBERNETES_NODE_PUBLIC_IP&gt;:8080
   ```

10. Refresh the page a few times.

11. Now, navigate back to the Grafana dashboard tab in your browser. In the top-right of the page, click **Last 6 hours** and change the time selection to **Last 5 minutes**.

12. Click **Back to dashboard** in the top-right of the page. Next, click **Save dashboard**, also in the top-right of the page.
13. Name the dashboard "Train Schedule Performance" and click **Save**.

Conclusion

Congratulations, you've completed this hands-on lab!

Tools

Lab Diagram Instant Terminal

Credentials

How do I connect?

Cloud Server Kubernetes Node

Username

cloud_user

Password

c[&d*4Wl

Kubernetes Node Public IP

54.211.83.188

Kubernetes Node Private IP

10.0.1.102

Launch Instant Terminal

How do I connect?

Additional Resources

Your team is building the train schedule app. They currently have it running on a Kubernetes cluster, but they need to monitor the performance of the cluster and the applications running on it. You have been tasked with installing and setting up Prometheus to aggregate data and Grafana to display this data. Both can be installed on the Kubernetes cluster itself. To make sure everything is working, you will need to create two dashboards in Grafana:

1. Import the `Kubernetes All Nodes` community dashboard to display basic metrics about the Kubernetes cluster.
2. Create a new Dashboard and add a graph showing requests per minute for the train-schedule app.

To accomplish this, you will need to:

- Initialize helm with: `helm init --stable-repo-url=https://charts.helm.sh/stable --wait --tiller-image ghcr.io/helm/tiller:v2.8.2`
- Clone the Kubernetes standard charts git repo and checkout a specific commit:

```
git clone https://github.com/kubernetes/charts
cd charts
git checkout efdcffe0b6973111ec6e5e83136ea74cdbe6527d
cd ../
```

- Create a *prometheus-values.yml* for prometheus to turn off persistent storage:

```
alertmanager:
  persistentVolume:
    enabled: false
server:
  persistentVolume:
    enabled: false
```

- Use helm to install prometheus in the `prometheus` namespace:

```
helm install -f ~/prometheus-values.yml ~/charts/stable/prometheus --name prometheus --namespace prometheus
```

- Create a *grafana-values.yml* for grafana to set an admin password:

```
adminPassword: password
```

- Use helm to install grafana in the `grafana` namespace:

```
helm install -f ~/grafana-values.yml ~/charts/stable/grafana --name grafana --namespace grafana
```

- Deploy a NodePort service to provide external access to grafana. Make a file called *grafana-ext.yml*:

```
kind: Service
apiVersion: v1
metadata:
  namespace: grafana
  name: grafana-ext
spec:
  type: NodePort
  selector:
    app: grafana
  ports:
  - protocol: TCP
    port: 3000
    nodePort: 8081
```

And deploy the service:

```
kubectl apply -f ~/grafana-ext.yml
```

- Log in to grafana at `<Kubernetes Node Public IP>:8081`.
- Add a datasource for prometheus. The type should be set to `Prometheus` and the url is `http://prometheus-server.prometheus.svc.cluster.local`.
- Add the `Kubernetes All Nodes` community dashboard with id `3131`.
- Create a new dashboard and add a requests per minute graph for the train-schedule app. You can use the following query:

```
sum(rate(http_request_duration_ms_count[2m])) by (service, route,
method, code)  * 60
```

Learning Objectives

0 of 2 completed

- [ ] **Install Prometheus in the Kubernetes cluster.**
- [ ] **Install Grafana in the Kubernetes cluster.**