

How to monitor applications on OpenShift 4.x with Prometheus Operator

OpenShift Container Platform includes a pre-configured, pre-installed, and self-updating monitoring stack that is based on the Prometheus open source project and its wider eco-system. It provides monitoring of cluster components and includes a set of alerts to immediately notify the cluster administrator about any occurring problems and a set of Grafana dashboards. The cluster monitoring stack is only supported for monitoring OpenShift Container Platform clusters and adding additional monitoring targets is not supported.

In this lab we will configure application monitoring stack on Openshift 4.x using Prometheus Operator for a sample Node.js microservice instrumented with Prometheus client library (instrumentation was covered in Lab-3).

Deploy an instrumented application

Use the following command and provided yaml file, to deploy sample Node.js microservice instrumented with Prometheus client library.

```
oc new-project b2m-nodejs
oc create -f b2m-nodejs.yml
```

In case of [problems with pulling the app image from Docker Hub](#), you can build the application image by yourself using:

```
oc new-app https://github.com/rafal-szypulka/b2m-nodejs-v2 \
--context-dir=lab-4/app --name b2m-nodejs \
--labels='name=b2m-nodejs' --insecure-registry=true
```

The monitor expects the service's port name to be `web`, so edit the service and change `spec.ports.name` to `web`:

```
spec:
  clusterIP: xxx.xxx.xxx.xxx
  ports:
    - name: web
```

Create route to expose this application externally:

```
oc expose svc b2m-nodejs
```

Collect the app URL:

```
$ oc get routes -n default
```

NAME	HOST/PORT	PATH
SERVICES	PORT	TERMINATION WILDCARD
b2m-nodejs	b2m-nodejs-b2m-nodejs.apps.rsocp.os.fyre.ibm.com	
b2m-nodejs	<all> edge	None

and make sure it works:

```
$ curl -k https://b2m-nodejs-b2m-nodejs.apps.rsocp.os.fyre.ibm.com

{"status":"ok","transactionTime":"353ms"}
```

