Install Service Monitors

Longhorn Servicemonitor

Our storage provisioner Longhorn, that we deployed somewhere near the start of this whole K3s Kubernetes cluster setup, also natively provides data for Prometheus.

Create a new folder, `monitoring`, that we will put most of our configs in, and create the file `longhorn-servicemonitor.yaml`.

```
cd
mkdir monitoring
cd monitoring
touch longhorn-servicemonitor.yaml
```

Edit the `longhorn-servicemonitor.yaml`:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: longhorn-prometheus-servicemonitor
  namespace: monitoring
  labels:
    name: longhorn-prometheus-servicemonitor
spec:
  selector:
    matchLabels:
      app: longhorn-manager
  namespaceSelector:
    matchNames:
    - longhorn-system
  endpoints:
  - port: manager
```

As you can see, we are not talking to Kubernetes API (we are.. but..), but to `apiVersion: monitoring.coreos.com/v1`, so we are basically telling Prometheus Operator to create something for us. In this case it's `kind: ServiceMonitor`.

Makes sense, right? Next, `metadata`, and here I'm not 100% sure about the `name:` and below `labels: -> name:`. I know that we refer to these later, or one of them, when we tell Prometheus which Service Monitor to collect data from.

This should be clear, `metadata: -> namespace: monitoring`, we are telling it to deploy into our monitoring namespace.

The rest under `spec:` is basically telling what app the Service Monitor should "bind to". It's looking for `app: longhorn-manager` in namespace `longhorn-system` and `port: manager`. This port could be a port number, but it also can have a name, so in this case it's named `manager`.

This is the `longhorn-manager` we are targeting.

```
root@control01:/home/ubuntu/monitoring# kubectl get daemonset -n longhorn-system
NAME                 DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
.
.
longhorn-manager     9         9         9       9            9           <none>          35d
```

And if you try to describe it with:

```
kubectl describe daemonset longhorn-manager -n longhorn-system

.
.
.
Port:       9500/TCP
.
.
.
```

You get that the port it is using is 9500/TCP, but I don't know where it's set that `manager == 9500`. If you know, please comment below.

Node-exporter

This is the daemon set we will deploy to collect metrics from individual cluster nodes, underlying HW, etc...

In the monitoring folder, create a new folder called 'node-exporter' and create the following files in it:

> **Info**
>
> Sorry guys, I always try to provide a source for the code, but for the life of me I can't find where I got this deployment from. If you have seen it before somewhere, let me know down in the comments, and I will update the source here.

cluster-role-binding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: node-exporter
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: node-exporter
subjects:
- kind: ServiceAccount
  name: node-exporter
  namespace: monitoring
```

cluster-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: node-exporter
rules:
- apiGroups:
  - authentication.k8s.io
  resources:
  - tokenreviews
  verbs:
  - create
- apiGroups:
  - authorization.k8s.io
  resources:
  - subjectaccessreviews
  verbs:
  - create
```

service-account.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: node-exporter
  namespace: monitoring
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/name: node-exporter
  name: node-exporter
  namespace: monitoring
spec:
  clusterIP: None
  ports:
  - name: https
    port: 9100
    targetPort: https
  selector:
    app.kubernetes.io/name: node-exporter
```

daemonset.yaml

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: node-exporter
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 1.1.0
    name: node-exporter
  name: node-exporter
  namespace: monitoring
spec:
  selector:
    matchLabels:
      app.kubernetes.io/component: exporter
      app.kubernetes.io/name: node-exporter
      app.kubernetes.io/part-of: kube-prometheus
  template:
    metadata:
      labels:
        app.kubernetes.io/component: exporter
        app.kubernetes.io/name: node-exporter
        app.kubernetes.io/part-of: kube-prometheus
        app.kubernetes.io/version: 1.1.0
    spec:
      containers:
```

