Red Hat OpenShift Lightspeed 1.0tp1

Configure

Configuring OpenShift Lightspeed

Last Updated: 2024-10-21

Red Hat OpenShift Lightspeed 1.0tp1 Configure

Configuring OpenShift Lightspeed

Legal Notice

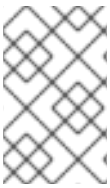
Abstract

This documentation provides information about configuring OpenShift Lightspeed.

Table of Contents

CHAPTER 1. CONFIGURING AND DEPLOYING OPENSHIFT LIGHTSPEED

After the OpenShift Lightspeed Operator is installed, configuring and deploying OpenShift Lightspeed consists of three tasks. First, you create a credential secret using the credentials for your Large Language Model (LLM) provider. Next, you create the **OLSConfig** custom resource (CR) that the Operator uses to deploy the service. Finally, you verify that the Lightspeed service is operating.



NOTE

The instructions assume that you are installing OpenShift Lightspeed using the **kubeadmin** user account. If you are using a regular user account with **cluster-admin** privileges, read the section of the documentation that discusses RBAC.

1.1. CREATING THE CREDENTIALS SECRET BY USING THE WEB CONSOLE

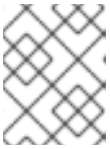
Create a file that is associated with the API token used to access the API of your LLM provider. Typically, you use API tokens to authenticate your LLM provider. Alternatively, Microsoft Azure also supports authentication using Microsoft Entra ID.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role. Alternatively, you are logged in to a user account that has permission to create a secret to store the Provider tokens.
- You have installed the OpenShift Lightspeed Operator.

Procedure

1. Click **Add** in the upper-right corner of the OpenShift web console.
2. Paste the following YAML content into the text area:



NOTE

The YAML parameter is always **apitoken** regardless of what the LLM provider calls the access details.

Credential secret for LLM provider

```
apiVersion: v1
kind: Secret
metadata:
  name: credentials
  namespace: openshift-lightspeed
type: Opaque
stringData:
  apitoken: <your_api_token> (1)
```

(1) The **apitoken** is not **base64** encoded.

Credential secret for IBM WatsonX

```
apiVersion: v1
data:
  apitoken: <your_api_token>
kind: Secret
metadata:
  name: watsonx-api-keys
  namespace: openshift-lightspeed
type: Opaque
```

Credential secret for Microsoft Azure OpenAI

```
apiVersion: v1
data:
  apitoken: <your_api_token>
kind: Secret
metadata:
  name: azure-api-keys
  namespace: openshift-lightspeed
type: Opaque
```

Alternatively, for Microsoft Azure OpenAI you can use Microsoft Entra ID to authenticate your LLM provider. Microsoft Entra ID users must configure the required roles for their Microsoft Azure OpenAI resource. For more information, see the official Microsoft Cognitive Services OpenAI Contributor(Microsoft Azure OpenAI Service documentation).

Credential secret for Microsoft Entra ID

```
apiVersion: v1
data:
  client_id: <base64_encoded_client_id>
  client_secret: <base64_encoded_client_secret>
  tenant_id: <base64_encoded_tenant_id>
kind: Secret
metadata:
  name: azure-api-keys
  namespace: openshift-lightspeed
type: Opaque
```

3. Click **Create**.

1.2. CREATING THE LIGHTSPEED CUSTOM RESOURCE FILE USING THE WEB CONSOLE

The Custom Resource (CR) file contains information that the Operator uses to deploy OpenShift Lightspeed. The specific content of the CR file is unique for each LLM provider. Choose the configuration file that matches your LLM provider.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role. Alternatively, you are logged in to a user account that has permission to create a cluster-scoped CR file.
- You have installed the OpenShift Lightspeed Operator.

Procedure

1. Click **Add** in the upper-right corner of the OpenShift web console.
2. Paste the YAML content for the LLM provider you use into the text area of the web console:

OpenAI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
      - name: myOpenai
        type: openai
        credentialsSecretRef:
          name: credentials
        url: https://api.openai.com/v1
        models:
          - name: gpt-3.5-turbo
  ols:
    defaultModel: gpt-3.5-turbo
    defaultProvider: myOpenai
```

Red Hat Enterprise Linux AI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
    - credentialsSecretRef:
        name: openai-api-keys
      models:
      - name: models/granite-7b-redhat-lab
      name: rhelai
      type: rhelai_vllm
      url: <URL> (1)
  ols:
    defaultProvider: rhelai
    defaultModel: models/granite-7b-redhat-lab
```

(1) The URL endpoint must end with **v1** to be valid. For example, **https://http://3.23.103.8:8000/v1**.

Red Hat OpenShift AI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
    - credentialsSecretRef:
        name: openai-api-keys
      models:
      - name: granite-8b-code-instruct-128k
      name: red_hat_openshift_ai
      type: rhoai_vllm
      url: <url> (1)
  ols:
    defaultProvider: red_hat_openshift_ai
    defaultModel: granite-8b-code-instruct-128k
```

(1) The URL endpoint must end with **v1** to be valid. For example, **https://granite-8b-code-instruct.my-domain.com:443/v1**.

Microsoft Azure OpenAI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
      - credentialsSecretRef:
          name: credentials
        deploymentName: <azure_ai_deployment_name>
        models:
          - name: gpt-35-turbo-16k
        name: myAzure
        type: azure_openai
        url: <azure_ai_deployment_url>
  ols:
    defaultModel: gpt-35-turbo-16k
    defaultProvider: myAzure
```

IBM WatsonX CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
```

```
    - name: myWatsonx
      type: watsonx
      credentialsSecretRef:
        name: credentials
      url: <ibm_watsonx_deployment_name>
      projectId: <ibm_watsonx_project_id>
      models:
        - name: ibm/granite-13b-chat-v2
  ols:
    defaultModel: ibm/granite-13b-chat-v2
    defaultProvider: myWatsonx
```

3. Click **Create**.

1.3. CREATING THE CREDENTIALS SECRET BY USING THE CLI

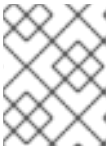
Create a file that is associated with the API token used to access the API of your LLM provider. Typically, you use API tokens to authenticate your LLM provider. Alternatively, Microsoft Azure also supports authentication using Microsoft Entra ID.

Prerequisites

- You have access to the OpenShift CLI (oc) as a user with the **cluster-admin** role. Alternatively, you are logged in to a user account that has permission to create a secret to store the Provider tokens.
- You have installed the OpenShift Lightspeed Operator.

Procedure

1. Create a file that contains the following YAML content:



NOTE

The YAML parameter is always **apitoken** regardless of what the LLM provider calls the access details.

Credential secret for LLM provider

```
apiVersion: v1
kind: Secret
metadata:
  name: credentials
  namespace: openshift-lightspeed
type: Opaque
stringData:
  apitoken: <your_api_token> (1)
```

(1) The **apitoken** is not **base64** encoded.

Credential secret for IBM WatsonX

```
apiVersion: v1
data:
  apitoken: <your_api_token>
kind: Secret
metadata:
  name: watsonx-api-keys
  namespace: openshift-lightspeed
type: Opaque
```

Credential secret for Microsoft Azure OpenAI

```
apiVersion: v1
data:
  apitoken: <your_api_token>
kind: Secret
metadata:
  name: azure-api-keys
  namespace: openshift-lightspeed
type: Opaque
```

Alternatively, for Microsoft Azure OpenAI you can use Microsoft Entra ID to authenticate your LLM provider. Microsoft Entra ID users must configure the required roles for their Microsoft Azure OpenAI resource. For more information, see the official Microsoft Cognitive Services OpenAI Contributor(Microsoft Azure OpenAI Service documentation).

Credential secret for Microsoft Entra ID

```
apiVersion: v1
data:
  client_id: <base64_encoded_client_id>
  client_secret: <base64_encoded_client_secret>
  tenant_id: <base64_encoded_tenant_id>
kind: Secret
metadata:
  name: azure-api-keys
  namespace: openshift-lightspeed
type: Opaque
```

2. Run the following command to create the secret:

```
$ oc create -f /path/to/secret.yaml
```

1.4. CREATING THE LIGHTSPEED CUSTOM RESOURCE FILE USING THE CLI

The Custom Resource (CR) file contains information that the Operator uses to deploy OpenShift Lightspeed. The specific content of the CR file is unique for each LLM provider. Choose the configuration file that matches your LLM provider.

Prerequisites

- You have access to the OpenShift CLI (oc) and are logged in as a user with the **cluster-admin** role. Alternatively, you are logged in to a user account that has permission to create a cluster-scoped CR file.
- You have installed the OpenShift Lightspeed Operator.

Procedure

1. Create an **OLSConfig** file that contains the YAML content for the LLM provider you use:

OpenAI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
      - name: myOpenai
        type: openai
        credentialsSecretRef:
          name: credentials
        url: https://api.openai.com/v1
        models:
          - name: gpt-3.5-turbo
  ols:
    defaultModel: gpt-3.5-turbo
    defaultProvider: myOpenai
```

Red Hat Enterprise Linux AI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
    - credentialsSecretRef:
        name: openai-api-keys
      models:
      - name: models/granite-7b-redhat-lab
      name: rhelai
      type: rhelai_vllm
      url: <URL> (1)
  ols:
    defaultProvider: rhelai
    defaultModel: models/granite-7b-redhat-lab
```

(1) The URL endpoint must end with **v1** to be valid. For example, **https://http://3.23.103.8:8000/v1**.

Red Hat OpenShift AI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
    - credentialsSecretRef:
        name: openai-api-keys
      models:
      - name: granite-8b-code-instruct-128k
      name: red_hat_openshift_ai
      type: rhoai_vllm
      url: <url> (1)
  ols:
    defaultProvider: red_hat_openshift_ai
    defaultModel: granite-8b-code-instruct-128k
```

1 The URL endpoint must end with **v1** to be valid. For example, **https://granite-8b-code-instruct.my-domain.com:443/v1**.

Microsoft Azure OpenAI CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
      - credentialsSecretRef:
          name: credentials
        deploymentName: <azure_ai_deployment_name>
        models:
          - name: gpt-35-turbo-16k
        name: myAzure
        type: azure_openai
        url: <azure_ai_deployment_url>
  ols:
    defaultModel: gpt-35-turbo-16k
    defaultProvider: myAzure
```

IBM WatsonX CR file

```
apiVersion: ols.openshift.io/v1alpha1
kind: OLSConfig
metadata:
  name: cluster
spec:
  llm:
    providers:
      - name: myWatsonx
        type: watsonx
        credentialsSecretRef:
```

```
      name: credentials
    url: <ibm_watsonx_deployment_name>
    projectId: <ibm_watsonx_project_id>
    models:
      - name: ibm/granite-13b-chat-v2
ols:
  defaultModel: ibm/granite-13b-chat-v2
  defaultProvider: myWatsonx
```

2. Run the following command:

```
$ oc create -f /path/to/config-cr.yaml
```

The Operator deploys OpenShift Lightspeed using the information in YAML configuration file.

1.5. VERIFYING THE OPENSHIFT LIGHTSPEED DEPLOYMENT

After the OpenShift Lightspeed service is deployed, verify that it is operating.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role.
- You have access to the OpenShift CLI (oc).
- You have installed the OpenShift Lightspeed Operator.
- You have created the credentials secret and the **OLSConfig** Custom Resource configuration file.

Procedure

1. In the OpenShift Container Platform web console, select the **Developer** perspective from the drop-down list at the top of the pane.
2. Click the **Project** drop-down lsit.
3. Enable the toggle switch to show default projects.
4. Select **openshift-lightspeed** from the list.
5. Click **Topology**.
   When the circle around the Lightspeed icon turns dark blue, the service is ready.
6. Verify that the OpenShift Lightspeed is ready by running the following command:

```
$ oc logs deployment/lightspeed-app-server -c lightspeed-service-api -n openshift-lightspeed
| grep Uvicorn
```

Example output