Verify that it properly exposes metrics in Prometheus format:

```
$ curl -k https://b2m-nodejs-b2m-nodejs.apps.rsocp.os.fyre.ibm.com/metrics

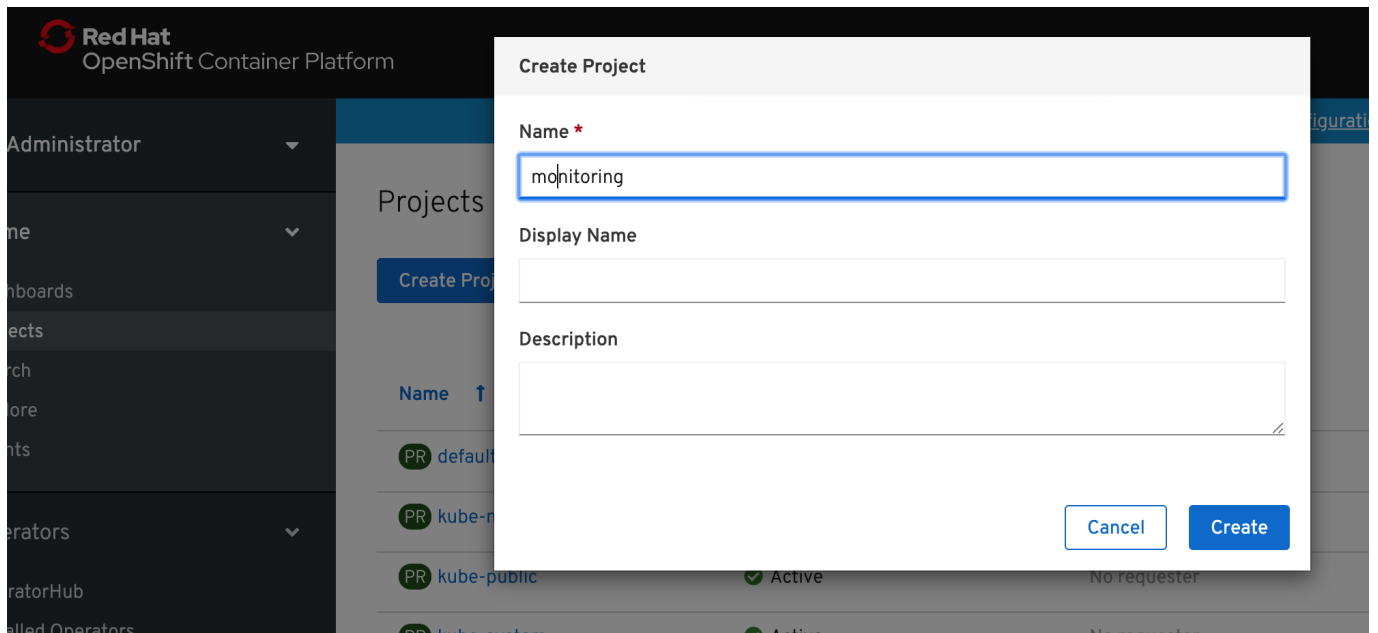
# HELP process_cpu_user_seconds_total Total user CPU time spent in
seconds.
# TYPE process_cpu_user_seconds_total counter
process_cpu_user_seconds_total 0.23436700000000005 1573764470969

# HELP process_cpu_system_seconds_total Total system CPU time spent in
seconds.
# TYPE process_cpu_system_seconds_total counter
process_cpu_system_seconds_total 0.069524 1573764470969

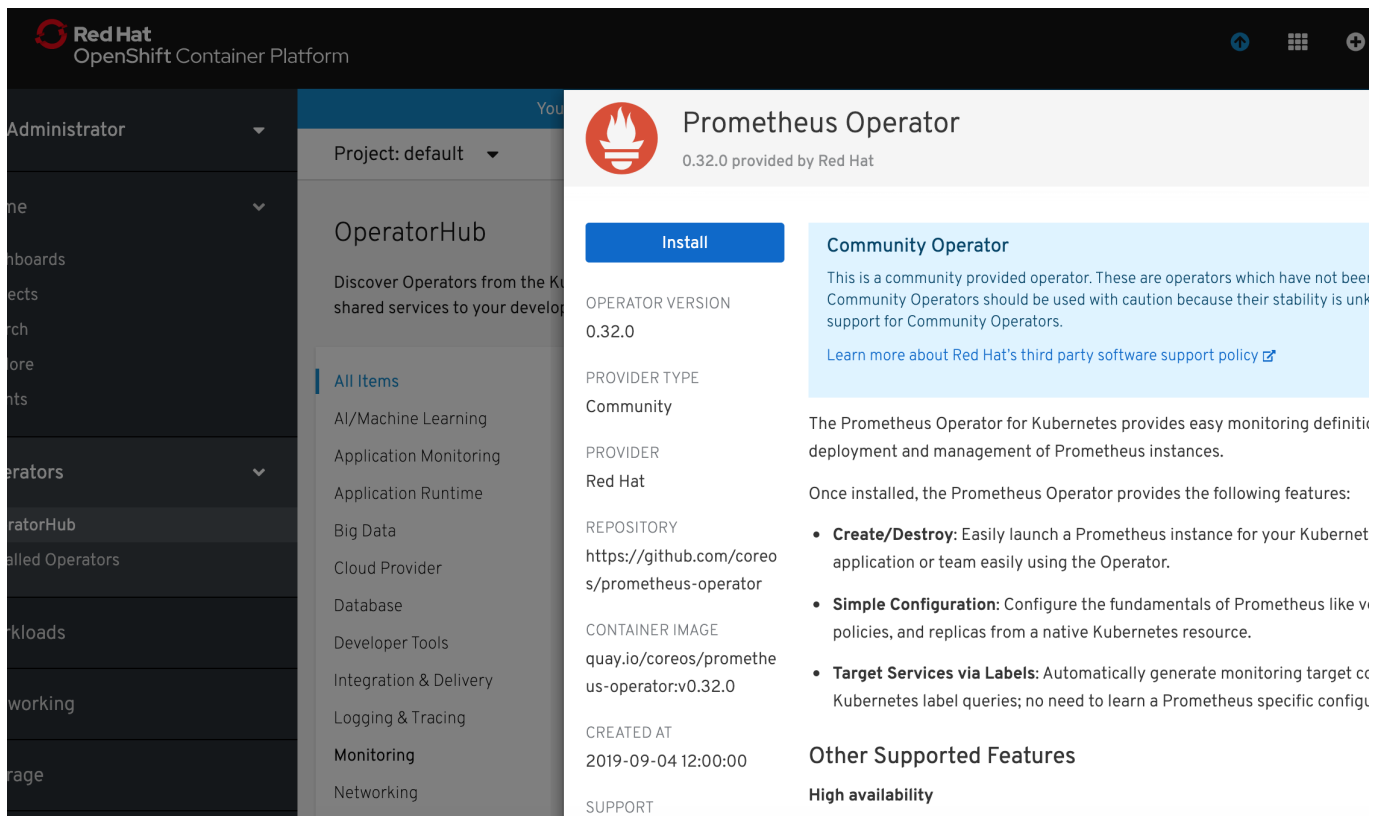
# HELP process_cpu_seconds_total Total user and system CPU time spent in
seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 0.30389100000000002 1573764470969
(...)
```

Deploy Prometheus monitoring stack for applications.

1). Create a new project for the Prometheus monitoring stack for applications.



2). Select Operators -> Operator Hub and select **Prometheus Operator**. Click **Install**.



3). In the **Create Operator Subscription** window click **Subscribe**.

Container Platform

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to

Create Operator Subscription

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines

Installation Mode *

☐ All namespaces on the cluster (default)
This mode is not supported by this Operator

☒ A specific namespace on the cluster
Operator will be available in a single namespace only.

PR

 monitoring


Update Channel *

☒ beta

Approval Strategy *

☒ Automatic

☐ Manual

 Prometheus Operator
provided by Red Hat

Provided APIs

PI

 Prometheus
A running Prometheus instance

PR

 Prometheus Rule
A Prometheus Rule configures sequentially evaluated recording and alerting rules.

