Red Hat Quay 3.11

管理 Red Hat Quay

管理 Red Hat Quay

Last Updated: 2024-06-06

Red Hat Quay 3.11 管理 Red Hat Quay

管理 Red Hat Quay

法律通告

摘要

管理 Red Hat Quay

目录

前言

部署 Red Hat Quay registry 后，可以通过多种方式配置和管理该部署。这里涵盖的主题包括：

- 高级 Red Hat Quay 配置
- 设置通知以提醒您新的 Red Hat Quay 发行版本
- 使用 SSL/TLS 证书保护连接
- 将操作日志存储定向到 Elasticsearch
- 使用 Clair 配置镜像安全扫描
- 使用 Container Security Operator 扫描 pod 镜像
- 将 Red Hat Quay 与 Quay Bridge Operator 集成
- 使用存储库镜像镜像镜像
- 使用 BitTorrent 服务共享 Red Hat Quay 镜像
- 使用 LDAP 验证用户
- 为 Prometheus 和 Grafana 指标启用 Quay
- 设置地理复制
- Red Hat Quay 故障排除

如需 Red Hat Quay 配置字段的完整列表，请参阅 Configure Red Hat Quay 页面。

第 1 章 高级 RED HAT QUAY 配置

您可以使用以下方法之一在初始部署后配置 Red Hat Quay：

- **编辑 config.yaml 文件**。**config.yaml** 文件包含 Red Hat Quay 集群的大部分配置信息。直接编辑 **config.yaml** 文件是高级调整和启用特定功能的主要方法。
- **使用 Red Hat Quay API**。一些 Red Hat Quay 功能可以通过 API 配置。

本节中的内容论述了如何使用上述每个接口以及如何使用高级功能配置部署。

1.1. 使用 API 修改 RED HAT QUAY

有关如何访问 Red Hat Quay API 的信息，请参阅 Red Hat Quay API 指南 。

1.2. 编辑 CONFIG.YAML 文件以修改 RED HAT QUAY

可以通过直接编辑 **config.yaml** 文件来实现高级功能。Red Hat Quay 功能和设置的所有配置字段都包括在 Red Hat Quay 配置指南 中。

以下示例是在 **config.yaml** 文件中直接更改的设置。在编辑 **config.yaml** 文件用于其他功能和设置时，请使用此示例。

1.2.1. 将名称和公司添加到 Red Hat Quay 登陆

通过将 **FEATURE_USER_METADATA** 字段设置为 **true**，用户在首次登录时提示用户输入其名称和公司。这是一个可选字段，但可以为您的 Red Hat Quay 用户提供额外的数据。

使用以下步骤将名称和公司添加到 Red Hat Quay 登录页面。

流程

1. 在 **config.yaml** 文件中添加或设置 **FEATURE_USER_METADATA** 配置字段为 **true**。例如：

```
# ...
FEATURE_USER_METADATA: true
# ...
```

1. 重新部署 Red Hat Quay。
2. 现在，当系统提示登录时，请求用户输入以下信息：

第 2 章 使用配置 API

配置工具公开 4 个端点，可用于构建、验证、捆绑包和部署配置。config-tool API 记录在 https://github.com/quay/config-tool/blob/master/pkg/lib/editor/API.md 中。在本节中，您将了解如何使用 API 来检索当前配置，以及如何验证您所做的任何更改。

2.1. 检索默认配置

如果您第一次运行配置工具，且没有现有配置，您可以检索默认配置。以 config 模式启动容器：

```
$ sudo podman run --rm -it --name quay_config \
  -p 8080:8080 \
  registry.redhat.io/quay/quay-rhel8:v3.11.1 config secret
```

使用 **配置** API 的配置端点来获取默认值：

```
$ curl -X GET -u quayconfig:secret http://quay-server:8080/api/v1/config  | jq
```

返回的值是 JSON 格式的默认配置：

```
{
  "config.yaml": {
    "AUTHENTICATION_TYPE": "Database",
    "AVATAR_KIND": "local",
    "DB_CONNECTION_ARGS": {
      "autorollback": true,
      "threadlocals": true
    },
    "DEFAULT_TAG_EXPIRATION": "2w",
    "EXTERNAL_TLS_TERMINATION": false,
    "FEATURE_ACTION_LOG_ROTATION": false,
    "FEATURE_ANONYMOUS_ACCESS": true,
    "FEATURE_APP_SPECIFIC_TOKENS": true,
    ....
  }

}
```

2.2. 检索当前配置

如果您已经配置和部署 Quay registry，请停止容器并以配置模式重启它，将现有配置载入为卷：

```
$ sudo podman run --rm -it --name quay_config \
  -p 8080:8080 \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1 config secret
```

使用 **API** 的配置端点来获取当前的配置：

```
$ curl -X GET -u quayconfig:secret http://quay-server:8080/api/v1/config  | jq
```

返回的值是 JSON 格式的当前配置，包括数据库和 Redis 配置数据：

```
{
  "config.yaml": {
    ....
    "BROWSER_API_CALLS_XHR_ONLY": false,
    "BUILDLOGS_REDIS": {
      "host": "quay-server",
      "password": "strongpassword",
      "port": 6379
    },
    "DATABASE_SECRET_KEY": "4b1c5663-88c6-47ac-b4a8-bb594660f08b",
    "DB_CONNECTION_ARGS": {
      "autorollback": true,
      "threadlocals": true
    },
    "DB_URI": "postgresql://quayuser:quaypass@quay-server:5432/quay",
    "DEFAULT_TAG_EXPIRATION": "2w",
    ....


  }

}
```

2.3. 使用 API 验证配置

您可以通过将其发布到 **config/validate** 端点来验证配置：

```
curl -u quayconfig:secret --header 'Content-Type: application/json' --request POST --data '
{
  "config.yaml": {
    ....
    "BROWSER_API_CALLS_XHR_ONLY": false,
    "BUILDLOGS_REDIS": {
      "host": "quay-server",
      "password": "strongpassword",
      "port": 6379
    },
    "DATABASE_SECRET_KEY": "4b1c5663-88c6-47ac-b4a8-bb594660f08b",
    "DB_CONNECTION_ARGS": {
      "autorollback": true,
      "threadlocals": true
    },
    "DB_URI": "postgresql://quayuser:quaypass@quay-server:5432/quay",
    "DEFAULT_TAG_EXPIRATION": "2w",
    ....


  }

} http://quay-server:8080/api/v1/config/validate | jq
```

返回的值是包含配置中找到的错误的数组。如果配置有效，则返回空数组 **[]**。

2.4. 确定必填字段

您可以通过将空配置结构发布到 **config/validate** 端点来确定必填字段：

```
curl -u quayconfig:secret --header 'Content-Type: application/json' --request POST --data '
{
  "config.yaml": {
  }

} http://quay-server:8080/api/v1/config/validate | jq
```

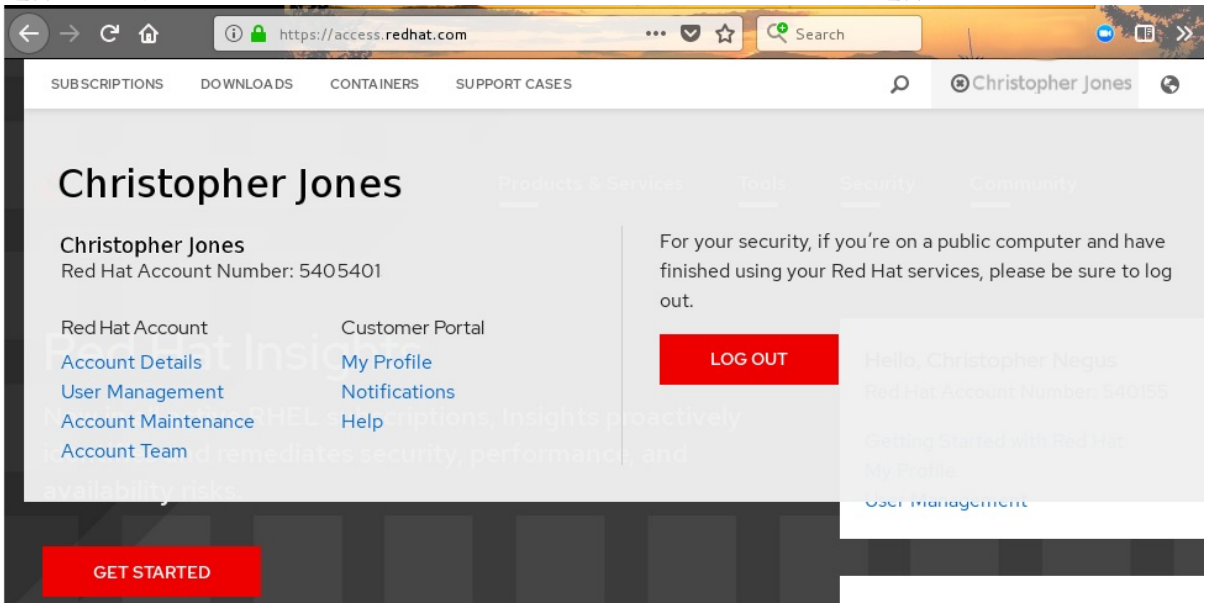
返回的值是代表需要哪些字段的数组：

```
[
  {
    "FieldGroup": "Database",
    "Tags": [
      "DB_URI"
    ],
    "Message": "DB_URI is required."
  },
  {
    "FieldGroup": "DistributedStorage",
    "Tags": [
      "DISTRIBUTED_STORAGE_CONFIG"
    ],
    "Message": "DISTRIBUTED_STORAGE_CONFIG must contain at least one storage location."
  },
  {
    "FieldGroup": "HostSettings",
    "Tags": [
      "SERVER_HOSTNAME"
    ],
    "Message": "SERVER_HOSTNAME is required"
  },
  {
    "FieldGroup": "HostSettings",
    "Tags": [
      "SERVER_HOSTNAME"
    ],
    "Message": "SERVER_HOSTNAME must be of type Hostname"
  },
  {
    "FieldGroup": "Redis",
    "Tags": [
      "BUILDLOGS_REDIS"
    ],
    "Message": "BUILDLOGS_REDIS is required"
  }
]
```

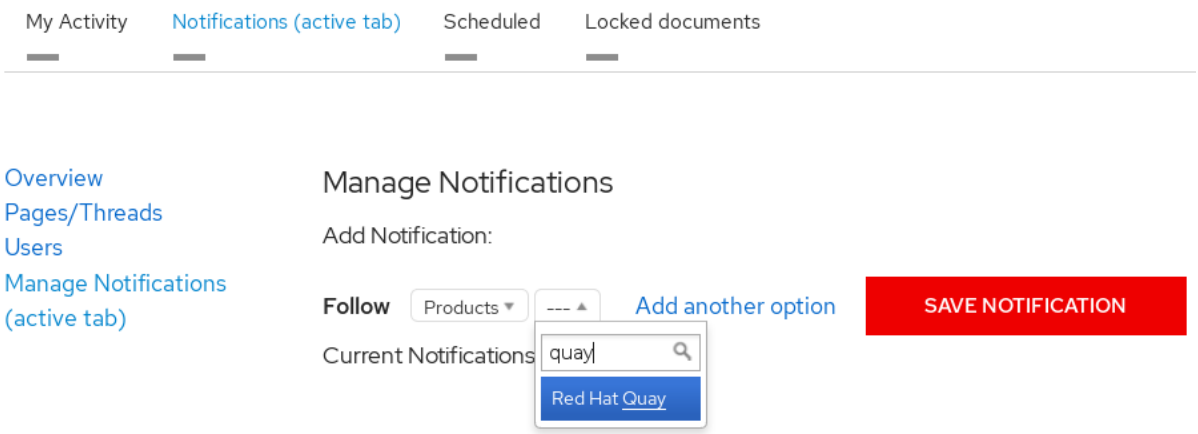
第 3 章 获取 RED HAT QUAY 发行通知

要保持最新的 Red Hat Quay 发行版本和其他与 Red Hat Quay 相关的更改，您可以 在红帽客户门户网站上 注册以更新通知。注册通知后，您将收到通知，了解是否有新的 Red Hat Quay 版本、更新文档或其他 Red Hat Quay 新闻。

1. 使用您的红帽客户帐户凭证登录 红帽客户门户。
2. 选择您的用户名（右上）来查看 Red Hat Account 和 Customer Portal 选择：



3. 选择通知。此时会出现您的个人资料活动页面。
4. 选择 Notifications 选项卡。
5. 选择 Manage Notifications。
6. 选择 Follow，然后从下拉菜单中选择 Products。
7. 从 Products 旁边的下拉菜单中，搜索并选择 Red Hat Quay：



8. 选择 SAVE NOTIFICATION 按钮。下一步，当对 Red Hat Quay 产品（如新版本）有更改时，您将收到通知。

第 4 章 使用 SSL 保护到 RED HAT QUAY 的连接

4.1. 使用 SSL/TLS

要使用自签名证书配置 Red Hat Quay，您必须创建一个证书颁发机构(CA)和名为 **ssl.cert** 和 **ssl.key** 的主密钥文件。

注意

以下示例假设您已使用 DNS 或其他命名机制配置了服务器主机名 **quay-server.example.com**，如在 **/etc/hosts** 文件中添加条目。如需更多信息，请参阅"为 Red Hat Quay 配置端口映射"。

4.2. 创建证书颁发机构

使用以下步骤创建证书颁发机构(CA)。

流程

1. 输入以下命令生成 root CA 密钥：

```
$ openssl genrsa -out rootCA.key 2048
```

2. 输入以下命令生成 root CA 证书：

```
$ openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
```

3. 输入您的证书请求中包含的信息，包括服务器主机名，例如：

```
Country Name (2 letter code) [XX]:IE
State or Province Name (full name) []:GALWAY
Locality Name (eg, city) [Default City]:GALWAY
Organization Name (eg, company) [Default Company Ltd]:QUAY
Organizational Unit Name (eg, section) []:DOCS
Common Name (eg, your name or your server's hostname) []:quay-server.example.com
```

4.2.1. 签名证书

使用以下步骤为证书签名。

流程

1. 输入以下命令生成服务器密钥：

```
$ openssl genrsa -out ssl.key 2048
```

2. 输入以下命令生成签名请求：

```
$ openssl req -new -key ssl.key -out ssl.csr
```

3. 输入您的证书请求中包含的信息，包括服务器主机名，例如：

```
Country Name (2 letter code) [XX]:IE
State or Province Name (full name) []:GALWAY
Locality Name (eg, city) [Default City]:GALWAY
Organization Name (eg, company) [Default Company Ltd]:QUAY
Organizational Unit Name (eg, section) []:DOCS
Common Name (eg, your name or your server's hostname) []:quay-server.example.com
```

4. 创建配置文件 **openssl.cnf**，指定服务器主机名，例如：

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 = quay-server.example.com
IP.1 = 192.168.1.112
```

5. 使用配置文件生成证书 **ssl.cert** ：

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

4.3. 使用命令行界面配置 SSL/TLS

使用以下步骤使用 CLI 配置 SSL/TLS。

先决条件

- 您已创建了证书颁发机构并签署证书。

流程

1. 将证书文件和主密钥文件复制到您的配置目录中，确保它们分别命名为 **ssl.cert** 和 **ssl.key** ：

```
cp ~/ssl.cert ~/ssl.key $QUAY/config
```

2. 输入以下命令进入 **$QUAY/config** 目录：

```
$ cd $QUAY/config
```

3. 编辑 **config.yaml** 文件并指定您希望 Red Hat Quay 处理 TLS/SSL：

config.yaml

```
...
SERVER_HOSTNAME: quay-server.example.com
```

```
...
PREFERRED_URL_SCHEME: https
...
```

4. 可选：输入以下命令将 rootCA.pem 文件的内容应用到 ssl.cert 文件的末尾：

```
$ cat rootCA.pem >> ssl.cert
```

5. 输入以下命令停止 **Quay** 容器：

```
$ sudo podman stop quay
```

6. 输入以下命令重启 registry：

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -v $QUAY/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1
```

4.4. 使用 RED HAT QUAY UI 配置 SSL/TLS

使用以下步骤使用 Red Hat Quay UI 配置 SSL/TLS。

要使用命令行界面配置 SSL/TLS，请参阅"使用命令行界面配置 SSL/TLS"。

先决条件

- 您已创建了证书颁发机构并签署证书。

流程

1. 以配置模式启动 **Quay** 容器：

```
$ sudo podman run --rm -it --name quay_config -p 80:8080 -p 443:8443
registry.redhat.io/quay/quay-rhel8:v3.11.1 config secret
```

2. 在 **Server Configuration** 部分中，选择 **Red Hat Quay 处理 SSL/TLS**的 TLS。上传之前创建的证书文件和私钥文件，确保 **Server Hostname** 与创建证书时使用的值匹配。

3. 验证并下载更新的配置。

4. 输入以下命令停止 **Quay** 容器，然后重启 registry：

```
$ sudo podman rm -f quay
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
--name=quay \
-v $QUAY/config:/conf/stack:Z \
-v $QUAY/storage:/datastorage:Z \
registry.redhat.io/quay/quay-rhel8:v3.11.1
```

4.5. 使用 CLI 测试 SSL/TLS 配置

使用以下步骤使用 CLI 测试 SSL/TLS 配置。

流程

- 输入以下命令尝试登录到启用了 SSL/TLS 的 Red Hat Quay registry：

```
$ sudo podman login quay-server.example.com
```

输出示例

```
Error: error authenticating creds for "quay-server.example.com": error pinging docker registry quay-server.example.com: Get "https://quay-server.example.com/v2/": x509: certificate signed by unknown authority
```

1. 因为 Podman 不信任自签名证书，所以必须使用 **--tls-verify=false** 选项：

```
$ sudo podman login --tls-verify=false quay-server.example.com
```

输**出示例**

```
Login Succeeded!
```

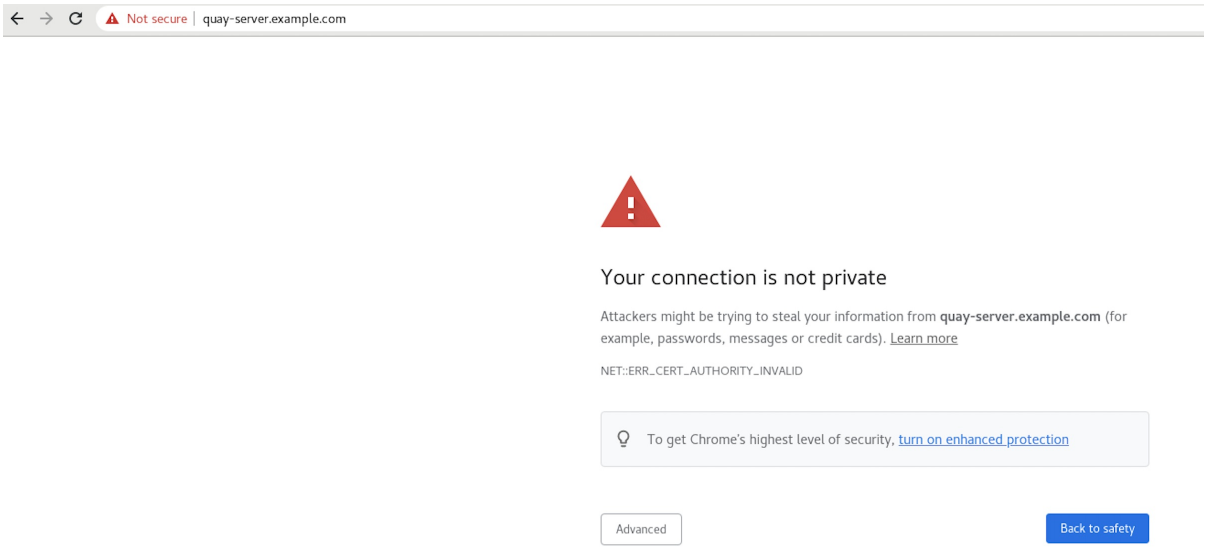
在后面的部分中，您要将 Podman 配置为信任 root 证书颁发机构。

4.6. 使用浏览器测试 SSL/TLS 配置

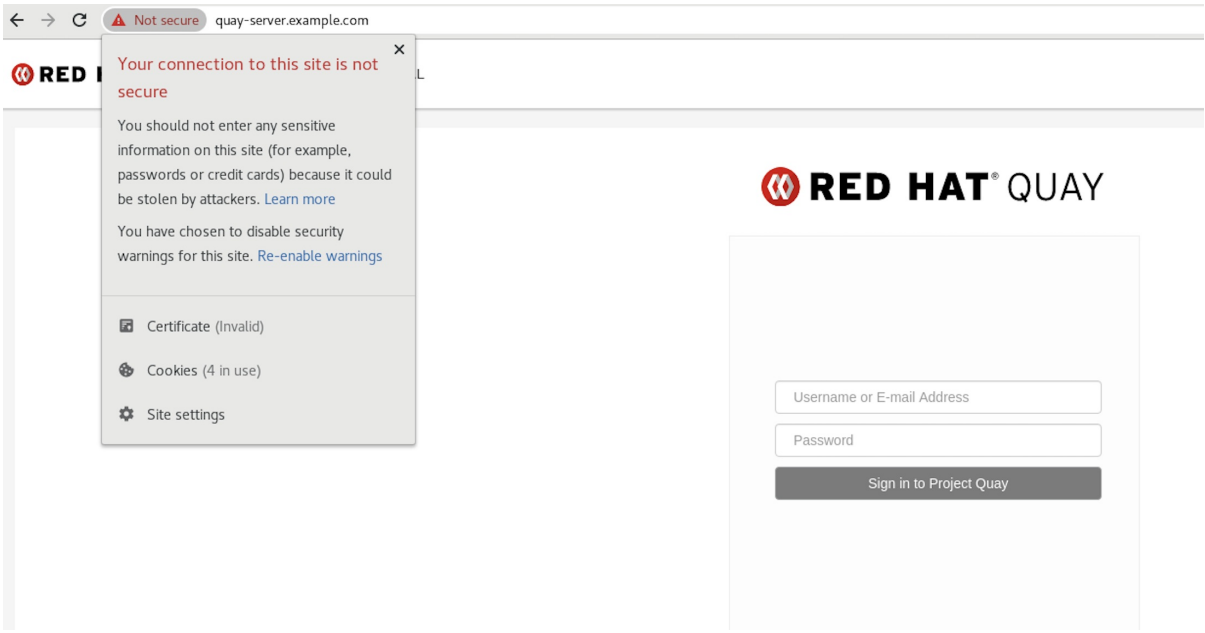
使用以下步骤使用浏览器测试 SSL/TLS 配置。

流程

1. 导航到您的 Red Hat Quay registry 端点，例如 **https://quay-server.example.com**。如果正确配置，浏览器会警告潜在的风险：



2. 继续登录屏幕。浏览器通知您连接不安全。例如：



在以下部分中，您要将 Podman 配置为信任 root 证书颁发机构。

4.7. 配置 PODMAN 以信任证书颁发机构

Podman 使用两个路径来查找证书颁发机构(CA)文件： **/etc/containers/certs.d/** 和 **/etc/docker/certs.d/**。使用以下步骤将 Podman 配置为信任 CA。

流程

1. 将 root CA 文件复制到 **/etc/containers/certs.d/** 或 **/etc/docker/certs.d/** 之一。使用由服务器主机名确定的确切路径，并将文件命名为 **ca.crt** ：

```
$ sudo cp rootCA.pem /etc/containers/certs.d/quay-server.example.com/ca.crt
```

2. 在登录到 Red Hat Quay registry 时，验证您不再需要使用 **--tls-verify=false** 选项：

```
$ sudo podman login quay-server.example.com
```

输**出示例**

```
Login Succeeded!
```

4.8. 将系统配置为信任证书颁发机构

使用以下步骤将系统配置为信任证书颁发机构。

流程

1. 输入以下命令将 **rootCA.pem** 文件复制到整合的系统范围信任存储中：

```
$ sudo cp rootCA.pem /etc/pki/ca-trust/source/anchors/
```

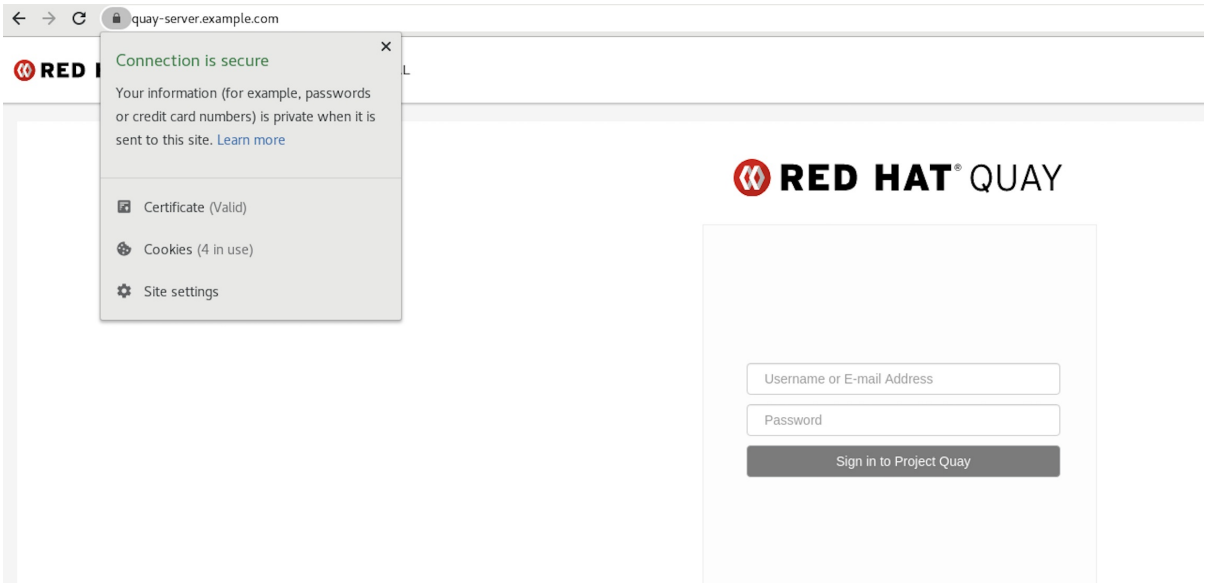
2. 输入以下命令更新系统范围的信任存储配置：

```
$ sudo update-ca-trust extract
```

3. 可选。您可以使用 **trust list** 命令确保已配置了 **Quay** 服务器：

```
$ trust list | grep quay
    label: quay-server.example.com
```

现在，当您浏览 **https://quay-server.example.com** 的 registry 时，锁定图标会显示连接是安全的：



4. 要从系统范围的信任中删除 **rootCA.pem** 文件，请删除该文件并更新配置：

```
$ sudo rm /etc/pki/ca-trust/source/anchors/rootCA.pem
```

```
$ sudo update-ca-trust extract
```

```
$ trust list | grep quay
```

如需更多信息，请参阅 使用共享系统证书的 章节中的 RHEL 9 文档。

第 5 章 将 TLS 证书添加到 RED HAT QUAY CONTAINER

要将自定义 TLS 证书添加到 Red Hat Quay，请在 Red Hat Quay config 目录下创建一个名为 **extra_ca_certs/** 的新目录。将任何所需的特定于站点的 TLS 证书复制到这个新目录中。

5.1. 将 TLS 证书添加到 RED HAT QUAY

1. 查看要添加到容器中的证书

```
$ cat storage.crt
-----BEGIN CERTIFICATE-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
[...]
-----END CERTIFICATE-----
```

2. 创建 certs 目录并复制证书

```
$ mkdir -p quay/config/extra_ca_certs
$ cp storage.crt quay/config/extra_ca_certs/
$ tree quay/config/
├── config.yaml
├── extra_ca_certs
│   ├── storage.crt
```

3. 使用 **podman ps** 获取 **Quay** 容器的 **CONTAINER ID** ：

```
$ sudo podman ps
CONTAINER ID  IMAGE                          COMMAND               CREATED
STATUS            PORTS
5a3e82c4a75f  <registry>/<repo>/quay:v3.11.1 "/sbin/my_init"  24 hours ago  Up
18 hours  0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp, 443/tcp  grave_keller
```

4. 使用该 ID 重启容器：

```
$ sudo podman restart 5a3e82c4a75f
```

5. 检查复制到容器命名空间中的证书：

```
$ sudo podman exec -it 5a3e82c4a75f cat /etc/ssl/certs/storage.pem
-----BEGIN CERTIFICATE-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
```

5.2. 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

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 6 章 为 ELASTICSEARCH 和 SPLUNK 配置操作日志存储

默认情况下，使用日志保存在 Red Hat Quay 数据库中，并通过 Web UI 在机构和存储库级别公开。查看日志条目需要适当的管理特权。对于有大量日志操作的部署，您可以将使用日志存储在 Elasticsearch 和 Splunk 中，而不是 Red Hat Quay 数据库后端。

6.1. 为 ELASTICSEARCH 配置操作日志存储

注意

要为 Elasticsearch 配置操作日志存储，您必须提供自己的 Elasticsearch 堆栈；它不包含在 Red Hat Quay 中作为可自定义组件。

通过更新 **config.yaml** 文件，可以在 Red Hat Quay 部署或部署后启用 Elasticsearch 日志记录。配置后，仍然通过 Web UI 为存储库和机构提供使用日志访问权限。

使用以下步骤为 Elasticsearch 配置操作日志存储：

流程

1. 获取 Elasticsearch 帐户。
2. 更新 Red Hat Quay **config.yaml** 文件，使其包含以下信息：

```
# ...
LOGS_MODEL: elasticsearch (1)
LOGS_MODEL_CONFIG:
   producer: elasticsearch (2)
   elasticsearch_config:
      host: http://<host.elasticsearch.example>:<port> (3)
      port: 9200 (4)
      access_key: <access_key> (5)
      secret_key: <secret_key> (6)
      use_ssl: True (7)
      index_prefix: <logentry> (8)
      aws_region: <us-east-1> (9)
# ...
```

(1) 处理日志数据的方法。

(2) 选择 Elasticsearch 或 Kinesis 将日志定向到 AWS 上的中间 Kinesis 流。您需要设置自己的管道，将日志从 Kinesis 发送到 Elasticsearch，例如 Logstash。

(3) 提供 Elasticsearch 服务的系统的主机名或 IP 地址。

(4) 在您刚刚输入的主机上提供 Elasticsearch 服务的端口号。请注意，端口必须可从运行 Red Hat Quay registry 的所有系统访问。默认为 TCP 端口 **9200**。

(5) 如果需要，需要访问 Elasticsearch 服务所需的访问密钥。

(6) 如果需要，获取 Elasticsearch 服务访问权限所需的 secret 密钥。

(7) 是否将 SSL/TLS 用于 Elasticsearch。默认值为 **True**。

8 选择要附加到日志条目的前缀。

9 如果您在 AWS 上运行，请设置 AWS 区域（否则，将其留空）。

3. 可选。如果使用 Kinesis 作为日志制作者，则必须在 **config.yaml** 文件中包含以下字段：

```
kinesis_stream_config:
  stream_name: <kinesis_stream_name> 1
  access_key: <aws_access_key> 2
  secret_key: <aws_secret_key> 3
  aws_region: <aws_region> 4
```

1 Kinesis 流的名称。

2 如果需要，需要访问 Kinesis 流所需的 AWS 访问密钥名称。

3 需要访问 Kinesis 流所需的 AWS secret 密钥名称（如果需要）。

4 AWS 区域。

4. 保存 **config.yaml** 文件并重启 Red Hat Quay 部署。

6.2. 为 SPLUNK 配置操作日志存储

Splunk 是 Elasticsearch 的替代选择，可以为 Red Hat Quay 数据提供日志分析。

使用配置工具，可以在 Red Hat Quay 部署或部署后启用 Splunk 日志记录。生成的配置存储在 **config.yaml** 文件中。配置后，通过 Splunk Web UI 为存储库和机构提供使用日志访问权限。

使用以下步骤为您的 Red Hat Quay 部署启用 Splunk。

6.2.1. 为 Splunk 安装并创建用户名

使用以下步骤安装并创建 Splunk 凭证。

流程

1. 进入 Splunk 并输入所需凭证来创建 Splunk 帐户。
2. 导航到 Splunk Enterprise **Free Trial** 页面，选择您的平台和安装软件包，然后单击 **Download Now**。
3. 在您的机器上安装 Splunk 软件。出现提示时，创建一个用户名，如 mvapich **_admin** 和密码。
4. 创建用户名和密码后，将为 Splunk 部署提供一个 localhost URL，例如 **http://<sample_url>.remote.csb:8000/**。在您首选的浏览器中打开 URL。
5. 使用安装期间创建的用户名和密码登录。您会被定向到 Splunk UI。

6.2.2. 生成 Splunk 令牌

使用以下步骤为 Splunk 创建 bearer 令牌。

6.2.2.1. 使用 Splunk UI 生成 Splunk 令牌

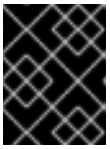
使用以下步骤，使用 Splunk UI 为 Splunk 创建 bearer 令牌。

先决条件

- 已安装 Splunk 并创建一个用户名。

流程

1. 在 Splunk UI 中，进入到 **Settings → Tokens**。
2. 单击 **Enable Token Authentication**。
3. 点 **Token Settings** 并选择 **Token Authentication**（如果需要），确保启用了 **Token Authentication**。
4. 可选：为您的令牌设置过期时间。默认值为 30 天。
5. 点击 **Save**。
6. 单击 **New Token**。
7. 输入 **User** 和 **Audience** 的信息。
8. 可选：设置 **Expiration** 和 **Not Before** 信息。
9. 点 **Create**。您的令牌会出现在 **Token** 框中。立即复制令牌。



重要

如果在复制令牌前开箱即用，则必须创建新令牌。在关闭 **New Token** 窗口后，整个令牌不可用。

6.2.2.2. 使用 CLI 生成 Splunk 令牌

使用以下步骤使用 CLI 为 Splunk 创建 bearer 令牌。

先决条件

- 已安装 Splunk 并创建一个用户名。

流程

1. 在 CLI 中，输入以下 **CURL** 命令以启用令牌身份验证，传递您的 Splunk 用户名和密码：

```
$ curl -k -u <username>:<password> -X POST <scheme>://<host>:
<port>/services/admin/token-auth/tokens_auth -d disabled=false
```

2. 输入以下 **CURL** 命令，传递 Splunk 用户名和密码来创建令牌。

```
$ curl -k -u <username>:<password> -X POST <scheme>://<host>:
<port>/services/authorization/tokens?output_mode=json --data name=<username> --data
audience=Users --data-urlencode expires_on=+30d
```

3. 保存生成的 bearer 令牌。

6.2.3. 将 Red Hat Quay 配置为使用 Splunk

使用以下步骤将 Red Hat Quay 配置为使用 Splunk。

先决条件

- 已安装 Splunk 并创建一个用户名。
- 您已生成了一个 Splunk bearer 令牌。

流程

1. 打开 Red Hat Quay **config.yaml** 文件并添加以下配置字段：

```
# ...
LOGS_MODEL: splunk
LOGS_MODEL_CONFIG:
    producer: splunk
    splunk_config:
        host: http://<user_name>.remote.csb (1)
        port: 8089 (2)
        bearer_token: <bearer_token> (3)
        url_scheme: <http/https> (4)
        verify_ssl: False (5)
        index_prefix: <splunk_log_index_name> (6)
        ssl_ca_path: <location_to_ssl-ca-cert.pem> (7)
# ...
```

(1) 字符串.Splunk 集群端点。

(2) 整数.Splunk 管理集群端点端口。与 Splunk GUI 托管端口不同。可以在 Splunk UI 的 **Settings → Server Settings → General Settings** 下找到。

(3) 字符串.为 Splunk 生成的 bearer 令牌。

(4) 字符串.访问 Splunk 服务的 URL 方案。如果将 Splunk 配置为使用 TLS/SSL，则必须是 **https**。

(5) 布尔值.是否启用 TLS/SSL。默认值为 **true**。

(6) 字符串.Splunk 索引前缀。可以是新的、使用的索引。可以从 Splunk UI 创建。

(7) 字符串.到包含用于 TLS/SSL 验证的证书颁发机构(CA)的单个 **.pem** 文件的相对容器路径。

2. 如果要配置 **ssl_ca_path**，您必须配置 SSL/TLS 证书，以便 Red Hat Quay 将信任它。

   a. 如果您使用 Red Hat Quay 的独立部署，可以通过将证书文件放在 **extra_ca_certs** 目录内或通过 **ssl_ca_path** 指定来提供 SSL/TLS 证书。

   b. 如果使用 Red Hat Quay Operator，请创建一个配置捆绑包 secret，包括 Splunk 服务器的证书颁发机构(CA)。例如：

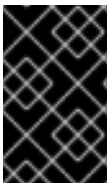
```
$ oc create secret generic --from-file config.yaml=./config_390.yaml --from-file
extra_ca_cert_splunkserver.crt=./splunkserver.crt config-bundle-secret
```

在 **config.yaml** 中指定 **conf/stack/extra_ca_certs/mvapichserver.crt** 文件。例如：

```
# ...
LOGS_MODEL: splunk
LOGS_MODEL_CONFIG:
   producer: splunk
   splunk_config:
      host: ec2-12-345-67-891.us-east-2.compute.amazonaws.com
      port: 8089
      bearer_token: eyJra
      url_scheme: https
      verify_ssl: true
      index_prefix: quay123456
      ssl_ca_path: conf/stack/splunkserver.crt
# ...
```

6.2.4. 创建操作日志

使用以下步骤创建可将操作日志转发到 Splunk 的用户帐户。



重要

您必须使用 Splunk UI 查看 Red Hat Quay 操作日志。目前，在 Red Hat Quay **Usage Logs** 页面中查看 Splunk 操作日志不受支持，并返回以下信息： **Method not implemented。Splunk 不支持日志查找。**

先决条件

- 已安装 Splunk 并创建一个用户名。
- 您已生成了一个 Splunk bearer 令牌。
- 您已配置了 Red Hat Quay **config.yaml** 文件来启用 Splunk。

流程

1. 登录到您的 Red Hat Quay 部署。
2. 点击您要用来为 Splunk 创建操作日志的组织名称。
3. 在导航窗格中，点 **Robot Accounts → Create Robot Account**。
4. 出现提示时，输入机器人帐户的名称，如 **spunkrobotaccount**，然后单击 **Create robot account**。
5. 在您的浏览器中，打开 Splunk UI。
6. 单击 **Search and Reporting**。
7. 在搜索栏中输入索引名称，例如 <mvapich **_log_index_name** > 并按 **Enter**。
   在 Splunk UI 上填充搜索结果。日志以 JSON 格式转发。响应可能类似如下：

```
{
  "log_data": {
    "kind": "authentication", (1)
    "account": "quayuser123", (2)
    "performer": "John Doe", (3)
    "repository": "projectQuay", (4)
    "ip": "192.168.1.100", (5)
    "metadata_json": {...}, (6)
    "datetime": "2024-02-06T12:30:45Z" (7)
  }
}
```

1. 指定日志事件的类型。在本例中，**身份验证** 表示日志条目与身份验证事件相关。
2. 事件涉及的用户帐户。
3. 执行该操作的个人。
4. 与事件关联的存储库。
5. 执行操作的 IP 地址。
6. 可能包含与事件相关的其他元数据。
7. 事件发生时的时间戳。

6.3. 了解使用日志

默认情况下，使用日志保存在 Red Hat Quay 数据库中。它们通过 Web UI、机构和存储库级别以及 **Superuser Admin Panel** 来公开。

数据库日志捕获 Red Hat Quay 中的各种事件，如更改帐户计划、用户操作和常规操作。日志条目包括诸如操作(**kind_id**)、执行操作的用户(**account_id** 或 **performer_id**)、时间戳(**日期时间**)以及与操作相关的其他相关数据(**metadata_json**)。

6.3.1. 查看数据库日志

以下流程演示了如何查看存储在 PostgreSQL 数据库中的存储库日志。

先决条件

- 有管理特权。
- 已安装 **psql** CLI 工具。

流程

1. 输入以下命令登录到您的 Red Hat Quay PostgreSQL 数据库：

```
$ psql -h <quay-server.example.com> -p 5432 -U <user_name> -d <database_name>
```

输**出示例**

-

```
psql (16.1, server 13.7)
Type "help" for help.
```

2. 可选。输入以下命令显示 PostgreSQL 数据库的表列表：

```
quay=> \dt
```

输**出示例**

```
                List of relations
 Schema |          Name           | Type  |  Owner
--------+-------------------------+-------+----------
 public | logentry                | table | quayuser
 public | logentry2               | table | quayuser
 public | logentry3               | table | quayuser
 public | logentrykind            | table | quayuser
...
```

3. 您可以输入以下命令来返回返回返回日志信息所需的 **repository_ids** 列表：

```
quay=> SELECT id, name FROM repository;
```

输**出示例**

```
 id |       name
----+--------------------
  3 | new_repository_name
  6 | api-repo
  7 | busybox
...
```

4. 输入以下命令使用 **logentry3** relationship 显示有关其中一个软件仓库的日志信息：

```
SELECT * FROM logentry3 WHERE repository_id = <repository_id>;
```

输**出示例**

```
 id | kind_id | account_id | performer_id | repository_id | datetime | ip |    metadata_json

 59 | 14 | 2 | 1 | 6 | 2024-05-13 15:51:01.897189 | 192.168.1.130 | {"repo": "api-repo",
"namespace": "test-org"}
```

在上例中，返回以下信息：

```
{
  "log_data": {
    "id": 59 (1)
    "kind_id": "14", (2)
    "account_id": "2", (3)
    "performer_id": "1", (4)
    "repository_id": "6", (5)
```

```
    "ip": "192.168.1.100", 6
    "metadata_json": {"repo": "api-repo", "namespace": "test-org"} 7
    "datetime": "2024-05-13 15:51:01.897189" 8
  }
}
```

1. 日志条目的唯一标识符。
2. 执行的操作。在本例中，它是 **14**。以下部分中的 key 或 表显示此 **kind_id** 与创建存储库相关。
3. 执行操作的帐户。
4. 操作的执行者。
5. 操作所在的存储库。在本例中，**6** 个与在第 3 步中发现的 **api-repo** 关联。
6. 执行操作的 IP 地址。
7. 元数据信息，包括存储库的名称及其命名空间。
8. 执行该操作的时间。

6.3.2. 日志条目 kind_ids

下表代表了与 Red Hat Quay 操作关联的 **kind_ids**。

kind_id	操作	描述
1	account_change_cc	信用卡信息的改变。
2	account_change_password	更改帐户密码。
3	account_change_plan	更改帐户计划.
4	account_convert	帐户转换.
5	add_repo_accesstoken	将访问令牌添加到存储库。
6	add_repo_notification	向存储库添加通知.
7	add_repo_permission	向存储库添加权限。
8	add_repo_webhook	将 webhook 添加到存储库。
9	build_dockerfile	构建 Dockerfile.
10	change_repo_permission	更改存储库的权限。
11	change_repo_visibility	更改存储库的可见性。

kind_id	操作	描述
12	create_application	创建应用程序。
13	create_prototype_permission	为原型创建权限。
14	create_repo	创建存储库.
15	create_robot	创建机器人（服务帐户或 bot）。
16	create_tag	创建标签.
17	delete_application	删除应用程序。
18	delete_prototype_permission	删除模型列表的权限。
19	delete_repo	删除存储库。
20	delete_repo_accesstoken	从存储库删除访问令牌。
21	delete_repo_notification	从存储库删除通知。
22	delete_repo_permission	从存储库删除权限。
23	delete_repo_trigger	删除存储库触发器。
24	delete_repo_webhook	从存储库删除 webhook。
25	delete_robot	删除机器人.
26	delete_tag	删除标签。
27	manifest_label_add	向清单添加标签。
28	manifest_label_delete	从清单中删除标签。
29	modify_prototype_permission	修改原型的权限。
30	move_tag	移动标签。
31	org_add_team_member	将成员添加到团队.
32	org_create_team	在组织内创建团队。
33	org_delete_team	删除机构中的一个团队。

kind_id	操作	描述
34	org_delete_team_member_invite	删除团队成员邀请。
35	org_invite_team_member	将成员邀请给机构中的一个团队。
36	org_remove_team_member	从团队中删除成员。
37	org_set_team_description	设置团队的描述。
38	org_set_team_role	设置团队的角色。
39	org_team_member_invite_accepted	接受团队成员邀请。
40	org_team_member_invite_declined	拒绝团队成员邀请。
41	pull_repo	从存储库拉取。
42	push_repo	推送到存储库。
43	regenerate_robot_token	重新生成机器人令牌.
44	repo_verb	通用存储库操作（可在其他位置定义）。
45	reset_application_client_secret	重置应用的客户端机密。
46	revert_tag	恢复标签。
47	service_key_approve	批准服务密钥。
48	service_key_create	创建服务密钥。
49	service_key_delete	删除服务密钥。
50	service_key_extend	扩展服务密钥。
51	service_key_modify	修改服务密钥。
52	service_key_rotate	轮转服务密钥。
53	setup_repo_trigger	设置存储库触发器。
54	set_repo_description	设置存储库的描述。

kind_id	操作	描述
55	take_ownership	获取资源的所有权。
56	update_application	更新应用程序。
57	change_repo_trust	更改存储库的信任级别。
58	reset_repo_notification	重置存储库通知.
59	change_tag_expiration	更改标签的过期日期。
60	create_app_specific_token	创建特定于应用的令牌。
61	revoke_app_specific_token	撤销特定于应用程序的令牌。
62	toggle_repo_trigger	切换打开或关闭存储库触发器。
63	repo_mirror_enabled	启用存储库镜像。
64	repo_mirror_disabled	禁用存储库镜像。
65	repo_mirror_config_changed	更改存储库镜像的配置。
66	repo_mirror_sync_started	启动存储库镜像同步。
67	repo_mirror_sync_failed	存储库镜像同步失败。
68	repo_mirror_sync_success	存储库镜像同步成功。
69	repo_mirror_sync_now_requested	请求的即时存储库镜像同步。
70	repo_mirror_sync_tag_success	存储库镜像标签同步成功。
71	repo_mirror_sync_tag_failed	仓库镜像标签同步失败。
72	repo_mirror_sync_test_success	存储库镜像同步测试成功。
73	repo_mirror_sync_test_failed	存储库镜像同步测试失败。
74	repo_mirror_sync_test_started	仓库镜像同步测试已启动。
75	change_repo_state	更改存储库的状态。

kind_id	操作	描述
76	create_proxy_cache_config	创建代理缓存配置。
77	delete_proxy_cache_config	删除代理缓存配置。
78	start_build_trigger	启动构建触发器。
79	cancel_build	取消构建.
80	org_create	创建机构。
81	org_delete	删除机构。
82	org_change_email	更改组织电子邮件.
83	org_change_invoicing	更改组织情况.
84	org_change_tag_expiration	更改机构标签过期。
85	org_change_name	更改机构名称。
86	user_create	创建用户。
87	user_delete	删除用户。
88	user_disable	禁用用户。
89	user_enable	启用用户.
90	user_change_email	更改用户电子邮件.
91	user_change_password	更改用户密码.
92	user_change_name	更改用户名。
93	user_change_invoicing	更改用户 invoicing.
94	user_change_tag_expiration	更改用户标签过期。
95	user_change_metadata	更改用户元数据。
96	user_generate_client_key	为用户生成客户端密钥。
97	login_success	成功登录。
98	logout_success	成功注销。

kind_id	操作	描述
99	permanently_delete_tag	永久删除标签。
100	autoprune_tag_delete	自动修剪标签删除。
101	create_namespace_autoprune_policy	创建命名空间自动修剪策略。
102	update_namespace_autoprune_policy	更新命名空间自动修剪策略。
103	delete_namespace_autoprune_policy	删除命名空间自动修剪策略。
104	login_failure	登录尝试失败。

第 7 章 CLAIR 安全扫描程序

7.1. 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。

7.1.1. 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。

7.2. 在独立 RED HAT QUAY 部署中设置 CLAIR

对于独立的 Red Hat Quay 部署，您可以手动设置 Clair。

流程

1. 在 Red Hat Quay 安装目录中，为 Clair 数据库数据创建一个新目录：

```
$ mkdir /home/<user-name>/quay-poc/postgres-clairv4
```

2. 输入以下命令为 **postgres-clairv4** 文件设置适当的权限：

```
$ setfacl -m u:26:-wx /home/<user-name>/quay-poc/postgres-clairv4
```

3. 输入以下命令部署 Clair PostgreSQL 数据库：

```
$ sudo podman run -d --name postgresql-clairv4 \
  -e POSTGRESQL_USER=clairuser \
  -e POSTGRESQL_PASSWORD=clairpass \
  -e POSTGRESQL_DATABASE=clair \
  -e POSTGRESQL_ADMIN_PASSWORD=adminpass \
  -p 5433:5432 \
  -v /home/<user-name>/quay-poc/postgres-clairv4:/var/lib/pgsql/data:Z \
  registry.redhat.io/rhel8/postgresql-13:1-109
```

4. 为您的 Clair 部署安装 PostgreSQL **uuid-ossp** 模块：

```
$ podman exec -it postgresql-clairv4 /bin/bash -c 'echo "CREATE EXTENSION IF NOT EXISTS \"uuid-ossp\"" | psql -d clair -U postgres'
```

输出示例

```
CREATE EXTENSION
```

注意

Clair 要求将 **uuid-ossp** 扩展添加到其 PostgreSQL 数据库中。对于具有适当特权的用户，创建扩展将由 Clair 自动添加。如果用户没有正确的特权，必须在启动 Clair 前添加扩展。

如果扩展不存在，在 Clair 尝试启动时会显示以下错误： **ERROR: 载入 "uuid-ossp" 扩展。(SQLSTATE 42501)**。

5. 如果 **Quay** 容器正在运行，并在配置模式中重启它，将现有配置载入为卷：

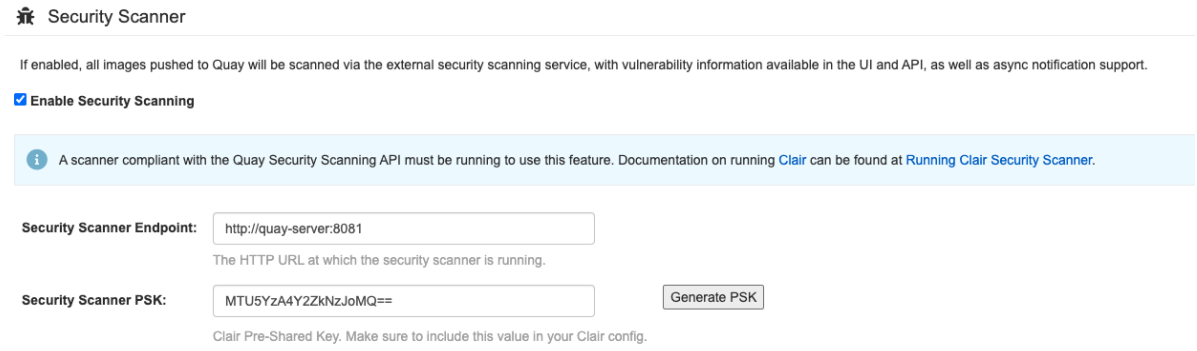
```
$ sudo podman run --rm -it --name quay_config \
  -p 80:8080 -p 443:8443 \
  -v $QUAY/config:/conf/stack:Z \
  {productrepo}/{quayimage}:{productminv} config secret
```

6. 登录到配置工具，再点 UI 的 **Security Scanner** 部分中的 **Enable Security Scanning**。

7. 使用 **quay-server** 系统上尚未使用的端口，为 Clair 设置 HTTP 端点，例如 **8081**。

8. 使用 **Generate PSK** 按钮创建一个预共享密钥(PSK)。

安全扫描器 UI



9. 验证并下载 Red Hat Quay 的 **config.yaml** 文件，然后停止运行配置编辑器的 **Quay** 容器。
10. 将新配置捆绑包提取到 Red Hat Quay 安装目录中，例如：

```
$ tar xvf quay-config.tar.gz -d /home/<user-name>/quay-poc/
```

11. 为您的 Clair 配置文件创建一个文件夹，例如：

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

12. 进入 Clair 配置文件夹：

```
$ cd /etc/opt/clairv4/config/
```

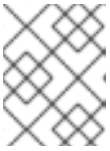
13. 创建 Clair 配置文件，例如：

```
http_listen_addr: :8081
introspection_addr: :8088
log_level: debug
indexer:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
matcher:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  max_conn_pool: 100
  migrations: true
  indexer_addr: clair-indexer
notifier:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  delivery_interval: 1m
  poll_interval: 5m
  migrations: true
auth:
  psk:
    key: "MTU5YzA4Y2ZkNzJoMQ=="
    iss: ["quay"]
# tracing and metrics
trace:
  name: "jaeger"
  probability: 1
  jaeger:
    agent:
      endpoint: "localhost:6831"
    service_name: "clair"
metrics:
  name: "prometheus"
```

有关 Clair 配置格式的更多信息，请参阅 Clair 配置参考。

14. 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 -d --name clairv4 \
-p 8081:8081 -p 8088:8088 \
-e CLAIR_CONF=/clair/config.yaml \
-e CLAIR_MODE=combo \
-v /etc/opt/clairv4/config:/clair:Z \
registry.redhat.io/quay/clair-rhel8:v3.11.1
```



注意

也可以运行多个 Clair 容器，但为了在单一容器之外部署场景，强烈建议使用 Kubernetes 或 OpenShift Container Platform 等容器编配器。

7.3. 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。

7.4. 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：

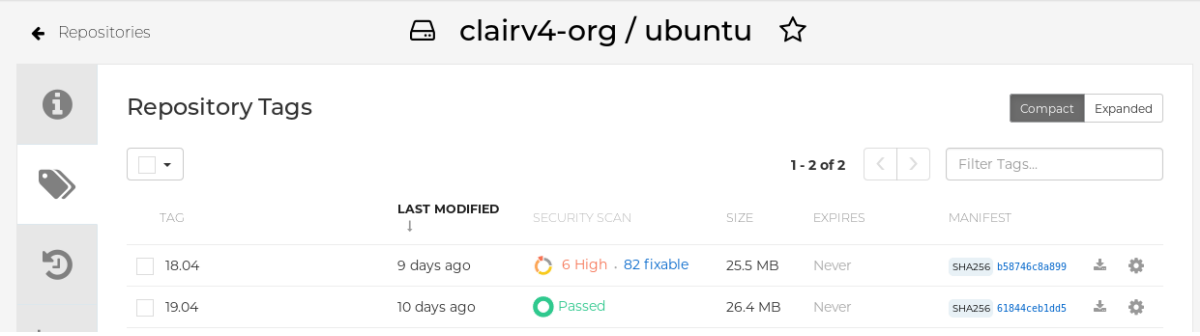
```
$ sudo podman tag docker.io/library/ubuntu:20.04 <quay-server.example.com>/<user-name>/ubuntu:20.04
```

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
```

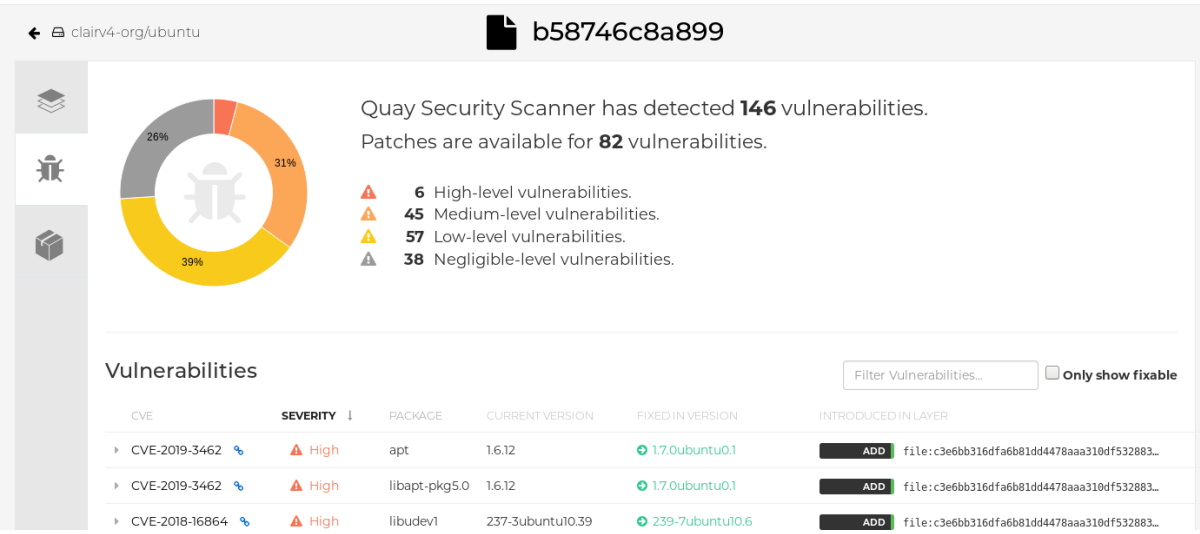
4. 通过 UI 登录您的 Red Hat Quay 部署。
5. 单击存储库名称，如 **quayadmin/ubuntu**。
6. 在导航窗格中，单击 **Tags**。

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。

第 8 章 存储库镜像

8.1. 存储库镜像

Red Hat Quay 存储库镜像可让您将外部容器 registry 或另一个本地 registry 的镜像 mirror 到 Red Hat Quay 集群。使用存储库镜像，您可以根据存储库名称和标签将镜像同步到 Red Hat Quay。

在启用了存储库镜像的 Red Hat Quay 集群中，您可以执行以下操作：

- 从外部 registry 中选择一个仓库(mirror)
- 添加用于访问外部 registry 的凭证
- 识别要同步的特定容器镜像存储库名称和标签
- 设置同步存储库的间隔
- 检查同步的当前状态

要使用镜像功能，您需要执行以下操作：

- 在 Red Hat Quay 配置文件中启用存储库镜像
- 运行存储库镜像 worker
- 创建镜像的存储库

所有存储库镜像配置都可以使用配置工具 UI 或 Red Hat Quay API 执行。

8.2. 与 GEO-REPLICATION 相比的存储库镜像

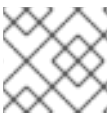
Red Hat Quay geo-replication 镜像在 2 个或更多不同存储后端间的整个镜像存储后端数据，而数据库是共享的一个 Red Hat Quay registry，它有两个不同的 blob 存储端点。地理复制的主要用例包括：

- 为地理分布的设置加快对二进制 blob 的访问
- 确保镜像内容在区域之间相同

存储库镜像将所选存储库或存储库的子集与另一个 registry 同步。registry 有所不同，每个 registry 都有单独的数据库和单独的镜像存储。

镜像的主要用例包括：

- 不同数据中心或区域中的独立 registry 部署，其中整个内容的某些子集应该在数据中心和区域间共享
- 从外部 registry 自动同步或镜像（允许列表）上游存储库到本地 Red Hat Quay 部署



注意

存储库镜像和异地复制可以同时使用。

表 8.1. Red Hat Quay 存储库镜像和异地复制比较

功能/能力	geo-replication	存储库镜像
设计的功能是什么？	共享、全局 registry	不同的 registry
如果复制或镜像还没有完成，会发生什么？	使用远程副本(slower)	未提供镜像
是否可以访问两个区域所需的所有存储后端？	是（所有 Red Hat Quay 节点）	no (distinct storage)
用户是否可以将镜像从两个站点推送到同一存储库？	是	否
所有 registry 内容和配置是否适用于所有区域（共享数据库）？	是	否
用户是否可以选择要镜像的独立命名空间或存储库？	否	是
用户能否将过滤器应用到同步规则？	否	是
每个区域是否允许独立/不同的角色访问控制配置	否	是

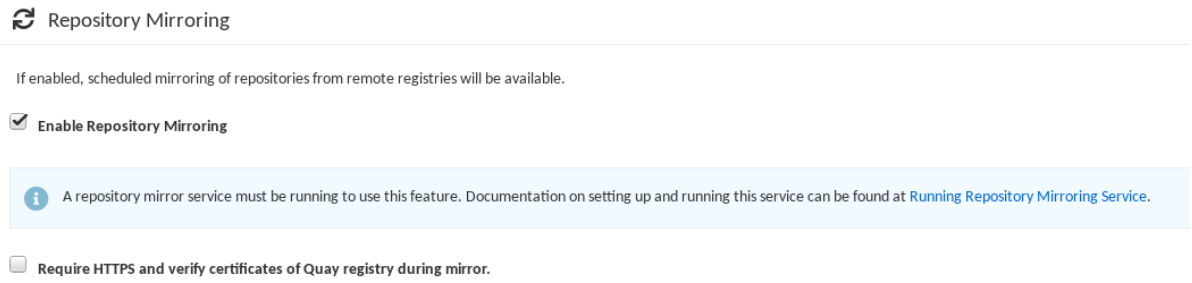
8.3. 使用存储库镜像

以下列表显示了 Red Hat Quay 存储库镜像的功能和限制：

- 使用存储库镜像，您可以镜像整个存储库或有选择限制哪些镜像同步。过滤器可以基于以逗号分隔的标签列表、一系列标签或其他通过 Unix shell 样式通配符识别标签的方法。如需更多信息，请参阅 通配符 的文档。
- 当将存储库设置为镜像时，您无法手动将其他镜像添加到该存储库中。
- 由于已镜像的存储库和标签基于您设置的存储库和标签，它只会保存由存储库和标签对表示的内容。例如，如果您更改了标签，使存储库中的一些镜像不再匹配，则这些镜像将被删除。
- 只有指定的机器人可以将镜像推送到已镜像的存储库，替换存储库上设置的任何基于角色的访问控制权限。
- 镜像可以配置为失败时回滚，*或者以* 最佳方式运行。
- 使用已镜像的存储库时，*具有读取权限* 的用户可以从存储库中拉取镜像，但不能将镜像推送到存储库。
- 在镜像的存储库上更改设置，可以在 Red Hat Quay 用户界面中使用您创建的已镜像存储库的 **Repositories → Mirrors** 选项卡执行。
- 镜像以设定间隔同步，但也可以按需同步。

8.4. 镜像配置 UI

1. 以配置模式启动 **Quay** 容器，再选中 Enable Repository Mirroring 复选框。如果您需要在镜像过程中需要 HTTPS 通信并验证证书，请选择 HTTPS 和证书验证复选框。



2. 验证并下载 **配置文件**，然后使用更新的配置文件在 registry 模式中重启 Quay。

8.5. 镜像配置字段

表 8.2. 镜像配置

字段	类型	描述
FEATURE_REPO_MIRROR	布尔值	启用或禁用存储库镜像 **默认值： false**
REPO_MIRROR_INTERVAL	Number	检查存储库镜像候选者之间的秒数 **默认值：30**
REPO_MIRROR_SERVER_HOSTNAME	字符串	将 **SERVER_HOSTNAME** 替换为镜像的目标。 **Default:** None **Example:** **openshift-quay-service**
REPO_MIRROR_TLS_VERIFY	布尔值	在镜像过程中需要 HTTPS 并验证 Quay registry 的证书。 **默认： false**
REPO_MIRROR_ROLLBACK	布尔值	当设置为 **true** 时，存储库会在镜像失败后回滚。 **默认 ：false**

8.6. 镜像 WORKER

使用以下步骤启动存储库镜像 worker。

流程

流程

- 如果您还没有使用 **/root/ca.crt** 证书配置 TLS 通信，请输入以下命令启动带有 **repomirror** 选项的 **Quay** pod：

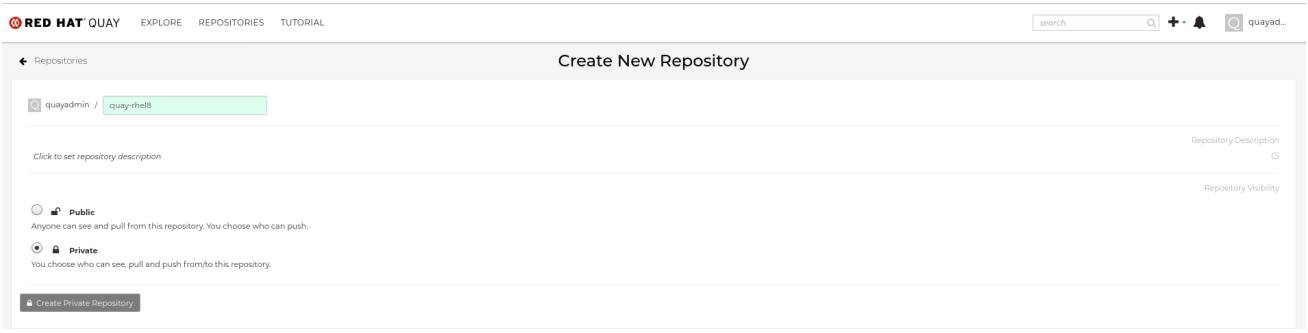
```
$ sudo podman run -d --name mirroring-worker \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1 repomirror
```

- 如果您使用 **/root/ca.crt** 证书配置了 TLS 通信，请输入以下命令启动存储库镜像 worker：

```
$ sudo podman run -d --name mirroring-worker \
  -v $QUAY/config:/conf/stack:Z \
  -v /root/ca.crt:/etc/pki/ca-trust/source/anchors/ca.crt:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1 repomirror
```

8.7. 创建已镜像的存储库

当从外部容器 registry 镜像存储库时，必须创建新的私有存储库。通常，与目标存储库相同的名称，例如 **quay-rhel8**。



8.7.1. 存储库镜像设置

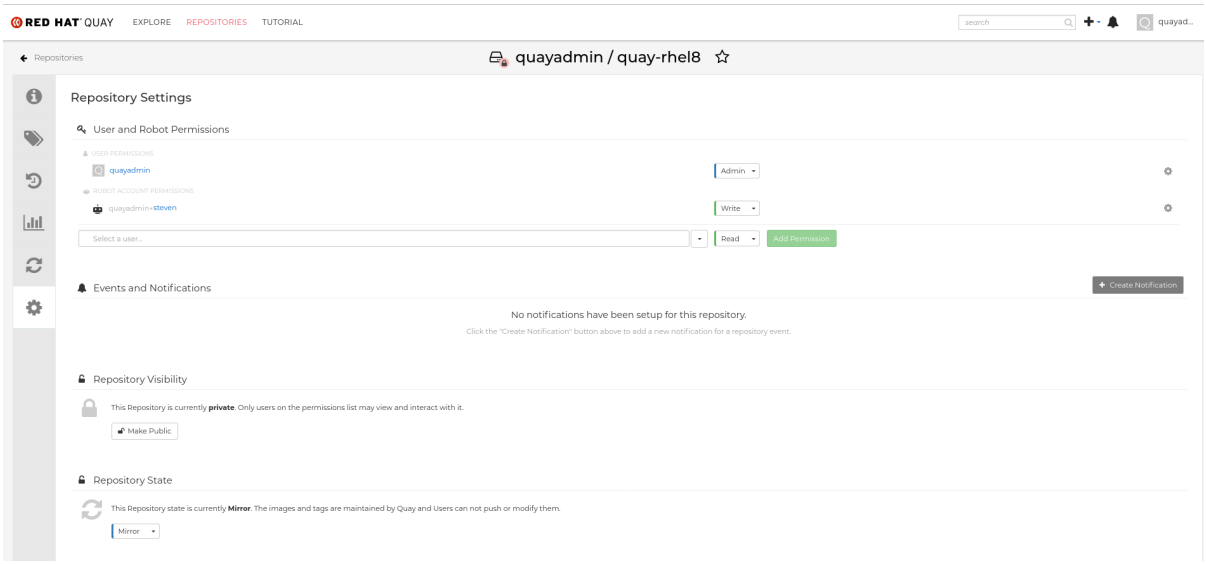
使用以下步骤调整镜像存储库的设置。

前提条件

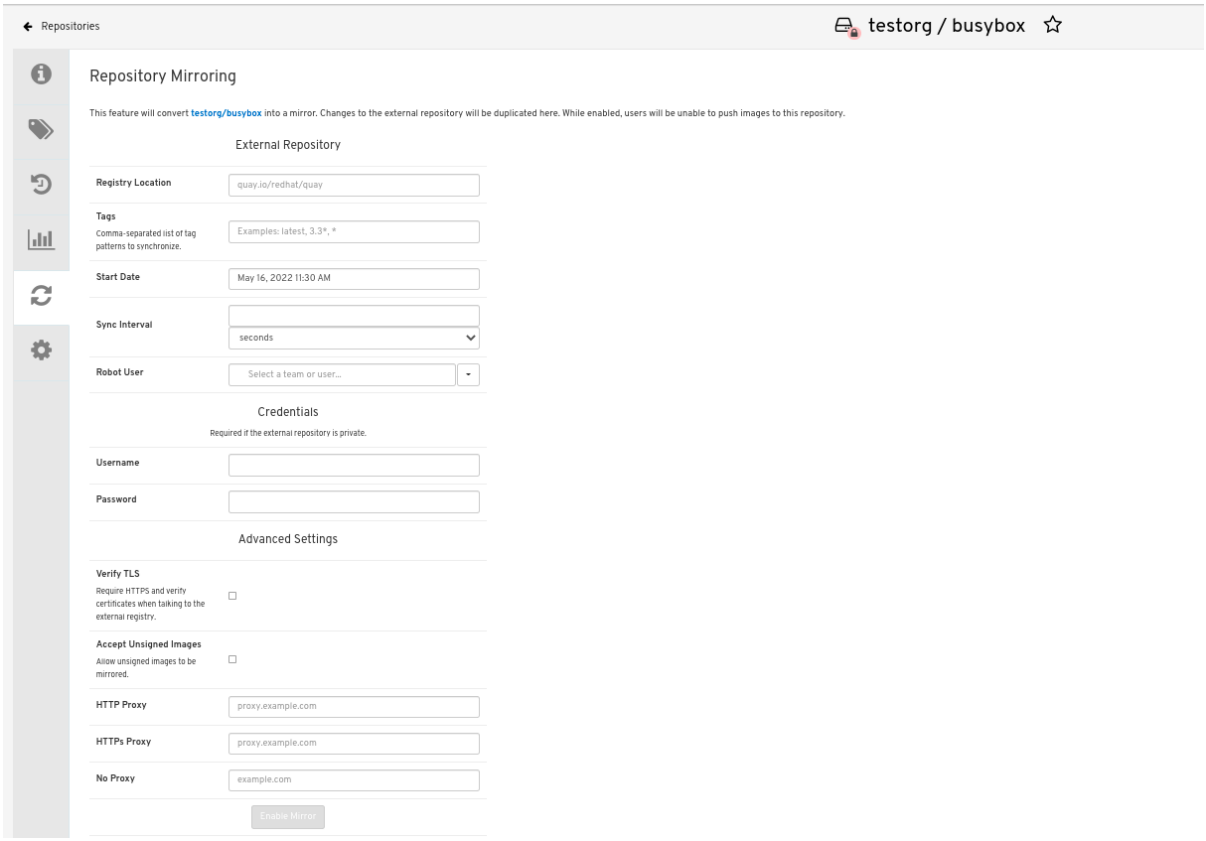
- 您已在 Red Hat Quay 配置文件中启用了存储库镜像。
- 您已部署了镜像 worker。

流程

1. 在 Settings 选项卡中，将 Repository State 设置为 **Mirror** ：



2. 在 Mirror 选项卡中，输入连接到外部 registry 的详情，以及标签、调度和访问信息：



3. 在以下字段中输入详情：

- registry **Location:** 要镜像的外部存储库，如 **registry.redhat.io/quay/quay-rhel8**
- **tags：** 此字段是必需的。您可以输入单个标签或标签模式的逗号分隔列表。（详细信息，请参阅 *Tag Patterns* 部分。）
- **Start Date：** 镜像开始的日期。默认使用当前日期和时间。
- **Sync Interval：** 默认为每 24 小时同步一次。您可以根据小时或天更改。
- **机器人用户：** 创建新的机器人帐户或选择现有的机器人帐户来进行镜像。
- **用户名 ：** 用于访问包含您要镜像存储库的外部 registry 的用户名。

- **Password** ： 与 Username 关联的密码。请注意，密码不能包含需要转义字符(\)的字符。

8.7.2. 高级设置

在 **Advanced Settings** 部分中，您可以使用以下选项配置 SSL/TLS 和代理：

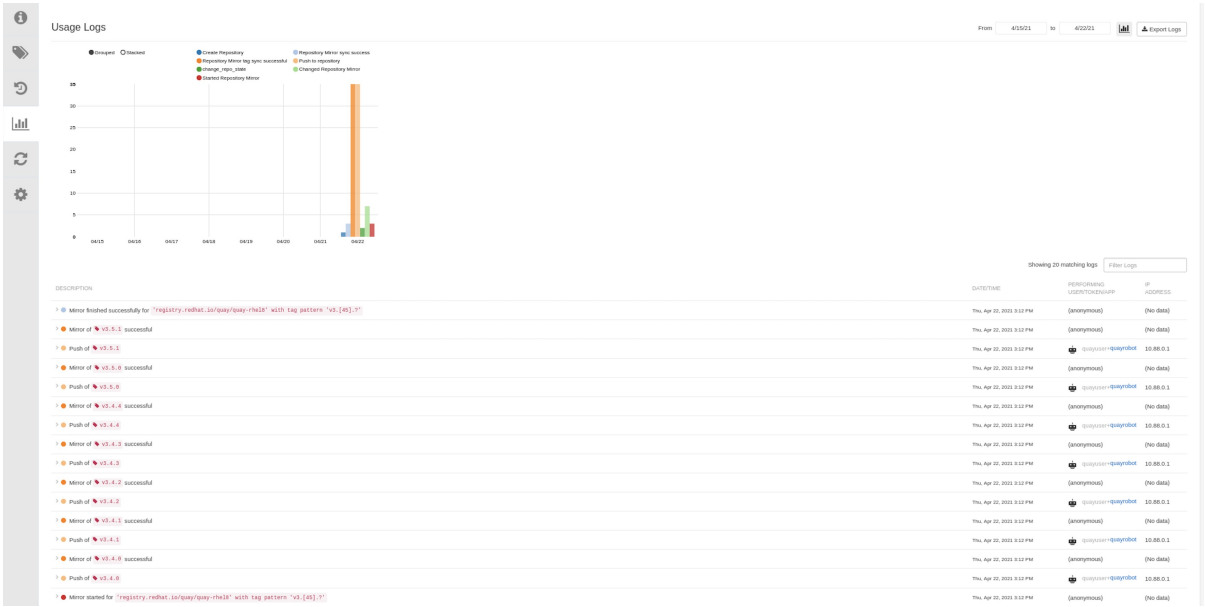
- **验证 TLS** ： 如果您需要 HTTPS 并在与目标远程 registry 通信时验证证书，请选择这个选项。
- **接受 Unsigned Images** ： 选择此选项允许镜像未签名的镜像。
- **HTTP Proxy**： 如果您需要 HTTPS，并在与目标远程 registry 通信时验证证书，请选择这个选项。
- **HTTPS PROXY**： 识别访问远程站点所需的 HTTPS 代理服务器（如果需要代理服务器）。
- **no Proxy**： 不需要代理的位置列表。

8.7.3. 现在同步

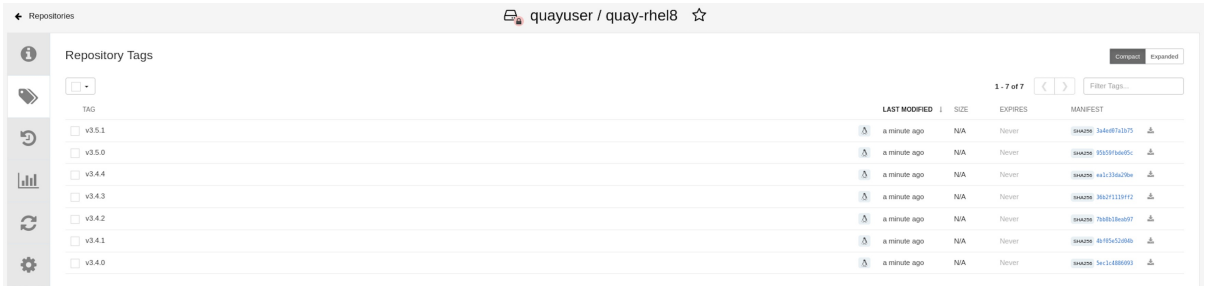
使用以下步骤启动镜像操作。

流程

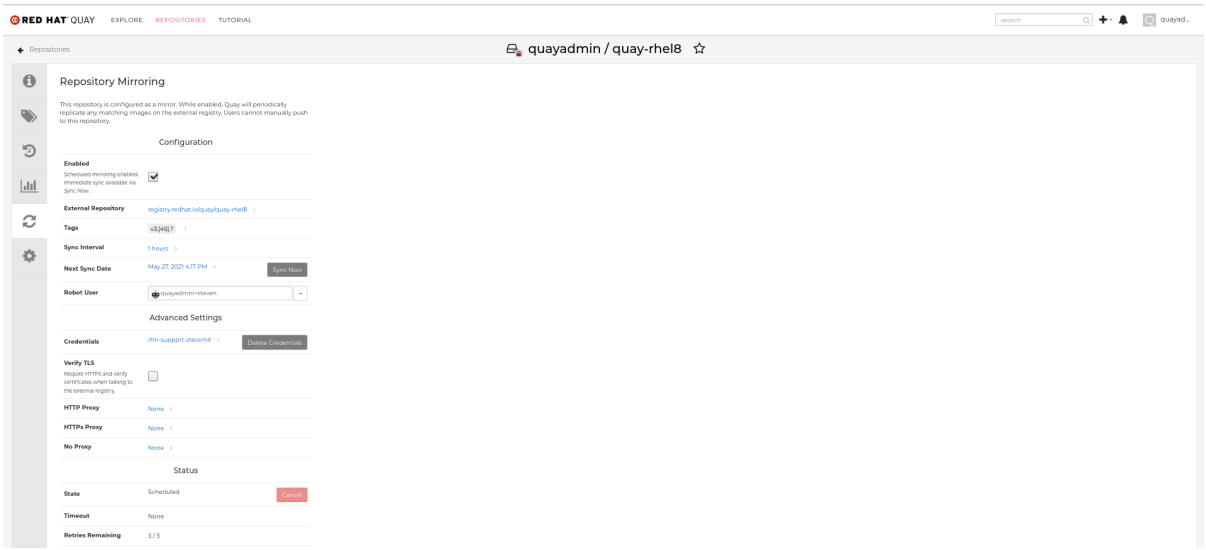
- 要执行即时镜像操作，请在存储库镜像选项卡中按立即同步按钮。日志在 Usage Logs 选项卡中提供：



镜像完成后，镜像将显示在标签选项卡中：



以下是已完成的存储库镜像屏幕的示例：



8.8. 镜像的事件通知

存储库镜像有三个通知事件：

- 仓库镜像已启动
- 仓库镜像成功
- 仓库 Mirror Unsuccessful

事件可以在每个存储库的 **Settings** 选项卡内配置，并且支持电子邮件、Slack、Quay UI 和 Webhook 等所有现有通知方法。

8.9. 镜像标签模式

必须至少输入一个标签。下表引用可能的镜像标签模式。

8.9.1. 特征语法

pattern	描述
*	匹配所有字符
?	匹配任何单个字符
[seq]	匹配 *seq* 中的任何字符
[!seq]	匹配没有在 *seq* 中的任何字符

8.9.2. 标签模式示例

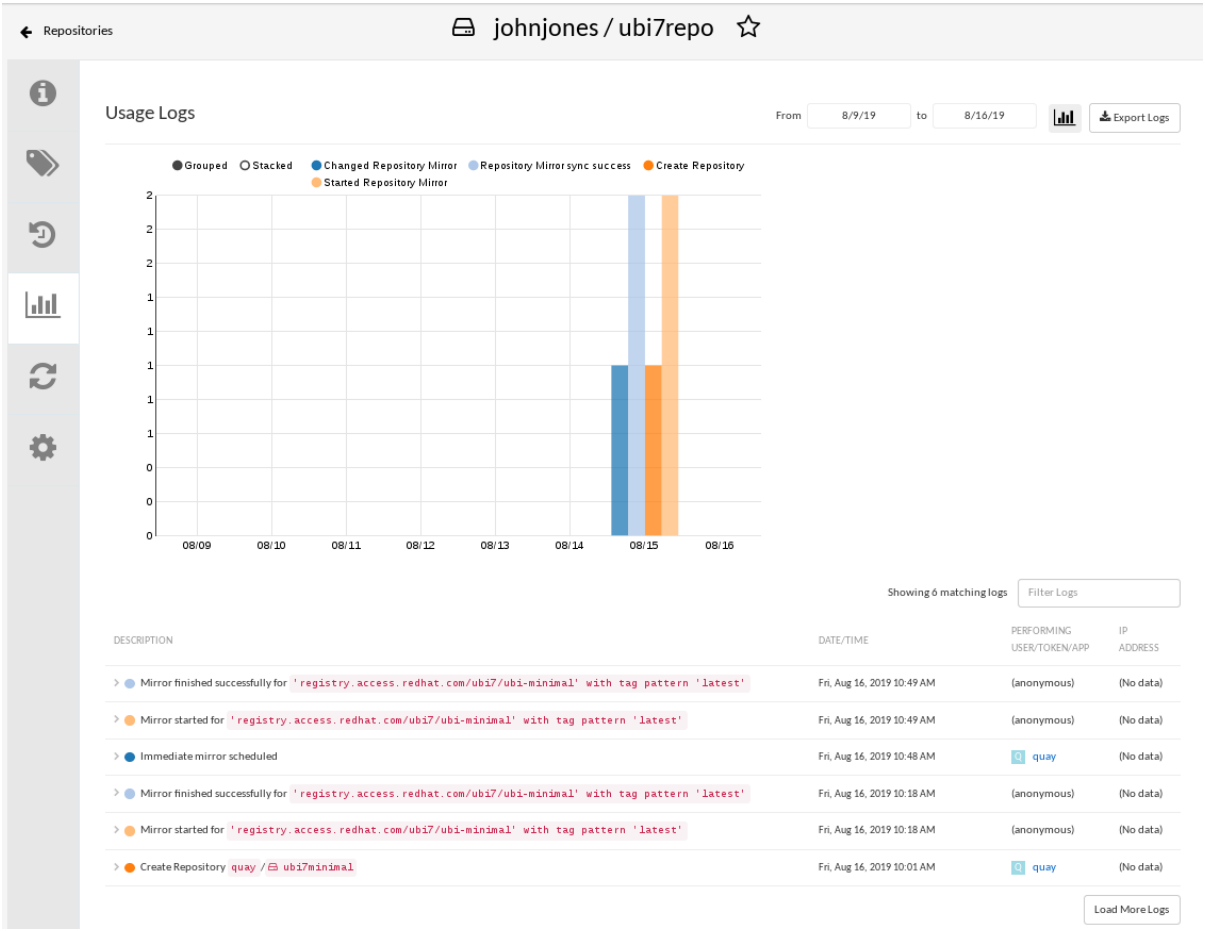
Pattern 示例	Matches 示例
v3*	v32, v3.1, v3.2, v3.2-4beta, v3.3

v3.*	v3.1, v3.2, v3.2-4beta
v3.?	v3.1, v3.2, v3.3
v3.[12]	v3.1, v3.2
v3.[12]*	v3.1, v3.2, v3.2-4beta
v3.[!1]*	v3.2, v3.2-4beta, v3.3

8.10. 使用镜像软件仓库

创建已镜像的存储库后，可以通过多种方式来使用该存储库。从 Repositories 页面中选择您镜像的存储库，并执行以下操作：

- **启用/禁用存储库** ：选择左侧列中的镜像按钮，然后切换 Enabled 复选框来临时启用或禁用存储库。
- **检查镜像日志** ：为确保已镜像存储库正常工作，您可以检查镜像日志。为此，请在左侧列中选择 Usage Logs 按钮。例如：



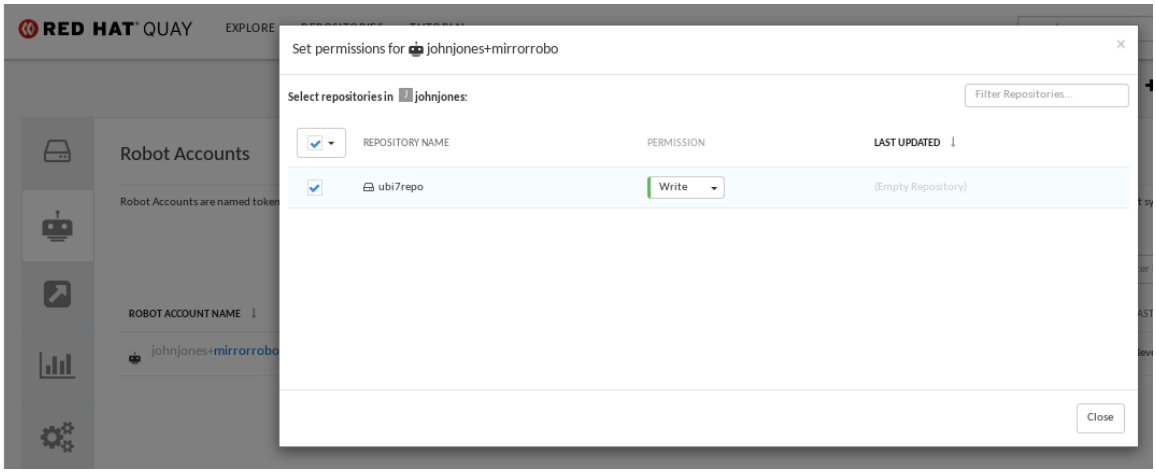
- **现在** 同步镜像： 要立即同步存储库中的镜像，请选择立即同步按钮。
- **更改凭证** ：要更改用户名和密码，请从 Credentials 行选择 DELETE。然后，选择 None 并添加在提示时登录到外部 registry 所需的用户名和密码。

- **取消镜像** ：要停止镜像(mirror)，这会保持当前镜像可用，但停止新的镜像同步，请选择 CANCEL 按钮。

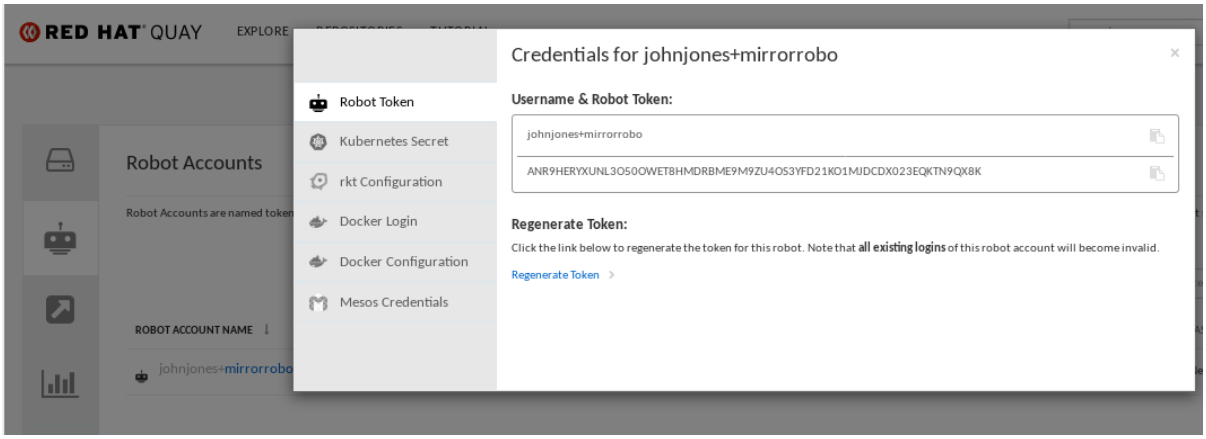
- **设置机器人权限** ：Red Hat Quay 机器人帐户命名为 token，其中包含用于访问外部存储库的凭证。通过为机器人分配凭据，该机器人可用于需要访问同一外部 registry 的多个镜像仓库。您可以通过转至帐户设置，然后选择左列中的 Robot Accounts 图标，将现有的机器人分配给存储库。对于机器人帐户，请选择 REPOSITORIES 列下的链接。在弹出窗口中，您可以：

  - 检查哪些存储库被分配给该机器人。

  - 从图中所示的 PERMISSION 字段为该机器人分配读取、写入或管理员权限：



- **更改机器人凭证** ：Robots 可以保存凭证，如 Kubernetes secret、Docker 登录信息和 Mesos 捆绑包。要更改机器人凭证，请在 Robot Accounts 窗口中选择机器人帐户行上的 Options gear，然后选择 View Credentials。为机器人需要访问的外部存储库添加适当的凭据。



- **检查并更改常规设置** ：从已镜像仓库页面的左列中选择 Settings 按钮(gear 图标)。在生成的页面中，您可以更改与已镜像仓库关联的设置。特别是，您可以更改 User 和 Robot Permissions，以指定哪些用户和机器人可以从存储库读取或写入到存储库。

8.11. 存储库镜像建议

存储库镜像的最佳实践包括：

- 存储库镜像 pod 可以在任何节点上运行。这意味着您可以在已运行 Red Hat Quay 的节点上运行镜像功能。

- 存储库镜像在数据库中调度，并批量运行。因此，存储库 worker 会检查每个存储库镜像配置文件，并在需要下一次同步时读取。更多镜像 worker 意味着可以同时镜像更多存储库。例如，运行 10 镜像 worker 意味着用户可以并行运行 10 个镜像 Operator。如果用户只有 2 个带有 10 个镜像配置的 worker，则只能执行 2 个 operator。

- 镜像 pod 的最佳数量取决于以下条件：
  - 要镜像的存储库总数
  - 存储库中的镜像和标签数量以及更改的频率
  - 并行批处理
    例如，如果用户镜像有 100 标签的存储库，则镜像将由一个 worker 完成。用户必须考虑要并行镜像多少个仓库，并基于 worker 的数量。

    同一存储库中的多个标签无法并行镜像。

第 9 章 IPV6 和双栈部署

您的独立 Red Hat Quay 部署现在可以在只支持 IPv6 的位置提供，如 Telco 和 Edge 环境。也支持双栈网络，以便 Red Hat Quay 部署可以同时侦听 IPv4 和 IPv6。

有关已知限制列表的信息，请参阅 IPv6 限制

9.1. 启用 IPV6 协议系列

使用以下步骤在独立 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，且不会被 ipv6-limitations[current limitations] 覆盖。

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。

9.2. 启用双栈协议系列

使用以下步骤在独立 Red Hat Quay 部署上启用双栈(IPv4 和 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 **双栈**，例如：

```
---
FEATURE_GOOGLE_LOGIN: false
FEATURE_INVITE_ONLY_USER_CREATION: false
FEATURE_LISTEN_IP_VERSION: dual-stack
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在侦听这两个频道：
   a. 对于 IPv4，输入以下命令：

```
$ curl --ipv4 <quay_endpoint>
{"data":{"services":
{"auth":true,"database":true,"disk_space":true,"registry_gunicorn":true,"service_key":true,"
web_gunicorn":true}},"status_code":200}
```

   b. 对于 IPv6，输入以下命令：

```
$ curl --ipv6 <quay_endpoint>
{"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** 中启用双栈后，所有 Red Hat Quay 功能都可以正常使用，只要您的环境是为双栈配置的。

9.3. IPV6 和 DUA-STACK 限制

- 目前，尝试使用通用 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 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。

第 10 章 RED HAT QUAY 的 LDAP 身份验证设置

轻量级目录访问协议(LDAP)是一个开放的、厂商中立的行业标准应用程序协议，用于通过互联网协议(IP)网络访问和维护分布式目录信息服务。Red Hat Quay 支持使用 LDAP 作为身份提供程序。

10.1. 启用 LDAP 时的注意事项

在为 Red Hat Quay 部署启用 LDAP 之前，您应该考虑以下内容：

现有的 Red Hat Quay 部署
当您为已经配置了用户的现有 Red Hat Quay 部署启用 LDAP 时，用户名之间可能会出现冲突。例如，在启用 LDAP 之前，在 Red Hat Quay 中手动创建了一个用户 **alice**。如果 LDAP 目录中也存在 username **alice**，当 alice 第一次使用 LDAP 登录时，Red Hat Quay 会自动创建一个新的用户 **alice-1**。Red Hat Quay 随后会自动将 LDAP 凭据映射到 **alice** 帐户。出于一致性的原因，您的 Red Hat Quay 部署可能会出现错误。建议您在启用 LDAP 之前从 Red Hat Quay 中删除任何可能冲突的本地帐户名称。

手动创建用户和 LDAP 身份验证
当为 LDAP 配置 Red Hat Quay 时，如果配置选项 **FEATURE_USER_CREATION** 设置为 **true**，则在第一次登录时会在 Red Hat Quay 的数据库中自动创建 LDAP 验证的用户。如果此选项设为 **false**，则 LDAP 用户自动创建用户会失败，并且不允许该用户登录。在这种情况下，超级用户需要首先创建所需的用户帐户。相反，如果将 **FEATURE_USER_CREATION** 设置为 **true**，这也意味着用户仍然可以从 Red Hat Quay 登录屏幕创建一个帐户，即使 LDAP 中存在对等用户。

10.2. 为 RED HAT QUAY 配置 LDAP

您可以通过直接更新 **config.yaml** 文件并重启部署来为 Red Hat Quay 配置 LDAP。在为 Red Hat Quay 配置 LDAP 时，请使用以下步骤作为参考。

1. 直接更新 **config.yaml** 文件，使其包含以下相关信息：

```
# ...
AUTHENTICATION_TYPE: LDAP (1)
# ...
LDAP_ADMIN_DN: uid=<name>,ou=Users,o=<organization_id>,dc=
<example_domain_component>,dc=com (2)
LDAP_ADMIN_PASSWD: ABC123 (3)
LDAP_ALLOW_INSECURE_FALLBACK: false (4)
LDAP_BASE_DN: (5)
   - o=<organization_id>
   - dc=<example_domain_component>
   - dc=com
LDAP_EMAIL_ATTR: mail (6)
LDAP_UID_ATTR: uid (7)
LDAP_URI: ldap://<example_url>.com (8)
LDAP_USER_FILTER: (memberof=cn=developers,ou=Users,dc=<domain_name>,dc=com)
(9)
LDAP_USER_RDN: (10)
   - ou=<example_organization_unit>
   - o=<organization_id>
   - dc=<example_domain_component>
   - dc=com
# ...
```

1. 必需。必须设置为 **LDAP**。
2. 必需。用于 LDAP 验证的管理 DN。
3. 必需。LDAP 身份验证的 admin 密码。
4. 必需。是否允许 SSL/TLS 不安全的 LDAP 身份验证回退。
5. 必需。LDAP 身份验证的基本 DN。
6. 必需。LDAP 身份验证的电子邮件属性。
7. 必需。LDAP 身份验证的 UID 属性。
8. 必需。LDAP URI。
9. 必需。LDAP 身份验证的用户过滤器。
10. 必需。用于 LDAP 身份验证的用户 RDN。

2. 添加所有所需的 LDAP 字段后，保存更改并重启 Red Hat Quay 部署。

10.3. 启用 LDAP_RESTRICTED_USER_FILTER 配置字段

LDAP_RESTRICTED_USER_FILTER 配置字段是 **LDAP_USER_FILTER** 配置字段的子集。配置后，此选项允许 Red Hat Quay 管理员在 Red Hat Quay 使用 LDAP 作为其身份验证提供程序时，将 LDAP 用户配置为受限用户。

使用以下步骤在 Red Hat Quay 部署中启用 LDAP 受限用户。

先决条件

- 您的 Red Hat Quay 部署使用 LDAP 作为其身份验证提供程序。
- 您已在 **config.yaml** 文件中配置了 **LDAP_USER_FILTER** 字段。

流程

1. 在部署的 **config.yaml** 文件中，添加 **LDAP_RESTRICTED_USER_FILTER** 参数并指定受限用户组，例如：**members**：

```
# ...
AUTHENTICATION_TYPE: LDAP
# ...
LDAP_ADMIN_DN: uid=<name>,ou=Users,o=<organization_id>,dc=
<example_domain_component>,dc=com
LDAP_ADMIN_PASSWD: ABC123
LDAP_ALLOW_INSECURE_FALLBACK: false
LDAP_BASE_DN:
    - o=<organization_id>
    - dc=<example_domain_component>
    - dc=com
LDAP_EMAIL_ATTR: mail
LDAP_UID_ATTR: uid
LDAP_URI: ldap://<example_url>.com
```

```
LDAP_USER_FILTER: (memberof=cn=developers,ou=Users,o=
<example_organization_unit>,dc=<example_domain_component>,dc=com)
LDAP_RESTRICTED_USER_FILTER: (<filterField>=<value>) (1)
LDAP_USER_RDN:
    - ou=<example_organization_unit>
    - o=<organization_id>
    - dc=<example_domain_component>
    - dc=com
# ...
```

1 将指定用户为受限用户。

2. 启动或重启您的 Red Hat Quay 部署。

启用 **LDAP_RESTRICTED_USER_FILTER** 功能后，您的 LDAP Red Hat Quay 用户会被限制为不能读取和写入内容并创建机构。

10.4. 启用 LDAP_SUPERUSER_FILTER 配置字段

配置 **LDAP_SUPERUSER_FILTER** 字段后，如果 Red Hat Quay 使用 LDAP 作为其身份验证提供程序，Red Hat Quay 管理员可以将轻量级目录访问协议(LDAP)用户配置为超级用户。

使用以下步骤在 Red Hat Quay 部署中启用 LDAP 超级用户。

先决条件

- 您的 Red Hat Quay 部署使用 LDAP 作为其身份验证提供程序。
- 您已在 **config.yaml** 文件中配置了 **LDAP_USER_FILTER** 字段。

流程

1. 在部署的 **config.yaml** 文件中，添加 **LDAP_SUPERUSER_FILTER** 参数，并添加您要配置为超级用户的用户组，例如 **root** ：

```
# ...
AUTHENTICATION_TYPE: LDAP
# ...
LDAP_ADMIN_DN: uid=<name>,ou=Users,o=<organization_id>,dc=
<example_domain_component>,dc=com
LDAP_ADMIN_PASSWD: ABC123
LDAP_ALLOW_INSECURE_FALLBACK: false
LDAP_BASE_DN:
    - o=<organization_id>
    - dc=<example_domain_component>
    - dc=com
LDAP_EMAIL_ATTR: mail
LDAP_UID_ATTR: uid
LDAP_URI: ldap://<example_url>.com
LDAP_USER_FILTER: (memberof=cn=developers,ou=Users,o=
<example_organization_unit>,dc=<example_domain_component>,dc=com)
LDAP_SUPERUSER_FILTER: (<filterField>=<value>) (1)
LDAP_USER_RDN:
    - ou=<example_organization_unit>
```

```
    - o=<organization_id>
    - dc=<example_domain_component>
    - dc=com
# ...
```

1 将指定的用户配置为超级用户。

2. 启动或重启您的 Red Hat Quay 部署。

启用 **LDAP_SUPERUSER_FILTER** 功能后，您的 LDAP Red Hat Quay 用户具有超级用户特权。以下选项可供超级用户使用：

- 管理用户
- 管理机构
- 管理服务密钥
- 查看更改日志
- 查询用量日志
- 创建全局可见的用户消息

10.5. 常见 LDAP 配置问题

以下错误可能会返回无效的配置。

- **无效凭据**。如果您收到这个错误，则管理员 DN 或管理员 DN 密码值不正确。确保您提供准确的管理员 DN 和密码值。
- *** 验证超级用户 %USERNAME% 失败**。返回此错误的原因如下：
  - 未找到用户名。
  - 用户在远程身份验证系统中不存在。
  - LDAP 授权配置不正确。
- **无法找到当前登录的用户**。为 Red Hat Quay 配置 LDAP 时，可能会出现使用 **管理员 DN** 字段中提供的用户名和密码成功建立 LDAP 连接的情况。但是，如果无法使用 **UID Attribute** 或 **Mail Attribute** 字段在指定的 **用户 Relative DN** 路径中找到当前登录的用户，则通常有两个潜在的原因：
  - 当前登录的用户没有 **用户 Relative DN** 路径。
  - **管理员 DN** 没有搜索或读取指定的 LDAP 路径的权限。
    要解决这个问题，请确保登录的用户包含在 **用户 Relative DN** 路径中，或者为 **管理员 DN** 帐户提供正确的权限。

10.6. LDAP 配置字段

有关 LDAP 配置字段的完整列表，请参阅 LDAP 配置字段

第 11 章 为 RED HAT QUAY 配置 OIDC

为 Red Hat Quay 配置 OpenID Connect (OIDC)可以为部署提供几个优点。例如，OIDC 允许用户使用来自 OIDC 供应商的现有凭证（如 Red Hat Single Sign-On、Google、Github、Microsoft 等）向 Red Hat Quay 进行身份验证。OIDC 的其他优点包括集中式用户管理、增强安全性和单点登录(SSO)。总体而言，OIDC 配置可以简化用户身份验证和管理，增强安全性，并为 Red Hat Quay 用户提供无缝的用户体验。

以下流程演示了如何在 Red Hat Quay 的独立部署中配置 Microsoft Entra ID，以及如何在基于 Operator 的 Red Hat Quay 部署上配置 Red Hat Single Sign-On。这些流程可根据您的部署类型进行交换。

注意

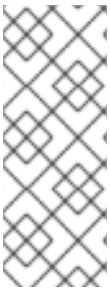
按照以下步骤，无论您选择使用什么身份提供程序，您都可以在 Red Hat Quay 中添加任何 OIDC 供应商。

11.1. 在 RED HAT QUAY 的独立部署中配置 MICROSOFT ENTRA ID OIDC

通过将 Microsoft Entra ID 身份验证与 Red Hat Quay 集成，您的组织可以利用 Microsoft Entra ID 提供的集中式用户管理和安全功能。某些功能包括根据 Microsoft Entra ID 角色和权限管理用户对 Red Hat Quay 存储库的访问权限，以及启用由 Microsoft Entra ID 提供的多因素身份验证和其他安全功能的功能。

Red Hat Quay 的 Azure Active Directory (Microsoft Entra ID)身份验证允许用户使用 Microsoft Entra ID 凭证验证和访问 Red Hat Quay。

使用以下步骤通过直接更新 Red Hat Quay **config.yaml** 文件来配置 Microsoft Entra ID。



流程

- 使用以下步骤，您可以在 Red Hat Quay 中添加任何 ODIC 供应商，而不考虑正在添加哪些身份提供程序。
- 如果您的系统启用了防火墙，或者启用了代理，则必须将创建的每个 Oauth 应用程序的所有 Azure API 端点列入白名单。否则，会返回以下错误： **x509: certificate signed by unknown authority**。

1. 使用以下引用并更新 **config.yaml** 文件，使用您需要的 OIDC 供应商凭证更新您的 config.yaml 文件：

```
# ...
AZURE_LOGIN_CONFIG: 1
  CLIENT_ID: <client_id> 2
  CLIENT_SECRET: <client_secret> 3
  OIDC_SERVER: <oidc_server_address_> 4
  SERVICE_NAME: Microsoft Entra ID 5
  VERIFIED_EMAIL_CLAIM_NAME: <verified_email> 6
# ...
```

1 包含 OIDC 配置设置的父键。在本例中，所用的父密钥是 **AZURE_LOGIN_CONFIG**，但字符串 **AZURE** 可以根据您的特定需求替换任意任意字符串，如 **ABC123**。但是，以下字符串不被接受： **GOOGLE,GITHUB**。这些字符串为其对应的身份平台保留，且需要在您使用的平台时进行特定的 **config.yaml** 条目。

2. 与身份提供程序一起注册的应用程序的客户端 ID。
3. 与身份提供程序一起注册的应用的客户端机密。
4. 用于身份验证的 OIDC 服务器地址。在本例中，您必须使用 **sts.windows.net** 作为签发者标识符。使用 **https://login.microsoftonline.com** 会导致以下错误： **Could not create provider for AzureAD。error: oidc: issuer 与供应商返回的签发者不匹配，预期的 "https://login.microsoftonline.com/73f2e714-xxxx-xxxx-xxxx-dffe1df8a5d5" got "https://sts.windows.net/73f2e714-xxxx-xxxx-xxxx-dffe1df8a5d5/"**。
5. 正在验证的服务的名称。
6. 用于验证用户电子邮件地址的声明名称。

2. 正确配置 Microsoft Entra ID 结果使用以下格式进行三个重定向：
   - **https://QUAY_HOSTNAME/oauth2/<name_of_service>/callback**
   - **https://QUAY_HOSTNAME/oauth2/<name_of_service>/callback/attach**
   - **https://QUAY_HOSTNAME/oauth2/<name_of_service>/callback/cli**
3. 重启 Red Hat Quay 部署。

11.2. 为 RED HAT QUAY 配置 RED HAT SINGLE SIGN-ON

根据 Keycloak 项目，Red Hat Single Sign-On (RH-SSO)是一个红帽提供的开源身份和访问管理(IAM)解决方案。RH-SSO 允许组织在其系统和应用程序之间管理用户身份、安全应用程序和强制实施访问控制策略。它还提供了一个统一的身份验证和授权框架，允许用户一次登录并获得对多个应用和资源的访问权限，而无需重新验证。如需更多信息，请参阅 Red Hat Single Sign-On 。

通过在 Red Hat Quay 上配置 Red Hat Single Sign-On，您可以在 Red Hat Quay 和其他 OpenShift Container Platform 等应用平台之间创建无缝身份验证集成。

11.2.1. 配置 Red Hat Single Sign-On Operator，以用于 Red Hat Quay Operator

使用以下步骤为 OpenShift Container Platform 上的 Red Hat Quay Operator 配置 Red Hat Single Sign-On。

先决条件

- 您已设置 Red Hat Single Sign-On Operator。如需更多信息，请参阅 Red Hat Single Sign-On Operator。
- 您已在 OpenShift Container Platform *部署和* Red Hat Single Sign-On 上为 Red Hat Quay 配置了 SSL/TLS。
- 您已生成了一个证书颁发机构(CA)，并将其上传到 Red Hat Single Sign-On Operator *和* Red Hat Quay 配置。

流程

1. 导航到 Red Hat Single Sign-On **Admin Console**。
   a. 在 OpenShift Container Platform **Web 控制台中** 导航至 **Network → Route**。

b. 从下拉列表中选择 **Red Hat Single Sign-On**项目。

c. 在 **Routes** 表中查找 Red Hat Single Sign-On **Admin 控制台**。

2. 选择您要用来配置 Red Hat Quay 的 Realm。

3. 单击导航面板的 **Configure** 部分下的 **Clients**，然后单击 **Create** 按钮为 Red Hat Quay 添加新的 OIDC。

4. 输入以下信息。

   - **客户端 ID： quay-enterprise**
   - **客户端协议： openid-connect**
   - **根 URL： https://<quay_endpoint>/**

5. 点击 **Save**。这会导致重定向到 **Clients** 设置面板。

6. 导航到 **Access Type** 并选择 **Confidential**。

7. 导航到 **Valid Redirect URI**。您必须提供三个重定向 URI。该值应该是 Red Hat Quay registry 的完全限定域名，附加了 **/oauth2/redhatsso/callback**。例如：

   - **https://<quay_endpoint>/oauth2/redhatsso/callback**
   - **https://<quay_endpoint>/oauth2/redhatsso/callback/attach**
   - **https://<quay_endpoint>/oauth2/redhatsso/callback/cli**

8. 点 **Save** 并进入新的 **Credentials** 设置。

9. 复制 Secret 的值。

11.2.1.1. 将 Red Hat Quay Operator 配置为使用 Red Hat Single Sign-On

使用以下步骤使用 Red Hat Quay Operator 配置 Red Hat Single Sign-On。

先决条件

- 您已设置 Red Hat Single Sign-On Operator。如需更多信息，请参阅 Red Hat Single Sign-On Operator。
- 您已在 OpenShift Container Platform *部署和* Red Hat Single Sign-On 上为 Red Hat Quay 配置了 SSL/TLS。
- 您已生成了一个证书颁发机构(CA)，并将其上传到 Red Hat Single Sign-On Operator *和* Red Hat Quay 配置。

流程

1. 进入 Operators → **Installed Operators** → Red Hat Quay → **Quay Registry** → **Config Bundle Secret** 来编辑 **Red Hat Quay config.yaml** 文件。然后，点击 **Actions** → **Edit Secret**。另外，您可以在本地更新 **config.yaml** 文件。

2. 在 OpenShift Container Platform **config.yaml** 文件中的 Red Hat Quay 中添加以下信息：

```
# ...
RHSSO_LOGIN_CONFIG: 1
  CLIENT_ID: <client_id> 2
  CLIENT_SECRET: <client_secret> 3
  OIDC_SERVER: <oidc_server_url> 4
  SERVICE_NAME: <service_name> 5
  SERVICE_ICON: <service_icon> 6
  VERIFIED_EMAIL_CLAIM_NAME: <example_email_address> 7
  PREFERRED_USERNAME_CLAIM_NAME: <preferred_username> 8
  LOGIN_SCOPES: 9
    - 'openid'
# ...
```

1. 包含 OIDC 配置设置的父键。在本例中，所用的父密钥是 **AZURE_LOGIN_CONFIG**，但字符串 **AZURE** 可以根据您的特定需求替换任意任意字符串，如 **ABC123**。但是，以下字符串不被接受： **GOOGLE,GITHUB**。这些字符串为其对应的身份平台保留，且需要在您使用的平台时进行特定的 **config.yaml** 条目。
2. 使用身份提供程序注册的应用程序的客户端 ID，如 **quay-enterprise**。
3. **quay-enterprise** OIDC 客户端设置的 **Credentials** 选项卡中的 Client Secret。
4. Red Hat Single Sign-On 实例的完全限定域名(FQDN)，附加 **/auth/realms/** 和 Realm 名称。您必须在末尾包括正斜杠，例如 **https://sso-redhat.example.com//auth/realms/<keycloak_realm_name>/**。
5. 在 Red Hat Quay 登录页面中显示的名称，例如 **Red Hat Single Sign On**。
6. 更改登录屏幕上的图标。例如： **/static/img/RedHat.svg**。
7. 用于验证用户电子邮件地址的声明名称。
8. 用于验证用户电子邮件地址的声明名称。
9. 执行登录流时发送到 OIDC 供应商的范围，例如 **openid**。

3. 在启用了 Red Hat Single Sign-On 的 OpenShift Container Platform 部署中重启 Red Hat Quay。

11.3. RED HAT QUAY OIDC 部署的团队同步

管理员可以利用支持组或团队同步的 OpenID Connect (OIDC)身份提供程序，将存储库权限应用到 Red Hat Quay 中的一组用户。这允许管理员避免在 Red Hat Quay 和 OIDC 组间手动创建和同步组定义。

11.3.1. 为 Red Hat Quay OIDC 部署启用同步

当 Red Hat Quay 部署使用 OIDC 身份验证程序时，请使用以下步骤启用团队同步。

重要

以下流程不使用特定的 OIDC 供应商。相反，它概述了如何在 OIDC 供应商和 Red Hat Quay 之间处理团队同步的最佳方法。任何 OIDC 供应商都可以用来启用团队同步，但设置可能会因您的供应商而异。

流程

1. 使用以下信息更新 **config.yaml** 文件：

```
AUTHENTICATION_TYPE: OIDC
# ...
OIDC_LOGIN_CONFIG:
  CLIENT_ID: 1
  CLIENT_SECRET: 2
  OIDC_SERVER: 3
  SERVICE_NAME: 4
  PREFERRED_GROUP_CLAIM_NAME: 5
  LOGIN_SCOPES: [ 'openid', '<example_scope>' ] 6
  OIDC_DISABLE_USER_ENDPOINT: false 7
# ...
FEATURE_TEAM_SYNCING: true 8
FEATURE_NONSUPERUSER_TEAM_SYNCING_SETUP: true 9
FEATURE_UI_V2: true
# ...
```

1. 必需。此 Red Hat Quay 实例注册的 OIDC 客户端 ID。
2. 必需。此 Red Hat Quay 实例注册的 OIDC 客户端 secret。
3. 必需。用于身份验证的 OIDC 服务器地址。这个 URL 应该是，对 < **OIDC_SERVER>/.well-known/openid-configuration** 的 **GET** 请求会返回供应商的配置信息。此配置信息对于依赖方(RP)与 OpenID Connect 供应商安全交互非常重要，并获取身份验证和授权流程的必要详情。
4. 必需。正在验证的服务的名称。
5. 必需。OIDC 令牌有效负载中的密钥名称，其中包含有关用户组成员资格的信息。此字段允许身份验证系统从 OIDC 令牌中提取组成员资格信息，以便它可用于 Red Hat Quay。
6. 必需。添加 Red Hat Quay 用来与 OIDC 供应商通信的其他范围。必须包括 **'openid'**。其他范围是可选的。
7. 是否允许或禁用 **/userinfo** 端点。如果使用 Azure Entra ID，请将此字段设置为 **true**。默认值为 **false**。
8. 必需。是否允许团队成员资格从身份验证引擎中的后备组中同步。
9. 可选。如果启用，非超级用户可以设置团队同步。

2. 重启 Red Hat Quay registry。

11.3.2. 为团队同步设置 Red Hat Quay 部署

1. 通过 OIDC 供应商登录到您的 Red Hat Quay registry。
2. 在 Red Hat Quay v2 UI 仪表板中，单击 **Create Organization**。
3. 输入 和 Organization 名称，如 **test-org**。

4. 点机构的名称。
5. 在导航窗格中，点 **Teams 和 membership**。
6. 点 **Create new team** 并输入名称，例如 **testteam**。
7. 在 **Create team** 弹出窗口中：
   a. 可选。将此团队添加到存储库。
   b. 通过在用户帐户名称中输入，添加团队成员，如 **user1**。
   c. 将机器人帐户添加到此团队。此页面提供了创建机器人帐户的选项。
8. 点击 **Next**。
9. 在 **Review and Finish** 页面中，查看您提供的信息，并点击 **Review and Finish**。
10. 要为 Red Hat Quay OIDC 部署启用团队同步，请点击 **Teams 和 membership** 页面上的 **Enable Directory Sync**。
11. 如果您的 OIDC 验证器是 Azure Entra ID，或者使用不同的供应商，则会提示您输入组对象 ID。请注意弹出窗口中的消息：

警告

请注意，在启用团队同步后，已经属于团队的用户成员资格将被撤销。OIDC 组将是单个数据源。这是一个不可逆的操作。Quay 中团队的用户成员资格将为只读。

12. 单击 **Enable Sync**。
13. 您将返回到 **Teams 和 membership** 页面。请注意，这个团队的用户已被删除，并在重新登录后重新添加。在这个阶段，只有机器人帐户仍然是团队的一部分。
    页面顶部的横幅确认团队已同步：

> This team is synchronized with a group in OIDC and its user membership is therefore read-only.

    单击 **Directory Synchronization Config** scala，会显示您的部署与之同步的 OIDC 组。
14. 从 Red Hat Quay registry 注销，再继续验证步骤。

验证

使用以下验证过程来确保 **user1** 显示为团队成员。

1. 重新登录您的 Red Hat Quay registry。
2. 点 **Organizations → test-org → test-team Teams 和 memberships**。**user1** 现在作为这个团队的团队成员显示。

验证

使用以下步骤通过 OIDC 供应商从组中删除 **user1**，然后将其从 Red Hat Quay 上的团队中删除。

1. 进入您的 OIDC 供应商管理控制台。
2. 进入 OIDC 供应商的 **Users** 页面。本页的名称因您的供应商而异。
3. 单击与 Red Hat Quay 关联的用户名，如 **user1**。
4. 从配置的身份提供程序中的组中删除该用户。
5. 从用户删除或取消分配访问权限。
6. 登录到您的 Red Hat Quay registry。
7. 点 **Organizations → test-org → test-team Teams 和 memberships**。**user1** 已从这个团队中删除。

第 12 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。

12.1. 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**。

12.2. 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。

12.3. 将 RED HAT QUAY 配置为使用 AWS STS

使用以下步骤编辑 Red Hat Quay **config.yaml** 文件以使用 AWS STS。

流程

1. 更新 Red Hat Quay 的 **config.yaml** 文件，使其包含以下信息：

```
# ...
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。

2. 重启 Red Hat Quay 部署。

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 13 章 RED HAT QUAY 下的 PROMETHEUS 和 GRAFANA 指标

Red Hat Quay 会为每个实例导出 Prometheus- 和 Grafana 兼容端点，以便轻松监控和警报。

13.1. 公开 PROMETHEUS 端点

13.1.1. 独立 Red Hat Quay

使用 **podman run** 启动 **Quay** 容器时，公开指标端口 **9091** ：

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443  -p 9091:9091\
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -v $QUAY/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1
```

现在，指标可用：

```
$ curl quay.example.com:9091/metrics
```

有关配置 Prometheus 和 Grafana 以监控 Quay 存储库计数的详细信息，请参阅使用 Prometheus 和 Grafana 监控 Quay 和 Grafana。

13.1.2. Red Hat Quay Operator

确定 **quay-metrics** 服务的集群 IP：

```
$ oc get services -n quay-enterprise
NAME                              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
example-registry-clair-app        ClusterIP   172.30.61.161    <none>        80/TCP,8089/TCP
18h
example-registry-clair-postgres   ClusterIP   172.30.122.136   <none>        5432/TCP
18h
example-registry-quay-app         ClusterIP   172.30.72.79     <none>
443/TCP,80/TCP,8081/TCP,55443/TCP   18h
example-registry-quay-config-editor   ClusterIP   172.30.185.61    <none>        80/TCP
18h
example-registry-quay-database    ClusterIP   172.30.114.192   <none>        5432/TCP
18h
example-registry-quay-metrics     ClusterIP   172.30.37.76     <none>        9091/TCP
18h
example-registry-quay-redis       ClusterIP   172.30.157.248   <none>        6379/TCP
18h
```

连接到集群，并使用 **quay-metrics** 服务的集群 IP 和端口访问指标：

```
$ oc debug node/master-0

sh-4.4# curl 172.30.37.76:9091/metrics

# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
```

```
go_gc_duration_seconds{quantile="0"} 4.0447e-05
go_gc_duration_seconds{quantile="0.25"} 6.2203e-05
...
```

13.1.3. 将 Prometheus 设置为消耗指标

Prometheus 需要一种方式来访问集群中运行的所有 Red Hat Quay 实例。在典型的设置中，通过在一个命名 DNS 条目中列出所有 Red Hat Quay 实例来完成，然后提供给 Prometheus。

13.1.4. Kubernetes 下的 DNS 配置

可将简单的 Kubernetes 服务配置为 为 Prometheus 提供 DNS 条目。

13.1.5. 手动集群的 DNS 配置

SkyDNS 是不使用 Kubernetes 时管理此 DNS 记录的简单解决方案。SkyDNS 可以在 etcd 集群上运行。可以在 etcd 存储中添加并删除集群中的每个 Red Hat Quay 实例的条目。SkyDNS 定期从那里读取它们，并相应地更新 DNS 记录中的 Quay 实例列表。

13.2. 指标简介

Red Hat Quay 提供了用于监控 registry 的指标，包括用于常规 registry 用量的指标、上传、下载、垃圾回收和验证。

13.2.1. 常规 registry 统计

常规 registry 统计可指示 registry 的增长量。

指标名称	描述
quay_user_rows	数据库中的用户数
quay_robot_rows	数据库中的机器人帐户数量
quay_org_rows	数据库中的组织数量
quay_repository_rows	数据库中的存储库数
quay_security_scanning_unscanned_images_remaining_total	最新安全扫描程序未扫描的镜像数

指标输出示例

```
# HELP quay_user_rows number of users in the database
# TYPE quay_user_rows gauge
quay_user_rows{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="65",process_name="globalpromstats.py"} 3

# HELP quay_robot_rows number of robot accounts in the database
# TYPE quay_robot_rows gauge
```

```
quay_robot_rows{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="65",process_name="globalpromstats.py"} 2

# HELP quay_org_rows number of organizations in the database
# TYPE quay_org_rows gauge
quay_org_rows{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="65",process_name="globalpromstats.py"} 2

# HELP quay_repository_rows number of repositories in the database
# TYPE quay_repository_rows gauge
quay_repository_rows{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="65",process_name="globalpromstats.py"} 4

# HELP quay_security_scanning_unscanned_images_remaining number of images that are not
scanned by the latest security scanner
# TYPE quay_security_scanning_unscanned_images_remaining gauge
quay_security_scanning_unscanned_images_remaining{host="example-registry-quay-app-
6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 5
```

13.2.2. 队列项目

queue items 指标提供有关 Quay 用于管理工作的多个队列的信息。

指标名称	描述
quay_queue_items_available	特定队列中的项目数
quay_queue_items_locked	正在运行的项目数
quay_queue_items_available_unlocked	等待处理的项目数

指标标签

- **queue_name ：** 队列的名称。其中之一：
  - **exportactionlogs：** 将请求排队到导出操作日志。然后，这些日志会被处理并放入存储中。然后，通过电子邮件将链接发送到请求者。
  - **namespacegc:**排队进行垃圾回收的命名空间
  - **通知：**要发送的存储库通知的队列
  - **repositorygc ：** 排队要垃圾回收的存储库
  - **secscanv4:** 特定于 Clair V4 的通知队列
  - **dockerfilebuild:** Quay docker build 的 Queue
  - **Imagestoragereplication：** 排队要在多个存储间复制的 Blob
  - **chunk_cleanup：** 需要删除的 Blob 段。这仅供某些存储实施使用，例如 Swift。

例如，队列标记 **repositorygc** 包含由存储库垃圾回收 worker 标记删除的存储库。对于带有 **repositorygc** 的 **queue_name** 标签的指标：

- **quay_queue_items_locked** 是当前正在删除的存储库数量。
- **quay_queue_items_available_unlocked** 是等待由 worker 处理的存储库数量。

指标输出示例

```
# HELP quay_queue_items_available number of queue items that have not expired
# TYPE quay_queue_items_available gauge
quay_queue_items_available{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="63",process_name="exportactionlogsworker.py",queue_name="exportactionlogs"} 0
...

# HELP quay_queue_items_available_unlocked number of queue items that have not expired and are not locked
# TYPE quay_queue_items_available_unlocked gauge
quay_queue_items_available_unlocked{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="63",process_name="exportactionlogsworker.py",queue_name="exportactionlogs"} 0
...

# HELP quay_queue_items_locked number of queue items that have been acquired
# TYPE quay_queue_items_locked gauge
quay_queue_items_locked{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="63",process_name="exportactionlogsworker.py",queue_name="exportactionlogs"} 0
```

13.2.3. 垃圾回收指标

这些指标显示已从垃圾回收(gc)中删除了多少个资源。它们显示 gc worker 运行的次数，并删除多少命名空间、存储库和 blob。

指标名称	描述
quay_gc_iterations_total	GCWorker 的迭代数
quay_gc_namespaces_purged_total	NamespaceGCWorker 清除的命名空间数量
quay_gc_repos_purged_total	RepositoryGCWorker 或 NamespaceGCWorker 清除的软件仓库数
quay_gc_storage_blobs_deleted_total	已删除的存储 Blob 数量

指标输出示例

```
# TYPE quay_gc_iterations_created gauge
quay_gc_iterations_created{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 1.6317823190189714e+09
...

# HELP quay_gc_iterations_total number of iterations by the GCWorker
```

```
# TYPE quay_gc_iterations_total counter
quay_gc_iterations_total{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
...

# TYPE quay_gc_namespaces_purged_created gauge
quay_gc_namespaces_purged_created{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"}
1.6317823190189433e+09
...

# HELP quay_gc_namespaces_purged_total number of namespaces purged by the
NamespaceGCWorker
# TYPE quay_gc_namespaces_purged_total counter
quay_gc_namespaces_purged_total{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
....

# TYPE quay_gc_repos_purged_created gauge
quay_gc_repos_purged_created{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"}
1.631782319018925e+09
...

# HELP quay_gc_repos_purged_total number of repositories purged by the RepositoryGCWorker or
NamespaceGCWorker
# TYPE quay_gc_repos_purged_total counter
quay_gc_repos_purged_total{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
...

# TYPE quay_gc_storage_blobs_deleted_created gauge
quay_gc_storage_blobs_deleted_created{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"}
1.6317823190189059e+09
...

# HELP quay_gc_storage_blobs_deleted_total number of storage blobs deleted
# TYPE quay_gc_storage_blobs_deleted_total counter
quay_gc_storage_blobs_deleted_total{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
...
```

13.2.3.1. 多部分上传指标

多部分上传指标显示 blob 上传到存储的数量(S3、Rados、GoogleCloudStorage、RHOCS)。当 Quay 无法正确将 Blob 上传到存储时，这有助于识别问题。

指标名称	描述
quay_multipart_uploads_started_total	启动的上传到 Quay 存储的 multipart 数
quay_multipart_uploads_completed_total	完成的上传到 Quay 存储的 multipart 数

指标输出示例

```
# TYPE quay_multipart_uploads_completed_created gauge
quay_multipart_uploads_completed_created{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 1.6317823308284895e+09
...

# HELP quay_multipart_uploads_completed_total number of multipart uploads to Quay storage that completed
# TYPE quay_multipart_uploads_completed_total counter
quay_multipart_uploads_completed_total{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0

# TYPE quay_multipart_uploads_started_created gauge
quay_multipart_uploads_started_created{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 1.6317823308284352e+09
...

# HELP quay_multipart_uploads_started_total number of multipart uploads to Quay storage that started
# TYPE quay_multipart_uploads_started_total counter
quay_multipart_uploads_started_total{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
...
```

13.2.4. 镜像推送/拉取指标

与推送和拉取镜像相关的可用指标数据的数量。

13.2.4.1. 镜像拉取总数

指标名称	描述
quay_registry_image_pulls_total	从 registry 下载的镜像数量。

指标标签

- **协议：** 所用的 registry 协议（应始终为 v2）
- **ref:** ref 用于拉取 - tag, manifest
- **status:** 请求的 http 返回码

13.2.4.2. 拉取的镜像字节

指标名称	描述
quay_registry_image_pulled_estimated_bytes_total	从 registry 下载的字节数

指标标签

- **协议：** 所用的 registry 协议（应始终为 v2）

13.2.4.3. 镜像拉取总数

指标名称	描述
quay_registry_image_pushes_total	从 registry 上传的镜像数量。

指标标签

- **协议：** 所用的 registry 协议（应始终为 v2）
- **pstatus：** 请求的 http 返回码
- **pmedia_type ：** 上传的清单类型

13.2.4.4. 推送的镜像字节数

指标名称	描述
quay_registry_image_pushed_bytes_total	上传到 registry 的字节数

指标输出示例

```
# HELP quay_registry_image_pushed_bytes_total number of bytes pushed to the registry
# TYPE quay_registry_image_pushed_bytes_total counter
quay_registry_image_pushed_bytes_total{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="221",process_name="registry:application"} 0
...
```

13.2.5. 身份验证指标

身份验证指标提供了身份验证请求的数量，根据类型进行标记，以及它是否成功。例如，此指标可用于监控失败的基本身份验证请求。

指标名称	描述
quay_authentication_attempts_total	registry 和 API 的身份验证尝试次数

指标标签

- **auth_kind：** 所用的验证类型，包括：
  - 基本的
  - oauth

- credentials

- **success:** true 或 false

指标输出示例

```
# TYPE quay_authentication_attempts_created gauge
quay_authentication_attempts_created{auth_kind="basic",host="example-registry-quay-app-
6df87f7b66-
9tfn6",instance="",job="quay",pid="221",process_name="registry:application",success="True"}
1.6317843039374158e+09
...

# HELP quay_authentication_attempts_total number of authentication attempts across the registry
and API
# TYPE quay_authentication_attempts_total counter
quay_authentication_attempts_total{auth_kind="basic",host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="221",process_name="registry:application",success="True"} 2
...
```

第 14 章 RED HAT QUAY 配额管理和强制概述

使用 Red Hat Quay，用户可以通过建立配置的存储配额限制来报告存储消耗并包含 registry 增长。内部 Red Hat Quay 用户现在带有以下功能来管理其环境的容量限制：

- **配额报告：** 通过此功能，超级用户可以跟踪其所有组织的存储消耗。此外，用户还可以跟踪其所分配组织的存储消耗。
- **配额管理：** 通过此功能，超级用户可以定义 Red Hat Quay 用户的软和硬性检查。软检查告知用户机构的存储消耗是否达到其配置的阈值。硬检查可防止用户在存储消耗达到配置的限制时推送到 registry。

这些功能一起允许 Red Hat Quay registry 的服务所有者定义服务级别协议并支持健康的资源预算。

14.1. 配额管理限制

配额管理有助于组织维护资源消耗。配额管理的一个限制是计算推送上的资源消耗，导致计算成为推送的关键路径的一部分。如果没有这种情况，使用数据可能会偏移。

最大存储配额大小取决于所选数据库：

表 14.1. worker 数量环境变量

变量	Description
Postgres	8388608 TB
MySQL	8388608 TB
SQL Server	16777216 TB

14.2. RED HAT QUAY 3.9 的配额管理

如果要升级到 Red Hat Quay 3.9，您必须重新配置配额管理功能。这是因为在 Red Hat Quay 3.9 中，计算方式不同。因此，Red Hat Quay 3.9 之前的总数不再有效。在 Red Hat Quay 3.9 中配置配额管理有两种方法，其在以下部分中详细介绍。

注意

- 这是在升级到 Red Hat Quay 3.9 后必须完成的一个时间计算。
- 创建、更新和删除配额需要超级用户权限。虽然可以为用户和机构设置配额，但您无法使用 Red Hat Quay UI 重新配置 *用户配额*，但您必须使用 API。

14.2.1. 选项 A：通过调整 QUOTA_TOTAL_DELAY 功能标记，为 Red Hat Quay 3.9 配置配额管理

通过调整 **QUOTA_TOTAL_DELAY** 功能标记，使用以下步骤重新计算 Red Hat Quay 3.9 配额管理。

注意

通过重新计算选项，总数显示为 0.00 KB，直到为 **QUOTA_TOTAL_DELAY** 指定的分配时间为止。

先决条件

- 您已升级到 Red Hat Quay 3.9。
- 以超级用户身份登录到 Red Hat Quay 3.9。

流程

1. 使用以下 **config.yaml** 设置部署 Red Hat Quay 3.9：

```
FEATURE_QUOTA_MANAGEMENT: true
FEATURE_GARBAGE_COLLECTION: true
PERMANENTLY_DELETE_TAGS: true
QUOTA_TOTAL_DELAY_SECONDS: 1800 1
RESET_CHILD_MANIFEST_EXPIRATION: true
```

1 **QUOTA_TOTAL_DELAY_SECONDS** 标志默认为 **1800** 秒或 30 分钟。这允许 Red Hat Quay 3.9 在配额管理功能开始计算推送的每个 Blob 的存储消耗前成功部署。将此标志设置为较低数字可能会导致计算错误；必须将其设置为一个大于 Red Hat Quay 部署启动的时间。**1800 是推荐的设置，但启动时间大于 30 分钟的大型部署可能需要较长的时间超过 1800。**

2. 导航到 Red Hat Quay UI，再点您的机构名称。
3. 使用 的总配额数应为 0.00 KB。此外，也应当存在 Backfill Queued 指示符。
4. 在分配的时间（例如 30 分钟）后，刷新您的 Red Hat Quay 部署页面并返回到您的机构。现在，应该会显示 Total Quota Consumed。

14.2.2. 选项 B：通过将 QUOTA_TOTAL_DELAY_SECONDS 设置为 0 来为 Red Hat Quay 3.9 配置配额管理

使用以下步骤，通过将 QUOTA_TOTAL_DELAY_SECONDS 设置为 0 来重新计算 Red Hat Quay 3.9 配额管理。

注意

使用此选项可防止进行错误计算，但会更多时间密集型。当 Red Hat Quay 部署将 FEATURE_QUOTA_MANAGEMENT 参数从 false 改为 true 时，请使用以下步骤。大多数用户都会找到 xref:

先决条件

- **您已升级到 Red Hat Quay 3.9。**
- **以超级用户身份登录到 Red Hat Quay 3.9。**

流程

1. 使用以下 config.yaml 设置部署 Red Hat Quay 3.9：

```
FEATURE_GARBAGE_COLLECTION: true
FEATURE_QUOTA_MANAGEMENT: true
QUOTA_BACKFILL: false
QUOTA_TOTAL_DELAY_SECONDS: 0
PERMANENTLY_DELETE_TAGS: true
RESET_CHILD_MANIFEST_EXPIRATION: true
```

2. 导航到 Red Hat Quay UI，再点您的机构名称。
3. 使用 的总配额数应为 0.00 KB。
4. 重新部署 Red Hat Quay，并将 QUOTA_BACKFILL 标志设为 true。例如：

```
QUOTA_BACKFILL: true
```

注意

如果您选择在计算总数后禁用配额管理，Red Hat Quay 会将这些总数标记为 stale。如果您将来再次重新启用配额管理功能，则回填 worker 会重新计算这些命名空间和存储库。

14.3. 测试 RED HAT QUAY 3.9 的配额管理

为 Red Hat Quay 3.9 配置配额管理后，复制镜像现在只计算在仓库总数中一次。

使用以下步骤测试 duplicative 镜像只计算到存储库总计一次。

先决条件

- 您已为 Red Hat Quay 3.9 配置了配额管理。

流程

1. 输入以下命令拉取示例镜像，如 iwl:18.04 ：

```
$ podman pull ubuntu:18.04
```

2. 输入以下命令标记同一镜像两次：

```
$ podman tag docker.io/library/ubuntu:18.04 quay-server.example.com/quota-test/ubuntu:tag1
```

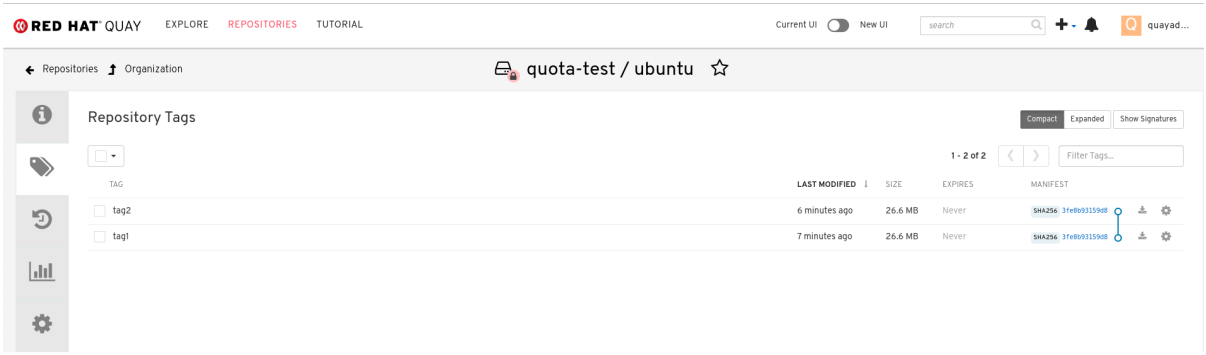
```
$ podman tag docker.io/library/ubuntu:18.04 quay-server.example.com/quota-test/ubuntu:tag2
```

3. 输入以下命令将示例镜像推送到您的机构：

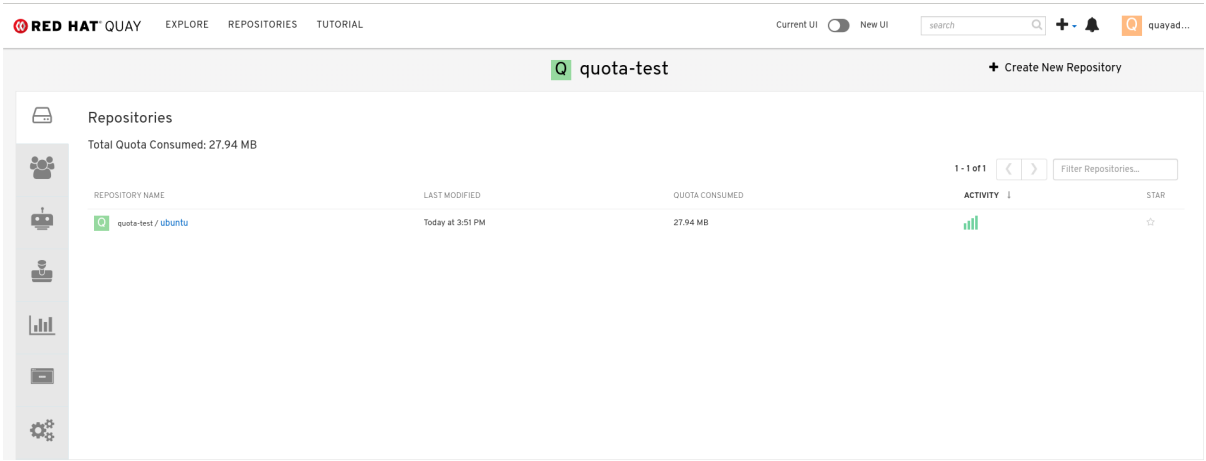
```
$ podman push --tls-verify=false quay-server.example.com/quota-test/ubuntu:tag1
```

```
$ podman push --tls-verify=false quay-server.example.com/quota-test/ubuntu:tag2
```

4. 在 Red Hat Quay UI 上，导航到 Organization，再单击 Repository Name，例如 quota-test/ubuntu。然后，单击 标签。应当有两个存储库标签 tag1 和 tag2，各自具有相同的清单。例如：



但是，通过单击 Organization 链接，可以看到 Total Quota Consumed为 27.94 MB，这意味着 Ubuntu 镜像仅被一次考虑：



如果您删除其中一个 Ubuntu 标签，则 总配额 Consumed 保持不变。

注意

如果您将 Red Hat Quay 时间机器配置为大于 0 秒，则只有这些标签通过时间窗时才会发生减法。如果要加快永久删除，请参阅 Red Hat Quay 3.9 中永久删除镜像标签。

14.4. 设置默认配额

要指定应用于每个机构和用户的系统范围默认存储配额，您可以使用 DEFAULT_SYSTEM_REJECT_QUOTA_BYTES 配置标志。

如果您为机构或用户配置特定的配额，然后删除该配额，则系统范围内的默认配额会在设置配额时应用。同样，如果您已经为某个机构或用户配置了特定的配额，然后修改系统范围的默认配额，则更新的系统范围默认设置将覆盖任何特定的设置。

有关 DEFAULT_SYSTEM_REJECT_QUOTA_BYTES 标志的更多信息，

请参阅链接：

14.5. 在 RED HAT QUAY UI 中建立配额

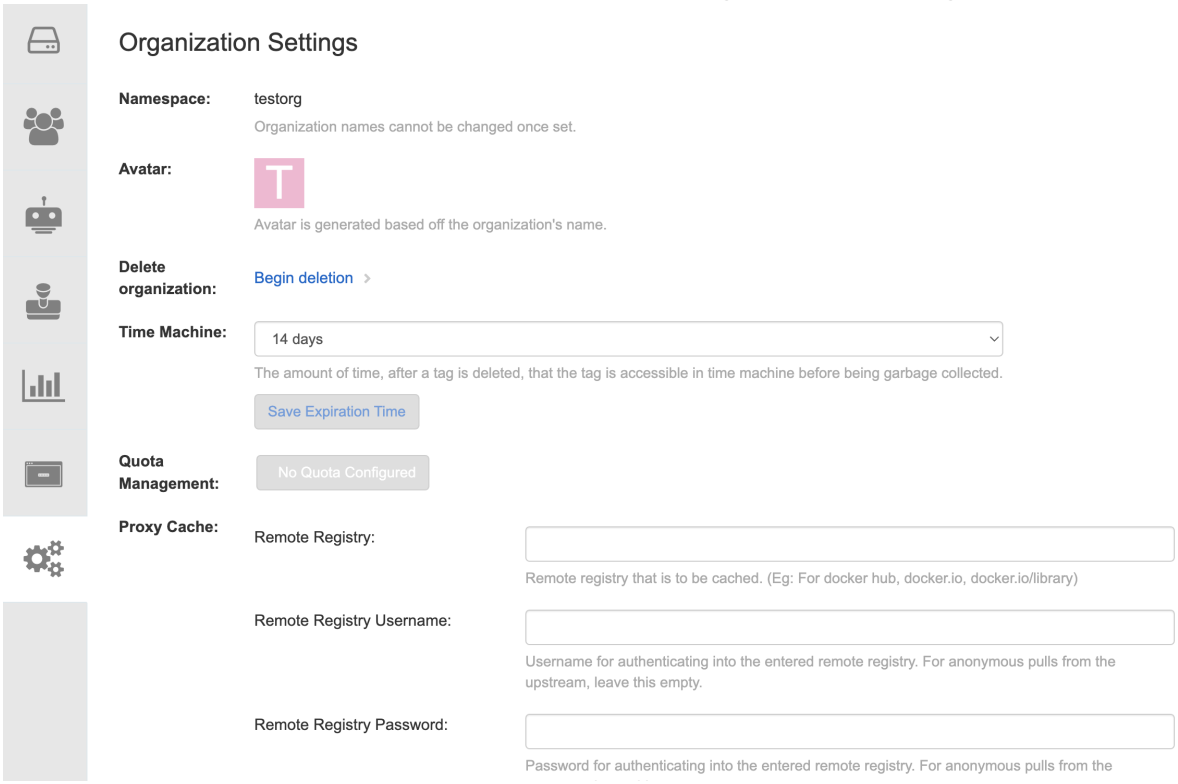
以下流程描述了如何报告存储消耗并建立存储配额限制。

先决条件

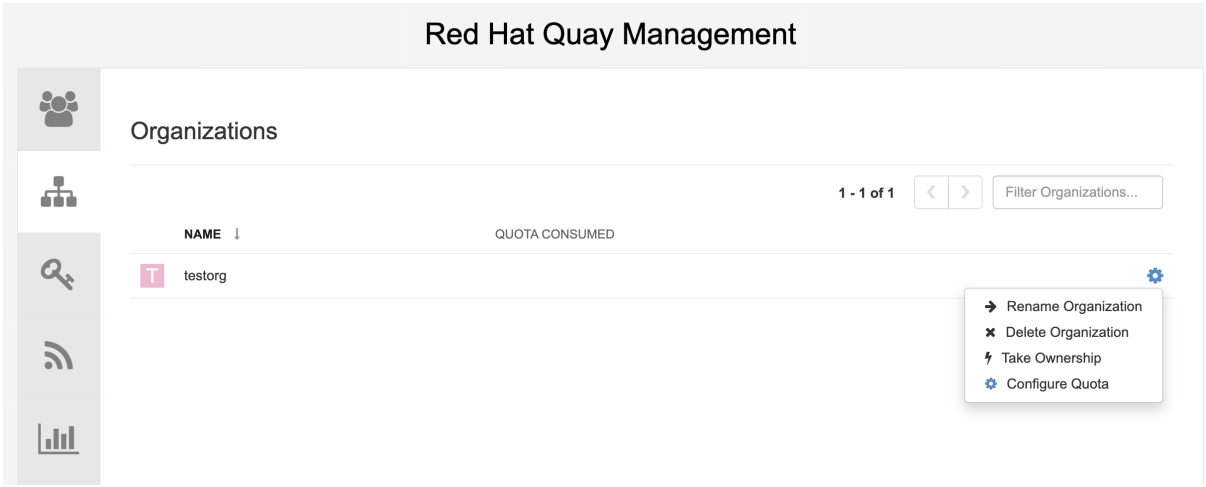
- Red Hat Quay registry。
- 超级用户帐户。
- 足够的存储来满足配额限制的需求。

流程

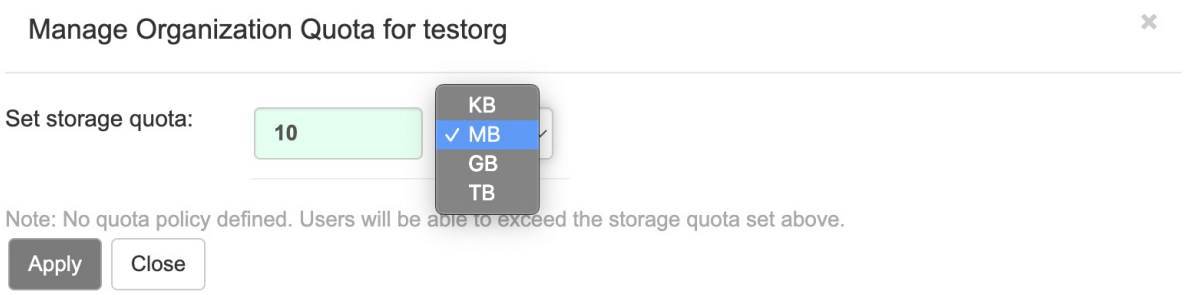
1. 创建新机构或选择现有机构。最初，没有配置配额，如 Organization Settings 选项卡中显示：



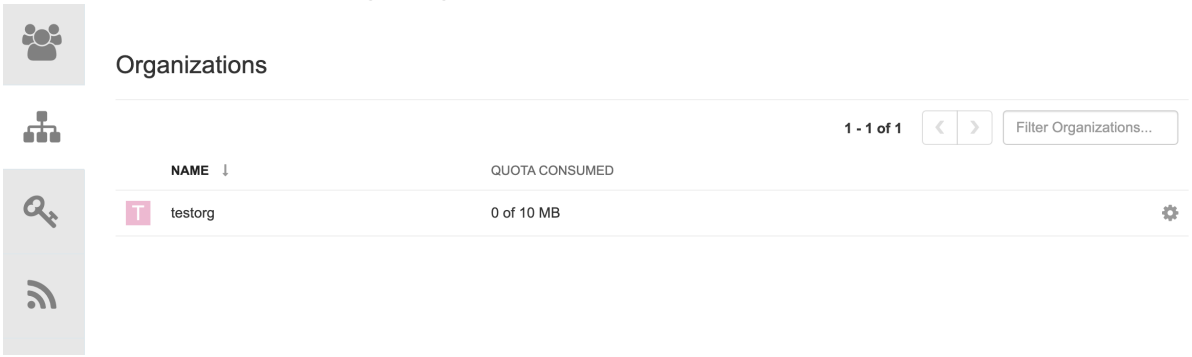
2. 以超级用户身份登录 registry，再导航到 Super User Admin Panel 上的 Manage Organizations 选项卡。点您要为其创建存储配额限制的机构 Options 图标：



3. 单击 Configure Quota，再输入初始配额，例如 10 MB。然后点 应用 和关闭 ：

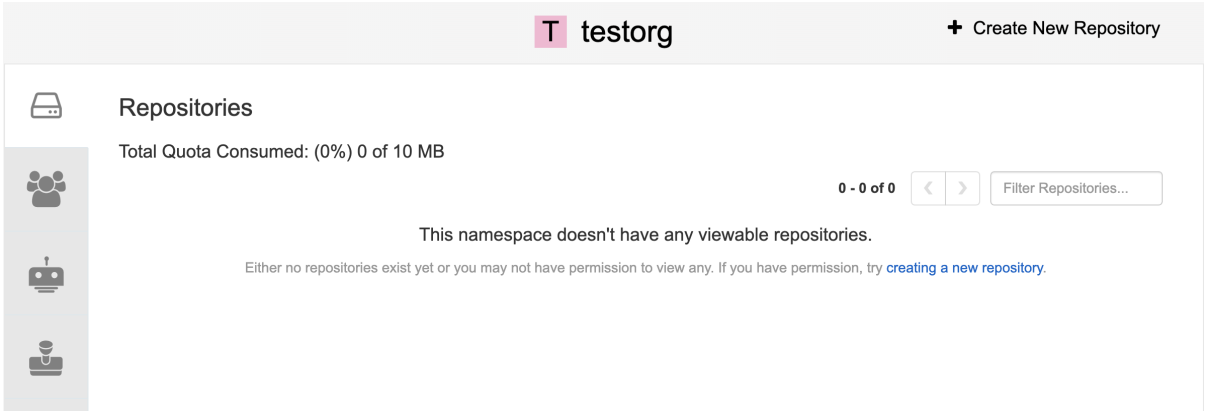


4. 检查超级用户面板中的 Manage Organizations 标签页上显示有 0 of 10 MB的配额：

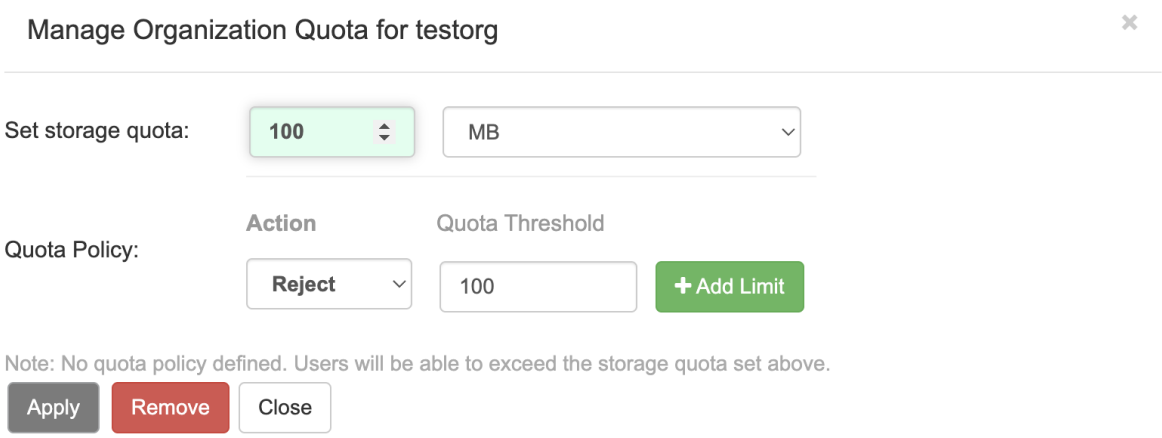


消耗的配额信息也可以在 Organization 页面中直接提供：

初始消耗的配额



5. 若要将配额增加到 100MB，可导航到超级用户面板中的 Manage Organizations 选项卡。单击 Options 图标，再选择 Configure Quota，将配额设置为 100 MB。点 Apply，然后关闭 ：



6. 输入以下命令拉取示例镜像：

```
$ podman pull ubuntu:18.04
```

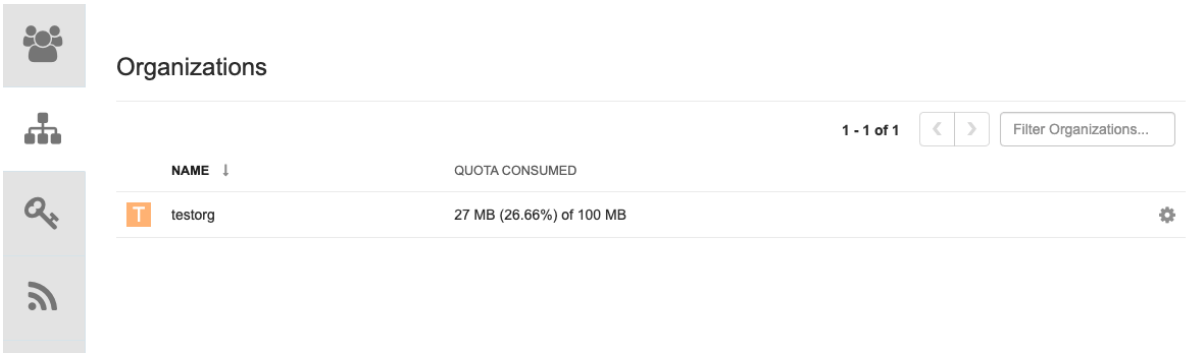
7. 输入以下命令标记示例镜像：

```
$ podman tag docker.io/library/ubuntu:18.04 example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:18.04
```

8. 输入以下命令将示例镜像推送到机构：

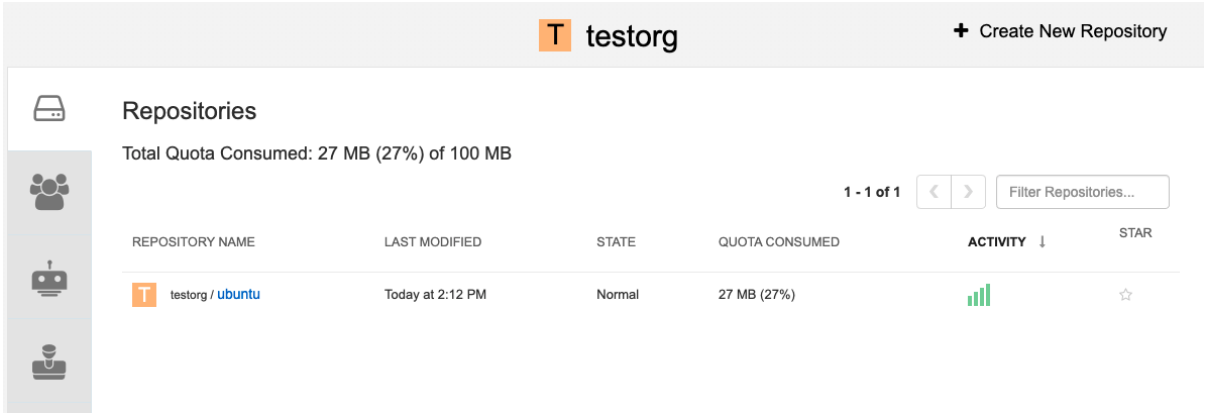
```
$ podman push --tls-verify=false example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:18.04
```

9. 在超级用户面板中，会显示每个机构消耗的配额：



10. Organization 页面显示镜像使用的配额的总比例：

    第一个镜像消耗的配额总数



11. 输入以下命令拉取第二个示例镜像：

```
$ podman pull nginx
```

12. 输入以下命令标记第二个镜像：

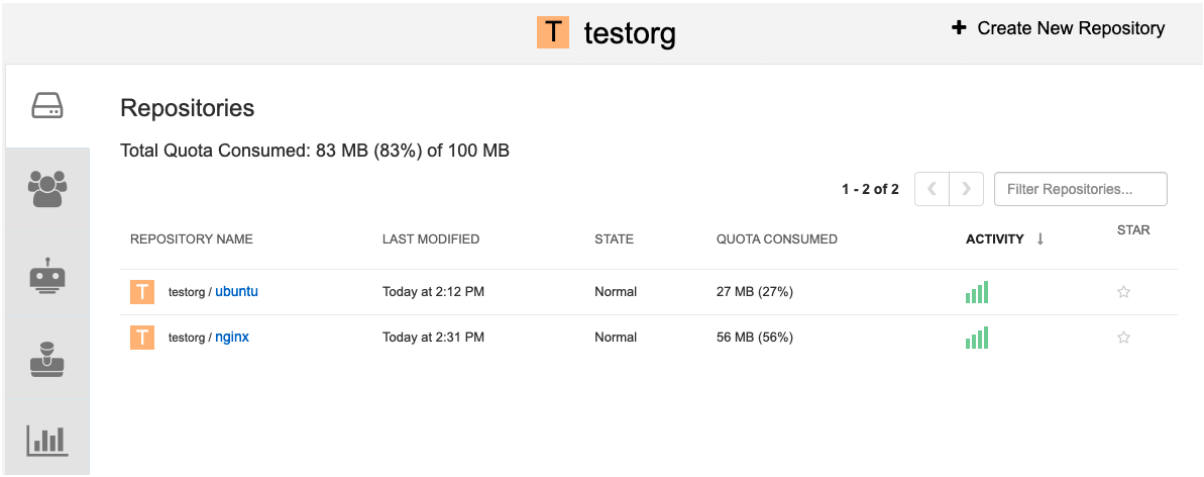
```
$ podman tag docker.io/library/nginx example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/nginx
```

13. 输入以下命令将第二个镜像推送到机构：

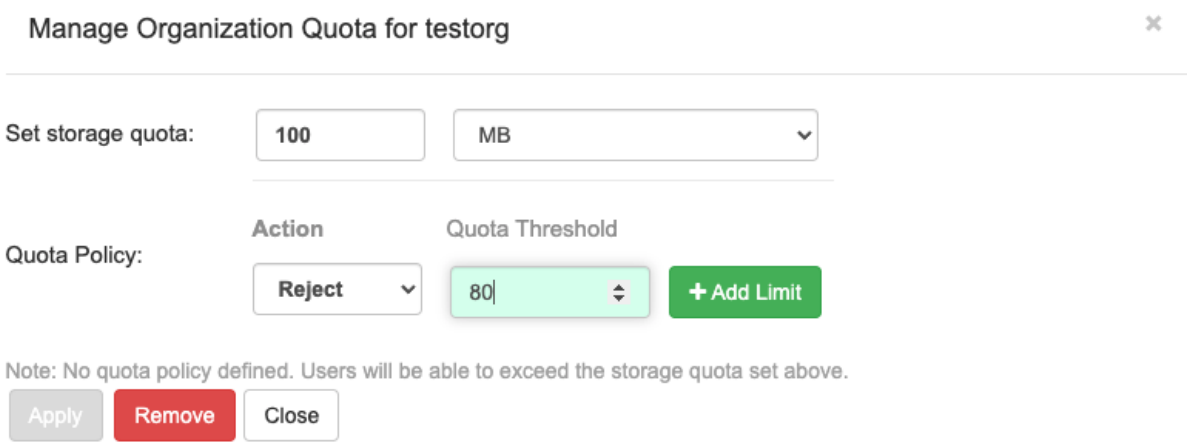
```
$ podman push --tls-verify=false example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/nginx
```

14. Organization 页面显示该机构中每个存储库使用的配额的总比例：

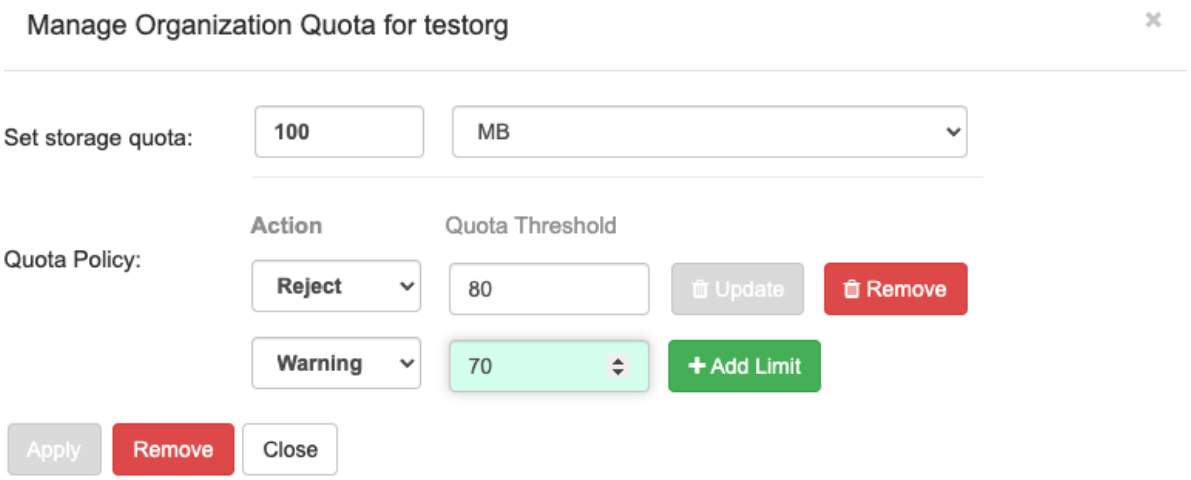
    每个仓库消耗的总配额



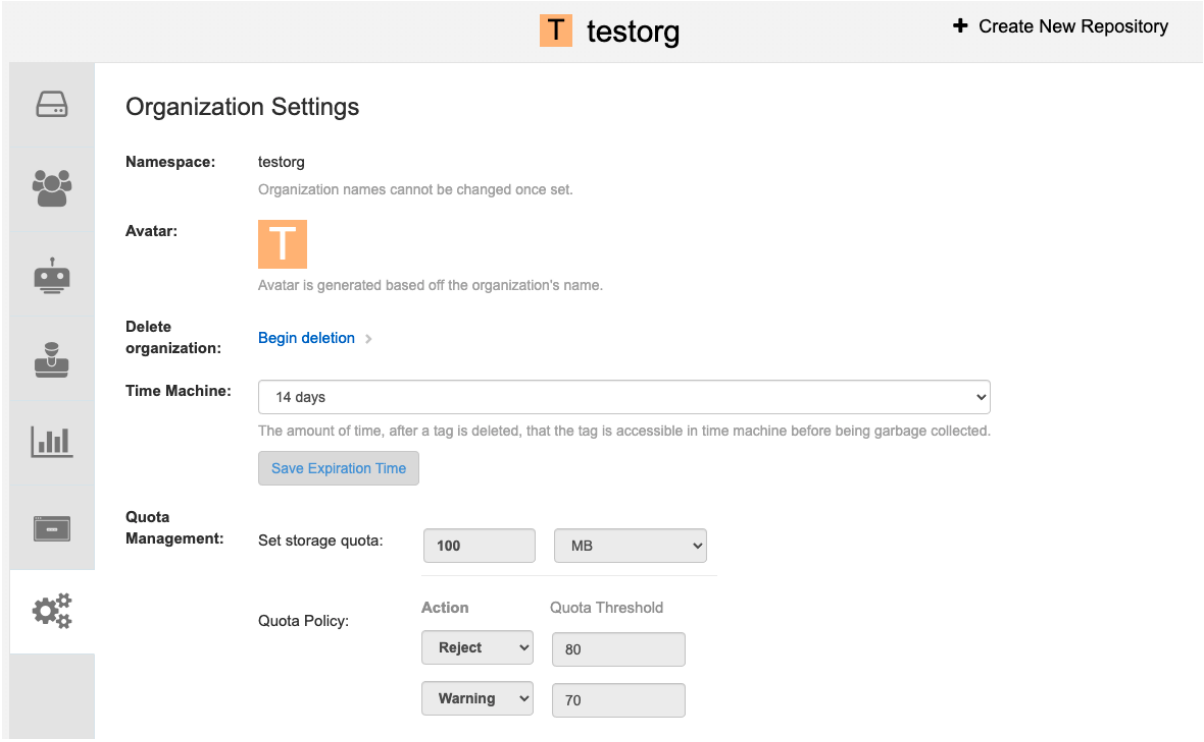
15. 创建 *reject* 和 *warning* 限制：
    从超级用户面板中，导航到 Manage Organizations 选项卡。单击 机构的 Options 图标，再选择 Configure Quota。在 Quota Policy 部分中，将 Action 类型设置为 Reject，将 Quota Threshold 设置为 80，然后单击 Add Limit ：



16. 要创建 *警告* 限制，请选择 Warning 作为 Action 类型，将 Quota Threshold 设置为 70，然后单击 Add Limit ：



17. 在配额弹出窗口中点 Close。限制可在 Organization 页面的 Settings 选项卡中查看，但不可编辑：



18. 推送超过 reject 限制的镜像：
由于 reject 限制(80%)已设置为低于当前存储库大小(~83%)，因此下一个推送的镜像会自动被拒绝。

镜像推送示例

```
$ podman pull ubuntu:20.04

$ podman tag docker.io/library/ubuntu:20.04 example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:20.04

$ podman push --tls-verify=false example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:20.04
```

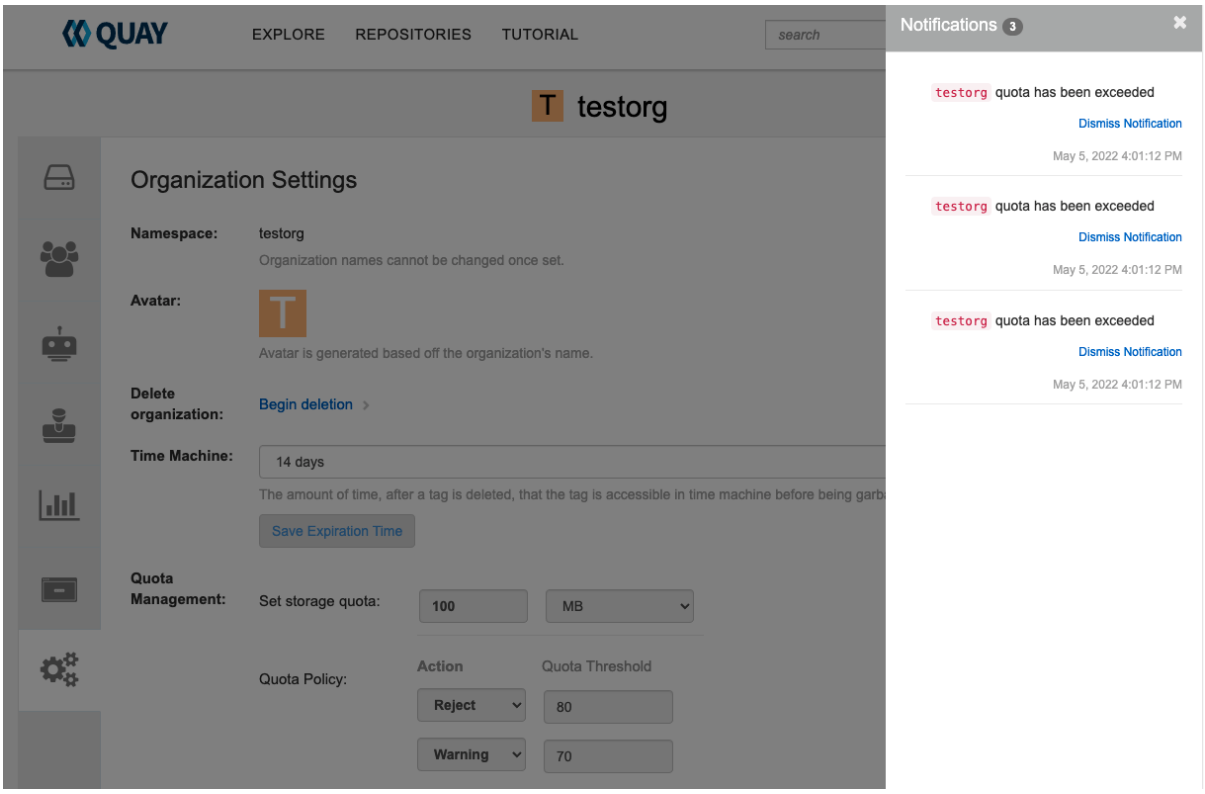
当超过配额时的输出示例

```
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
WARN[0002] failed, retrying in 1s ... (1/3). Error: Error writing blob: Error initiating layer upload to /v2/testorg/ubuntu/blobs/uploads/ in example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been exceeded on namespace
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
WARN[0005] failed, retrying in 1s ... (2/3). Error: Error writing blob: Error initiating layer upload to /v2/testorg/ubuntu/blobs/uploads/ in example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been exceeded on
```

```
namespace
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
WARN[0009] failed, retrying in 1s ... (3/3). Error: Error writing blob: Error initiating
layer upload to /v2/testorg/ubuntu/blobs/uploads/ in example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been exceeded on
namespace
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
Error: Error writing blob: Error initiating layer upload to
/v2/testorg/ubuntu/blobs/uploads/ in example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been exceeded on
namespace
```

19. **超过限制后，UI 中会显示通知：**

配额通知



14.6. 使用 RED HAT QUAY API 建立配额

首次创建机构时，它不会应用配额。使用 /api/v1/organization/{organization}/quota 端点：

示例命令

```
$ curl -k -X GET -H "Authorization: Bearer <token>" -H 'Content-Type: application/json'
https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/api/v1/organization/testorg/quota  | jq
```

输出示例

```
[]
```

14.6.1. 设置配额

要为机构设置配额，请将数据 POST 到 /api/v1/organization/{orgname}/quota 端点：.Sample

```
$ curl -k -X POST -H "Authorization: Bearer <token>" -H 'Content-Type: application/json' -d
'{"limit_bytes": 10485760}' https://example-registry-quay-quay-
enterprise.apps.docs.quayteam.org/api/v1/organization/testorg/quota | jq
```

输出示例

```
"Created"
```

14.6.2. 查看配额

要查看应用的配额，请来自 /api/v1/organization/{orgname}/quota 端点的 GET 数据：

示例命令

```
$ curl -k -X GET -H "Authorization: Bearer <token>" -H 'Content-Type: application/json'
https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/api/v1/organization/testorg/quota | jq
```

输出示例

```
[
  {
    "id": 1,
    "limit_bytes": 10485760,
    "default_config": false,
    "limits": [],
    "default_config_exists": false
  }
]
```

14.6.3. 修改配额

要修改现有的配额（在这个实例中从 10 MB 改为 100 MB），使用到 /api/v1/organization/{orgname}/quota/{quota_id} 端点的 PUT 数据：

示例命令

```
$ curl -k -X PUT -H "Authorization: Bearer <token>" -H 'Content-Type: application/json' -d
'{"limit_bytes": 104857600}' https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/api/v1/organization/testorg/quota/1 | jq
```

输出示例

```
{
  "id": 1,
  "limit_bytes": 104857600,
  "default_config": false,
  "limits": [],
  "default_config_exists": false
}
```


14.6.4. 推送镜像

要查看消耗的存储，请将各种镜像推送到机构。

14.6.4.1. 推送 iwl:18.04

从命令行将 ubuntu:18.04 推送到机构：

示例命令

```
$ podman pull ubuntu:18.04

$ podman tag docker.io/library/ubuntu:18.04 example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:18.04

$ podman push --tls-verify=false example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:18.04
```

14.6.4.2. 使用 API 查看配额使用量

要查看消耗的存储，针对 /api/v1/repository 端点使用 GET 数据：

示例命令

```
$ curl -k -X GET -H "Authorization: Bearer <token>" -H 'Content-Type: application/json' 'https://example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/api/v1/repository?last_modified=true&namespace=testorg&popularity=true&public=true' | jq
```

输出示例


```
{
  "repositories": [
    {
      "namespace": "testorg",
      "name": "ubuntu",
      "description": null,
      "is_public": false,
      "kind": "image",
      "state": "NORMAL",
      "quota_report": {
        "quota_bytes": 27959066,
        "configured_quota": 104857600
      },
      "last_modified": 1651225630,
      "popularity": 0,
```

```
      "is_starred": false
    }
  ]
}
```

14.6.4.3. 推送另一个镜像

1. 拉取、标签和推送第二个镜像，如 nginx ：

   示例命令

   ```
   $ podman pull nginx

   $ podman tag docker.io/library/nginx example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/nginx

   $ podman push --tls-verify=false example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/testorg/nginx
   ```

2. 要查看机构中存储库的配额报告，请使用 /api/v1/repository 端点：

   示例命令

   ```
   $ curl -k -X GET -H "Authorization: Bearer <token>" -H 'Content-Type: application/json' 'https://example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org/api/v1/repository?last_modified=true&namespace=testorg&popularity=true&public=true'
   ```

   输出示例

   ```
   {
     "repositories": [
       {
         "namespace": "testorg",
         "name": "ubuntu",
         "description": null,
         "is_public": false,
         "kind": "image",
         "state": "NORMAL",
         "quota_report": {
           "quota_bytes": 27959066,
           "configured_quota": 104857600
         },
         "last_modified": 1651225630,
         "popularity": 0,
         "is_starred": false
       },
       {
         "namespace": "testorg",
         "name": "nginx",
         "description": null,
         "is_public": false,
         "kind": "image",
         "state": "NORMAL",
   ```

```
    "quota_report": {
      "quota_bytes": 59231659,
      "configured_quota": 104857600
    },
    "last_modified": 1651229507,
    "popularity": 0,
    "is_starred": false
  }
 ]
}
```


3. 要查看机构详情中的配额信息，请使用 /api/v1/organization/{orgname} 端点：

   示例命令

```
$ curl -k -X GET -H "Authorization: Bearer <token>" -H 'Content-Type: application/json'
'https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/api/v1/organization/testorg' | jq
```

   输出示例


```
{
 "name": "testorg",
 ...
 "quotas": [
  {
   "id": 1,
   "limit_bytes": 104857600,
   "limits": []
  }
 ],
 "quota_report": {
  "quota_bytes": 87190725,
  "configured_quota": 104857600
 }
}
```


14.6.5. 使用配额限制拒绝推送

如果镜像推送超过定义的配额限制，则会出现软或硬检查：

- 对于软检查，或*警告*，用户会收到通知。
- 对于硬检查 *或拒绝*，推送将被终止。

14.6.5.1. 设置 reject 和 warning 限制

要设置 *reject* 和 *warning* 限制，POST 数据到 /api/v1/organization/{orgname}/quota/{quota_id}/limit 端点：

reject limit 命令示例

```
$ curl -k -X POST -H "Authorization: Bearer <token>" -H 'Content-Type: application/json' -d
'{"type":"Reject","threshold_percent":80}'  https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/api/v1/organization/testorg/quota/1/limit
```

警告限制命令示例

```
$ curl -k -X POST -H "Authorization: Bearer <token>" -H 'Content-Type: application/json' -d
'{"type":"Warning","threshold_percent":50}'  https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/api/v1/organization/testorg/quota/1/limit
```

14.6.5.2. 查看拒绝和警告限制

要查看 *reject* 和 *warning* 限制，请使用 /api/v1/organization/{orgname}/quota 端点：

查看配额限制

```
$  curl -k -X GET -H "Authorization: Bearer <token>" -H 'Content-Type: application/json'
https://example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/api/v1/organization/testorg/quota | jq
```

配额限制的输出示例

```
[
  {
    "id": 1,
    "limit_bytes": 104857600,
    "default_config": false,
    "limits": [
      {
        "id": 2,
        "type": "Warning",
        "limit_percent": 50
      },
      {
        "id": 1,
        "type": "Reject",
        "limit_percent": 80
      }
    ],
    "default_config_exists": false
  }
]
```

14.6.5.3. 超过 reject 限制时推送镜像

在本例中，reject 限制(80%)已设置为低于当前存储库大小(~83%)，因此下一个推送应自动被拒绝。

从命令行将示例镜像推送到机构：

镜像推送示例

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

```
$ podman tag docker.io/library/ubuntu:20.04 example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:20.04

$ podman push --tls-verify=false example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org/testorg/ubuntu:20.04
```

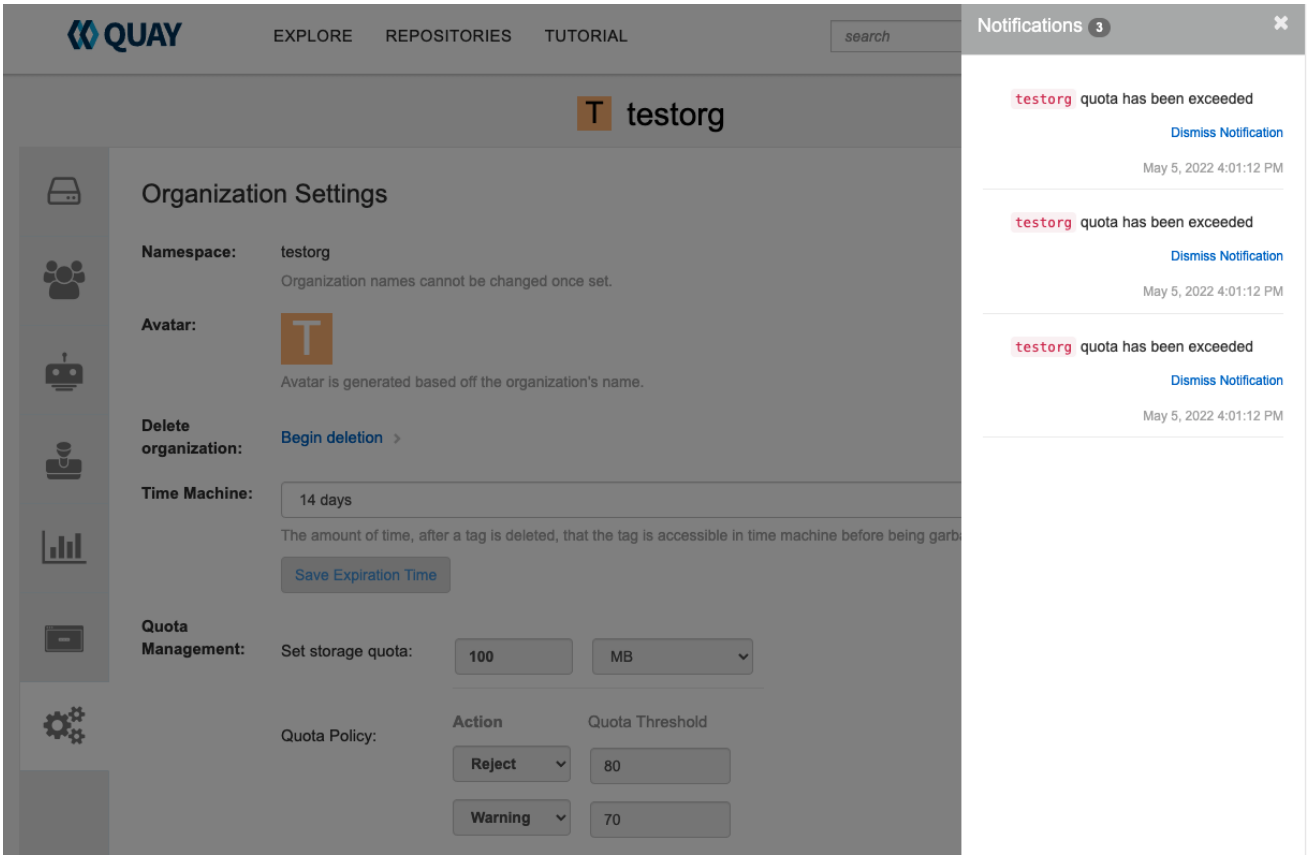
当超过配额时的输出示例

```
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
WARN[0002] failed, retrying in 1s ... (1/3). Error: Error writing blob: Error initiating layer
upload to /v2/testorg/ubuntu/blobs/uploads/ in example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been exceeded on namespace
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
WARN[0005] failed, retrying in 1s ... (2/3). Error: Error writing blob: Error initiating layer
upload to /v2/testorg/ubuntu/blobs/uploads/ in example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been exceeded on namespace
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
WARN[0009] failed, retrying in 1s ... (3/3). Error: Error writing blob: Error initiating layer
upload to /v2/testorg/ubuntu/blobs/uploads/ in example-registry-quay-quay-
enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been exceeded on namespace
Getting image source signatures
Copying blob d4dfaa212623 [--------------------------------------] 8.0b / 3.5KiB
Copying blob cba97cc5811c [--------------------------------------] 8.0b / 15.0KiB
Copying blob 0c78fac124da [--------------------------------------] 8.0b / 71.8MiB
Error: Error writing blob: Error initiating layer upload to /v2/testorg/ubuntu/blobs/uploads/ in
example-registry-quay-quay-enterprise.apps.docs.gcp.quaydev.org: denied: Quota has been
exceeded on namespace
```

14.6.5.4. 超过限制的通知

超过限制时，会出现通知：

配额通知



14.7. 计算 RED HAT QUAY 3.9 中 REGISTRY 的总大小

使用以下步骤对 registry 的总计算进行队列。

注意

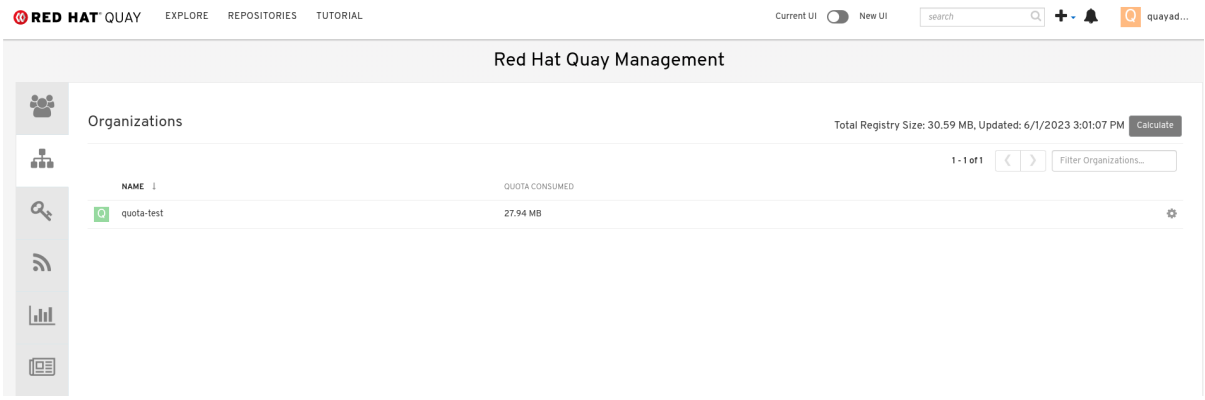
这个功能是按需完成的，计算 registry 总计是数据库密集型。请谨慎使用。

先决条件

- 您已升级到 Red Hat Quay 3.9。
- 以 Red Hat Quay 超级用户身份登录。

流程

1. 在 Red Hat Quay UI 中，点击您的 username → Super User Admin Panel。
2. 在导航窗格中，单击 Manage Organizations。
3. 点 Calculate，单击 Total Registry Size: 0.00 KB, Updated: Never , Calculation required。然后，单击确定。
4. 几分钟后，根据 registry 的大小，刷新页面。现在，应计算 Total Registry Size。例如：



14.8. 永久删除镜像标签

在某些情况下，用户可能想要删除时间窗外的镜像标签。使用以下步骤手动删除镜像标签。

重要

以下流程的结果无法撤消。请谨慎使用。

14.8.1. 使用 Red Hat Quay v2 UI 永久删除镜像标签

使用以下步骤通过 Red Hat Quay v2 UI 永久删除镜像标签。

先决条件

- 您已在 config.yaml 文件中将 FEATURE_UI_V2 设置为 true。

流程

1. 在 config.yaml 文件中，确保将 PERMANENTLY_DELETE_TAGS 和 RESET_CHILD_MANIFEST_EXPIRATION 参数设置为 true。例如：

```
PERMANENTLY_DELETE_TAGS: true
RESET_CHILD_MANIFEST_EXPIRATION: true
```

2. 在导航窗格中，单击 Repositories。
3. 单击存储库的名称，例如 quayadmin/busybox。
4. 选中要删除的镜像标签框，例如 test。
5. 点 Actions → Permanently Delete。

重要

此操作是永久的，无法撤销。

14.8.2. 使用 Red Hat Quay 传统 UI 永久删除镜像标签

使用以下步骤通过 Red Hat Quay legacy UI 永久删除镜像标签。

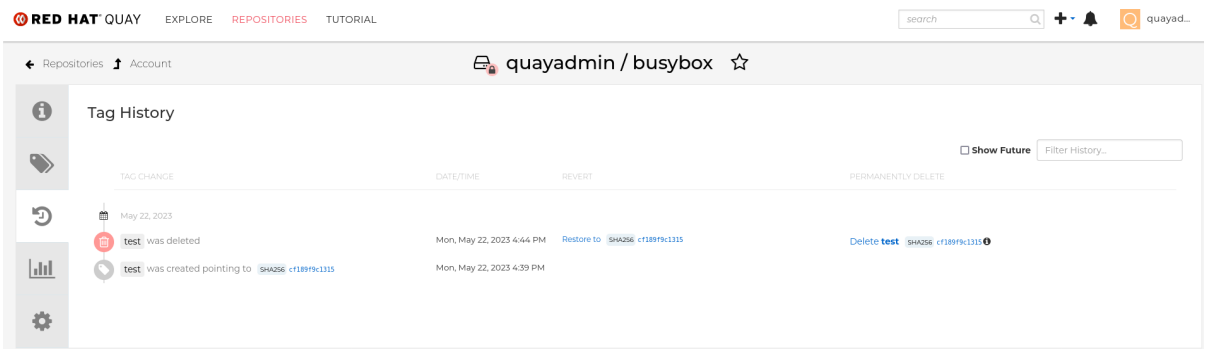
流程

1. 在 config.yaml 文件中，确保将 PERMANENTLY_DELETE_TAGS 和 RESET_CHILD_MANIFEST_EXPIRATION 参数设置为 true。例如：

```
PERMANENTLY_DELETE_TAGS: true
RESET_CHILD_MANIFEST_EXPIRATION: true
```

2. 在 Red Hat Quay UI 上，点 Repositories 以及包含要删除的镜像标签的存储库名称，如 quayadmin/busybox。
3. 在导航窗格中，单击 Tags。
4. 选中您要删除的标签名称框，例如 test。
5. 点 Actions 下拉菜单，再选择 Delete Tags → Delete Tag。
6. 在导航窗格中，单击 Tag History。
7. 在刚删除的标签名称上，例如 test，单击 Permanently Delete 类别下的 Delete test。例如：

永久删除镜像标签



重要

此操作是永久的，无法撤销。

第 15 章 RED HAT QUAY AUTO-PRUNING 概述

Red Hat Quay 管理员可以对命名空间和存储库设置自动运行策略。此功能允许根据指定条件在命名空间中自动删除镜像标签或存储库，这可让 Red Hat Quay 机构所有者通过自动修剪内容保存在存储配额下。

目前，添加了两个策略：

- 按标签数修剪镜像。对于此策略，当实际标签数量超过所需标签数量时，最旧的标签会被创建日期删除，直到达到所需的标签数量为止。
- 按创建日期修剪镜像标签。对于此策略，任何具有超过给定时间范围的创建日期的标签（如 10 天）都将被删除。

在标签被自动修剪后，它们进入 Red Hat Quay 时间机器或时间，在标签被删除后，标签可以在收集垃圾回收前访问。镜像标签的过期时间取决于您的机构设置。如需更多信息，请参阅 Red Hat Quay 垃圾回收。

用户只能为每个命名空间或存储库配置一个策略；这可以通过 Red Hat Quay v2 UI 完成。也可以通过命令行界面(CLI)使用 API 端点来设置策略。

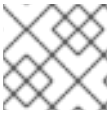
15.1. 自动运行的先决条件和限制

以下先决条件和限制适用于 auto-pruning 功能：

- 使用 Red Hat Quay 传统 UI 时无法使用此功能。您必须使用 v2 UI 创建、查看或修改自动运行策略。
- 只有支持 FOR UPDATE SKIP LOCKED SQL 命令的数据库才支持自动运行。

15.2. 使用 RED HAT QUAY UI 管理自动运行策略

auto-pruning 策略使用 Red Hat Quay v2 UI 创建。这可以在配置 Red Hat Quay config.yaml 文件后完成，以启用自动运行功能和 v2 UI。



注意

使用 Red Hat Quay 传统 UI 时无法使用此功能。

15.2.1. 配置 Red Hat Quay auto-pruning 功能

使用以下步骤配置 Red Hat Quay config.yaml 文件来启用 auto-pruning 功能。

先决条件

- 您已在 config.yaml 文件中将 FEATURE_UI_V2 设置为 true。

流程

- 在 Red Hat Quay config.yaml 文件中，添加和设置 FEATURE_AUTO_PRUNE 环境变量为 True。例如：

```
# ...
FEATURE_AUTO_PRUNE: true
# ...
```

15.2.2. 使用 Red Hat Quay v2 UI 为命名空间创建自动修剪策略

使用以下步骤，使用 Red Hat Quay v2 UI 为命名空间创建自动修剪策略。

先决条件

- 您已启用了 FEATURE_AUTO_PRUNE 功能。

流程

1. 标记四个示例镜像，如 busybox，它将推送到启用了 auto-pruning 的存储库。例如：

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

```
$ podman tag docker.io/library/busybox <quay-server.example.com>/<quayadmin>/busybox:test2
```

```
$ podman tag docker.io/library/busybox <quay-server.example.com>/<quayadmin>/busybox:test3
```

```
$ podman tag docker.io/library/busybox <quay-server.example.com>/<quayadmin>/busybox:test4
```

2. 输入以下命令将四个示例镜像（如 busybox ）推送到启用了 auto-pruning 的存储库：

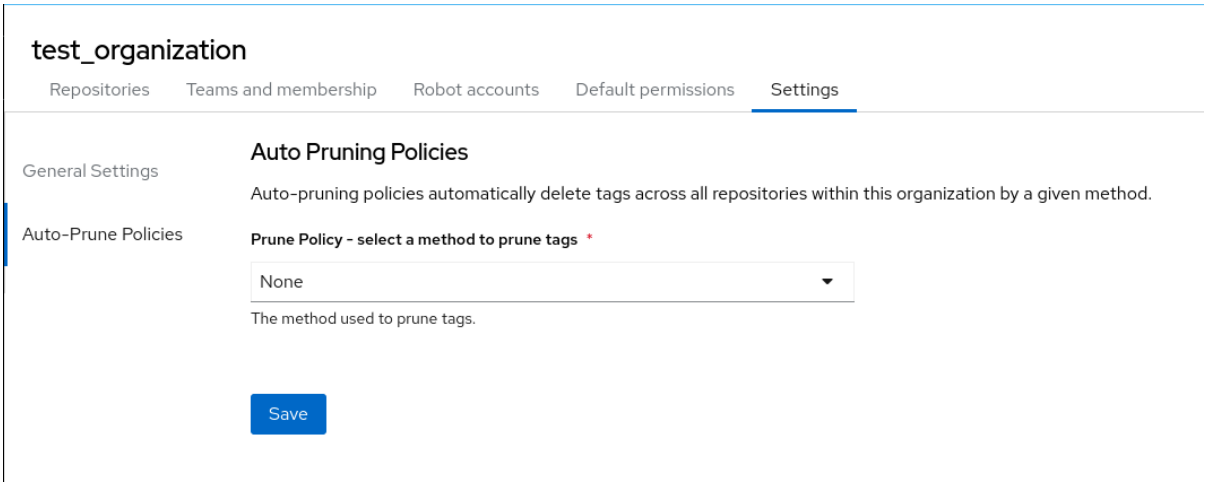
```
$ podman push <quay-server.example.com>/quayadmin/busybox:test
```

```
$ podman push <quay-server.example.com>/<quayadmin>/busybox:test2
```

```
$ podman push <quay-server.example.com>/<quayadmin>/busybox:test3
```

```
$ podman push <quay-server.example.com>/<quayadmin>/busybox:test4
```

3. 检查您的存储库中是否有四个标签。
4. 在 Red Hat Quay v2 UI 上，单击导航窗格中的 Organizations。
5. 选择您要将 auto-pruning 功能应用到的机构名称，如 test_organization。
6. 单击 Settings。
7. 单击 Auto-Prune Policies。例如：



8. 单击下拉菜单并选择所需的策略，例如，按标签数。
9. 选择要保留所需的标签数量。默认情况下，这在 20 个标签中设置。在本例中，要保留的标签数量设置为 3。
10. 点击 Save。此时会出现自动修剪策略的通知。

验证

- 导航到您组织存储库的 Tags 页面。在本例中，标签标记为从标签最旧的创建日期开始删除。几分钟后，auto-pruner worker 会删除不再在所建立的标准中适合的标签。在本例中，它会删除 busybox:test 标签，并保留 busybox:test2、busybox:test3 和 busybox:test4 标签。
在标签被自动修剪后，它们进入 Red Hat Quay 时间机器，或者在标签被删除前删除标签的时间。镜像标签的过期时间取决于您的机构设置。如需更多信息，请参阅 Red Hat Quay 垃圾回收。

15.2.3. 使用 Red Hat Quay API 为命名空间创建自动修剪策略

您可以使用 Red Hat Quay API 端点来管理命名空间的自动修剪策略。

先决条件

- 您已在 config.yaml 文件中设置 BROWSER_API_CALLS_XHR_ONLY: false。
- 您已创建了 OAuth 访问令牌。
- 您已登录到 Red Hat Quay。

流程

1. 输入以下 POST 命令创建一个新策略，该策略限制机构中允许的标签数：

```
$ curl -X POST -H "Authorization: Bearer <access_token>" -H "Content-Type: application/json" -d '{"method": "number_of_tags", "value": 10}' http://<quay-server.example.com>/api/v1/organization/<organization_name>/autoprunepolicy/
```

另外，您可以在创建日期后将标签设置为在指定时间过期：

```
$ curl -X POST -H "Authorization: Bearer <access_token>" -H "Content-Type: application/json" -d '{
```

```
"method": "creation_date", "value": "7d"}' http://<quay-
server.example.com>/api/v1/organization/<organization_name>/autoprunepolicy/
```

输出示例

```
{"uuid": "73d64f05-d587-42d9-af6d-e726a4a80d6e"}
```

尝试创建多个策略会返回以下错误：

```
{"detail": "Policy for this namespace already exists, delete existing to create new
policy", "error_message": "Policy for this namespace already exists, delete existing to
create new policy", "error_type": "invalid_request", "title": "invalid_request", "type":
"http://<quay-server.example.com>/api/v1/error/invalid_request", "status": 400}
```

2. 输入以下命令检查您的自动修剪策略：

```
$ curl -X GET -H "Authorization: Bearer <access_token>" http://<quay-
server.example.com>/api/v1/organization/<organization_name>/autoprunepolicy/
```

输出示例

```
{"policies": [{"uuid": "73d64f05-d587-42d9-af6d-e726a4a80d6e", "method":
"creation_date", "value": "7d"}]}
```

3. 您可以输入以下命令删除自动修剪策略。请注意，删除策略需要 UUID。

```
$ curl -X DELETE -H "Authorization: Bearer <access_token>" http://<quay-
server.example.com>/api/v1/organization/<organization_name>/autoprunepolicy/73d64
f05-d587-42d9-af6d-e726a4a80d6e
```

15.2.4. 使用 API 为当前用户为命名空间创建自动修剪策略

您可以使用 Red Hat Quay API 端点来管理帐户的自动修剪策略。

注意

在以下命令中使用 /user/ 代表当前登录到 Red Hat Quay 的用户。

先决条件

- 您已在 config.yaml 文件中设置 BROWSER_API_CALLS_XHR_ONLY: false。
- 您已创建了 OAuth 访问令牌。
- 您已登录到 Red Hat Quay。

流程

1. 输入以下 POST 命令创建一个新策略，该策略限制当前用户的标签数：

```
$ curl -X POST -H "Authorization: Bearer <access_token>" -H "Content-Type: application/json" -d '{"method": "number_of_tags", "value": 10}' http://<quay-server.example.com>/api/v1/<user>/autoprunepolicy/
```

输出示例

```
{"uuid": "8c03f995-ca6f-4928-b98d-d75ed8c14859"}
```

2. 输入以下命令检查您的自动修剪策略：

```
$ curl -X GET -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/<user>/autoprunepolicy/8c03f995-ca6f-4928-b98d-d75ed8c14859
```

或者，您可以包含 UUID：

```
$ curl -X GET -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/<user>/autoprunepolicy/
```

输出示例

```
{"policies": [{"uuid": "8c03f995-ca6f-4928-b98d-d75ed8c14859", "method": "number_of_tags", "value": 10}]}
```

3. 您可以输入以下命令删除自动修剪策略。请注意，删除策略需要 UUID。

```
$ curl -X DELETE -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/<user>/autoprunepolicy/8c03f995-ca6f-4928-b98d-d75ed8c14859
```

输出示例

```
{"uuid": "8c03f995-ca6f-4928-b98d-d75ed8c14859"}
```

15.2.5. 使用 Red Hat Quay v2 UI 为存储库创建自动修剪策略

使用以下步骤，使用 Red Hat Quay v2 UI 为存储库创建自动修剪策略。

先决条件

- 您已启用了 FEATURE_AUTO_PRUNE 功能。

流程

1. 标记四个示例镜像，如 busybox，它将推送到启用了 auto-pruning 的存储库。例如：

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

```
$ podman tag docker.io/library/busybox <quay-server.example.com>/<organization_name>/<repository_name>:test2
```

```
$ podman tag docker.io/library/busybox <quay-
server.example.com>/<organization_name>/<repository_name>:test3
```

```
$ podman tag docker.io/library/busybox <quay-
server.example.com>/<organization_name>/<repository_name>:test4
```

2. 输入以下命令将四个示例镜像（如 busybox ）推送到启用了 auto-pruning 的存储库：

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

```
$ podman push <quay-
server.example.com>/<organization_name>/<repository_name>:test2
```

```
$ podman push <quay-
server.example.com>/<organization_name>/<repository_name>:test3
```

```
$ podman push <quay-
server.example.com>/<organization_name>/<repository_name>:test4
```

3. 检查您的存储库中是否有四个标签。
4. 在 Red Hat Quay v2 UI 上，单击导航窗格中的 Repository。
5. 选择您要将 auto-pruning 功能应用到的机构名称，例如 <organization_name>/<repository_name>。
6. 单击 Settings。
7. 单击 Repository Auto-Prune Policies。
8. 单击下拉菜单并选择所需的策略，例如，按标签数。
9. 选择要保留所需的标签数量。默认情况下，这在 20 个标签中设置。在本例中，要保留的标签数量设置为 3。
10. 点击 Save。此时会出现自动修剪策略的通知。

验证

- 导航到您组织存储库的 Tags 页面。在本例中，标签标记为从标签最旧的创建日期开始删除。几分钟后，auto-pruner worker 会删除不再在所建立的标准中适合的标签。在本例中，它会删除 busybox:test 标签，并保留 busybox:test2、busybox:test3 和 busybox:test4 标签。在标签被自动修剪后，它们进入 Red Hat Quay 时间机器，或者在标签被删除前删除标签的时间。镜像标签的过期时间取决于您的机构设置。如需更多信息，请参阅 Red Hat Quay 垃圾回收。

15.2.6. 使用 Red Hat Quay API 为存储库创建自动修剪策略

您可以使用 Red Hat Quay API 端点来管理存储库的自动修剪策略。

先决条件

- 您已在 config.yaml 文件中设置 BROWSER_API_CALLS_XHR_ONLY: false。
- 您已创建了 OAuth 访问令牌。
- 您已登录到 Red Hat Quay。

流程

1. 输入以下 POST 命令创建一个新策略，该策略限制机构中允许的标签数：

```
$ curl -X POST -H "Authorization: Bearer <access_token>" -H "Content-Type: application/json" -d '{"method": "number_of_tags","value": 2}' http://<quay-server.example.com>/api/v1/repository/<organization_name>/<repository_name>/autoprunepolicy/
```

另外，您可以在创建日期后将标签设置为在指定时间过期：

```
$ curl -X POST -H "Authorization: Bearer <access_token>" -H "Content-Type: application/json" -d '{"method": "creation_date", "value": "7d"}' http://<quay-server.example.com>/api/v1/repository/<organization_name>/<repository_name>/autoprunepolicy/
```

输出示例

```
{"uuid": "ce2bdcc0-ced2-4a1a-ac36-78a9c1bed8c7"}
```

尝试创建多个策略会返回以下错误：

```
{"detail": "Policy for this namespace already exists, delete existing to create new policy", "error_message": "Policy for this namespace already exists, delete existing to create new policy", "error_type": "invalid_request", "title": "invalid_request", "type": "http://quay-server.example.com/api/v1/error/invalid_request", "status": 400}
```

2. 输入以下命令检查您的自动修剪策略：

```
$ curl -X GET -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/repository/<organization_name>/<repository_name>/autoprunepolicy/
```

或者，您可以包含 UUID：

```
$ curl -X GET -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/repository/<organization_name>/<repository_name>/autoprunepolicy/ce2bdcc0-ced2-4a1a-ac36-78a9c1bed8c7
```

输出示例

```
{"policies": [{"uuid": "ce2bdcc0-ced2-4a1a-ac36-78a9c1bed8c7", "method": "number_of_tags", "value": 10}]}
```

3. 您可以输入以下命令删除自动修剪策略。请注意，删除策略需要 UUID。

```
$ curl -X DELETE -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/repository/<organization_name>/<repository_name>/autoprunepolicy/ce2bdcc0-ced2-4a1a-ac36-78a9c1bed8c7
```