```
      - args:
        - --web.listen-address=127.0.0.1:9100
        - --path.sysfs=/host/sys
        - --path.rootfs=/host/root
        - --no-collector.wifi
        - --no-collector.hwmon
        - --collector.filesystem.ignored-mount-points=^/(dev|proc|sys|var/lib/docker/.+|var/lib/kubelet/pods/.+)($|/)
        - --collector.netclass.ignored-devices=^(veth.*)$
        - --collector.netdev.device-exclude=^(veth.*)$
        image: quay.io/prometheus/node-exporter:v1.1.0
        name: node-exporter
        resources:
          limits:
            cpu: 250m
            memory: 180Mi
          requests:
            cpu: 102m
            memory: 180Mi
        volumeMounts:
        - mountPath: /host/sys
          mountPropagation: HostToContainer
          name: sys
          readOnly: true
        - mountPath: /host/root
          mountPropagation: HostToContainer
          name: root
          readOnly: true
      - args:
        - --logtostderr
        - --secure-listen-address=[$(IP)]:9100
        - --tls-cipher-
suites=TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECD

        - --upstream=http://127.0.0.1:9100/
        env:
        - name: IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
        image: quay.io/brancz/kube-rbac-proxy:v0.8.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 9100
          hostPort: 9100
          name: https
        resources:
          limits:
            cpu: 20m
            memory: 40Mi
          requests:
            cpu: 10m
            memory: 20Mi
        securityContext:
          runAsGroup: 65532
          runAsNonRoot: true
          runAsUser: 65532
      hostNetwork: true
      hostPID: true
      nodeSelector:
        kubernetes.io/os: linux
      securityContext:
        runAsNonRoot: true
        runAsUser: 65534
      serviceAccountName: node-exporter
      tolerations:
      - operator: Exists
      volumes:
      - hostPath:
          path: /sys
        name: sys
      - hostPath:
          path: /
        name: root
  updateStrategy:
    rollingUpdate:
      maxUnavailable: 10%
    type: RollingUpdate
```

service-monitor.yaml

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: node-exporter
    name: node-exporter
  name: node-exporter
  namespace: monitoring
spec:
  endpoints:
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    interval: 15s
    port: https
    relabelings:
    - action: replace
      regex: (.*)
      replacement: $1
      sourceLabels:
      - __meta_kubernetes_pod_node_name
      targetLabel: instance
    scheme: https
    tlsConfig:
      insecureSkipVerify: true
  jobLabel: app.kubernetes.io/name
  selector:
    matchLabels:
      app.kubernetes.io/name: node-exporter
```

You can deploy all the yaml files by going in folder above and apply -f on the folder.

```
cd ..
kubectl apply -f node-exporter/
```

This will create all permissions, and deploy the pod with the application Node Exporter, that will read metrics from Linux.

After doing so, you should see node-exporter-xxxx pods in the `monitoring` namespace; I have 9 nodes so it's there 9 times.

```
root@control01:/home/ubuntu/monitoring/node-exporter# kubectl get pods -n monitoring
NAME                                  READY   STATUS    RESTARTS   AGE
.
.
node-exporter-8stdd                   2/2     Running   0          15d
node-exporter-cfgmh                   2/2     Running   0          15d
node-exporter-gdr7f                   2/2     Running   0          15d
node-exporter-kkd68                   2/2     Running   0          15d
node-exporter-mmkk5                   2/2     Running   0          15d
node-exporter-mp6g5                   2/2     Running   0          15d
node-exporter-pfmcm                   2/2     Running   2          15d
node-exporter-rdlwn                   2/2     Running   0          15d
node-exporter-rvz6l                   2/2     Running   0          15d
.
.
```

Kube State Metrics

This is a simple service that listens to the Kubernetes API, and generates metrics about the state of the objects.

Link to official github: kube-state-metrics .

