Red Hat OpenShift AI Cloud Service 1

使用分布式工作负载

使用分布式工作负载，实现更快、效率更高的数据处理和模型培训

Last Updated: 2024-06-26

Red Hat OpenShift AI Cloud Service 1 使用分布式工作负载

使用分布式工作负载，实现更快、效率更高的数据处理和模型培训

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

分布式工作负载使数据科学家可以并行使用多个集群节点，以获得更快、效率更高的数据处理和模型培训。CodeFlare 框架简化了任务编配和监控，并提供与高级 GPU 支持的自动化资源扩展和最佳节点利用率的无缝集成。

目录

前言

要更快地培训复杂的机器学习模型或处理数据，数据科学家可以使用分布式工作负载功能并行在多个 OpenShift worker 节点上运行作业。这种方法可显著减少任务完成时间，并允许使用更大的数据集和更复杂的模型。

第 1 章 分布式工作负载概述

您可以使用分布式工作负载功能来排队、扩展和管理在 OpenShift 集群中多个节点运行数据科学工作负载所需的资源。通常，数据科学工作负载包括几种人工智能(AI)工作负载，包括机器学习(ML)和 Python 工作负载。

分布式工作负载提供以下优点：

- 由于减少处理时间，您可以更快速、更频繁地进行实验。
- 您可以使用较大的数据集，这可能会导致更准确的模型。
- 您可以使用无法在单个节点上接受的复杂模型。
- 您可以随时提交分布式工作负载，系统然后在所需资源可用时调度分布式工作负载。

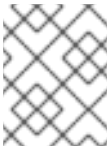
分布式工作负载基础架构包括以下组件：

CodeFlare Operator

保护部署的 Ray 集群，并授予对其 URL 的访问

CodeFlare SDK

为任何基于 Python 的环境定义和控制远程分布式计算作业和基础架构



注意

CodeFlare SDK 未作为 OpenShift AI 的一部分安装，但它包含在 OpenShift AI 提供的一些笔记本镜像中。

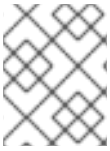
KubeRay

在 OpenShift 上管理远程 Ray 集群，以运行分布式计算工作负载

Kueue

管理配额以及分布式工作负载如何使用配额，以及管理分布式工作负载的队列与配额相关

您可以从数据科学管道、Jupyter 笔记本或 Microsoft Visual Studio Code 文件运行分布式工作负载。



注意

Data Science Pipelines (DSP)工作负载不由分布式工作负载功能管理，且不包含在分布式工作负载指标中。

1.1. KUEUE 资源概述

集群管理员可以配置 Kueue 资源类型、集群队列和本地队列，以管理 OpenShift 集群中多个节点上的分布式工作负载资源。

1.1.1. Resource flavour

Kueue **ResourceFlavor** 对象描述了集群中可用的资源变体。

集群中的资源可以是 *同构或 异构的* ：

- 同构资源在集群中是相同的：相同的节点类型、CPU、内存、加速器等。

- 异构资源在集群之间有变化。

如果集群具有同构资源，或者不需要为资源的不同类别管理单独的配额，集群管理员可以创建名为 **default-flavor** 的空 **ResourceFlavor** 对象，如下所示：

homegeneous 资源的空 Kueue 资源类别

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: ResourceFlavor
metadata:
  name: default-flavor
```

如果集群具有异构资源，集群管理员可以为可用资源的不同变体定义不同的资源类别。示例变化包括不同的 CPU、不同的内存或不同的加速器。然后，集群管理员可以使用标签、污点和容限将资源类别与集群节点关联，如下例所示。

异构资源的 Kueue 资源类型示例

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: ResourceFlavor
metadata:
  name: "spot"
spec:
  nodeLabels:
    instance-type: spot
  nodeTaints:
  - effect: NoSchedule
    key: spot
    value: "true"
  tolerations:
  - key: "spot-taint"
    operator: "Exists"
    effect: "NoSchedule"
```

注意

在 OpenShift AI 1 中，红帽支持每个集群一个集群队列（即同构集群），且只支持空资源类型。

有关配置资源类型的更多信息，请参阅 Kueue 文档中的 资源类型。https://kueue.sigs.k8s.io/docs/concepts/resource_flavor/