SM

 Service Monitor

Subscribe

Cancel

Administrator

Home

Dashboards

Projects

Search

Explore

Events

Operators

OperatorHub

Installed Operators




Workloads

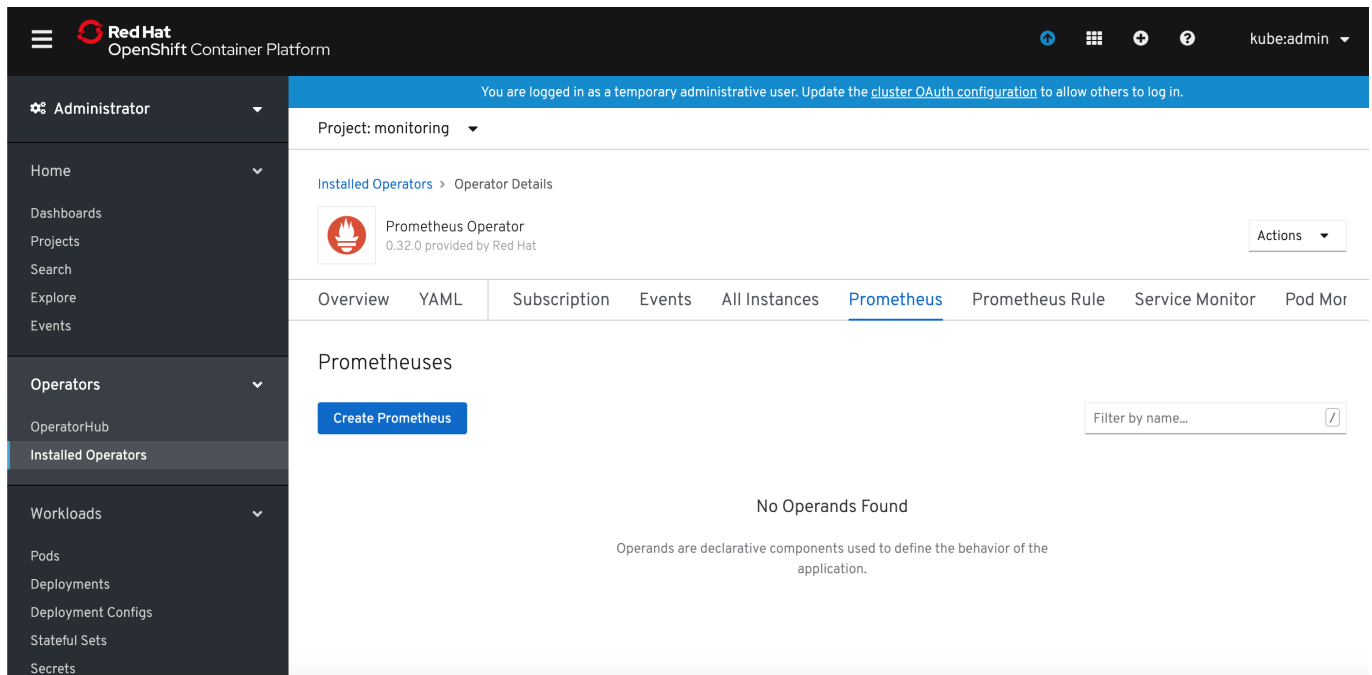
You are logged in as a temporary administrative user. [Update the cluster OAuth configuration](#) to use your own account.

Project: monitoring

Installed Operators

Installed Operators are represented by Cluster Service Versions within this namespace. For more information, see the [Operator Lifecycle Manager documentation](#). Or create an Operator and Cluster Service Version using the [Operator SDK](#).