Again as before, create a new folder in our monitoring folder, called `kube-state-metrics`. Create the following files in it:

kube-state-metrics-clusterRole.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
rules:
- apiGroups:
  - ""
  resources:
  - configmaps
  - secrets
  - nodes
  - pods
  - services
  - resourcequotas
  - replicationcontrollers
  - limitranges
  - persistentvolumeclaims
  - persistentvolumes
  - namespaces
  - endpoints
  verbs:
  - list
  - watch
- apiGroups:
  - extensions
  resources:
  - daemonsets
  - deployments
  - replicasets
  - ingresses
  verbs:
  - list
  - watch
- apiGroups:
  - apps
```

```
  resources:
  - statefulsets
  - daemonsets
  - deployments
  - replicasets
  verbs:
  - list
  - watch
- apiGroups:
  - batch
  resources:
  - cronjobs
  - jobs
  verbs:
  - list
  - watch
- apiGroups:
  - autoscaling
  resources:
  - horizontalpodautoscalers
  verbs:
  - list
  - watch
- apiGroups:
  - authentication.k8s.io
  resources:
  - tokenreviews
  verbs:
  - create
- apiGroups:
  - authorization.k8s.io
  resources:
  - subjectaccessreviews
  verbs:
  - create
- apiGroups:
  - policy
  resources:
  - poddisruptionbudgets
  verbs:
  - list
  - watch
- apiGroups:
  - certificates.k8s.io
  resources:
  - certificatesigningrequests
  verbs:
  - list
  - watch
- apiGroups:
  - storage.k8s.io
  resources:
  - storageclasses
  - volumeattachments
  verbs:
  - list
  - watch
- apiGroups:
  - admissionregistration.k8s.io
  resources:
  - mutatingwebhookconfigurations
  - validatingwebhookconfigurations
  verbs:
  - list
  - watch
- apiGroups:
  - networking.k8s.io
  resources:
  - networkpolicies
  verbs:
  - list
  - watch
```

kube-state-metrics-clusterRoleBinding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kube-state-metrics
subjects:
- kind: ServiceAccount
  name: kube-state-metrics
  namespace: monitoring
```

kube-state-metrics-serviceAccount.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
  namespace: monitoring
```

kube-state-metrics-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
  namespace: monitoring
spec:
  clusterIP: None
  ports:
  - name: https-main
    port: 8443
    targetPort: https-main
  - name: https-self
    port: 9443
    targetPort: https-self
  selector:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/part-of: kube-prometheus
```

kube-state-metrics-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 1.9.7
  name: kube-state-metrics
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/component: exporter
      app.kubernetes.io/name: kube-state-metrics
      app.kubernetes.io/part-of: kube-prometheus
  template:
    metadata:
      labels:
        app.kubernetes.io/component: exporter
        app.kubernetes.io/name: kube-state-metrics
        app.kubernetes.io/part-of: kube-prometheus
        app.kubernetes.io/version: 1.9.7
    spec:
      containers:
      - args:
        - --host=127.0.0.1
        - --port=8081
        - --telemetry-host=127.0.0.1
        - --telemetry-port=8082
        image: carlosedp/kube-state-metrics:v1.9.6
        name: kube-state-metrics
        resources:
          limits:
            cpu: 100m
            memory: 250Mi
          requests:
            cpu: 10m
            memory: 190Mi
      - args:
        - --logtostderr
        - --secure-listen-address=:8443
        - --tls-cipher-
suites=TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECD

        - --upstream=http://127.0.0.1:8081/
        image: quay.io/brancz/kube-rbac-proxy:v0.8.0
        name: kube-rbac-proxy-main
        ports:
```

```
        - containerPort: 8443
          name: https-main
        resources:
          limits:
            cpu: 20m
            memory: 40Mi
          requests:
            cpu: 10m
            memory: 20Mi
        securityContext:
          runAsGroup: 65532
          runAsNonRoot: true
          runAsUser: 65532
      - args:
        - --logtostderr
        - --secure-listen-address=:9443
        - --tls-cipher-
suites=TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,TLS_ECD

        - --upstream=http://127.0.0.1:8082/
        image: quay.io/brancz/kube-rbac-proxy:v0.8.0
        name: kube-rbac-proxy-self
        ports:
        - containerPort: 9443
          name: https-self
        resources:
          limits:
            cpu: 20m
            memory: 40Mi
          requests:
            cpu: 10m
            memory: 20Mi
        securityContext:
          runAsGroup: 65532
          runAsNonRoot: true
          runAsUser: 65532
      nodeSelector:
        node-type: worker
      serviceAccountName: kube-state-metrics
```

> **Info**
>
> Above you can see I'm using `image: carlosedp/kube-state-metrics:v1.9.6`. This is not the official image. (This is the official Docker Hub Link), but official image is not compiled for arm64, therefore I have changed it to Docker Hub Link. This is the same software, just compiled for more architectures.

kube-state-metrics-serviceMonitor.yaml

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/component: exporter
    app.kubernetes.io/name: kube-state-metrics
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 1.9.7
    name: kube-state-metrics
  name: kube-state-metrics
  namespace: monitoring
spec:
  endpoints:
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    honorLabels: true
    interval: 30s
    port: https-main
    relabelings:
    - action: labeldrop
      regex: (pod|service|endpoint|namespace)
    scheme: https
    scrapeTimeout: 30s
    tlsConfig:
      insecureSkipVerify: true
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    interval: 30s
    port: https-self
    scheme: https
    tlsConfig:
      insecureSkipVerify: true
  jobLabel: app.kubernetes.io/name
  selector:
    matchLabels:
      app.kubernetes.io/component: exporter
      app.kubernetes.io/name: kube-state-metrics
      app.kubernetes.io/part-of: kube-prometheus
```

And again, jump one folder up and apply everything:

```
cd ..
kubectl apply -f kube-state-metrics/
```

Check the pods in the `monitoring` namespace if you have kube-state-metrics-xxx up and running:

```
root@control01:/home/ubuntu/monitoring/kube-state-metrics# kubectl get pods -n monitoring
NAME                                  READY   STATUS    RESTARTS   AGE
.
.
kube-state-metrics-6f8f5578-p99td     3/3     Running   0          14d
.
.
```

We have two more Service Monitors to go. 🙂

Kubelet

Kubelet, in case you did not know, is an essential part of Kubernetes' control plane, and is also something that exposes Prometheus metrics by default in the port 10255. So, it makes sense to create a Service Monitor for it as well.

'But', you surely ask me, 'I just deployed kube-state-metrics, why the fuck do I need another Kubelet thingy monitor?' Well, kube-state-metrics collects lots of data, and some of them overlap with Kubelet provided metrics, but not all; some information can be collected only from Kubelet.

We only need to create one file: `kubelet-servicemonitor.yaml`:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app.kubernetes.io/name: kubelet
    name: kubelet
  name: kubelet
  namespace: monitoring
spec:
  endpoints:
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    honorLabels: true
    interval: 30s
    metricRelabelings:
    - action: drop
      regex:
kubelet_(pod_worker_latency_microseconds|pod_start_latency_microseconds|cgroup_manager_latency_microseconds|pod_worker_start_latency_microseconds|pleg_relist_latency_mic

      sourceLabels:
      - __name__
    - action: drop
      regex:
scheduler_(e2e_scheduling_latency_microseconds|scheduling_algorithm_predicate_evaluation|scheduling_algorithm_priority_evaluation|scheduling_algorithm_preemption_evaluat

      sourceLabels:
      - __name__
    - action: drop
      regex:
apiserver_(request_count|request_latencies|request_latencies_summary|dropped_requests|storage_data_key_generation_latencies_microseconds|storage_transformation_failures_

      sourceLabels:
      - __name__
    - action: drop
      regex: kubelet_docker_(operations|operations_latency_microseconds|operations_errors|operations_timeout)
      sourceLabels:
      - __name__
    - action: drop
      regex: reflector_(items_per_list|items_per_watch|list_duration_seconds|lists_total|short_watches_total|watch_duration_seconds|watches_total)
      sourceLabels:
      - __name__
    - action: drop
      regex:
etcd_(helper_cache_hit_count|helper_cache_miss_count|helper_cache_entry_count|request_cache_get_latencies_summary|request_cache_add_latencies_summary|request_latencies_s

