Red Hat Quay 3

Red Hat Quay Operator 的功能

高级 Red Hat Quay Operator 功能

Last Updated: 2024-08-02

Red Hat Quay 3 Red Hat Quay Operator 的功能

高级 Red Hat Quay Operator 功能

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

高级 Red Hat Quay Operator 功能

目录

第 1 章 FEDERAL INFORMATION PROCESSING STANDARD (FIPS)就绪度和合规性

美国国家标准与技术研究所(NIST)开发的联邦信息处理标准(FIPS)被认为是保护和加密敏感数据的高度规范性领域，特别是银行、医疗和公共部门等高度监管区域。Red Hat Enterprise Linux (RHEL)和 OpenShift Container Platform 通过提供 *FIPS 模式* 来支持 FIPS，系统只允许使用特定 FIPS 验证的加密模块，如 **openssl**。这样可确保 FIPS 合规性。

1.1. 启用 FIPS 合规性

使用以下步骤在 Red Hat Quay 部署中启用 FIPS 合规性。

前提条件

- 如果您正在运行独立部署 Red Hat Quay，您的 Red Hat Enterprise Linux (RHEL)部署是版本 8 或更高版本，启用了 FIPS。
- 如果要在 OpenShift Container Platform 上部署 Red Hat Quay，OpenShift Container Platform 是 4.10 或更高版本。
- 您的 Red Hat Quay 版本为 3.5.0 或更高版本。
- 如果您在 IBM Power 或 IBM Z 集群上的 OpenShift Container Platform 中使用 Red Hat Quay：
  - OpenShift Container Platform 版本 4.14 或更高版本是必需的
  - Red Hat Quay 版本 3.10 或更高版本是必需的
- 您有 Red Hat Quay 部署的管理特权。

流程

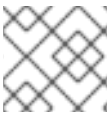
- 在 Red Hat Quay **config.yaml** 文件中，将 **FEATURE_FIPS** 配置字段设置为 **true**。例如：

```
---
FEATURE_FIPS = true
---
```

  将 **FEATURE_FIPS** 设置为 **true** 时，Red Hat Quay 会使用 FIPS 兼容的哈希功能运行。

第 2 章 控制台监控和警报

Red Hat Quay 支持从 OpenShift Container Platform 控制台内部使用 Red Hat Quay Operator 部署监控实例。新的监控功能包括 Grafana 仪表板、访问单个指标和警报，以通知频繁重启 **Quay** pod。

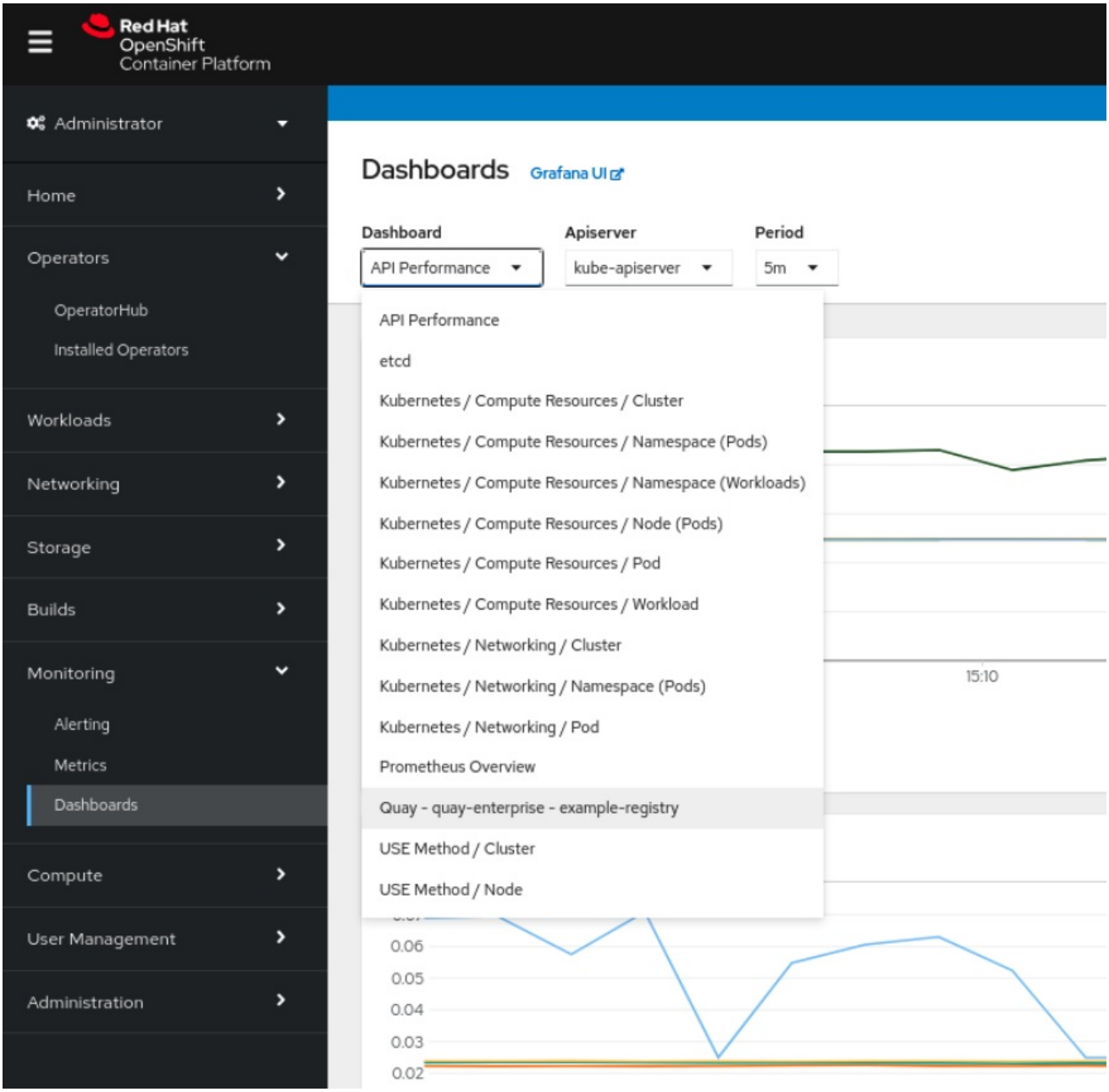


注意

要启用监控功能，必须在 **All Namespaces** 模式中安装 Red Hat Quay Operator。

2.1. DASHBOARD

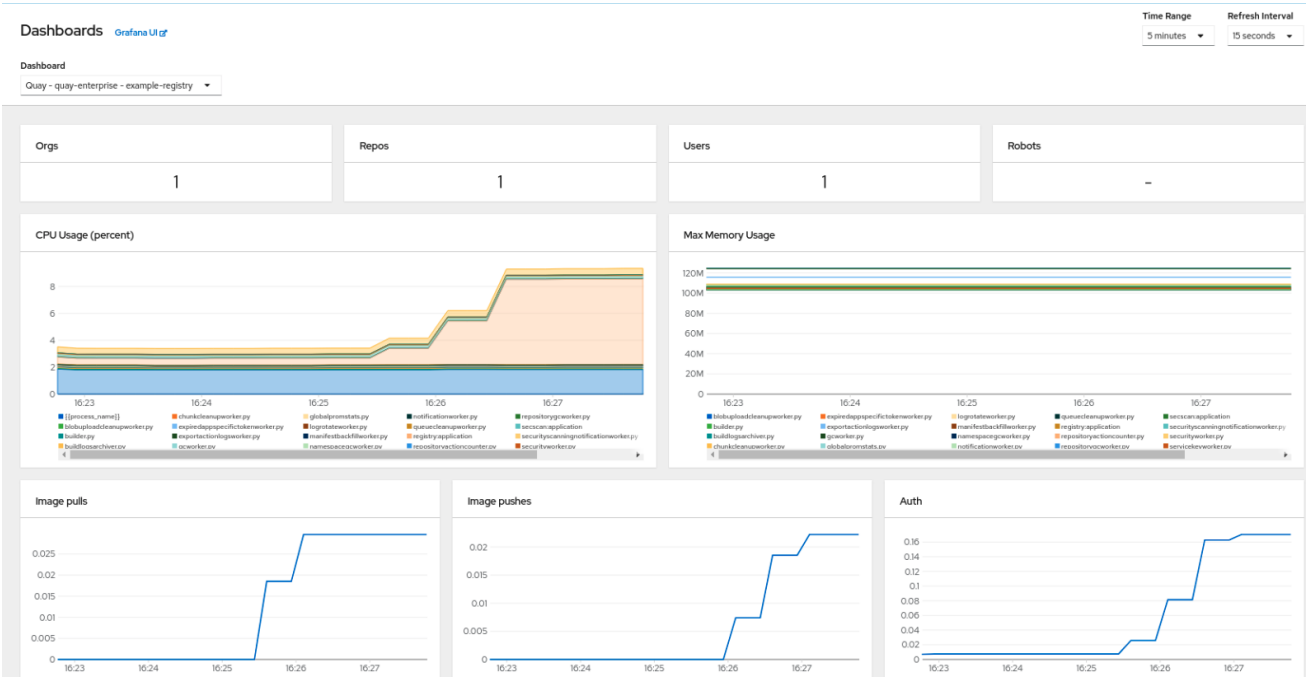
在 OpenShift Container Platform 控制台中，点 **Monitoring → Dashboards** 并搜索所需 Red Hat Quay registry 实例的仪表板：



仪表板显示各种统计信息，包括：

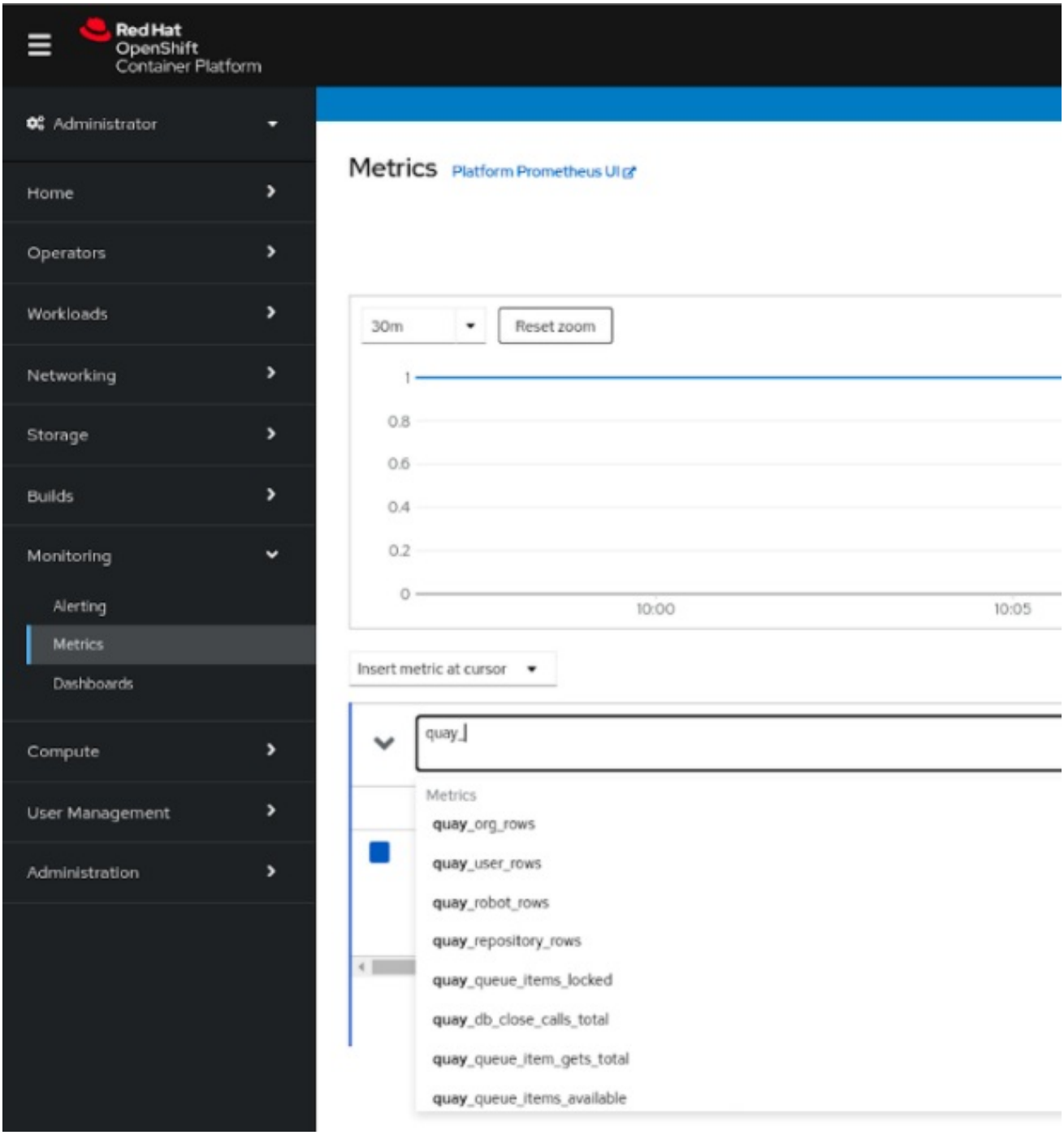
- **机构、存储库、用户和 Robot 帐户的数量**

- CPU 用量
- 最大内存用量
- 拉取和推送率，以及身份验证请求
- API 请求率
- latencies

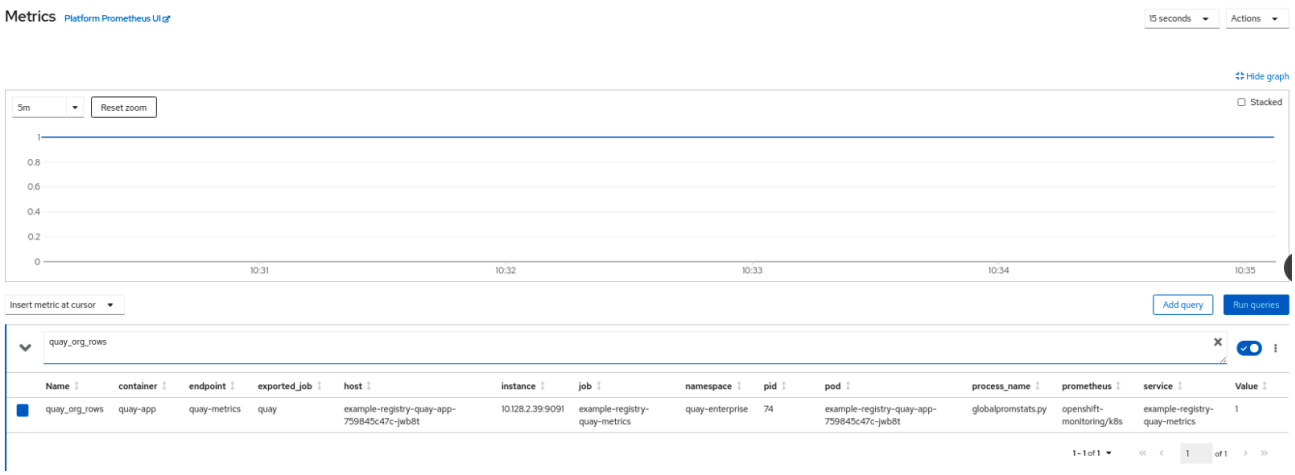


2.2. 指标

您可以通过在 UI 中访问 **Monitoring → Metrics** 来查看 Red Hat Quay 仪表板后面的底层指标。在 **Expression** 字段中，输入文本 **quay_** 以查看可用指标列表：



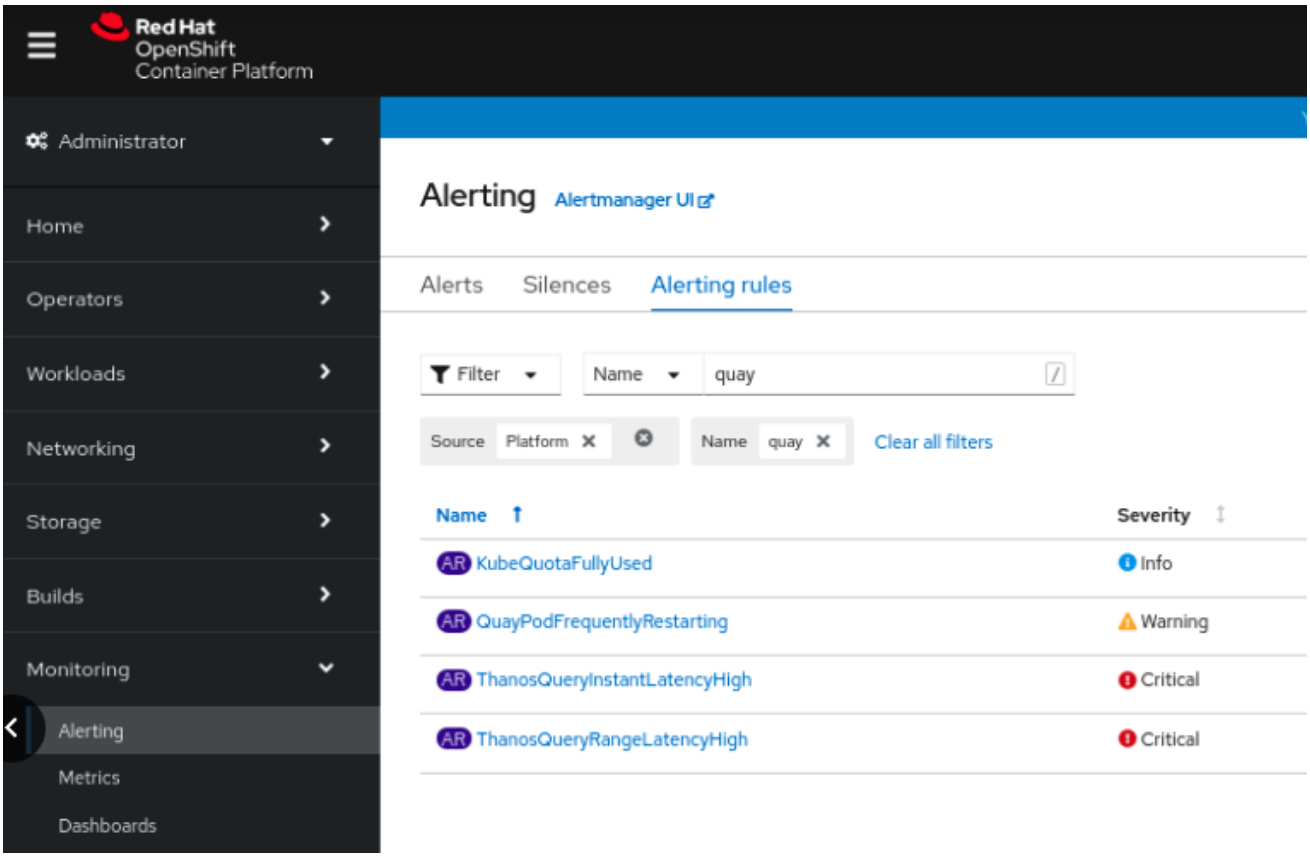
选择一个指标示例，如 **quay_org_rows** ：



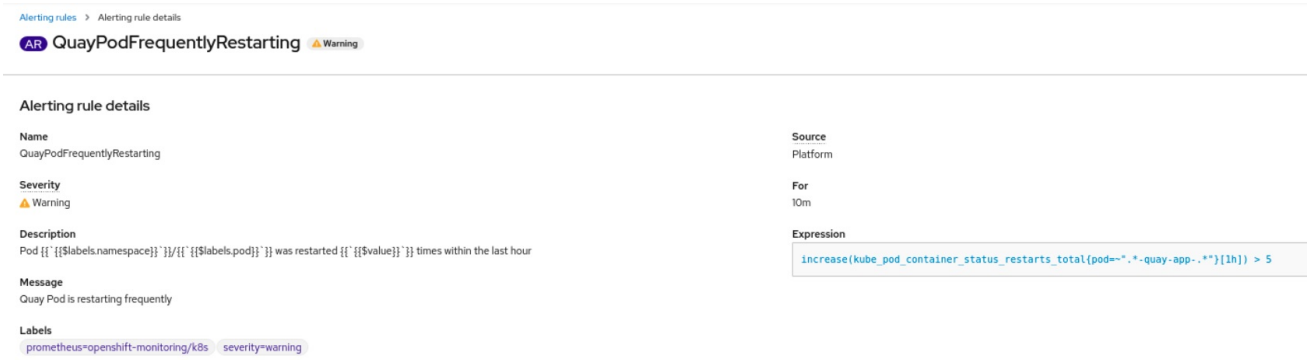
此指标显示 registry 中的机构数量。它在仪表板中也直接显示。

2.3. 警报

如果 **Quay** Pod 重启太频繁，则会引发警报。通过在控制台 UI 中从 **Monitoring → Alerting** 访问 **Alerting** rules 选项卡并搜索 Quay 特定警报，可以配置警报：



选择 **QuayPodFrequentlyRestarting** 规则详情来配置警报：



第 3 章 CLAIR 安全扫描程序

3.1. CLAIR 漏洞数据库

Clair 使用以下漏洞数据库来报告镜像中的问题：

- Ubuntu Oval 数据库
- Debian 安全跟踪器
- Red Hat Enterprise Linux (RHEL) Oval 数据库
- SUSE Oval 数据库
- Oracle Oval 数据库
- alpine SecDB 数据库
- VMware Photon OS 数据库
- Amazon Web Services (AWS) UpdateInfo
- 开源漏洞(OSV)数据库

有关 Clair 如何对不同数据库进行安全映射的信息，请参阅 Claircore Severity 映射。

3.1.1. Clair 的开源漏洞(OSV)数据库的信息

开源漏洞(OSV)是一种漏洞数据库和监控服务，侧重于跟踪和管理开源软件中的安全漏洞。

OSV 提供开源项目中已知安全漏洞的全面、最新的数据库。它涵盖了广泛的开源软件，包括库、框架和软件开发中使用的其他组件。有关包含生态系统的完整列表，请参阅 定义的生态系统。

Clair 还通过开源漏洞(OSV)数据库报告 **golang**、**java** 和 **ruby** 生态系统的漏洞和安全信息。

通过利用 OSV，开发人员和组织可以主动监控和解决其使用的开源组件中的安全漏洞，这有助于降低安全漏洞和数据遭受攻击的风险。

有关 OSV 的更多信息，请参阅 OSV 网站。

3.2. OPENSHIFT CONTAINER PLATFORM 上的 CLAIR

要在 OpenShift Container Platform 上的 Red Hat Quay 部署上设置 Clair v4 (Clair)，建议使用 Red Hat Quay Operator。默认情况下，Red Hat Quay Operator 会随 Red Hat Quay 部署一起安装或升级 Clair 部署，并自动配置 Clair。

3.3. 测试 CLAIR

使用以下步骤在独立 Red Hat Quay 部署或基于 OpenShift Container Platform Operator 的部署中测试 Clair。

先决条件

- 您已部署了 Clair 容器镜像。

流程

1. 输入以下命令拉取示例镜像：

```
$ podman pull ubuntu:20.04
```

2. 输入以下命令将镜像标记到 registry：

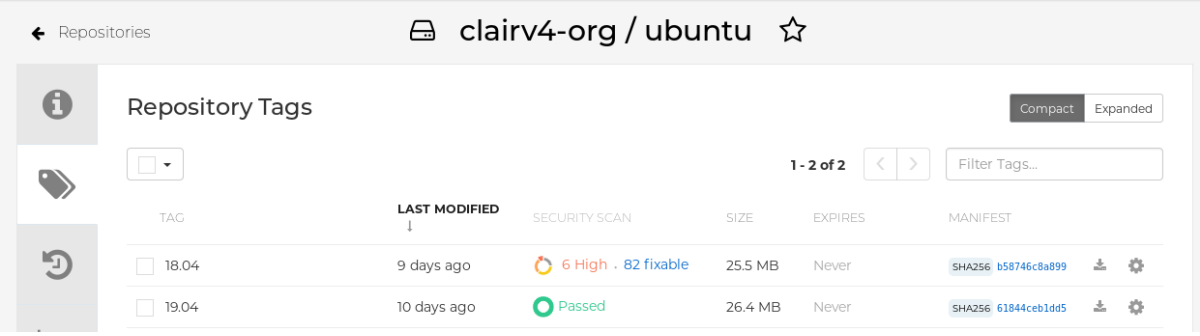
```
$ sudo podman tag docker.io/library/ubuntu:20.04 <quay-server.example.com>/<user-name>/ubuntu:20.04
```