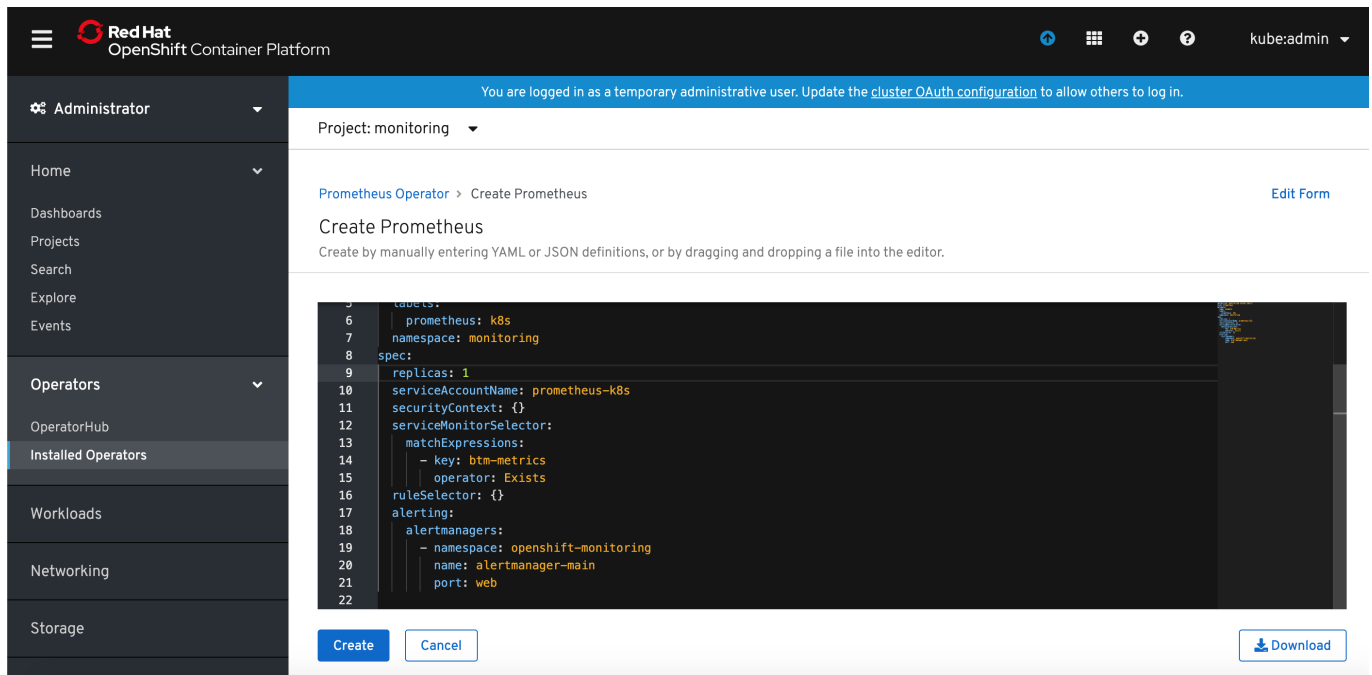
Name	Namespace	Deployment	Status
 Prometheus Operator 0.32.0 provided by Red Hat	 monitoring	 prometheus-operator	Up to date



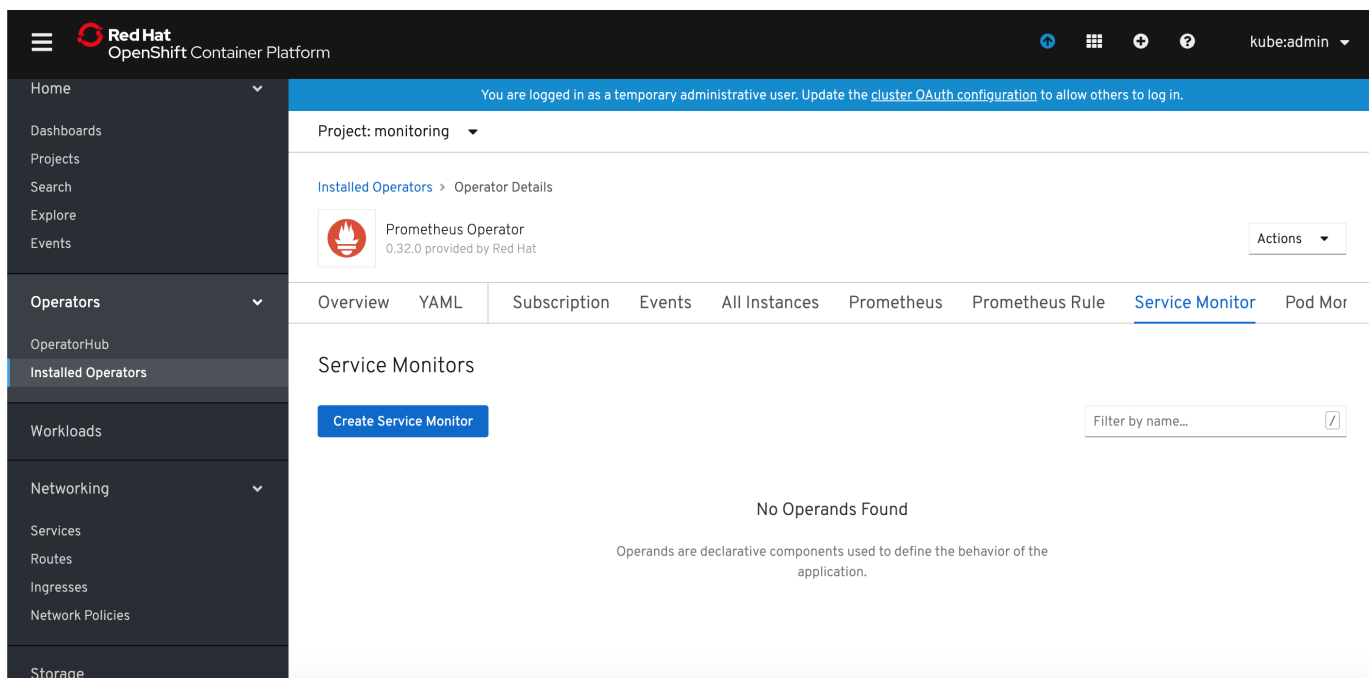
6). Modify default YAML template for Prometheus. I added `serviceMonitorSelector` definition which will instruct defined Prometheus instance to match `ServiceMonitors` with label `key=btm-metrics`. I also changed the Prometheus instance name to `app-monitor`.

```
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: app-monitor
  labels:
    prometheus: k8s
  namespace: monitoring
spec:
  replicas: 1
  serviceAccountName: prometheus-k8s
  securityContext: {}
  serviceMonitorSelector:
    matchExpressions:
      - key: btm-metrics
        operator: Exists
  ruleSelector:
    matchLabels:
      prometheus: app-monitor
      role: alert-rules
  alerting:
    alertmanagers: {}
```

Click `Create` button.



