Linux Academy

Home \ Blog \ Kubernetes \ Running Prometheus on Kubernetes

Running Prometheus on Kubernetes

Posted on **February 28, 2019** by TravisThomsen

Kubernetes

Would you like to know how to get **Prometheus running on Kubernetes**? For the most part, it's pretty easy. We'll be using a series of **YAML** files to deploy everything out.

Before we jump in to the technical stuff, there are a few assumptions that are being made. The first one is that you have a Kubernetes stack already in place. This post will not cover setting it up. The second is that you have ports `9090-9094` configured for your

Kubernetes cluster. If you do not, you may need to change the targetPort for the Prometheus service.

Create the monitoring namespace

We are going to start off by creating the monitoring namespace. Using the editor of your choice, create `namespace.yml` and add the contents below:

```
{
  "kind": "Namespace",
  "apiVersion": "v1",
  "metadata": {
    "name": "monitoring",
    "labels": {
      "name": "monitoring"
    }
  }
}
```

Namespaces act as virtual clusters in Kubernetes. We want to make sure that we run all of the Prometheus pods and services in the monitoring namespace. When you go to list anything you deploy out, you will need to use the `-n` flag and define `monitoring` as the namespace.

For example, if you want to list the Prometheus pods, you will need to do the following:

```
kubectl get pods -n monitoring
```

Apply the namespace

Now apply the namespace by executing the `kubectl apply` command:

```
kubectl apply -f namespace.yml
```

Next we will set up `clusterRole.yml` . This will be used to set up the cluster's roles. We need to set this up so that Prometheus has the correct permissions to the Kubernetes API. Create the `clusterRole.yml` file and add the following contents to it:

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: default
  namespace: monitoring
```

Apply cluster roles

Apply the cluster roles to the Kubernetes cluster:

```
kubectl apply -f clusterRole.yml
```

We are going to use a ConfigMap to decouple any configuration artifacts from image content. This will help keep containerized applications more portable. We will be using this to manage the `prometheus.yml` configuration file.
Create `config-map.yml` and add the following:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-server-conf
  labels:
    name: prometheus-server-conf
  namespace: monitoring
data:
  prometheus.yml: |-
    global:
      scrape_interval: 5s
      evaluation_interval: 5s

    scrape_configs:
      - job_name: 'kubernetes-apiservers'
        kubernetes_sd_configs:
        - role: endpoints
        scheme: https
        tls_config:
          ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
        relabel_configs:
        - source_labels: [__meta_kubernetes_namespace, __meta_kubernetes_servi
          action: keep
          regex: default;kubernetes;https

      - job_name: 'kubernetes-nodes'
        scheme: https
        tls_config:
          ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
        kubernetes_sd_configs:
        - role: node
        relabel_configs:
        - action: labelmap
          regex: __meta_kubernetes_node_label_(.+)
        - target_label: __address__
          replacement: kubernetes.default.svc:443
        - source_labels: [__meta_kubernetes_node_name]
          regex: (.+)
          target_label: __metrics_path__
          replacement: /api/v1/nodes/${1}/proxy/metrics

      - job_name: 'kubernetes-pods'
        kubernetes_sd_configs:
        - role: pod
        relabel_configs:
        - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrap
          action: keep
          regex: true
        - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
          action: replace
          target_label: __metrics_path__
          regex: (.+)
        - source_labels: [__address__, __meta_kubernetes_pod_annotation_promet
          action: replace
          regex: ([^:]+)(?::\d+)?;(\d+)
          replacement: $1:$2
          target_label: __address__
        - action: labelmap
          regex: __meta_kubernetes_pod_label_(.+)
        - source_labels: [__meta_kubernetes_namespace]
          action: replace
```

```
        target_label: kubernetes_namespace
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: kubernetes_pod_name

    - job_name: 'kubernetes-cadvisor'
      scheme: https
      tls_config:
        ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
      bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
      kubernetes_sd_configs:
      - role: node
      relabel_configs:
      - action: labelmap
        regex: __meta_kubernetes_node_label_(.+)
      - target_label: __address__
        replacement: kubernetes.default.svc:443
      - source_labels: [__meta_kubernetes_node_name]
        regex: (.+)
        target_label: __metrics_path__
        replacement: /api/v1/nodes/${1}/proxy/metrics/cadvisor

    - job_name: 'kubernetes-service-endpoints'
      kubernetes_sd_configs:
      - role: endpoints
      relabel_configs:
      - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_s
        action: keep
        regex: true
      - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_s
        action: replace
        target_label: __scheme__
        regex: (https?)
      - source_labels: [__meta_kubernetes_service_annotation_prometheus_io_p
        action: replace
        target_label: __metrics_path__
```

```
          regex: (.+)
        - source_labels: [__address__, __meta_kubernetes_service_annotation_pr
          action: replace
          target_label: __address__
          regex: ([^:]+)(?::\d+)?;(\d+)
          replacement: $1:$2
        - action: labelmap
          regex: __meta_kubernetes_service_label_(.+)
        - source_labels: [__meta_kubernetes_namespace]
          action: replace
          target_label: kubernetes_namespace
        - source_labels: [__meta_kubernetes_service_name]
          action: replace
          target_label: kubernetes_name
```

This file has a lot going on in it. In a nutshell, we are creating the Prometheus targets using service discovery with the Kubernetes API. This is the reason why we needed to configure the cluster roles earlier. Without it, Prometheus wouldn't have the necessary permissions to access the APIs to discover the targets.

