Red Hat OpenShift AI Cloud Service 1

服务模型

Red Hat OpenShift AI Cloud Service 中的服务模型

Last Updated: 2024-06-26

Red Hat OpenShift AI Cloud Service 1 服务模型

Red Hat OpenShift AI Cloud Service 中的服务模型

法律通告

摘要

Red Hat OpenShift AI Cloud Service 中的服务模型。通过服务经过培训的模式，您可以测试和实施智能应用程序。

目录

第 1 章 关于模型服务

在 Red Hat OpenShift AI 上服务经过培训的模型意味着在 OpenShift 集群上部署模型进行测试，然后将它们集成到智能应用程序中。部署模型使其可作为服务使用 API 访问。这可让您根据通过 API 调用提供的数据输入返回预测。这个过程被称为 *推断(inencing)的型号*。当您在 OpenShift AI 上提供模型时，您可以在仪表板中显示您可以访问部署模型的端点。

OpenShift AI 提供以下模型服务平台：

单模式服务平台

对于部署大型语言模型(LLMs)的大型 *模型，OpenShift AI 包括一个基于 KServe 组件的单型号* 服务平台。由于每个模型都从其自身的模型服务器部署，因此单模式服务平台可帮助您部署、监控、扩展和维护需要增加资源的大型模型。

多模式服务平台

对于部署中小型模型和中型模型，OpenShift AI 包括一个基于 ModelMesh 组件的 *多模型服务平台*。在多模型服务平台上，您可以在同一模型服务器上部署多个模型。每个部署的模型共享服务器资源。这种方法可能对具有有限计算资源或 pod 的 OpenShift 集群具有优势。

第 2 章 提供中小型模型和中型模型

对于部署中小型模型和中型模型，OpenShift AI 包括一个基于 ModelMesh 组件的 *多模型服务平台*。在多模型服务平台上，可以从同一模型服务器部署多个模型，并共享服务器资源。

2.1. 配置模型服务器

2.1.1. 启用多型号服务平台

要使用多型号服务平台，您必须首先启用平台。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的 admin 组的一部分（例如，**rhoai-admins**）。
- 您的集群管理员 *没有* 编辑 OpenShift AI 仪表板配置，以禁用选择多型号服务平台的功能，该平台使用 ModelMesh 组件。如需更多信息，请参阅 控制面板配置选项。

流程

1. 在 OpenShift AI 仪表板的左侧菜单中，点 **Settings → Cluster settings**。
2. 找到 **Model serving platform** 部分。
3. 选中 **Multi-model serving platform** 复选框。
4. 点 **Save Changes**。

2.1.2. 为多模型服务平台添加自定义模型运行时

模型运行时增加了对指定模型框架和这些框架支持的模型格式的支持。默认情况下，多型号服务平台包括 OpenVINO Model Server 运行时。如果默认运行时不满足您的需要，您还可以添加自己的自定义运行时，如支持特定的模型格式。

作为管理员，您可以使用 Red Hat OpenShift AI 仪表板添加和启用自定义模型运行时。然后，当您为多模型服务平台创建新模型服务器时，您可以选择自定义运行时。

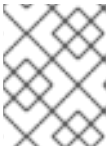
注意

OpenShift AI 允许您添加自己的自定义运行时，但不支持运行时本身。您需要正确配置和维护自定义运行时。您还负责确保您获得许可，以使用您添加的任何自定义运行时。

先决条件

- 您已以管理员身份登录到 OpenShift AI。
- 熟悉如何 在项目中添加模型服务器。添加自定义模型运行时后，您必须配置新的模型服务器以使用运行时。

- 您已查看了 kserve/modelmesh-serving 存储库中的运行时示例。您可以使用这些示例作为起点。但是，每个运行时都需要一些进一步的修改，然后才能在 OpenShift AI 中部署它。以下流程描述了所需的修改。



注意

OpenShift AI 默认包括 OpenVINO Model Server 运行时。您不需要将此运行时添加到 OpenShift AI。

流程

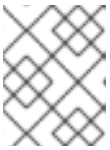
1. 在 OpenShift AI 仪表板中点 **Settings** > **Serving runtime**。
Serving 运行时 页面将打开，并显示已安装和启用的 model-serving 运行时。

2. 要添加自定义运行时，请选择以下选项之一：

   - 要使用现有运行时（如 OpenVINO Model Server runtime）启动，请点现有运行时旁的操作菜单(RCU)，然后点 **Duplicate**。

   - 要添加新的自定义运行时，请单击 **Add serving runtime**。

3. 在 **Select the model serving Platform this runtime support**列表中，选择 **Multi-model serving platform**。



注意

多模式服务平台只支持 REST 协议。因此，您无法更改 **Select the API 协议中的默认值，这个运行时** 支持列表。

4. 可选：如果您启动新的运行时（而不是复制现有运行时），请选择以下选项之一来添加代码：

   - **上传 YAML 文件**

     a. 点 **Upload files**。

     b. 在文件浏览器中，选择计算机上的 YAML 文件。此文件可能是您从 kserve/modelmesh-serving 存储库下载的示例运行时之一。
     嵌入的 YAML 编辑器将打开，并显示您上传的文件内容。

   - **在编辑器中直接输入 YAML 代码**

     a. 点 **Start from scratch**。

     b. 在嵌入式编辑器中直接输入或粘贴 YAML 代码。您粘贴的 YAML 可能会从 kserve/modelmesh-serving 存储库中的一个示例运行时复制。

5. 可选：如果您要在 kserve/modelmesh-serving 仓库中添加其中一个示例运行时，请执行以下修改：

   a. 在 YAML 编辑器中，找到运行时的 **kind** 字段。将此字段的值更新为 **ServingRuntime**。

   b. 在 kserve/modelmesh-serving 存储库中的 kustomization.yaml 文件中，记录您要添加的运行时的 **newName** 和 **newTag** 值。您将在后续步骤中指定这些值。

   c. 在自定义运行时的 YAML 编辑器中，找到 **containers.image** 字段。

d. 根据之前在 kustomization.yaml 文件中记录的值，以 **newName:newTag** 格式更新 **containers.image** 字段的值。显示了一些示例。

NVIDIA Triton Inference Server

image: nvcr.io/nvidia/tritonserver:23.04-py3

Seldon Python MLServer

Image: seldonio/mlserver:1.3.2

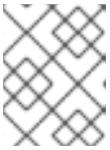
TorchServe

Image: pytorch/torchserve:0.7.1-cpu

6. 在 **metadata.name** 字段中，确保添加的运行时值是唯一的（即，该值与您已添加的运行时不匹配）。

7. 可选： 要为要添加的运行时配置自定义显示名称，请添加 **metadata.annotations.openshift.io/display-name** 字段并指定值，如下例所示：

```
apiVersion: serving.kserve.io/v1alpha1
kind: ServingRuntime
metadata:
  name: mlserver-0.x
  annotations:
    openshift.io/display-name: MLServer
```



注意

如果没有为运行时配置自定义显示名称，OpenShift AI 会显示 **metadata.name** 字段的值。

8. 点击 **Add**。
Serving 运行时页面将打开，并显示所安装的运行时的更新列表。观察您添加的运行时会自动启用。

9. 可选： 要编辑自定义运行时，点操作菜单（需要）并选择 **Edit**。

验证

- 您添加的自定义 model-serving 运行时会处于 **Serving 运行时** 页面中的启用状态。

其他资源

- 要了解如何配置使用您添加的自定义模型运行时的模型服务器，请参阅 向数据科学项目中添加模型服务器。

2.1.3. 为多型号服务平台添加模型服务器

当您启用了多型号服务平台后，您必须配置模型服务器来部署模型。如果您需要额外的计算能力以用于大型数据集，您可以将加速器分配给您的模型服务器。

先决条件

- 您已登陆到 Red Hat OpenShift AI。

- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh **oai-users** 或 **rhoai-admins** ）。
- 您已创建了可添加模型服务器的数据科学项目。
- 您已启用了多型号服务平台。
- 如果要将自定义模型运行时用于模型服务器，您添加了并启用运行时。请参阅 添加自定义模型运行时。
- 如果要将图形处理单元(GPU)与模型服务器搭配使用，在 OpenShift AI 中启用了 GPU 支持。请参阅 OpenShift AI 中的启用 GPU 支持 。

流程

1. 在 OpenShift AI 仪表板的左侧菜单中，单击 **Data Science Projects**。
Data Science Projects 页面将打开。
2. 点您要为其配置模型服务器的项目名称。
此时会打开项目详情页面。
3. 点 **Models** 选项卡。
4. 执行以下操作之一：
   - 如果您看到 **Multi-model 服务平台** 标题，点标题上的 **Add model server**。
   - 如果没有看到任何标题，点 **Add model server** 按钮。

   此时会打开 **Add model server** 对话框。
5. 在 **Model server name** 字段中输入模型服务器的唯一名称。
6. 从 **Serving 运行时** 列表中，选择一个在 OpenShift AI 部署中安装并启用的 model-serving 运行时。

注意

如果您将 *自定义* 模型运行时与您的模型服务器搭配使用，并希望使用 GPU，您必须确保自定义运行时支持 GPU，并适当地配置为使用它们。

7. 在 **Number of model replicas to deploy** 字段中，指定一个值。
8. 从 **Model server size** 列表中选择一个值。
9. 可选：如果您在上一步中选择了 **Custom**，请在 **Model server size** 部分中配置以下设置以自定义模型服务器：
   a. 在 **CPU 请求** 字段中，指定用于您的模型服务器的 CPU 数量。使用此字段旁边的列表来指定内核或 millicores 的值。
   b. 在 **CPU limit** 字段中，指定用于模型服务器的最大 CPU 数量。使用此字段旁边的列表来指定内核或 millicores 的值。
   c. 在 **Memory requested** 字段中，以 KB (Gi)为单位为模型服务器指定请求的内存。
   d. 在 **Memory limit** 字段中，以 KB (Gi)为单位为模型服务器指定最大内存限值。

10. 可选：在 **Accelerator** 列表中选择一个加速器。
    a. 如果您在上一步中选择了加速器，请指定要使用的加速器数量。
11. 可选：在 **Model route** 部分中，选择 **Make deployed model available via an external route**复选框，使部署的模型可供外部客户端使用。
12. 可选： 在 **Token authorization** 部分中，选择 **Require token authentication** 复选框，以为您的模型服务器要求令牌身份验证。要完成配置令牌身份验证，请执行以下操作：
    a. 在 **Service account name**字段中输入要为其生成令牌的服务帐户名称。配置模型服务器时，生成的令牌会被创建并显示在 **Token secret** 字段中。
    b. 要添加额外服务帐户，请点 **Add a service account** 并输入另一个服务帐户名称。
13. 点击 **Add**。
    - 您配置的模型服务器会出现在项目的 **Models 和 model servers** 列表中。
14. 可选： 要更新模型服务器，点模型服务器**旁的操作菜**单(HBAC)，然后选择 **Edit model server**。