7). Select **Service Monitor** tab and click **Create Service Monitor**.



8). Modify default YAML template for ServiceMonitor. I added **namespaceSelector** definition to limit the scope to namespace **default** where my app has been deployed and modified **selector** that to look for services with label **name=b2m-nodejs**. I also changed the Service monitor name to **app-monitor**.

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    btm-metrics: b2m-nodejs
    name: app-monitor
    namespace: monitoring
spec:
  endpoints:
```

```

- interval: 30s
  port: web
namespaceSelector:
  matchNames:
    - b2m-nodejs
selector:
  matchLabels:
    name: b2m-nodejs

```

The screenshot shows the Red Hat OpenShift Container Platform console. The left sidebar contains navigation links for Dashboards, Projects, Search, Explore, Events, Operators, OperatorHub, Installed Operators, Workloads, Pods, Deployments, Deployment Configs, Stateful Sets, Secrets, Config Maps, Cron Jobs, and Jobs. The main content area shows the 'ServiceMonitor Details' for 'app-monitor' in the 'monitoring' namespace. The 'YAML' tab is selected, displaying the following configuration:

```

9 namespace: monitoring
10 resourceVersion: '172764'
11 selfLink: >-
12 /apis/monitoring.coreos.com/v1/namespaces/monitoring/servicemonitors/app-monitor
13 uid: 2f181761-0607-11ea-b6b0-00000a100c64
14 spec:
15   endpoints:
16     - interval: 30s
17       port: web
18   namespaceSelector:
19     matchNames:
20       - default
21   selector:
22     matchLabels:
23       name1: b2m-nodejs
24

```

At the bottom of the console, there are buttons for 'Save', 'Reload', 'Cancel', and 'Download'.

9). Grant **view** cluster role to the Service Account created by the operator and used by Prometheus.

```

oc adm policy add-cluster-role-to-user view
system:serviceaccount:monitoring:prometheus-k8s

```

or, if you want to limit it to the application namespace, add **view** role only to the app namespace:

```

oc adm policy add-role-to-user view
system:serviceaccount:monitoring:prometheus-k8s -n default

```

10). Expose app monitoring Prometheus route:

```

oc expose svc/prometheus-operated -n monitoring

```

11). Collect the app monitoring Prometheus URL:

```
$ oc get routes -n monitoring
NAME                                HOST/PORT
PATH    SERVICES                      PORT    TERMINATION  WILDCARD
prometheus-operated    prometheus-operated-
monitoring.apps.rsocp.os.fyre.ibm.com    prometheus-operated    web
None
```

12). Verify that app monitoring Prometheus can scrape **b2m-nodejs** app. Access the Prometheus URL via browser and select Status -> Targets.

Prometheus Alerts Graph Status ▾ Help

Targets

All Unhealthy

monitoring/app-monitor/0 (1/1 up) [show less](#)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.254.0.35:3001/metrics	UP	endpoint="web" instance="10.254.0.35:3001" job="b2m-nodejs" namespace="default" pod="b2m-nodejs-5677cff49-dpkpw" service="b2m-nodejs"	2.664s ago	13.95ms	

13). Verify that instrumented metrics are collected:

Prometheus Alerts Graph Status ▾ Help

☐ Enable query history

Execute

Graph Console

⏪ Moment ⏩

Element	Value
http_request_duration_ms_bucket{code="200",endpoint="web",instance="10.254.0.35:3001",job="b2m-nodejs",le="+Inf",method="GET",namespace="default",pod="b2m-nodejs-5677cff49-dpkpw",route="/",service="b2m-nodejs"}	12
http_request_duration_ms_bucket{code="200",endpoint="web",instance="10.254.0.35:3001",job="b2m-nodejs",le="+Inf",method="GET",namespace="default",pod="b2m-nodejs-5677cff49-dpkpw",route="/healthz",service="b2m-nodejs"}	440
http_request_duration_ms_bucket{code="200",endpoint="web",instance="10.254.0.35:3001",job="b2m-nodejs",le="0.1",method="GET",namespace="default",pod="b2m-nodejs-5677cff49-dpkpw",route="/",service="b2m-nodejs"}	0
http_request_duration_ms_bucket{code="200",endpoint="web",instance="10.254.0.35:3001",job="b2m-nodejs",le="0.1",method="GET",namespace="default",pod="b2m-nodejs-5677cff49-dpkpw",route="/healthz",service="b2m-nodejs"}	148
http_request_duration_ms_bucket{code="200",endpoint="web",instance="10.254.0.35:3001",job="b2m-nodejs",le="100",method="GET",namespace="default",pod="b2m-nodejs-5677cff49-dpkpw",route="/",service="b2m-nodejs"}	2
http_request_duration_ms_bucket{code="200",endpoint="web",instance="10.254.0.35:3001",job="b2m-nodejs",le="100",method="GET",namespace="default",pod="b2m-nodejs-5677cff49-dpkpw",route="/healthz",service="b2m-nodejs"}	440
http_request_duration_ms_bucket{code="200",endpoint="web",instance="10.254.0.35:3001",job="b2m-nodejs",le="15",method="GET",namespace="default",pod="b2m-nodejs-5677cff49-dpkpw",route="/",service="b2m-nodejs"}	2

Load time: 266ms
Resolution: 14s
Total time series: 20

Deploy the Grafana Operator

1). Deploy the Grafana Operator from OperatorHub using the same steps as for Prometheus Operator. Now you should see it in **Operators -> Installed Operators**.

Red Hat OpenShift Container Platform

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in.

Project: monitoring

Installed Operators

Installed Operators are represented by Cluster Service Versions within this namespace. For more information, see the [Operator Lifecycle Manager documentation](#). Or create an Operator and Cluster Service Version using the [Operator SDK](#).

Filter by name...

Name	Namespace	Deployment	Status	Provided APIs
Grafana Operator 2.0.0 provided by Red Hat	monitoring	grafana-operator	InstallSucceeded Up to date	Grafana Grafana Dashboard Grafana Data Source
Prometheus Operator 0.32.0 provided by Red Hat	monitoring	prometheus-operator	InstallSucceeded Up to date	Prometheus Prometheus Rule Service Monitor Pod Monitor View 1 more...

2). Click on the Grafana Operator link, select **Grafana** tab and click **Create Grafana**.

Red Hat OpenShift Container Platform

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in.

Project: monitoring

[Grafana Operator](#) > Create Grafana [Edit Form](#)

Create Grafana

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

