Red Hat Quay 3.11

在 OpenShift Container Platform 上部署 Red Hat Quay Operator

在 OpenShift Container Platform 上部署 Red Hat Quay Operator

Last Updated: 2024-06-28

Red Hat Quay 3.11 在 OpenShift Container Platform 上部署 Red Hat Quay Operator

在 OpenShift Container Platform 上部署 Red Hat Quay Operator

法律通告

摘要

在 OpenShift Container Platform 集群中部署 Red Hat Quay Operator

目录

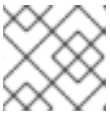
前言

Red Hat Quay 是一个企业级容器 registry。使用 Red Hat Quay 来构建和存储容器镜像，然后将其提供给整个企业部署。

Red Hat Quay Operator 提供了在 OpenShift 集群上部署和管理 Red Hat Quay 的简单方法。

随着 Red Hat Quay 3.4.0 的发布，Red Hat Quay Operator 被重新编写，以提供增强的体验，并为第 2 天操作添加更多支持。现在，Red Hat Quay Operator 更为简单。与 Red Hat Quay 3.4.0 之前的版本的主要区别包括：

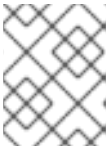
- **QuayEcosystem** 自定义资源已被 **QuayRegistry** 自定义资源替代。
- 默认安装选项生成完全支持的 Red Hat Quay 环境，其中包含所有受管依赖项，如数据库、缓存、对象存储等，支持在生产环境中使用。



注意

有些组件可能没有高可用性。

- Red Hat Quay 配置的新验证库。
- 对象存储现在可以使用 **ObjectBucketClaim** Kubernetes API 由 Red Hat Quay Operator 管理



注意

Red Hat OpenShift Data Foundation 可用于在 OpenShift Container Platform 上提供此 API 支持的实现。

- 自定义部署的 pod 用于测试和开发场景的容器镜像。

第 1 章 RED HAT QUAY OPERATOR 简介

使用本章中的内容执行以下操作：

- 使用 Red Hat Quay Operator 在 OpenShift Container Platform 上安装 Red Hat Quay
- 配置受管或非受管对象存储
- 配置非受管组件，如数据库、Redis、路由、TLS 等
- 使用 Red Hat Quay Operator 在 OpenShift Container Platform 上部署 Red Hat Quay registry
- 使用 Red Hat Quay 支持的高级功能
- 使用 Red Hat Quay Operator 升级 Red Hat Quay registry

1.1. RED HAT QUAY OPERATOR 组件

Red Hat Quay 有很多依赖项。这些依赖项包括数据库、对象存储、Red Hat Redis 等。Red Hat Quay Operator 管理一个建议部署 Red Hat Quay 及其 Kubernetes 的依赖关系。这些依赖项被视为 *组件*，并通过 **QuayRegistry** API 配置。

在 **QuayRegistry** 自定义资源中，**spec.components** 字段配置组件。每个组件包含两个字段： **kind**（组件名称） **和管理** （用于解决组件生命周期是否由 Red Hat Quay Operator 处理）的布尔值。

默认情况下，所有组件在协调后都会管理和自动填充，以了解可见性：

QuayRegistry 资源示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
  spec:
    configBundleSecret: config-bundle-secret
    components:
    - kind: quay
      managed: true
    - kind: postgres
      managed: true
    - kind: clair
      managed: true
    - kind: redis
      managed: true
    - kind: horizontalpodautoscaler
      managed: true
    - kind: objectstorage
      managed: true
    - kind: route
      managed: true
    - kind: mirror
      managed: true
    - kind: monitoring
      managed: true
```

```
- kind: tls
  managed: true
- kind: clairpostgres
  managed: true
```

1.2. 使用受管组件

除非 **QuayRegistry** 自定义资源另有指定，否则 Red Hat Quay Operator 将默认值用于以下受管组件：

- Quay ：保存在 OpenShift Container Platform 上部署 Red Hat Quay 的覆盖，如环境变量和副本数。此组件是 Red Hat Quay 3.7 的新组件，不能设置为 unmanaged。
- **postgres**：若要存储 registry 元数据，从 Red Hat Quay 3.9 开始，使用来自 Software Collections 的 PostgreSQL 13 版本。

注意

当从 Red Hat Quay 3.8 → 3.9 升级时，Operator 会自动处理将 PostgreSQL 10 升级到 PostgreSQL 13。这个升级是必需的。PostgreSQL 10 在 2022 年 11 月 10 日有其最终发行版本，不再被支持。

- **Clair**：提供镜像漏洞策略扫描。
- **Redis**：存储实时构建器日志和 Red Hat Quay 指南。还包括垃圾回收所需的锁定机制。
- **HorizontalPodAutoscaler**：根据内存/cpu 消耗调整 **Quay** pod 的数量。
- **ObjectStorage** ：为了存储镜像层 Blob，请使用由 Noobaa 或 Red Hat OpenShift Data Foundation 提供的 **ObjectBucketClaim** Kubernetes API。
- **Route**：从 OpenShift Container Platform 之外为 Red Hat Quay registry 提供外部入口点。
- **mirror** ：配置存储库镜像 worker 以支持可选存储库镜像。
- **监控** ：功能包括 Grafana 仪表板、访问单个指标以及经常重启 **Quay** pod 的通知。
- **tls** ：配置 Red Hat Quay 还是 OpenShift Container Platform 处理 SSL/TLS。
- **clairpostgres** ：配置受管 Clair 数据库。这是独立的数据库，与用于部署 Red Hat Quay 的 PostgreSQL 数据库不同。

Red Hat Quay Operator 处理 Red Hat Quay 使用受管组件所需的任何配置和安装工作。如果 Red Hat Quay Operator 执行的 opinionated 部署不适合您的环境，您可以提供 Red Hat Quay Operator 带有 **非受管资源或** 覆盖的 Red Hat Quay Operator，如以下部分所述。

1.3. 对依赖项使用非受管组件

如果您要与 Red Hat Quay 一起使用的现有组件，如 PostgreSQL、Redis 或对象存储，则首先在 Red Hat Quay 配置捆绑包或 **config.yaml** 文件中配置它们。然后，必须在 **QuayRegistry** 捆绑包中引用它们作为 Kubernetes **Secret**，同时指定哪些组件是非受管的。

注意

如果您使用非受管 PostgreSQL 数据库，且版本为 PostgreSQL 10，则强烈建议您升级到 PostgreSQL 13。PostgreSQL 10 在 2022 年 11 月 10 日有其最终发行版本，不再被支持。如需更多信息，请参阅 PostgreSQL 版本策略。

有关配置非受管组件，请参阅以下部分：

- 使用现有的 PostgreSQL 数据库
- 使用非受管 Horizontal Pod Autoscaler
- 使用非受管存储
- 使用一个非受管 NooBaa 实例
- 使用非受管 Redis 数据库
- 禁用路由组件
- 禁用监控组件
- 禁用镜像组件

1.4. 配置捆绑包 SECRET

spec.configBundleSecret 字段是与 **QuayRegistry** 资源相同的命名空间中的 **Secret** 的 **metadata.name** 的引用。此 **Secret** 必须包含 **config.yaml** 键/值对。

config.yaml 文件是一个 Red Hat Quay **config.yaml** 文件。此字段是可选的，如果未提供，由 Red Hat Quay Operator 自动填充。如果提供，它将作为基础配置字段，这些字段将与任何受管组件的其他字段合并，以形成最终输出 **Secret**，然后挂载到 Red Hat Quay 应用程序 pod 中。

1.5. OPENSHIFT CONTAINER PLATFORM 上 RED HAT QUAY 的先决条件

在使用 Red Hat Quay Operator 在 OpenShift Container Platform 上部署 Red Hat Quay 前，请考虑以下前提条件。

1.5.1. OpenShift Container Platform 集群

要部署 Red Hat Quay Operator，您必须有一个 OpenShift Container Platform 4.5 或更高版本的集群，并可以访问管理帐户。管理帐户必须能够在集群范围内创建命名空间。

1.5.2. 资源要求

每个 Red Hat Quay 应用程序 pod 都具有以下资源要求：

- 8 Gi 内存
- 2000 millicores CPU

Red Hat Quay Operator 会为每个它管理的 Red Hat Quay 部署创建一个应用程序 pod。确保 OpenShift Container Platform 集群有足够的计算资源来满足这些要求。

1.5.3. 对象存储

默认情况下，Red Hat Quay Operator 使用 **ObjectBucketClaim** Kubernetes API 来置备对象存储。使用此 API 将 Red Hat Quay Operator 与任何特定于供应商的实现分离。Red Hat OpenShift Data Foundation 通过其 NooBaa 组件提供此 API，该组件用作本文档中的示例。

Red Hat Quay 可以手动配置为使用以下任何受支持的云存储选项：

- Amazon S3 （请参阅 S3 IAM Bucket 策略，以了解有关为 Red Hat Quay 配置 S3 存储桶策略的详细信息）
- Microsoft Azure Blob Storage
- Google Cloud Storage
- Ceph 对象网关(RADOS)
- OpenStack Swift
- CloudFront + S3

1.5.4. StorageClass

当使用 Red Hat **Quay** Operator 部署 Quay 和 **Clair** PostgreSQL 数据库时，会在集群中配置一个默认 **StorageClass**。

Red Hat Quay Operator 使用的默认 **StorageClass** 会置备 **Quay** 和 **Clair** 数据库所需的持久性卷声明。这些 PVC 用于永久存储数据，确保您的 Red Hat Quay registry 和 Clair 漏洞扫描程序仍然可用，并在重启或失败时维护其状态。

在继续安装前，请验证集群中配置了默认 **StorageClass**，以确保为 **Quay** 和 **Clair** 组件无缝置备存储。

第 2 章 从 OPERATORHUB 安装 RED HAT QUAY OPERATOR

使用以下步骤从 OpenShift Container Platform OperatorHub 安装 Red Hat Quay Operator。

流程

1. 使用 OpenShift Container Platform 控制台，选择 **Operators → OperatorHub**。
2. 在搜索框中，键入 **Red Hat Quay** 并选择红帽提供的官方 Red Hat Quay Operator。这会将您定向到 **Installation** 页面，它概述了功能、先决条件和部署信息。
3. 选择 **Install**。这会将您定向到 **Operator 安装** 页面。
4. 以下选择可用于自定义安装：
    a. **更新频道 ：** 为最新版本选择更新频道，如 **stable-3.11**。
    b. **安装模式 ：**
        i. 如果您希望 Red Hat Quay Operator 在整个集群范围内可用，请选择 **All namespaces on the cluster**。建议您在集群范围安装 Red Hat Quay Operator。如果选择单个命名空间，则默认无法使用监控组件。
        ii. 如果您希望只在单一命名空间中部署，请选择 **A specific namespace on the cluster**。
            - **批准策略 ：** 选择批准自动或手动更新。建议自动更新策略。
5. 选择 **Install**。

第 3 章 在部署前配置 RED HAT QUAY

在 OpenShift Container Platform 上部署时，Red Hat Quay Operator 可以管理所有 Red Hat Quay 组件。但是，这是默认配置，当您想要对设置进行更多控制时，您可以在外部管理一个或多个组件。

使用以下模式配置非受管 Red Hat Quay 组件。

流程

1. 使用适当的设置创建 **config.yaml** 配置文件。对最小配置使用以下引用：