1.1.2. 集群队列

Kueue **ClusterQueue** 对象管理集群资源池，如 pod、CPU、内存和加速器。个集群可以有多个集群队列，每个集群队列可以引用多个资源类型。

集群管理员可以配置集群队列，以定义队列管理的资源类型，并为各个资源类别中的每个资源分配配额。集群管理员也可以配置使用限制和排队策略，以便在集群中的多个集群队列之间应用公平共享规则。

以下示例将集群队列配置为分配 9 个 CPU、36 GiB 内存、5 个 pod 和 5 NVIDIA GPU 的配额。

集群队列示例

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: ClusterQueue
metadata:
  name: "cluster-queue"
spec:
  namespaceSelector: {} # match all.
  resourceGroups:
  - coveredResources: ["cpu", "memory", "pods", "nvidia.com/gpu"]
    flavors:
    - name: "default-flavor"
      resources:
      - name: "cpu"
        nominalQuota: 9
      - name: "memory"
        nominalQuota: 36Gi
      - name: "pods"
        nominalQuota: 5
      - name: "nvidia.com/gpu"
        nominalQuota: '5'
```

只有在所需资源达到这些配额限制时，集群队列才会启动分布式工作负载。如果分布式工作负载中资源的请求总和大于集群队列中该资源的指定配额，集群队列不接受分布式工作负载。

有关配置集群队列的更多信息，请参阅 Kueue 文档中的 Cluster Queue。

1.1.3. 本地队列

Kue **LocalQueue** 对象组在一个项目中紧密相关的分布式工作负载。集群管理员可以配置本地队列来指定项目名称和关联的集群队列。然后，每个本地队列都授予对指定集群队列管理的资源的访问权限。集群管理员可以选择在项目中定义一个本地队列，作为该项目的默认本地队列。

在配置分布式工作负载时，用户指定了本地队列名称。如果集群管理员配置了默认的本地队列，用户可以从分布式工作负载代码中省略本地队列规格。

如果请求的总资源位于集群队列中指定的配额限制中，Kueue 从与集群队列的集群队列中为分布式工作负载分配资源。

以下示例为 **team-a** 项目配置了一个名为 **team-a-queue** 的本地队列，并将 **cluster-queue** 指定为关联的集群队列。

本地队列示例

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: LocalQueue
metadata:
  namespace: team-a
  name: team-a-queue
  annotations:
    kueue.x-k8s.io/default-queue: "true"
spec:
  clusterQueue: cluster-queue
```

在本例中，kueu **e.x-k8s.io/default-queue: "true"** 注解将此本地队列定义为 **team-a** 项目的默认本地队列。如果用户在 **team-a** 项目中提交分布式工作负载，且分布式工作负载没有在集群配置中指定本地队列，Kue 会自动将分布式工作负载路由到 **team-a-queue** 本地队列。然后，分布式工作负载可以访问

cluster-queue 集群队列管理的资源。

有关配置本地队列的更多信息，请参阅 Kueue 文档中的 Local Queue。

第 2 章 配置分布式工作负载

要为数据科学家配置分布式工作负载功能，以便在 OpenShift AI 中使用，您必须创建所需的 Kueue 资源，在 Red Hat OpenShift AI Add-on 中启用多个组件，并选择性地配置 CodeFlare Operator。

2.1. 配置分布式工作负载组件

要为数据科学家配置分布式工作负载功能，以便在 OpenShift AI 中使用，您必须启用几个组件。

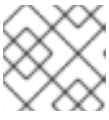
先决条件

- 已使用 **cluster-admin** 角色登录到 OpenShift。
- 您可以访问数据科学项目。
- 已安装 Red Hat OpenShift AI。
- 您有足够的资源。除了 安装和部署 OpenShift AI 中描述的最小 OpenShift AI 资源外，您需要 1.6 vCPU 和 2 GiB 内存来部署分布式工作负载基础架构。
- 您可以访问 Ray 集群镜像。有关如何创建 Ray 集群的详情，请参考 Ray Clusters 文档。

注意

在 OpenShift AI 的 CodeFlare 组件中默认启用 mutual Transport Layer Security (mTLS)。OpenShift AI 1 不支持 Ray 作业规格中的 submit **Mode=K8sJobMode** 设置，因此 KubeRay Operator 无法创建一个提交的 Kubernetes 作业来提交 Ray 任务。相反，用户必须配置 Ray 作业规格来设置 **commitMode=HTTPMode**，以便 KubeRay Operator 向 RayCluster 发送请求以创建 Ray 作业。

- 您可以访问分布式工作负载使用的数据集和型号。
- 您可以访问分布式工作负载的 Python 依赖项。
- 您已删除了之前安装的 CodeFlare Operator 实例，如知识库文章 如何在数据科学集群中从单独安装的 CodeFlare Operator 迁移。
- 如果要使用图形处理单元(GPU)，已在 OpenShift AI 中启用 GPU 支持。请参阅 OpenShift AI 中的启用 GPU 支持。



注意

在 OpenShift AI 1 中，红帽只支持分布式工作负载的 NVIDIA GPU 加速器。

- 如果要使用自签名证书，已将它们添加到中央证书颁发机构(CA)捆绑包中，如 使用证书 中所述。https://access.redhat.com/documentation/zh-cn/red_hat_openshift_ai_cloud_service/1/html/installing_and_uninstalling_openshift_ai_cloud_service/working-with-certificates_certs不需要额外的配置来将这些证书与分布式工作负载一起使用。集中配置的自签名证书会在以下挂载点的工作负载 pod 中自动提供：
  - 集群范围的 CA 捆绑包：

```
/etc/pki/tls/certs/odh-trusted-ca-bundle.crt
/etc/ssl/certs/odh-trusted-ca-bundle.crt
```

- 自定义 CA 捆绑包：

```
/etc/pki/tls/certs/odh-ca-bundle.crt
/etc/ssl/certs/odh-ca-bundle.crt
```

流程

1. 在 OpenShift 控制台中，点 **Operators → Installed Operators**。
2. 搜索 **Red Hat OpenShift AI** Operator，然后点 Operator 名称以打开 Operator 详情页面。
3. 点 **Data Science Cluster** 选项卡。
4. 单击默认实例名称（如 **default-dsc**），以打开实例详情页面。
5. 点 **YAML** 选项卡显示实例规格。
6. 启用所需的分布式工作负载组件。在 **spec:components** 部分中，为所需组件正确设置 **managementState** 字段。所需组件列表取决于分布式工作负载是否从管道或笔记本运行，如下表所示。

表 2.1. 分布式工作负载所需的组件

组件	仅限管道	仅限笔记本	管道和笔记本
codeflare	**受管**	**受管**	**受管**
dashboard	**受管**	**受管**	**受管**
datasciencepipelines	**受管**	**删除**	**受管**
kueue	**受管**	**受管**	**受管**
ray	**受管**	**受管**	**受管**
工作台	**删除**	**受管**	**受管**

7. 点击 **Save**。片刻后，带有 **Managed** 状态的组件已就绪。

验证

检查 **codeflare-operator-manager**、**kuberay-operator** 和 **kue-controller-manager** pod 的状态，如下所示：

1. 在 OpenShift 控制台中，从 **Project** 列表中选择 **redhat-ods-applications**。
2. 点 **Workloads → Deployments**。
3. 搜索 **codeflare-operator-manager**、**kuberay-operator** 和 **kueue-controller-manager** 部署。在每个情况下，检查状态，如下所示：
   a. 单击部署名称以打开部署详情页面。
   b. 点 **Pods** 选项卡。

c. 检查 pod 状态。
当 **codeflare-operator-manager-** *<pod-id>*, **kuberay-operator-** *<pod-id>* , 和 **kueue-controller-manager-** *<pod-id>* pod 为 **Running** 时，pod 已就绪。

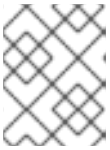
d. 要查看有关每个 pod 的更多信息，请点 pod 名称以打开 pod 详情页面，然后点 **Logs** 选项卡。

2.2. 为分布式工作负载配置配额管理

在集群中为分布式工作负载配置配额，以便您可以在多个数据科学项目之间共享资源。

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)。
- 您已启用了所需的分布式工作负载组件，如 配置分布式工作负载组件 中所述。
- 您已创建了包含工作台的数据科学项目，工作台正在运行包含 CodeFlare SDK 的默认笔记本镜像，如 **Standard Data Science** 笔记本。有关如何创建项目的详情，请参考 创建数据科学项目。
- 您有足够的资源。除了基本 OpenShift AI 资源外，还需要 1.6 个 vCPU 和 2 GiB 内存来部署分布式工作负载基础架构。
- 这些资源在集群中物理可用。



注意

在 OpenShift AI 1 中，红帽支持每个集群一个集群队列（即同构集群），且只支持空资源类型。有关 Kueue 资源的更多信息，请参阅 Kueue 资源概述。

- 如果要使用图形处理单元(GPU)，已在 OpenShift AI 中启用 GPU 支持。请参阅 OpenShift AI 中的启用 GPU 支持。

注意

在 OpenShift AI 1 中，红帽只支持分布式工作负载的 NVIDIA GPU 加速器。

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. 创建一个空 Kue 资源类别，如下所示：

a. 创建名为 **default_flavor.yaml** 的文件，并使用以下内容填充：

空Kue 资源类别

```
apiVersion: kueue.x-k8s.io/v1beta1
```

```
kind: ResourceFlavor
metadata:
  name: default-flavor
```

b. 应用配置以创建 **default-flavor** 对象：

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

3. 创建一个集群队列来管理空 Kue 资源类别，如下所示：

a. 创建名为 **cluster_queue.yaml** 的文件，并使用以下内容填充：

集群队列示例

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: ClusterQueue
metadata:
  name: "cluster-queue"
spec:
  namespaceSelector: {}  # match all.
  resourceGroups:
  - coveredResources: ["cpu", "memory", "nvidia.com/gpu"]
    flavors:
    - name: "default-flavor"
      resources:
      - name: "cpu"
        nominalQuota: 9
      - name: "memory"
        nominalQuota: 36Gi
      - name: "nvidia.com/gpu"
        nominalQuota: 5
```

b. 将示例配额值(9 个 CPU、36 GiB 内存和 5 NVIDIA GPU)替换为集群队列的适当值。只有所需资源在这些配额限值内时，集群队列才会启动分布式工作负载。
您必须为每个用户可以请求的资源指定一个配额，即使请求的值为 0，更新 **spec.resourceGroups** 部分，如下所示：

- 在 **coveredResources** 列表中包含资源名称。
- 在 **flavor.resources** 部分中指定 **资源名称** 和 **nominalQuota**，即使 **nominalQuota** 值为 0。

c. 应用配置以创建 **cluster-queue** 对象：

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

4. 创建一个指向集群队列的本地队列，如下所示：

a. 创建名为 **local_queue.yaml** 的文件，并使用以下内容填充：

本地队列示例

```
apiVersion: kueue.x-k8s.io/v1beta1
kind: LocalQueue
metadata:
```

```
  namespace: test
  name: local-queue-test
  annotations:
    kueue.x-k8s.io/default-queue: 'true'
spec:
  clusterQueue: cluster-queue
```

kueue.x-k8s.io/default-queue: 'true' 注解将这个队列定义为默认队列。如果在数据科学项目管道或 Jupyter 笔记本或 Microsoft Visual Studio Code 文件的 **ClusterConfiguration** 部分中没有指定 **local_queue** 值，则会将分布式工作负载提交到此队列。

b. 更新 **namespace** 值，以指定在创建 Ray 集群的 **ClusterConfiguration** 部分中相同的命名空间。

c. 可选：相应地更新 **name** 值。

d. 应用配置以创建 local-queue 对象：

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

集群队列分配资源以在本地队列中运行分布式工作负载。

验证

检查项目中本地队列的状态，如下所示：

```
$ oc get -n <project-name> localqueues
```

其他资源

- Kueue 文档

2.3. 配置 CODEFLARE OPERATOR

如果要为 OpenShift AI 中的分布式工作负载更改 CodeFlare Operator 的默认配置，您可以编辑关联的配置映射。

先决条件

- 已使用 **cluster-admin** 角色登录到 OpenShift。
- 您已启用了所需的分布式工作负载组件，如 配置分布式工作负载组件 中所述。

流程

1. 在 OpenShift 控制台中，点击 **Workloads → ConfigMaps**。
2. 从 **Project** 列表中，选择 **redhat-ods-applications**。
3. 搜索 **codeflare-operator-config** 配置映射，然后点配置映射名称以打开 ConfigMap 详情页面。
4. 点 **YAML** 选项卡显示配置映射规格。
5. 在 **data:config.yaml:kuberay** 部分中，您可以编辑以下条目：

ingressDomain

这个配置选项默认为 null (**ingressDomain: ""**)。不要更改这个选项，除非 Ingress Controller 在 OpenShift 上运行。OpenShift AI 使用这个值为每个 Ray Cluster 生成仪表板和客户端路由，如下例所示：

仪表板和客户端路由示例

```
ray-dashboard-<clustername>-<namespace>.<your.ingress.domain>
ray-client-<clustername>-<namespace>.<your.ingress.domain>
```

mtlsEnabled

默认启用此配置选项(**mTLSEnabled: true**)。启用此选项后，Ray 集群 pod 创建用于 mutual Transport Layer Security (mTLS)的证书，这是 Ray Cluster 节点之间的 mutual 身份验证形式。启用此选项时，Ray 客户端无法连接到 Ray head 节点，除非它们从 **ca-secret-_<cluster_name>_** secret 下载生成的证书，然后为 mTLS 通信生成必要的证书，然后设置所需的 Ray 环境变量。然后，用户必须重新初始化 Ray 客户端以应用更改。CodeFlare SDK 提供了以下功能来简化 Ray 客户端的身份验证过程：

Ray 客户端身份验证代码示例

```
from codeflare_sdk import generate_cert

generate_cert.generate_tls_cert(cluster.config.name, cluster.config.namespace)
generate_cert.export_env(cluster.config.name, cluster.config.namespace)

ray.init(cluster.cluster_uri())
```

rayDashboardOauthEnabled

这个配置选项会被默认启用(**rayDashboardOAuthEnabled: true**)。启用此选项后，OpenShift AI 在 Ray Cluster head 节点前放置 OpenShift OAuth 代理。然后，在通过浏览器访问 Ray Dashboard 时，用户必须使用其 OpenShift 集群登录凭据进行身份验证。如果用户希望以其他方式访问 Ray Dashboard （例如，使用 Ray **JobSubmissionClient** 类），则必须设置一个授权标头作为其请求的一部分，如下例所示：

授权标头示例

```
{Authorization: "Bearer <your-openshift-token>"}
```

6. 要保存您的更改，请单击 **Save**。
7. 要应用您的更改，请删除 pod：
   a. 点击 **Workloads → Pods**。
   b. 找到 **codeflare-operator-manager-** *<pod-id>* pod。
   c. 点该 pod 的选项菜单（需要），然后点 **Delete Pod**。pod 重启，并显示您的更改。

验证

检查 **codeflare-operator-manager** pod 的状态，如下所示：

1. 在 OpenShift 控制台中，点 **Workloads → Deployments**。

2. 搜索 **codeflare-operator-manager** 部署，然后单击部署名称以打开部署详情页面。

3. 点 **Pods** 选项卡。当 **codeflare-operator-manager-** *<pod-id>* pod 处于 **Running** 时，pod 已准备好使用。要查看有关 pod 的更多信息，请点 pod 名称以打开 pod 详情页面，然后点 **Logs** 选项卡。

第 3 章 运行分布式工作负载

在 OpenShift AI 中，您可以从笔记本或管道运行分布式工作负载。如果可以访问所有所需的软件，您还可以在断开连接的环境中运行分布式工作负载。

3.1. 从笔记本运行分布式数据科学工作负载

要从笔记本中运行分布式数据科学工作负载，您必须首先提供到您的 Ray 集群镜像的链接。

先决条件

- 您可以访问配置为运行分布式工作负载的数据科学项目，如 配置分布式工作负载 中所述。
- 集群管理员已创建了所需的 Kueue 资源，如 为分布式工作负载配置配额管理 中所述。
- 可选：您的集群管理员通过创建一个 **LocalQueue** 资源，并在其配置详情中添加以下注解，如 为分布式工作负载配置配额管理 中所述，为 Ray 集群定义了 *默认的* 本地队列：

```
"kueue.x-k8s.io/default-queue": "true"
```

注意

如果集群管理员没有定义默认的本地队列，则必须在每个笔记本中指定本地队列。

- 您已创建了包含工作台的数据科学项目，工作台正在运行包含 CodeFlare SDK 的默认笔记本镜像，如 **Standard Data Science** 笔记本。有关如何创建项目的详情，请参考 创建数据科学项目。
- 您有数据科学项目的 Admin 访问权限。
  - 如果创建项目，则自动具有 Admin 访问权限。
  - 如果没有创建项目，您的集群管理员必须授予 Admin 访问权限。
- 您已启动笔记本服务器并登录到笔记本编辑器。此流程中的示例引用 JupyterLab 集成开发环境 (IDE)。

流程

1. 下载 CodeFlare SDK 提供的演示笔记本。演示笔记本提供了如何在您自己的笔记本中使用 CodeFlare 堆栈的指南。
   要访问演示笔记本，请克隆 **codeflare-sdk** 存储库，如下所示：
   a. 在 JupyterLab 界面中，点 **Git > Clone a Repository**。
   b. 在 "Clone a repo" 对话框中，输入 **https://github.com/project-codeflare/codeflare-sdk.git**，然后点 **Clone**。**codeflare-sdk** 存储库列在左侧导航窗格中。
2. 找到下载的演示笔记本，如下所示：
   a. 在 JupyterLab 界面中，在左侧导航窗格中，双击 **codeflare-sdk**。
   b. 双击 **demo-notebooks**，然后双击 **guided-demos**。
3. 更新每个示例演示笔记本，如下所示：

a. 如果尚未指定，请更新 **导入** 部分以导入 **generate_cert** 组件：

更新了导入部分

```
from codeflare_sdk import generate_cert
```

b. 将 default 命名空间值替换为 data Science 项目的名称。

c. 在笔记本代码的 **TokenAuthentication** 部分中，提供令牌和服务器详情，以使用 CodeFlare SDK 向 OpenShift 集群进行身份验证。

d. 将到示例 community 镜像的链接替换为到您的 Ray 集群镜像的链接。

e. 确保 Ray 集群创建部分后包含以下 Ray 集群身份验证代码。

Ray 集群身份验证代码

```
generate_cert.generate_tls_cert(cluster.config.name, cluster.config.namespace)
generate_cert.export_env(cluster.config.name, cluster.config.namespace)
```

注意

在 OpenShift AI 的 CodeFlare 组件中默认启用 mutual Transport Layer Security (mTLS)。您必须包含 Ray 集群身份验证代码，以启用在笔记本中运行的 Ray 客户端，以连接到启用了 mTLS 的安全 Ray 集群。

f. 如果您还没有配置默认本地队列，方法是包括 **kueue.x-k8s.io/default-queue: 'true'** 注解，如 为分布式工作负载配置配额管理 中所述，更新 **ClusterConfiguration** 部分来为 Ray 集群指定本地队列，如下例所示：

本地队列分配示例

```
local_queue="your_local_queue_name"
```

g. 可选：在 **ClusterConfiguration** 部分中，为 Ray 集群分配一个 **标签** 参数的字典来进行识别和管理，如下例所示：

标签分配示例

```
labels = {"exampleLabel1": "exampleLabel1Value", "exampleLabel2":
"exampleLabel2Value"}
```

4. 运行笔记本。

验证

笔记本运行完成后且无错误。在笔记本中，**cluster.status （）** 函数或 **cluster.details （）** 函数的输出表示 Ray 集群是 **Active**。

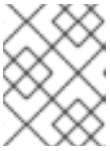
3.2. 从数据科学项目运行分布式数据科学工作负载

要从数据科学项目管道运行分布式数据科学工作负载，您必须首先更新管道，使其包含到 Ray 集群镜像的链接。

先决条件

- 已使用 **cluster-admin** 角色登录到 OpenShift。
- 您可以访问配置为运行分布式工作负载的数据科学项目，如 配置分布式工作负载 中所述。
- 集群管理员已创建了所需的 Kueue 资源，如 为分布式工作负载配置配额管理 中所述。
- 可选：您的集群管理员通过创建一个 **LocalQueue** 资源，并在该 **LocalQueue** 资源的配置详情中添加以下注解，如 为分布式 工作负载配置配额管理 中所述，为 Ray 集群定义了 *默认* 本地队列：

```
"kueue.x-k8s.io/default-queue": "true"
```



注意

如果集群管理员没有定义默认的本地队列，则必须在每个管道中指定一个本地队列。

- 您可以访问 S3 兼容对象存储。
- 您已登陆到 Red Hat OpenShift AI。
- 您已创建了包含工作台的数据科学项目，工作台正在运行包含 CodeFlare SDK 的默认笔记本镜像，如 **Standard Data Science** 笔记本。有关如何创建项目的详情，请参考 创建数据科学项目。
- 您有数据科学项目的 Admin 访问权限。
  - 如果创建项目，则自动具有 Admin 访问权限。
  - 如果没有创建项目，您的集群管理员必须授予 Admin 访问权限。

流程

1. 创建数据连接以将对象存储连接到您的数据科学项目，如 向数据科学项目添加数据连接 中所述。
2. 将管道服务器配置为使用数据连接，如 配置管道服务器 中所述。
3. 按如下方式创建数据科学项目管道：
   a. 安装 **kfp** Python 软件包，这是所有管道都需要的：

```
$ pip install kfp
```

   b. 安装管道所需的任何其他依赖项。
   c. 在 Python 代码中构建您的数据科学项目管道。
   例如，使用以下内容创建一个名为 **compile_example.py** 的文件：

```
from kfp import dsl


@dsl.component(
    base_image="registry.redhat.io/ubi8/python-39:latest",
    packages_to_install=['codeflare-sdk']
)
```

```
def ray_fn():
  import ray (1)
  import time (2)
  from codeflare_sdk import Cluster, ClusterConfiguration, generate_cert (3)


  cluster = Cluster( (4)
    ClusterConfiguration(
      namespace="my_project", (5)
      name="raytest",
      num_workers=1,
      head_cpus="500m",
      min_memory=1,
      max_memory=1,
      num_gpus=0,
      image="quay.io/project-codeflare/ray:latest-py39-cu118", (6)
      local_queue="local_queue_name", (7)
    )
  )


  print(cluster.status())
  cluster.up() (8)
  // cluster.wait_ready()
  time.sleep(180) (9)
  print(cluster.status())
  print(cluster.details())


  ray_dashboard_uri = cluster.cluster_dashboard_uri()
  ray_cluster_uri = cluster.cluster_uri()
  print(ray_dashboard_uri, ray_cluster_uri)

  # Enable Ray client to connect to secure Ray cluster that has mTLS enabled
  generate_cert.generate_tls_cert(cluster.config.name, cluster.config.namespace) (10)
  generate_cert.export_env(cluster.config.name, cluster.config.namespace)


  ray.init(address=ray_cluster_uri)
  print("Ray cluster is up and running: ", ray.is_initialized())


  @ray.remote
  def train_fn(): (11)
    # complex training function
    return 100


  result = ray.get(train_fn.remote())
  assert 100 == result
  ray.shutdown()
  cluster.down() (12)
```

```
    auth.logout()
    return result


@dsl.pipeline( 13
    name="Ray Simple Example",
    description="Ray Simple Example",
)


def ray_integration():
    ray_fn()


if __name__ == '__main__': 14
     from kfp.compiler import Compiler
     Compiler().compile(ray_integration, 'compiled-example.yaml')
```

1. 导入 Ray。
2. 导入 **time** 软件包，以便您可以使用 **sleep** 功能在代码执行期间等待，作为 RHOAIENG-7346 的临时解决方案。
3. 从 CodeFlare SDK 中导入软件包以定义集群功能。
4. 指定 Ray 集群配置：将这些示例值替换为 Ray 集群的值。
5. 可选：指定创建 Ray 集群的项目。将示例值替换为项目的名称。如果省略这一行，则会在当前项目中创建 Ray 集群。
6. 指定 Ray 集群镜像的位置。如果您在断开连接的环境中运行这个代码，请将默认值替换为您的环境的位置。
7. 指定将向其提交 Ray 集群的本地队列。如果配置了默认的本地队列，您可以省略这一行。
8. 使用指定的镜像和配置创建 Ray 集群。
9. 在继续操作前，等待 Ray 集群就绪。作为 RHOAIENG-7346 的一个临时解决方案，请使用 **time.sleep (180)** 而不是 **cluster.wait_ready （）**。
10. 启用 Ray 客户端连接到启用了 mutual Transport Layer Security (mTLS)的安全 Ray 集群。OpenShift AI 中的 CodeFlare 组件中默认启用 mTLS。
11. 将本节中的示例详情替换为您的工作负载详情。
12. 当工作负载完成后，删除 Ray 集群。
13. 将示例名称和描述替换为您的工作负载值。
14. 编译 Python 代码，并将输出保存到 YAML 文件中。

d. 编译 Python 文件（本例中为 **compile_example.py** 文件）：

```
$ python compile_example.py
```

这个命令会创建一个 YAML 文件（在这个示例中是 **compiled-example.yaml**），您可以在下一步中导入该文件。

4. 导入您的数据科学管道，如 导入数据科学管道 中所述。
5. 调度管道运行，如 调度管道运行 中所述。
6. 当管道运行完成后，确认它包含在触发管道运行列表中，如 查看管道运行的详情 中所述。

验证

创建 YAML 文件，管道运行完成且无错误。

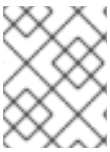
您可以查看运行的详情，如 查看管道运行的详情 中所述。

其他资源

- 使用数据科学项目管道
- Ray Clusters 文档

第 4 章 监控分布式工作负载

在 OpenShift AI 中，您可以查看分布式工作负载的项目指标，并查看所选项目中所有分布式工作负载的状态。您可以使用这些指标来监控分布式工作负载使用的资源，评估项目资源是否已正确分配，跟踪分布式工作负载的进度，并在需要时识别正确的操作。



注意

Data Science Pipelines (DSP)工作负载不由分布式工作负载功能管理，且不包含在分布式工作负载指标中。

4.1. 查看分布式工作负载的项目指标

在 OpenShift AI 中，您可以查看分布式工作负载的以下项目指标：

- cpu - 所选项目中所有分布式工作负载目前使用的 CPU 内核数。
- Memory - 当前被所选项目中所有分布式工作负载使用的内存量（以千字节(GiB)）。

您可以使用这些指标来监控分布式工作负载使用的资源，并评估项目资源是否已正确分配。

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**）。
- 您的数据科学项目包含分布式工作负载。

流程

1. 在 OpenShift AI左侧导航窗格中，单击 **Distributed Workloads Metrics**。
2. 从 **Project** 列表中，选择包含您要监控的分布式工作负载的项目。
3. 点 **Project metrics** 选项卡。
4. 可选：在 **Refresh interval** 列表中，选择一个值来指定指标页面上的图形要刷新的频率，以显示最新的数据。
   您可以选择其中一个值：15 秒、**30 秒** 、**1 分钟**、**5 分钟**、**15 分钟**、**30 分钟**、30 分钟、**1小时**、2 小时或 **1 天**。
5. 在 **Requested resources** 部分中，查看 **CPU** 和 **Memory** 图来识别分布式工作负载请求的资源，如下所示：
   - 所选项目请求
   - 所有项目请求，包括您无法访问的所选项目和项目
   - 由集群队列提供的所有项目共享配额总数

对于每个资源类型(CPU 和 Memory)，从 Total shared quota 值中减去所有项目值所请求的值，以计算尚未请求的资源配额数量，并可用于所有项目。

6. 向下滚动到 Top resource- consuming distributed workload部分，以查看以下图形：
   - 消耗最多 CPU 资源的 5 个分布式工作负载
   - 使用最多内存的 5 个分布式工作负载

   您还可以识别每个情形中正在使用的 CPU 或内存量。
7. 向下滚动以查看 分布式工作负载资源指标表，其中列出了所选项目中的所有分布式工作负载，并指示当前资源使用量以及每个分布式工作负载的状态。
   在每个表条目中，进度条表示此分布式工作负载目前正在使用请求的 CPU 和内存量。要查看 CPU 实际使用量和请求使用的数字值（以内核表示）和内存（以 GiB 为单位），将光标悬停在每个进度条上。将实际使用量与请求的使用量进行比较，以评估分布式工作负载配置。如有必要，重新配置分布式工作负载，以减少或增加请求的资源。

验证

在 Project metrics 选项卡中，图形和表为所选项目中的分布式工作负载提供资源使用数据。

4.2. 查看分布式工作负载的状态

在 OpenShift AI 中，您可以查看所选项目中所有分布式工作负载的状态。您可以跟踪分布式工作负载的进度，并在需要时识别正确的操作。

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）。
- 您的数据科学项目包含分布式工作负载。

流程

1. 在 OpenShift AI左侧导航窗格中，单击 Distributed Workloads Metrics。
2. 从 Project 列表中，选择包含您要监控的分布式工作负载的项目。
3. 点 Distributed workload status选项卡。
4. 可选：在 Refresh interval 列表中，选择一个值来指定指标页面上的图形要刷新的频率，以显示最新的数据。
   您可以选择其中一个值：15 秒、30 秒 、1 分钟、5 分钟、15 分钟、30 分钟、30 分钟、1小时、2 小时或 1 天。
5. 在 Status overview 部分中，查看所选项目中所有分布式工作负载的状态概述。
   状态可以是 Pending,Inadmissible,Admitted,Running,Evicted,Succeeded, 或 Failed。

6. 向下滚动以查看 分布式工作负载 表，该表列出了所选项目中的所有分布式工作负载。表提供了每个分布式工作负载的优先级、状态、创建日期和最新消息。
   latest 消息提供有关分布式工作负载当前状态的更多信息。查看最新的消息，以识别所需的任何纠正操作。例如，分布式工作负载 可能会丢失， 因为请求的资源超过可用资源。在这种情况下，您可以重新配置分布式工作负载来减少请求的资源，或者为项目重新配置集群队列以增加资源配额。

验证

在 分布式工作负载状态 选项卡上，图提供了所选项目中所有分布式工作负载状态的总结视图，表提供了有关每个分布式工作负载状态的更多详情。