2.1.4. 删除模型服务器

当您不再需要模型服务器到主机模型时，您可以从您的数据科学项目中删除它。

注意

当您删除模型服务器时，您也会删除托管在该模型服务器上的模型。因此，应用程序不再使用模型。

先决条件

- 您已创建了数据科学项目和关联的模型服务器。
- 您已通知用户访问模型将不再可用的应用程序。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh **oai-users** 或 **rhoai-admins**）。

流程

1. 在 OpenShift AI 仪表板中，单击 **Data Science Projects**。
   Data Science Projects 页面将打开。
2. 点您要从中删除模型服务器的项目名称。
   此时会打开项目详情页面。
3. 点 **Models** 选项卡。
4. 点击您要删除的模型服务器项目**旁边的操作菜**单（需要），然后点 **Delete model server**。
   此时会打开 **Delete model server**对话框。
5. 在文本字段中输入模型服务器的名称，以确认您想要删除它。
6. 点 **Delete model server**。

验证

验证

- 您删除的模型服务器不再显示在项目的 **Models** 选项卡中。

2.2. 使用部署的模型

2.2.1. 使用多型号服务平台部署模型

您可以在 OpenShift AI 上部署受培训的模型，以便您测试和实施它们到智能应用程序中。部署模型使其可作为服务使用 API 访问。这可让您根据数据输入返回预测。

当您启用了多型号服务平台后，您可以在平台上部署模型。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组（例如，rho **ai-users**）的一部分。
- 您已启用了多型号服务平台。
- 您已创建了数据科学项目并添加了模型服务器。
- 您可以访问 S3 兼容对象存储。
- 对于您要部署的模型，您知道 S3 兼容对象存储存储桶中的关联文件夹路径。

流程

1. 在 OpenShift AI 仪表板的左侧菜单中，单击 **Data Science Projects**。
   Data Science Projects 页面将打开。
2. 点击您要在其中部署模型的项目的名称。
   此时会打开项目详情页面。
3. 点 **Models** 选项卡。
4. 点 **Deploy model**。
5. 配置部署模型的属性，如下所示：
   a. 在 **Model name** 字段中，输入您要部署的模型的唯一名称。
   b. 从 **Model 框架** 列表中，为您的模型选择一个框架。

注意

Model 框架 列表仅显示配置模型服务器时指定的模型运行时支持的框架。

   c. 要指定您要从 S3 兼容对象存储部署的模型位置，请执行以下任一操作：
   - **使用现有数据连接**
     i. 选择 **Existing data connection**。

ii. 在 **Name** 列表中选择您之前定义的数据连接。

iii. 在 **Path** 字段中，输入包含指定数据源中的模型的文件夹路径。

- **使用新数据连接**

i. 要定义模型可以访问的新数据连接，请选择 **New data connection**。

ii. 在 **Name** 字段中输入数据连接的唯一名称。

iii. 在 **Access key** 字段中，输入 S3 兼容对象存储供应商的访问密钥 ID。

iv. 在 **Secret key** 字段中，为您指定的 S3 兼容对象存储帐户输入 secret 访问密钥。

v. 在 **Endpoint** 字段中，输入 S3 兼容对象存储桶的端点。

vi. 在 **Region** 字段中，输入 S3 兼容对象存储帐户的默认区域。

vii. 在 **Bucket** 字段中，输入 S3 兼容对象存储桶的名称。

viii. 在 **Path** 字段中，在 S3 兼容对象存储中输入包含您的数据文件的文件夹路径。

d. 点 **Deploy**。

验证

- 确认部署的模型显示在项目的 **Models** 选项卡中，并在仪表板的 **Model Serving** 页面中显示 **Status** 列中的 checkmark。

2.2.2. 查看部署的模型

要分析工作的结果，您可以查看 Red Hat OpenShift AI 上部署的模型列表。您还可以查看部署模型及其端点的当前状态。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh **oai-users** 或 **rhoai-admins**）。

流程

1. 在 OpenShift AI 仪表板中，点 **Model Serving**。
   此时会打开 **Deployed models** 页面。

   对于每个模型，页面会显示模型名称、部署模型的项目、模型使用的模型运行时以及部署状态等详细信息。

2. 可选： 对于给定模型，点 **Inference endpoint** 列中的链接来查看部署的模型的 inference 端点。

验证

- 在 **Deployed** model 页面中显示之前部署的数据科学模型列表。

2.2.3. 更新已部署模型的部署属性

您可以更新之前部署的模型的部署属性。这可让您更改模型的数据连接和名称。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh **oai-users** 或 **rhoai-admins**）。
- 您已在 OpenShift AI 上部署了模型。

流程

1. 在 OpenShift AI 仪表板中点 **Model serving**。
   此时会打开 **Deployed models** 页面。
2. 点您要更新的部署属性旁的操作菜单（⋮），然后点 **Edit**。
   此时会打开 **Deploy model** 对话框。
3. 更新模型的部署属性，如下所示：
   a. 在 **Model Name** 字段中输入模型的新唯一名称。
   b. 从 **Model 框架** 列表中，为您的模型选择一个框架。

注意

Model 框架 列表仅显示配置模型服务器时指定的模型运行时支持的框架。

   c. 要更新如何指定模型的位置，请执行以下操作之一：
      - **如果您之前指定了现有数据连接**
        i. 在 **Path** 字段中，更新在指定数据源中包含模型的文件夹路径。
      - **如果您之前指定了新数据连接**
        i. 在 **Name** 字段中，更新数据连接的唯一名称。
        ii. 在 **Access key** 字段中，更新 S3 兼容对象存储供应商的访问密钥 ID。
        iii. 在 **Secret key** 字段中，更新您指定的 S3 兼容对象存储帐户的 secret 访问密钥。
        iv. 在 **Endpoint** 字段中，更新 S3 兼容对象存储桶的端点。
        v. 在 **Region** 字段中，更新 S3 兼容对象存储帐户的默认区域。
        vi. 在 **Bucket** 字段中，更新 S3 兼容对象存储桶的名称。
        vii. 在 **Path** 字段中，更新 S3 兼容对象存储中的文件夹路径，其中包含您的数据文件。
   d. 点 **Deploy**。

验证

- 您更新的部署属性的模型显示在仪表板的 **Model Serving** 页面中。

2.2.4. 删除部署的模型

您可以删除之前部署的模型。这可让您删除不再需要的部署模型。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh **oai-users** 或 **rhoai-admins**）。
- 您已部署了模型。

流程

1. 在 OpenShift AI 仪表板中点 **Model serving**。
   此时会打开 **Deployed models** 页面。
2. 点您要删除的部署模型旁的操作菜单（⋮），然后点 **Delete**。
   此时会打开 **Delete deployed model** 对话框。
3. 在文本字段中输入部署模型的名称，以确认您想要删除它。
4. 点 **Delete deployed model**。

验证

- 您删除的模型不再显示在 **Deployed models** 页面中。

2.3. 查看多模型服务平台的模型服务运行时指标

集群管理员为多模型服务平台配置了监控后，非管理员用户可以使用 OpenShift Web 控制台查看 ModelMesh 组件的模型运行时指标。

先决条件

- 您可以使用开发人员或具有查看指标的项目查看权限的用户访问 OpenShift 集群。
- 熟悉在用户定义的项目中查询指标。请参阅 以开发者(Red Hat OpenShift Dedicated)身份 查询用户定义的项目的指标，或以开发者(Red Hat OpenShift Service on AWS)身份 查询用户定义的项目的指标。

流程

1. 登录 OpenShift Web 控制台。
2. 切换到 **Developer** 视角。
3. 在左侧菜单中点 **Observe**。
4. 如以开发者(Red Hat OpenShift Dedicated) 查询用户定义的项目的指标 中所述，或以开发者身份查询用户定义的项目的指标 (Red Hat OpenShift Service on AWS)，使用 Web 控制台对 **modelmesh FirmwareSchema** 指标运行查询。

2.4. 监控模型性能

2.4.1. 查看模型服务器上所有模型的性能指标

在 OpenShift AI 中，您可以监控模型服务器上部署的所有模型的以下指标：

- **HTTP 请求** - 服务器上所有模型的 HTTP 请求数或成功。
  注： 您还可以查看特定模型失败的 HTTP 请求数，如 查看已部署模型的 HTTP 请求指标 中所述。
- **平均响应时间(ms)** - 对于服务器上的所有模型，对请求进行平均用时。
- **CPU 使用率(%)** - 服务器上所有模型目前使用的 CPU 容量的百分比。
- **内存使用率(%)** - 服务器上所有模型当前使用的系统内存的百分比。

您可以为这些指标指定时间范围和刷新间隔，以帮助您确定峰值用量时间以及模型在指定时间如何执行。

先决条件

- 已安装 Red Hat OpenShift AI。
- 在安装 OpenShift AI 的 OpenShift 集群中，启用了用户工作负载监控。
- 您已登录到 OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh **oai-users** 或 **rhoai-admins**）。
- 您已在多模型服务平台上部署了模型。

流程

1. 在 OpenShift AI 仪表板导航菜单中点 **Data Science Projects**。
   Data Science Projects 页面将打开。
2. 点包含要监控的数据科学模型的项目名称。
3. 在项目详情页面中，点 **Models** 选项卡。
4. 在您感兴趣的模型服务器行中，点操作菜单(HBAC)，然后选择 **View model server metrics**。
5. 可选：在模型服务器的指标页面中设置以下选项：
   - **时间范围** - 指定跟踪指标的时间长度。您可以选择这些值之一：1 小时、24 小时、7 天和 30 天。
   - **刷新间隔** - 指定指标页面中图形的刷新频率（以显示最新的数据）。您可以选择这些值之一：15 秒、30 秒、1 分钟、5 分钟、15 分钟、30 分钟、1 小时、2 小时和 1 天。
6. 向下滚动以查看 HTTP 请求、平均响应时间、CPU 使用率和内存使用率的数据图形。

验证

在模型服务器的指标页面中，图形提供性能指标数据。

2.4.2. 查看已部署模型的 HTTP 请求指标

您可以查看显示在多模型服务平台上部署的特定模型的 HTTP 请求失败或成功的图形。

先决条件

- 已安装 Red Hat OpenShift AI。
- 在安装 OpenShift AI 的 OpenShift 集群中，启用了用户工作负载监控。
- 您的集群管理员 *没有* 编辑 OpenShift AI 仪表板配置，以隐藏 **Model Serving** 页面中的 **Endpoint Performance** 选项卡。如需更多信息，请参阅 控制面板配置选项。
- 您已登录到 OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh **oai-users** 或 **rhoai-admins**）。
- 您已在多模型服务平台上部署了模型。