```
$ touch config.yaml
```

```
AUTHENTICATION_TYPE: Database
BUILDLOGS_REDIS:
    host: <quay-server.example.com>
    password: <strongpassword>
    port: 6379
    ssl: false
DATABASE_SECRET_KEY: <0ce4f796-c295-415b-bf9d-b315114704b8>
DB_URI: <postgresql://quayuser:quaypass@quay-server.example.com:5432/quay>
DEFAULT_TAG_EXPIRATION: 2w
DISTRIBUTED_STORAGE_CONFIG:
    default:
        - LocalStorage
        - storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - default
PREFERRED_URL_SCHEME: http
SECRET_KEY: <e8f9fe68-1f84-48a8-a05f-02d72e6eccba>
SERVER_HOSTNAME: <quay-server.example.com>
SETUP_COMPLETE: true
TAG_EXPIRATION_OPTIONS:
    - 0s
    - 1d
    - 1w
    - 2w
    - 4w
USER_EVENTS_REDIS:
    host: <quay-server.example.com>
    port: 6379
    ssl: false
```

2. 输入以下命令，使用配置文件创建 **Secret** ：

```
$ oc create secret generic --from-file config.yaml=./config.yaml config-bundle-secret
```

3. 创建一个 **quayregistry.yaml** 文件，标识非受管组件并引用创建的 **Secret**，例如：

QuayRegistry YAML 文件示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
```

```
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  configBundleSecret: <config_bundle_secret>
  components:
    - kind: objectstorage
      managed: false
# ...
```

4. 输入以下命令使用 **quayregistry.yaml** 文件部署 registry：

```
$ oc create -n quay-enterprise -f quayregistry.yaml
```

3.1. 预配置 RED HAT QUAY 用于自动化

Red Hat Quay 支持多种启用自动化的配置选项。用户可以在部署前配置这些选项，以减少与用户界面交互的需求。

3.1.1. 允许 API 创建第一个用户

要创建第一个用户，用户需要将 **FEATURE_USER_INITIALIZE** 参数设置为 **true**，并调用 **/api/v1/user/initialize** API。与需要现有机构中 OAuth 应用生成的 OAuth 令牌的所有其他 registry API 调用不同，API 端点不需要身份验证。

在部署 Red Hat Quay 后，用户可以使用 API 创建如 **quayadmin** 的用户，只要没有创建其他用户。如需更多信息，请参阅使用 API 创建第一个用户。

3.1.2. 启用常规 API 访问

用户应将 **BROWSER_API_CALLS_XHR_ONLY** 配置选项设置为 **false**，以允许对 Red Hat Quay registry API 的常规访问。

3.1.3. 添加超级用户

部署 Red Hat Quay 后，用户可以创建用户，并授予第一个具有完整权限的用户管理员特权。用户可以使用 **SUPER_USER** 配置对象提前配置完整的权限。例如：

```
# ...
SERVER_HOSTNAME: quay-server.example.com
SETUP_COMPLETE: true
SUPER_USERS:
  - quayadmin
# ...
```

3.1.4. 限制用户创建

配置超级用户后，您可以通过将 **FEATURE_USER_CREATION** 设置为 **false** 来限制创建新用户到超级用户组的能力。例如：

```
# ...
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
```

```
SUPER_USERS:
- quayadmin
FEATURE_USER_CREATION: false
# ...
```

3.1.5. 在 Red Hat Quay 3.11 中启用新功能

要使用新的 Red Hat Quay 3.11 功能，请启用以下一些或所有功能：

```
# ...
FEATURE_UI_V2: true
FEATURE_UI_V2_REPO_SETTINGS: true
FEATURE_AUTO_PRUNE: true
ROBOTS_DISALLOW: false
# ...
```

3.1.6. 建议自动化配置

建议对自动化使用以下 **config.yaml** 参数：

```
# ...
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- quayadmin
FEATURE_USER_CREATION: false
# ...
```

3.2. 配置对象存储

您需要在安装 Red Hat Quay 前配置对象存储，无论您是否允许 Red Hat Quay Operator 管理存储还是自行管理。

如果您希望 Red Hat Quay Operator 负责管理存储，请参阅受管存储部分，以了解有关 安装和配置 NooBaa 和 Red Hat OpenShift Data Foundations Operator 的信息。

如果使用单独的存储解决方案，请在配置 Operator 时将 **objectstorage** 设置为 **unmanaged**。请参见以下部分。非受管存储，以了解有关配置现有存储的详细信息。

3.2.1. 使用非受管存储

本节为您提供了非受管存储的配置示例。有关如何设置对象存储的完整说明，请参阅 Red Hat Quay 配置指南。

3.2.1.1. AWS S3 存储

在为您的 Red Hat Quay 部署配置 AWS S3 存储时，请使用以下示例。

```
DISTRIBUTED_STORAGE_CONFIG:
  s3Storage:
    - S3Storage
    - host: s3.us-east-2.amazonaws.com
```

```
      s3_access_key: ABCDEFGHIJKLMN
      s3_secret_key: OL3ABCDEFGHIJKLMN
      s3_bucket: quay_bucket
      s3_region: <region>
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
   - s3Storage
```

3.2.1.2. Google Cloud 存储

在为您的 Red Hat Quay 部署配置 Google Cloud 存储时，请使用以下示例。

```
DISTRIBUTED_STORAGE_CONFIG:
   googleCloudStorage:
      - GoogleCloudStorage
      - access_key: GOOGQIMFB3ABCDEFGHIJKLMN
        bucket_name: quay-bucket
        secret_key: FhDAYe2HeuAKfvZCAGyOioNaaRABCDEFGHIJKLMN
        storage_path: /datastorage/registry
        boto_timeout: 120 (1)
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
   - googleCloudStorage
```

(1) 可选。从连接时抛出超时异常的时间（以秒为单位）。默认值为 **60** 秒。另外，还包括时间（以秒为单位），直到尝试进行连接时抛出超时异常。默认值为 **60** 秒。

3.2.1.3. Microsoft Azure 存储

在为您的 Red Hat Quay 部署配置 Microsoft Azure 存储时，请使用以下示例。

```
DISTRIBUTED_STORAGE_CONFIG:
  azureStorage:
    - AzureStorage
    - azure_account_name: azure_account_name_here
      azure_container: azure_container_here
      storage_path: /datastorage/registry
      azure_account_key: azure_account_key_here
      sas_token: some/path/
      endpoint_url: https://[account-name].blob.core.usgovcloudapi.net (1)
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
   - azureStorage
```

(1) Microsoft Azure 存储的 **endpoint_url** 参数是可选的，可用于 Microsoft Azure Government (MAG) 端点。如果留空，则 **endpoint_url** 将连接到普通的 Microsoft Azure 区域。

从 Red Hat Quay 3.7 开始，您必须使用 MAG Blob 服务的主端点。使用 MAG Blob 服务的 Secondary 端点将导致以下错误： **AuthenticationErrorDetail:Cannot find the claimed account when trying to GetProperties for the account whusc8-secondary**.

3.2.1.4. Ceph/RadosGW Storage

在为您的 Red Hat Quay 部署配置 Ceph/RadosGW 存储时使用以下示例。

```
DISTRIBUTED_STORAGE_CONFIG:
  radosGWStorage: #storage config name
    - RadosGWStorage #actual driver
    - access_key: access_key_here #parameters
      secret_key: secret_key_here
      bucket_name: bucket_name_here
      hostname: hostname_here
      is_secure: 'true'
      port: '443'
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE: #must contain name of the storage config
    - radosGWStorage
```

3.2.1.5. Swift 存储

在为您的 Red Hat Quay 部署配置 Swift 存储时，请使用以下示例。

```
DISTRIBUTED_STORAGE_CONFIG:
  swiftStorage:
    - SwiftStorage
    - swift_user: swift_user_here
      swift_password: swift_password_here
      swift_container: swift_container_here
      auth_url: https://example.org/swift/v1/quay
      auth_version: 1
      ca_cert_path: /conf/stack/swift.cert"
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - swiftStorage
```

3.2.1.6. NooBaa 非受管存储

使用以下步骤将 NooBaa 部署为您的非受管存储配置。

流程

1. 进入 **Storage → Object Bucket Claims**，在 Red Hat Quay 控制台中创建一个 NooBaa Object Bucket Claim。
2. 检索 Object Bucket Claim Data details，包括 Access Key, Bucket Name, Endpoint (hostname)和 Secret Key。
3. 创建 **config.yaml** 配置文件，该文件使用 Object Bucket Claim 的信息：

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
    - RHOCSStorage
    - access_key: WmrXtSGk8B3nABCDEFGH
```

```
      bucket_name: my-noobaa-bucket-claim-8b844191-dc6c-444e-9ea4-87ece0abcdef
      hostname: s3.openshift-storage.svc.cluster.local
      is_secure: true
      port: "443"
      secret_key: X9P5SDGJtmSuHFCMSLMbdNCMfUABCDEFGH+C5QD
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
  - default
```

有关配置对象 Bucket 声明的更多信息，请参阅 Object Bucket Claim。

3.2.2. 使用一个非受管 NooBaa 实例

使用以下步骤为您的 Red Hat Quay 部署使用非受管 NooBaa 实例。

流程

1. 在控制台的 Storage → Object Bucket Claims 创建一个 NooBaa Object Bucket Claim。
2. 检索对象 Bucket 声明数据详细信息，包括 **Access Key**,**Bucket Name**,**Endpoint (hostname)**, 和 **Secret Key**。
3. 使用 Object Bucket Claim 的信息创建 **config.yaml** 配置文件。例如：

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
    - RHOCSStorage
    - access_key: WmrXtSGk8B3nABCDEFGH
      bucket_name: my-noobaa-bucket-claim-8b844191-dc6c-444e-9ea4-87ece0abcdef
      hostname: s3.openshift-storage.svc.cluster.local
      is_secure: true
      port: "443"
      secret_key: X9P5SDGJtmSuHFCMSLMbdNCMfUABCDEFGH+C5QD
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
  - default
```

3.2.3. 受管存储

如果您希望 Red Hat Quay Operator 管理 Red Hat Quay 的对象存储，您的集群需要能够通过 **ObjectBucketClaim** API 提供对象存储。使用 Red Hat OpenShift Data Foundation Operator，有两个支持的选项：

- 由本地 Kubernetes **PersistentVolume** 存储支持的 Multi-Cloud Object Gateway 的独立实例
  - 不可用
  - Red Hat Quay 订阅中包含
  - 不需要单独订阅 Red Hat OpenShift Data Foundation
- 带有横向扩展对象服务和 Ceph 的 Red Hat OpenShift Data Foundation 的生产环境部署

- 高可用性
- 为 Red Hat OpenShift Data Foundation 需要单独的订阅

要使用独立实例选项，请继续阅读以下内容。有关 Red Hat OpenShift Data Foundation 的生产部署，请参阅 官方文档。

注意

带有 50 GiB 的 Red Hat Quay Operator 会自动分配对象存储磁盘空间。这个数字代表了大多数中小型 Red Hat Quay 安装可用存储量，但可能不适用于您的用例。重新定义 Red Hat OpenShift Data Foundation 卷的大小目前不是由 Red Hat Quay Operator 处理。如需了解更多详细信息，请参见下面的有关调整受管存储大小的部分。

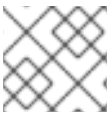
3.2.3.1. 为 Red Hat Quay 使用 Red Hat OpenShift Data Foundation Operator 中的 Multicloud 对象网关组件

作为 Red Hat Quay 订阅的一部分，用户有权使用 Red Hat OpenShift Data Foundation Operator 的 *Multicloud Object Gateway* 组件（以前称为 OpenShift Container Storage Operator）。此网关组件允许您将 S3 兼容对象存储接口提供给由基于 Kubernetes **PersistentVolume**- 的块存储支持的 Red Hat Quay。使用量仅限于由 Operator 管理的 Red Hat Quay 部署，以及 multicloud Object Gateway 实例的确切规格，如下所述。

由于 Red Hat Quay 不支持本地文件系统存储，因此用户可以结合使用网关与 Kubernetes **PersistentVolume** 存储，以提供受支持的部署。**PersistentVolume** 直接挂载到网关实例上，作为对象存储的后备存储，支持任何基于块的 **StorageClass**。

通过 **PersistentVolume** 的性质，这不是横向扩展、高度可用的解决方案，不会取代像 Red Hat OpenShift Data Foundation 等横向扩展的存储系统。只有一个网关实例正在运行。如果因为重新调度、更新或计划外停机运行网关的 pod 不可用，这会导致连接的 Red Hat Quay 实例出现临时降级。

使用以下步骤，安装 Local Storage Operator、Red Hat OpenShift Data Foundation，并创建独立多云对象网关，以便在 OpenShift Container Platform 上部署 Red Hat Quay。



注意

以下文档与官方 Red Hat OpenShift Data Foundation 文档 共享通用性。

3.2.3.1.1. 在 OpenShift Container Platform 上安装 Local Storage Operator

在本地存储设备上创建 Red Hat OpenShift Data Foundation 集群前，请使用以下步骤从 **OperatorHub** 安装 Local Storage Operator。

1. 登录 **OpenShift Web 控制台**。
2. 点 **Operators → OperatorHub**。
3. 在搜索框中输入 **本地存储**，从 Operator 列表中选择 Local Storage Operator。点 **Local Storage**。
4. 点 **Install**。
5. 在 Install Operator 页面中设置以下选项：
   - 对于 Update channel，选择 **stable**。

- 对于 Installation 模式，选择 **A specific namespace on the cluster**。
- 对于 Installed Namespace，请选择 **Operator recommended namespace openshift-local-storage**。
- 对于 Update approval，请选择 **Automatic**。

6. 点 **Install**。

3.2.3.1.2. 在 OpenShift Container Platform 上安装 Red Hat OpenShift Data Foundation

使用以下步骤在 OpenShift Container Platform 上安装 Red Hat OpenShift Data Foundation。

先决条件

- 使用具有 **cluster-admin** 和 Operator 安装权限的账户访问 OpenShift Container Platform 集群。
- OpenShift Container Platform 集群中必须至少有三个 worker 节点。
- 有关其他资源要求，请参阅规划您的部署指南。

流程

1. 登录 **OpenShift Web 控制台**。
2. 点 **Operators → OperatorHub**。
3. 在搜索框中输入 **OpenShift Data Foundation**。单击 **OpenShift Data Foundation**。
4. 点 **Install**。
5. 在 Install Operator 页面中设置以下选项：
   - 对于 Update channel，选择最新的稳定版本。
   - 对于 Installation 模式，选择 **A specific namespace on the cluster**。
   - 对于 Installed Namespace，选择 **Operator recommended Namespace: openshift-storage**。
   - 对于 Update approval，请选择 **Automatic** 或 **Manual**。
     如果选择 **Automatic** 更新，Operator Lifecycle Manager(OLM)将自动升级 Operator 的运行实例，而无需任何干预。

     如果选择 **手动** 更新，则 OLM 会创建一个更新请求。作为集群管理员，您必须手动批准该更新请求，才能将 Operator 更新至更新的版本。
   - 对于 Console 插件，选择 **Enable**。
6. 点 **Install**。
   安装 Operator 后，用户界面中会出现一个带有消息的弹出窗口，**Web 控制台更新** 就会出现在用户界面中。点这个弹出窗口中的 **Refresh web console** 来反映控制台的更改。
7. 继续以下部分"创建独立多云对象网关"，以利用 Red Hat Quay 的多云对象网关组件。

3.2.3.1.3. 使用 OpenShift Container Platform UI 创建独立多云对象网关

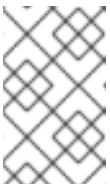
使用以下步骤创建独立多云对象网关。

先决条件

- 已安装 Local Storage Operator。
- 已安装 Red Hat OpenShift Data Foundation Operator。

流程

1. 在 **OpenShift Web 控制台中**，点 **Operators → Installed Operators** 查看所有已安装的 Operator。
   确保命名空间为 **openshift-storage**。
2. 单击 **Create StorageSystem**。
3. 在 **Backing storage** 页面中，选择以下内容：
   a. 为 **Deployment 类型**选择 **Multicloud Object Gateway**。
   b. 选择 **Create a new StorageClass using the local storage devices**选项。
   c. 点 **Next**。



注意

如果还没有安装，系统会提示您安装 Local Storage Operator。点 **Install**，并按照 "Installing the Local Storage Operator on OpenShift Container Platform" 中所述的步骤进行操作。

4. 在 **Create local volume set** 页面中，提供以下信息：
   a. 为 **LocalVolumeSet** 和 **StorageClass** 输入一个名称。默认情况下，存储类名称会出现本地卷集名称。您可以更改名称。
   b. 选择以下任意一项：
      - **所有节点上的磁盘**
        使用与所有节点上所选过滤器匹配的可用磁盘。
      - **所选节点上的磁盘**
        仅在所选节点上使用与所选过滤器匹配的可用磁盘。
   c. 从可用 **Disk Type** 列表中，选择 **SSD/NVMe**。
   d. 展开 **Advanced** 部分并设置以下选项：

卷模式	文件系统会被默认选择。始终为**卷模式**选择 Filesystem。
设备类型	从下拉列表中选择一个或多个设备类型。

磁盘大小	为设备设置最小 100GB 大小，以及需要包含的设备的最大可用大小。
磁盘限制上限	这表示节点上可以创建的 PV 数量上限。如果此字段留空，则为匹配节点上的所有可用磁盘创建 PV。

e. 点 **Next**
此时会显示一个用于确认创建 **LocalVolumeSet** 的弹出窗口。

f. 单击 **Yes** 以继续。

5. 在 **Capacity 和 nodes** 页面中，配置以下内容：

a. **可用的原始容量**会根据与存储类关联的所有附加磁盘填充容量值。这将需要一些时间才能出现。**Selected nodes** 列表根据存储类显示节点。

b. 点 **Next** 继续。

6. 可选。选择 **Connect to an external key management service**复选框。这是集群范围加密的可选选项。

a. 从 **Key Management Service Provider**下拉列表中，选择 **Vault** 或 **Thales CipherTrust Manager (using KMIP)**。如果选择了 **Vault**，请进入下一步。如果您选择了 **Thales CipherTrust Manager （using KMIP）**，请转到步骤 iii。

b. 选择**身份验证方法**。
使用令牌验证方法

- 输入唯一的连**接名称** , Vault 服务器的主机**地址**（'https://<hostname 或 ip>'），**端口**号和**令牌**。
- 展开 **Advanced Settings**，以根据您的 **Vault** 配置输入其他设置和证书详情：
  - 在 **后端路径**中输入为 OpenShift Data Foundation 专用且唯一的 Key Value secret 路径。
  - （可选）输入 **TLS 服务器名称**和 **Vault Enterprise 命名空间**。
  - 上传对应的 PEM 编码证书文件，以提供 **CA 证书、 客户端证书和客户端私钥**。
  - 点 **Save** 并跳过步骤 iv。
    使用 Kubernetes 验证方法
- 输入唯一的 Vault **Connection Name**, Vault 服务器的主机**地址**（'https://<hostname 或 ip>'）、**端口**号和**角色**名称。
- 展开 **Advanced Settings** 以根据您的 Vault 配置输入额外的设置和证书详情：
  - 在 后端路径中输入为 Red Hat OpenShift Data Foundation 专用 **且唯一的** Key Value secret 路径。
  - 可选：输入 **TLS Server Name** 和 **Authentication Path**（如果适用）。
  - 上传对应的 PEM 编码证书文件，以提供 **CA 证书、客户端证书和客户端私钥**。

- 点 **Save** 并跳过步骤 iv。

c. 要使用 **Thales CipherTrust Manager (using KMIP)** 作为 KMS 供应商，请按照以下步骤执行：

i. 在项目中输入密钥管理服务的唯一连**接名称**。

ii. 在 **Address** 和 **Port** 部分中，输入 Thales CipherTrust Manager 的 IP 以及在其中启用了 KMIP 接口的端口。例如：

- **地址**: 123.34.3.2
- **端口** ：5696

iii. 上传 **客户端证书**、**CA 证书**和 **客户端私钥**。

iv. 如果启用了 StorageClass 加密，请输入用于加密和解密的唯一标识符。

v. **TLS Server** 字段是可选的，并在没有 KMIP 端点的 DNS 条目时使用。例如，kmip**_all_<port>.ciphertrustmanager.local**。

d. 选择 **网络**。

e. 点 **Next**。

7. 在 **Review and create** 页面中，检查配置详情。若要修改任何配置设置，请单击 **Back**。

8. 单击 **Create StorageSystem**。

3.2.3.1.4. 使用 CLI 创建独立多云对象网关

使用以下步骤安装 Red Hat OpenShift Data Foundation （以前称为 OpenShift Container Storage）Operator 并配置单个实例 Multi-Cloud Gateway 服务。

注意

以下配置无法在安装 Red Hat OpenShift Data Foundation 的集群上并行运行。

流程

1. 在 **OpenShift Web 控制台中**，然后选择 **Operators → OperatorHub**。
2. 搜索 **Red Hat OpenShift Data Foundation**，然后选择 **Install**。
3. 接受所有默认选项，然后选择 **Install**。
4. 通过查看 **Status** 列确认已安装 Operator，它应标记为 **Succeeded**。

警告

安装 Red Hat OpenShift Data Foundation Operator 后，系统会提示您创建存储系统。不要遵循这个指令。反之，按照以下步骤创建 NooBaa 对象存储。

5. 在机器上，使用以下信息创建一个名为 **noobaa.yaml** 的文件：

```
apiVersion: noobaa.io/v1alpha1
kind: NooBaa
metadata:
  name: noobaa
  namespace: openshift-storage
spec:
 dbResources:
   requests:
     cpu: '0.1'
     memory: 1Gi
 dbType: postgres
 coreResources:
   requests:
     cpu: '0.1'
     memory: 1Gi
```

这将创建 *多云对象网关* 的单一实例部署。

6. 使用以下命令应用配置：

```
$ oc create -n openshift-storage -f noobaa.yaml
```

输出示例

```
noobaa.noobaa.io/noobaa created
```

7. 几分钟后，*Multi-cloud 对象网关* 应该完成调配。您可以输入以下命令来检查其状态：

```
$ oc get -n openshift-storage noobaas noobaa -w
```

输出示例

```
NAME     MGMT-ENDPOINTS            S3-ENDPOINTS              IMAGE
PHASE   AGE
noobaa   [https://10.0.32.3:30318]   [https://10.0.32.3:31958]   registry.redhat.io/ocs4/mcg-
core-
rhel8@sha256:56624aa7dd4ca178c1887343c7445a9425a841600b1309f6deace37ce6b8678d
Ready   3d18h
```

8. 通过创建以下 YAML 文件，名为 **noobaa-pv-backing-store.yaml**，为网关配置后备存储：

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: noobaa-pv-backing-store
  namespace: openshift-storage
spec:
  pvPool:
    numVolumes: 1
    resources:
      requests:
        storage: 50Gi (1)
    storageClass: STORAGE-CLASS-NAME (2)
  type: pv-pool
```

1. 对象存储服务的整体容量。根据需要进行调整。
2. 用于请求的 **PersistentVolume 的 StorageClass**。删除此属性以使用集群默认值。

9. 输入以下命令应用配置：

```
$ oc create -f noobaa-pv-backing-store.yaml
```

输出示例

```
backingstore.noobaa.io/noobaa-pv-backing-store created
```

这会为网关创建后备存储配置。Red Hat Quay 中的所有镜像将通过网关存储在由上述配置创建的 **PersistentVolume** 中的对象。

10. 运行以下命令，使 **PersistentVolume** 后端存储 Red Hat Quay Operator 发布的所有 **ObjectBucketClaims** 的默认 ：