The following jobs are being configured as targets using service discovery.

- `kubernetes-apiservers` : Gets metrics on the Kubernetes APIs.
- `kubernetes-nodes` : Gets metrics on the Kubernetes nodes.
- `kubernetes-pods` : Gets metrics from Pods that have the prometheus.io/scrape and prometheus.io/port annotations defined in the metadata.
- `kubernetes-cadvisor` : Gets cAdvisor metrics reported from the Kubernetes cluster.
- `kubernetes-service-endpoints` : Gets metrics from Services that have the `prometheus.io/scrape` and `prometheus.io/port` annotations defined in the metadata.

By using service discovery, we don't need to update the `prometheus.conf` file with new pods and services as they come online and offline. As long as the `prometheus.io/scrape` and `prometheus.io/port` annotations are defined in the metadata of your pods and services, Prometheus will automatically be updated with the targets.

Apply the ConfigMap

Now apply the ConfigMap:

```
kubectl apply -f config-map.yml
```

Now that the ConfigMap is in place, we can create the Prometheus Deployment and Service. Create `prometheus-deployment.yml` and add the following contents to it:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: prometheus-deployment
  namespace: monitoring
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: prometheus-server
    spec:
      containers:
        - name: prometheus
          image: prom/prometheus:v2.2.1
          args:
            - "--config.file=/etc/prometheus/prometheus.yml"
            - "--storage.tsdb.path=/prometheus/"
          ports:
            - containerPort: 9090
          volumeMounts:
            - name: prometheus-config-volume
              mountPath: /etc/prometheus/
            - name: prometheus-storage-volume
              mountPath: /prometheus/
      volumes:
        - name: prometheus-config-volume
          configMap:
            defaultMode: 420
            name: prometheus-server-conf

        - name: prometheus-storage-volume
          emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: prometheus-service
  namespace: monitoring
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '9090'

spec:
  selector:
    app: prometheus-server
  type: NodePort
  ports:
    - port: 8080
      targetPort: 9090
      nodePort: 30000
```

There are a few things I want to point out. Two volume mounts are being created. These are `prometheus-config-volume` and `prometheus-storage-volume` .

```
...
volumeMounts:
            - name: prometheus-config-volume
              mountPath: /etc/prometheus/
            - name: prometheus-storage-volume
              mountPath: /prometheus/
...
```

`prometheus-config-volume` will be using our ConfigMap to manage `prometheus.yml`, which is reflected in the volumes section.

```
...
- name: prometheus-config-volume
         configMap:
           defaultMode: 420
           name: prometheus-server-conf
...
```

This is how we are able to use the `prometheus-server-conf` ConfigMap with the Prometheus deployment. For `prometheus-storage-volume`, we are creating an emptyDir to store the Prometheus data.

```
...
- name: prometheus-storage-volume
          emptyDir: {}
...
```

This volume is ephemeral and is created and destroyed with the Pod. This means if you delete the Pod for any reason, the data in the `prometheus-storage-volume` is deleted with it. If you want this data to be persistent, then you will need to use a persistent volume instead.

Now lets take a look at the metadata defined in the service.

```
metadata:
  name: prometheus-service
  namespace: monitoring
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '9090'
```

Here we are setting up the annotation so that this service will be discovered by Prometheus as a target to be scraped. To make the service available, set `prometheus.io/scrape` to `true`. Then, you need to make sure that `prometheus.io/port` is the targetPort defined in the service. If you don't, the target will not be discovered.

```
ports:
    - port: 8080
      targetPort: 9090
      nodePort: 30000
```

Because the targetPort is set to `9090` we will use that port with `prometheus.io/port`.

```
annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '9090'
```

Create the deployment and service by executing `kubectl apply`.

```
kubectl apply -f prometheus-deployment.yml
```

Let's verify that the pod and service were created.

```
kubectl get pods -n monitoring
kubectl get services -n monitoring
```

Once the pod and service are available, you can access Prometheus's Expression Browser by going to `http://<KUBERNETES_MASTER_IP>:9090`.

You can now monitor your pods and services that you deploy to your cluster.

More about Running Prometheus on Kubernetes

If you want to learn more about running Prometheus on Kubernetes, go check out my latest hands-on course **Monitoring Kubernetes With Prometheus**.

Happy monitoring!

Other Kubernetes Resources:

- **Get Started with Kubernetes**
- **Kubernetes Quick Start**
- **Kubernetes Essentials**