Using hostPath Volume Type to Inject Configuration Files

In this lesson, we will get familiar with Prometheus and configure it with hostPath Volume.

WE'LL COVER THE FOLLOWING

Using Prometheus

We are about to deploy Prometheus for the first time. We won't go into details behind the application except to say that it's fantastic and that you should consider it for your monitoring and alerting needs. We're using it only to demonstrate a few Kubernetes concepts. We're not trying to learn how to operate it.

Looking into the Definition

Let's take a look at the application's definition.

```
cat volume/prometheus.yml
```

The **output** is as follows.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: prometheus
  annotations:
    kubernetes.io/ingress.class: "nginx"
```

```
    ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
spec:
  rules:
  - http:
      paths:
      - path: /prometheus
        backend:
          serviceName: prometheus
          servicePort: 9090

---

apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
spec:
  selector:
    matchLabels:
      type: monitor
      service: prometheus
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        type: monitor
        service: prometheus
    spec:
      containers:
      - name: prometheus
        image: prom/prometheus:v2.0.0
        command:
        - /bin/prometheus
        args:
        - "--config.file=/etc/prometheus/prometheus.yml"
        - "--storage.tsdb.path=/prometheus"
        - "--web.console.libraries=/usr/share"
        - "--web.external-url=http://192.168.99.100/prometheus"

---

apiVersion: v1
kind: Service
metadata:
  name: prometheus
spec:
  ports:
  - port: 9090
  selector:
    type: monitor
    service: prometheus
```

There’s nothing genuinely new in that YAML file. It defines an Ingress, a Deployment, and a Service. There is, however, one thing we might need to change.

Configuring the IP

Configuring the IP

Prometheus needs a full `external-url` if we want to change the base path. At the moment, it's set to the IP of our Minikube VM. In your case, that IP might be different. We'll fix that by adding a bit of `sed` "magic" that will make sure the IP matches that of your Minikube VM.

```
cat volume/prometheus.yml | sed -e \
    "s/192.168.99.100/$(minikube ip)/g" \
    | kubectl create -f - \
    --record --save-config

kubectl rollout status deploy prometheus
```

We output the contents of the `volume/prometheus.yml` file, we used `sed` to replace the hard-coded IP with the actual value of your Minikube instance, and we passed the result to `kubectl create`.

> 📝 Please note that, this time, the `create` command has dash (`-`) instead of the path to the file. That's an indication that `stdin` should be used instead.

Once we created the application, we used the `kubectl rollout status` command to confirm that the deployment finished.

Testing Prometheus

Now we can open Prometheus in a browser.

```
open "http://$(minikube ip)/prometheus"
```

At first glance, the application seems to be running correctly. However, since the targets are the crucial part of the application, we should check them as well. For those not familiar with Prometheus, it pulls data from targets (external data sources) and, by default, comes with only one target pre-configured: Prometheus itself. Prometheus will always pull data from this target unless we configure it otherwise.

Let's take a look at its targets.

```
open "http://$(minikube ip)/prometheus/targets"
```

There's something wrong. The default target is not reachable. Before we start panicking, we should take a closer look at its configuration.

```
open "http://$(minikube ip)/prometheus/config"
```

The problem is with the `metrics_path` field. By default, it is set to `/metrics`. However, since we changed the base path to `/prometheus`, the field should have `/prometheus/metrics` as the value.

Changing Prometheus Configuration

Long story short, we must change the Prometheus configuration.

We could, for example, enter the container, update the configuration file, and send the reload request to Prometheus. That would be a terrible solution since it would last only until the next time we update the application, or until the container fails, and Kubernetes decides to reschedule it.

Let's explore alternative solutions. We could, for example, use `hostPath` Volume for this as well. If we can guarantee that the correct configuration file is inside the VM, the Pod could attach it to the `prometheus` container. Let's try it out.

```
cat volume/prometheus-host-path.yml
```

The **output**, limited to relevant parts, is as follows.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
spec:
  selector:
    ...
    spec:
      containers:
        ...
        volumeMounts:
        - mountPath: /etc/prometheus/prometheus.yml
          name: prom-conf
      volumes:
      - name: prom-conf
        hostPath:
```

```
        hostPath:
          path: /files/prometheus-conf.yml
          type: File
...
```

The only significant difference, when compared with the previous definition, is in the added `volumeMounts` and `volumes` fields. We're using the same schema as before, except that, this time, the `type` is set to `File`. Once we `apply` this Deployment, the file `/files/prometheus-conf.yml` on the host will be available as `/etc/prometheus/prometheus.yml` inside the container.

If you recall, we copied one file to the `~/.minikube/files` directory, and Minikube copied it to the `/files` directory inside the VM.

> In some cases, files might end up being copied to the VM's root (`/`), instead of to `/files`. If this has happened to you, please enter the VM (`minikube ssh`), and move the files to `/files`, by executing the commands that follow (**only if the `/files` directory does not exist or is empty**).

```
minikube ssh
```

```
sudo mkdir /files
```

```
sudo mv /prometheus-conf.yml  /files/
```

```
exit
```

> 📝 Run the above commands one by one.

The time has come to take a look at the content of the file.

```
minikube ssh sudo chmod +rw \
    /files/prometheus-conf.yml

minikube ssh cat \
    /files/prometheus-conf.yml
```

We changed the permissions of the file and displayed its content.

The **output** is as follows.

```
global:
  scrape_interval:     15s

scrape_configs:
  - job_name: prometheus
    metrics_path: /prometheus/metrics
    static_configs:
      - targets:
        - localhost:9090
```

This configuration is almost identical to what Prometheus uses by default. The only difference is in the `metrics_path`, which is now pointing to `/prometheus/metrics`.

In the next lesson, we will work with the updated configuration of the Prometheus application.