Red Hat OpenShift API Management 1

Administering Red Hat OpenShift API Management

Administering Red Hat OpenShift API Management.

Last Updated: 2024-04-11

Red Hat OpenShift API Management 1 Administering Red Hat OpenShift API Management

Administering Red Hat OpenShift API Management.

Legal Notice

Abstract

This guide provides information about administering Red Hat OpenShift API Management.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. ADMINISTERING OPENSHIFT API MANAGEMENT

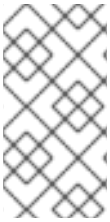
You can perform configuration and monitoring tasks for managed services. With OpenShift API Management you can manage network configurations, define policies, monitor API usage, and update notification settings.

1.1. OVERVIEW OF RATE LIMITING, ALERTING, AND MONITORING IN OPENSHIFT API MANAGEMENT

Red Hat OpenShift API Management provides a limited number of API calls. The API request hard limits are set as part of your Red Hat OpenShift API Management purchase.

You are alerted to ensure you do not exceed the set API usage hard limit. OpenShift API Management uses a tiered approach for alerting customers when the API usage is close to the hard limit, to ensure request calls are not rejected and service is not interrupted.

An email notification is sent to the email address provided during the OpenShift API Management installation. The email notification includes a link to the Grafana dashboard, which provides a visualization of your API usage.



NOTE

You can also access the Grafana dashboard through the OpenShift Dedicated console. To access the dashboard, click the application launcher in OpenShift Dedicated and then click **API Management Dashboards** in the **OpenShift Managed Services** drop-down menu.

A notification is sent in the following instances:

- If API usage is between 80% and 90% for a period of 4 hours, a notification is sent every 4 hours.
- If API usage is between 90% and 95% for a period of 2 hours, a notification is sent every 2 hours.
- If API usage is over 95% for a period of 30 minutes, a notification is sent every 30 minutes.

OpenShift API Management offers you the flexibility to share rate limiting quotas across multiple environments. You can divide a purchased OpenShift API Management subscription level across multiple OpenShift Dedicated environments. For example, if a 10 million API calls per day subscription level is selected, you can allocate 5 million API calls per day to both a stage environment and a production environment. For more information on increasing your Red Hat OpenShift API Management subscription level, refer to the service definition.

1.2. MONITORING SERVICE CAPACITY WITH GRAFANA

You can monitor the service capacity and API usage of OpenShift API Management with Grafana. Access the Grafana dashboard from the OpenShift Dedicated console, to monitor the OpenShift API Management API rate limits for the following durations:

- last minute
- last 24 hours

In the dashboard, you can view, share, and inspect the data for the OpenShift API Management API requests.

Prerequisites

- Red Hat OpenShift API Management was added to your OpenShift Dedicated cluster.

Procedure

1. Click the application launcher in OpenShift Dedicated.
2. Click **API Management Dashboards** in the **OpenShift Managed Services** drop-down menu.
3. You are prompted to log in. Click **Log in with OpenShift**.
4. Enter login credentials, if prompted.
5. Click **Allow selected permissions** to authorize monitoring access.
6. In the Grafana console, click the **Dashboards** dropdown menu from the main menu to access the available dashboards and folders.
7. Click **Manage**.
8. Click **redhat-rhoam-customer-monitoring-operator** to open the folder.
9. Click **Rate Limiting** to access the monitoring data.

NOTE

Alternatively, click **Home** to search for a dashboard by name.

CHAPTER 2. NETWORK POLICIES

A cluster hosts two types of projects:

- Projects associated with managed services. These projects support inbound and outbound connections.
- User projects. These projects support communication from managed services.

In OpenShift Dedicated, there are two approaches to enabling communications:

- Using network policies
- Using the **join-project** option of the **oc** command

In OpenShift API Management, you can use network policies to enable communication and allow 3scale to communicate directly with the service endpoint, instead of the external URL.

You cannot use the **join-projects** option of the **oc** command with managed services projects.

2.1. ENABLING COMMUNICATION BETWEEN MANAGED SERVICES AND CUSTOMER APPLICATIONS

You can create **NetworkPolicy** objects to define granular rules describing the Ingress network traffic that is allowed for projects in your cluster. By default, when you create projects in a cluster, communication between the projects is disabled.

This procedure describes how to enable communication for a project so that managed services, such as 3scale, can access customer applications.

Prerequisites

- You have installed the OpenShift command-line interface (CLI), commonly known as **oc**.

Procedure

1. Log in to the cluster using the **oc** login command.
2. Use the following command to change the project:

   ```
   $ oc project <project_name>
   ```

   where **<project_name>** is the name of a project that you want to accept communications from other projects.
3. Create a **NetworkPolicy** object:
   a. Create a **allow-from-middleware-namespaces.yaml** file.
   b. Define a policy in the file you just created, such as in the following example:

   ```
   apiVersion: networking.k8s.io/v1
   kind: NetworkPolicy
   metadata:
     name: allow-from-middleware-namespaces
   ```

```
spec:
  podSelector:
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              integreatly-middleware-service: 'true'
```

c. Run the following command to create the policy object:

```
$ oc create -f allow-from-middleware-namespaces.yaml -n <project>

networkpolicy "allow-from-middleware-namespaces" created
```

2.2. ENABLING COMMUNICATION BETWEEN MANAGED SERVICES AND PROJECTS

By default, when you create projects in a cluster, communication between the projects is disabled. Use this procedure to enable communication in a project.

Prerequisites

- You have installed the OpenShift command-line interface (CLI), commonly known as **oc**.

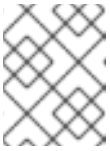
Procedure

1. Log in to the cluster using the **oc** login command.
2. Use the following command to change the project:

   ```
   $ oc project <project_name>
   ```

   where **<project_name>** is the name of a project that you want to accept communications from other projects.
3. Create a NetworkPolicy object:
   a. Create a **NetworkPolicy.yaml** file.
   b. Define a policy in the file you just created, such as in the following example. This policy enables incoming communication for all projects in the cluster:

   ```
   kind: NetworkPolicy
   apiVersion: networking.k8s.io/v1
   metadata:
     name: allow-all
   spec:
     podSelector:
     ingress:
     - {}
   ```



NOTE

This policy configuration enables this project to communicate with all projects in the cluster.

c. Run the following command to create the policy object:

```
$ oc create -f <policy-name>.yaml -n <project>
```

2.3. ENABLING COMMUNICATION BETWEEN CUSTOMER APPLICATIONS

You can enable communication between user applications.

Prerequisites

- You have installed the OpenShift command-line interface (CLI), commonly known as **oc**.

Procedure

1. Log in to the cluster using the **oc** login command.
2. Use the following command to change the project:

```
$ oc project <project_name>
```

<project_name> is the name of a project that you want to accept communications from.

3. Create a NetworkPolicy object:
   a. Create a **allow-from-myproject-namespace.yaml** file.
   b. Define a policy in the file you just created, such as in the following example. This policy enables incoming communication for a specific project (**myproject**):

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-myproject-namespace
spec:
  podSelector:
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              project: myproject
```

4. Run the following commands to create the policy object:

```
$ oc create -f allow-from-myproject-namespace.yaml -n <project>
networkpolicy "allow-from-myproject-namespace" created
```

2.4. DISABLING COMMUNICATION FROM A MANAGED SERVICE TO A PROJECT

By default, projects are created with a template that allows communication from a managed service. For example, 3scale can communicate with all of your projects.

You can disable the communication from a managed service to a project.

Prerequisites

- You have installed the OpenShift command-line interface (CLI), commonly known as **oc**
- You have a project you want to isolate from the managed services.

Procedure

1. Log in to the cluster using the **oc** login command.
2. Use the following command to change the project:

```
$ oc project <project_name>
```

   where **<project_name>** is the name of a project that you want to isolate from the managed services.

3. Create a NetworkPolicy object:
   a. Create a **deny-all.yaml** file.
   b. Define a policy in the file you just created, such as in the following example:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-all
spec:
  podSelector: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              integreatly-middleware-service: 'true'
```

   c. Run the following command to create the policy object:

```
$ oc create -f <policy-name>.yaml -n <project>
```

2.5. ADDITIONAL RESOURCES

- Networking in OpenShift Dedicated

CHAPTER 3. DEFINING METHODS AND METRICS

An application plan sets limits and pricing rules for consumer access to your API. To enable enforcement of limits and rules, designate methods in your API for which to collect individual usage data or add metrics. Add a mapping rule to each designated method and each custom metric. The mapping rule specifies details about the usage data that you want to capture.

For more information about methods and metrics, see Designating methods and adding metrics for capturing usage details.

3.1. ADDING METHODS TO PRODUCTS AND BACKENDS

Adding a method to a product or backend means that you are designating a method in your API for which you want to capture individual usage details. An application plan provides the ability to set a limit for each method that you add to a product or backend. The procedure for adding a method or metric to a product is similar to adding a method or metric to a backend.

Procedure

1. Navigate to **[Your_product_name] > Integration > Methods & Metricsor [Your_backend_name] > Methods & Metrics**
2. Click *New method*.
3. In the **Friendly name** field, enter a short description of the method. This name is displayed in different sections of the 3scale Admin Portal. The friendly name must be unique for the product.

IMPORTANT

Be careful with changing the system name of the methods or deleting them. These changes can break your already deployed 3scale integration if there are mapping rules pointing to the previous system name of the method.

4. In the **System name** field, enter the name of the method in your API to use to report the usage through the 3scale Service Management API. The system name must conform to these rules:
   - Unique in the product or backend
   - Contain only alphanumeric characters, underscore _, hyphen **-** or forward slash /
   - No spaces

   Otherwise, you are free to decide what the system name looks like. It can be the same as the endpoint (/**status**), or, for example, it can include the method and the path (**GET_/status**).
5. Optional: In the **Description** field, enter a more detailed description of the method.
6. Click **Create Method**.

Verification steps

- Added methods are available in your application plans.

Next steps

- Edit limits and pricing rules for each method by going to **[Your_product_name] > Applications > Application Plans > [plan_you_want_to_edit]**.

3.2. ADDING METRICS TO PRODUCTS AND BACKENDS

Adding a metric specifies a usage unit that you want to capture for all calls to your API. An application plan provides the ability to set a limit for each metric that you add to a product or backend. The procedure for adding a method or metric to a product is similar to adding a method or metric to a backend.

Procedure

1. Navigate to **[Your_product_name] > Integration > Methods & Metricsor [Your_backend_name] > Methods & Metrics**
2. Click *New metric*.
3. In the **Friendly name** field, enter a short description of the metric. This name is displayed in different sections of the 3scale Admin Portal. The friendly name must be unique for the product.

IMPORTANT

Be careful with changing the system name of the metrics or deleting them. These changes can break your already deployed 3scale integration if there are mapping rules pointing to the previous system name of the metric.

4. In the **System name** field, enter the name of the metric in your API to use to report the usage through the 3scale Service Management API. The system name must conform to these rules:
   - Unique in the product or backend
   - Contain only alphanumeric characters, underscore _, hyphen **-** or forward slash /
   - No spaces

   Otherwise, you are free to decide what the system name looks like.
5. In the **Unit** field, enter the unit.
   - Use a singular noun, for example, *hit*. The singular will become plural in the analytics charts.
6. Optional: In the **Description** field, enter a more detailed description of the metric.
7. Click **Create Metric**

Verification steps

- Added metrics are available in your application plans.

Next steps

- Edit limits and pricing rules for each metric by going to **[Your_product_name] > Applications > Application Plans > [plan_you_want_to_edit]**.
- Map your metrics to one or more URL pattern by going to **[Your_product_name] > Integration > Mapping Rules**. See Adding mapping rules to methods and metrics .

3.3. ALTERNATIVES FOR IMPORTING METHODS AND METRICS

If your API has multiple endpoints, there are two ways to automatically designate methods and add metrics to 3scale products and backends:

- Importing via Swagger spec .
- Importing via RAML spec .

3.4. ADDING MAPPING RULES TO METHODS AND METRICS

Mapping rules are operations that are mapped to previously created methods and metrics in your products and backends.

NOTE

Mapping rules are required in your previously created methods, however, they are optional for metrics.

Procedure

1. Navigate to **[Your_product_name] > Integration > Mapping Rules**
2. Click **Add Mapping Rule**.
3. The **Verb** field is pre-populated with the HTTP method, **GET**, however you can select other options from the dropdown list.
4. In the **Pattern** field, add a valid URL that starts with an forward slash /. The URL can be from a wildcard you specified inside curly brackets **{}**.
5. In the **Metric or Method to increment**field, select from one of your previously created methods or metrics.
6. The **Increment by** field is pre-populated with **1**, however, change this to suit your own needs.
7. Click the **Create Mapping Rule** button.

Verification steps

- To verify your mapping rules, navigate to **[Your_product_name] > Integration > Methods & Metrics**. Each method and metric should have a check mark in the *Mapped* column.