```
$ oc patch bucketclass noobaa-default-bucket-class --patch '{"spec":{"placementPolicy":
{"tiers":[{"backingStores":["noobaa-pv-backing-store"]}]}}}' --type merge -n openshift-storage
```

第 4 章 配置流量入口

4.1. 配置 SSL/TLS 和路由

通过新的受管组件 **tls** 增加了对 OpenShift Container Platform *边缘终止路由* 的支持。这会将 **路由** 组件与 SSL/TLS 分开，并允许用户单独配置它们。

EXTERNAL_TLS_TERMINATION: true 是建议的设置。

注意

- Managed **tls** 表示使用默认的集群通配符证书。
- unmanaged **tls** 表示用户提供的密钥和证书对注入路由中。

ssl.cert 和 **ssl.key** 现在被移到一个单独的持久的 secret 中，这样可确保每次协调时不会重新生成密钥和证书对。密钥和证书对现在格式化为 **边缘路由**，并挂载到 **Quay** 容器中的同一目录中。

在配置 SSL/TLS 和路由时，可以进行多个攻击，但会应用以下规则：

- 如果 SSL/TLS 是 **管理的**，则您的路由也必须被 **管理**。
- 如果 SSL/TLS 是 **非受管** 的，则必须在配置捆绑包中直接提供证书。

下表描述了有效的选项：

表 4.1. TLS 和路由的有效配置选项

选项	Route	TLS	提供的证书	结果
我自己的负载均衡器处理 TLS	受管	受管	否	带有默认通配符证书的边缘路由
Red Hat Quay 处理 TLS	受管	Unmanaged	是	带有挂载在 pod 中的证书的 passthrough 路由
Red Hat Quay 处理 TLS	Unmanaged	Unmanaged	是	证书是在 **quay** pod 内设置，但必须手动创建路由

4.1.1. 使用 SSL/TLS 证书和密钥对创建配置捆绑包 secret

使用以下步骤创建包含您自己的 SSL/TLS 证书和密钥对的配置捆绑包 secret。

流程

- 输入以下命令来创建包含您自己的 SSL/TLS 证书和密钥对的配置捆绑包 secret：

```
$ oc create secret generic --from-file config.yaml=./config.yaml --from-file ssl.cert=./ssl.cert --from-file ssl.key=./ssl.key config-bundle-secret
```

第 5 章 在 OPENSHIFT CONTAINER PLATFORM 上为受管组件配置资源

您可以为运行 pod 的以下组件手动调整 OpenShift Container Platform 上的 Red Hat Quay 上的资源：

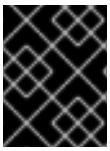
- **Quay**
- **Clair**
- **镜像**
- **clairpostgres**
- **postgres**

此功能允许用户运行较小的测试集群，或者请求更多资源前期，以避免部分降级的 **Quay** pod。限制和请求可根据 Kubernetes 资源单元 设置。

以下组件不应小于其最低要求。这可能会导致部署出现问题，并在某些情况下会导致 pod 部署失败。

- **Quay** ：最小 6 GB、2vCPU
- **Clair** ：推荐 2 GB 内存，2 个 vCPU
- **clairpostgres** ：最小 200 MB

您可以在 OpenShift Container Platform UI 上配置资源请求，或通过更新 **QuayRegistry** YAML 来直接配置资源请求。



重要

为这些组件设置的默认值是推荐的值。设置资源请求过高或低率可能会导致资源利用率低效，或者性能下降。

5.1. 使用 OPENSHIFT CONTAINER PLATFORM UI 配置资源请求

使用以下步骤使用 OpenShift Container Platform UI 配置资源。

流程

1. 在 OpenShift Container Platform 开发人员控制台中，点 **Operators → Installed Operators → Red Hat Quay**。
2. 点 **QuayRegistry**。
3. 点 registry 的名称，例如 **example-registry**。
4. 点 **YAML**。
5. 在 **spec.components** 字段中，您可以通过为 **.overrides.resources.limits** 和 **overrides.resources.requests** 字段设置值来覆盖 **quay**、**clair**、**mirror clairpostgres** 和 **postgres** 资源的资源。例如：

```
spec:
  components:
```

```
  - kind: clair
    managed: true
    overrides:
      resources:
        limits:
          cpu: "5"     # Limiting to 5 CPU (equivalent to 5000m or 5000 millicpu)
          memory: "18Gi"  # Limiting to 18 Gibibytes of memory
        requests:
          cpu: "4"     # Requesting 4 CPU
          memory: "4Gi"   # Requesting 4 Gibibytes of memory
  - kind: postgres
    managed: true
    overrides:
      resources:
        limits: {} (1)
        requests:
          cpu: "700m"   # Requesting 700 millicpu or 0.7 CPU
          memory: "4Gi"   # Requesting 4 Gibibytes of memory
  - kind: mirror
    managed: true
    overrides:
      resources:
        limits: (2)
        requests:
          cpu: "800m"   # Requesting 800 millicpu or 0.8 CPU
          memory: "1Gi"   # Requesting 1 Gibibyte of memory
  - kind: quay
    managed: true
    overrides:
      resources:
        limits:
          cpu: "4"    # Limiting to 4 CPU
          memory: "10Gi"   # Limiting to 10 Gibibytes of memory
        requests:
          cpu: "4"   # Requesting 4 CPU
          memory: "10Gi"   # Requesting 10 Gibi of memory
  - kind: clairpostgres
    managed: true
    overrides:
      resources:
        limits:
          cpu: "800m"   # Limiting to 800 millicpu or 0.8 CPU
          memory: "3Gi"   # Limiting to 3 Gibibytes of memory
        requests: {}
```

(1) 将 **limits** 或 **requests** 字段设置为 **{}** 使用这些资源的默认值。

(2) 将 **limits** 或 **requests** 字段留空会使这些资源没有限制。

6. 可选。覆盖默认值后，您可以使用 null （由 **{}** 表示）重置回分配给组件的默认值。例如：

```
# ...
  - kind: clairpostgres
    managed: true
    overrides:
```

```
      resources:
        limits: {}
        requests: {}
# ...
```

5.2. 通过编辑 QUAYREGISTRY YAML 配置资源请求

您可以在部署 registry 后重新配置 Red Hat Quay 来配置资源请求。这可以通过直接编辑 **QuayRegistry** YAML 文件，然后重新部署 registry 来完成。

流程

1. 可选：如果您没有 **QuayRegistry** YAML 文件的本地副本，请输入以下命令来获取该文件：

```
$ oc get quayregistry <registry_name> -n <namespace> -o yaml > quayregistry.yaml
```

2. 打开此流程步骤 1 中创建的 **quayregistry.yaml**，并进行所需的更改。例如：

```
  - kind: quay
    managed: true
    overrides:
      resources:
        limits: {}
        requests:
          cpu: "0.7"   # Requesting 0.7 CPU (equivalent to 500m or 500 millicpu)
          memory: "512Mi"   # Requesting 512 Mebibytes of memory
```

3. 保存更改。

4. 运行以下命令，使用更新的配置应用 Red Hat Quay registry：

```
$ oc replace -f quayregistry.yaml
```

输出示例

```
quayregistry.quay.redhat.com/example-registry replaced
```

第 6 章 配置数据库

6.1. 使用现有的 POSTGRESQL 数据库

如果使用外部管理的 PostgreSQL 数据库，则必须手动启用 **pg_trgm** 扩展才能成功部署。

使用以下步骤部署现有的 PostgreSQL 数据库。

流程

1. 使用所需的数据库字段创建 **config.yaml** 文件。例如：

   config.yaml 文件示例：

   ```
   DB_URI: postgresql://test-quay-database:postgres@test-quay-database:5432/test-quay-database
   ```

2. 使用配置文件创建 **Secret** ：

   ```
   $ kubectl create secret generic --from-file config.yaml=./config.yaml config-bundle-secret
   ```

3. 创建一个 **QuayRegistry.yaml** 文件，该文件将 **postgres** 组件标记为 **非受管**，并引用创建的 **Secret**。例如：

   quayregistry.yaml 文件示例

   ```
   apiVersion: quay.redhat.com/v1
   kind: QuayRegistry
   metadata:
     name: example-registry
     namespace: quay-enterprise
   spec:
     configBundleSecret: config-bundle-secret
     components:
       - kind: postgres
         managed: false
   ```

后续步骤

- 继续以下部分来部署 registry。

6.1.1. 数据库配置

本节论述了 Red Hat Quay 部署可用的数据库配置字段。

6.1.1.1. 数据库 URI

使用 Red Hat Quay 时，使用所需的 **DB_URI** 字段配置与数据库的连接。

下表描述了 **DB_URI** 配置字段：

表 6.1. 数据库 URI

字段	类型	描述
DB_URI （必需）	字符串	用于访问数据库的 URI，包括任何凭据。 **DB_URI** 字段示例： postgresql://quayuser:quaypass@quay-server.example.com:5432/quay

6.1.1.2. 数据库连接参数

可选的连接参数由 **DB_CONNECTION_ARGS** 参数配置。**DB_CONNECTION_ARGS** 下定义的部分键值对是通用的，另一些则特定于数据库。

下表描述了数据库连接参数：

表 6.2. 数据库连接参数

字段	类型	描述
DB_CONNECTION_ARGS	对象	数据库的可选连接参数，如超时和 SSL/TLS。
.autorollback	布尔值	是否使用线程本地连接。 应该始终为 **true**
.threadlocals	布尔值	是否使用自动滚动连接。 应该始终为 **true**

6.1.1.2.1. PostgreSQL SSL/TLS 连接参数

使用 SSL/TLS 时，配置取决于您要部署的数据库。以下示例显示了 PostgreSQL SSL/TLS 配置：

```
DB_CONNECTION_ARGS:
  sslmode: verify-ca
  sslrootcert: /path/to/cacert
```

sslmode 选项决定是否或具有什么优先级，安全 SSL/TLS TCP/IP 连接将与服务器协商。有六个模式：

表 6.3. SSL/TLS 选项

模式	描述
disable	您的配置只尝试非 SSL/TLS 连接。
allow	您的配置首先尝试非 SSL/TLS 连接。失败时，尝试 SSL/TLS 连接。

模式	描述
首选 （默认）	您的配置首先尝试 SSL/TLS 连接。失败时，尝试非 SSL/TLS 连接。
require	您的配置只尝试 SSL/TLS 连接。如果存在 root CA 文件，它会以与指定 verify-ca 相同的方式验证证书。
verify-ca	您的配置只尝试 SSL/TLS 连接，并验证服务器证书是否由可信证书颁发机构(CA)发布。
verify-full	只尝试 SSL/TLS 连接，并验证服务器证书是否由可信 CA 发布，并且请求的服务器主机名是否与证书中的匹配。

有关 PostgreSQL 的有效参数的更多信息，请参阅 Database Connection Control Functions。

6.1.1.2.2. MySQL SSL/TLS 连接参数

以下示例显示了 MySQL SSL/TLS 配置示例：

```
DB_CONNECTION_ARGS:
  ssl:
    ca: /path/to/cacert
```

有关 MySQL 的有效连接参数的信息，请访问 使用类似URI的字符串或键-值对连接到服务器。

6.1.2. 使用管理的 PostgreSQL 数据库

使用 Red Hat Quay 3.9，如果您的数据库由 Red Hat Quay Operator 管理，从 Red Hat Quay 3.8 → 3.9 更新会自动处理将 PostgreSQL 10 升级到 PostgreSQL 13。

重要

- 需要具有受管数据库的用户从 10 → 13 升级其 PostgreSQL 数据库。
- 如果您的 Red Hat Quay 和 Clair 数据库由 Operator 管理，则每个组件的数据库升级必须成功升级 3.9.0。如果任何一个数据库升级失败，则整个 Red Hat Quay 版本升级会失败。这是预期的行为。

如果您不希望 Red Hat Quay Operator 从 PostgreSQL 10 → 13 升级 PostgreSQL 部署，则必须在 **quayregistry.yaml** 文件中将 PostgreSQL 参数设置为 **managed: false**。有关将数据库设置为非受管的更多信息，请参阅使用现有的 Postgres 数据库。

重要

- 强烈建议您升级到 PostgreSQL 13。PostgreSQL 10 在 2022 年 11 月 10 日有其最终发行版本，不再被支持。如需更多信息，请参阅 PostgreSQL 版本策略。

如果您希望 PostgreSQL 数据库与 Red Hat Enterprise Linux (RHEL)系统相同的版本匹配，请参阅 迁移到 RHEL 8 的 RHEL 8 版本的 PostgreSQL，或迁移到 RHEL 9 的 RHEL 9 版本 。

如需有关 Red Hat Quay 3.8 → 3.9 流程的更多信息，请参阅升级 Red Hat Quay Operator 概述 。

6.1.2.1. PostgreSQL 数据库建议

Red Hat Quay 团队建议以下内容来管理 PostgreSQL 数据库。

- 数据库备份应该使用 PostgreSQL 镜像中提供的工具或您自己的备份基础架构定期执行。Red Hat Quay Operator 目前不会确保 PostgreSQL 数据库已备份。
- 必须使用 PostgreSQL 工具和流程从备份中恢复 PostgreSQL 数据库。请注意，在数据库恢复过程中，您的 **Quay** Pod 不应运行。
- 数据库磁盘空间由带有 50 GiB 的 Red Hat Quay Operator 自动分配。这个数字代表了大多数中小型 Red Hat Quay 安装可用存储量，但可能不适用于您的用例。Red Hat Quay Operator 目前不会处理数据库卷的大小。

6.2. 配置外部 REDIS

使用本节中的内容设置外部 Redis 部署。

6.2.1. 使用非受管 Redis 数据库

使用以下步骤设置外部 Redis 数据库。

流程

1. 使用以下 Redis 字段创建 **config.yaml** 文件：

```
# ...
BUILDLOGS_REDIS:
    host: <quay-server.example.com>
    port: 6379
    ssl: false
# ...
USER_EVENTS_REDIS:
    host: <quay-server.example.com>
    port: 6379
    ssl: false
# ...
```

2. 输入以下命令使用配置文件创建 secret：

```
$ oc create secret generic --from-file config.yaml=./config.yaml config-bundle-secret
```

3. 创建一个 **quayregistry.yaml** 文件，将 Redis 组件设置为 **非受管** 并引用所创建的 secret：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
```

```
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: redis
      managed: false
# ...
```

4. 部署 Red Hat Quay registry。

其他资源

Redis 配置字段

6.2.2. 使用非受管 Horizontal Pod Autoscaler

现在，**Clair**、**Quay** 和 **Mirror** pod 包括了 Pod 横向自动扩展(HPA)，以便在负载激增过程中自动扩展。

由于 HPA 默认配置为被管理，因此 **Clair**、**Quay** 和 **Mirror** pod 的数量被设置为 2。这有助于在通过 Operator 更新或重新调度事件期间更新或重新配置 Red Hat Quay 时停机。

6.2.2.1. 禁用 Horizontal Pod Autoscaler

要禁用自动扩展或创建自己的 **HorizontalPodAutoscaler**，请在 **QuayRegistry** 实例中将组件指定为 **unmanaged**。例如：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  components:
    - kind: horizontalpodautoscaler
      managed: false
# ...
```

6.2.3. 禁用 Route 组件

使用以下步骤防止 Red Hat Quay Operator 创建路由。

流程

1. 在 **quayregistry.yaml** 文件中将组件设置为 **managed: false** ：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  components:
    - kind: route
      managed: false
```

2. 编辑 **config.yaml** 文件，以指定 Red Hat Quay 处理 SSL/TLS。例如：

```
# ...
EXTERNAL_TLS_TERMINATION: false
# ...
SERVER_HOSTNAME: example-registry-quay-quay-enterprise.apps.user1.example.com
# ...
PREFERRED_URL_SCHEME: https
# ...
```

如果您没有正确配置非受管路由，则返回以下错误：

```
{
  {
    "kind":"QuayRegistry",
    "namespace":"quay-enterprise",
    "name":"example-registry",
    "uid":"d5879ba5-cc92-406c-ba62-8b19cf56d4aa",
    "apiVersion":"quay.redhat.com/v1",
    "resourceVersion":"2418527"
  },
  "reason":"ConfigInvalid",
  "message":"required component `route` marked as unmanaged, but `configBundleSecret` is missing necessary fields"
}
```

注意

禁用默认路由意味着您现在负责 **创建路由、服务或 Ingress** 以访问 Red Hat Quay 实例。此外，您使用的任何 DNS 都必须与 Red Hat Quay 配置中的 **SERVER_HOSTNAME** 匹配。

6.2.4. 禁用监控组件

如果在单一命名空间中安装 Red Hat Quay Operator，则监控组件会自动设置为 **managed: false**。使用以下引用来显式禁用监控。

非受管监控

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  components:
    - kind: monitoring
      managed: false
```

要在这种情况下启用监控，请参阅在 单一命名空间中安装 Red Hat Quay Operator 时启用监控。

6.2.5. 禁用镜像组件

要禁用镜像，请使用以下 YAML 配置：

非受管镜像 YAML 配置示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  components:
    - kind: mirroring
      managed: false
```

第 7 章 使用 OPERATOR 部署 RED HAT QUAY

Red Hat Quay on OpenShift Container Platform 可以使用命令行界面或 OpenShift Container Platform 控制台进行部署。这些步骤基本是相同的。

7.1. 从命令行部署 RED HAT QUAY

使用以下步骤使用命令行界面(CLI)从 部署 Red Hat Quay。

先决条件

- 已使用 CLI 登录 OpenShift Container Platform。

流程

1. 输入以下命令创建一个命名空间，如 **quay-enterprise** ：

```
$ oc new-project quay-enterprise
```

2. 可选。如果要预配置 Red Hat Quay 部署的任何方面，请为配置捆绑包创建一个 **Secret** ：

```
$ oc create secret generic quay-enterprise-config-bundle --from-file=config-
bundle.tar.gz=/path/to/config-bundle.tar.gz
```

3. 在名为 **quayregistry.yaml**的文件中创建 **QuayRegistry** 自定义资源

   a. 对于最小部署，使用所有默认值：

   quayregistry.yaml:

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
```

   b. 可选。如果要有一些未管理的组件，请在 **spec** 字段中添加此信息。最小部署可能类似以下示例：

   带有非受管组件的 quayregistry.yaml 示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  components:
    - kind: clair
      managed: false
    - kind: horizontalpodautoscaler
      managed: false
    - kind: mirror
```

```
    managed: false
  - kind: monitoring
    managed: false
```

c. 可选。如果您已创建了配置捆绑包，如 **init-config-bundle-secret**，请在 **quayregistry.yaml** 文件中引用它：

带有配置捆绑包的 quayregistry.yaml 示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  configBundleSecret: init-config-bundle-secret
```

d. 可选。如果您配置了代理，您可以使用 Red Hat Quay、Clair 和 mirror 覆盖添加信息：

配置了代理的 quayregistry.yaml 示例

```
  kind: QuayRegistry
  metadata:
    name: quay37
  spec:
    configBundleSecret: config-bundle-secret
    components:
      - kind: objectstorage
        managed: false
      - kind: route
        managed: true
      - kind: mirror
        managed: true
        overrides:
          env:
            - name: DEBUGLOG
              value: "true"
            - name: HTTP_PROXY
              value: quayproxy.qe.devcluster.openshift.com:3128
            - name: HTTPS_PROXY
              value: quayproxy.qe.devcluster.openshift.com:3128
            - name: NO_PROXY
              value:
svc.cluster.local,localhost,quay370.apps.quayperf370.perfscale.devcluster.openshift.com
      - kind: tls
        managed: false
      - kind: clair
        managed: true
        overrides:
          env:
            - name: HTTP_PROXY
              value: quayproxy.qe.devcluster.openshift.com:3128
            - name: HTTPS_PROXY
              value: quayproxy.qe.devcluster.openshift.com:3128
            - name: NO_PROXY
```

```
            value:
svc.cluster.local,localhost,quay370.apps.quayperf370.perfscale.devcluster.openshift.com
      - kind: quay
        managed: true
        overrides:
          env:
            - name: DEBUGLOG
              value: "true"
            - name: NO_PROXY
              value:
svc.cluster.local,localhost,quay370.apps.quayperf370.perfscale.devcluster.openshift.com
            - name: HTTP_PROXY
              value: quayproxy.qe.devcluster.openshift.com:3128
            - name: HTTPS_PROXY
              value: quayproxy.qe.devcluster.openshift.com:3128
```

4. 输入以下命令在指定命名空间中创建 **QuayRegistry** ：

```
$ oc create -n quay-enterprise -f quayregistry.yaml
```

5. 输入以下命令来查看何时填充 **status.registryEndpoint** ：

```
$ oc get quayregistry -n quay-enterprise example-registry -o jsonpath="
{.status.registryEndpoint}" -w
```

其他资源

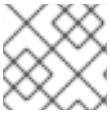
- 有关如何跟踪 Red Hat Quay 部署的进度的更多信息，请参阅监控和调试部署过程。

7.1.1. 使用 API 创建第一个用户

使用以下步骤在 Red Hat Quay 组织中创建第一个用户。

先决条件

- 配置选项 **FEATURE_USER_INITIALIZE** 必须设置为 **true**。
- 数据库中不能已存在任何用户。



流程

此流程通过指定 **"access_token": true** 来请求 OAuth 令牌。

1. 打开 Red Hat Quay 配置文件并更新以下配置字段：

```
FEATURE_USER_INITIALIZE: true
SUPER_USERS:
    - quayadmin
```

2. 输入以下命令停止 Red Hat Quay 服务：

```
$ sudo podman stop quay
```

3. 输入以下命令启动 Red Hat Quay 服务：

```
$ sudo podman run -d -p 80:8080 -p 443:8443 --name=quay -v $QUAY/config:/conf/stack:Z
-v $QUAY/storage:/datastorage:Z {productrepo}/{quayimage}:{productminv}
```

4. 运行以下 **CURL** 命令以使用用户名、密码、电子邮件和访问令牌生成新用户：

```
$ curl -X POST -k  http://quay-server.example.com/api/v1/user/initialize --header 'Content-
Type: application/json' --data '{ "username": "quayadmin", "password":"quaypass12345",
"email": "quayadmin@example.com", "access_token": true}'
```

如果成功，命令会返回带有用户名、电子邮件和加密密码的对象。例如：

```
{"access_token":"6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED",
"email":"quayadmin@example.com","encrypted_password":"1nZMLH57RIE5UGdL/yYpDOHL
qiNCgimb6W9kfF8MjZ1xrfDpRyRs9NUnUuNuAitW","username":"quayadmin"} #
gitleaks:allow
```

如果用户已在数据库中，则返回错误：

```
{"message":"Cannot initialize user in a non-empty database"}
```

如果您的密码至少不是八个字符或包含空格，则返回错误：

```
{"message":"Failed to initialize user: Invalid password, password must be at least 8
characters and contain no whitespace."}
```

5. 输入以下命令登录到您的 Red Hat Quay 部署：

```
$ sudo podman login -u quayadmin -p quaypass12345 http://quay-server.example.com --tls-
verify=false
```

输出示例

```
Login Succeeded!
```

7.1.2. 使用命令行查看创建的组件

使用以下步骤查看部署的 Red Hat Quay 组件。

先决条件

- 您已在 OpenShift Container Platform 上部署了 Red Hat Quay。

流程

1. 输入以下命令查看部署的组件：

```
$ oc get pods -n quay-enterprise
```

输出示例

-

```
NAME                                               READY   STATUS      RESTARTS   AGE
example-registry-clair-app-5ffc9f77d6-jwr9s        1/1     Running     0          3m42s
example-registry-clair-app-5ffc9f77d6-wgp7d        1/1     Running     0          3m41s
example-registry-clair-postgres-54956d6d9c-rgs8l   1/1     Running     0          3m5s
example-registry-quay-app-79c6b86c7b-8qnr2         1/1     Running     4          3m42s
example-registry-quay-app-79c6b86c7b-xk85f         1/1     Running     4          3m41s
example-registry-quay-app-upgrade-5kl5r            0/1     Completed   4          3m50s
example-registry-quay-database-b466fc4d7-tfrnx     1/1     Running     2          3m42s
example-registry-quay-mirror-6d9bd78756-6lj6p      1/1     Running     0          2m58s
example-registry-quay-mirror-6d9bd78756-bv6gq      1/1     Running     0          2m58s
example-registry-quay-postgres-init-dzbmx          0/1     Completed   0          3m43s
example-registry-quay-redis-8bd67b647-skgqx        1/1     Running     0          3m42s
```

7.1.3. Pod 横向自动扩展

默认部署显示以下正在运行的 pod：

- 两个用于 Red Hat Quay 应用程序本身的 pod (**example-registry-quay-app可以'**)
- 一个 Red Hat Quay 日志记录的 Redis pod (**example-registry-quay-redis**可以)
- 一个数据库 pod for PostgreSQL，供 Red Hat Quay 用于元数据存储(**example-registry-quay-database114**)
- 两个 **Quay** 镜像 pod (**example-registry-quay-mirror114**)
- Clair 应用程序的两个 pod (**example-registry-clair-app114**)
- 用于 Clair 的 PostgreSQL pod (**example-registry-clair-postgres suppress**)

水平 PPod 自动扩展默认配置为 **受管**，Quay 的 pod 数量，Clair 和存储库镜像设置为 2。这有助于在通过 Red Hat Quay Operator 更新或重新调度事件期间更新或重新配置 Red Hat Quay 时停机。您可以输入以下命令来查看 HPA 对象的信息：

```
$ oc get hpa -n quay-enterprise
```

输出示例

```
NAME                       REFERENCE                             TARGETS          MINPODS   MAXPODS
REPLICAS   AGE
example-registry-clair-app     Deployment/example-registry-clair-app     16%/90%, 0%/90%   2
10        2          13d
example-registry-quay-app      Deployment/example-registry-quay-app      31%/90%, 1%/90%   2
20        2          13d
example-registry-quay-mirror   Deployment/example-registry-quay-mirror   27%/90%, 0%/90%   2
20        2          13d
```

其他资源

如需有关预配置 Red Hat Quay 部署的更多信息，请参阅 为自动化配置 Red Hat Quay 部分

7.1.4. 监控和调试部署过程

用户现在可以在部署阶段排除问题。**QuayRegistry** 对象中的状态可帮助您在部署期间监控组件的健康状况，以帮助您调试可能会出现任何问题。

流程

1. 输入以下命令检查部署的状态：

```
$ oc get quayregistry -n quay-enterprise -o yaml
```

输出示例

部署后，Quay **Registry** 对象将显示基本配置：

```
apiVersion: v1
items:
- apiVersion: quay.redhat.com/v1
  kind: QuayRegistry
  metadata:
    creationTimestamp: "2021-09-14T10:51:22Z"
    generation: 3
    name: example-registry
    namespace: quay-enterprise
    resourceVersion: "50147"
    selfLink: /apis/quay.redhat.com/v1/namespaces/quay-enterprise/quayregistries/example-
registry
    uid: e3fc82ba-e716-4646-bb0f-63c26d05e00e
  spec:
    components:
    - kind: postgres
      managed: true
    - kind: clair
      managed: true
    - kind: redis
      managed: true
    - kind: horizontalpodautoscaler
      managed: true
    - kind: objectstorage
      managed: true
    - kind: route
      managed: true
    - kind: mirror
      managed: true
    - kind: monitoring
      managed: true
    - kind: tls
      managed: true
    - kind: clairpostgres
      managed: true
    configBundleSecret: example-registry-config-bundle-kt55s
kind: List
metadata:
  resourceVersion: ""
  selfLink: ""
```

2. 使用 **oc get pods** 命令查看部署组件的当前状态：

```
$ oc get pods -n quay-enterprise
```

输出示例

```
NAME                                             READY   STATUS              RESTARTS   AGE
example-registry-clair-app-86554c6b49-ds7bl      0/1     ContainerCreating   0          2s
example-registry-clair-app-86554c6b49-hxp5s      0/1     Running             1          17s
example-registry-clair-postgres-68d8857899-lbc5n 0/1     ContainerCreating   0          17s
example-registry-quay-app-upgrade-h2v7h          0/1     ContainerCreating   0          9s
example-registry-quay-database-66f495c9bc-wqsjf  0/1     ContainerCreating   0          17s
example-registry-quay-mirror-854c88457b-d845g    0/1     Init:0/1            0          2s
example-registry-quay-mirror-854c88457b-fghxv    0/1     Init:0/1            0          17s
example-registry-quay-postgres-init-bktdt        0/1     Terminating         0          17s
example-registry-quay-redis-f9b9d44bf-4htpz      0/1     ContainerCreating   0          17s
```

3. 在部署进行时，Quay **Registry** 对象将显示当前状态。在这个实例中，数据库迁移会被发生，其他组件会等到完成为止：

```
  status:
    conditions:
    - lastTransitionTime: "2021-09-14T10:52:04Z"
      lastUpdateTime: "2021-09-14T10:52:04Z"
      message: all objects created/updated successfully
      reason: ComponentsCreationSuccess
      status: "False"
      type: RolloutBlocked
    - lastTransitionTime: "2021-09-14T10:52:05Z"
      lastUpdateTime: "2021-09-14T10:52:05Z"
      message: running database migrations
      reason: MigrationsInProgress
      status: "False"
      type: Available
    lastUpdated: 2021-09-14 10:52:05.371425635 +0000 UTC
    unhealthyComponents:
      clair:
      - lastTransitionTime: "2021-09-14T10:51:32Z"
        lastUpdateTime: "2021-09-14T10:51:32Z"
        message: 'Deployment example-registry-clair-postgres: Deployment does not have
minimum availability.'
        reason: MinimumReplicasUnavailable
        status: "False"
        type: Available
      - lastTransitionTime: "2021-09-14T10:51:32Z"
        lastUpdateTime: "2021-09-14T10:51:32Z"
        message: 'Deployment example-registry-clair-app: Deployment does not have minimum
availability.'
        reason: MinimumReplicasUnavailable
        status: "False"
        type: Available
      mirror:
      - lastTransitionTime: "2021-09-14T10:51:32Z"
        lastUpdateTime: "2021-09-14T10:51:32Z"
        message: 'Deployment example-registry-quay-mirror: Deployment does not have
minimum availability.'
```

```
      reason: MinimumReplicasUnavailable
      status: "False"
      type: Available
```

4. 当部署过程成功完成时，QuayRegistry 对象中的状态不会显示不健康的组件：

```
status:
  conditions:
  - lastTransitionTime: "2021-09-14T10:52:36Z"
    lastUpdateTime: "2021-09-14T10:52:36Z"
    message: all registry component healthchecks passing
    reason: HealthChecksPassing
    status: "True"
    type: Available
  - lastTransitionTime: "2021-09-14T10:52:46Z"
    lastUpdateTime: "2021-09-14T10:52:46Z"
    message: all objects created/updated successfully
    reason: ComponentsCreationSuccess
    status: "False"
    type: RolloutBlocked
  currentVersion: {producty}
  lastUpdated: 2021-09-14 10:52:46.104181633 +0000 UTC
  registryEndpoint: https://example-registry-quay-quay-
enterprise.apps.docs.quayteam.org
  unhealthyComponents: {}
```

7.2. 从 OPENSHIFT CONTAINER PLATFORM 控制台部署 RED HAT QUAY

1. 创建一个命名空间，如 quay-enterprise。

2. 选择 Operators → Installed Operators，然后选择 Quay Operator 以导航到 Operator 详情视图。

3. 在 'Provided APIs' 下的 'quay Registry' 上点 'Create Instance'。

4. （可选）更改 QuayRegistry 的 'Name'。这将影响 registry 的主机名。所有其他字段都已填充默认值。

5. 点 'Create' 提交 QuayRegistry 以供 Quay Operator 部署。

6. 您应该被重定向到 QuayRegistry 列表视图。点您刚才创建的 QuayRegistry 以查看详情视图。

7. 在 'Registry Endpoint' 有值后，可以在 UI 中点来访问新的 Quay registry。现在，您可以选择"Create Account"来创建用户并登录。

7.2.1. 使用 Red Hat Quay UI 创建第一个用户

使用以下步骤，通过 Red Hat Quay UI 创建第一个用户。

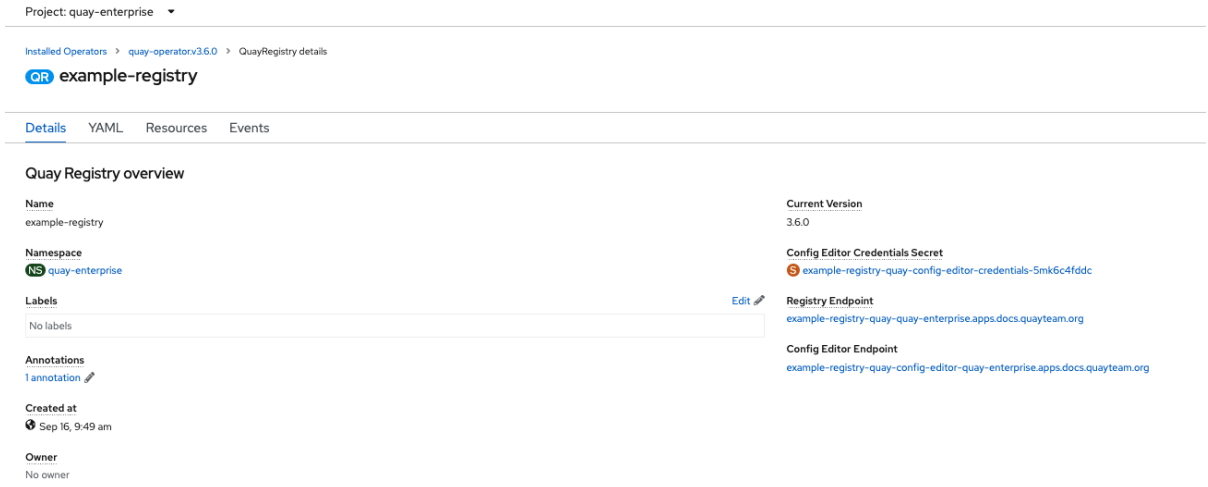
注意

此流程假设 FEATURE_USER_CREATION 配置选项没有设置为 false。如果是 false，则 UI 上的 Create Account 功能将被禁用，您必须使用 API 来创建第一个用户。

流程

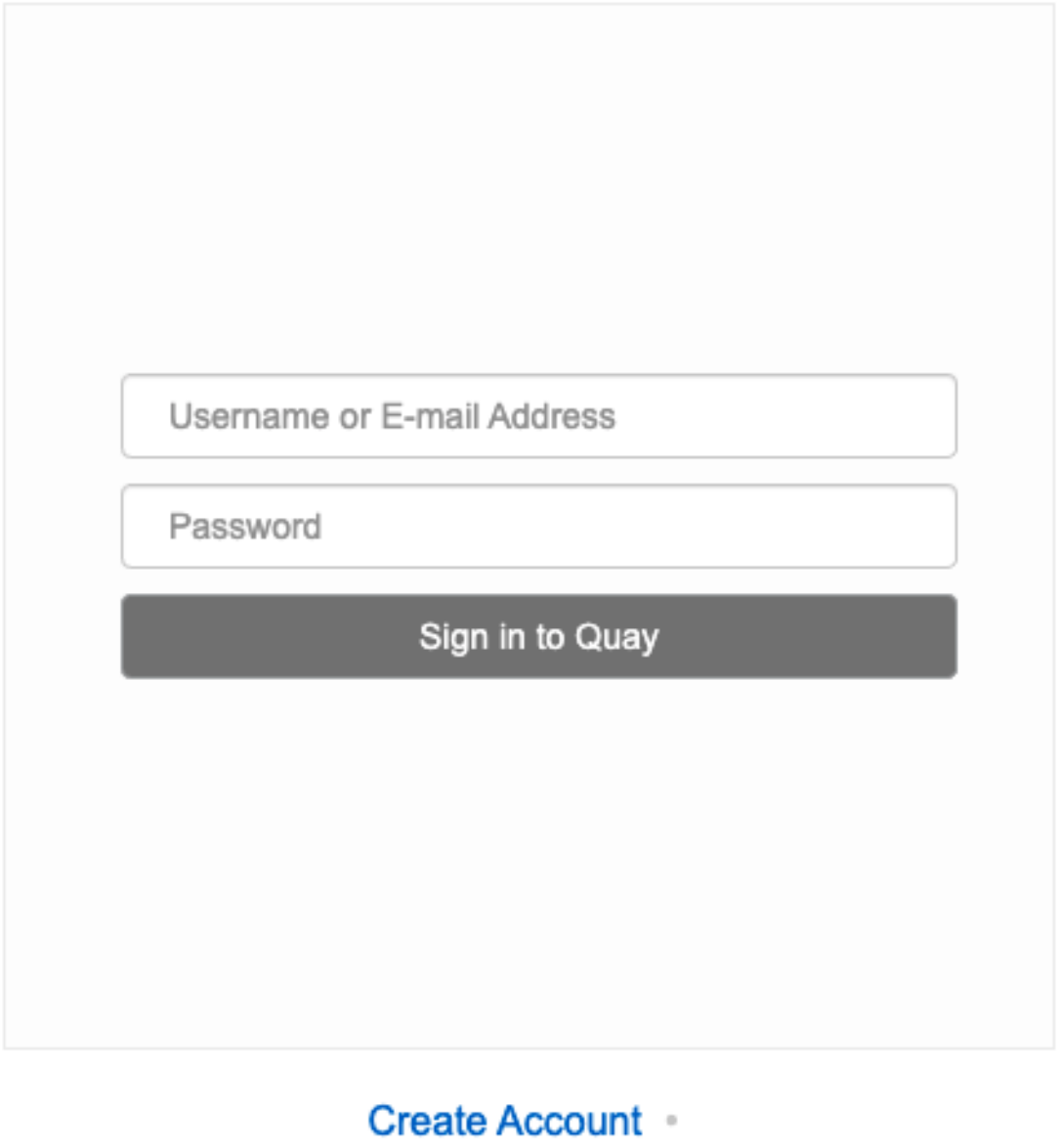
1. 在 OpenShift Container Platform 控制台中，使用适当的命名空间 / 项目导航到 Operators → Installed Operators。

2. 点新安装的 QuayRegistry 对象查看详情。例如：

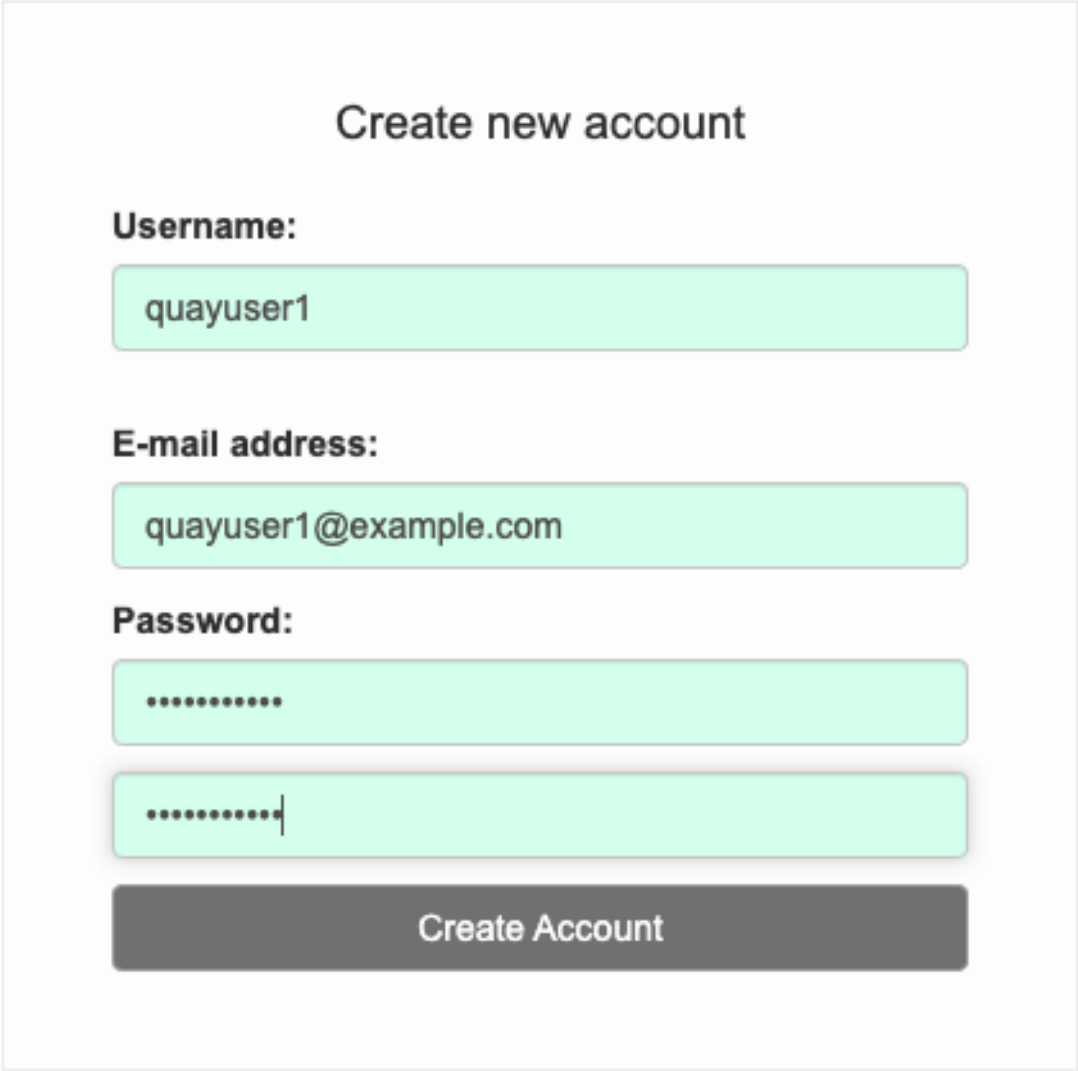


3. 在 Registry Endpoint 有值后，导航到浏览器中的这个 URL。

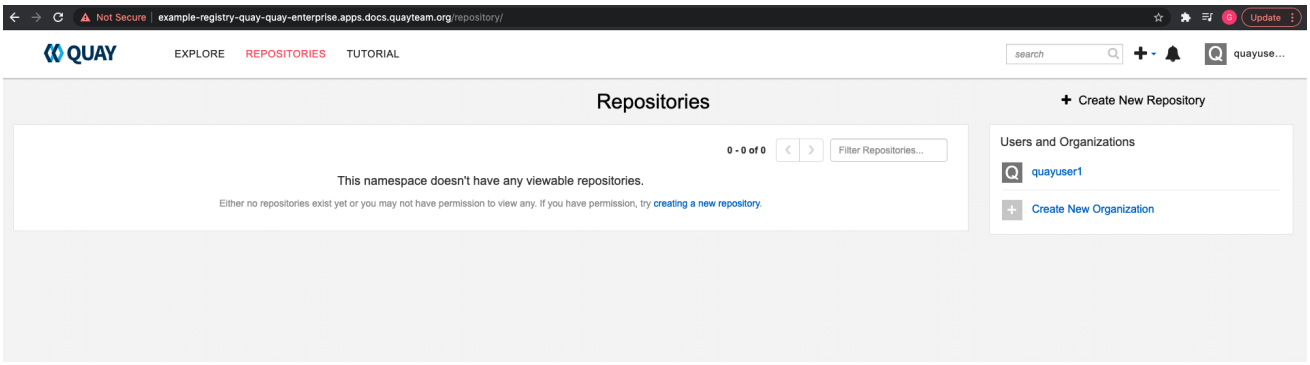
4. 在 Red Hat Quay registry UI 中，选择 Create Account 来创建用户。例如：



5. 输入用户名、密码、电子邮件的详细信息，然后单击 **Create Account**。例如：



创建第一个用户后，会自动登录到 **Red Hat Quay registry**。例如：



第 8 章 查看 QUAYREGISTRY 对象的状态

给定 Red Hat Quay 部署的生命周期可观察性会在对应的 QuayRegistry 对象的 status 部分中报告。Red Hat Quay Operator 持续更新本节，这应该是第一个位置，用于查找 Red Hat Quay 或其受管依赖项中的任何问题或状态更改。

8.1. 查看 REGISTRY 端点

当 Red Hat Quay 准备好使用后，status.registryEndpoint 字段将填充 registry 的公开可用主机名。

8.2. 查看正在使用的 RED HAT QUAY 版本

运行的 Red Hat Quay 的当前版本将报告为 status.currentVersion。

8.3. 查看 RED HAT QUAY 部署的条件

某些条件将在 status.conditions 中报告。

第 9 章 在 OPENSHIFT CONTAINER PLATFORM 上自定义 RED HAT QUAY

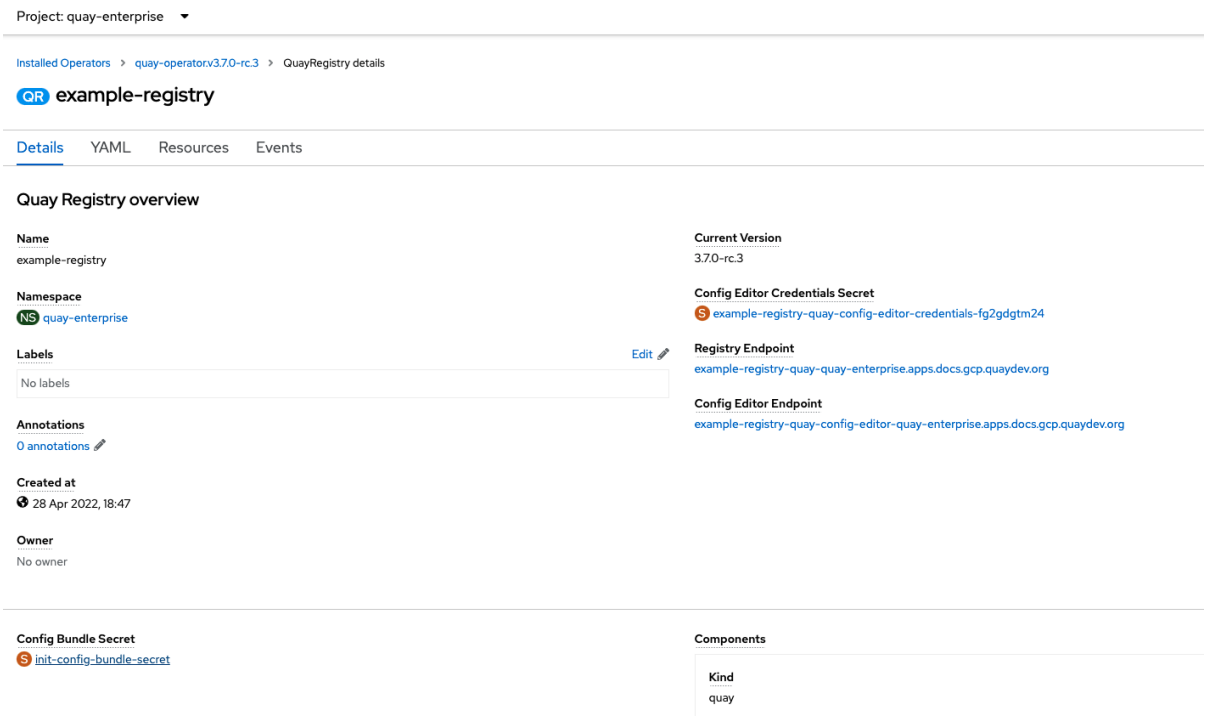
部署后，您可以通过编辑 Red Hat Quay 配置捆绑包 secret spec.configBundleSecret 来自定义 Red Hat Quay 应用程序。您还可以更改组件的受管状态，并为 QuayRegistry 资源的 spec.components 对象中某些组件配置资源请求。

9.1. 在 OPENSHIFT CONTAINER PLATFORM 控制台中编辑 CONFIG BUNDLE SECRET

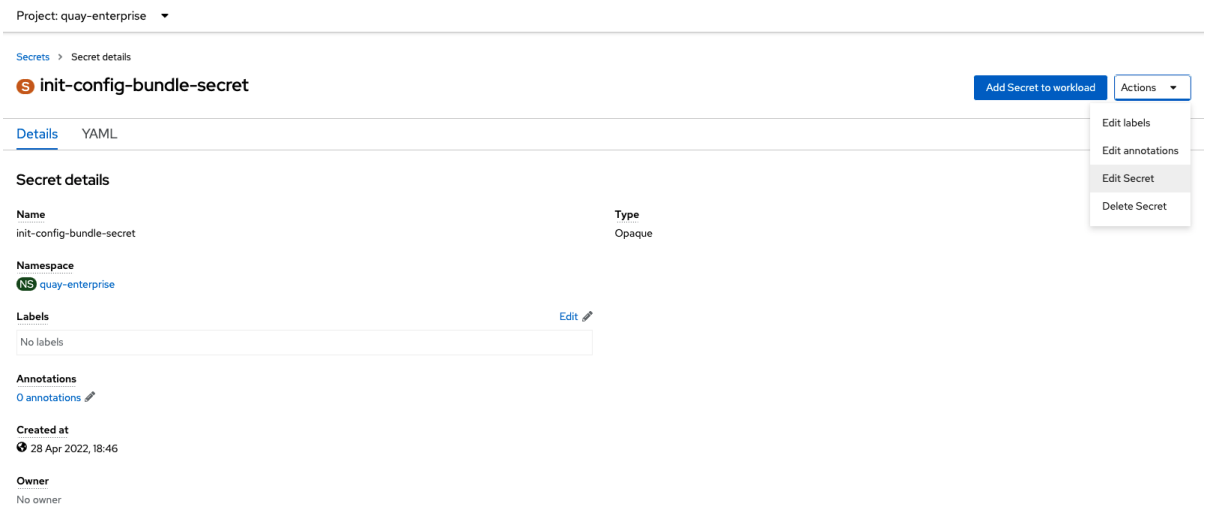
使用以下步骤编辑 OpenShift Container Platform 控制台中的配置捆绑包 secret。

流程

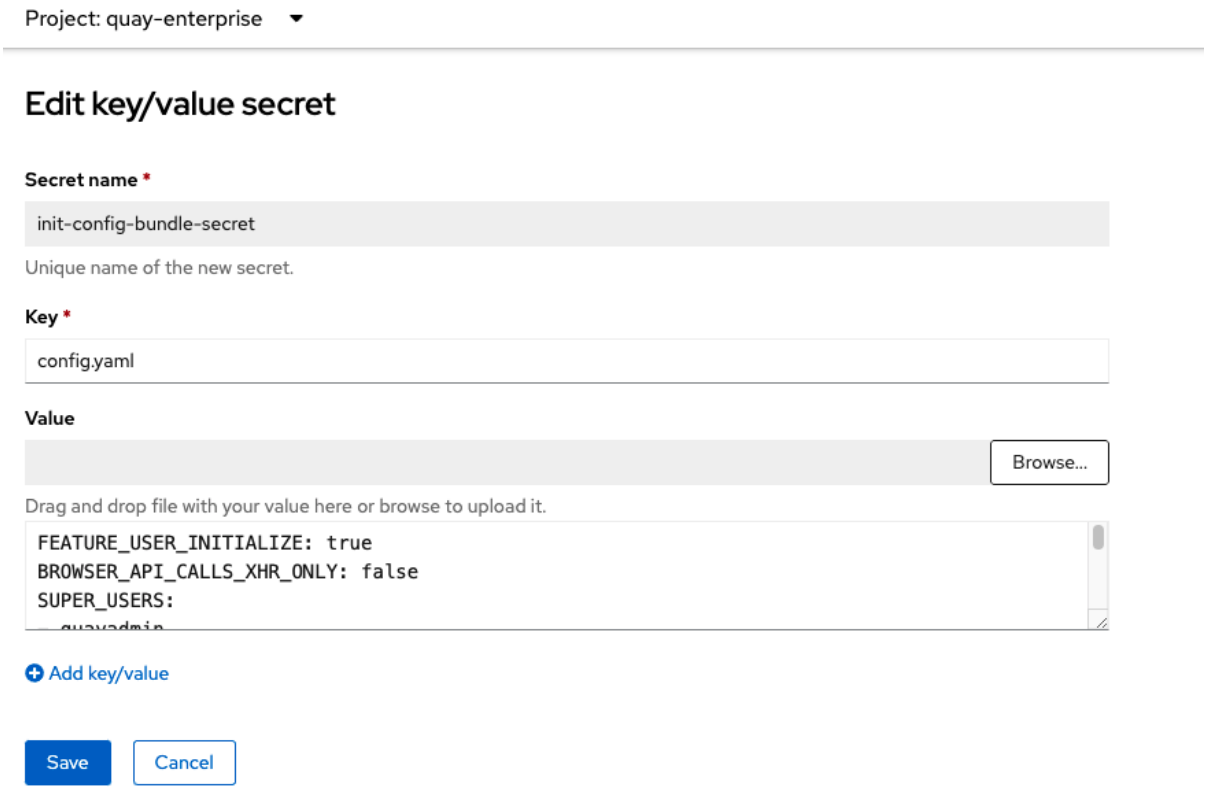
1. 在 Red Hat Quay Registry 概述屏幕上，单击 Config Bundle Secret 的链接。



2. 要编辑 secret，请点击 Actions → Edit Secret。



3. **修改配置并保存更改。**



4. **监控部署以确保成功完成，并且配置更改已生效。**

9.2. 确定 QUAYREGISTRY 端点和 SECRET

使用以下步骤查找 QuayRegistry 端点和 secret。

流程

1. 您可以通过输入以下命令来查找当前的端点和 secret，使用 oc describe quayregistry 或 oc get quayregistry -o yaml 检查 QuayRegistry 资源：

```
$ oc get quayregistry example-registry -n quay-enterprise -o yaml
```

输出示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  ...
  name: example-registry
  namespace: quay-enterprise
  ...
spec:
  components:
  - kind: quay
    managed: true
  ...
  - kind: clairpostgres
    managed: true
  configBundleSecret: init-config-bundle-secret (1)
status:
  currentVersion: 3.7.0
  lastUpdated: 2022-05-11 13:28:38.199476938 +0000 UTC
  registryEndpoint: https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org (2)
```

(1) config bundle secret，其中包含 config.yaml 文件和任何 SSL/TLS 证书。

(2) registry 的 URL，用于浏览器访问 registry UI，以及 registry API 端点。

9.3. 下载现有配置

以下流程详细介绍了如何使用不同的策略下载现有配置。

9.3.1. 使用 config bundle secret 下载现有配置

您可以使用 config bundle secret 下载现有配置。

流程

1. 输入以下命令来获取 secret 数据：

```
$ oc get secret -n quay-enterprise init-config-bundle-secret -o jsonpath='{.data}'
```

输出示例

```
{
  "config.yaml": "RkVBVFVSRV9VU0 ... MDAwMAo="
}
```

2. 输入以下命令解码数据：

```
$ echo 'RkVBVFVSRV9VU0 ... MDAwMAo=' | base64 --decode
```

输出示例

```
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- quayadmin
FEATURE_USER_CREATION: false
FEATURE_QUOTA_MANAGEMENT: true
FEATURE_PROXY_CACHE: true
FEATURE_BUILD_SUPPORT: true
DEFAULT_SYSTEM_REJECT_QUOTA_BYTES: 102400000
```

9.4. 使用配置捆绑包配置自定义 SSL/TLS 证书

您可以在初始部署前配置自定义 SSL/TLS 证书，或者在 OpenShift Container Platform 上部署 Red Hat Quay 后配置。这可以通过创建或更新配置捆绑包 secret 来完成。

如果要将证书添加到现有部署中，则必须在新配置捆绑包 secret 中包含现有的 config.yaml 文件，即使您没有进行任何更改。

使用以下步骤添加自定义 SSL/TLS 证书。

流程

1. 在 QuayRegistry YAML 文件中，将 kind: tls 设置为 managed:false，例如：

```
- kind: tls
  managed: false
```

2. 导航到 Events 页面，这应该显示更改被阻止，直到您设置适当的配置为止。例如：

```
- lastTransitionTime: '2022-03-28T12:56:49Z'
  lastUpdateTime: '2022-03-28T12:56:49Z'
  message: >-
    required component `tls` marked as unmanaged, but `configBundleSecret`
    is missing necessary fields
  reason: ConfigInvalid
  status: 'True'
```

3. 使用嵌入式数据或使用文件创建 secret。

   a. 直接将配置详情嵌入到 Secret 资源 YAML 文件中。例如：

custom-ssl-config-bundle.yaml

```
apiVersion: v1
```

```
kind: Secret
metadata:
  name: custom-ssl-config-bundle-secret
  namespace: quay-enterprise
data:
  config.yaml: |
    FEATURE_USER_INITIALIZE: true
    BROWSER_API_CALLS_XHR_ONLY: false
    SUPER_USERS:
    - quayadmin
    FEATURE_USER_CREATION: false
    FEATURE_QUOTA_MANAGEMENT: true
    FEATURE_PROXY_CACHE: true
    FEATURE_BUILD_SUPPORT: true
    DEFAULT_SYSTEM_REJECT_QUOTA_BYTES: 102400000
  extra_ca_cert_my-custom-ssl.crt: |
    -----BEGIN CERTIFICATE-----
    MIIDsDCCApigAwIBAgIUCqlzkHjF5i5TXLFy+sepFrZr/UswDQYJKoZIhvcNAQEL

BQAwbzELMAkGA1UEBhMCSUUxDzANBgNVBAgMBkdBTFdBWTEPMA0GA1UEB
wwGR0FM
    ....
    -----END CERTIFICATE-----
```

b. **从 YAML 文件创建 secret：**

```
$ oc create  -f custom-ssl-config-bundle.yaml
```

..

4. **另外，您可以创建包含所需信息的文件，然后从这些文件创建 secret。**

a. **输入以下命令创建一个包含 config.yaml 文件和 custom-ssl.crt 文件的通用 Secret 对象：**

```
$ oc create secret generic custom-ssl-config-bundle-secret \
  --from-file=config.yaml \
  --from-file=extra_ca_cert_my-custom-ssl.crt=my-custom-ssl.crt
```

b. **创建或更新 QuayRegistry YAML 文件，引用创建的 Secret，例如：**

QuayRegistry YAML 文件示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  configBundleSecret: custom-ssl-config-bundle-secret
```

c. **输入以下命令使用 YAML 文件部署或更新 registry：**

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

后续步骤

- **Red Hat Quay 功能**