3. 输入以下命令将镜像推送到 Red Hat Quay registry：

```
$ sudo podman push --tls-verify=false quay-server.example.com/quayadmin/ubuntu:20.04
```

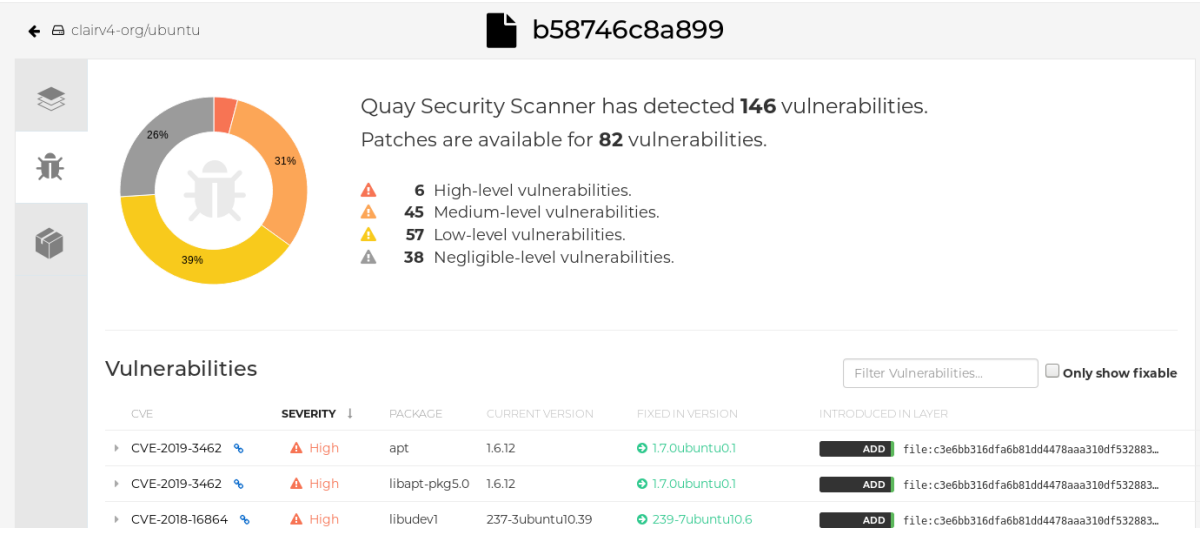
4. 通过 UI 登录您的 Red Hat Quay 部署。
5. 单击存储库名称，如 **quayadmin/ubuntu**。
6. 在导航窗格中，单击 **Tags**。

报告概述



7. 点镜像报告（如 **45 介质**）来显示更详细的报告：

报告详情



注意

在某些情况下，Clair 会显示镜像重复报告，如 **ubi8/nodejs-12** 或 **ubi8/nodejs-16**。这是因为名称相同的漏洞用于不同的软件包。这个行为应该带有 Clair 漏洞报告，且不会作为程序错误解决。

3.4. 高级 CLAIR 配置

使用以下部分中的步骤配置高级 Clair 设置。

3.4.1. 非受管 Clair 配置

Red Hat Quay 用户可以使用 Red Hat Quay OpenShift Container Platform Operator 运行非受管 Clair 配置。此功能允许用户创建非受管 Clair 数据库，或在没有非受管数据库的情况下运行自定义 Clair 配置。

非受管 Clair 数据库允许 Red Hat Quay Operator 在地理复制环境中工作，其中 Operator 的多个实例必须与同一数据库通信。当用户需要在集群外存在的高可用性(HA) Clair 数据库时，也可以使用非受管 Clair 数据库。

3.4.1.1. 使用非受管 Clair 数据库运行自定义 Clair 配置

使用以下步骤将 Clair 数据库设置为 unmanaged。

流程

- 在 Quay Operator 中，将 **QuayRegistry** 自定义资源的 **clairpostgres** 组件设置为 **managed: false** ：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: quay370
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: clairpostgres
      managed: false
```

3.4.1.2. 使用非受管 Clair 数据库配置自定义 Clair 数据库

OpenShift Container Platform 上的 Red Hat Quay 允许用户提供自己的 Clair 数据库。

使用以下步骤创建自定义 Clair 数据库。

注意

以下流程使用 SSL/TLS 认证设置 Clair。要查看没有使用 SSL/TSL 认证设置 Clair 的类似流程，请参阅"使用受管 Clair 配置配置自定义 Clair 数据库"。

流程

1. 输入以下命令，创建包含 **clair-config.yaml** 的 Quay 配置捆绑包 secret：

```
$ oc create secret generic --from-file config.yaml=./config.yaml --from-file extra_ca_cert_rds-ca-2019-root.pem=./rds-ca-2019-root.pem --from-file clair-config.yaml=./clair-config.yaml --from-file ssl.cert=./ssl.cert --from-file ssl.key=./ssl.key config-bundle-secret
```

Clair config.yaml 文件示例

```
indexer:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslrootcert=/run/certs/rds-ca-2019-root.pem sslmode=verify-ca
    layer_scan_concurrency: 6
    migrations: true
    scanlock_retry: 11
log_level: debug
matcher:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslrootcert=/run/certs/rds-ca-2019-root.pem sslmode=verify-ca
    migrations: true
metrics:
    name: prometheus
notifier:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslrootcert=/run/certs/rds-ca-2019-root.pem sslmode=verify-ca
    migrations: true
```

注意

- 数据库证书挂载到 **clair-config.yaml** 中的 Clair 应用程序 pod 的 **/run/certs/rds-ca-2019-root.pem** 下。在配置 **clair-config.yaml** 时必须指定它。
- Clair on OpenShift config 中包括了一个 **clair-config.yaml** 示例。

2. 将 **clair-config.yaml** 文件添加到捆绑包 secret 中，例如：

```
apiVersion: v1
kind: Secret
metadata:
  name: config-bundle-secret
  namespace: quay-enterprise
data:
  config.yaml: <base64 encoded Quay config>
  clair-config.yaml: <base64 encoded Clair config>
  extra_ca_cert_<name>: <base64 encoded ca cert>
  ssl.crt: <base64 encoded SSL certificate>
  ssl.key: <base64 encoded SSL private key>
```

注意

更新时，提供的 **clair-config.yaml** 文件将挂载到 Clair pod。任何未提供的字段都使用 Clair 配置模块自动填充默认值。

3. 您可以通过点 **Build History** 页面中的提交或运行 **oc get pods -n <namespace>** > 来检查 Clair pod 的状态。例如：

```
$ oc get pods -n <namespace>
```

输出示例

```
NAME                                                  READY   STATUS    RESTARTS   AGE
f192fe4a-c802-4275-bcce-d2031e635126-9l2b5-25lg2   1/1     Running   0          7s
```

3.4.2. 使用受管 Clair 数据库运行自定义 Clair 配置

在某些情况下，用户可能希望使用受管 Clair 数据库运行自定义 Clair 配置。这在以下情况中很有用：

- 当用户希望禁用特定的更新器资源时。
- 当用户在断开连接的环境中运行 Red Hat Quay 时。有关在断开连接的环境中运行 Clair 的更多信息，请参阅 断开连接的环境中的 Clair。

注意

- 如果您在断开连接的环境中运行 Red Hat Quay，则 **clair-config.yaml** 的 **airgap** 参数必须设置为 **true**。
- 如果您在断开连接的环境中运行 Red Hat Quay，您应该禁用所有更新组件。

3.4.2.1. 将 Clair 数据库设置为受管

使用以下步骤将 Clair 数据库设置为 managed。

流程

- 在 Quay Operator 中，将 **QuayRegistry** 自定义资源的 **clairpostgres** 组件设置为 **managed: true** ：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: quay370
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: clairpostgres
      managed: true
```

3.4.2.2. 使用受管 Clair 配置配置自定义 Clair 数据库

OpenShift Container Platform 上的 Red Hat Quay 允许用户提供自己的 Clair 数据库。

使用以下步骤创建自定义 Clair 数据库。

流程

1. 输入以下命令，创建包含 **clair-config.yaml** 的 Quay 配置捆绑包 secret：

```
$ oc create secret generic --from-file config.yaml=./config.yaml --from-file extra_ca_cert_rds-ca-2019-root.pem=./rds-ca-2019-root.pem --from-file clair-config.yaml=./clair-config.yaml config-bundle-secret
```

Clair config.yaml 文件示例

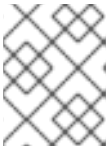
```
indexer:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslmode=disable
    layer_scan_concurrency: 6
    migrations: true
    scanlock_retry: 11
log_level: debug
matcher:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslmode=disable
    migrations: true
metrics:
    name: prometheus
notifier:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslmode=disable
    migrations: true
```

注意

- 数据库证书挂载到 **clair-config.yaml** 中的 Clair 应用程序 pod 的 **/run/certs/rds-ca-2019-root.pem** 下。在配置 **clair-config.yaml** 时必须指定它。
- Clair on OpenShift config 中包括了一个 **clair-config.yaml** 示例。

2. 将 **clair-config.yaml** 文件添加到捆绑包 secret 中，例如：

```
apiVersion: v1
kind: Secret
metadata:
  name: config-bundle-secret
  namespace: quay-enterprise
data:
  config.yaml: <base64 encoded Quay config>
  clair-config.yaml: <base64 encoded Clair config>
```



注意

- 更新时，提供的 **clair-config.yaml** 文件将挂载到 Clair pod。任何未提供的字段都使用 Clair 配置模块自动填充默认值。

3. 您可以通过点 **Build History** 页面中的提交或运行 **oc get pods -n <namespace>** > 来检查 Clair pod 的状态。例如：

```
$ oc get pods -n <namespace>
```

输出示例

```
NAME                                        READY   STATUS    RESTARTS   AGE
f192fe4a-c802-4275-bcce-d2031e635126-9l2b5-25lg2   1/1     Running   0          7s
```

3.4.3. 在断开连接的环境中的 Clair

注意

目前，IBM Power 和 IBM Z 不支持在断开连接的环境中部署 Clair。

Clair 使用一组称为 *updaters* 的组件来处理从各种漏洞数据库获取和解析数据。默认情况下，设置了 updaters，以直接从互联网拉取漏洞数据，并可立即使用。但是，有些用户可能需要 Red Hat Quay 在断开连接的环境中运行，或者不需要直接访问互联网的环境。Clair 支持断开连接的环境，方法是使用不同类型的更新工作流来考虑网络隔离。这通过使用 **clairctl** 命令行界面工具工作，它通过使用开放主机从互联网获取更新器数据，安全地将数据传送到隔离主机，然后对隔离主机上的更新器数据非常重要。

使用本指南在断开连接的环境中部署 Clair。

重要

由于已知问题 PROJQUAY-6577，Red Hat Quay Operator 无法正确呈现自定义的 Clair **config.yaml** 文件。因此，以下步骤目前无法正常工作。

用户必须从头开始创建整个 Clair 配置，而不依赖于 Operator 来填充字段。要做到这一点，请按照 流程在断开连接的环境中对镜像进行 Clair 扫描 的说明。

注意

目前，Clair 增强数据是 CVSS 数据。目前，在断开连接的环境中不支持增强数据。

有关 Clair 更新器的更多信息，请参阅"Clair updaters"。

3.4.3.1. 在断开连接的 OpenShift Container Platform 集群中设置 Clair

使用以下步骤在断开连接的 OpenShift Container Platform 集群中设置置备的 Clair pod。

重要

由于已知问题 PROJQUAY-6577，Red Hat Quay Operator 无法正确呈现自定义的 Clair **config.yaml** 文件。因此，以下步骤目前无法正常工作。

用户必须从头开始创建整个 Clair 配置，而不依赖于 Operator 来填充字段。要做到这一点，请按照 流程在断开连接的环境中对镜像进行 Clair 扫描 的说明。

3.4.3.1.1. 为 OpenShift Container Platform 部署安装 clairctl 命令行工具

使用以下步骤为 OpenShift Container Platform 部署安装 **clairctl** CLI 工具。

流程

1. 输入以下命令，在 OpenShift Container Platform 集群中为 Clair 部署安装 **clairctl** 程序：

```
$ oc -n quay-enterprise exec example-registry-clair-app-64dd48f866-6ptgw -- cat
/usr/bin/clairctl > clairctl
```

注意

不正式使用，可以下载 **clairctl** 工具

2. 设置 **clairctl** 文件的权限，以便用户可以执行并运行它，例如：

```
$ chmod u+x ./clairctl
```

3.4.3.1.2. 为 OpenShift Container Platform 上的 Clair 部署检索并解码 Clair 配置 secret

使用以下步骤为 OpenShift Container Platform 上置备的 OpenShift Container Platform 置备 Clair 实例检索并解码配置 secret。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。

流程

1. 输入以下命令来检索和解码配置 secret，然后将其保存到 Clair 配置 YAML 中：

```
$ oc get secret -n quay-enterprise example-registry-clair-config-secret  -o "jsonpath=
{$.data['config\.yaml']}" | base64 -d > clair-config.yaml
```

2. 更新 **clair-config.yaml** 文件，使 **disable_updaters** 和 **airgap** 参数设置为 **true**，例如：

```
---
indexer:
  airgap: true
---
matcher:
  disable_updaters: true
---
```

3.4.3.1.3. 从连接的 Clair 实例导出更新程序捆绑包

使用以下步骤从可访问互联网的 Clair 实例导出更新器捆绑包。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已检索并解码 Clair 配置 secret，并将其保存到 Clair **config.yaml** 文件中。
- 在 Clair **config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。

流程

- 在可以访问互联网的 Clair 实例中，将 **clairctl** CLI 工具与配置文件一起使用，以导出更新器捆绑包。例如：

```
$ ./clairctl --config ./config.yaml export-updaters updates.gz
```

3.4.3.1.4. 在断开连接的 OpenShift Container Platform 集群中配置对 Clair 数据库的访问

使用以下步骤在断开连接的 OpenShift Container Platform 集群中配置对 Clair 数据库的访问。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已检索并解码 Clair 配置 secret，并将其保存到 Clair **config.yaml** 文件中。
- 在 Clair **config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 Clair 实例导出了更新器捆绑包。

流程

1. 使用 **oc** CLI 工具确定 Clair 数据库服务，例如：

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

输出示例

```
NAME                              TYPE          CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
example-registry-clair-app        ClusterIP     172.30.224.93    <none>
80/TCP,8089/TCP                   4d21h
example-registry-clair-postgres   ClusterIP     172.30.246.88    <none>        5432/TCP
4d21h
...
```

2. 转发 Clair 数据库端口，使其可从本地机器访问。例如：

```
$ oc port-forward -n quay-enterprise service/example-registry-clair-postgres 5432:5432
```

3. 更新 Clair **config.yaml** 文件，例如：

```
indexer:
  connstring: host=localhost port=5432 dbname=postgres user=postgres
password=postgres sslmode=disable (1)
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
  scanner:
    repo:
      rhel-repository-scanner: (2)
        repo2cpe_mapping_file: /data/cpe-map.json
    package:
      rhel_containerscanner: (3)
        name2repos_mapping_file: /data/repo-map.json
```

1. 在多 **connstring** 字段中使用 **localhost** 替换 **host** 的值。
2. 有关 **rhel-repository-scanner** 参数的更多信息，请参阅 "Mapping repository to Common Product Enumeration information"。
3. 有关 **rhel_containerscanner** 参数的更多信息，请参阅 "Mapping repository to Common Product Enumeration information"。

3.4.3.1.5. 将更新器捆绑包导入到断开连接的 OpenShift Container Platform 集群中

使用以下步骤将 updaters 捆绑包导入到断开连接的 OpenShift Container Platform 集群中。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已检索并解码 Clair 配置 secret，并将其保存到 Clair **config.yaml** 文件中。
- 在 Clair **config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 Clair 实例导出了更新器捆绑包。
- 您已将更新程序捆绑包传输到断开连接的环境中。

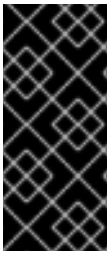
流程

- 使用 **clairctl** CLI 工具将更新器捆绑包导入到 OpenShift Container Platform 部署的 Clair 数据库中。例如：

```
$ ./clairctl --config ./clair-config.yaml import-updaters updates.gz
```

3.4.3.2. 为断开连接的 OpenShift Container Platform 集群设置 Clair 的自我管理部署

使用以下步骤为断开连接的 OpenShift Container Platform 集群设置 Clair 的自我管理部署。



重要

由于已知问题 PROJQUAY-6577，Red Hat Quay Operator 无法正确呈现自定义的 Clair **config.yaml** 文件。因此，以下步骤目前无法正常工作。

用户必须从头开始创建整个 Clair 配置，而不依赖于 Operator 来填充字段。要做到这一点，请按照 流程在断开连接的环境中对镜像进行 Clair 扫描 的说明。

3.4.3.2.1. 在 OpenShift Container Platform 上安装用于自我管理的 Clair 部署的 clairctl 命令行工具

使用以下步骤在 OpenShift Container Platform 上安装用于自我管理的 Clair 部署的 **clairctl** CLI 工具。

流程

1. 使用 **podman cp** 命令为自我管理的 Clair 部署安装 **clairctl** 程序，例如：

```
$ sudo podman cp clairv4:/usr/bin/clairctl ./clairctl
```

2. 设置 **clairctl** 文件的权限，以便用户可以执行并运行它，例如：

```
$ chmod u+x ./clairctl
```

3.4.3.2.2. 为断开连接的 OpenShift Container Platform 集群部署自我管理的 Clair 容器

使用以下步骤为断开连接的 OpenShift Container Platform 集群部署自我管理的 Clair 容器。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。

流程

1. 为您的 Clair 配置文件创建一个文件夹，例如：

```
$ mkdir /etc/clairv4/config/
```

2. 创建 Clair 配置文件，并将 **disable_updaters** 参数设置为 **true**，例如：

```
---
indexer:
  airgap: true
---
matcher:
  disable_updaters: true
---
```

3. 使用容器镜像启动 Clair，从您创建的文件中挂载到配置中：

```
$ sudo podman run -it --rm --name clairv4 \
-p 8081:8081 -p 8088:8088 \
-e CLAIR_CONF=/clair/config.yaml \
```

```
-e CLAIR_MODE=combo \
-v /etc/clairv4/config:/clair:Z \
registry.redhat.io/quay/clair-rhel8:v3.11.1
```

3.4.3.2.3. 从连接的 Clair 实例导出更新程序捆绑包

使用以下步骤从可访问互联网的 Clair 实例导出更新器捆绑包。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已部署了 Clair。
- 在 Clair **config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。

流程

- 在可以访问互联网的 Clair 实例中，将 **clairctl** CLI 工具与配置文件一起使用，以导出更新器捆绑包。例如：

```
$ ./clairctl --config ./config.yaml export-updaters updates.gz
```

3.4.3.2.4. 在断开连接的 OpenShift Container Platform 集群中配置对 Clair 数据库的访问

使用以下步骤在断开连接的 OpenShift Container Platform 集群中配置对 Clair 数据库的访问。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已部署了 Clair。
- 在 Clair **config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 Clair 实例导出了更新器捆绑包。

流程

1. 使用 **oc** CLI 工具确定 Clair 数据库服务，例如：

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

输出示例

```
NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
example-registry-clair-app          ClusterIP      172.30.224.93    <none>
80/TCP,8089/TCP              4d21h
example-registry-clair-postgres     ClusterIP      172.30.246.88    <none>        5432/TCP
4d21h
...
```

2. 转发 Clair 数据库端口，使其可从本地机器访问。例如：

```
$ oc port-forward -n quay-enterprise service/example-registry-clair-postgres 5432:5432
```

3. 更新 Clair **config.yaml** 文件，例如：

```
indexer:
  connstring: host=localhost port=5432 dbname=postgres user=postgres
password=postgres sslmode=disable (1)
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
  scanner:
    repo:
      rhel-repository-scanner: (2)
        repo2cpe_mapping_file: /data/cpe-map.json
    package:
      rhel_containerscanner: (3)
        name2repos_mapping_file: /data/repo-map.json
```

(1) 在多 **connstring** 字段中使用 **localhost** 替换 **host** 的值。

(2) 有关 **rhel-repository-scanner** 参数的更多信息，请参阅 "Mapping repository to Common Product Enumeration information"。

(3) 有关 **rhel_containerscanner** 参数的更多信息，请参阅 "Mapping repository to Common Product Enumeration information"。

3.4.3.2.5. 将更新器捆绑包导入到断开连接的 OpenShift Container Platform 集群中

使用以下步骤将 updaters 捆绑包导入到断开连接的 OpenShift Container Platform 集群中。

先决条件

- 您已安装了 **clairctl** 命令行工具工具。
- 您已部署了 Clair。
- 在 Clair **config.yaml** 文件中，**disable_updaters** 和 **airgap** 参数被设置为 **true**。
- 您已从可访问互联网的 Clair 实例导出了更新器捆绑包。
- 您已将更新程序捆绑包传输到断开连接的环境中。

流程

- 使用 **clairctl** CLI 工具将更新器捆绑包导入到 OpenShift Container Platform 部署的 Clair 数据库中：

```
$ ./clairctl --config ./clair-config.yaml import-updaters updates.gz
```

3.4.4. 将软件仓库映射到通用产品枚举信息

注意

目前，IBM Power 和 IBM Z 不支持将软件仓库映射到通用产品枚举信息。

Clair 的 Red Hat Enterprise Linux (RHEL)扫描程序依赖于通用产品枚举(CPE)文件将 RPM 软件包映射到对应的安全数据，以生成匹配的结果。这些文件归产品安全性及每日更新。

必须存在 cp 文件，或者必须允许访问该文件，以便扫描程序正确处理 RPM 软件包。如果文件不存在，则不会扫描容器镜像中安装的 RPM 软件包。

表 3.1. Clair cp 映射文件

CPE	JSON 映射文件的链接
repos2cpe	Red Hat Repository-to-CPE JSON
names2repos	Red Hat Name-to-Repos JSON.