流程

1. 在 OpenShift AI 仪表板导航菜单中选择 **Model Serving**。
2. 在 **Deployed models** 页面中，选择您感兴趣的模型。
3. 可选：在 **Endpoint performance** 选项卡中设置以下选项：
   - **时间范围** - 指定跟踪指标的时间长度。您可以选择这些值之一：1 小时、24 小时、7 天和 30 天。
   - **刷新间隔** - 指定指标页面中图形的刷新频率（以显示最新的数据）。您可以选择这些值之一：15 秒、30 秒、1 分钟、5 分钟、15 分钟、30 分钟、1 小时、2 小时和 1 天。

验证

Endpoint performance 选项卡显示模型的 HTTP 指标图。

第 3 章 提供大型模型

对于部署大型语言模型(LLMs)的大型 *模型，Red Hat OpenShift AI 包括一个基于 KServe 组件的模型服务平台*。由于每个模型都从自己的模型服务器部署，单一模型服务平台可帮助您部署、监控、扩展和维护需要增加资源的大型模型。

3.1. 关于单模式服务平台

对于部署大型语言模型(LLMs)的大型模型，OpenShift AI 包括一个基于 KServe 组件的单型号服务平台。由于每个模型都部署在自己的模型服务器上，单模式服务平台可帮助您部署、监控、扩展和维护需要增加资源的大型模型。

3.1.1. 组件

- KServe: 一个 Kubernetes 自定义资源定义(CRD)，用于编配所有类型的模型服务。KServe 包括模型-serving 运行时，用于实现给定类型的模型服务器加载。KServe 还处理部署对象、存储访问和网络设置的生命周期。
- Red Hat OpenShift Serverless: 一个云原生开发模型，允许无服务器部署模型。OpenShift Serverless 基于开源 Knative 项目。
- Red Hat OpenShift Service Mesh: 服务网格网络层，用于管理流量流并强制实施访问策略。OpenShift Service Mesh 基于开源 Istio 项目。

3.1.2. 安装选项

要安装单模式服务平台，有以下选项：

自动化安装

如果您还没有在 OpenShift 集群上创建 **ServiceMeshControlPlane** 或 **KNativeServing** 资源，您可以将 Red Hat OpenShift AI Operator 配置为安装 KServe 及其依赖项。
有关自动安装的更多信息，请参阅配置自动安装 KServe。

手动安装

如果您已在 OpenShift 集群上创建了 **ServiceMeshControlPlane** 或 **KNativeServing** 资源，*则无法*配置 Red Hat OpenShift AI Operator 来安装 KServe 及其依赖项。在这种情况下，您必须手动安装 KServe。
有关手动安装的更多信息，请参阅 手动安装 KServe。

3.1.3. model-serving 运行时

安装 KServe 后，您可以使用 OpenShift AI 仪表板使用预安装或自定义模型运行时部署模型。

OpenShift AI 为 KServe 包含以下预安装的运行时：

- **TGIS Standalone ServingRuntime for KServe** 为启用 TGI 的模型提供服务的运行时
- **Caikit-TGIS ServingRuntime for KServe** 以 Caikit 格式提供模型的复合运行时
- **OpenVINO Model Server** ：为为 Intel 架构优化的服务模型的可扩展、高性能运行时
- **vLLM ServingRuntime for KServe** 一个高吞吐量和内存效率的推测，为大型语言模型提供运行时

注意

- 文本 Generation Inference Server (TGIS) 基于 Hugging Face TGI 的早期分叉。红帽将继续开发独立 TGIS 运行时来支持 TGI 模型。如果模型无法在 OpenShift AI 的当前版本中工作，则可能会在以后的版本中添加支持。同时，您还可以添加自己的自定义运行时来支持 TGI 模型。如需更多信息，请参阅为单模式服务平台添加自定义模型运行时。
- 复合 Caikit-TGIS 运行时基于 Caikit 和 Text Generation Inference Server (TGIS)。要使用这个运行时，您必须将模型转换为 Caikit 格式。例如，请参阅 caikit-tgis-serving 存储库中的将 Hugging Face Hub 模型转换为 Caikit 格式。

3.1.4. 授权

您可以将 Authorino 添加为单一模型服务平台的授权提供程序。通过添加授权供应商，您可以为您在平台上部署的模型启用令牌授权，这样可确保只有授权方才能对模型发出请求。

要在单型号服务平台上将 Authorino 添加为授权提供程序，您可以以下选项：

- 如果集群中可以自动安装单模型服务平台，您可以在自动安装过程中包含 Authorino。
- 如果您需要手动安装单模式服务平台，还必须手动配置 Authorino。

有关为单模式服务平台选择安装选项的指导，请参阅 安装选项。

3.1.5. 监控

您可以为单型号服务平台配置监控，并使用 Prometheus 为每个预安装的 model-serving 运行时提取指标。

3.2. 配置 KSERVE 的自动安装

如果您还没有在 OpenShift 集群上创建 **ServiceMeshControlPlane** 或 **KNativeServing** 资源，您可以将 Red Hat OpenShift AI Operator 配置为安装 KServe 及其依赖项。

重要

如果您在集群中创建了 **ServiceMeshControlPlane** 或 **KNativeServing** 资源，Red Hat OpenShift AI Operator 无法安装 KServe 及其依赖项，且安装不会进行。在这种情况下，您必须按照手动安装说明来安装 KServe。

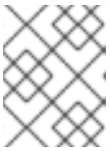
先决条件

- 有 OpenShift 集群的集群管理员特权。
- 您的集群有 4 个 CPU 和 16 GB 内存的节点。
- 您已下载并安装 OpenShift 命令行界面(CLI)。如需更多信息，请参阅安装 OpenShift CLI (Red Hat OpenShift Dedicated)或安装 OpenShift CLI (Red Hat OpenShift Service on AWS)。
- 已安装 Red Hat OpenShift Service Mesh Operator 和依赖的 Operator。

注意

要启用 KServe 的自动安装，*请只* 为 Red Hat OpenShift Service Mesh 安装所需的 Operator。不要执行任何其他配置或创建 **ServiceMeshControlPlane** 资源。

- 已安装 Red Hat OpenShift Serverless Operator。



注意

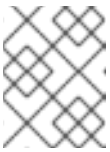
要启用 KServe 的自动安装，*请只* 安装 Red Hat OpenShift Serverless Operator。不要执行任何其他配置或创建 **KNativeServing** 资源。

- 要将 Authorino 添加为授权提供程序，以便您可以为部署模型启用令牌授权，您必须安装 **Red Hat - Authorino** Operator。请参阅安装 Authorino Operator。

流程

1. 以集群管理员身份登录 OpenShift Web 控制台。
2. 在 Web 控制台中，点 **Operators → Installed Operators**，然后点 Red Hat OpenShift AI Operator。
3. 安装 OpenShift Service Mesh，如下所示：
   a. 单击 **DSC 初始化** 选项卡。
   b. 点 **default-dsci** 对象。
   c. 点 **YAML** 标签。
   d. 在 **spec** 部分中，验证 **serviceMesh** 组件的 **managementState** 字段的值是否已设置为 **Managed**，如下所示：

```
spec:
 applicationsNamespace: redhat-ods-applications
 monitoring:
   managementState: Managed
   namespace: redhat-ods-monitoring
 serviceMesh:
   controlPlane:
     metricsCollection: Istio
     name: data-science-smcp
     namespace: istio-system
   managementState: Managed
```



注意

不要更改默认为 **serviceMesh** 组件指定的 **istio-system** 命名空间。不支持其他命名空间值。

   e. 点击 **Save**。
   根据您添加到 **DSCInitialization** 对象的配置，Red Hat OpenShift AI Operator 会安装 OpenShift Service Mesh。

4. （仅限 Red Hat OpenShift Service on AWS）：如果您的 OpenShift 集群在 AWS (ROSA)上运行，则需要额外的设置才能使服务网格 control plane 配置正常工作。要添加此设置，请编辑 **data-science-smcp** 服务网格 control plane 对象，如下所示：

   a. 在 Web 控制台中，点 **Operators → Installed Operators**，然后点 Red Hat OpenShift Service Mesh Operator。

   b. 点 **Istio Service Mesh Control Plane**标签页。

   c. 点 **data-science-smcp** 对象。

   d. 点 **YAML** 标签。

   e. 在 **spec.security.identity** 部分中，添加名为 的字段，并将值设为 **Third party**，如下所示。

```
security:
  dataPlane:
    mtls: true
  identity:
    type: ThirdParty
```

   f. **点击 Save。**

5. **安装 KServe 和 OpenShift Serverless，如下所示：**

   a. **在 Web 控制台中，点 Operators → Installed Operators，然后点 Red Hat OpenShift AI Operator。**

   b. **点 Data Science Cluster 选项卡。**

   c. 单击 **default-dsc DSC 对象。**

   d. **点 YAML 标签。**

   e. **在 spec.components 部分中，配置 kserve 组件，如下所示。**

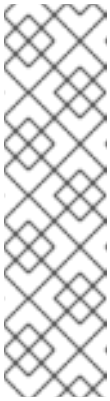
```
spec:
 components:
   kserve:
     managementState: Managed
     serving:
       ingressGateway:
```

```
    certificate:
      secretName: knative-serving-cert
      type: SelfSigned
  managementState: Managed
  name: knative-serving
```

f. 点击 **Save**。

上述配置为 **OpenShift Serverless** 创建一个入口网关，以接收来自 **OpenShift Service Mesh** 的流量。在这个配置中，观察以下详情：

- 显示的配置会生成自签名证书来保护到 **OpenShift** 集群的传入流量，并将证书存储在 **secretName** 字段中指定的 **knative-serving-cert secret** 中。要提供 ***您自己的证书***，请更新 **secretName** 字段的值，以指定您的 **secret** 名称，并将 **type** 字段的值更改为 **Provided**。



注意

如果提供自己的证书，证书必须指定 **OpenShift** 集群的 **ingress** 控制器使用的域名。您可以运行以下命令来检查这个值：

```
$ oc get ingresses.config.openshift.io cluster -o
jsonpath='{.spec.domain}'
```

- 对于 **kserve** 和 **serving** 组件，您必须将 **managementState** 字段的值设置为 **Managed**。将 **kserve.managementState** 设置为 **Managed** 会触发 **KServe** 的自动安装。将 **service.managementState** 设置为 **Managed** 触发器自动安装 **OpenShift Serverless**。但是，如果 **kserve.managementState** 没有设置为 **Managed**，则 ***不会触发*** **OpenShift Serverless** 的安装。

验证

- 验证 **OpenShift Service Mesh** 安装，如下所示：
  - 在 **web** 控制台中，点 **Workloads → Pods**。
  - 在项目列表中，选择 **istio-system**。这是安装 **OpenShift Service Mesh** 的项目。