```

1  apiVersion: integreatly.org/v1alpha1
2  kind: Grafana
3  metadata:
4    name: app-monitoring-grafana
5    namespace: monitoring
6  spec:
7    ingress:
8      enabled: true
9    config:
10     auth:
11       disable_signout_menu: true
12     auth.anonymous:
13       enabled: true
14     log:
15       level: warn
16       mode: console
17     security:
18       admin_password: secret

```

[Create](#) [Cancel](#) [Download](#)

3). Modify the name of the Grafana instance to something meaningful. I named it **app-monitoring-grafana**. Click **Create** button. Modify also the admin user name and password.

4). Return to the Grafana Operator details, select **Grafana Data Source** and click **Create Grafana Data Source** button. Rename the **name:** to something meaningful (I named it **app-monitoring-grafana-datasource**) and modify **spec.datasources.url** to your app monitoring prometheus instance. In my case it was **http://prometheus-operated:9090**.

Red Hat OpenShift Container Platform

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in.

Project: monitoring

Grafana Operator > Create Grafana Data Source [Edit Form](#)

Create Grafana Data Source

Create by manually entering YAML or JSON definitions, or by dragging and dropping a file into the editor.

```
1 apiVersion: integreatly.org/v1alpha1
2 kind: GrafanaDataSource
3 metadata:
4   name: app-monitoring-grafanadatasource
5   namespace: monitoring
6 spec:
7   datasources:
8     - access: proxy
9       editable: true
10      isDefault: true
11      jsonData:
12        {
13          "timeInterval": "5s",
14          "name": "Prometheus",
15          "type": "prometheus",
16          "url": "http://prometheus-operated:9090"
17        }
18      version: 1
19   name: example-datasources.yaml
```

[Create](#) [Cancel](#) [Download](#)

The prometheus hostname is the same as the app monitoring prometheus service name. You can find it in **Networking->Services** (filtered by the project where app monitoring prometheus has been deployed).

5). Make the route for Grafana has been created in **Networking->Routes** (project **monitoring**). If it is not listed, create it with command:

```
oc create route edge --service=grafana-service -n monitoring
```

Access the Grafana console URL and logon to Grafana.

Verify the Prometheus datasource has been created and can connect to app monitoring Prometheus.

6). Import provided grafana dashboard: **b2m-nodejs-v2/lab-4/app-monitoring-dashboard.json**.

7). Verify that Grafana dashboard has been provisioned:



Deploy Alertmanager

The Prometheus Operator introduces an **Alertmanager** resource, which allows users to declaratively describe an Alertmanager cluster. To successfully deploy an Alertmanager cluster, it is important to understand the contract between Prometheus and Alertmanager.

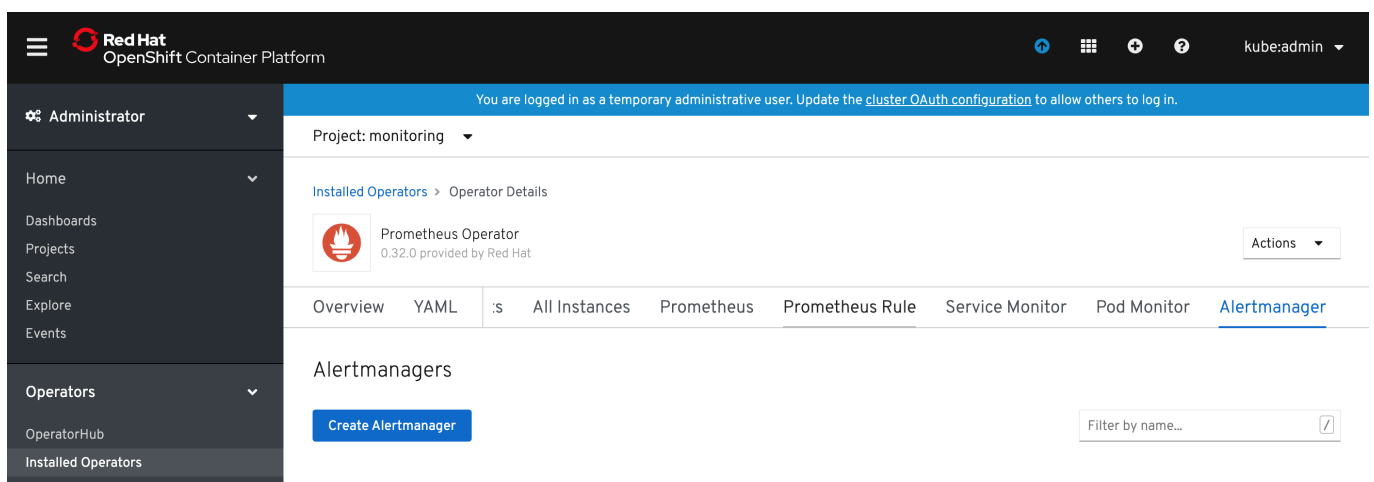
The Alertmanager may be used to:

- Deduplicate alerts fired by Prometheus
- Silence alerts
- Route and send grouped notifications via providers (PagerDuty, OpsGenie, Slack, Netcool Message Bus Probe, etc.)

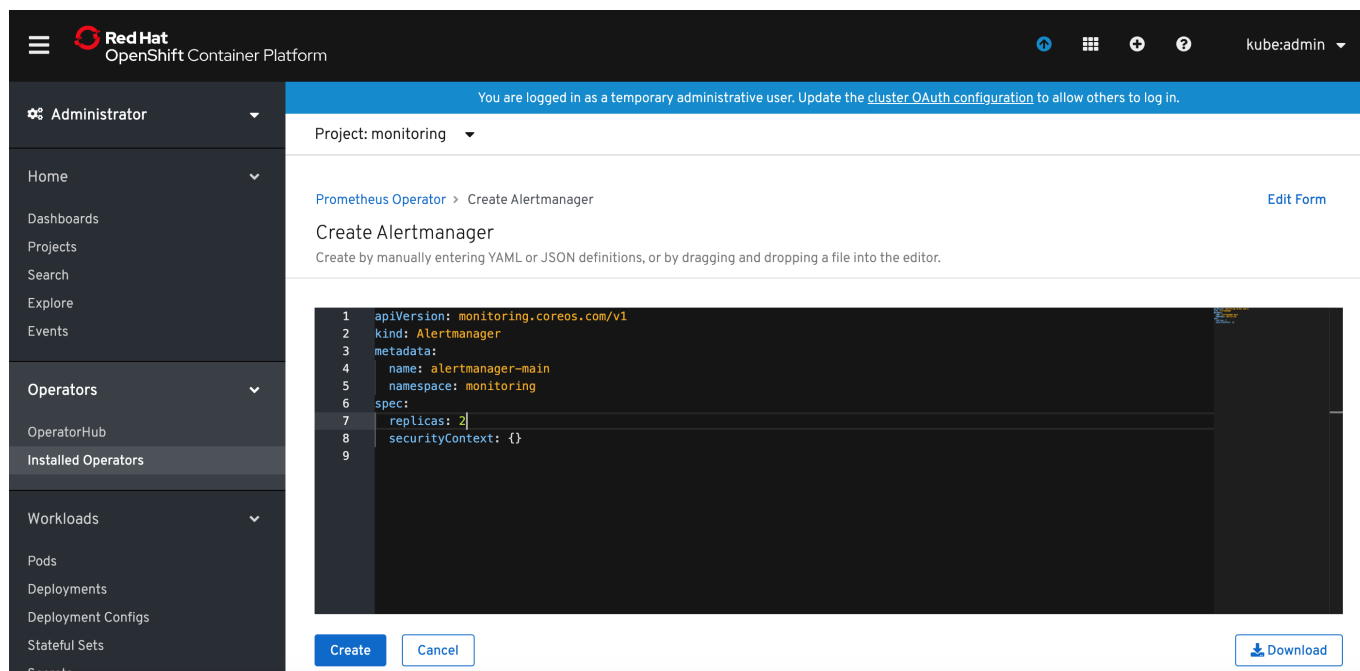
Prometheus' configuration also includes "rule files", which contain the alerting rules. When an alerting rule triggers, it fires that alert against all Alertmanager instances, on every rule evaluation interval. The Alertmanager instances communicate to each other which notifications have already been sent out.

In OpenShift console go to Installed Operators, click on Prometheus Operator instance, scroll tabs to Alertmanager tab.

Click **Create Alertmanager** button.



Specify the desired number of replicas and click **Create** button.



Now you can list the resources of Alertmanager and you should see Alertmanager pods in **Pending** state. This is because Alertmanager can't run without a configuration file.

The Alertmanager instances will not be able to start up, unless a valid configuration is given. The following example configuration sends notifications against a non-existent webhook, allowing the Alertmanager to start up, without issuing any notifications. For more information on configuring Alertmanager, see the [Prometheus Alerting Configuration](#) document.