除了将 CVE 信息上传到断开连接的 Clair 安装的数据库外，还必须在本地提供映射文件：

- 对于独立的 Red Hat Quay 和 Clair 部署，必须将映射文件加载到 Clair pod。
- 对于 OpenShift Container Platform 部署的 Red Hat Quay，您必须将 Clair 组件设置为 **unmanaged**。然后，必须手动部署 Clair，将配置设置为加载映射文件的本地副本。

3.4.4.1. 将软件仓库映射到通用产品枚举示例配置

使用 Clair 配置中的 **repo2cpe_mapping_file** 和 **name2repos_mapping_file** 字段，使其包含 CPE JSON 映射文件。例如：

```
indexer:
 scanner:
   repo:
     rhel-repository-scanner:
       repo2cpe_mapping_file: /data/cpe-map.json
   package:
     rhel_containerscanner:
       name2repos_mapping_file: /data/repo-map.json
```

如需更多信息，请参阅如何准确匹配 OVAL 安全数据以安装 RPM。

第 4 章 在基础架构节点上部署 RED HAT QUAY

默认情况下，在使用 Red Hat **Quay** Operator 部署 registry 时，Quay 相关的 pod 会放置在任意 worker 节点上。有关如何使用机器集将节点配置为仅托管基础架构组件的更多信息，请参阅 创建基础架构机器集。

如果您不使用 OpenShift Container Platform 机器集资源来部署 infra 节点，本文档中的部分演示了如何手动标记和污点节点用于基础架构目的。在手动配置基础架构节点或使用机器集后，您可以使用节点选择器和容限来控制在这些节点上放置 **Quay** pod。

4.1. 为基础架构使用标记和污点节点

使用以下步骤为基础架构使用标记和包含节点。

1. 输入以下命令显示 master 和 worker 节点。在本例中，有三个 master 节点和 6 个 worker 节点。

```
$ oc get nodes
```

输出示例

```
NAME                                          STATUS   ROLES    AGE     VERSION
user1-jcnp6-master-0.c.quay-devel.internal        Ready    master   3h30m   v1.20.0+ba45583
user1-jcnp6-master-1.c.quay-devel.internal        Ready    master   3h30m   v1.20.0+ba45583
user1-jcnp6-master-2.c.quay-devel.internal        Ready    master   3h30m   v1.20.0+ba45583
user1-jcnp6-worker-b-65plj.c.quay-devel.internal   Ready    worker   3h21m
v1.20.0+ba45583
user1-jcnp6-worker-b-jr7hc.c.quay-devel.internal   Ready    worker   3h21m
v1.20.0+ba45583
user1-jcnp6-worker-c-jrq4v.c.quay-devel.internal   Ready    worker   3h21m
v1.20.0+ba45583
user1-jcnp6-worker-c-pwxfp.c.quay-devel.internal   Ready    worker   3h21m
v1.20.0+ba45583
user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal   Ready    worker   3h22m
v1.20.0+ba45583
user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal   Ready    worker   3h21m
v1.20.0+ba45583
```

2. 输入以下命令为基础架构使用标记三个 worker 节点：

```
$ oc label node --overwrite user1-jcnp6-worker-c-pwxfp.c.quay-devel.internal node-
role.kubernetes.io/infra=
```

```
$ oc label node --overwrite user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal node-
role.kubernetes.io/infra=
```

```
$ oc label node --overwrite user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal node-
role.kubernetes.io/infra=
```

3. 现在，当列出集群中的节点时，最后三个 worker 节点具有 **infra** 角色。例如：

```
$ oc get nodes
```

示例

```
NAME                                          STATUS   ROLES        AGE     VERSION
user1-jcnp6-master-0.c.quay-devel.internal    Ready    master       4h14m
v1.20.0+ba45583
user1-jcnp6-master-1.c.quay-devel.internal    Ready    master       4h15m
v1.20.0+ba45583
user1-jcnp6-master-2.c.quay-devel.internal    Ready    master       4h14m
v1.20.0+ba45583
user1-jcnp6-worker-b-65plj.c.quay-devel.internal   Ready    worker       4h6m
 v1.20.0+ba45583
user1-jcnp6-worker-b-jr7hc.c.quay-devel.internal   Ready    worker       4h5m
 v1.20.0+ba45583
user1-jcnp6-worker-c-jrq4v.c.quay-devel.internal   Ready    worker       4h5m
 v1.20.0+ba45583
user1-jcnp6-worker-c-pwxfp.c.quay-devel.internal   Ready    infra,worker   4h6m
 v1.20.0+ba45583
user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal   Ready    infra,worker   4h6m
 v1.20.0+ba45583
user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal   Ready    infra,worker   4h6m
 v1.20.0+ba4558
```

4. 当为 worker 节点分配 **infra** 角色时，用户工作负载可能会意外地分配给 infra 节点。要避免这种情况，您可以将污点应用到 infra 节点，然后为您要控制的 pod 添加容限。例如：

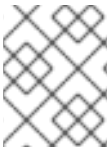
```
$ oc adm taint nodes user1-jcnp6-worker-c-pwxfp.c.quay-devel.internal node-
role.kubernetes.io/infra:NoSchedule
```

```
$ oc adm taint nodes user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal node-
role.kubernetes.io/infra:NoSchedule
```

```
$ oc adm taint nodes user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal node-
role.kubernetes.io/infra:NoSchedule
```

4.2. 使用节点选择器和容限创建项目

使用以下步骤创建带有节点选择器和容限的项目。



注意

以下流程也可以通过删除安装 Red Hat Quay Operator 和创建部署时使用的命名空间或命名空间来完成。然后，用户可以使用以下注解创建新资源：

流程

1. 输入以下命令编辑部署 Red Hat Quay 的命名空间，以及以下注解：

```
$ oc annotate namespace <namespace> openshift.io/node-selector='node-
role.kubernetes.io/infra='
```

输出示例

```
namespace/<namespace> annotated
```

2. 输入以下命令来获取可用 pod 列表：

```
$ oc get pods -o wide
```

输出示例

```
NAME                                               READY   STATUS      RESTARTS        AGE     IP
NODE                                    NOMINATED NODE   READINESS GATES
example-registry-clair-app-5744dd64c9-9d5jt        1/1     Running     0               173m
10.130.4.13   stevsmit-quay-ocp-tes-5gwws-worker-c-6xkn7   <none>          <none>
example-registry-clair-app-5744dd64c9-fg86n        1/1     Running     6 (3h21m ago)   3h24m
10.131.0.91   stevsmit-quay-ocp-tes-5gwws-worker-c-dnhdp   <none>          <none>
example-registry-clair-postgres-845b47cd88-vdchz   1/1     Running     0               3h21m
10.130.4.10   stevsmit-quay-ocp-tes-5gwws-worker-c-6xkn7   <none>          <none>
example-registry-quay-app-64cbc5bcf-8zvgc          1/1     Running     1 (3h24m ago)   3h24m
10.130.2.12   stevsmit-quay-ocp-tes-5gwws-worker-a-tk8dx   <none>          <none>
example-registry-quay-app-64cbc5bcf-pvlz6          1/1     Running     0               3h24m
10.129.4.10   stevsmit-quay-ocp-tes-5gwws-worker-b-fjhz4   <none>          <none>
example-registry-quay-app-upgrade-8gspn            0/1     Completed   0               3h24m
10.130.2.10   stevsmit-quay-ocp-tes-5gwws-worker-a-tk8dx   <none>          <none>
example-registry-quay-database-784d78b6f8-2vkml    1/1     Running     0               3h24m
10.131.4.10   stevsmit-quay-ocp-tes-5gwws-worker-c-2frtg   <none>          <none>
example-registry-quay-mirror-d5874d8dc-fmknp       1/1     Running     0               3h24m
10.129.4.9    stevsmit-quay-ocp-tes-5gwws-worker-b-fjhz4   <none>          <none>
example-registry-quay-mirror-d5874d8dc-t4mff       1/1     Running     0               3h24m
10.129.2.19   stevsmit-quay-ocp-tes-5gwws-worker-a-k7w86   <none>          <none>
example-registry-quay-redis-79848898cb-6qf5x       1/1     Running     0               3h24m
10.130.2.11   stevsmit-quay-ocp-tes-5gwws-worker-a-tk8dx   <none>          <none>
```

3. 输入以下命令删除可用的 pod：

```
$ oc delete pods --selector quay-operator/quayregistry=example-registry -n quay-enterprise
```

输出示例

```
pod "example-registry-clair-app-5744dd64c9-9d5jt" deleted
pod "example-registry-clair-app-5744dd64c9-fg86n" deleted
pod "example-registry-clair-postgres-845b47cd88-vdchz" deleted
pod "example-registry-quay-app-64cbc5bcf-8zvgc" deleted
pod "example-registry-quay-app-64cbc5bcf-pvlz6" deleted
pod "example-registry-quay-app-upgrade-8gspn" deleted
pod "example-registry-quay-database-784d78b6f8-2vkml" deleted
pod "example-registry-quay-mirror-d5874d8dc-fmknp" deleted
pod "example-registry-quay-mirror-d5874d8dc-t4mff" deleted
pod "example-registry-quay-redis-79848898cb-6qf5x" deleted
```

在 pod 被删除后，它们会自动进行循环，并应调度到专用基础架构节点上。

4.3. 在特定命名空间中安装 RED HAT QUAY

使用以下步骤在特定命名空间中的 OpenShift Container Platform 上安装 Red Hat Quay。

- 要在特定命名空间中安装 Red Hat Quay Operator，您必须明确指定适当的项目命名空间，如以下命令所示。
  在以下示例中，使用了 **quay-registry** 命名空间。这会导致 **quay-operator** pod 登录三个基础架构节点之一。例如：

```
$ oc get pods -n quay-registry -o wide
```

输出示例

```
NAME                                    READY   STATUS    RESTARTS   AGE   IP            NODE

quay-operator.v3.4.1-6f6597d8d8-bd4dp   1/1     Running   0          30s   10.131.0.16   user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal
```

4.4. 创建 RED HAT QUAY REGISTRY

使用以下步骤创建 Red Hat Quay registry。

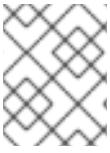
- 运行以下命令来创建 Red Hat Quay registry。然后，等待部署标记为 **就绪**。在以下示例中，您应看到它们仅调度到为基础架构目的标记的三个节点上。

```
$ oc get pods -n quay-registry -o wide
```

输出示例

```
NAME                                               READY   STATUS      RESTARTS   AGE     IP            NODE
example-registry-clair-app-789d6d984d-gpbwd        1/1     Running     1          5m57s   10.130.2.80   user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal
example-registry-clair-postgres-7c8697f5-zkzht     1/1     Running     0          4m53s   10.129.2.19   user1-jcnp6-worker-c-pwxfp.c.quay-devel.internal
example-registry-quay-app-56dd755b6d-glbf7         1/1     Running     1          5m57s   10.129.2.17   user1-jcnp6-worker-c-pwxfp.c.quay-devel.internal
example-registry-quay-database-8dc7cfd69-dr2cc     1/1     Running     0          5m43s   10.129.2.18   user1-jcnp6-worker-c-pwxfp.c.quay-devel.internal
example-registry-quay-mirror-78df886bcc-v75p9      1/1     Running     0          5m16s   10.131.0.24   user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal
example-registry-quay-postgres-init-8s8g9          0/1     Completed   0          5m54s   10.130.2.79   user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal
example-registry-quay-redis-5688ddcdb6-ndp4t       1/1     Running     0          5m56s   10.130.2.78   user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal
quay-operator.v3.4.1-6f6597d8d8-bd4dp              1/1     Running     0          22m     10.131.0.16   user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal
```

4.5. 在单一命名空间中安装 RED HAT QUAY OPERATOR 时启用监控



注意

目前，在 IBM Power 和 IBM Z 不支持在一个命名空间中安装 Red Hat Quay Operator 时启用监控。

当在单一命名空间中安装 Red Hat Quay Operator 时，监控组件被设置为 **非受管** (unmanaged)。要配置监控，您必须为 OpenShift Container Platform 中的用户定义的命名空间启用它。

如需更多信息，请参阅 配置监控堆栈 和 为用户定义的项目启用监控 的 OpenShift Container Platform 文档。

以下小节演示了如何根据 OpenShift Container Platform 文档为 Red Hat Quay 启用监控。

4.5.1. 创建集群监控配置映射

使用以下步骤检查 **cluster-monitoring-config ConfigMap** 对象是否存在。

流程

1. 输入以下命令检查 **cluster-monitoring-config** ConfigMap 对象是否存在：

```
$ oc -n openshift-monitoring get configmap cluster-monitoring-config
```

输出示例

```
Error from server (NotFound): configmaps "cluster-monitoring-config" not found
```

2. 可选： 如果 **ConfigMap** 对象不存在，请创建一个 YAML 清单。在以下示例中，该文件名为 **cluster-monitoring-config.yaml**。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
```

3. 可选：如果 **ConfigMap** 对象不存在，请创建 **ConfigMap** 对象：

```
$ oc apply -f cluster-monitoring-config.yaml
```

输出示例

```
configmap/cluster-monitoring-config created
```

4. 运行以下命令，确保 **ConfigMap** 对象存在：

```
$ oc -n openshift-monitoring get configmap cluster-monitoring-config
```

输出示例

```
NAME                     DATA   AGE
cluster-monitoring-config   1      12s
```

4.5.2. 创建用户定义的工作负载监控 ConfigMap 对象

使用以下步骤检查 **user-workload-monitoring-config ConfigMap** 对象是否存在。

流程

1. 输入以下命令检查 **user-workload-monitoring-config ConfigMap** 对象是否存在：

```
$ oc -n openshift-user-workload-monitoring get configmap user-workload-monitoring-config
```

输出示例

```
Error from server (NotFound): configmaps "user-workload-monitoring-config" not found
```

2. 如果 **ConfigMap** 对象不存在，请创建一个 YAML 清单。在以下示例中，该文件名为 **user-workload-monitoring-config.yaml**。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: user-workload-monitoring-config
  namespace: openshift-user-workload-monitoring
data:
  config.yaml: |
```

3. 可选：输入以下命令创建 **ConfigMap** 对象：

```
$ oc apply -f user-workload-monitoring-config.yaml
```

输出示例

```
configmap/user-workload-monitoring-config created
```

4.5.3. 为用户定义的项目启用监控

使用以下步骤为用户定义的项目启用监控。

流程

1. 输入以下命令检查用户定义的项目的监控是否正在运行：

```
$ oc get pods -n openshift-user-workload-monitoring
```

输出示例

```
No resources found in openshift-user-workload-monitoring namespace.
```

2. 输入以下命令编辑 **cluster-monitoring-config ConfigMap** ：

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

3. 在 **config.yaml** 文件中设置 **enableUserWorkload: true**，为集群上的用户定义的项目启用监控：

```
apiVersion: v1
data:
  config.yaml: |
    enableUserWorkload: true
kind: ConfigMap
metadata:
  annotations:
```

4. 输入以下命令保存文件，应用更改并确保适当的 pod 正在运行：

```
$ oc get pods -n openshift-user-workload-monitoring
```

输出示例

```
NAME                                  READY   STATUS    RESTARTS   AGE
prometheus-operator-6f96b4b8f8-gq6rl  2/2     Running   0          15s
prometheus-user-workload-0            5/5     Running   1          12s
prometheus-user-workload-1            5/5     Running   1          12s
thanos-ruler-user-workload-0          3/3     Running   0          8s
thanos-ruler-user-workload-1          3/3     Running   0          8s
```

4.5.4. 创建 Service 对象以公开 Red Hat Quay 指标

使用以下步骤创建 **Service** 对象来公开 Red Hat Quay 指标。

流程

1. 为 Service 对象创建 YAML 文件：

```
$ cat <<EOF >  quay-service.yaml

apiVersion: v1
kind: Service
metadata:
  annotations:
  labels:
    quay-component: monitoring
    quay-operator/quayregistry: example-registry
  name: example-registry-quay-metrics
  namespace: quay-enterprise
spec:
  ports:
  - name: quay-metrics
    port: 9091
    protocol: TCP
    targetPort: 9091
  selector:
    quay-component: quay-app
    quay-operator/quayregistry: example-registry
  type: ClusterIP
EOF
```

2. 运行以下命令来创建 **Service** 对象：

```
-
```

```
$  oc apply -f quay-service.yaml
```

输出示例

```
service/example-registry-quay-metrics created
```

4.5.5. 创建 ServiceMonitor 对象

使用以下步骤将 OpenShift Monitoring 配置为通过创建 **ServiceMonitor** 资源来提取指标。

流程

1. 为 **ServiceMonitor** 资源创建 YAML 文件：

```
$ cat <<EOF >  quay-service-monitor.yaml

apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    quay-operator/quayregistry: example-registry
  name: example-registry-quay-metrics-monitor
  namespace: quay-enterprise
spec:
  endpoints:
  - port: quay-metrics
  namespaceSelector:
    any: true
  selector:
    matchLabels:
      quay-component: monitoring
EOF
```

2. 运行以下命令来创建 **ServiceMonitor** 资源：

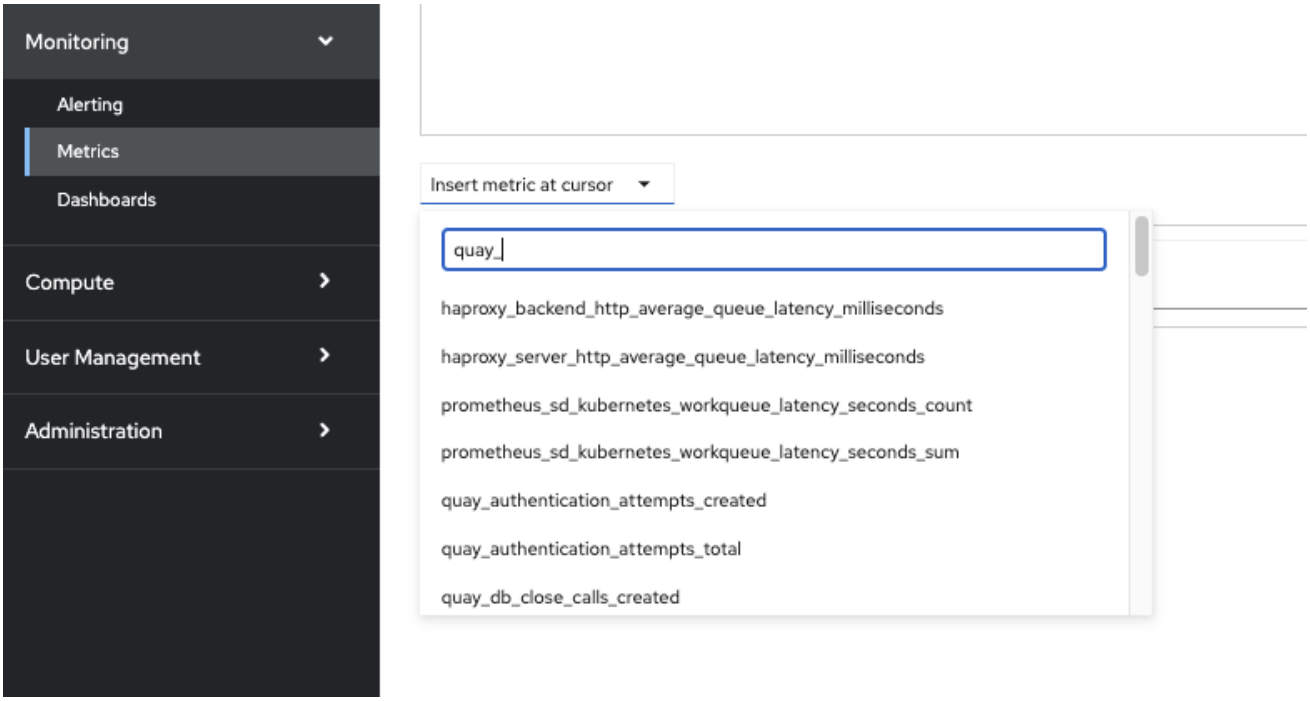
```
$ oc apply -f quay-service-monitor.yaml
```

输出示例

```
servicemonitor.monitoring.coreos.com/example-registry-quay-metrics-monitor created
```

4.5.6. 在 OpenShift Container Platform 中查看指标

您可以在 OpenShift Container Platform 控制台中访问 **Monitoring → Metrics** 下的指标。在 Expression 字段中，输入 **quay_** 以查看可用指标列表：



例如，如果您将用户添加到 registry 中，请选择 **quay-users_rows** 指标：

4.6. 调整受管存储的大小

在 OpenShift Container Platform 上部署 Red Hat Quay 时，会部署三个不同的持久性卷声明(PVC)：

- 一个用于 PostgreSQL 13 registry。
- 一个用于 Clair PostgreSQL 13 registry。
- 一个使用 NooBaa 作为后端存储。

注意

Red Hat Quay 和 NooBaa 之间的连接通过 OpenShift Container Platform 中的 S3 API 和 ObjectBucketClaim API 进行。Red Hat Quay 利用该 API 组在 NooBaa 中创建存储桶，获取访问密钥，并自动设置所有内容。在后端或 NooBaa 上，该存储桶是在后备存储中创建的。因此，NooBaa PVC 不会被挂载或连接到 Red Hat Quay pod。

PostgreSQL 13 和 Clair PostgreSQL 13 PVC 的默认大小设置为 50 GiB。您可以按照以下流程在 OpenShift Container Platform 控制台中为这些 PVC 扩展存储。

注意

以下流程与 Red Hat OpenShift Data Foundation 上的 扩展持久性卷声明 共享通用性。

4.6.1. 在 Red Hat Quay 中重新定义 PostgreSQL 13 PVC 的大小

使用以下步骤调整 PostgreSQL 13 和 Clair PostgreSQL 13 PVC 的大小。

先决条件

- 在 OpenShift Container Platform 上具有集群管理员特权。

流程

1. 登录到 OpenShift Container Platform 控制台并选择 **Storage → Persistent Volume Claims**。
2. 为 PostgreSQL 13 或 Clair PostgreSQL 13 选择所需的 **PersistentVolumeClaim**，例如 **example-registry-quay-postgres-13**。
3. 在 **Action** 菜单中，选择 **Expand PVC**。
4. 输入持久性卷声明的新大小，然后选择 **Expand**。
   几分钟后，扩展的大小应反映在 PVC 的 **Capacity** 字段中。

4.7. 自定义默认 OPERATOR 镜像

注意

目前，IBM Power 和 IBM Z 不支持自定义默认 Operator 镜像。

在某些情况下，覆盖 Red Hat Quay Operator 使用的默认镜像可能很有用。这可以通过在 Red Hat Quay Operator **ClusterServiceVersion** 中设置一个或多个环境变量来实现。

重要

在 Red Hat Quay 环境中不支持使用此机制，强烈建议仅用于开发或测试目的。在 Red Hat Quay Operator 中使用非默认镜像时，不能保证部署可以正常工作。

4.7.1. 环境变量

Red Hat Quay Operator 中使用以下环境变量覆盖组件镜像：

环境变量	组件
RELATED_IMAGE_COMPONENT_QUAY	**base**
RELATED_IMAGE_COMPONENT_CLAIR	**Clair**

RELATED_IMAGE_COMPONENT_POSTGRES	**postgres** 和 **clair** 数据库
RELATED_IMAGE_COMPONENT_REDIS	**redis**

注意

覆盖的 **镜像必**须 由清单(@sha256:)引用，而不能通过标签(:latest)引用。

4.7.2. 将覆盖应用到正在运行的 Operator

当通过 Operator Lifecycle Manager (OLM) 安装 Red Hat Quay Operator 时，可以通过修改 **ClusterServiceVersion** 对象来轻松覆盖受管组件容器镜像。

使用以下步骤将覆盖应用到正在运行的 Red Hat Quay Operator。

流程

1. **ClusterServiceVersion** 对象是集群中正在运行的 Operator 的 Operator 的表示。使用 Kubernetes UI 或 **kubectl/oc** CLI 工具查找 Red Hat Quay Operator 的 **ClusterServiceVersion**。例如：

```
$ oc get clusterserviceversions -n <your-namespace>
```

2. 使用 UI、**oc edit** 或其他方法修改 Red Hat Quay **ClusterServiceVersion**，使其包含上面概述的环境变量以指向覆盖镜像：
JSONPath: **spec.install.spec.deployments[0].spec.template.spec.containers[0].env**

```
- name: RELATED_IMAGE_COMPONENT_QUAY
  value:
quay.io/projectquay/quay@sha256:c35f5af964431673f4ff5c9e90bdf45f19e38b8742b5903d41c
10cc7f6339a6d
- name: RELATED_IMAGE_COMPONENT_CLAIR
  value:
quay.io/projectquay/clair@sha256:70c99feceb4c0973540d22e740659cd8d616775d3ad1c169
8ddf71d0221f3ce6
- name: RELATED_IMAGE_COMPONENT_POSTGRES
  value: centos/postgresql-10-
centos7@sha256:de1560cb35e5ec643e7b3a772ebaac8e3a7a2a8e8271d9e91ff023539b4dfb3
3
- name: RELATED_IMAGE_COMPONENT_REDIS
  value: centos/redis-32-
centos7@sha256:06dbb609484330ec6be6090109f1fa16e936afcf975d1cbc5fff3e6c7cae7542
```

注意

这在 Operator 级别上完成，因此每个 **QuayRegistry** 都将使用相同的覆盖进行部署。

4.8. AWS S3 CLOUDFRONT

注意

目前，IBM Power 和 IBM Z 不支持使用 AWS S3 CloudFront。

如果您将 AWS S3 Cloudfront 用于后端 registry 存储，请使用以下步骤。

流程

1. 输入以下命令指定 registry 密钥：

```
$ oc create secret generic --from-file config.yaml=./config_awss3cloudfront.yaml --from-file
default-cloudfront-signing-key.pem=./default-cloudfront-signing-key.pem test-config-bundle
```