- 确认已经为服务网格 control plane、入口网关和出口网关运行 pod。这些 pod 具有以下示例中显示的命名模式：

```
NAME                                  READY   STATUS    RESTARTS   AGE
istio-egressgateway-7c46668687-fzsqj   1/1     Running   0          22h
istio-ingressgateway-77f94d8f85-fhsp9  1/1     Running   0          22h
istiod-data-science-smcp-cc8cfd9b8-2rkg4   1/1     Running   0          22h
```

- 验证 OpenShift Serverless 的安装，如下所示：
  - 在 web 控制台中，点 Workloads → Pods。
  - 从项目列表中，选择 knative-serving。这是安装 OpenShift Serverless 的项目。
  - 确认 knative-serving 项目中有多个正在运行的 pod，包括 activator、自动扩展器、控制器和域映射 pod，以及 Knative Istio 控制器（控制 OpenShift Serverless 和 OpenShift Service Mesh 集成）的 pod。此时会显示一个示例。

```
NAME                                    READY   STATUS    RESTARTS  AGE
activator-7586f6f744-nvdlb              2/2     Running   0         22h
activator-7586f6f744-sd77w              2/2     Running   0         22h
autoscaler-764fdf5d45-p2v98             2/2     Running   0         22h
autoscaler-764fdf5d45-x7dc6             2/2     Running   0         22h
autoscaler-hpa-7c7c4cd96d-2lkzg         1/1     Running   0         22h
autoscaler-hpa-7c7c4cd96d-gks9j         1/1     Running   0         22h
controller-5fdfc9567c-6cj9d             1/1     Running   0         22h
controller-5fdfc9567c-bf5x7             1/1     Running   0         22h
domain-mapping-56ccd85968-2hjvp         1/1     Running   0         22h
domain-mapping-56ccd85968-lg6mw         1/1     Running   0         22h
domainmapping-webhook-769b88695c-gp2hk  1/1     Running   0         22h
domainmapping-webhook-769b88695c-npn8g  1/1     Running   0         22h
net-istio-controller-7dfc6f668c-jb4xk   1/1     Running   0         22h
net-istio-controller-7dfc6f668c-jxs5p   1/1     Running   0         22h
net-istio-webhook-66d8f75d6f-bgd5r      1/1     Running   0         22h
net-istio-webhook-66d8f75d6f-hld75      1/1     Running   0         22h
webhook-7d49878bc4-8xjbr                1/1     Running   0         22h
webhook-7d49878bc4-s4xx4                1/1     Running   0         22h
```

- 验证 KServe 的安装，如下所示：
  - 在 web 控制台中，点 Workloads → Pods。

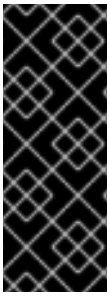
- 从项目列表中，选择 **redhat-ods-applications**。这是安装 **OpenShift AI** 组件的项目，包括 **KServe**。

- 确认项目包含 **KServe** 控制器管理器运行的 **pod**，如下例所示：

```
NAME                                        READY   STATUS    RESTARTS   AGE
kserve-controller-manager-7fbb7bccd4-t4c5g  1/1     Running   0          22h
odh-model-controller-6c4759cc9b-cftmk       1/1     Running   0          129m
odh-model-controller-6c4759cc9b-ngj8b       1/1     Running   0          129m
odh-model-controller-6c4759cc9b-vnhq5       1/1     Running   0          129m
```

3.3. 手动安装 KSERVE

如果您已经安装了 **Red Hat OpenShift Service Mesh Operator** 并创建了 **ServiceMeshControlPlane** 资源，或者已安装 **Red Hat OpenShift Serverless Operator** 并创建了 **KNativeServing** 资源，**Red Hat OpenShift AI Operator** 无法安装 **KServe** 及其依赖项。在这种情况下，您必须手动安装 **KServe**。



重要

本节中的步骤演示了如何执行 **KServe** 及其依赖项的新安装，并作为完整的安装和配置参考。*如果您已经安装并配置了 OpenShift Service Mesh 或 OpenShift Serverless，则可能不需要遵循所有步骤。如果您不确定要应用到现有配置的哪些更新以使用 KServe，请联系红帽支持。*

3.3.1. 安装 KServe 依赖项

在安装 KServe 前，您必须安装和配置一些依赖项。具体来说，您必须创建 Red Hat OpenShift Service Mesh 和 Knative Serving 实例，然后为 Knative Serving 配置安全网关。

3.3.1.1. 创建 OpenShift Service Mesh 实例

以下流程演示了如何创建 Red Hat OpenShift Service Mesh 实例。

先决条件

- *有 OpenShift 集群的集群管理员特权。*

- *您的集群有 4 个 CPU 和 16 GB 内存的节点。*

- 您已下载并安装 OpenShift 命令行界面(CLI)。请参阅安装 OpenShift CLI (Red Hat OpenShift Dedicated)或安装 OpenShift CLI (Red Hat OpenShift Service on AWS)。

- 已安装 Red Hat OpenShift Service Mesh Operator 和依赖的 Operator。

流程

1. 在终端窗口中，如果您还没有以集群管理员身份登录到 OpenShift 集群，请登录 OpenShift CLI，如下例所示：

```
$ oc login <openshift_cluster_url> -u <admin_username> -p <password>
```

2. 为 Red Hat OpenShift Service Mesh 创建所需的命名空间。

```
$ oc create ns istio-system
```

您会看到以下输出：

```
namespace/istio-system created
```

3. 在名为 smcp.yaml 的 YAML 文件中定义 ServiceMeshControlPlane 对象，其内容如下：

```
apiVersion: maistra.io/v2
kind: ServiceMeshControlPlane
metadata:
  name: minimal
  namespace: istio-system
spec:
  tracing:
    type: None
  addons:
    grafana:
      enabled: false
    kiali:
      name: kiali
      enabled: false
    prometheus:
      enabled: false
    jaeger:
      name: jaeger
  security:
    dataPlane:
      mtls: true
```

```
  identity:
    type: ThirdParty
 techPreview:
  meshConfig:
    defaultConfig:
      terminationDrainDuration: 35s
 gateways:
  ingress:
    service:
      metadata:
        labels:
          knative: ingressgateway
 proxy:
  networking:
    trafficControl:
      inbound:
        excludedPorts:
          - 8444
          - 8022
```

如需有关 YAML 文件中值的更多信息，请参阅 Service Mesh control plane 配置参考。

4. **创建服务网格 control plane。**

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

验证

- **验证服务网格实例的创建，如下所示：**
  - **在 OpenShift CLI 中输入以下命令：**

```
$ oc get pods -n istio-system
```

前面的命令列出了 istio-system 项目中所有正在运行的 pod。这是安装 OpenShift Service Mesh 的项目。

  - **确认已经为服务网格 control plane、入口网关和出口网关运行 pod。这些 pod 具有以下命名模式：**

```
NAME                                  READY   STATUS    RESTARTS   AGE
istio-egressgateway-7c46668687-fzsqj  1/1     Running   0          22h
istio-ingressgateway-77f94d8f85-fhsp9 1/1     Running   0          22h
```

```
istiod-data-science-smcp-cc8cfd9b8-2rkg4      1/1     Running   0          22h
```

3.3.1.2. 创建 Knative Serving 实例

以下流程演示了如何安装 Knative Serving，然后创建实例。

先决条件

- 有 OpenShift 集群的集群管理员特权。
- 您的集群有 4 个 CPU 和 16 GB 内存的节点。
- 您已下载并安装 OpenShift 命令行界面(CLI)。请参阅安装 OpenShift CLI (Red Hat OpenShift Dedicated)或安装 OpenShift CLI (Red Hat OpenShift Service on AWS)。
- 您已创建了 Red Hat OpenShift Service Mesh 实例。
- 已安装 Red Hat OpenShift Serverless Operator。

流程

1. 在终端窗口中，如果您还没有以集群管理员身份登录到 OpenShift 集群，请登录 OpenShift CLI，如下例所示：

```
$ oc login <openshift_cluster_url> -u <admin_username> -p <password>
```

2. 检查 Knative Serving 所需的项目（即 命名空间）是否已存在。

```
$ oc get ns knative-serving
```

如果项目存在，您会看到类似以下示例的输出：

```
NAME              STATUS   AGE
knative-serving   Active   4d20h
```

3. *如果 **knative-serving** 项目不存在，请创建它。*

```
$ oc create ns knative-serving
```

您会看到以下输出：

```
namespace/knative-serving created
```

4. *在名为 **default-smm.yaml** 的 **YAML** 文件中定义 **ServiceMeshMember** 对象，其内容如下：*

```
apiVersion: maistra.io/v1
kind: ServiceMeshMember
metadata:
  name: default
  namespace: knative-serving
spec:
  controlPlaneRef:
    namespace: istio-system
    name: minimal
```

5. *在 **istio-system** 命名空间中创建 **ServiceMeshMember** 对象。*

```
$ oc apply -f default-smm.yaml
```

您会看到以下输出：

```
servicemeshmember.maistra.io/default created
```

6. *在名为 **knativeserving-istio.yaml** 的 **YAML** 文件中定义 **KnativeServing** 对象，其内容如下：*

```
apiVersion: operator.knative.dev/v1beta1
kind: KnativeServing
metadata:
  name: knative-serving
  namespace: knative-serving
  annotations:
    serverless.openshift.io/default-enable-http2: "true"
spec:
  workloads:
    - name: net-istio-controller
```

```
    env:
      - container: controller
        envVars:
          - name: ENABLE_SECRET_INFORMER_FILTERING_BY_CERT_UID
            value: 'true'
  - annotations:
      sidecar.istio.io/inject: "true" 1
      sidecar.istio.io/rewriteAppHTTPProbers: "true" 2
    name: activator
  - annotations:
      sidecar.istio.io/inject: "true"
      sidecar.istio.io/rewriteAppHTTPProbers: "true"
    name: autoscaler
ingress:
  istio:
    enabled: true
config:
  features:
    kubernetes.podspec-affinity: enabled
    kubernetes.podspec-nodeselector: enabled
    kubernetes.podspec-tolerations: enabled
```

前面的文件为 KnativeServing 对象定义自定义资源(CR)。CR 还为每个激活器和自动扩展 pod 添加以下操作：

1

将 Istio sidecar 注入 pod。这使得 pod 成为服务网格的一部分。

2

启用 Istio sidecar 为 pod 重写 HTTP 存活度和就绪度探测。

注意

如果为 Knative 服务配置自定义域，您可以使用 TLS 证书来保护映射的服务。要做到这一点，您必须创建一个 TLS secret，然后更新 DomainMapping CR 以使用您创建的 TLS secret。如需更多信息，请参阅 Red Hat OpenShift Serverless 文档中的使用 TLS 证书保护映射的服务。

7. 在指定的 knative-serving 命名空间中创建 KnativeServing 对象。

```
$ oc apply -f kniveserving-istio.yaml
```

您会看到以下输出：

```
knativeserving.operator.knative.dev/knative-serving created
```

验证

- 查看 istio-system 命名空间中的默认 ServiceMeshMemberRoll 对象。

```
$ oc describe smmr default -n istio-system
```

在 ServiceMeshMemberRoll 对象的描述中，找到 Status.Members 字段，并确认它包含 knative-serving 命名空间。

- 验证 Knative Serving 实例的创建，如下所示：
  - 在 OpenShift CLI 中输入以下命令：

```
$ oc get pods -n knative-serving
```

前面的命令列出了 knative-serving 项目中所有正在运行的 pod。这是您在其中创建 Knative Serving 实例的项目。

  - 确认 knative-serving 项目中有多个正在运行的 pod，包括 activator、自动扩展器、控制器和域映射 pod，以及控制 OpenShift Serverless 和 OpenShift Service Mesh 的集成的 pod。此时会显示一个示例。

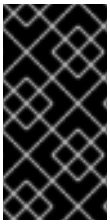
```
NAME                                READY   STATUS    RESTARTS   AGE
activator-7586f6f744-nvdlb          2/2     Running   0          22h
activator-7586f6f744-sd77w          2/2     Running   0          22h
autoscaler-764fdf5d45-p2v98         2/2     Running   0          22h
autoscaler-764fdf5d45-x7dc6         2/2     Running   0          22h
autoscaler-hpa-7c7c4cd96d-2lkzg     1/1     Running   0          22h
autoscaler-hpa-7c7c4cd96d-gks9j     1/1     Running   0          22h
controller-5fdfc9567c-6cj9d         1/1     Running   0          22h
controller-5fdfc9567c-bf5x7         1/1     Running   0          22h
domain-mapping-56ccd85968-2hjvp     1/1     Running   0          22h
domain-mapping-56ccd85968-lg6mw     1/1     Running   0          22h
domainmapping-webhook-769b88695c-gp2hk   1/1     Running   0          22h
domainmapping-webhook-769b88695c-npn8g   1/1     Running   0          22h
net-istio-controller-7dfc6f668c-jb4xk    1/1     Running   0          22h
net-istio-controller-7dfc6f668c-jxs5p    1/1     Running   0          22h
```

```
net-istio-webhook-66d8f75d6f-bgd5r      1/1     Running   0          22h
net-istio-webhook-66d8f75d6f-hld75      1/1     Running   0          22h
webhook-7d49878bc4-8xjbr                1/1     Running   0          22h
webhook-7d49878bc4-s4xx4                1/1     Running   0          22h
```

3.3.1.3. 为 Knative Serving 创建安全网关

要保护 Knative Serving 实例和服务网格之间的流量，您必须为 Knative Serving 实例创建安全网关。

以下流程演示了如何使用 OpenSSL 生成通配符证书和密钥，然后使用它们为 Knative Serving 创建本地和入口网关。



重要

如果您有自己的通配符证书和密钥在配置网关时指定，您可以跳过这个过程的第 11 步。

先决条件

- 有 OpenShift 集群的集群管理员特权。
- 您已下载并安装 OpenShift 命令行界面(CLI)。请参阅安装 OpenShift CLI (Red Hat OpenShift Dedicated)或安装 OpenShift CLI (Red Hat OpenShift Service on AWS)。
- 您已创建了 Red Hat OpenShift Service Mesh 实例。
- 您已创建了 Knative Serving 实例。
- 如果要生成通配符证书和密钥，您已 下载并安装 OpenSSL。

流程

1. 在终端窗口中，如果您还没有以集群管理员身份登录到 OpenShift 集群，请登录 OpenShift CLI，如下例所示：

```
$ oc login <openshift_cluster_url> -u <admin_username> -p <password>
```

重要

如果您有自己的通配符证书和密钥在配置网关时指定，请跳至此流程的第 11 步。

2. 设置环境变量，以定义用于为网关生成通配符证书和密钥的基础目录。

```
$ export BASE_DIR=/tmp/kserve
$ export BASE_CERT_DIR=${BASE_DIR}/certs
```

3. 设置环境变量，以定义 **OpenShift** 集群的入口控制器使用的通用名称。

```
$ export COMMON_NAME=$(oc get ingresses.config.openshift.io cluster -o
jsonpath='{.spec.domain}' | awk -F'.' '{print $(NF-1)"."$NF}')
```

4. 设置环境变量，以定义 **OpenShift** 集群的入口控制器使用的域名。

```
$ export DOMAIN_NAME=$(oc get ingresses.config.openshift.io cluster -o
jsonpath='{.spec.domain}')
```

5. 根据之前设置的环境变量，为证书生成创建所需的基础目录。

```
$ mkdir ${BASE_DIR}
$ mkdir ${BASE_CERT_DIR}
```

6. 创建用于生成通配符证书的 **OpenSSL** 配置。

```
$ cat <<EOF> ${BASE_DIR}/openssl-san.config
[ req ]
distinguished_name = req
[ san ]
subjectAltName = DNS:*.${DOMAIN_NAME}
EOF
```

7. 生成 **root** 证书。

```
$ openssl req -x509 -sha256 -nodes -days 3650 -newkey rsa:2048 \
-subj "/O=Example Inc./CN=${COMMON_NAME}" \
-keyout $BASE_DIR/root.key \
```

```
-out $BASE_DIR/root.crt
```

8. 生成由 root 证书签名的通配符证书。

```
$ openssl req -x509 -newkey rsa:2048 \
-sha256 -days 3560 -nodes \
-subj "/CN=${COMMON_NAME}/O=Example Inc." \
-extensions san -config ${BASE_DIR}/openssl-san.config \
-CA $BASE_DIR/root.crt \
-CAkey $BASE_DIR/root.key \
-keyout $BASE_DIR/wildcard.key  \
-out $BASE_DIR/wildcard.crt

$ openssl x509 -in ${BASE_DIR}/wildcard.crt -text
```

9. 验证通配符证书。

```
$ openssl verify -CAfile ${BASE_DIR}/root.crt ${BASE_DIR}/wildcard.crt
```

10. 将脚本创建的通配符密钥和证书导出到新环境变量。

```
$ export TARGET_CUSTOM_CERT=${BASE_CERT_DIR}/wildcard.crt
$ export TARGET_CUSTOM_KEY=${BASE_CERT_DIR}/wildcard.key
```

11. 可选：要将 自己的 通配符密钥和证书导出到新环境变量，请输入以下命令：

```
$ export TARGET_CUSTOM_CERT=<path_to_certificate>
$ export TARGET_CUSTOM_KEY=<path_to_key>
```

注意

在您提供的证书中，您必须指定 OpenShift 集群的 ingress 控制器使用的域名。您可以运行以下命令来检查这个值：

```
$ oc get ingresses.config.openshift.io cluster -o
jsonpath='{.spec.domain}'
```

12. 使用您为通配符证书和密钥设置的环境变量在 istio-system 命名空间中创建 TLS secret。

```
$ oc create secret tls wildcard-certs --cert=${TARGET_CUSTOM_CERT} --
key=${TARGET_CUSTOM_KEY} -n istio-system
```

13. **使用以下内容创建一个 gateways.yaml YAML 文件：**

```
apiVersion: v1
kind: Service (1)
metadata:
  labels:
    experimental.istio.io/disable-gateway-port-translation: "true"
  name: knative-local-gateway
  namespace: istio-system
spec:
  ports:
    - name: http2
      port: 80
      protocol: TCP
      targetPort: 8081
  selector:
    knative: ingressgateway
  type: ClusterIP
---
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: knative-ingress-gateway (2)
  namespace: knative-serving
spec:
  selector:
    knative: ingressgateway
  servers:
    - hosts:
        - '*'
      port:
        name: https
        number: 443
        protocol: HTTPS
      tls:
        credentialName: wildcard-certs
        mode: SIMPLE
---
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
 name: knative-local-gateway (3)
 namespace: knative-serving
spec:
 selector:
   knative: ingressgateway
 servers:
   - port:
       number: 8081
       name: https
```

```
    protocol: HTTPS
  tls:
    mode: ISTIO_MUTUAL
  hosts:
    - "*"
```

1 为 **Knative 本地网关在 istio-system 命名空间中定义一个服务。**

2 **在 knative-serving 命名空间中 定义一个入口网关。网关使用您此流程前面创建的 TLS secret。入口网关处理到 Knative 的外部流量。**

3 **在 knative-serving 命名空间中为 Knative 定义本地网关。**

14. *应用 **gateways.yaml** 文件以创建定义的资源。*

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

您会看到以下输出：

```
service/knative-local-gateway created
gateway.networking.istio.io/knative-ingress-gateway created
gateway.networking.istio.io/knative-local-gateway created
```

验证

- *查看您创建的网关。*

```
$ oc get gateway --all-namespaces
```

确认您看到在 knative-serving 命名空间中创建的本地和入口网关，如下例所示：

```
NAMESPACE          NAME                      AGE
knative-serving    knative-ingress-gateway   69s
knative-serving    knative-local-gateway     2m
```

3.3.2. 安装 KServe

要完成 KServe 的手动安装，您必须安装 Red Hat OpenShift AI Add-on，它会安装 Red Hat OpenShift AI Operator。然后，您可以使用 Operator 安装 KServe。

先决条件

- *有 OpenShift 集群的集群管理员特权。*
- *您的集群有 4 个 CPU 和 16 GB 内存的节点。*
- *您已下载并安装 OpenShift 命令行界面(CLI)。请参阅安装 OpenShift CLI (Red Hat OpenShift Dedicated)或安装 OpenShift CLI (Red Hat OpenShift Service on AWS)。*
- *您已创建了 Red Hat OpenShift Service Mesh 实例。*
- *您已创建了 Knative Serving 实例。*
- *您已为 Knative Serving 创建安全网关。*
- *您已在 OpenShift 集群中安装了 Red Hat OpenShift AI Add-on。这会安装 Red Hat OpenShift AI Operator 并创建一个默认的 DataScienceCluster 对象。*

流程

1. *以集群管理员身份登录 OpenShift Web 控制台。*
2. *在 Web 控制台中，点 Operators → Installed Operators，然后点 Red Hat OpenShift AI Operator。*
3. *要安装 KServe，请配置 OpenShift Service Mesh 组件，如下所示：*
    a. *单击 DSC 初始化 选项卡。*

b. 点 default-dsci 对象。

c. 点 YAML 标签。

d. 在 spec 部分，添加并配置 serviceMesh 组件，如下所示：

```
spec:
 serviceMesh:
  managementState: Unmanaged
```

e. 点击 Save。

4. 要安装 KServe，请配置 KServe 和 OpenShift Serverless 组件，如下所示：

a. 在 Web 控制台中，点 Operators → Installed Operators，然后点 Red Hat OpenShift AI Operator。

b. 点 Data Science Cluster 选项卡。

c. 单击 default-dsc DSC 对象。

d. 点 YAML 标签。

e. 在 spec.components 部分中，配置 kserve 组件，如下所示：

```
spec:
 components:
  kserve:
   managementState: Managed
```

f. 在 kserve 组件中，添加 serving 组件并进行配置，如下所示：

```
spec:
 components:
```

```
kserve:
  managementState: Managed
  serving:
    managementState: Unmanaged
```

g. 点击 **Save**。

3.3.3. 手动添加授权供应商

您可以将 Authorino 添加为单一模型服务平台的授权提供程序。通过添加授权供应商，您可以为您在平台上部署的模型启用令牌授权，这样可确保只有授权方才能对模型发出请求。

要手动添加 Authorino 作为授权供应商，您必须安装 Red Hat - Authorino Operator，创建一个 Authorino 实例，然后配置 OpenShift Service Mesh 和 KServe 组件以使用实例。

重要

要手动添加授权供应商，您必须对 OpenShift Service Mesh 实例进行配置更新。为确保 OpenShift Service Mesh 实例处于支持状态，请只 进行本节中显示的更新。

先决条件

- 您已查看了将 Authorino 添加为授权提供程序的选项，并根据适当的选项识别手动安装。请参阅 添加授权提供程序。
- 您已手动安装 KServe 及其依赖项，包括 OpenShift Service Mesh。请参阅 手动安装 KServe。
- 手动安装 KServe 时，您可以将 serviceMesh 组件的 managementState 字段的值设置为 Unmanaged。手动添加 Authorino 需要此设置。请参阅安装 KServe。

3.3.3.1. 安装 Red Hat Authorino Operator

在将 Autorino 添加为授权供应商前，您必须在 OpenShift 集群上安装 Red Hat - Authorino Operator。

先决条件

- 有 OpenShift 集群的集群管理员特权。

流程

1. 以集群管理员身份登录 OpenShift Web 控制台。
2. 在 Web 控制台中，点 Operators → OperatorHub。
3. 在 OperatorHub 页面中，在 Filter by keyword 字段中输入 Red Hat - Authorino。
4. 点 Red Hat - Authorino Operator。
5. 在 Red Hat - Authorino Operator 页面中，查看 Operator 信息，然后点 Install。
6. 在 Install Operator 页面中，保留 Update channel,Version,Installation mode,Installed Namespace 和 Update Approval 的默认值。
7. 点 Install。

验证

- 在 OpenShift Web 控制台中，点 Operators → Installed Operators，并确认 Red Hat - Authorino Operator 显示了以下状态之一：
  - Installing - 安装正在进行中；等待它变为 Succeeded。这可能需要几分钟。
  - Succeeded - 安装成功。

3.3.3.2. 创建 Authorino 实例

在 OpenShift 集群上安装 Red Hat - Authorino Operator 时，您必须创建一个 Authorino 实例。

先决条件

先决条件

- 已安装 Red Hat - Authorino Operator。
- 您有将资源添加到创建 OpenShift Service Mesh 实例的项目中的权限。请参阅创建 OpenShift Service Mesh 实例。

  如需有关 OpenShift 权限的更多信息，请参阅使用 RBAC 定义并应用权限 (Red Hat OpenShift Dedicated)或使用 RBAC 定义并应用权限 (Red Hat OpenShift Service on AWS)。

流程

1. 打开一个新的终端窗口。
2. 登录到 OpenShift 命令行界面(CLI)，如下所示：

```
$ oc login <openshift_cluster_url> -u <username> -p <password>
```

3. 创建命名空间来安装 Authorino 实例。

```
$ oc new-project <namespace_for_authorino_instance>
```

注意

自动安装过程为 Authorino 实例创建一个名为 redhat-ods-applications-auth-provider 的命名空间。考虑将相同的命名空间名称用于手动安装。

4. 要在现有 OpenShift Service Mesh 实例中为 Authorino 实例注册新命名空间，请创建一个包含以下内容的新 YAML 文件：

```
apiVersion: maistra.io/v1
kind: ServiceMeshMember
metadata:
  name: default
  namespace: <namespace_for_authorino_instance>
spec:
  controlPlaneRef:
    namespace: <namespace_for_service_mesh_instance>
    name: <name_of_service_mesh_instance>
```

5. 保存 YAML 文件。

6. 在集群中创建 ServiceMeshMember 资源。

```
$ oc create -f <file_name>.yaml
```

7. 要配置 Authorino 实例，请创建一个新的 YAML 文件，如下例所示：

```
apiVersion: operator.authorino.kuadrant.io/v1beta1
kind: Authorino
metadata:
  name: authorino
  namespace: <namespace_for_authorino_instance>
spec:
  authConfigLabelSelectors: security.opendatahub.io/authorization-group=default
  clusterWide: true
  listener:
    tls:
      enabled: false
  oidcServer:
    tls:
      enabled: false
```

8. 保存 YAML 文件。

9. 在集群中创建 Authorino 资源。

```
$ oc create -f <file_name>.yaml
```

10. 修补 Authorino 部署来注入 Istio sidecar，这使得 Authorino 实例成为 OpenShift Service Mesh 实例的一部分。

```
$ oc patch deployment <name_of_authorino_instance> -n
<namespace_for_authorino_instance> -p '{"spec": {"template":{"metadata":{"labels":
{"sidecar.istio.io/inject":"true"}}}} }'
```

验证

- 确认 Authorino 实例正在运行，如下所示：

1. 检查在您为 Authorino 实例创建的命名空间中运行的 pod （和容器），如下例所示：

```
$ oc get pods -n redhat-ods-applications-auth-provider -o="custom-columns=NAME:.metadata.name,STATUS:.status.phase,CONTAINERS:.spec.containers[*].name"
```

2. 确认输出类似以下示例：

```
NAME                        STATUS    CONTAINERS
authorino-6bc64bd667-kn28z  Running   authorino,istio-proxy
```

如示例所示，为 Authorino 实例有一个正在运行的 pod。pod 具有 Authorino 和您注入的 Istio sidecar 的容器。

3.3.3.3. 将 OpenShift Service Mesh 实例配置为使用 Authorino

创建 Authorino 实例时，您必须将 OpenShift Service Mesh 实例配置为使用 Authorino 作为授权供应商。

重要

为确保 OpenShift Service Mesh 实例处于支持状态，请只 进行以下流程中显示的配置更新。

先决条件

- 您已创建了 Authorino 实例，并在 OpenShift Service Mesh 实例中为 Authorino 实例注册命名空间。
- 有修改 OpenShift Service Mesh 实例的权限。请参阅创建 OpenShift Service Mesh 实例。

流程

1. 在终端窗口中，如果您还没有以具有更新 OpenShift Service Mesh 实例的权限的用户登录到 OpenShift 集群，请登录 OpenShift CLI，如下例所示：

```
$ oc login <openshift_cluster_url> -u <username> -p <password>
```

2. 创建包含以下内容的 YAML 文件：

```
spec:
 techPreview:
  meshConfig:
   extensionProviders:
   - name: redhat-ods-applications-auth-provider
    envoyExtAuthzGrpc:
     service: <name_of_authorino_instance>-authorino-
authorization.<namespace_for_authorino_instance>.svc.cluster.local
     port: 50051
```

3. 保存 YAML 文件。

4. 使用 oc patch 命令将 YAML 文件应用到 OpenShift Service Mesh 实例。

```
$ oc patch smcp <name_of_service_mesh_instance> --type merge -n
<namespace_for_service_mesh_instance> --patch-file <file_name>.yaml
```

重要

只有在 OpenShift Service Mesh 实例中尚未指定其他扩展供应商时，才能应用显示为补丁的配置。如果您已经指定了其他扩展供应商，您必须手动编辑 ServiceMeshControlPlane 资源以添加配置。

验证

- 验证您的 Authorino 实例是否已作为扩展供应商添加到 OpenShift Service Mesh 配置中，如下所示：

1. 检查 OpenShift Service Mesh 实例的 ConfigMap 对象：

```
$ oc get configmap istio-<name_of_service_mesh_instance> -n
<namespace_for_service_mesh_instance> --output=jsonpath={.data.mesh}
```

2. 确认您看到的输出类似以下示例，这表明 Authorino 实例已成功添加为扩展提供程序。

```
defaultConfig:
  discoveryAddress: istiod-data-science-smcp.istio-system.svc:15012
```

```
  proxyMetadata:
    ISTIO_META_DNS_AUTO_ALLOCATE: "true"
    ISTIO_META_DNS_CAPTURE: "true"
    PROXY_XDS_VIA_AGENT: "true"
  terminationDrainDuration: 35s
  tracing: {}
dnsRefreshRate: 300s
enablePrometheusMerge: true
extensionProviders:
- envoyExtAuthzGrpc:
    port: 50051
    service: authorino-authorino-authorization.opendatahub-auth-provider.svc.cluster.local
  name: opendatahub-auth-provider
ingressControllerMode: "OFF"
rootNamespace: istio-system
trustDomain: null%
```

3.3.3.4. 为 KServe 配置授权

要将单型号服务平台配置为使用 Authorino，您必须创建一个全局 AuthorizationPolicy 资源，该资源应用于部署模型时创建的 KServe predictor pod。另外，要考虑对模型发出请求时发生的多个网络跃点，您必须创建一个 EnvoyFilter 资源，该资源会持续将 HTTP 主机标头重置为最初包含在请求中的。

先决条件

- 您已创建了 Authorino 实例，并将 OpenShift Service Mesh 配置为使用它。
- 您有更新集群中的 KServe 部署的权限。
- 您有将资源添加到创建 OpenShift Service Mesh 实例的项目中的权限。请参阅创建 OpenShift Service Mesh 实例。

流程

1. 在终端窗口中，如果您还没有以具有更新 KServe 部署权限的用户身份登录 OpenShift 集群，请登录 OpenShift CLI，如下例所示：

```
$ oc login <openshift_cluster_url> -u <username> -p <password>
```

2. 创建包含以下内容的 YAML 文件：

```
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
```

```
metadata:
  name: kserve-predictor
spec:
  action: CUSTOM
  provider:
    name: redhat-ods-applications-auth-provider 1
  rules:
    - to:
        - operation:
            notPaths:
              - /healthz
              - /debug/pprof/
              - /metrics
              - /wait-for-drain
  selector:
    matchLabels:
      component: predictor
```

1 *您指定的名称必须与添加到 OpenShift Service Mesh 实例的扩展供应商的名称匹配。*

3. *保存 YAML 文件。*

4. *在命名空间中为 OpenShift Service Mesh 实例创建 AuthorizationPolicy 资源。*

```
$ oc create -n <namespace_for_service_mesh_instance> -f <file_name>.yaml
```

5. *创建包含以下内容的另一个新 YAML 文件：*

```
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: activator-host-header
spec:
  priority: 20
  workloadSelector:
    labels:
      component: predictor
  configPatches:
  - applyTo: HTTP_FILTER
    match:
      listener:
        filterChain:
          filter:
            name: envoy.filters.network.http_connection_manager
    patch:
```

```
    operation: INSERT_BEFORE
    value:
      name: envoy.filters.http.lua
      typed_config:
        '@type': type.googleapis.com/envoy.extensions.filters.http.lua.v3.Lua
        inlineCode: |
          function envoy_on_request(request_handle)
            local headers = request_handle:headers()
            if not headers then
              return
            end
            local original_host = headers:get("k-original-host")
            if original_host then
              port_seperator = string.find(original_host, ":", 7)
              if port_seperator then
                original_host = string.sub(original_host, 0, port_seperator-1)
              end
              headers:replace('host', original_host)
            end
          end
```

*显示的 **EnvoyFilter** 资源持续将 **HTTP** 主机标头重置为最初包含在任何 **inference** 请求中的标头。*

6. *在 **OpenShift Service Mesh** 实例的命名空间中创建 **EnvoyFilter** 资源。*

```
$ oc create -n <namespace_for_service_mesh_instance> -f <file_name>.yaml
```

验证

- *检查 **AuthorizationPolicy** 资源是否已成功创建。*

```
$ oc get authorizationpolicies -n <namespace_for_service_mesh_instance>
```

确认您会看到类似以下示例的输出：

```
NAME              AGE
kserve-predictor   28h
```

- *检查 **EnvoyFilter** 资源是否已成功创建。*

```
$ oc get envoyfilter -n <namespace_for_service_mesh_instance>
```

确认您会看到类似以下示例的输出：

```
NAME                          AGE
activator-host-header         28h
```

3.4. 为单型号服务平台添加授权供应商

您可以将 Authorino 添加为单一模型服务平台的授权提供程序。通过添加授权供应商，您可以为您在平台上部署的模型启用令牌授权，这样可确保只有授权方才能对模型发出请求。

要将 Authorino 添加为授权提供程序的方法取决于您如何安装单模式服务平台。如下为平台安装选项：

自动化安装

如果您还没有在 OpenShift 集群上创建 ServiceMeshControlPlane 或 KNativeServing 资源，您可以将 Red Hat OpenShift AI Operator 配置为安装 KServe 及其依赖项。您可以将 Authorino 作为自动安装过程的一部分。

有关自动安装的更多信息，包括 Authorino，请参阅配置 KServe 的自动安装。

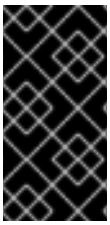
手动安装

如果您已在 OpenShift 集群上创建了 ServiceMeshControlPlane 或 KNativeServing 资源，则无法 配置 Red Hat OpenShift AI Operator 来安装 KServe 及其依赖项。在这种情况下，您必须手动安装 KServe。您还必须手动配置 Authorino。

有关手动安装的更多信息，包括 Authorino，请参阅 手动安装 KServe。

3.5. 使用单模式服务平台部署模型

在单模式服务平台上，每个模型都部署在自己的模型服务器上。这有助于您部署、监控、扩展和维护需要增加资源的大型模型。



重要

如果要使用单模式服务平台从使用自签名 SSL 证书的 S3 兼容存储部署模型，您必须在 OpenShift 集群上安装证书颁发机构(CA)捆绑包。如需更多信息，请参阅使用证书。

3.5.1. 启用单模式服务平台

安装 KServe 后，您可以使用 Red Hat OpenShift AI 仪表板启用单模式服务平台。您还可以使用控制面板为平台启用模型服务运行时。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的 admin 组的一部分（例如，rhoai-admins）。
- 您已安装了 KServe。
- 您的集群管理员 没有 编辑 OpenShift AI 仪表板配置，以禁用选择单一模型服务平台的功能，该平台使用 KServe 组件。如需更多信息，请参阅 控制面板配置选项。

流程

1. 启用单模式服务平台，如下所示：
   a. 在左侧菜单中，点击 Settings → Cluster settings。
   b. 找到 Model serving platform 部分。
   c. 要为项目启用单模式服务平台，请选择 Single-model serving 平台 复选框。
   d. 点 Save Changes。
2. 为单型号服务平台启用预安装的运行时，如下所示：
   a. 在 OpenShift AI 仪表板的左侧菜单中，点 Settings → Serving runtime。

Serving 运行时 页面显示您添加的任何自定义运行时，以及以下预安装的运行时：

- Caikit TGIS ServingRuntime for KServe
- OpenVINO Model Server
- TGIS 独立 ServingRuntime for KServe
- vLLM ServingRuntime for KServe

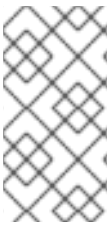
b. 设置您要用来 启用 的运行时。

单模式服务平台现在可用于模型部署。

3.5.2. 为单型号服务平台添加自定义模型运行时

模型运行时增加了对指定模型框架和这些框架支持的模型格式的支持。您可以使用 OpenShift AI 中包含的 预安装运行时。如果默认运行时不满足您的需要，您还可以添加自己的自定义运行时。例如，如果 TGIS 运行时不支持 Hugging Face Text Generation Inference (TGI) 支持的模型格式，您可以创建自定义运行时来添加对模型的支持。

作为管理员，您可以使用 OpenShift AI 接口添加和启用自定义模型运行时。然后，当您在单型号服务平台上部署模型时，您可以选择自定义运行时。



注意

OpenShift AI 允许您添加自己的自定义运行时，但不支持运行时本身。您需要正确配置和维护自定义运行时。您还负责确保您获得许可，以使用您添加的任何自定义运行时。

先决条件

- 您已以管理员身份登录到 OpenShift AI。

- 您已构建了自定义运行时，并将镜像添加到容器镜像存储库中，如 Quay。

流程

1. 在 OpenShift AI 仪表板中点 Settings > Serving runtime。

   Serving 运行时 页面将打开，并显示已安装和启用的 model-serving 运行时。

2. 要添加自定义运行时，请选择以下选项之一：

   - 要使用现有运行时（例如，TGIS Standalone ServingRuntime for KServe）启动，请点击现有运行时旁的操作菜单(uildDefaults)，然后点 Duplicate。
   - 要添加新的自定义运行时，请单击 Add serving runtime。

3. 在 Select the model serving Platform this runtime support 列表中，选择 Single-model serving platform。

4. 在 Select the API 协议此运行时支持列表，选择 REST 或 gRPC。

5. 可选：如果您启动新的运行时（而不是复制现有运行时），请选择以下选项之一来添加代码：

   - 上传 YAML 文件

     a. 点 Upload files。

     b. 在文件浏览器中，选择计算机上的 YAML 文件。

     嵌入的 YAML 编辑器将打开，并显示您上传的文件内容。

- 在编辑器中直接输入 YAML 代码

a. 点 Start from scratch。

b. 在嵌入式编辑器中直接输入或粘贴 YAML 代码。

注意

在很多情况下，创建自定义运行时需要将新的或自定义参数添加到 ServingRuntime 规格的 env 部分。

6. 点击 Add。

Serving 运行时页面将打开，并显示所安装的运行时的更新列表。观察您添加的自定义运行时会自动启用。显示您在创建运行时时指定的 API 协议。

7. 可选： 要编辑自定义运行时，点操作菜单（需要）并选择 Edit。

验证

- 您添加的自定义 model-serving 运行时会处于 Serving 运行时 页面中的启用状态。

3.5.3. 在单型号服务平台上部署模型

当您启用了单型号服务平台后，您可以启用预安装或自定义模型运行时，并开始在平台上部署模型。

注意

文本 Generation Inference Server (TGIS) 基于 Hugging Face TGI 的早期分叉。红帽将继续开发独立 TGIS 运行时来支持 TGI 模型。如果模型无法在 OpenShift AI 的当前版本中工作，则可能会在以后的版本中添加支持。同时，您还可以添加自己的自定义运行时来支持 TGI 模型。如需更多信息，请参阅为单模式服务平台添加自定义模型运行时。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分（例如，rh oai-users 或 rhoai-admins）。
- 您已安装了 KServe。
- 您已启用了单模式服务平台。
- 您已创建了数据科学项目。
- 您可以访问 S3 兼容对象存储。
- 对于您要部署的模型，您知道 S3 兼容对象存储存储桶中的关联文件夹路径。
- 要使用 Caikit-TGIS 运行时，已将模型转换为 Caikit 格式。例如，请参阅 cai kit-tgis-serving 存储库中的将 Hugging Face Hub 模型转换为 Caikit 格式。
- 如果要将图形处理单元(GPU)与模型服务器搭配使用，在 OpenShift AI 中启用了 GPU 支持。请参阅 OpenShift AI 中的启用 GPU 支持。
- 要使用 vLLM 运行时，已在 OpenShift AI 中启用 GPU 支持，并在集群中安装并配置了 Node Feature Discovery operator。如需更多信息，请参阅 在 OpenShift AI 中安装 Node Feature Discovery operator和启用 GPU 支持

流程

1. 在左侧菜单中，单击 Data Science Projects。

   Data Science Projects 页面将打开。

2. 点击您要在其中部署模型的项目的名称。

此时会打开项目详情页面。

3. 点 Models 选项卡。

4. 执行以下操作之一：

   - 如果您看到 Single-model 服务平台 标题，点有关标题的 Deploy model。
   - 如果没有看到任何标题，请点击 Deploy model 按钮。

   此时会打开 Deploy model 对话框。

5. 在 Model name 字段中，输入您要部署的模型的唯一名称。

6. 在 Serving runtime 字段中，选择一个启用的运行时。

7. 从 Model 框架 列表中，选择一个值。

8. 在 Number of model replicas to deploy 字段中，指定一个值。

9. 从 Model server size 列表中选择一个值。

10. 要为部署的模型要求令牌授权，请执行以下操作：

    a. 选择 Require token authorization。

    b. 在 Service account name 字段中，输入令牌要为其生成的服务帐户名称。

11.

11. 要指定模型的位置，请执行以下操作之一：

- 使用现有数据连接

a. 选择 **Existing data connection**。

b. 在 **Name** 列表中选择您之前定义的数据连接。

c. 在 **Path** 字段中，输入包含指定数据源中的模型的文件夹路径。

重要

OpenVINO Model Server 运行时对如何指定模型路径有具体要求。如需更多信息，请参阅 OpenShift AI 发行注记中的已知问题 RHOAIENG-3025。

- 使用新数据连接

a. 要定义模型可以访问的新数据连接，请选择 **New data connection**。

b. 在 **Name** 字段中输入数据连接的唯一名称。

c. 在 **Access key** 字段中，输入 S3 兼容对象存储供应商的访问密钥 ID。

d. 在 **Secret key** 字段中，为您指定的 S3 兼容对象存储帐户输入 secret 访问密钥。

e. 在 **Endpoint** 字段中，输入 S3 兼容对象存储桶的端点。

f. 在 **Region** 字段中，输入 S3 兼容对象存储帐户的默认区域。

g. 在 Bucket 字段中，输入 S3 兼容对象存储桶的名称。

h. 在 Path 字段中，在 S3 兼容对象存储中输入包含您的数据文件的文件夹路径。

重要

OpenVINO Model Server 运行时对如何指定模型路径有具体要求。如需更多信息，请参阅 OpenShift AI 发行注记中的已知问题 RHOAIENG-3025。

12. 点 Deploy。

验证

- 确认部署的模型显示在项目的 Models 选项卡中，并在仪表板的 Model Serving 页面中显示 Status 列中的 checkmark。

3.6. 对部署在单型号服务平台上的模型发出请求

当您使用单模式服务平台部署模型时，模型可作为服务使用 API 请求访问。这可让您根据数据输入返回预测。要使用 API 请求与部署的模型交互，您必须知道模型的 inference 端点。

另外，如果您通过启用令牌授权来保护 inference 端点，您必须了解如何访问授权令牌，以便在您的请求中指定它。

3.6.1. 访问已部署模型的授权令牌

如果通过启用令牌授权来保护模型 inference 端点，您必须了解如何访问授权令牌，以便在您的请求中指定它。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分

（例如，rh oai-users 或 rhoai-admins）。

- 已使用单模式服务平台部署了模型。

流程

1. 在 OpenShift AI 仪表板中，单击 Data Science Projects。

   Data Science Projects 页面将打开。

2. 点包含部署模型的项目的名称。

   此时会打开项目详情页面。

3. 点 Models 选项卡。

4. 在 Models and model servers 列表中，展开您的模型部分。

   您的授权令牌在 Token secret 字段中的 Token authorization 部分中显示。

5. 可选： 要复制用于 inference 请求的授权令牌，请点令牌值旁的 Copy 按钮(
   
)。

3.6.2. 访问已部署模型的 inference 端点

要对部署的模型发出请求，您必须了解如何访问可用的 inference 端点。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的用户组或 admin 组的一部分

（例如，rh oai-users 或 rhoai-admins）。

- *已使用单模式服务平台部署了模型。*
- *如果为部署的模型启用了令牌授权，则具有关联的令牌值。*

流程

1. *在 OpenShift AI 仪表板中，点 Model Serving。*

   模型的 inference 端点显示在 Inference 端点 字段中。

2. *根据您要对模型执行的操作（以及如果模型支持该操作），复制 inference 端点，然后将以下路径之一添加到 URL 的末尾：*

   Caikit TGIS ServingRuntime for KServe

   - *:443/api/v1/task/text-generation*
   - *:443/api/v1/task/server-streaming-text-generation*

   TGIS 独立 ServingRuntime for KServe

   - *:443 fmaas.GenerationService/Generate*
   - *:443 fmaas.GenerationService/GenerateStream*

注意

要查询 TGIS 独立运行时的端点，还必须在 Open Data Hub 文本 generation-inference 存储库的 proto 目录中下载文件。

OpenVINO Model Server

- */v2/models/<model-name>/infer*

vLLM ServingRuntime for KServe

- *:443/version*
- *:443/docs*
- *:443/v1/models*
- *:443/v1/chat/completions*
- *:443/v1/completions*
- *:443/v1/embeddings*

注意

vLLM 运行时与 OpenAI REST API 兼容。有关 vLLM 运行时支持的模型列表，请参阅支持的模型。

注意

要在 vLLM 中使用 embeddings inference 端点，您必须使用 vLLM 支持的嵌入式模型。您不能将 embeddings 端点与 generative 模型搭配使用。如需更多信息，请参阅 vLLM 中支持的嵌入模型。

如所示的路径所示，单模式服务平台使用 OpenShift 路由器的 HTTPS 端口（通常是端口 443）来提供外部 API 请求。

3. 使用端点向部署的模型发出 API 请求，如下例所示。

Caikit TGIS ServingRuntime for KServe

```
curl --json '{"model_id": "<model_name>", "inputs": "<text>"}'
https://<inference_endpoint_url>:443/api/v1/task/server-streaming-text-generation -H
'Authorization: Bearer <token>' (1)
```

(1) 只有在 部署模型时启用了令牌授权时，才必须添加 Authorization 标头并指定令牌值。

TGIS 独立 ServingRuntime for KServe

```
grpcurl -proto text-generation-inference/proto/generation.proto -d '{"requests": [{"text":"
<text>"}]}' -H 'Authorization: Bearer <token>' -insecure <inference_endpoint_url>:443
fmaas.GenerationService/Generate (1)
```

(1) 只有在 部署模型时启用了令牌授权时，才必须添加 Authorization 标头并指定令牌值。

OpenVINO Model Server

```
curl -ks <inference_endpoint_url>/v2/models/<model_name>/infer -d '{ "model_name": "
<model_name>", "inputs": [{ "name": "<name_of_model_input>", "shape": [<shape>],
"datatype": "<data_type>", "data": [<data>] }]}' -H 'Authorization: Bearer <token>' (1)
```

(1) 只有在 部署模型时启用了令牌授权时，才必须添加 Authorization 标头并指定令牌值。

vLLM ServingRuntime for KServe

```
curl -v https://<inference_endpoint_url>:443/v1/chat/completions -H "Content-Type:
application/json" -d '{ "messages": [{ "role": "<role>", "content": "<content>" }] -H
'Authorization: Bearer <token>' (1)
```

-

1

只有在 部署模型时启用了令牌授权时，才必须添加 Authorization 标头并指定令牌值。

其他资源

- 文本 Generation Inference Server (TGIS)
- Caikit API 文档
- OpenVINO KServe-compatible REST API 文档
- OpenAI API 文档

3.7. 查看用于单型号服务平台的模型运行时指标

当集群管理员为单型号服务平台配置了监控时，非管理员用户可以使用 OpenShift Web 控制台查看 KServe 组件的模型运行时指标。

先决条件

- 您可以使用开发人员或具有查看指标的项目查看权限的用户访问 OpenShift 集群。
- 熟悉在用户定义的项目中查询指标。请参阅 以开发者(Red Hat OpenShift Dedicated)身份查询用户定义的项目的指标，或以开发者(Red Hat OpenShift Service on AWS)身份 查询用户定义的项目的指标。

流程

1. 登录 OpenShift Web 控制台。
2. 切换到 Developer 视角。

3. 在左侧菜单中点 Observe。

4. 如 查询用户定义的项目的指标 中所述，作为开发者 (Red Hat OpenShift Dedicated)或 查询用户定义的项目的指标 (Red Hat OpenShift Service on AWS)，使用 Web 控制台运行对 caikit swig、tgioverlayfs、ovms swig 和 vllm:* model-serving 运行时指标的查询。您还可以对与 OpenShift Service Mesh 相关的 istio swig 指标运行查询。显示了一些示例。

   a. 以下查询显示了使用 vLLM 运行时部署模型的时间段内成功推测请求数：

   ```
   sum(increase(vllm:request_success_total{namespace=${namespace},model_name=${model_name}}[${rate_interval}]))
   ```

   b. 以下查询显示了使用独立 TGIS 运行时部署的模型的成功推测请求数：

   ```
   sum(increase(tgi_request_success{namespace=${namespace}, pod=~${model_name}-predictor-.*}[${rate_interval}]))
   ```

   c. 以下查询显示了使用 OpenVINO Model Server 运行时部署模型的每次时间成功推测请求数：

   ```
   sum(increase(ovms_requests_success{namespace=${namespace},name=${model_name}}[${rate_interval}]))
   ```

其他资源

- OVMS 指标
- TGIS 指标
- vLLM 指标

3.8. 在单型号服务平台上进行性能调优

某些性能问题可能需要您调整 inference 服务的参数或模型运行时。

3.8.1. 解决 CUDA 内存不足错误

在某些情况下，根据所使用的模型和硬件加速器，TGIS 内存自动调整算法可能会降低处理长序列所需的 GPU 内存量。此错误计算可能会导致来自模型服务器的 Compute Unified Architecture (CUDA)内存不足(OOM)错误响应。在这种情况下，您必须更新或添加 TGIS 模型-serving 运行时中的其他参数，如以下步骤所述。

先决条件

- 您已登陆到 Red Hat OpenShift AI。
- 如果您使用专用的 OpenShift AI 组，则作为 OpenShift 中的 admin 组的一部分（例如，rhoai-admins）。

流程

1. 在 OpenShift AI 仪表板中点 Settings > Serving runtime。

   Serving 运行时 页面将打开，并显示已安装和启用的 model-serving 运行时。

2. 根据您用于部署模型的运行时，执行以下操作之一：

   - 如果您为 KServe 运行时使用了预安装的 TGIS Standalone ServingRuntime，请复制运行时以创建自定义版本，然后按照此流程的其余部分操作。有关复制预安装的 TGIS 运行时的更多信息，请参阅为单模式服务平台添加自定义模型运行时。
   - 如果您已使用自定义 TGIS 运行时，点运行时旁的操作菜单(RCU)，然后选择 Edit。

     嵌入的 YAML 编辑器会打开并显示自定义 model-serving 运行时的内容。

3. 添加或更新 BATCH_SAFETY_MARGIN 环境变量，并将值设为 30。同样，添加或更新 ESTIMATE_MEMORY_BATCH_SIZE 环境变量，并将值设为 8。

```
spec:
  containers:
    env:
    - name: BATCH_SAFETY_MARGIN
      value: 30
    - name: ESTIMATE_MEMORY_BATCH
      value: 8
```

注意

BATCH_SAFETY_MARGIN 参数设定了可用 GPU 内存的百分比，以保存回来，以避免 OOM 条件。BATCH_SAFETY_MARGIN 的默认值为 20。ESTIMATE_MEMORY_BATCH_SIZE 参数设置内存自动调整算法中使用的批处理大小。ESTIMATE_MEMORY_BATCH_SIZE 的默认值为 16。

4. *点 Update。*

   Serving 运行时 页面将打开，并显示已安装的运行时列表。请注意，显示您更新的自定义 model-serving 运行时。

5. *要重新部署模型以使参数更新生效，请执行以下操作：*

   a. *在 OpenShift AI 仪表板中，点 Model Serving > Deployed Models。*

   b. *找到您要重新部署的模型，点模型旁的操作菜单(RCU)，然后选择 Delete。*

   c. *如 在单模式服务平台上部署模型 中所述，重新部署模型。*

验证

- *您从模型服务器收到成功响应，不再看到 CUDA OOM 错误。*