      sourceLabels:
      - __name__
    - action: drop
      regex: transformation_(transformation_latencies_microseconds|failures_total)
      sourceLabels:
      - __name__
    - action: drop
      regex:
(admission_quota_controller_adds|crd_autoregistration_controller_work_duration|APIServiceOpenAPIAggregationControllerQueue1_adds|AvailableConditionController_retries|crd_

      sourceLabels:
      - __name__
    port: https-metrics
    relabelings:
    - sourceLabels:
      - __metrics_path__
      targetLabel: metrics_path
    scheme: https
    tlsConfig:
      insecureSkipVerify: true
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    honorLabels: true
    honorTimestamps: false
    interval: 30s
    metricRelabelings:
    - action: drop
```

```
      regex: container_(network_tcp_usage_total|network_udp_usage_total|tasks_state|cpu_load_average_10s)
      sourceLabels:
      - __name__
    path: /metrics/cadvisor
    port: https-metrics
    relabelings:
    - sourceLabels:
      - __metrics_path__
      targetLabel: metrics_path
    scheme: https
    tlsConfig:
      insecureSkipVerify: true
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    honorLabels: true
    interval: 30s
    path: /metrics/probes
    port: https-metrics
    relabelings:
    - sourceLabels:
      - __metrics_path__
      targetLabel: metrics_path
    scheme: https
    tlsConfig:
      insecureSkipVerify: true
  jobLabel: k8s-app
  namespaceSelector:
    matchNames:
    - kube-system
  selector:
    matchLabels:
      k8s-app: kubelet
```

Apply, and it's done.

```
kubectl apply -f kubelet-servicemonitor.yaml
```

Traefik

I do not use Traefik much in my setup, but it is there, and it also exposes Prometheus-ready data, so why not...

Create file: `traefik-servicemonitor.yaml`:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app: traefik
    release: prometheus
    name: traefik
  name: traefik
  namespace: monitoring
spec:
  endpoints:
  - port: metrics
  namespaceSelector:
    matchNames:
    - kube-system
  selector:
    matchLabels:
      app: traefik
```

And apply:

```
kubectl apply -f traefik-servicemonitor.yaml
```

OpenFaas

Don't worry about this one; if you deployed as described in my guide/notes you already have Prometheus set up and collecting data.

```
root@control01:/home/ubuntu/monitoring# kubectl get pods -n openfaas
NAME                                READY   STATUS    RESTARTS   AGE
alertmanager-7b849dbf96-hkkg6       1/1     Running   0          7d18h
basic-auth-plugin-6dc5dd97f-7vq6j   1/1     Running   0          7d18h
gateway-8d768fb57-4rqnr             2/2     Running   0          35d
nats-7fdd76d465-6677w               1/1     Running   0          35d
prometheus-54c655c4fd-89g9k         1/1     Running   0          35d
queue-worker-64b75867c8-kn7br       1/1     Running   0          35d
```

We will point Grafana to suck data from this instance later as well.

Done for now!

I know, so much text! I could probably print the whole K3s Kubernetes cluster setup as a book, drop it on somebody, and it would flatten them 😀.

We should now have the following Service Monitors up and ready to be scraped by Prometheus.

```
root@control01:/home/ubuntu/monitoring/kube-state-metrics# kubectl get ServiceMonitor -n monitoring
NAME                              AGE
kube-state-metrics                14d
kubelet                           12d
longhorn-prometheus-servicemonitor   15d
node-exporter                     15d
traefik                           12d
```

Phew! we have the hardest and longest part behind us; you deserve a drink, and if you found this useful, help me to get one too. I would appreciate that a lot.

Move on to Prometheus

Last update: October 20, 2021

Comments