```
global:
  resolve_timeout: 5m
route:
  group_by: ['job']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: 'webhook'
receivers:
- name: 'webhook'
  webhook_configs:
  - url: 'http://alertmanagerwh:30500/'
```

Save the above Alertmanager config in a file called `alertmanager.yaml` and create a secret from it using **oc**.

```
oc create secret generic alertmanager-alertmanager-main --from-
file=alertmanager.yaml
```

Alertmanager pods should change the status to **Running**.

The screenshot shows the Red Hat OpenShift Container Platform console. The left sidebar contains navigation menus for Administrator, Home, Operators, and Workloads. The main content area displays the 'alertmanager-main' resource in the 'monitoring' project. The 'Resources' tab is selected, showing a table of resources:

Name	Kind	Status	Created
alertmanager-alertmanager-main	StatefulSet	Created	Nov 20, 9:41 am
alertmanager-alertmanager-main-0	Pod	Running	Nov 20, 9:41 am
alertmanager-alertmanager-main-1	Pod	Running	Nov 20, 9:41 am
alertmanager-operated	Service	Created	Nov 20, 9:41 am

The service `alertmanager-operated` has been created automatically and if you want to externally expose Alertmanager UI, create the route using the following command:

```
$ oc create route edge --service=alertmanager-operated -n monitoring
```

Collect the Alertmanager URL:

```
oc get routes alertmanager-operated
NAME                                HOST/PORT
PATH    SERVICES                PORT    TERMINATION  WILDCARD
alertmanager-operated  alertmanager-operated-
monitoring.apps.rsocp.os.fyre.ibm.com  alertmanager-operated  web
edge                    None
```

and verify using web browser:

The screenshot shows the Alertmanager web interface in a web browser. The address bar displays the URL: `alertmanager-operated-monitoring.apps.rsocp.os.fyre.ibm.com/#/alerts`. The interface includes a navigation bar with links for Alertmanager, Alerts, Silences, Status, and Help. A 'New Silence' button is visible in the top right. Below the navigation bar, there is a section for filtering alerts, including a 'Filter' tab, a 'Group' dropdown, and a 'Receiver: All' selector. A search bar is present with a placeholder text 'Custom matcher, e.g. env=production'. At the bottom, a yellow message box states 'No alert groups found'.

This Alertmanager cluster is now fully functional and highly available, but no alerts are fired against it. Configure Prometheus resource to fire alerts to our Alertmanager cluster.

Edit Prometheus resource `spec.alerting` section:

```
spec:
  alerting:
    alertmanagers:
      - name: alertmanager-operated
        namespace: monitoring
        port: web
```

and click **Save**.

Configure Alerting Rules

Alerting Rules for application monitoring can be created from the **Operator Details** view of our Prometheus Operator instance. Click on the **Prometheus Rule** tab and then on **Create Prometheus Rule** button.

The screenshot shows the Red Hat OpenShift Container Platform console interface. The top navigation bar includes the Red Hat logo, the text "OpenShift Container Platform", and user information "kube:ac". A blue banner indicates the user is logged in as a temporary administrative user. The left sidebar contains navigation links: Administrator, Home, Dashboards, Projects, Search, Explore, Events, Operators, OperatorHub, and Installed Operators. The main content area shows the "Prometheus Operator" details for version 0.32.0. The "Prometheus Rule" tab is selected, displaying a "Create Prometheus Rule" button and a filter input field.

Specify alert rule definition in the YAML file. You can use provided ExampleAlert.yaml as an example.

[Installed Operators](#) > [prometheusoperator.0.37.0](#) > [PrometheusRule Details](#)

prometheus-example-rules

[Details](#) [YAML](#) [Resources](#)

```
1  apiVersion: monitoring.coreos.com/v1
2  kind: PrometheusRule
3  metadata:
4    creationTimestamp: '2020-08-10T16:31:54Z'
5    generation: 3
6    labels:
7      prometheus: app-monitor
8      role: alert-rules
9    name: prometheus-example-rules
10   namespace: monitoring
11   resourceVersion: '54972321'
12   selfLink: >=
13     /apis/monitoring.coreos.com/v1/namespaces/monitoring/prometheusrules/prometheus-example-rules
14   uid: 59580574-366f-45a7-a4e6-b6f906c4cc62
15 spec:
16   groups:
17     - name: ./example.rules
18       rules:
19         - alert: ExampleAlert
20           annotations:
21             summary: High request latency
22           expr: >=
23             histogram_quantile(0.95,
24               sum(rate(http_request_duration_ms_bucket[1m])) by (le, service,
25                 route, method)) > 0.08
26           labels:
27             severity: warning
28
```

After short time verify that your alert(s) have been activated using Prometheus UI: