Health metrics
Health metrics
Abstract

Read more to learn about metrics and monitoring across your clusters and components.
# Table of Contents

CHAPTER 1. HEALTH METRICS ................................................................. 3
  1.1. USING THE METRICS SERVICE ........................................... 3
    1.1.1. Accessing the hub cluster metrics service ...................... 3
    1.1.2. Scraping with Prometheus ........................................... 3
      1.1.2.1. Scraping the hub cluster .................................... 3
      1.1.2.2. Scraping the managed cluster ............................... 5
    1.1.3. Scraping the standalone cluster .................................. 6
CHAPTER 1. HEALTH METRICS

You can use metrics to monitor the health of your components.

See the following documentation:

- Using the metrics service

1.1. USING THE METRICS SERVICE

You can use metrics to monitor component health across Red Hat Advanced Cluster Management for Kubernetes. Many custom metrics are documented in the Metrics chronicle overview.

1.1.1. Accessing the hub cluster metrics service

To view the collected metrics, you must expose the metrics service on the hub cluster. If your metrics are already exposed in the Grafana dashboard, this procedure is optional.

From the OpenShift Container Platform console, find the metrics service. Click Observe > Metrics.

If you do not see the metrics in the Grafana dashboard, Grafana Explorer, or in the OpenShift Container Platform console, Prometheus might not be configured to scrape metrics. Continue with Scraping with Prometheus to expose your metrics.

1.1.2. Scraping with Prometheus

You can use Prometheus to expose metrics that are not exposed from the product console. See the procedures for both the hub and managed cluster metrics.

1.1.2.1. Scraping the hub cluster

See the following procedure to expose metrics for the hub cluster. These files are within the openshift-monitoring namespace:

1. Create a ServiceMonitor for collecting services and exposing metrics. See the following YAML example:

   ```yaml
   apiVersion: monitoring.coreos.com/v1
   kind: ServiceMonitor
   metadata:
     name: hub-subscription-metrics
     namespace: monitoring
   spec:
     endpoints:
     - port: metrics
       namespaceSelector:
       - open-cluster-management
       selector:
       - app: hub-subscription-metrics
   ```

2. Run the following command to apply the file:
oc apply -f

3. Create a **Role** for setting the permissions for monitoring. See the following YAML file:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: prometheus-k8s-monitoring
  namespace: open-cluster-management
rules:
- apiGroups:
  - "
  resources:
  - services
  - endpoints
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - get
  - list
  - watch
```

4. Run the following command to apply the file:

```
oc apply -f
```

5. Create a **RoleBinding** for binding the role to the Prometheus monitoring **ServiceAccount**, as it is in the following example:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: prometheus-k8s-monitoring-binding
  namespace: open-cluster-management
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: prometheus-k8s-monitoring
subjects:
```
6. Run the following command to apply the file:

    oc apply -f

7. To verify, run the following query in the dashboard to find metrics that are reported by the Subscription Operator Metrics Service:

    {service="hub-subscription-metrics"}

### 1.1.2.2. Scraping the managed cluster

See the following procedure to expose metrics for managed clusters. These files are within the `openshift-monitoring` namespace:

1. Create a **ServiceMonitor** for collecting services exposing metrics. See the following YAML file example:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
    name: mc-subscription-metrics
    namespace: monitoring
spec:
    endpoints:
    - port: metrics
      namespaceSelector:
        matchNames:
        - open-cluster-management-agent-addon
      selector:
        matchLabels:
        app: mc-subscription-metrics
```

2. Run the following command to apply your file:

    oc apply -f

3. Create a **Role** for setting the permissions for monitoring. See the following YAML file example:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
    name: prometheus-k8s-monitoring
    namespace: open-cluster-management-agent-addon
rules:
    - apiGroups:
      - ""
        resources:
        - services
        - endpoints
        - pods
```
4. Run the following command to apply the file:

   `oc apply -f`

5. Create a **RoleBinding** for binding the **Role** to the Prometheus monitoring **ServiceAccount**:

   ```yaml
   apiVersion: rbac.authorization.k8s.io/v1
   kind: RoleBinding
   metadata:
     name: prometheus-k8s-monitoring-binding
     namespace: open-cluster-management-agent-addon
   roleRef:
     apiGroup: rbac.authorization.k8s.io
     kind: Role
     name: prometheus-k8s-monitoring
   subjects:
     - kind: ServiceAccount
       name: prometheus-k8s
       namespace: monitoring
   ```

6. Run the following command to apply the file:

   `oc apply -f`

7. Verify in the **Prometheus** dashboard by running the following query to find metrics that are reported by the Subscription Operator Metrics Service:

   ```sql
   {service="mc-subscription-metrics"}
   ```

### 1.1.3. Scraping the standalone cluster

1. Create a **ServiceMonitor** for collecting services exposing metrics:
2. Create a **Role** for setting the permissions for monitoring:

```
oc apply -f
```

3. Create a **RoleBinding** for binding the **Role** to the Prometheus monitoring **ServiceAccount**.

See the following YAML file example:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: standalone-subscription-metrics
namespace: monitoring
spec:
  endpoints:
  - port: metrics
  namespaceSelector:
    matchNames:
      - open-cluster-management
  selector:
    matchLabels:
      app: standalone-subscription-metrics

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: prometheus-k8s-monitoring
namespace: open-cluster-management
rules:
- apiGroups:
  - ""    
  resources:
  - services
  - endpoints
  - pods
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - networking.k8s.io
  resources:
  - ingresses
  verbs:
  - get
  - list
  - watch
```
4. Run the following command to apply the file:

   
   ```bash
   oc apply -f
   ```

5. Verify in the Prometheus dashboard by running the following query to find metrics that are reported by the Subscription Operator Metrics Service:

   ```sql
   {service="standalone-subscription-metrics")
   ```