第 5 章 RED HAT QUAY 构建增强

Red Hat Quay 构建可以在虚拟平台上运行。对运行之前构建配置的后向兼容性也可用。

5.1. RED HAT QUAY 构建限制

在非特权上下文中在 Red Hat Quay 中运行构建可能会导致一些在以前的构建策略下工作的命令失败。尝试更改构建策略可能会导致构建出现性能问题和可靠性。

直接在容器中运行构建与使用虚拟机的隔离没有相同的隔离。更改构建环境也可能会导致以前工作的构建失败。

5.2. 使用 OPENSHIFT CONTAINER PLATFORM 创建 RED HAT QUAY 构建器环境

本节中的步骤解释了如何使用 OpenShift Container Platform 创建 Red Hat Quay 虚拟构建器环境。

5.2.1. OpenShift Container Platform TLS 组件

tls 组件允许您控制 TLS 配置。

注意

当 TLS 组件由 Operator 管理时，Red Hat Quay 3 不支持构建器。

如果将 **tls** 设置为 **unmanaged**，则提供自己的 **ssl.cert** 和 **ssl.key** 文件。在本实例中，如果希望集群支持构建器，您必须将 Quay 路由和构建器路由名称添加到证书中的 SAN 列表中，或使用通配符。

要添加 builder 路由，请使用以下格式：

```
[quayregistry-cr-name]-quay-builder-[ocp-namespace].[ocp-domain-name]:443
```

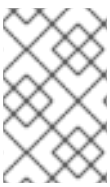
5.2.2. 使用 OpenShift Container Platform for Red Hat Quay builders

构建器需要 SSL/TLS 证书。有关 SSL/TLS 证书的更多信息，请参阅 向 Red Hat Quay 容器添加 TLS 证书。

如果使用 Amazon Web Service (AWS) S3 存储，您必须在运行构建器前修改 AWS 控制台中的存储桶。如需所需参数，请参阅以下部分"修改 AWS S3 存储桶"。

5.2.2.1. 为虚拟构建器准备 OpenShift Container Platform

使用以下步骤为 Red Hat Quay 虚拟构建器准备 OpenShift Container Platform。



注意

- 此流程假设您已置备集群并运行 Quay Operator。
- 此流程是在 OpenShift Container Platform 上设置虚拟命名空间。

流程

1. 使用集群管理员帐户登录到 Red Hat Quay 集群。
2. 运行以下命令，创建一个运行您的虚拟构建器（如 **virtual-builders** ）的新项目：

```
$ oc new-project virtual-builders
```

3. 输入以下命令在项目中创建一个 **ServiceAccount**，它将用于运行构建：

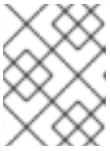
```
$ oc create sa -n virtual-builders quay-builder
```

4. 为创建的服务帐户提供编辑权限，以便它可以运行构建：

```
$ oc adm policy -n virtual-builders add-role-to-user edit system:serviceaccount:virtual-builders:quay-builder
```

5. 输入以下命令授予 Quay builder **anyuid scc** 权限：

```
$ oc adm policy -n virtual-builders add-scc-to-user anyuid -z quay-builder
```



注意

此操作需要集群管理员特权。这是必要的，因为构建器必须以 Podman 用户身份运行，才能使非特权或无根构建正常工作。

6. 获取 Quay builder 服务帐户的令牌。
   a. 如果使用 OpenShift Container Platform 4.10 或更早的版本，请输入以下命令：

```
oc sa get-token -n virtual-builders quay-builder
```

   b. 如果使用 OpenShift Container Platform 4.11 或更高版本，请输入以下命令：

```
$ oc create token quay-builder -n virtual-builders
```

注意

当令牌过期时，您需要请求新令牌。另外，您还可以添加自定义过期。例如，指定 **--duration 20160m** 以保留令牌两周。

输出示例

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IldfQUJkaDVmb3ltTHZ0dGZMYjhIWnYxZTQzN2dJVEJxcDJscldSdEUtYWsifQ...
```

7. 输入以下命令确定构建程序路由：

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

输出示例

```
NAME                    HOST/PORT                                   PATH
```

```
SERVICES                        PORT   TERMINATION     WILDCARD
...
example-registry-quay-builder        example-registry-quay-builder-quay-
enterprise.apps.docs.quayteam.org                example-registry-quay-app            grpc
edge/Redirect   None
...
```

8. 输入以下命令使用 .crt 扩展生成自签名 SSL/TLS 证书：

```
$ oc extract cm/kube-root-ca.crt -n openshift-apiserver
```

输出示例

```
ca.crt
```

9. 输入以下命令将 **ca.crt** 文件重命名为 **extra_ca_cert_build_cluster.crt** ：

```
$ mv ca.crt extra_ca_cert_build_cluster.crt
```

10. 在**控制台**中找到配置捆绑包的 secret，然后选择 **Actions → Edit Secret** 并添加适当的构建程序配置：

```
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- <superusername>
FEATURE_USER_CREATION: false
FEATURE_QUOTA_MANAGEMENT: true
FEATURE_BUILD_SUPPORT: True
BUILDMAN_HOSTNAME: <sample_build_route> 1
BUILD_MANAGER:
  - ephemeral
  - ALLOWED_WORKER_COUNT: 1
    ORCHESTRATOR_PREFIX: buildman/production/
    JOB_REGISTRATION_TIMEOUT: 3600 2
    ORCHESTRATOR:
      REDIS_HOST: <sample_redis_hostname> 3
      REDIS_PASSWORD: ""
      REDIS_SSL: false
      REDIS_SKIP_KEYSPACE_EVENT_SETUP: false
    EXECUTORS:
      - EXECUTOR: kubernetesPodman
        NAME: openshift
        BUILDER_NAMESPACE: <sample_builder_namespace> 4
        SETUP_TIME: 180
        MINIMUM_RETRY_THRESHOLD: 0
        BUILDER_CONTAINER_IMAGE: <sample_builder_container_image> 5
        # Kubernetes resource options
        K8S_API_SERVER: <sample_k8s_api_server> 6
        K8S_API_TLS_CA: <sample_crt_file> 7
        VOLUME_SIZE: 8G
        KUBERNETES_DISTRIBUTION: openshift
        CONTAINER_MEMORY_LIMITS: 300m 8
```

```
    CONTAINER_CPU_LIMITS: 1G 9
    CONTAINER_MEMORY_REQUEST: 300m 10
    CONTAINER_CPU_REQUEST: 1G 11
    NODE_SELECTOR_LABEL_KEY: ""
    NODE_SELECTOR_LABEL_VALUE: ""
    SERVICE_ACCOUNT_NAME: <sample_service_account_name>
    SERVICE_ACCOUNT_TOKEN: <sample_account_token> 12
```

1. 构建路由通过运行 **oc get route -n** 及 OpenShift Operator 的命名空间的名称来获取。路由末尾必须提供一个端口，它应使用以下格式： **[quayregistry-cr-name]-quay-builder-[ocp-namespace].[ocp-domain-name]:443**。
2. 如果设置了 **JOB_REGISTRATION_TIMEOUT** 参数，您可能会收到以下错误： **failed to register job to build manager: rpc error: code = Unauthenticated desc = Invalid build token: Signature has expired**.建议将此参数设置为至少 240。
3. 如果您的 Redis 主机有密码或 SSL/TLS 证书，您必须相应地更新。
4. 设置，以匹配虚拟构建器命名空间的名称，如 **virtual-builders**。
5. 对于早期访问，**BUILDER_CONTAINER_IMAGE** 目前是 **quay.io/projectquay/quay-builder:3.7.0-rc.2**。请注意，这可能会在早期访问窗口中更改。如果出现这种情况，则会警告客户。
6. **K8S_API_SERVER** 通过运行 **oc cluster-info** 获取。
7. 您必须手动创建并添加自定义 CA 证书，例如 **K8S_API_TLS_CA: /conf/stack/extra_ca_certs/build_cluster.crt**。
8. 如果未指定，则默认为 **5120Mi**。
9. 对于虚拟构建，您必须确保集群中有足够的资源。如果未指定，则默认为 **1000m**。
10. 如果未指定，则默认为 **3968Mi**。
11. 如果未指定，则默认为 **500m**。
12. 在运行 **oc create sa** 时获得。

配置示例

```
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- quayadmin
FEATURE_USER_CREATION: false
FEATURE_QUOTA_MANAGEMENT: true
FEATURE_BUILD_SUPPORT: True
BUILDMAN_HOSTNAME: example-registry-quay-builder-quay-
enterprise.apps.docs.quayteam.org:443
BUILD_MANAGER:
  - ephemeral
  - ALLOWED_WORKER_COUNT: 1
    ORCHESTRATOR_PREFIX: buildman/production/
    JOB_REGISTRATION_TIMEOUT: 3600
```

```
    ORCHESTRATOR:
      REDIS_HOST: example-registry-quay-redis
      REDIS_PASSWORD: ""
      REDIS_SSL: false
      REDIS_SKIP_KEYSPACE_EVENT_SETUP: false
    EXECUTORS:
      - EXECUTOR: kubernetesPodman
        NAME: openshift
        BUILDER_NAMESPACE: virtual-builders
        SETUP_TIME: 180
        MINIMUM_RETRY_THRESHOLD: 0
        BUILDER_CONTAINER_IMAGE: quay.io/projectquay/quay-builder:3.7.0-rc.2
        # Kubernetes resource options
        K8S_API_SERVER: api.docs.quayteam.org:6443
        K8S_API_TLS_CA: /conf/stack/extra_ca_certs/build_cluster.crt
        VOLUME_SIZE: 8G
        KUBERNETES_DISTRIBUTION: openshift
        CONTAINER_MEMORY_LIMITS: 1G
        CONTAINER_CPU_LIMITS: 1080m
        CONTAINER_MEMORY_REQUEST: 1G
        CONTAINER_CPU_REQUEST: 580m
        NODE_SELECTOR_LABEL_KEY: ""
        NODE_SELECTOR_LABEL_VALUE: ""
        SERVICE_ACCOUNT_NAME: quay-builder
        SERVICE_ACCOUNT_TOKEN:
"eyJhbGciOiJSUzI1NiIsImtpZCI6IldfQUJkaDVmb3ltTHZ0dGZMYjhIWnYxZTQzN2dJVEJxcDJs
cldSdEUtYWsifQ"
```

5.2.2.2. 手动添加 SSL/TLS 证书

由于配置工具的已知问题，您必须手动将自定义 SSL/TLS 证书添加到正确运行。使用以下步骤手动添加自定义 SSL/TLS 证书。

有关创建 SSL/TLS 证书的更多信息，请参阅在 Red Hat Quay 容器中添加 TLS 证书 。

5.2.2.2.1. 创建和签名证书

使用以下步骤创建并签署 SSL/TLS 证书。

流程

- 创建证书颁发机构并签署证书。如需更多信息，请参阅创建证书颁发机构并签署证书。

  openssl.cnf

```
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
```

```
[alt_names]
DNS.1 = example-registry-quay-quay-enterprise.apps.docs.quayteam.org (1)
DNS.2 = example-registry-quay-builder-quay-enterprise.apps.docs.quayteam.org (2)
```

(1) 必须包含 Red Hat Quay registry URL 的 **alt_name**。

(2) **BUILDMAN_HOSTNAME**的 **alt_name**

示例命令

```
$ openssl genrsa -out rootCA.key 2048
$ openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
$ openssl genrsa -out ssl.key 2048
$ openssl req -new -key ssl.key -out ssl.csr
$ openssl x509 -req -in ssl.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out
ssl.cert -days 356 -extensions v3_req -extfile openssl.cnf
```

5.2.2.2.2. 将 TLS 设置为非受管

使用以下步骤将 **king:tls** 设置为 unmanaged。

流程

1. 在 Red Hat Quay Registry YAML 中，将 **kind: tls** 设置为 **managed: false** ：

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

2. 在 **Events** 页面中，更改会被阻断，直到您设置了适当的 **config.yaml** 文件。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'
```

5.2.2.2.3. 创建临时 secret

使用以下步骤为 CA 证书创建临时 secret。

流程

1. 在默认命名空间中创建 CA 证书的 secret：

```
$ oc create secret generic -n quay-enterprise temp-crt --from-file
extra_ca_cert_build_cluster.crt
```

2. 在 default 命名空间中为 **ssl.key** 和 **ssl.cert** 文件创建一个 secret：

```
$ oc create secret generic -n quay-enterprise quay-config-ssl --from-file ssl.cert --from-file
ssl.key
```

5.2.2.2.4. 将 secret 数据复制到配置 YAML 中

使用以下步骤将 secret 数据复制到 **config.yaml** 文件中。

流程

1. 在控制台 UI 中找到 **Workloads → Secrets** 的新 secret。
2. 对于每个 secret，找到 YAML 视图：

```
kind: Secret
apiVersion: v1
metadata:
  name: temp-crt
  namespace: quay-enterprise
  uid: a4818adb-8e21-443a-a8db-f334ace9f6d0
  resourceVersion: '9087855'
  creationTimestamp: '2022-03-28T13:05:30Z'
...
data:
  extra_ca_cert_build_cluster.crt: >-
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURNakNDQWhxZ0F3SUJBZ0l....
type: Opaque
```

```
kind: Secret
apiVersion: v1
metadata:
  name: quay-config-ssl
  namespace: quay-enterprise
  uid: 4f5ae352-17d8-4e2d-89a2-143a3280783c
  resourceVersion: '9090567'
  creationTimestamp: '2022-03-28T13:10:34Z'
...
data:
  ssl.cert: >-
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUVaakNDQTA2Z0F3SUJBZ0lVT...
  ssl.key: >-
    LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcFFJQkFBS0NBUUVBc...
type: Opaque
```

3. 在 UI 中查找 Red Hat Quay registry 配置捆绑包的 secret，或通过运行以下命令来使用命令行：

```
$ oc get quayregistries.quay.redhat.com -o jsonpath="{.items[0].spec.configBundleSecret}
{'\n'}"  -n quay-enterprise
```

4. 在 OpenShift Container Platform 控制台中，选择配置捆绑包 secret 的 YAML 选项卡，并从您创建的两个 secret 中添加数据：

```
kind: Secret
apiVersion: v1
metadata:
```

```
  name: init-config-bundle-secret
  namespace: quay-enterprise
  uid: 4724aca5-bff0-406a-9162-ccb1972a27c1
  resourceVersion: '4383160'
  creationTimestamp: '2022-03-22T12:35:59Z'
...
data:
  config.yaml: >-
    RkVBVFVSRV9VU0VSX0lOSVRJQUxJWkU6IHRydWUKQlJ...
  extra_ca_cert_build_cluster.crt: >-

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURNakNDQWhxZ0F3SUJBZ0ldw....
  ssl.cert: >-
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUVaakNDQTA2Z0F3SUJBZ0lVT...
  ssl.key: >-
    LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcFFJQkFBS0NBUUVBc...
type: Opaque
```

5. 点击 **Save**。

6. 输入以下命令查看您的 pod 是否重启：

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

输出示例

```
NAME                                             READY   STATUS          RESTARTS   AGE
...
example-registry-quay-app-6786987b99-vgg2v           0/1     ContainerCreating   0          2s
example-registry-quay-app-7975d4889f-q7tvl           1/1     Running             0          5d21h
example-registry-quay-app-7975d4889f-zn8bb           1/1     Running             0          5d21h
example-registry-quay-app-upgrade-lswsn              0/1     Completed           0          6d1h
example-registry-quay-config-editor-77847fc4f5-nsbbv   0/1     ContainerCreating   0          2s
example-registry-quay-config-editor-c6c4d9ccd-2mwg2    1/1     Running             0
5d21h
example-registry-quay-database-66969cd859-n2ssm      1/1     Running             0          6d1h
example-registry-quay-mirror-764d7b68d9-jmlkk        1/1     Terminating         0          5d21h
example-registry-quay-mirror-764d7b68d9-jqzwg        1/1     Terminating         0          5d21h
example-registry-quay-redis-7cc5f6c977-956g8         1/1     Running             0          5d21h
```

7. 重新配置 Red Hat Quay registry 后，输入以下命令检查 Red Hat Quay 应用程序 pod 是否正在运行：

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

输出示例

```
example-registry-quay-app-6786987b99-sz6kb           1/1     Running             0          7m45s
example-registry-quay-app-6786987b99-vgg2v           1/1     Running             0          9m1s
example-registry-quay-app-upgrade-lswsn              0/1     Completed           0          6d1h
example-registry-quay-config-editor-77847fc4f5-nsbbv   1/1     Running             0          9m1s
example-registry-quay-database-66969cd859-n2ssm      1/1     Running             0          6d1h
```

```
example-registry-quay-mirror-758fc68ff7-5wxlp    1/1   Running   0   8m29s
example-registry-quay-mirror-758fc68ff7-lbl82    1/1   Running   0   8m29s
example-registry-quay-redis-7cc5f6c977-956g8     1/1   Running   0   5d21h
```

8. 在您的浏览器中，访问 registry 端点并验证证书是否已适当更新。例如：

```
Common Name (CN) example-registry-quay-quay-enterprise.apps.docs.quayteam.org
Organisation (O) DOCS
Organisational Unit (OU) QUAY
```

5.2.2.3. 使用 UI 创建构建触发器

使用以下步骤使用 UI 创建构建触发器。

流程

1. 登录到您的 Red Hat Quay 存储库。
2. 单击 **Create New Repository**，再创建一个新 registry，如 **testrepo**。
3. 在 **Repositories** 页面上，单击导航窗格上的 **Builds** 选项卡。或者，直接使用对应的 URL：

```
https://example-registry-quay-quay-
enterprise.apps.docs.quayteam.org/repository/quayadmin/testrepo?tab=builds
```

重要

在某些情况下，构建器可能会遇到解析主机名的问题。这个问题可能与作业对象上的 **dnsPolicy** 设置为 **default** 相关。目前，这个问题还没有临时解决方案。它将在以后的 Red Hat Quay 版本中解决。

4. 点 **Create Build Trigger→ Custom Git Repository Push**。
5. 输入用于克隆 Git 存储库的 HTTPS 或 SSH 样式 URL，然后单击 **Continue**。例如：

```
https://github.com/gabriel-rh/actions_test.git
```

6. **使用分支或标签名称检查 Tag 清单**，然后单击 **Continue**。
7. 在调用触发器时输入要构建的 Dockerfile 的位置，如 **/Dockerfile**，然后单击 **Continue**。
8. 输入 Docker 构建的上下文的位置，如 /，然后单击 **Continue**。
9. 如果保证，请创建一个 Robot 帐户。否则，点 **Continue**。
10. 点 **Continue** 来验证参数。
11. 在 **Builds** 页面中，点 Trigger Name 的 **Options** 图标，然后点 **Run Trigger Now**。
12. 从 Git 存储库输入提交 SHA，然后单击 **Start Build**。
13. 您可以通过单击 **Build History** 页面中的提交或运行 **oc get pods -n virtual-builders** 来检查构建的状态。例如：

```
$ oc get pods -n virtual-builders
```

输出示例

```
NAME                                     READY   STATUS    RESTARTS   AGE
f192fe4a-c802-4275-bcce-d2031e635126-9l2b5-25lg2   1/1     Running   0          7s
```

```
$ oc get pods -n virtual-builders
```

输出示例

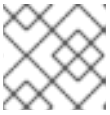
```
NAME                                     READY   STATUS      RESTARTS   AGE
f192fe4a-c802-4275-bcce-d2031e635126-9l2b5-25lg2   1/1     Terminating   0          9s
```

```
$ oc get pods -n virtual-builders
```

输出示例

```
No resources found in virtual-builders namespace.
```

14. 构建完成后，您可以检查导航窗格上 **Tags** 下的标签状态。



注意

通过早期访问，目前构建的完整构建日志和时间戳当前不可用。

5.2.2.4. 修改 AWS S3 存储桶

如果使用 AWS S3 存储，则必须在运行构建器前在 AWS 控制台中更改存储桶。

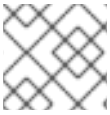
流程

1. 在 s3.console.aws.com 上登录到 AWS 控制台。
2. 在搜索栏中，搜索 **S3**，然后单击 **S3**。
3. 点存储桶的名称，如 **myawsbucket**。
4. 单击**权限**选项卡。
5. 在 **Cross-origin 资源共享(CORS)** 下，包含以下参数：

```
[
    {
        "AllowedHeaders": [
            "Authorization"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
```

```
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 3000
    },
    {
        "AllowedHeaders": [
            "Content-Type",
            "x-amz-acl",
            "origin"
        ],
        "AllowedMethods": [
            "PUT"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 3000
    }
]
```

5.2.2.5. 修改 Google Cloud Platform 对象存储桶



注意

目前，IBM Power 和 IBM Z 不支持修改 Google Cloud Platform 对象存储桶。

使用以下步骤为虚拟构建器配置跨原始资源共享(CORS)。

注意

如果没有 CORS 配置，上传构建 Dockerfile 会失败。

流程

1. 使用以下引用来创建 JSON 文件，以满足您的特定 CORS 需求。例如：

```
$ cat gcp_cors.json
```

输出示例

```
[
  {
    "origin": ["*"],
    "method": ["GET"],
    "responseHeader": ["Authorization"],
    "maxAgeSeconds": 3600
  },
  {
    "origin": ["*"],
    "method": ["PUT"],
    "responseHeader": [
          "Content-Type",
```

```
            "x-goog-acl",
            "origin"],
      "maxAgeSeconds": 3600
    }
]
```

2. 输入以下命令更新 GCP 存储桶：

```
$ gcloud storage buckets update gs://<bucket_name> --cors-file=./gcp_cors.json
```

输出示例

```
Updating
  Completed 1
```

3. 您可以运行以下命令来显示 GCP 存储桶的更新 CORS 配置：

```
$ gcloud storage buckets describe gs://<bucket_name>  --format="default(cors)"
```

输出示例

```
cors:
- maxAgeSeconds: 3600
  method:
  - GET
  origin:
  - '*'
  responseHeader:
  - Authorization
- maxAgeSeconds: 3600
  method:
  - PUT
  origin:
  - '*'
  responseHeader:
  - Content-Type
  - x-goog-acl
  - origin
```

第 6 章 GEO-REPLICATION

注意

目前，IBM Power 不支持 geo-replication 功能。

地理复制允许多个地理分布的 Red Hat Quay 部署从客户端或用户的角度来看作为单个 registry 工作。它显著提高了在全局分布式 Red Hat Quay 设置中的推送和拉取性能。镜像数据在后台异步复制，并带有透明故障转移，并为客户端重定向。

在独立和 Operator 部署中支持部署带有 geo-replication 的 Red Hat Quay。

其他资源

- 有关异地复制功能架构的更多信息，请参阅 架构指南， 包括技术图和高级别概述。

6.1. 地域复制功能

- 配置 geo-replication 后，容器镜像推送将写入该 Red Hat Quay 实例的首选存储引擎。这通常是区域内最接近的存储后端。
- 在初始推送后，镜像数据将在后台复制到其他存储引擎。
- 复制位置列表可以配置，它们可以是不同的存储后端。
- 镜像拉取(pull)将始终使用最接近的可用存储引擎来最大化拉取性能。
- 如果复制还没有完成，则拉取将使用源存储后端。

6.2. 地域复制要求和限制

- 在地理复制设置中，Red Hat Quay 要求所有区域都可以读取和写入所有其他区域的对象存储。对象存储必须可以被所有其他区域访问。
- 如果一个地理复制站点的对象存储系统失败，该站点的 Red Hat Quay 部署必须被关闭，以便客户端由全局负载均衡器重定向到具有完整存储系统的剩余站点。否则，客户端将遇到拉取和推送失败。
- Red Hat Quay 没有内部感知连接的对象存储系统的健康状态或可用性。用户必须配置一个全局负载均衡器(LB)，以监控分布式系统的健康状况，并根据存储状态将流量路由到不同的站点。
- 要检查 geo-replication 部署的状态，您必须使用 **/health/endtoend** checkpoint，该检查点用于全局健康监控。您必须使用 **/health/endtoend** 端点手动配置重定向。**/health/instance** 端点仅检查本地实例健康状况。
- 如果一个站点的对象存储系统不可用，则其余站点或站点没有自动重定向到剩余的存储系统或系统。
- 地理复制（geo-replication）是异步的。如果一个站点永久丢失，则已存储在该站点的对象存储系统中，但在失败时还没有复制到剩余的站点的数据会丢失。
- 单个数据库（因此所有元数据和 Red Hat Quay 配置）在所有区域之间共享。
  地理复制不会复制数据库。如果出现停机，启用了地理复制功能的 Red Hat Quay 不会切换到另一个数据库。

- 单个 Redis 缓存在整个 Red Hat Quay 设置间共享，需要可以被所有 Red Hat Quay pod 访问。
- 完全相同的配置应该在所有区域间使用，但存储后端除外，该配置也可以使用 **QUAY_DISTRIBUTED_STORAGE_PREFERENCE** 环境变量明确配置。
- 地理复制需要每个区域中的对象存储。它不适用于本地存储。
- 每个区域必须能够访问每个区域中的每个存储引擎，这需要一个网络路径。
- 或者，可以使用存储代理选项。
- 整个存储后端（如所有 blob）都会被复制。相反，存储库镜像可以限制为存储库或镜像。
- 所有 Red Hat Quay 实例都必须共享相同的入口点，通常通过负载均衡器。
- 所有 Red Hat Quay 实例都必须具有相同的超级用户集合，因为它们在通用配置文件中定义。
- 地理复制要求您的 Clair 配置设置为 **unmanaged**。非受管 Clair 数据库允许 Red Hat Quay Operator 在地理复制环境中工作，其中多个 Red Hat Quay Operator 实例必须与同一数据库通信。如需更多信息，请参阅高级 Clair 配置。
- geo-Replication 需要 SSL/TLS 证书和密钥。如需更多信息，请参阅使用 SSL/TLS 保护到 Red Hat Quay 的连接。

如果无法满足上述要求，您应该使用两个不同的 Red Hat Quay 部署，并利用存储库镜像功能。

6.2.1. 在 OpenShift Container Platform 中设置地理复制

使用以下步骤在 OpenShift Container Platform 上设置异地复制。

流程

1. 为 Red Hat Quay 部署 postgres 实例。
2. 输入以下命令登录到数据库：

```
psql -U <username> -h <hostname> -p <port> -d <database_name>
```

3. 为 Red Hat Quay 创建名为 **quay** 的数据库。例如：

```
CREATE DATABASE quay;
```

4. 在数据库中启用 pg_trm 扩展

```
\c quay;
CREATE EXTENSION IF NOT EXISTS pg_trgm;
```

5. 部署 Redis 实例：

注意

- 如果您的云供应商有自己的服务，则部署 Redis 实例可能是必需的。
- 如果您使用 Builders，则需要部署 Redis 实例。

a. 为 Redis 部署虚拟机

b. 验证可以从运行 Red Hat Quay 的集群访问它

c. 必须打开端口 6379/TCP

d. 在实例内运行 Redis

```
sudo dnf install -y podman
podman run -d --name redis -p 6379:6379 redis
```

6. 创建两个对象存储后端，每个集群一个。理想情况下，一个对象存储桶将接近第一个或主、集群，另一个将接近第二个或次要集群。

7. 使用环境变量覆盖来部署具有相同配置捆绑包的集群，以便为单个集群选择适当的存储后端。

8. 配置负载均衡器以为集群提供单一入口点。

6.2.1.1. 在 OpenShift Container Platform 上为 Red Hat Quay 配置 geo-replication

使用以下步骤为 OpenShift Container Platform 上的 Red Hat Quay 配置 geo-replication。

流程

1. 创建在集群之间共享的 **config.yaml** 文件。此 **config.yaml** 文件包含常见 PostgreSQL、Redis 和存储后端的详情：

geo-replication config.yaml 文件

```
SERVER_HOSTNAME: <georep.quayteam.org or any other name> 1
DB_CONNECTION_ARGS:
  autorollback: true
  threadlocals: true
DB_URI: postgresql://postgres:password@10.19.0.1:5432/quay 2
BUILDLOGS_REDIS:
  host: 10.19.0.2
  port: 6379
USER_EVENTS_REDIS:
  host: 10.19.0.2
  port: 6379
DATABASE_SECRET_KEY: 0ce4f796-c295-415b-bf9d-b315114704b8
DISTRIBUTED_STORAGE_CONFIG:
  usstorage:
    - GoogleCloudStorage
    - access_key: GOOGQGPGVMASAAMQABCDEFG
      bucket_name: georep-test-bucket-0
      secret_key: AYWfEaxX/u84XRA2vUX5C987654321
      storage_path: /quaygcp
  eustorage:
    - GoogleCloudStorage
    - access_key: GOOGQGPGVMASAAMQWERTYUIOP
      bucket_name: georep-test-bucket-1
      secret_key: AYWfEaxX/u84XRA2vUX5Cuj12345678
      storage_path: /quaygcp
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS:
```

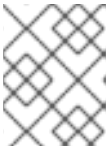
```
  - usstorage
  - eustorage
DISTRIBUTED_STORAGE_PREFERENCE:
  - usstorage
  - eustorage
FEATURE_STORAGE_REPLICATION: true
```

1. 必须使用正确的 **SERVER_HOSTNAME** 用于路由，并且必须与全局负载均衡器的主机名匹配。
2. 要检索使用 OpenShift Container Platform Operator 部署的 Clair 实例的配置文件，请参阅检索 Clair 配置。

2. 运行以下命令来创建 **configBundleSecret** ：

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

3. 在每个集群中，设置 **configBundleSecret**，并使用 **QUAY_DISTRIBUTED_STORAGE_PREFERENCE** 环境变量覆盖来配置该集群的相应存储。例如：



注意

两个部署之间的 **config.yaml** 文件都必须匹配。如果对一个集群进行更改，还必须在另一个集群中更改它。

美国集群 QuayRegistry 示例

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  configBundleSecret: georep-config-bundle
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: postgres
      managed: false
    - kind: clairpostgres
      managed: false
    - kind: redis
      managed: false
    - kind: quay
      managed: true
      overrides:
        env:
        - name: QUAY_DISTRIBUTED_STORAGE_PREFERENCE
```

```
      value: usstorage
  - kind: mirror
    managed: true
    overrides:
      env:
      - name: QUAY_DISTRIBUTED_STORAGE_PREFERENCE
        value: usstorage
```

注意

因为 SSL/TLS 是非受管的，并且路由被管理，所以您必须在配置捆绑包中直接提供证书。如需更多信息，请参阅配置 TLS 和路由。

欧洲集群

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  configBundleSecret: georep-config-bundle
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: postgres
      managed: false
    - kind: clairpostgres
      managed: false
    - kind: redis
      managed: false
    - kind: quay
      managed: true
      overrides:
        env:
        - name: QUAY_DISTRIBUTED_STORAGE_PREFERENCE
          value: eustorage
    - kind: mirror
      managed: true
      overrides:
        env:
        - name: QUAY_DISTRIBUTED_STORAGE_PREFERENCE
          value: eustorage
```

注意

因为 SSL/TLS 是非受管的，并且路由被管理，所以您必须在配置捆绑包中直接提供证书。如需更多信息，请参阅配置 TLS 和路由。

6.2.2. 地域复制的混合存储

Red Hat Quay geo-replication 支持使用不同和多个复制目标，例如，在公共云中使用 AWS S3 存储并在内部使用 Ceph 存储。这与授予对所有 Red Hat Quay Pod 和集群节点中所有存储后端的访问权限的关键要求。因此，建议您使用以下内容：

- VPN 以防止内部存储可见性，*或者*
- 允许访问 Red Hat Quay 使用的指定存储桶的令牌对

这会导致 Red Hat Quay 的公共云实例可以访问内部存储，但网络会被加密、保护并将使用 ACL，从而满足安全要求。

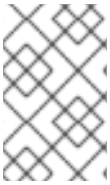
如果您无法实现这些安全措施，则最好部署两个不同的 Red Hat Quay registry，并使用存储库镜像作为地理复制的替代选择。

6.3. 在 OPENSHIFT CONTAINER PLATFORM 上升级 RED HAT QUAY 的 GEO-REPLICATION 部署

使用以下步骤升级 OpenShift Container Platform 部署上的地理复制 Red Hat Quay。

重要

- 当将 OpenShift Container Platform 上的 geo-replicated Red Hat Quay 部署升级到下一个 y-stream 版本（例如，Red Hat Quay 3.7 → Red Hat Quay 3.8），您必须在升级前停止操作。
- 故障时间从一个 y-stream 版本升级到下一个版本会间歇性。
- 在升级前，强烈建议您在 OpenShift Container Platform 上部署备份 Red Hat Quay。



流程

此流程假设您在三个或更多系统上运行 Red Hat Quay registry。对于此过程，我们将假设三个名为 **System A**、 **System B** 和 **System C** 的系统。**系统 A** 将充当部署 Red Hat Quay Operator 的主要系统。

1. 在 System B 和 System C 上，缩减您的 Red Hat Quay registry。这可以通过禁用自动扩展并覆盖 Red Hat Quay、镜像 worker 和 Clair 的副本计数（如果被管理）。使用以下 **quayregistry.yaml** 文件作为参考：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: registry
  namespace: ns
spec:
  components:
    …
    - kind: horizontalpodautoscaler
      managed: false (1)
    - kind: quay
      managed: true
```

```
    overrides: 2
      replicas: 0
  - kind: clair
    managed: true
    overrides:
      replicas: 0
  - kind: mirror
    managed: true
    overrides:
      replicas: 0
  ...
```

1 禁用 **Quay**、**Clair** 和 **mirror** worker 的自动扩展

2 对于访问数据库和 objectstorage 的组件，将副本数设置为 0

注意

您必须保留在 System A 上运行的 Red Hat Quay registry。不要更新 System A 上的 **quayregistry.yaml** 文件。

2. 等待 **registry-quay-app**、**registry-quay-mirror** 和 **registry-clair-app** pod 消失。输入以下命令检查其状态：

```
oc get pods -n <quay-namespace>
```

输出示例

```
quay-operator.v3.7.1-6f9d859bd-p5ftc               1/1     Running     0             12m
quayregistry-clair-postgres-7487f5bd86-xnxpr       1/1     Running     1 (12m ago)   12m
quayregistry-quay-app-upgrade-xq2v6                0/1     Completed   0             12m
quayregistry-quay-redis-84f888776f-hhgms           1/1     Running     0             12m
```

3. 在 System A 上，启动 Red Hat Quay 升级到最新的 y-stream 版本。这是手动过程。有关升级安装的 Operator 的更多信息，请参阅 升级已安装的 Operator。如需有关 Red Hat Quay 升级路径的更多信息，请参阅升级 Red Hat Quay Operator。

4. 安装新的 Red Hat Quay registry 后，集群中的必要升级会自动完成。之后，新的 Red Hat Quay pod 会使用最新的 y-stream 版本启动。另外，还会调度并启动新的 **Quay** pod。

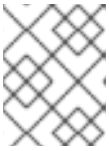
5. 通过进入到 Red Hat Quay UI 确认更新是否正常工作：

   a. 在 **OpenShift** 控制台中，导航到 **Operators → Installed Operators**，然后点击 **Registry Endpoint** 链接。

重要

在 Red Hat Quay UI 可用前，不要执行以下步骤。在系统 A 上提供 UI 前，不要升级 System B 和 System C 上的 Red Hat Quay registry。

6. 确认更新已在 System A 上正常工作，在 System B 和 System C 上启动 Red Hat Quay 升级。Operator 升级会生成升级的 Red Hat Quay 安装，并且 pod 被重启。



注意

因为数据库架构对于新的 y-stream 安装正确，System B 和 System C 上的新 pod 应该快速启动。

7. 更新后，通过删除对组件的 **覆盖** 来恢复此步骤 1 中所做的更改。例如：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: registry
  namespace: ns
spec:
  components:
    …
    - kind: horizontalpodautoscaler
      managed: true (1)
    - kind: quay
      managed: true
    - kind: clair
      managed: true
    - kind: mirror
      managed: true
    …
```

1. 如果在升级过程前将 **horizontalpodautoscaler** 资源设置为 **true**，或者您希望 Red Hat Quay 在资源短时扩展，请将其设为 **true**。

6.3.1. 从 OpenShift Container Platform 部署的 Red Hat Quay 中删除地理复制站点

通过按照以下流程，Red Hat Quay 管理员可以删除地理复制设置中的站点。

先决条件

- 已登陆到 OpenShift Container Platform。
- 您已为 Red Hat Quay geo-replication 配置至少两个站点，例如 **usstorage** 和 **eustorage**。
- 每个站点都有自己的组织、存储库和镜像标签。

流程

1. 运行以下命令，在所有定义的站点间同步 Blob：

```
$ python -m util.backfillreplication
```

警告

在从 Red Hat Quay **config.yaml** 文件中删除存储引擎 **前，您必须确保** 所有 Blob 在所有定义的站点间同步。

运行此命令时，会创建复制 worker 提取的复制作业。如果存在需要复制的 Blob，脚本会返回要复制的 Blob 的 UUID。如果您多次运行此命令，并且返回脚本的输出为空，这并不意味着复制过程已完成；这意味着没有为复制进行排队。在继续操作前，客户应该使用适当的 judgement，因为分配的时间复制取决于检测到的 Blob 数量。

另外，您可以使用 Microsoft Azure 等第三方云工具来检查同步状态。

在继续操作前，必须完成此步骤。

2. 在站点 **usstorage** 的 Red Hat Quay **config.yaml** 文件中，删除 **eustorage** 站点的 **DISTRIBUTED_STORAGE_CONFIG** 条目。

3. 输入以下命令识别您的 **Quay** 应用程序 pod：

```
$ oc get pod -n <quay_namespace>
```

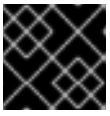
输出示例

```
quay390usstorage-quay-app-5779ddc886-2drh2
quay390eustorage-quay-app-66969cd859-n2ssm
```

4. 输入以下命令在 **usstorage** pod 中打开交互式 shell 会话：

```
$ oc rsh quay390usstorage-quay-app-5779ddc886-2drh2
```

5. 输入以下命令永久删除 **eustorage** 站点：



重要

无法撤销以下操作。请谨慎使用。

```
sh-4.4$ python -m util.removelocation eustorage
```

输出示例

```
WARNING: This is a destructive operation. Are you sure you want to remove eustorage from
your storage locations? [y/n] y
Deleted placement 30
Deleted placement 31
Deleted placement 32
Deleted placement 33
Deleted location eustorage
```

第 7 章 备份和恢复由 RED HAT QUAY OPERATOR 管理的 RED HAT QUAY

在 OpenShift Container Platform 上由 Red Hat Quay Operator 管理时，使用本节中的内容来备份和恢复 Red Hat Quay

7.1. 可选：在 OPENSHIFT CONTAINER PLATFORM 中为 RED HAT QUAY 启用只读模式

在 OpenShift Container Platform 部署中为 Red Hat Quay 启用只读模式，您可以管理 registry 的操作。管理员可以启用只读模式来限制对 registry 的写入访问，这有助于确保数据完整性，降低维护窗口期间的风险，并提供防止对 registry 数据的意外修改。它还可帮助您确保 Red Hat Quay registry 保持在线状态，并可供用户提供镜像。

在备份和恢复时，您需要在 OpenShift Container Platform 部署中缩减 Red Hat Quay。这会导致在备份期间服务不可用，在某些情况下，可能无法接受。在 OpenShift Container Platform 部署中为 Red Hat Quay 启用只读模式可确保在备份和恢复过程中服务可用性。

先决条件

- 如果您使用 Red Hat Enterprise Linux (RHEL) 7.x：
  - 您已启用了 Red Hat Software Collections List (RHSCL)。
  - 已安装 Python 3.6。
  - 您已下载了 **virtualenv** 软件包。
  - 已安装 **git** CLI。
- 如果您使用 Red Hat Enterprise Linux (RHEL) 8：
  - 您已在机器上安装 Python 3。
  - 您已下载了 **python3-virtualenv** 软件包。
  - 已安装 **git** CLI。
- 您已克隆了 **https://github.com/quay/quay.git** 软件仓库。
- 已安装 **oc** CLI。
- 您可以使用 **cluster-admin** 权限访问集群。

7.1.1. 在 OpenShift Container Platform 上为 Red Hat Quay 创建服务密钥

Red Hat Quay 使用服务密钥与各种组件通信。这些密钥用于签署已完成的请求，如请求扫描镜像、登录、存储访问等。

流程

1. 输入以下命令获取 Red Hat Quay pod 列表：

```
$ oc get pods -n <namespace>
```

输出示例

```
example-registry-clair-app-7dc7ff5844-4skw5          0/1     Error                 0          70d
example-registry-clair-app-7dc7ff5844-nvn4f          1/1     Running               0          31d
example-registry-clair-app-7dc7ff5844-x4smw          0/1     ContainerStatusUnknown   6 (70d
ago)   70d
example-registry-clair-app-7dc7ff5844-xjnvt          1/1     Running               0          60d
example-registry-clair-postgres-547d75759-75c49      1/1     Running               0          70d
example-registry-quay-app-76c8f55467-52wjz           1/1     Running               0          70d
example-registry-quay-app-76c8f55467-hwz4c           1/1     Running               0          70d
example-registry-quay-app-upgrade-57ghs              0/1     Completed             1          70d
example-registry-quay-database-7c55899f89-hmnm6      1/1     Running               0
70d
example-registry-quay-mirror-6cccbd76d-btsnb         1/1     Running               0          70d
example-registry-quay-mirror-6cccbd76d-x8g42         1/1     Running               0          70d
example-registry-quay-redis-85cbdf96bf-4vk5m         1/1     Running               0          70d
```

2. 输入以下命令打开到 **Quay** 容器的远程 shell 会话：

```
$ oc rsh example-registry-quay-app-76c8f55467-52wjz
```

3. 输入以下命令来创建所需的服务密钥：

```
sh-4.4$ python3 tools/generatekeypair.py quay-readonly
```

输出示例

```
Writing public key to quay-readonly.jwk
Writing key ID to quay-readonly.kid
Writing private key to quay-readonly.pem
```

7.1.2. 将密钥添加到 PostgreSQL 数据库

使用以下步骤将服务密钥添加到 PostgreSQL 数据库中。

先决条件

- 您已创建了服务密钥。

流程

1. 输入以下命令进入 Red Hat Quay 数据库环境：

```
$ oc rsh example-registry-quay-app-76c8f55467-52wjz psql -U <database_username> -d
<database_name>
```

2. 输入以下命令显示 **servicekeyapproval** 的批准类型和相关备注：

```
quay=# select * from servicekeyapproval;
```

输出示例

```
 id | approver_id |         approval_type          |       approved_date        | notes
----+-------------+--------------------------------+----------------------------+-------
  1 |             | ServiceKeyApprovalType.AUTOMATIC | 2024-05-07 03:47:48.181347 |
  2 |             | ServiceKeyApprovalType.AUTOMATIC | 2024-05-07 03:47:55.808087 |
  3 |             | ServiceKeyApprovalType.AUTOMATIC | 2024-05-07 03:49:04.27095  |
  4 |             | ServiceKeyApprovalType.AUTOMATIC | 2024-05-07 03:49:05.46235  |
  5 |           1 | ServiceKeyApprovalType.SUPERUSER | 2024-05-07 04:05:10.296796 |
...
```

3. 输入以下查询，将服务密钥添加到您的 Red Hat Quay 数据库中：

```
quay=# INSERT INTO servicekey
  (name, service, metadata, kid, jwk, created_date, expiration_date)
  VALUES ('quay-readonly',
          'quay',
          '{}',
          '{<contents_of_.kid_file>}',
          '{<contents_of_.jwk_file>}',
          '{<created_date_of_read-only>}',
          '{<expiration_date_of_read-only>}');
```

输出示例

```
INSERT 0 1
```

4. 接下来，使用以下查询添加密钥批准：

```
quay=# INSERT INTO servicekeyapproval ('approval_type', 'approved_date', 'notes')
  VALUES ("ServiceKeyApprovalType.SUPERUSER", "CURRENT_DATE",
          {include_notes_here_on_why_this_is_being_added});
```

输出示例

```
INSERT 0 1
```

5. 将创建的服务键行中的 **approval_ id** 字段设置为创建的服务密钥批准中的 id 字段。您可以使用以下 **SELECT** 语句来获取必要的 ID：

```
UPDATE servicekey
SET approval_id = (SELECT id FROM servicekeyapproval WHERE approval_type =
'ServiceKeyApprovalType.SUPERUSER')
WHERE name = 'quay-readonly';
```

```
UPDATE 1
```

7.1.3. 在 OpenShift Container Platform 中配置只读模式 Red Hat Quay

在创建了服务密钥并添加到 PostgreSQL 数据库后，您必须在 OpenShift Container Platform 部署中重启 **Quay** 容器。

重要

以只读模式在 OpenShift Container Platform 上部署 Red Hat Quay 要求您修改存储在 OpenShift Container Platform 集群中的 secret。强烈建议您在更改 secret 前创建 secret 备份。

先决条件

- 您已创建了服务密钥，并将它们添加到 PostgreSQL 数据库中。

流程

1. 输入以下命令在 OpenShift Container Platform 部署中读取 Red Hat Quay 的 secret 名称：

```
$ oc get deployment -o yaml <quay_main_app_deployment_name>
```

2. 使用 **base64** 命令对 **quay-readonly.kid** 和 **quay-readonly.pem** 文件进行编码：

```
$ base64 -w0 quay-readonly.kid
```

输出示例

```
ZjUyNDFm...
```

```
$ base64 -w0 quay-readonly.pem
```

输出示例

```
LS0tLS1CRUdJTiBSU0E...
```

3. 输入以下命令来获取当前的配置捆绑包和 secret：

```
$ oc get secret quay-config-secret-name -o json | jq '.data."config.yaml"' | cut -d '"' -f2 |
base64 -d -w0 > config.yaml
```

4. 编辑 **config.yaml** 文件并添加以下信息：

```
# ...
REGISTRY_STATE: readonly
INSTANCE_SERVICE_KEY_KID_LOCATION: 'conf/stack/quay-readonly.kid'
INSTANCE_SERVICE_KEY_LOCATION: 'conf/stack/quay-readonly.pem'
# ...
```

5. 运行以下命令保存文件和 **base64** 编码：

```
$ base64 -w0 quay-config.yaml
```

6. 将 Red Hat Quay Operator pod 缩减为 **0**。这样可确保 Operator 在编辑后不会协调 secret。

```
$ oc scale --replicas=0 deployment quay-operator -n openshift-operators
```

7. 编辑 secret 使其包含新内容：

```
$ oc edit secret quay-config-secret-name -n quay-namespace
```

```
# ...
data:
  "quay-readonly.kid": "ZjUyNDFm..."
  "quay-readonly.pem": "LS0tLS1CRUdJTiBSU0E..."
  "config.yaml": "QUNUSU9OX0xPR19..."
# ...
```

在只读模式下使用 Red Hat Quay on OpenShift Container Platform 部署，您可以安全地管理 registry 的操作并执行诸如备份和恢复的操作。

7.1.3.1. 从只读部署扩展 OpenShift Container Platform 上的 Red Hat Quay

当您不再需要 OpenShift Container Platform 上的 Red Hat Quay 处于只读模式时，您可以扩展部署，并从 secret 中删除添加的内容。

流程

1. 编辑 **config.yaml** 文件并删除以下信息：

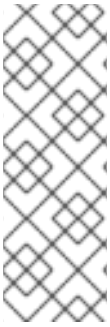
```
# ...
REGISTRY_STATE: readonly
INSTANCE_SERVICE_KEY_KID_LOCATION: 'conf/stack/quay-readonly.kid'
INSTANCE_SERVICE_KEY_LOCATION: 'conf/stack/quay-readonly.pem'
# ...
```

2. 输入以下命令扩展 Red Hat Quay Operator：

```
oc scale --replicas=1 deployment quay-operator -n openshift-operators
```

7.2. 备份 RED HAT QUAY

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 不确保 PostgreSQL 数据库已备份。



注意

此流程涵盖了备份 Red Hat Quay PostgreSQL 数据库。它不包括备份 Clair PostgreSQL 数据库。严格说，不需要备份 Clair PostgreSQL 数据库，因为它可以重新创建。如果您选择从头开始重新创建，在 Red Hat Quay 部署中的所有镜像被扫描后，将等待信息被重新填充。在这个停机时间中，安全报告不可用。

如果您考虑备份 Clair PostgreSQL 数据库，您必须考虑其大小取决于存储在 Red Hat Quay 中的镜像数量。因此，数据库可能会非常大。

此流程描述了如何使用 Operator 在 OpenShift Container Platform 上创建 Red Hat Quay 备份。

先决条件

- 使用 Red Hat Quay Operator 在 OpenShift Container Platform 上部署健康的 Red Hat Quay 部署。**Available** 状态条件被设置为 **true**。

- 组件 **quay**、**postgres** 和 **objectstorage** 设置为 **managed: true**
- 如果组件 **clair** 设为 **managed: true**，则组件 **clairpostgres** 也被设为 **managed: true** （从 Red Hat Quay v3.7 或更高版本开始）

注意

如果您的部署包含部分非受管数据库或存储组件，且您使用外部服务进行 PostgreSQL 或 S3 兼容对象存储来运行 Red Hat Quay 部署，则必须参考服务提供商或厂商文档来创建数据的备份。您可以参阅本指南中描述的工具，作为如何备份外部 PostgreSQL 数据库或对象存储的起点。

7.2.1. Red Hat Quay 配置备份

使用以下步骤备份 Red Hat Quay 配置。

流程

1. 要通过导出 **QuayRegistry** 自定义资源来支持 QuayRegistry 自定义资源，请输入以下命令：

```
$ oc get quayregistry <quay_registry_name> -n <quay_namespace> -o yaml > quay-registry.yaml
```

2. 编辑生成的 **quayregistry.yaml** 并删除 status 部分和以下 metadata 字段：

```
metadata.creationTimestamp
metadata.finalizers
metadata.generation
metadata.resourceVersion
metadata.uid
```

3. 输入以下命令备份受管密钥 secret：

注意

如果您正在运行一个早于 Red Hat Quay 3.7.0 的版本，可以跳过此步骤。第一次部署 Red Hat Quay 时会自动生成一些 secret。它们存储在 **QuayRegistry** 资源命名空间中的名为 &lt **;quay_registry_name>-quay_registry_managed_secret_keys** 的 secret 中。

```
$ oc get secret -n <quay_namespace>
<quay_registry_name>_quay_registry_managed_secret_keys -o yaml >
managed_secret_keys.yaml
```

4. 编辑生成的 **managed_secret_keys.yaml** 文件，并删除条目 **metadata.ownerReferences**。**managed_secret_keys.yaml** 文件应该类似如下：

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: <quayname>_quay_registry_managed_secret_keys>
  namespace: <quay_namespace>
```

```
data:
  CONFIG_EDITOR_PW: <redacted>
  DATABASE_SECRET_KEY: <redacted>
  DB_ROOT_PW: <redacted>
  DB_URI: <redacted>
  SECRET_KEY: <redacted>
  SECURITY_SCANNER_V4_PSK: <redacted>
```

data 属性下的所有信息都应保持不变。

5. 输入以下命令重定向当前的 **Quay** 配置文件：

```
$ oc get secret -n <quay-namespace>  $(oc get quayregistry <quay_registry_name> -n <quay_namespace>  -o jsonpath='{.spec.configBundleSecret}') -o yaml > config-bundle.yaml
```

6. 备份挂载到 **Quay** pod 中的 **/conf/stack/config.yaml** 文件：

```
$ oc exec -it quay_pod_name -- cat /conf/stack/config.yaml > quay_config.yaml
```

7.2.2. 缩减 Red Hat Quay 部署

使用以下步骤缩减 Red Hat Quay 部署。

重要

此步骤需要创建 Red Hat Quay 部署状态的一致性备份。不要省略这一步，包括在由外部服务（由 Red Hat Quay Operator 管理）提供的 PostgreSQL 数据库和/或 S3 兼容对象存储的设置中。

流程

1. 根据 Red Hat Quay 部署的版本，使用以下选项之一缩减部署。

   a. **对于 Operator 版本 3.7 及更新版本：** 通过禁用自动扩展并覆盖 Red Hat Quay、mirror worker 和 Clair （如果管理）的副本数来扩展 Red Hat Quay 部署。您的 **QuayRegistry** 资源应类似于如下：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: registry
  namespace: ns
spec:
  components:
    …
    - kind: horizontalpodautoscaler
      managed: false (1)
    - kind: quay
      managed: true
      overrides: (2)
        replicas: 0
    - kind: clair
      managed: true
      overrides:
```

```
    replicas: 0
  - kind: mirror
    managed: true
    overrides:
      replicas: 0
  …
```

1. 禁用 Quay、Clair 和镜像 worker 的自动扩展
2. 对于访问数据库和 objectstorage 的组件，将副本数设置为 0

b. **对于 Operator 版本 3.6 及更早版本**：首先缩减 Red Hat Quay registry，然后缩减托管的 Red Hat Quay 资源，缩减 Red Hat Quay 部署：

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-operator-namespace>|awk '/^quay-operator/ {print $1}') -n <quay-operator-namespace>
```

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-namespace>|awk '/quay-app/ {print $1}') -n <quay-namespace>
```

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-namespace>|awk '/quay-mirror/ {print $1}') -n <quay-namespace>
```

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-namespace>|awk '/clair-app/ {print $1}') -n <quay-namespace>
```

2. 等待 **registry-quay-app**、**registry-quay-mirror** 和 **registry-clair-app** pod （取决于您被设置为由 Red Hat Quay Operator 管理的组件）消失。您可以运行以下命令来检查其状态：

```
$ oc get pods -n <quay_namespace>
```

输出示例：

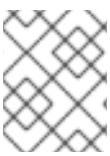
```
$ oc get pod
```

输出示例

```
quay-operator.v3.7.1-6f9d859bd-p5ftc               1/1     Running     0          12m
quayregistry-clair-postgres-7487f5bd86-xnxpr       1/1     Running     1 (12m ago)   12m
quayregistry-quay-app-upgrade-xq2v6                0/1     Completed   0          12m
quayregistry-quay-database-859d5445ff-cqthr        1/1     Running     0          12m
quayregistry-quay-redis-84f888776f-hhgms           1/1     Running     0          12m
```

7.2.3. 备份 Red Hat Quay 管理的数据库

使用以下步骤备份 Red Hat Quay 管理的数据库。



注意

如果您的 Red Hat Quay 部署配置了外部或不需要的 PostgreSQL 数据库，请参阅您的厂商文档，了解如何创建这些数据库的一致性备份。

流程

1. 识别 Quay PostgreSQL pod 名称：

```
$ oc get pod -l quay-component=postgres -n <quay_namespace> -o
jsonpath='{.items[0].metadata.name}'
```

输出示例：

```
quayregistry-quay-database-59f54bb7-58xs7
```

2. 获取 Quay 数据库名称：

```
$ oc -n <quay_namespace> rsh $(oc get pod -l app=quay -o NAME -n <quay_namespace>
|head -n 1) cat /conf/stack/config.yaml|awk -F"/" '/^DB_URI/ {print $4}'
quayregistry-quay-database
```

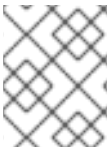
3. 下载备份数据库：

```
$ oc exec quayregistry-quay-database-59f54bb7-58xs7 -- /usr/bin/pg_dump -C quayregistry-
quay-database  > backup.sql
```

7.2.3.1. 备份 Red Hat Quay 管理的对象存储

使用以下步骤备份 Red Hat Quay 管理的对象存储。本节中的说明适用于以下配置：

- 独立、多云对象网关配置
- OpenShift Data Foundations 存储要求 Red Hat Quay Operator 通过 ObjectStorageBucketClaim API 从中置备 S3 对象存储桶



注意

如果您的 Red Hat Quay 部署配置了外部(unmanged)对象存储，请参阅厂商的文档，了解如何创建 Quay 存储存储桶的内容副本。

流程

1. 输入以下命令解码并导出 **AWS_ACCESS_KEY_ID** ：

```
$ export AWS_ACCESS_KEY_ID=$(oc get secret -l app=noobaa -n <quay-namespace>  -o
jsonpath='{.items[0].data.AWS_ACCESS_KEY_ID}' |base64 -d)
```

2. 输入以下命令解码并导出 **AWS_SECRET_ACCESS_KEY_ID** ：

```
$ export AWS_SECRET_ACCESS_KEY=$(oc get secret -l app=noobaa -n <quay-
namespace> -o jsonpath='{.items[0].data.AWS_SECRET_ACCESS_KEY}' |base64 -d)
```

3. 创建新目录：

```
$ mkdir blobs
```

注意

您还可以使用 rclone 或 sc3md，而不是 AWS 命令行工具。

1. 输入以下命令将所有 Blob 复制到目录中：

```
$ aws s3 sync --no-verify-ssl --endpoint https://$(oc get route s3 -n openshift-storage  -o
jsonpath='{.spec.host}')  s3://$(oc get cm -l app=noobaa -n <quay-namespace> -o
jsonpath='{.items[0].data.BUCKET_NAME}') ./blobs
```

7.2.4. 扩展 Red Hat Quay 部署

1. 根据 Red Hat Quay 部署的版本，使用以下选项之一扩展部署。

   a. **对于 Operator 版本 3.7 及更新版本：**通过重新启用自动扩展（如果需要），并删除 Quay、镜像 worker 和 Clair 的副本覆盖来扩展 Red Hat Quay 部署。您的 **QuayRegistry** 资源应类似于如下：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: registry
  namespace: ns
spec:
  components:
    …
    - kind: horizontalpodautoscaler
      managed: true (1)
    - kind: quay (2)
      managed: true
    - kind: clair
      managed: true
    - kind: mirror
      managed: true
    …
```

   (1) 重新启用 Quay 的自动扩展、Clair 和镜像 worker （如果需要）

   (2) 副本覆盖会再次删除以扩展 Quay 组件

   b. **对于 Operator 版本 3.6 及更早版本：**通过扩展 Red Hat Quay registry 来扩展 Red Hat Quay 部署：

```
$ oc scale --replicas=1 deployment $(oc get deployment -n
<quay_operator_namespace> | awk '/^quay-operator/ {print $1}') -n
<quay_operator_namespace>
```

2. 输入以下命令检查 Red Hat Quay 部署的状态：

```
$ oc wait quayregistry registry --for=condition=Available=true -n <quay_namespace>
```

输出示例：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  ...
  name: registry
  namespace: <quay-namespace>
  ...
spec:
  ...
status:
  - lastTransitionTime: '2022-06-20T05:31:17Z'
    lastUpdateTime: '2022-06-20T17:31:13Z'
    message: All components reporting as healthy
    reason: HealthChecksPassing
    status: 'True'
    type: Available
```

7.3. 恢复 RED HAT QUAY

当 Red Hat Quay Operator 管理数据库时，请使用以下步骤恢复 Red Hat Quay。它应在执行 Red Hat Quay registry 备份后执行。如需更多信息，请参阅 备份 Red Hat Quay。

先决条件

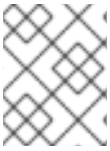
- 使用 Red Hat Quay Operator 在 OpenShift Container Platform 上部署 Red Hat Quay。
- 遵循 备份 Red Hat Quay Operator 中的说明创建了由 Red Hat Quay Operator 管理的 Red Hat Quay 配置的备份
- 您的 Red Hat Quay 数据库已被备份。
- Red Hat Quay 使用的对象存储存储桶已被备份。
- 组件 **quay**、**postgres** 和 **objectstorage** 设置为 **managed: true**
- 如果组件 **clair** 设为 **managed: true**，则组件 **clairpostgres** 也被设置为 **managed: true** （从 Red Hat Quay v3.7 或更高版本开始）
- 在 OpenShift Container Platform 集群上的目标命名空间中没有运行由 Red Hat Quay Operator 管理的 Red Hat Quay 部署

注意

如果您的部署包含部分非受管数据库或存储组件，且您使用外部服务进行 PostgreSQL 或 S3 兼容对象存储来运行 Red Hat Quay 部署，则必须在恢复 Red Hat Quay 前从备份中恢复其数据。

7.3.1. 从备份中恢复 Red Hat Quay 及其配置

使用以下步骤从备份中恢复 Red Hat Quay 及其配置文件。



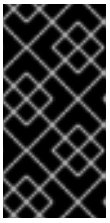
注意

这些说明假设您遵循了 备份 Red Hat Quay 指南中的流程，并创建具有相同名称的备份文件。

流程

1. 输入以下命令恢复备份的 Red Hat Quay 配置：

```
$ oc create -f ./config-bundle.yaml
```



重要

如果您接收到错误 **Error from server (AlreadyExists): error when creating "./config-bundle.yaml": secrets "config-bundle-secret" already exists**，您需要使用 **$ oc delete Secret config-bundle-secret -n <quay-namespace>** 删除现有的资源，并使用 **$ oc create -f ./config-bundle.yaml** 重新创建它。

2. 输入以下命令从备份中恢复生成的密钥：

```
$ oc create -f ./managed-secret-keys.yaml
```

3. 恢复 **QuayRegistry** 自定义资源：

```
$ oc create -f ./quay-registry.yaml
```

4. 检查 Red Hat Quay 部署的状态并等待它可用：

```
$ oc wait quayregistry registry --for=condition=Available=true -n <quay-namespace>
```

7.3.2. 缩减 Red Hat Quay 部署

使用以下步骤缩减 Red Hat Quay 部署。

流程

1. 根据 Red Hat Quay 部署的版本，使用以下选项之一缩减部署。

   a. **对于 Operator 版本 3.7 及更新版本：** 通过禁用自动扩展并覆盖 Quay、镜像 worker 和 Clair 的副本数（如果管理），扩展 Red Hat Quay 部署。您的 **QuayRegistry** 资源应类似于如下：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: registry
  namespace: ns
spec:
  components:
    …
    - kind: horizontalpodautoscaler
      managed: false (1)
    - kind: quay
      managed: true
```

```
    overrides: 2
      replicas: 0
  - kind: clair
    managed: true
    overrides:
      replicas: 0
  - kind: mirror
    managed: true
    overrides:
      replicas: 0
  …
```

1. 禁用 Quay、Clair 和镜像 worker 的自动扩展
2. 对于访问数据库和 objectstorage 的组件，将副本数设置为 0

b. **对于 Operator 版本 3.6 及更早版本：**首先缩减 Red Hat Quay registry，然后缩减托管的 Red Hat Quay 资源，缩减 Red Hat Quay 部署：

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-operator-namespace>|awk '/^quay-operator/ {print $1}') -n <quay-operator-namespace>
```

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-namespace>|awk '/quay-app/ {print $1}') -n <quay-namespace>
```

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-namespace>|awk '/quay-mirror/ {print $1}') -n <quay-namespace>
```

```
$ oc scale --replicas=0 deployment $(oc get deployment -n <quay-namespace>|awk '/clair-app/ {print $1}') -n <quay-namespace>
```

2. 等待 **registry-quay-app**、**registry-quay-mirror** 和 **registry-clair-app** pod （取决于您设置由 Red Hat Quay Operator 管理的组件）消失。您可以运行以下命令来检查其状态：

```
$ oc get pods -n <quay-namespace>
```

输出示例：

```
registry-quay-config-editor-77847fc4f5-nsbbv   1/1     Running     0          9m1s
registry-quay-database-66969cd859-n2ssm        1/1     Running     0          6d1h
registry-quay-redis-7cc5f6c977-956g8           1/1     Running     0          5d21h
```

7.3.3. 恢复 Red Hat Quay 数据库

使用以下步骤恢复您的 Red Hat Quay 数据库。

流程

1. 输入以下命令识别您的 **Quay** 数据库 pod：

```
$ oc get pod -l quay-component=postgres -n  <quay-namespace> -o jsonpath='{.items[0].metadata.name}'
```

输出示例：

```
quayregistry-quay-database-59f54bb7-58xs7
```

2. 通过从本地环境复制到 pod 来上传备份：

```
$ oc cp ./backup.sql -n <quay-namespace> registry-quay-database-66969cd859-n2ssm:/tmp/backup.sql
```

3. 输入以下命令在数据库中打开远程终端：

```
$ oc rsh -n <quay-namespace> registry-quay-database-66969cd859-n2ssm
```

4. 运行以下命令来输入 psql ：

```
bash-4.4$ psql
```

5. 您可以运行以下命令来列出数据库：

```
postgres=# \l
```

输出示例

```
                                  List of databases
         Name          |          Owner           | Encoding |  Collate   |   Ctype    |   Access privileges
---------------------------+--------------------------+----------+------------+------------+-----------------------
postgres                   | postgres                 | UTF8     | en_US.utf8 | en_US.utf8 |
quayregistry-quay-database | quayregistry-quay-database | UTF8     | en_US.utf8 | en_US.utf8 |
```

6. 输入以下命令丢弃数据库：

```
postgres=# DROP DATABASE "quayregistry-quay-database";
```

输出示例

```
DROP DATABASE
```

7. 退出 postgres CLI 以重新输入 bash-4.4：

```
\q
```

8. 将 PostgreSQL 数据库重定向到备份数据库：

```
sh-4.4$ psql < /tmp/backup.sql
```

9. 输入以下命令退出 bash：

```
sh-4.4$ exit
```

7.3.4. 恢复 Red Hat Quay 对象存储数据

使用以下步骤恢复 Red Hat Quay 对象存储数据。

流程

1. 输入以下命令导出 **AWS_ACCESS_KEY_ID** ：

```
$ export AWS_ACCESS_KEY_ID=$(oc get secret -l app=noobaa -n <quay-namespace>  -o jsonpath='{.items[0].data.AWS_ACCESS_KEY_ID}' |base64 -d)
```

2. 输入以下命令导出 **AWS_SECRET_ACCESS_KEY** ：

```
$ export AWS_SECRET_ACCESS_KEY=$(oc get secret -l app=noobaa -n <quay-namespace> -o jsonpath='{.items[0].data.AWS_SECRET_ACCESS_KEY}' |base64 -d)
```

3. 运行以下命令，将所有 Blob 上传到存储桶：

```
$ aws s3 sync --no-verify-ssl --endpoint https://$(oc get route s3 -n openshift-storage  -o jsonpath='{.spec.host}') ./blobs  s3://$(oc get cm -l app=noobaa -n <quay-namespace> -o jsonpath='{.items[0].data.BUCKET_NAME}')
```

注意

您还可以使用 rclone 或 sc3md，而不是 AWS 命令行工具。

7.3.5. 扩展 Red Hat Quay 部署

1. 根据 Red Hat Quay 部署的版本，使用以下选项之一扩展部署。

   a. **对于 Operator 版本 3.7 及更新版本：** 通过重新启用自动扩展（如果需要），并删除 Quay、镜像 worker 和 Clair 的副本覆盖来扩展 Red Hat Quay 部署。您的 **QuayRegistry** 资源应类似于如下：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: registry
  namespace: ns
spec:
  components:
    …
    - kind: horizontalpodautoscaler
      managed: true (1)
    - kind: quay (2)
      managed: true
    - kind: clair
      managed: true
    - kind: mirror
      managed: true
    …
```

1. 再次重新启用 Red Hat Quay 的自动扩展、Clair 和镜像 worker （如果需要）
2. 副本覆盖会再次删除以扩展 Red Hat Quay 组件

b. **对于 Operator 版本 3.6 及更早版本：**通过再次扩展 Red Hat Quay registry 来扩展 Red Hat Quay 部署：

```
$ oc scale --replicas=1 deployment $(oc get deployment -n <quay-operator-namespace>
| awk '/^quay-operator/ {print $1}') -n <quay-operator-namespace>
```

2. 检查 Red Hat Quay 部署的状态：

```
$ oc wait quayregistry registry --for=condition=Available=true -n <quay-namespace>
```

输出示例：

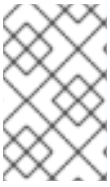
```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  ...
  name: registry
  namespace: <quay-namespace>
  ...
spec:
  ...
status:
  - lastTransitionTime: '2022-06-20T05:31:17Z'
    lastUpdateTime: '2022-06-20T17:31:13Z'
    message: All components reporting as healthy
    reason: HealthChecksPassing
    status: 'True'
    type: Available
```

第 8 章 卷大小覆盖

您可以指定为受管组件置备的存储资源所需的大小。Clair 和 PostgreSQL 数据库的默认大小为 **50Gi**。现在，您可以预先选择足够大的容量，无论是出于性能的原因，或者在您的存储后端没有调整功能的情况下。

在以下示例中，Clair 和 Quay PostgreSQL 数据库的卷大小被设置为 **70Gi** ：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: quay-example
  namespace: quay-enterprise
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: clair
      managed: true
      overrides:
        volumeSize: 70Gi
    - kind: postgres
      managed: true
      overrides:
        volumeSize: 70Gi
    - kind: clairpostgres
      managed: true
```

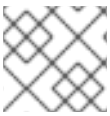


注意

clairpostgres 组件的卷大小无法被覆盖。要覆盖 **clairpostgres** 组件，您必须覆盖 **clair** 组件。这是一个已知问题，并将在以后的 Red Hat Quay 版本中解决。(PROJQUAY-4301)

第 9 章 使用 CONTAINER SECURITY OPERATOR 扫描 POD 镜像

Container Security Operator (CSO)是 OpenShift Container Platform 和其他 Kubernetes 平台上可用的 Clair 安全扫描程序的附加组件。使用 CSO 时，用户可以扫描与活跃 pod 关联的容器镜像以了解已知的漏洞。



注意

没有 Red Hat Quay 和 Clair，CSO 无法正常工作。

Container Security Operator (CSO)包括以下功能：

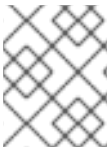
- 监视与指定或所有命名空间上的 pod 关联的容器。
- 查询容器来自漏洞信息的容器注册表，只要镜像的 registry 支持镜像扫描，如带有 Clair 扫描的 Red Hat Quay registry。
- 通过 Kubernetes API 中的 **ImageManifestVuln** 对象公开漏洞。

注意

要查看在 Kubernetes 上安装 CSO 的说明，请从 Container Security OperatorHub.io 页面中选择 **Install** 按钮。

9.1. 在 OPENSHIFT CONTAINER PLATFORM 中下载并运行 CONTAINER SECURITY OPERATOR

使用以下步骤下载 Container Security Operator (CSO)。



注意

在以下步骤中，CSO 安装在 **marketplace-operators** 命名空间中。这允许在 OpenShift Container Platform 集群的所有命名空间中使用 CSO。

流程

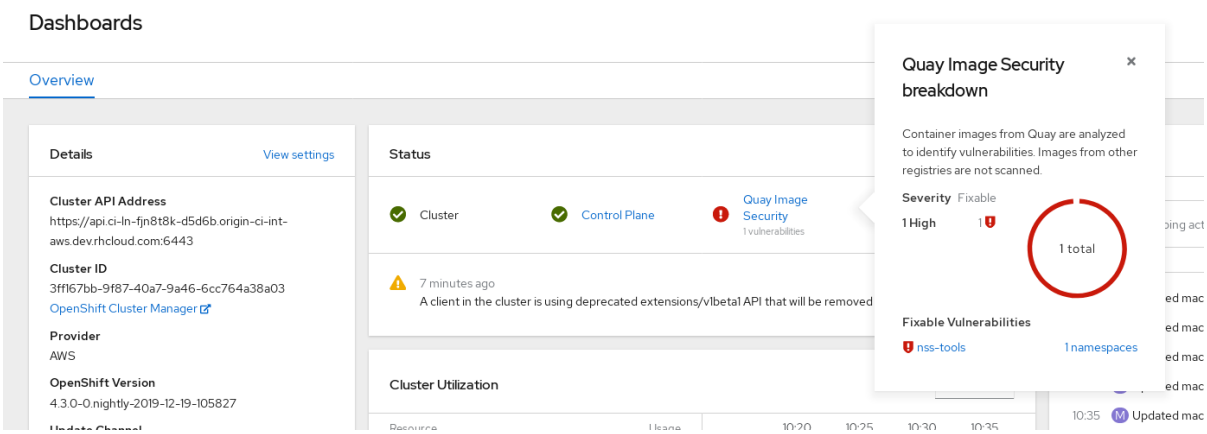
1. 在 OpenShift Container Platform 控制台页面中，选择 **Operators → OperatorHub** 并搜索 **Container Security Operator**。
2. 选择 Container Security Operator，然后选择 **Install** 进入 **Create Operator Subscription** 页面。
3. 检查设置（默认为所有命名空间和自动批准策略），然后选择 **Subscribe**。在 **Installed Operators** 屏幕中几分钟后会出现 **容器安全性**。
4. 可选：您可以将自定义证书添加到 CSO 中。在本例中，在当前目录中创建一个名为 **quay.crt** 的证书。然后，运行以下命令将证书添加到 CSO 中：

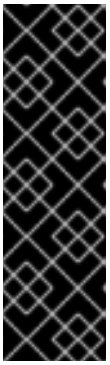
```
$ oc create secret generic container-security-operator-extra-certs --from-file=quay.crt -n
openshift-operators
```

注意

您必须重启 Operator pod 才能使新证书生效。

5. 导航到 **Home → Dashboards**。镜像 **安全性** 的链接显示在 status 部分，其中列出了目前发现的漏洞数量。选择链接以查看安全分类，如下图所示：



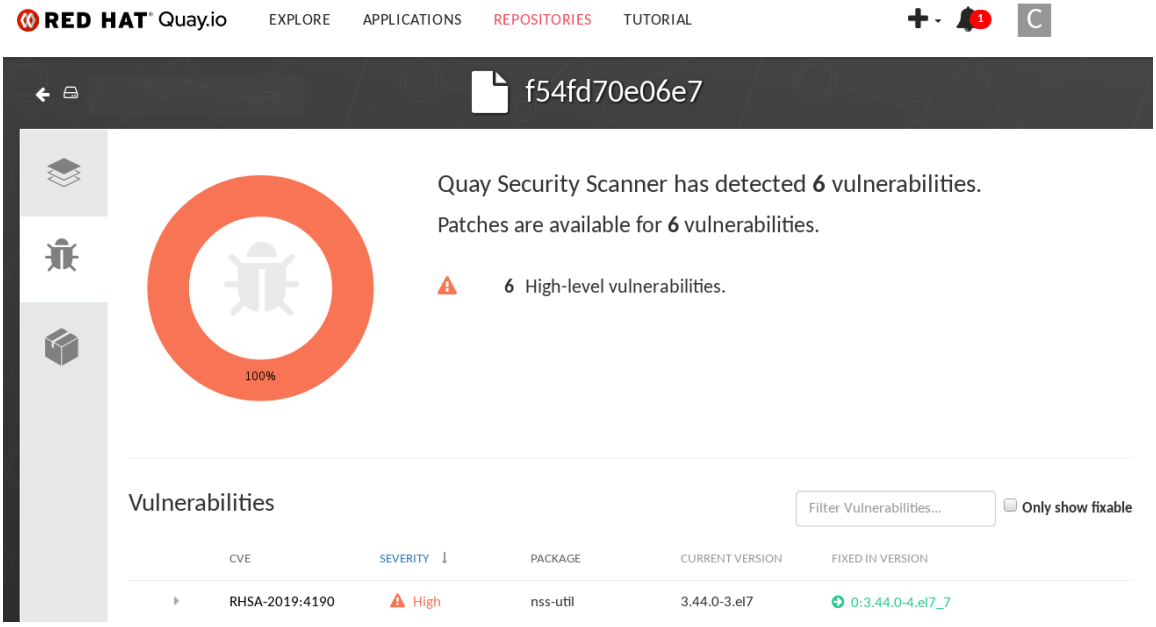


重要

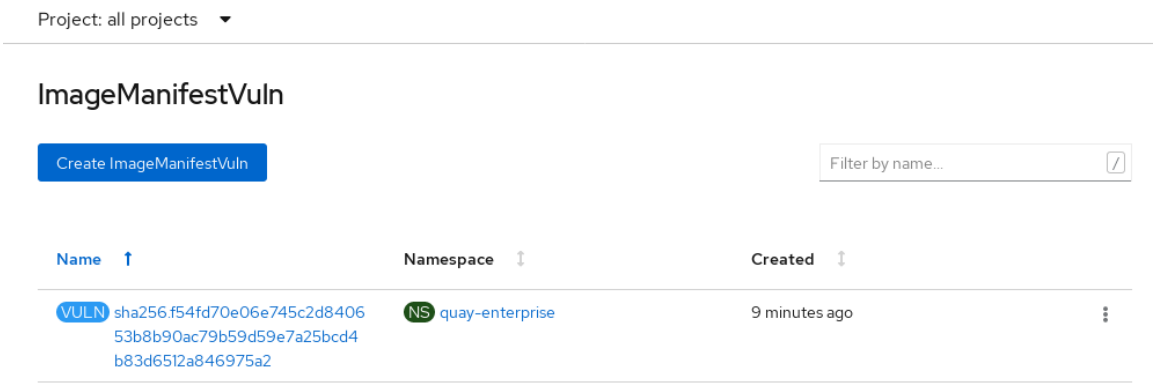
Container Security Operator 目前为 Red Hat Security 公告提供有问题的链接。例如，可能会提供以下链接： **https://access.redhat.com/errata/RHSA-2023:1842%20https://access.redhat.com/security/cve/CVE-2023-23916**。URL 中的 **%20** 代表空格字符，但当前会将两个 URL 组合成一个不完整的 URL，例如 **https://access.redhat.com/errata/RHSA-2023:1842** 和 **https://access.redhat.com/security/cve/CVE-2023-23916**。作为临时解决方案，您可以将每个 URL 复制到浏览器中，以导航到正确的页面。这是一个已知问题，并将在以后的 Red Hat Quay 版本中解决。

6. 对于任何检测到的安全漏洞，您可以在此时进行两个操作之一：

   a. 选择到这个漏洞的链接。您使用容器 registry、Red Hat Quay 或其他容器来自的 registry，您可以在其中查看漏洞的信息。下图显示了从 Quay.io registry 中检测到的漏洞示例：



b. 选择 namespaces 链接以进入 **ImageManifestVuln** 界面，您可以在其中查看所选镜像的名称以及该镜像正在运行的所有命名空间。下图表示，特定存在安全漏洞的镜像在两个命名空间中运行：



执行此流程后，您可以了解哪些镜像存在安全漏洞，您必须执行什么操作来修复这些漏洞，以及镜像中运行的每个命名空间。知道这一点，您可以执行以下操作：

- 提醒运行镜像的用户，用户需要更正此漏洞。
- 通过删除启动镜像所在 pod 的部署或对象来停止镜像运行。

注意

如果删除 pod，可能需要几分钟时间才能在仪表板上重置漏洞。

9.2. 通过 CLI 查询镜像漏洞

使用以下步骤从命令行界面(CLI)查询镜像漏洞。

流程

1. 输入以下命令查询检测到的漏洞：

```
$ oc get vuln --all-namespaces
```

输出示例

```
NAMESPACE     NAME            AGE
default       sha256.ca90...  6m56s
skynet        sha256.ca90...  9m37s
```

2. 可选。要显示特定漏洞的详情，识别特定的漏洞及其命名空间，并使用 **oc describe** 命令。以下示例显示了一个活跃的容器，其镜像包含带有漏洞的 RPM 软件包：

```
$ oc describe vuln --namespace mynamespace sha256.ac50e3752...
```

输出示例

```
Name:         sha256.ac50e3752...
Namespace:    quay-enterprise
...
```

```
Spec:
  Features:
    Name:            nss-util
    Namespace Name:  centos:7
    Version:         3.44.0-3.el7
    Versionformat:   rpm
    Vulnerabilities:
      Description: Network Security Services (NSS) is a set of libraries...
```

第 10 章 为 RED HAT QUAY 配置 AWS STS

对 Amazon Web Services (AWS)安全令牌服务(STS)的支持适用于独立的 Red Hat Quay 部署，在 OpenShift Container Platform 上支持 Red Hat Quay。AWS STS 是一个 web 服务，用于请求 AWS Identity and Access Management (IAM)用户的临时、具有有限权限的凭证，以及用于您验证或 *联邦用户的用户*。此功能对于将 Amazon S3 用作对象存储的集群很有用，允许 Red Hat Quay 使用 STS 协议与 Amazon S3 进行身份验证，这可以提高集群的整体安全性，并帮助确保正确验证并授权对敏感数据的访问。

配置 AWS STS 是一个多步骤，需要创建 AWS IAM 用户、创建 S3 角色并配置 Red Hat Quay **config.yaml** 文件使其包含正确的资源。

使用以下步骤为 Red Hat Quay 配置 AWS STS。

10.1. 创建 IAM 用户

使用以下步骤创建 IAM 用户。

流程

1. 登录到 Amazon Web Services (AWS)控制台，再进入到 Identity and Access Management (IAM)控制台。
2. 在导航窗格中，在 **Access management** 下点 **Users**。
3. 点 **Create User** 并输入以下信息：
   a. 输入有效的用户名，如 **quay-user**。
   b. 对于 **权限选项**，请单击 **Add user to group**。
4. 在 **review and create** 页面上，单击 **Create user**。您将被重定向到 **Users** 页面。
5. 单击用户名，如 **quay-user**。
6. 复制用户的 ARN，例如 **arn:aws:iam::123492922789:user/quay-user**。
7. 在同一页面上，单击 **Security credentials** 选项卡。
8. 导航到 **Access keys**。
9. 点 **Create access key**。
10. 在 **Access key 最佳实践和 alternatives** 页面中，点 **Command Line Interface (CLI)**，然后选中确认框。然后单击**"下一步"**。
11. 可选。在 **Set description tag - 可选** 页面中，输入描述。
12. 点 **Create access key**。
13. 复制并存储 access key 和 secret access key。

重要

这是查看或下载 secret 访问密钥的唯一时间。您不能稍后恢复。但是，您可以随时创建新访问密钥。

14. 点 **Done**。

10.2. 创建 S3 角色

使用以下步骤为 AWS STS 创建 S3 角色。

先决条件

- 您已创建了 IAM 用户，并存储访问密钥和 secret 访问密钥。

流程

1. 如果还没有，点 **Dashboard** 进入 IAM 仪表板。
2. 在导航窗格中，单击 **Access management** 下的 **Roles**。
3. 单击 **Create role**。
   - 点 **Custom Trust Policy**，它显示了一个可编辑的 JSON 策略。默认情况下，它显示以下信息：

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Statement1",
   "Effect": "Allow",
   "Principal": {},
   "Action": "sts:AssumeRole"
  }
 ]
}
```

4. 在 **Principal** configuration 字段中，添加 AWS ARN 信息。例如：

```
{
   "Version": "2012-10-17",
   "Statement": [
    {
     "Sid": "Statement1",
     "Effect": "Allow",
     "Principal": {
      "AWS": "arn:aws:iam::123492922789:user/quay-user"
     },
     "Action": "sts:AssumeRole"
    }
   ]
}
```

5. 点击 **Next**。
6. 在 **Add permissions** 页面中，在搜索框中输入 **AmazonS3FullAccess**。选中该复选框，将该策略添加到 S3 角色，然后单击 **Next**。
7. 在 **Name, review, and create** 页面中，输入以下信息：

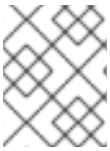
a. 输入角色名称，如 **example-role**。

b. 可选。添加描述。

8. 点 **Create role** 按钮。您将进入 **Roles** 页面。在 **Role name** 下，新创建的 S3 应可用。

10.3. 在 OPENSHIFT CONTAINER PLATFORM 上配置 RED HAT QUAY 以使用 AWS STS

使用以下步骤编辑 OpenShift Container Platform **config.yaml** 文件中的 Red Hat Quay 以使用 AWS STS。



注意

您还可以直接在 OpenShift Container Platform **config.yaml** 文件中编辑和重新部署 Red Hat Quay，而不是使用 OpenShift Container Platform UI。

先决条件

- 您已配置了角色 ARN。
- 您已生成了一个 User Access Key。
- 您已生成了一个 User Secret 密钥。

流程

1. 在 OpenShift Container Platform 部署的 **Home** 页面中，点 **Operators → Installed Operators**。
2. 点 **Red Hat Quay**。
3. 单击 **Quay Registry**，然后单击 Red Hat Quay registry 的名称。
4. 在 **Config Bundle Secret** 下，点 registry 配置捆绑包的名称，如 **quay-registry-config-bundle-qet56**。
5. 在配置捆绑包页面中，单击 **Actions** 以显示下拉菜单。然后点 **Edit Secret**。
6. 使用以下信息更新 **config.yaml** 文件的 **DISTRIBUTED_STORAGE_CONFIG** 字段：

```
# ...
DISTRIBUTED_STORAGE_CONFIG:
  default:
   - STSS3Storage
   - sts_role_arn: <role_arn> 1
     s3_bucket: <s3_bucket_name> 2
     storage_path: <storage_path> 3
     sts_user_access_key: <s3_user_access_key> 4
     sts_user_secret_key: <s3_user_secret_key> 5
# ...
```

1 配置 AWS STS 时需要的唯一 Amazon 资源名称(ARN)

2. s3 存储桶的名称。
3. 数据的存储路径。通常 **/datastorage**。
4. 配置 AWS STS 时生成的 AWS S3 用户访问密钥。
5. 配置 AWS STS 时生成的 AWS S3 用户 secret 密钥。

7. 点击 **Save**。

验证

1. 标记示例镜像，如 **busybox**，它将推送到存储库。例如：

```
$ podman tag docker.io/library/busybox <quay-
server.example.com>/<organization_name>/busybox:test
```

2. 运行以下命令来推送示例镜像：

```
$ podman push <quay-server.example.com>/<organization_name>/busybox:test
```

3. 导航到您在 Red Hat Quay registry → **Tags** 中将镜像推送到的 Organization，以验证推送是否成功。
4. 进入到 Amazon Web Services (AWS)控制台，找到您的 s3 存储桶。
5. 点 s3 存储桶的名称。
6. 在 **Objects** 页面上，单击 **datastorage/**。
7. 在 **datastorage/** 页面中，应该看到以下资源：
   - **sha256/**
   - **上传/**
     这些资源表示推送成功，并且 AWS STS 已被正确配置。

第 11 章 将 RED HAT QUAY 与 OPENSHIFT CONTAINER PLATFORM 与 QUAY BRIDGE OPERATOR 集成

Quay Bridge Operator 将集成的 OpenShift Container Platform registry 的功能复制到新的 Red Hat Quay registry 中。使用 Quay Bridge Operator，您可以将 OpenShift Container Platform 中的集成容器 registry 替换为 Red Hat Quay registry。

Quay Bridge Operator 启用的功能包括：

- 将 OpenShift Container Platform 命名空间同步为 Red Hat Quay 机构。
- 为每个 default 命名空间服务帐户创建机器人帐户。
- 为每个创建的机器人帐户创建 secret，并将每个机器人 secret 与一个服务账户关联作为 **Mountable** 和 **Image Pull Secret**。
- 将 OpenShift Container Platform 镜像流同步为 Red Hat Quay 存储库。
- 自动重写使用镜像流的新构建来输出到 Red Hat Quay。
- 构建完成后自动导入镜像流标签。

使用以下步骤，您可以启用 Red Hat Quay 和 OpenShift Container Platform 集群之间的双向通信。

11.1. 为 QUAY BRIDGE OPERATOR 设置 RED HAT QUAY BRIDGE

在此过程中，您将创建一个专用的 Red Hat Quay 机构，并从该机构中创建的新应用程序中生成 OAuth 令牌，以用于 OpenShift Container Platform 中的 Quay Bridge Operator。

流程

1. 通过 Web UI 登录 Red Hat Quay。
2. 选择要配置外部应用程序的组织。
3. 在导航窗格中，选择 **Applications**。
4. 选择 **Create New Application**，再输入新应用的名称，如 **openshift**。
5. 在 **OAuth Applications** 页面中，选择您的应用程序，如 **openshift**。
6. 在导航窗格中，选择 **Generate Token**。
7. 选择以下字段：
   - **管理机构**
   - **管理软件仓库**
   - **创建软件仓库**
   - **查看所有可见的存储库**
   - **对任何可访问的软件仓库的读/写**
   - **管理用户**

- **读取用户信息**

8. 检查分配的权限。
9. 选择 **Authorize Application**，然后选择 **Authorize Application** 来确认授权。
10. 保存生成的访问令牌。

重要

Red Hat Quay 不提供令牌管理。您无法列出令牌、删除令牌或修改令牌。生成的访问令牌仅显示一次，在关闭页面后无法重新获取。

11.2. 在 OPENSHIFT CONTAINER PLATFORM 上安装 QUAY BRIDGE OPERATOR

在此过程中，您将在 OpenShift Container Platform 上安装 Quay Bridge Operator。

先决条件

- 您已设置 Red Hat Quay 并获取了访问令牌。
- 具有集群管理员权限的 OpenShift Container Platform 4.6 或更高版本环境。

流程

1. 打开 Web 控制台的 **Administrator** 视角，并导航到导航窗格中的 **Operators → OperatorHub**。
2. 搜索 **Quay Bridge Operator**，单击 **Quay Bridge Operator** 标题，然后单击 **Install**。
3. 选择要安装的版本，如 **stable-3.7**，然后单击 **Install**。
4. 安装完成后，点 **View Operator** 进入 Quay Bridge Operator 的 **Details** 页面。另外，您可以点 **Installed Operators → Red Hat Quay Bridge Operator** 进入 **Details** 页面。

11.3. 为 OAUTH 令牌创建 OPENSHIFT CONTAINER PLATFORM SECRET

在此过程中，您将添加之前获取的访问令牌，以便与 Red Hat Quay 部署进行通信。访问令牌将存储在 OpenShift Container Platform 中，作为 secret。

先决条件

- 您已设置 Red Hat Quay 并获取了访问令牌。
- 您已在 OpenShift Container Platform 上部署了 Quay Bridge Operator。
- 具有集群管理员权限的 OpenShift Container Platform 4.6 或更高版本环境。
- 已安装 OpenShift CLI（oc）。

流程

- 创建一个包含 **openshift-operators** 命名空间中的访问令牌的 secret：

```
$ oc create secret -n openshift-operators generic <secret-name> --from-literal=token=
<access_token>
```

11.4. 创建 QUAYINTEGRATION 自定义资源

在此过程中，您将创建一个 **QuayIntegration** 自定义资源，该资源可以从 Web 控制台或命令行完成。

先决条件

- 您已设置 Red Hat Quay 并获取了访问令牌。
- 您已在 OpenShift Container Platform 上部署了 Quay Bridge Operator。
- 具有集群管理员权限的 OpenShift Container Platform 4.6 或更高版本环境。
- 可选： 已安装 OpenShift CLI (oc)。

11.4.1. 可选：使用 CLI 创建 QuayIntegration 自定义资源

按照以下步骤使用命令行创建 **QuayIntegration** 自定义资源。

流程

1. 创建 **quay-integration.yaml** ：

```
$ touch quay-integration.yaml
```

2. 对 **QuayIntegration** 自定义资源的最小部署使用以下配置：

```
apiVersion: quay.redhat.com/v1
kind: QuayIntegration
metadata:
  name: example-quayintegration
spec:
  clusterID: openshift 1
  credentialsSecret:
    namespace: openshift-operators
    name: quay-integration 2
  quayHostname: https://<QUAY_URL> 3
  insecureRegistry: false 4
```

1. clusterID 值在整个生态系统中应是唯一的。这个值是必需的，默认为 **openshift**。
2. **credentialsSecret** 属性引用包含之前创建的令牌的 secret 的命名空间和名称。
3. 将 **QUAY_URL** 替换为 Red Hat Quay 实例的主机名。
4. 如果 Red Hat Quay 使用自签名证书，请将属性设置为 **insecureRegistry: true**。

有关所有配置字段的列表，请参阅"QuayIntegration 配置字段"。

1. 创建 **QuayIntegration** 自定义资源：

```
$ oc create -f quay-integration.yaml
```

11.4.2. 可选：使用 Web 控制台创建 QuayIntegration 自定义资源

按照以下步骤，使用 Web 控制台创建 **QuayIntegration** 自定义资源。

流程

1. 打开 Web 控制台的 **Administrator** 视角，并导航到 **Operators → Installed Operators**。
2. 点 **Red Hat Quay Bridge Operator**。
3. 在 Quay Bridge Operator 的 **Details** 页面中，点 **Quay Integration** API 卡上的 **Create Instance**。
4. 在 **Create QuayIntegration** 页面中，在 **Form view** 或 **YAML view** 中输入以下所需信息：
   - **名称** ：引用 **QuayIntegration** 自定义资源对象的名称。
   - **集群 ID**：与此集群关联的 ID。这个值应该在整个生态系统中唯一。如果未指定，则默认为 **openshift**。
   - **凭证 secret** ：请参阅包含之前创建的令牌的 secret 的命名空间和名称。
   - **Quay 主机名** ：Quay registry 的主机名。

有关所有配置字段的列表，请参阅"QuayIntegration 配置字段"。

创建 **QuayIntegration** 自定义资源后，您的 OpenShift Container Platform 集群将链接到您的 Red Hat Quay 实例。应该为 OpenShift Container Platform 环境的相关命名空间创建 Red Hat Quay registry 中的机构。

11.5. 使用 QUAY BRIDGE OPERATOR

使用以下步骤使用 Quay Bridge Operator。

先决条件

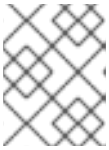
- 已安装 Red Hat Quay Operator。
- 已作为集群管理员登录到 OpenShift Container Platform。
- 已登陆到 Red Hat Quay registry。
- 已安装 Quay Bridge Operator。
- 您已配置了 **QuayIntegration** 自定义资源。

流程

1. 输入以下命令创建一个名为 **e2e-demo** 的新 OpenShift Container Platform 项目：

```
$ oc new-project e2e-demo
```

2. 在创建了新项目后，Red Hat Quay 中会创建一个新机构。导航到 Red Hat Quay registry，并确认您已创建一个名为 **openshift_e2e-demo** 的新组织。



注意

如果 **QuayIntegration** 资源中的 clusterID 使用不同的值，则机构的 **openshift** 值可能会有所不同。

3. 在 Red Hat Quay UI 上，单击新组织的名称，如 **openshift_e2e-demo**。

4. 在导航窗格中，单击 **Robot Accounts**。作为新项目的一部分，应创建以下 Robot Accounts：

   - **openshift_e2e-demo+deployer**
   - **openshift_e2e-demo+default**
   - **openshift_e2e-demo+builder**

5. 输入以下命令确认已创建了三个包含与适用 Robot Accounts 关联的 Docker 配置的 secret：

```
$ oc get secrets builder-quay-openshift deployer-quay-openshift default-quay-openshift
```

输出示例

```
stevsmit@stevsmit ocp-quay $ oc get secrets builder-quay-openshift deployer-quay-openshift
default-quay-openshift
NAME                    TYPE                             DATA   AGE
builder-quay-openshift    kubernetes.io/dockerconfigjson   1      77m
deployer-quay-openshift   kubernetes.io/dockerconfigjson   1      77m
default-quay-openshift    kubernetes.io/dockerconfigjson   1      77m
```

6. 输入以下命令显示 **builder** ServiceAccount (SA)的详细信息，包括其 secret、令牌过期以及关联的角色和角色绑定。这样可确保项目通过 Quay Bridge Operator 集成。

```
$ oc describe sa builder default deployer
```

输出示例

```
...
Name:                builder
Namespace:           e2e-demo
Labels:              <none>
Annotations:         <none>
Image pull secrets:  builder-dockercfg-12345
                     builder-quay-openshift
Mountable secrets:   builder-dockercfg-12345
                     builder-quay-openshift
Tokens:              builder-token-12345
Events:              <none>
...
```

7. 输入以下命令来创建并部署名为 **httpd-template** 的新应用程序：

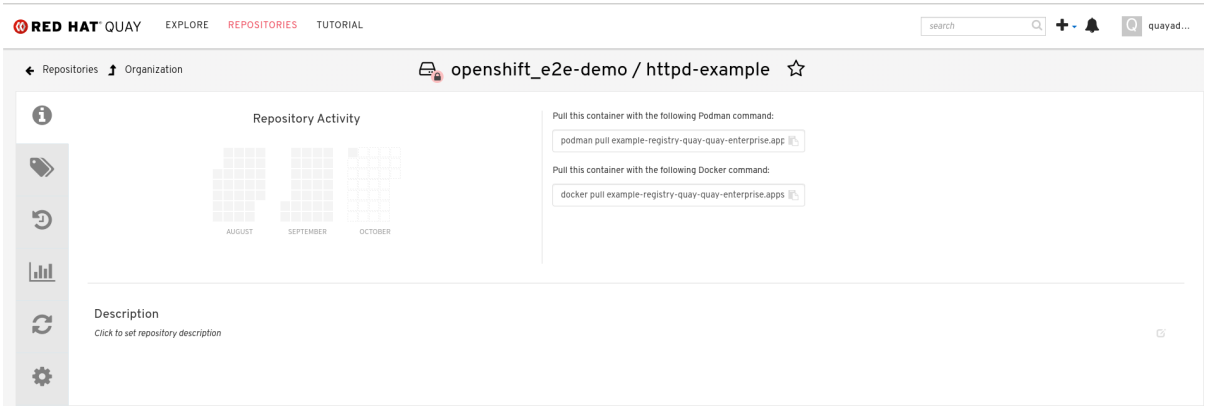
```
$ oc new-app --template=httpd-example
```

-

输出示例

```
--> Deploying template "e2e-demo/httpd-example" to project e2e-demo
...
--> Creating resources ...
    service "httpd-example" created
    route.route.openshift.io "httpd-example" created
    imagestream.image.openshift.io "httpd-example" created
    buildconfig.build.openshift.io "httpd-example" created
    deploymentconfig.apps.openshift.io "httpd-example" created
--> Success
    Access your application via route 'httpd-example-e2e-demo.apps.quay-ocp.gcp.quaydev.org'
    Build scheduled, use 'oc logs -f buildconfig/httpd-example' to track its progress.
    Run 'oc status' to view your app.
```

运行此命令后，会创建 **BuildConfig**、**ImageStream** 、**Service**、**Route** 和 **DeploymentConfig** 资源。创建 **ImageStream** 资源时，会在 Red Hat Quay 中创建关联的存储库。例如：



8. 当 Apache HTTPD 镜像位于 **openshift** 命名空间中的 Apache HTTPD 镜像被解决时，**BuildConfig** 的 **ImageChangeTrigger** 会触发新的 Build。创建新 Build 时，**MutatingWebhookConfiguration** 会自动重写输出以指向 Red Hat Quay。您可以通过运行以下命令来查询构建的输出字段来确认构建已完成：

```
$ oc get build httpd-example-1 --template='{{ .spec.output.to.name }}'
```

输出示例

```
example-registry-quay-quay-enterprise.apps.quay-ocp.gcp.quaydev.org/openshift_e2e-demo/httpd-example:latest
```

9. 在 Red Hat Quay UI 上，导航到 **openshift_e2e-demo** 组织，再选择 **httpd-example** 存储库。

10. 单击导航窗格中的 **Tags**，并确认 **latest** 标签已成功推送。

11. 输入以下命令来确保 latest 标签已解决：

```
$ oc describe is httpd-example
```

输出示例

```
Name:   httpd-example
Namespace:  e2e-demo
Created:  55 minutes ago
Labels:   app=httpd-example
  template=httpd-example
Description:  Keeps track of changes in the application image
Annotations:  openshift.io/generated-by=OpenShiftNewApp
  openshift.io/image.dockerRepositoryCheck=2023-10-02T17:56:45Z
Image Repository: image-registry.openshift-image-registry.svc:5000/e2e-demo/httpd-
example
Image Lookup:  local=false
Unique Images:  0
Tags:   1

latest
  tagged from example-registry-quay-quay-enterprise.apps.quay-
ocp.gcp.quaydev.org/openshift_e2e-demo/httpd-example:latest
```

12. 在 **ImageStream** 被 resolwillved 后，应该会触发新部署。输入以下命令生成 URL 输出：

```
$ oc get route httpd-example --template='{{ .spec.host }}'
```

输出示例

```
httpd-example-e2e-demo.apps.quay-ocp.gcp.quaydev.org
```

13. 访问此 URL。如果显示示例网页，则代表部署成功。

14. 输入以下命令删除资源并清理 Red Hat Quay 存储库：

```
$ oc delete project e2e-demo
```

注意

命令将等待到项目资源已被删除。这可以通过将 **--wait=false** 添加到上述命令来绕过

15. 命令完成后，导航到您的 Red Hat Quay 存储库，并确认 **openshift_e2e-demo** 组织不再可用。

其他资源

- 最佳实践规定客户端和镜像 registry 之间的所有通信都通过安全的方法促进。通信应该与各方之间的证书信任利用 HTTPS/TLS。虽然 Red Hat Quay 可以配置为提供不安全的配置，但应该在服务器上使用正确的证书并在客户端上配置。按照 OpenShift Container Platform 文档 在容器运行时级别添加和管理证书。

第 12 章 在 OPENSHIFT CONTAINER PLATFORM 上的 RED HAT QUAY 上部署 IPV6

注意

目前，IBM Power 和 IBM Z 不支持在 OpenShift Container Platform 上部署 IPv6。

OpenShift Container Platform 部署的 Red Hat Quay 现在可在仅支持 IPv6 的位置提供，如 Telco 和 Edge 环境。

有关已知限制列表，请参阅 IPv6 限制

12.1. 启用 IPV6 协议系列

使用以下步骤在 Red Hat Quay 部署中启用 IPv6 支持。

先决条件

- 您已将 Red Hat Quay 更新至 3.8。
- 您的主机和容器软件平台(Docker、Podman)必须配置为支持 IPv6。

流程

1. 在部署的 **config.yaml** 文件中，添加 **FEATURE_LISTEN_IP_VERSION** 参数并将其设置为 **IPv6**，例如：

```
---
FEATURE_GOOGLE_LOGIN: false
FEATURE_INVITE_ONLY_USER_CREATION: false
FEATURE_LISTEN_IP_VERSION: IPv6
FEATURE_MAILING: false
FEATURE_NONSUPERUSER_TEAM_SYNCING_SETUP: false
---
```

2. 启动或重启您的 Red Hat Quay 部署。

3. 输入以下命令检查您的部署是否侦听 IPv6：

```
$ curl <quay_endpoint>/health/instance
{"data":{"services":
{"auth":true,"database":true,"disk_space":true,"registry_gunicorn":true,"service_key":true,"web_gunicorn":true}},"status_code":200}
```

在部署的 **config.yaml** 中启用 IPv6 后，所有 Red Hat Quay 功能都可以正常使用，只要您的环境被配置为使用 IPv6，且不会受 IPv6 和双栈限制的影响。

警告

如果您的环境被配置为 IPv4，但 **FEATURE_LISTEN_IP_VERSION** 配置字段被设置为 **IPv6**，则 Red Hat Quay 将无法部署。

12.2. IPV6 限制

- 目前，尝试使用通用 Microsoft Azure Blob 存储配置 Red Hat Quay 部署无法在 IPv6 单堆栈环境中工作。因为 Microsoft Azure Blob Storage 的端点不支持 IPv6，所以这个问题还没有临时解决方案。
  如需更多信息，请参阅 PROJQUAY-4433。

- 目前，尝试使用 Amazon S3 CloudFront 配置 Red Hat Quay 部署无法在 IPv6 单堆栈环境中工作。因为 Amazon S3 CloudFront 的端点不支持 IPv6，所以这个问题还没有临时解决方案。
  如需更多信息，请参阅 PROJQUAY-4470。

- 目前，当在 IPv6 单堆栈环境中部署 Red Hat Quay 时，Red Hat OpenShift Data Foundation 不被支持。因此，Red Hat OpenShift Data Foundation 无法在 IPv6 环境中使用。计划在以后的 OpenShift Data Foundation 版本中修复这个限制。

- 目前，双栈(IPv4 和 IPv6)支持不适用于 Red Hat Quay OpenShift Container Platform 部署。当 Red Hat Quay 3.8 部署在启用了双栈支持的 OpenShift Container Platform 上时，Red Hat Quay Operator 生成的 Quay Route 只会生成 IPv4 地址，而不是 IPv6 地址。因此，带有 IPv6 地址的客户端无法访问 OpenShift Container Platform 上的 Red Hat Quay 应用程序。计划在以后的 OpenShift Container Platform 版本中修复这个限制。

第 13 章 当在 KUBERNETES 上部署 RED HAT QUAY 时添加自定义 SSL/TLS 证书

在 Kubernetes 上部署时，Red Hat Quay 将作为卷挂载到 secret 中以存储配置资产。目前，这会破坏超级用户面板的上传证书功能。

作为临时解决方案，在部署 Red Hat Quay *后*，**base64** 编码证书可以添加到 secret 中。

在 Kubernetes 上部署 Red Hat Quay 时，请使用以下步骤添加自定义 SSL/TLS 证书。

先决条件

- Red Hat Quay 已部署。
- 您有一个自定义 **ca.crt** 文件。

流程

1. 输入以下命令对 SSL/TLS 证书的内容进行 Base64 编码：

```
$ cat ca.crt | base64 -w 0
```

输出示例

```
...c1psWGpqeGlPQmNEWkJPMjJ5d0pDemVnR2QNCnRsbW9JdEF4YnFSdVd3PT0KLS0tLS1
FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
```

2. 输入以下 **kubectl** 命令来编辑 **quay-enterprise-config-secret** 文件：

```
$ kubectl --namespace quay-enterprise edit secret/quay-enterprise-config-secret
```

3. 为证书添加一个条目，并将完整的 **base64** 编码字符串er 粘贴到该条目下。例如：

```
  custom-cert.crt:
c1psWGpqeGlPQmNEWkJPMjJ5d0pDemVnR2QNCnRsbW9JdEF4YnFSdVd3PT0KLS0tLS1F
TkQgQ0VSVElGSUNBVEUtLS0tLQo=
```

4. 使用 **kubectl delete** 命令删除所有 Red Hat Quay pod。例如：

```
$ kubectl delete pod quay-operator.v3.7.1-6f9d859bd-p5ftc quayregistry-clair-postgres-
7487f5bd86-xnxpr quayregistry-quay-app-upgrade-xq2v6  quayregistry-quay-database-
859d5445ff-cqthr quayregistry-quay-redis-84f888776f-hhgms
```

之后，Red Hat Quay 部署会自动将 pod 替换为新证书数据。

第 14 章 升级 RED HAT QUAY OPERATOR 概述

Red Hat Quay Operator 遵循一个 *同步的版本* 方案，这意味着每个 Operator 版本都绑定到 Red Hat Quay 的版本及其管理的组件。**QuayRegistry** 自定义资源中没有字段，它设置要部署的 Red Hat Quay 版本 ; Operator 只能部署所有组件的单一版本。选择这个方案来确保所有组件都可以正常工作，并降低 Operator 的复杂性，了解如何管理 Kubernetes 上不同版本的 Red Hat Quay 的生命周期。

14.1. OPERATOR LIFECYCLE MANAGER

Red Hat Quay Operator 应使用 Operator Lifecycle Manager (OLM) 来安装和升级。当使用默认 **approvalStrategy: Automatic** 创建订阅时，每当有新版本可用时，OLM 将自动升级 Red Hat Quay Operator。

警告

当 Operator Lifecycle Manager 安装 Red Hat Quay Operator 时，可能会将其配置为支持自动或手动升级。这个选项在安装过程中在 Red Hat Quay Operator 的 OperatorHub 页面中显示。它还可以通过 approvalStrategy 字段在 Red Hat Quay Operator Subscription 对象中找到。选择 Automatic 意味着，当发布新的 Operator 版本时，您的 Red Hat Quay Operator 将自动升级。如果这不必要，则应选择 Manual 批准策略。

14.2. 升级 RED HAT QUAY OPERATOR

在 OpenShift Container Platform 上升级已安装的 Operator 的标准方法包括在 升级安装的 Operator 中。

通常，Red Hat Quay 仅支持从以前的(N-1)次版本进行升级。例如，不支持直接从 Red Hat Quay 3.0.5 升级到 3.5 的最新版本。相反，用户必须升级，如下所示：

1. 3.0.5 → 3.1.3
2. 3.1.3 → 3.2.2
3. 3.2.2 → 3.3.4

4. **3.3.4 → 3.4.z**

5. **3.4.z → 3.5.z**

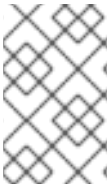
这需要在升级过程中确保正确完成任何必要的数据库迁移。

在某些情况下，Red Hat Quay 支持从之前(N-2、N-3)的直接、单步升级。这简化了旧版本客户的升级过程。Red Hat Quay 3 支持以下升级路径：

- **3.9.z → 3.11.z**
- **3.10.z → 3.11.z**

注意

不支持从 3.8.z 升级到 3.11。用户必须首先升级到 3.9 或 3.10，然后升级到 3.11。



注意

Red Hat Quay Operator 可以从 3.10.X for IBM Power 和 IBM Z 升级。

对于独立部署 Red Hat Quay 希望升级到 3.11 的用户，请参阅 独立 升级指南。

14.2.1. 升级 Red Hat Quay

要将 Red Hat Quay 从一个次版本升级到下一个次版本，如 3.10 → 3.11，您必须更改 Red Hat Quay Operator 的更新频道。

对于 z 流升级，如 3.10.1 → 3.10.2，更新会在用户最初在安装过程中选择的 major-minor 频道中发布。执行 z 流升级的步骤取决于上述的 approvalStrategy。如果批准策略被设置为 Automatic，Red Hat Quay Operator 会自动升级到最新的 z 流。这会导致自动的、滚动 Red Hat Quay 更新到较新的 z 流，而无需停机。否则，在开始安装前必须手动批准更新。

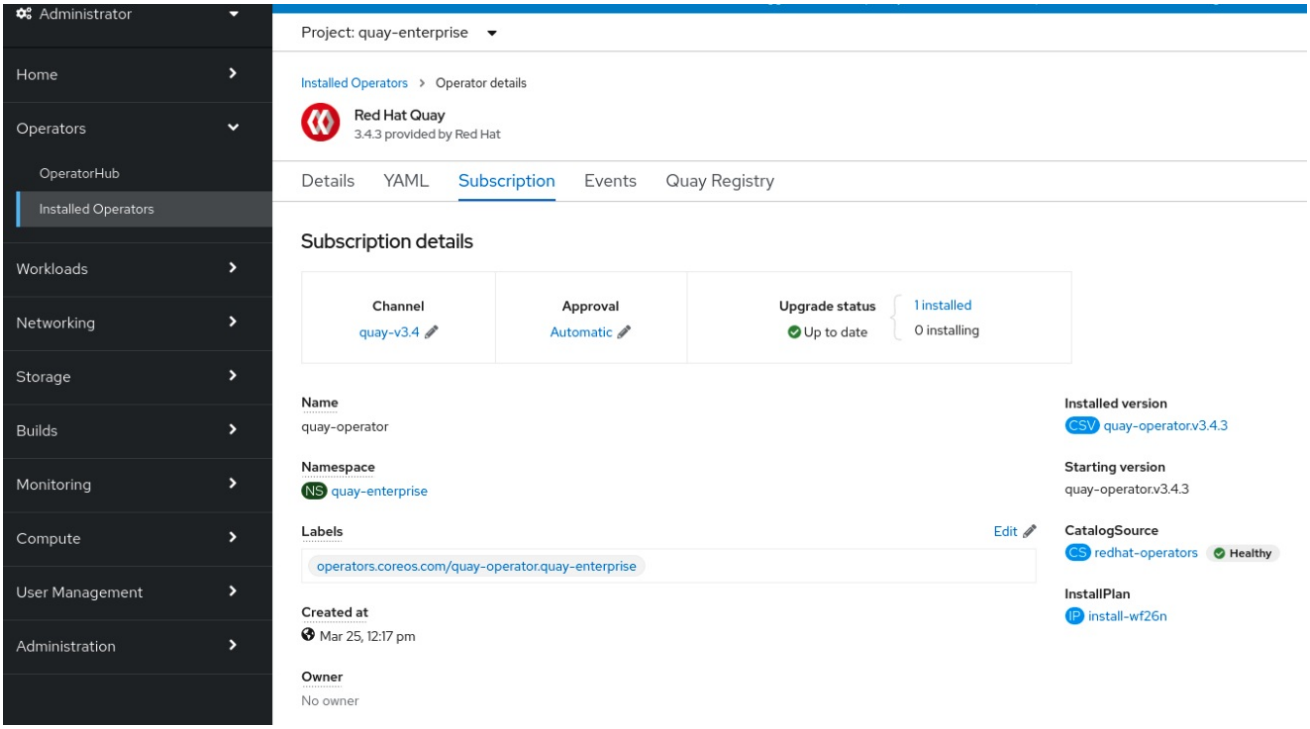
14.2.2. 更改 Red Hat Quay Operator 的更新频道

已安装的 Operator 的订阅指定一个更新频道，用于跟踪和接收 Operator 的更新。要升级 Red Hat Quay Operator 以开始跟踪并从更新频道接收更新，请更改安装的 Red Hat Quay Operator 的 Subscription 选项卡中的更新频道。对于带有 自动批准策略 的订阅，升级会自动开始，并可在列出 Installed Operators 的页面中监控。

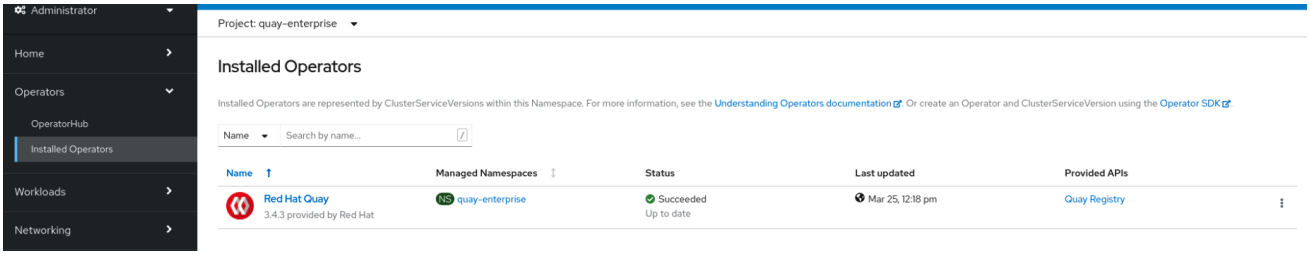
14.2.3. 手动批准待处理的 Operator 升级

如果安装的 Operator 的订阅设置为 Manual，则当其当前更新频道中发布新更新时，必须在开始安装前手动批准更新。如果 Red Hat Quay Operator 有一个待处理的升级，这个状态将显示在 Installed Operators 列表中。在 Red Hat Quay Operator 的 Subscription 选项卡中，您可以预览安装计划并查看列出可用于升级的资源。如果满意，点 Approve 并返回到列出 Installed Operators 以监控升级进度的页面。

下图显示了 UI 中的 Subscription 选项卡，包括 更新频道、Approval 策略、Upgrade 状态和 InstallPlan ：



Installed Operators 列表提供了当前 Quay 安装的高级别概述：



14.3. 升级 QUAYREGISTRY 资源

当 Red Hat Quay Operator 启动时，它会立即查找它被配置为监视的命名空间中找到的任何 QuayRegistries。当找到一个时，会使用以下逻辑：

- 如果未设置 status.currentVersion，请正常协调。
- 如果 status.currentVersion 等于 Operator 版本，请正常进行协调。
- 如果 status.currentVersion 不等于 Operator 版本，请检查是否可以升级它。如果可以，执行升级任务，并将 status.currentVersion 设置为 Operator 的版本。如果无法升级，则返回错误，并只保留 QuayRegistry 及其部署的 Kubernetes 对象。

14.4. 升级 QUAYECOSYSTEM

升级从以前的 Operator 版本支持，该版本将 QuayEcosystem API 用于一组有限的配置。为确保迁移不会意外发生，需要将一个特殊标签应用到 QuayEcosystem，以便它被迁移。将为 Operator 创建新的 QuayRegistry 以管理，但旧的 QuayEcosystem 将保留下来，直到手动删除为止，以确保在出现问题时可以回滚并仍然访问 Quay。要将现有的 QuayEcosystem 迁移到新的 QuayRegistry，请使用以下步骤。

流程

1. 将 "quay-operator/migrate": "true" 添加到 QuayEcosystem 的 metadata.labels 中。

```
$ oc edit quayecosystem <quayecosystemname>
```

```
metadata:
  labels:
    quay-operator/migrate: "true"
```

2.

等待 QuayRegistry 使用与 QuayEcosystem 相同的 metadata.name 创建。QuayEcosystem 将标记为标签 "quay-operator/migration-complete": "true "。

3. 设置了新的 QuayRegistry 的 status.registryEndpoint 后，访问 Red Hat Quay 并确认所有数据和设置都已成功迁移。

4. 如果一切都正常工作，您可以删除 QuayEcosystem 和 Kubernetes 垃圾回收会清理所有旧资源。

14.4.1. 恢复 QuayEcosystem 升级

如果在从 QuayEcosystem 升级到 QuayRegistry 过程中出现错误，请按照以下步骤恢复回使用 QuayEcosystem ：

流程

1. 使用 UI 或 kubectl 删除 QuayRegistry ：

```
$ kubectl delete -n <namespace> quayregistry <quayecosystem-name>
```

2. 如果使用 Route 提供外部访问，将 Route 更改为指回原始的 Service（使用 UI 或 kubectl）。

注意

如果您的 QuayEcosystem 管理 PostgreSQL 数据库，升级过程会将您的数据迁移到升级的 Operator 管理的新 PostgreSQL 数据库。旧数据库不会被更改或删除，但 Red Hat Quay 在迁移完成后将不再使用它。如果在数据迁移过程中出现问题，升级过程将退出，建议您继续将数据库作为非受管组件。

14.4.2. 升级支持的 QuayEcosystem 配置

如果迁移 QuayEcosystem 组件失败或不受支持，Red Hat Quay Operator 会在其日志中报告错误，并处于 status.conditions 中。所有非受管组件都应成功迁移，因为不需要采用 Kubernetes 资源，并且 Red Hat Quay 的 config.yaml 文件中已提供了所有必需的值。

数据库

不支持临时数据库（必须设置volumeSize 字段）。

Redis

不需要特别需要。

外部访问

对于自动迁移，只支持 passthrough Route 访问。其他方法需要手动迁移。

- 没有自定义主机名的 LoadBalancer ：当 QuayEcosystem 使用标签 "quay-operator/migration-complete": "true" 标记后，在删除 QuayEcosystem *前*需要从现存的 Service 中删除 metadata.ownerReferences 字段，这可以防止 Kubernetes 对 Service 进行垃圾回收并并删除负载均衡器。一个新的 Service 将被创建，带有 metadata.name 格式 <QuayEcosystem-name>-quay-app。编辑现有 Service 的 spec.selector，使其与新 Service 的 spec.selector 匹配，以便进入旧负载均衡器端点的流量现在会被定向到新的 pod。旧的 Service 将会被您管理; Quay Operator 将不再管理它。

- 带有自定义主机名的 LoadBalancer/NodePort/Ingress ：将创建一个类型为 LoadBalancer 的新的 Service，带有 metadata.name 格式 <QuayEcosystem-name>-quay-app。将您的 DNS 设置更改为指向由新 Service 提供的 status.loadBalancer 端点。

Clair

不需要特别需要。

对象存储

QuayEcosystem 没有受管对象存储组件，因此对象存储始终标记为 unmanaged。不支持本地存储。

仓库镜像

不需要特别需要。

其他资源

- 如需有关 Red Hat Quay Operator 的更多信息，请参阅上游 quay-operator 项目。