```
INFO:  Uvicorn running on https://0.0.0.0:8443 (Press CTRL+C to quit)
```

1.6. ABOUT LIGHTSPEED AND ROLE BASED ACCESS CONTROL (RBAC)

Role-Based Access Control (RBAC) is a system security approach to restricting system access to authorized users who have defined roles and permissions.

OpenShift Lightspeed RBAC is binary. By default, not all cluster users have access to the OpenShift Lightspeed interface. Access must be granted by a user who can grant permissions. All users of an OpenShift cluster with OpenShift Lightspeed installed can see the Lightspeed button; however, only users with permissions can submit questions to Lightspeed.

If you want to evaluate the RBAC features of OpenShift Lightspeed, your cluster will need users other than the **kubeadmin** account. The **kubeadmin** account always has access to OpenShift Lightspeed.

1.6.1. Granting access to an individual user

This procedure explains how to grant access to an individual user.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role. Alternatively, you are logged in as a user with the ability to grant permissions.
- You have deployed the OpenShift Lightspeed service.
- You have access to the OpenShift CLI (oc).

Procedure

1. Run the following command at the command line:

```
$ oc adm policy add-cluster-role-to-user \
lightspeed-operator-query-access <user_name>
```

Alternatively, you can use a YAML file when granting access to an individual user by using the following command:

```
$ oc adm policy add-cluster-role-to-user lightspeed-operator-query-access <user_name> -o
yaml --dry-run
```

The terminal returns the following output:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  creationTimestamp: null
  name: lightspeed-operator-query-access
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: lightspeed-operator-query-access
subjects:
```

```
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: <user_name> 1
```

1 Enter the actual user name in place of **<user_name>** before creating the object.

Save the output as a YAML file, and run the following command to grant user access:

```
$ oc create -f <yaml_filename>
```

1.6.2. Granting access to a user group

This procedure explains how to grant access to a user group. If your cluster has more advanced identity management configured, including user groups, you can grant all users of a specific group access to the OpenShift Lightspeed service.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role. Alternatively, you are logged in as a user with the ability to grant permissions.
- You have deployed the OpenShift Lightspeed service.
- You have access to the OpenShift CLI (oc).

Procedure

1. Run the following command at the command line:

```
$ oc adm policy add-cluster-role-to-group \
lightspeed-operator-query-access <group_name>
```

Alternatively, you can use a YAML file when granting access to a user group by using the following command:

```
$ oc adm policy add-cluster-role-to-group lightspeed-operator-query-access <group_name> -
o yaml --dry-run
```

The terminal returns the following output:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  creationTimestamp: null
  name: lightspeed-operator-query-access
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: lightspeed-operator-query-access
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: <user_group> 1
```

–

1 Enter the actual user group in place of **<user_group>** before creating the object.

Save the output as a YAML file, and run the following command to grant access to the user group:

```
$ oc create -f <yaml_filename>
```

1.7. FILTERING AND REDACTING INFORMATION

You can configure OpenShift Lightspeed to filter or redact information from being sent to the LLM provider. The following example shows how to modify the **OLSConfig** file to redact IP addresses.

NOTE

You should test your regular expressions against sample data to confirm that they are catching the information you want to filter or redact, and that they are not accidentally catching information you do not want to filter or redact. There are several third-party websites that you can use to test your regular expressions. When using third-party sites, you should practice caution with regards to sharing your private data. Alternatively, you can test the regular expressions locally using Python. In Python, it is possible to design very computationally-expensive regular expressions. Using several complex expressions as query filters can adversely impact the performance of OpenShift Lightspeed.

Prerequisites

- You are logged in to the OpenShift Container Platform web console as a user with the **cluster-admin** role.
- You have access to the OpenShift CLI (oc).
- You have installed the OpenShift Lightspeed Operator and deployed the OpenShift Lightspeed service.

Procedure

1. Modify the **OLSConfig** file and create an entry for each regular expression to filter. The following example redacts IP addresses:

   Example custom resource file

   ```
   spec:
     ols:
       queryFilters:
         - name: ip-address
           pattern: '((25[0-5]|(2[0-4]|1\d|[1-9]|)\d)\.?\b){4}'
           replaceWith: <IP_ADDRESS>
   ```

2. Run the following command to apply the modified OpenShift Lightspeed custom configuration:

   ```
   $ oc apply -f OLSConfig.yaml
   ```