输出示例

```
{"uuid": "ce2bdcc0-ced2-4a1a-ac36-78a9c1bed8c7"}
```

15.2.7. 使用 API 的用户在存储库上创建自动修剪策略

您可以使用 Red Hat Quay API 端点为非您自己的用户帐户管理存储库上的自动修剪策略，只要存储库具有 admin 特权。

先决条件

- 您已在 config.yaml 文件中设置 BROWSER_API_CALLS_XHR_ONLY: false。
- 您已创建了 OAuth 访问令牌。
- 您已登录到 Red Hat Quay。
- 在您要为其创建策略的存储库上具有 admin 权限。

流程

1. 输入以下 POST 命令创建一个新策略，该策略限制当前用户的标签数：

```
$ curl -X POST -H "Authorization: Bearer <access_token>" -H "Content-Type: application/json" -d '{"method": "number_of_tags","value": 2}' http://<quay-server.example.com>/api/v1/repository/<user_account>/<user_repository>/autoprunepolicy/
```

输出示例

```
{"uuid": "7726f79c-cbc7-490e-98dd-becdc6fefce7"}
```

2. 输入以下命令检查您的自动修剪策略：

```
$ curl -X GET -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/repository/<user_account>/<user_repository>/autoprunepolicy/
```

或者，您可以包含 UUID：

```
$ curl -X GET -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/repository/<user_account>/<user_repository>/autoprunepolicy/7726f79c-cbc7-490e-98dd-becdc6fefce7
```

输出示例

```
{"policies": [{"uuid": "7726f79c-cbc7-490e-98dd-becdc6fefce7", "method": "number_of_tags", "value": 2}]}
```

3. **您可以输入以下命令删除自动修剪策略。请注意，删除策略需要 UUID。**

```
$ curl -X DELETE -H "Authorization: Bearer <access_token>" http://<quay-server.example.com>/api/v1/user/autoprunepolicy/7726f79c-cbc7-490e-98dd-becdc6fefce7
```

输**出示例**

```
{"uuid": "7726f79c-cbc7-490e-98dd-becdc6fefce7"}
```

第 16 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。

16.1. 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。**

16.2. 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。

16.2.1. 为独立 Red Hat Quay 启用存储复制

使用以下步骤在 Red Hat Quay 上启用存储复制。

流程

1. 更新 config.yaml 文件，使其包含要复制数据的存储引擎。您必须列出所有要使用的存储引擎：

```
# ...
FEATURE_STORAGE_REPLICATION: true
# ...
DISTRIBUTED_STORAGE_CONFIG:
  usstorage:
    - RHOCSStorage
    - access_key: <access_key>
      bucket_name: <example_bucket>
      hostname: my.noobaa.hostname
      is_secure: false
      port: "443"
      secret_key: <secret_key>
      storage_path: /datastorage/registry
  eustorage:
    - S3Storage
    - host: s3.amazon.com
      port: "443"
      s3_access_key: <access_key>
      s3_bucket: <example bucket>
      s3_secret_key: <secret_key>
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
```

```
    - usstorage
    - eustorage
# ...
```

2. 可选。如果需要将所有镜像复制到所有存储引擎，您可以通过手动设置 DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS 字段来将镜像复制到存储引擎。这可确保所有镜像复制到该存储引擎。例如：

```
# ...
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS:
  - usstorage
  - eustorage
# ...
```

注意

要启用每个命名空间复制，请联系红帽 Quay 支持。

3. 为异地复制 默认将存储并启用 Replicate 到存储引擎 后，您必须在所有存储间同步现有镜像数据。要做到这一点，您必须运行以下命令来在容器中执行：

```
$ podman exec -it <container_id>
```

4. 要在添加新存储后同步内容，请输入以下命令：

```
# scl enable python27 bash
```

```
# python -m util.backfillreplication
```

注意

这是添加新存储后同步内容的一次性操作。

16.2.2. 使用存储首选项运行 Red Hat Quay

1. 将 config.yaml 复制到运行 Red Hat Quay 的所有机器

2. 对于每个区域中的每个机器，使用机器运行的区域的首选存储引擎添加 QUAY_DISTRIBUTED_STORAGE_PREFERENCE 环境变量。
例如，对于在欧洲运行的机器，主机上有 $QUAY/config 的配置目录：

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443  \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -e QUAY_DISTRIBUTED_STORAGE_PREFERENCE=europestorage \
  registry.redhat.io/quay/quay-rhel8:v3.11.1
```

注意

指定的环境变量值必须与 config 面板中定义的 Location ID 的名称匹配。

3. 重启所有 Red Hat Quay 容器

16.2.3. 从独立的 Red Hat Quay 部署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

- 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。在继续操作前，请完成此步骤。

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

3. 输入以下命令获取正在运行的容器列表：

```
$ podman ps
```

输出示例

```
CONTAINER ID  IMAGE                                                  COMMAND
CREATED        STATUS            PORTS                           NAMES
92c5321cde38  registry.redhat.io/rhel8/redis-5:1                      run-redis      11
days ago     Up 11 days ago     0.0.0.0:6379->6379/tcp              redis
4e6d1ecd3811  registry.redhat.io/rhel8/postgresql-13:1-109            run-
postgresql  33 seconds ago  Up 34 seconds ago  0.0.0.0:5432->5432/tcp
postgresql-quay
d2eadac74fda  registry-proxy.engineering.redhat.com/rh-osbs/quay-quay-rhel8:v3.9.0-
131  registry       4 seconds ago   Up 4 seconds ago   0.0.0.0:80->8080/tcp, 0.0.0.0:443-
>8443/tcp  quay
```

4. 输入以下命令在 PostgreSQL 容器内执行 shell：

```
$ podman exec -it postgresql-quay -- /bin/bash
```

5. 运行以下命令来输入 psql ：

```
bash-4.4$ psql
```

6. 输入以下命令在 geo-replicated 部署中显示站点列表：

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

输出示例

```
 id |      name
----+-----------------
  1 | usstorage
  2 | eustorage
```

7. 输入以下命令退出 postgres CLI 以重新输入 bash-4.4 ：

```
\q
```

8. 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。

```
bash-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
```

16.2.4. 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。

16.2.4.1. 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。

16.2.5. 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 Quayconfig.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
```

16.3. 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。

第 17 章 在独立部署中备份和恢复 RED HAT QUAY

使用本节中的内容在独立部署中备份和恢复 Red Hat Quay。

17.1. 可选：为 RED HAT QUAY 启用只读模式

通过为 Red Hat Quay 部署启用只读模式，您可以管理 registry 的操作。Red Hat Quay 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。

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。

17.1.1. 为独立 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. 如果您的 Red Hat Quay registry 可用，您可以在 Quay registry 容器内生成服务密钥。
   a. 输入以下命令在 Quay 容器内生成密钥对：

   ```
   $ podman exec quay python3 tools/generatekeypair.py quay-readonly
   ```

2. 如果您的 Red Hat Quay 不可用，您必须在虚拟环境内生成您的服务密钥。
   a. 进到 Red Hat Quay 部署的目录，并在该目录内创建一个虚拟环境：

   ```
   $ cd <$QUAY>/quay && virtualenv -v venv
   ```

   b. 输入以下命令激活虚拟环境：

   ```
   $ source venv/bin/activate
   ```

c. 可选。如果没有安装 pip CLI 工具：

```
$ venv/bin/pip install --upgrade pip
```

d. 在 Red Hat Quay 目录中，创建一个包含以下内容的 requirements-generatekeys.txt 文件：

```
$ cat << EOF > requirements-generatekeys.txt
cryptography==3.4.7
pycparser==2.19
pycryptodome==3.9.4
pycryptodomex==3.9.4
pyjwkest==1.4.2
PyJWT==1.7.1
Authlib==1.0.0a2
EOF
```

e. 输入以下命令安装 requirements-generatekeys.txt 文件中定义的 Python 依赖项：

```
$ venv/bin/pip install -r requirements-generatekeys.txt
```

f. 输入以下命令来创建所需的服务密钥：

```
$ PYTHONPATH=. venv/bin/python
/<path_to_cloned_repo>/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
```

g. 输入以下命令取消激活虚拟环境：

```
$ deactivate
```

17.1.2. 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：

```
$ podman exec -it postgresql-quay psql -U postgres -d quay
```

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
```

17.1.3. 为独立 Red Hat Quay 配置只读模式

在创建了服务密钥并添加到 PostgreSQL 数据库后，您必须在独立部署上重启 Quay 容器。

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. 关闭所有虚拟机上的所有 Red Hat Quay 实例。例如：

```
$ podman stop <quay_container_name_on_virtual_machine_a>
```

```
$ podman stop <quay_container_name_on_virtual_machine_b>
```

2. 输入以下命令将 quay-readonly.kid 文件的内容和 quay-readonly.pem 文件复制到包含 Red Hat Quay 配置捆绑包的目录中：

```
$ cp quay-readonly.kid quay-readonly.pem $Quay/config
```

3. 输入以下命令在配置捆绑包文件夹中的所有文件中设置文件权限：

```
$ setfacl -m user:1001:rw $Quay/config/*
```

4. 修改 Red Hat Quay 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. 将新配置捆绑包分发到所有 Red Hat Quay 实例。

6. 输入以下命令启动 Red Hat Quay：

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

7. 启动 Red Hat Quay 后，您的实例中内的横幅会告知用户 Red Hat Quay 以只读模式运行。推送应被拒绝，并且应记录 405 错误。您可以运行以下命令来测试它：

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

输出示例

```
613be09ab3c0: Preparing
denied: System is currently read-only. Pulls will succeed but all write operations are
currently suspended.
```

在只读模式下使用 Red Hat Quay 部署，您可以安全地管理 registry 的操作并执行此类操作，如备份和恢复。

8. 可选。完成只读模式后，您可以通过从 config.yaml 文件中删除以下信息来返回到正常操作。然后，重启 Red Hat Quay 部署：

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

```
$ podman restart <container_id>
```

17.1.4. 更新只读过期时间

Red Hat Quay read-only 键具有过期日期，并在该日期通过密钥时停用。在密钥过期前，可以在数据库中更新其过期时间。要更新密钥，请使用前面描述的方法连接 Red Hat Quay production 数据库，并发出以下查询：

```
quay=# UPDATE servicekey SET expiration_date = 'new-date' WHERE id = servicekey_id;
```

可以通过运行以下查询来获取服务密钥 ID 列表：

```
SELECT id, name, expiration_date FROM servicekey;
```

17.2. 在独立部署中备份 RED HAT QUAY

此流程描述了如何在独立部署中创建 Red Hat Quay 备份。

流程

1. 创建一个临时备份目录，如 quay-backup ：

```
$ mkdir /tmp/quay-backup
```

2. 以下示例命令表示 Red Hat Quay 启动的本地目录，例如 /opt/quay-install ：

```
$ podman run --name quay-app \
  -v /opt/quay-install/config:/conf/stack:Z \
  -v /opt/quay-install/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1
```

运行以下命令，进入绑定挂载到容器内的 /conf/stack 的目录，如 /opt/quay-install ：

```
$ cd /opt/quay-install
```

3. 输入以下命令将 Red Hat Quay 部署的内容压缩到 quay-backup 目录中的存档中：

```
$ tar cvf /tmp/quay-backup/quay-backup.tar.gz *
```

输出示例：

```
config.yaml
```

```
config.yaml.bak
extra_ca_certs/
extra_ca_certs/ca.crt
ssl.cert
ssl.key
```

4. 输入以下命令备份 Quay 容器服务：

```
$ podman inspect quay-app | jq -r '.[0].Config.CreateCommand | .[]' | paste -s -d ' ' -

 /usr/bin/podman run --name quay-app \
 -v /opt/quay-install/config:/conf/stack:Z \
 -v /opt/quay-install/storage:/datastorage:Z \
 registry.redhat.io/quay/quay-rhel8:v3.11.1
```

5. 输入以下命令将 conf/stack/config.yaml 文件的内容重定向到临时 quay-config.yaml 文件中：

```
$ podman exec -it quay cat /conf/stack/config.yaml > /tmp/quay-backup/quay-config.yaml
```

6. 输入以下命令来获取位于临时 quay-config.yaml 中的 DB_URI ：

```
$ grep DB_URI /tmp/quay-backup/quay-config.yaml
```

输出示例：

```
$ postgresql://<username>:test123@172.24.10.50/quay
```

7. 输入以下命令将 PostgreSQL 内容提取到备份 .sql 文件中的临时备份目录中：

```
$ pg_dump -h 172.24.10.50  -p 5432 -d quay  -U  <username>   -W -O > /tmp/quay-backup/quay-backup.sql
```

8. 输入以下命令输出 DISTRIBUTED_STORAGE_CONFIG 的内容：

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
   - S3Storage
   - s3_bucket: <bucket_name>
     storage_path: /registry
     s3_access_key: <s3_access_key>
     s3_secret_key: <s3_secret_key>
     host: <host_name>
```

9. 使用在第 7 步中获取的 access_key 凭证导出 AWS_ACCESS_KEY_ID ：

```
$ export AWS_ACCESS_KEY_ID=<access_key>
```

10. 使用在第 7 步中获取的 secret_key 导出 AWS_SECRET_ACCESS_KEY ：

```
$ export AWS_SECRET_ACCESS_KEY=<secret_key>
```

11. **将 quay 存储桶同步到来自您的 DISTRIBUTED_STORAGE_CONFIG 的 hostname 的 /tmp/quay-backup/blob-backup/ 目录：**

```
$ aws s3 sync s3://<bucket_name>  /tmp/quay-backup/blob-backup/ --source-region us-east-2
```

输出示例：

```
download: s3://<user_name>/registry/sha256/9c/9c3181779a868e09698b567a3c42f3744584ddb1398efe2c4ba569a99b823f7a to registry/sha256/9c/9c3181779a868e09698b567a3c42f3744584ddb1398efe2c4ba569a99b823f7a
download: s3://<user_name>/registry/sha256/e9/e9c5463f15f0fd62df3898b36ace8d15386a6813ffb470f332698ecb34af5b0d to registry/sha256/e9/e9c5463f15f0fd62df3898b36ace8d15386a6813ffb470f332698ecb34af5b0d
```

建议您在同步 quay 存储桶后删除 quay-config.yaml 文件，因为它包含敏感信息。quay-config.yaml 文件不会丢失，因为它在 quay-backup.tar.gz 文件中备份。

17.3. 在独立部署中恢复 RED HAT QUAY

此流程描述了如何在独立部署中恢复 Red Hat Quay。

先决条件

- **您已备份了 Red Hat Quay 部署。**

流程

1. **创建一个新目录，它将绑定挂载到 Red Hat Quay 容器内的 /conf/stack ：**

```
$ mkdir /opt/new-quay-install
```

2. **将在 独立部署上备份 Red Hat Quay 中创建的临时备份目录的内容复制到在第 1 步中创建的 new-quay-install1 目录：**

```
$ cp /tmp/quay-backup/quay-backup.tar.gz /opt/new-quay-install/
```

3. **输入以下命令进入 new-quay-install 目录：**

```
$ cd /opt/new-quay-install/
```

4. **提取 Red Hat Quay 目录的内容：**

```
$ tar xvf /tmp/quay-backup/quay-backup.tar.gz *
```

输出示例：

```
config.yaml
config.yaml.bak
```

```
extra_ca_certs/
extra_ca_certs/ca.crt
ssl.cert
ssl.key
```

5. 输入以下命令从备份的 config.yaml 文件中重新调用 DB_URI ：

```
$ grep DB_URI config.yaml
```

输出示例：

```
postgresql://<username>:test123@172.24.10.50/quay
```

6. 运行以下命令来输入 PostgreSQL 数据库服务器：

```
$ sudo postgres
```

7. 输入 psql 并在 172.24.10.50 中创建一个新数据库，以恢复 quay 数据库，例如 example_restore_registry_quay_database，请输入以下命令：

```
$ psql "host=172.24.10.50  port=5432 dbname=postgres user=<username>
password=test123"
postgres=> CREATE DATABASE example_restore_registry_quay_database;
```

输出示例：

```
CREATE DATABASE
```

8. 运行以下命令来连接到数据库：

```
postgres=# \c "example-restore-registry-quay-database";
```

输出示例：

```
You are now connected to database "example-restore-registry-quay-database" as user
"postgres".
```

9. 运行以下命令，为您的 Quay 数据库创建一个 pg_trmg 扩展：

```
example_restore_registry_quay_database=> CREATE EXTENSION IF NOT EXISTS
pg_trgm;
```

输出示例：

```
CREATE EXTENSION
```

10. 输入以下命令退出 postgres CLI：

```
\q
```

11. 运行以下命令，将数据库备份导入到您的新数据库中：

```
$ psql "host=172.24.10.50 port=5432
dbname=example_restore_registry_quay_database user=<username>
password=test123"  -W <  /tmp/quay-backup/quay-backup.sql
```

输出示例：

```
SET
SET
SET
SET
SET
```

在重启 Red Hat Quay 部署前，将 config.yaml 中的 DB_URI 的值从postgresql://<username>:test123@172.24.10.50/quay 更新到 postgresql://<username>:test123@172.24.10.50/example-restore-registry-quay-database。

注意

DB_URI 格式为 DB_ URI postgresql://<login_user_name>:<login_user_password>@<postgresql_host>/<quay_database>。如果您要从一个 PostgreSQL 服务器移到另一个 PostgreSQL 服务器，请同时更新 < login_user_name>、< login_user_password> 和 <postgresql_host > 的值。

12. 在 /opt/new-quay-install 目录中，打印 DISTRIBUTED_STORAGE_CONFIG 捆绑包的内容：

```
$ cat config.yaml | grep DISTRIBUTED_STORAGE_CONFIG -A10
```

输出示例：

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
DISTRIBUTED_STORAGE_CONFIG:
  default:
   - S3Storage
   - s3_bucket: <bucket_name>
     storage_path: /registry
     s3_access_key: <s3_access_key>
     s3_secret_key: <s3_secret_key>
     host: <host_name>
```

注意

在重启 Red Hat Quay 部署前，必须更新 /opt/new-quay-install 中的 DISTRIBUTED_STORAGE_CONFIG。

13. 使用在第 13 步中获取的 access_key 凭证导出 AWS_ACCESS_KEY_ID ：

```
$ export AWS_ACCESS_KEY_ID=<access_key>
```

14. 使用在第 13 步中获取的 secret_key 导出 AWS_SECRET_ACCESS_KEY ：

```
$ export AWS_SECRET_ACCESS_KEY=<secret_key>
```

15. 输入以下命令创建新 s3 存储桶：

```
$ aws s3 mb s3://<new_bucket_name>  --region us-east-2
```

输出示例：

```
$ make_bucket: quay
```

16. 输入以下命令将所有 Blob 上传到新的 s3 存储桶：

```
$ aws s3 sync --no-verify-ssl \
--endpoint-url <example_endpoint_url> (1)
/tmp/quay-backup/blob-backup/. s3://quay/
```

(1) 在备份和恢复前，Red Hat Quay registry 端点必须相同。

输出示例：

```
upload: ../../tmp/quay-backup/blob-backup/datastorage/registry/sha256/50/505edb46ea5d32b5cbe275eb766d960842a52ee77ac225e4dc8abb12f409a30d to s3://quay/datastorage/registry/sha256/50/505edb46ea5d32b5cbe275eb766d960842a52ee77ac225e4dc8abb12f409a30d
upload: ../../tmp/quay-backup/blob-backup/datastorage/registry/sha256/27/27930dc06c2ee27ac6f543ba0e93640dd21eea458eac47355e8e5989dea087d0 to s3://quay/datastorage/registry/sha256/27/27930dc06c2ee27ac6f543ba0e93640dd21eea458eac47355e8e5989dea087d0
upload: ../../tmp/quay-backup/blob-backup/datastorage/registry/sha256/8c/8c7daf5e20eee45ffe4b36761c4bb6729fb3ee60d4f588f712989939323110ec to s3://quay/datastorage/registry/sha256/8c/8c7daf5e20eee45ffe4b36761c4bb6729fb3ee60d4f588f712989939323110ec
...
```

17. 在重启 Red Hat Quay 部署前，更新 config.yaml 中的存储设置：

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
DISTRIBUTED_STORAGE_CONFIG:
  default:
   - S3Storage
   - s3_bucket: <new_bucket_name>
     storage_path: /registry
     s3_access_key: <s3_access_key>
     s3_secret_key: <s3_secret_key>
     host: <host_name>
```

第 18 章 将独立 RED HAT QUAY 部署迁移到 RED HAT QUAY OPERATOR 部署

以下流程允许您备份独立 Red Hat Quay 部署，并将其迁移到 OpenShift Container Platform 上的 Red Hat Quay Operator。

18.1. 备份 RED HAT QUAY 的独立部署

流程

1. 备份独立 Red Hat Quay 部署的 config.yaml ：

```
$ mkdir /tmp/quay-backup
$ cp /path/to/Quay/config/directory/config.yaml /tmp/quay-backup
```

2. 创建独立 Red Hat Quay 部署正在使用的数据库备份：

```
$ pg_dump -h DB_HOST -p 5432 -d QUAY_DATABASE_NAME -U
QUAY_DATABASE_USER -W -O > /tmp/quay-backup/quay-database-backup.sql
```

3. 如果还没有安装 AWS CLI，请进行安装。

4. 创建 ~/.aws/ 目录：

```
$ mkdir ~/.aws/
```

5. 从独立部署的 config.yaml 获取 access_key 和 secret_key ：

```
$ grep -i DISTRIBUTED_STORAGE_CONFIG -A10 /tmp/quay-backup/config.yaml
```

输出示例：

```
DISTRIBUTED_STORAGE_CONFIG:
    minio-1:
      - RadosGWStorage
      - access_key: ##########
        bucket_name: quay
        hostname: 172.24.10.50
        is_secure: false
        port: "9000"
        secret_key: ##########
        storage_path: /datastorage/registry
```

6. 将 config.yaml 文件中的 access_key 和 secret_key 存储在 ~/.aws 目录中：

```
$ touch ~/.aws/credentials
```

7. 可选：检查您的 access_key 和 secret_key 是否已存储：

```
$ cat > ~/.aws/credentials << EOF
[default]
aws_access_key_id = ACCESS_KEY_FROM_QUAY_CONFIG
```

```
aws_secret_access_key = SECRET_KEY_FROM_QUAY_CONFIG
EOF
```

输出示例：

```
aws_access_key_id = ACCESS_KEY_FROM_QUAY_CONFIG
aws_secret_access_key = SECRET_KEY_FROM_QUAY_CONFIG
```

注意

如果 aws cli 没有从 '~/.aws/credentials 文件 自动收集 access_key 和 secret_key，您可以通过运行 aws 配置并手动输入凭证来配置 它们。

8. 在 quay-backup 目录中，创建一个 bucket_backup 目录：

```
$ mkdir /tmp/quay-backup/bucket-backup
```

9. 从 S3 存储备份所有 Blob：

```
$ aws s3 sync --no-verify-ssl --endpoint-url https://PUBLIC_S3_ENDPOINT:PORT
s3://QUAY_BUCKET/ /tmp/quay-backup/bucket-backup/
```

注意

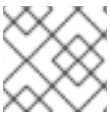
PUBLIC_S3_ENDPOINT 可以从位于 DISTRIBUTED_STORAGE_CONFIG 的主机名 下的 Red Hat Quayconfig.yaml 文件中读取。如果端点不安全，请在端点 URL 中使用 http 而不是 https。

此时，您应该对本地存储的所有 Red Hat Quay 数据、blob、数据库和 config.yaml 文件的完整备份。在以下部分中，您要将独立部署备份迁移到 OpenShift Container Platform 上的 Red Hat Quay。

18.2. 使用备份的独立内容迁移到 OPENSHIFT CONTAINER PLATFORM。

先决条件

- 您的独立 Red Hat Quay 数据、blob、数据库和 config.yaml 已被备份。
- 使用 Red Hat Quay Operator 在 OpenShift Container Platform 上部署 Red Hat Quay。
- 将带有所有组件的 QuayRegistry 设置为 managed。



流程

本文档中的步骤使用以下命名空间： quay-enterprise。

1. 缩减 Red Hat Quay Operator：

```
$ oc scale --replicas=0 deployment quay-operator.v3.6.2 -n openshift-operators
```

2. 缩减应用程序和镜像部署：

```
$ oc scale --replicas=0 deployment QUAY_MAIN_APP_DEPLOYMENT
QUAY_MIRROR_DEPLOYMENT
```

3. 将数据库 SQL 备份复制到 Quay PostgreSQL 数据库实例中：

```
$ oc cp /tmp/user/quay-backup/quay-database-backup.sql quay-
enterprise/quayregistry-quay-database-54956cdd54-p7b2w:/var/lib/pgsql/data/userdata
```

4. 从 Operator 创建的 config.yaml 文件获取数据库密码：

```
$ oc get deployment quay-quay-app -o json | jq
'.spec.template.spec.volumes[].projected.sources' | grep -i config-secret
```

输出示例：

```
      "name": "QUAY_CONFIG_SECRET_NAME"
```

```
$ oc get secret quay-quay-config-secret-9t77hb84tb -o json | jq '.data."config.yaml"' |
cut -d '"' -f2 | base64 -d -w0 > /tmp/quay-backup/operator-quay-config-yaml-
backup.yaml
```

```
cat /tmp/quay-backup/operator-quay-config-yaml-backup.yaml | grep -i DB_URI
```

输出示例：

```
postgresql://QUAY_DATABASE_OWNER:PASSWORD@DATABASE_HOST/QUAY_DATAB
ASE_NAME
```

5. 在数据库 pod 中执行 shell：

```
# oc exec -it quay-postgresql-database-pod -- /bin/bash
```

6. 输入 psql ：

```
bash-4.4$ psql
```

7. 丢弃数据库：

```
postgres=# DROP DATABASE "example-restore-registry-quay-database";
```

输出示例：

```
DROP DATABASE
```

8. 创建新数据库并将所有者设置为相同的名称：

```
postgres=# CREATE DATABASE "example-restore-registry-quay-database" OWNER
"example-restore-registry-quay-database";
```

输出示例：

```
CREATE DATABASE
```

9. 连接到数据库：

```
postgres=# \c "example-restore-registry-quay-database";
```

输出示例：

```
You are now connected to database "example-restore-registry-quay-database" as user
"postgres".
```

10. 为您的 Quay 数据库创建一个 pg_trmg 扩展：

```
example-restore-registry-quay-database=# create extension pg_trgm ;
```

输出示例：

```
CREATE EXTENSION
```

11. 退出 postgres CLI 以重新输入 bash-4.4：

```
\q
```

12. 为您的 PostgreSQL 部署设置密码：

```
bash-4.4$ psql -h localhost -d "QUAY_DATABASE_NAME" -U
QUAY_DATABASE_OWNER -W < /var/lib/pgsql/data/userdata/quay-database-
backup.sql
```

输出示例：

```
SET
SET
SET
SET
SET
```

13. 退出 bash 模式：

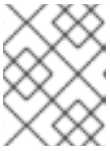
```
bash-4.4$ exit
```

14. 为 Red Hat Quay Operator 创建新的配置捆绑包。

```
$ touch config-bundle.yaml
```

15. 在新的 config-bundle.yaml 中，包含 registry 所需的所有信息，如 LDAP 配置、密钥和其他旧 registry 拥有的修改。运行以下命令，将 secret_key 移到 config-bundle.yaml 中：

```
$ cat /tmp/quay-backup/config.yaml | grep SECRET_KEY > /tmp/quay-backup/config-
bundle.yaml
```



注意

您必须手动复制所有 LDAP、OIDC 和其他信息，并将其添加到 /tmp/quay-backup/config-bundle.yaml 文件中。

16. **在 OpenShift 集群内创建配置捆绑包 secret：**

```
$ oc create secret generic new-custom-config-bundle --from-
file=config.yaml=/tmp/quay-backup/config-bundle.yaml
```

17. **扩展 Quay pod:**

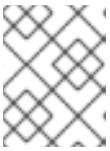
```
$ oc scale --replicas=1 deployment quayregistry-quay-app
deployment.apps/quayregistry-quay-app scaled
```

18. **扩展镜像 pod：**

```
$ oc scale --replicas=1  deployment quayregistry-quay-mirror
deployment.apps/quayregistry-quay-mirror scaled
```

19. **对 QuayRegistry CRD 进行补丁，使其包含对新的自定义配置捆绑包的引用：**

```
$ oc patch quayregistry QUAY_REGISTRY_NAME --type=merge -p '{"spec":
{"configBundleSecret":"new-custom-config-bundle"}}'
```



注意

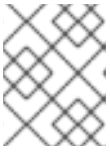
如果 Red Hat Quay 返回 500 内部服务器错误，您可能需要将 DISTRIBUTED_STORAGE_CONFIG 的位置 更新为 默认值。

20. **在 /.aws/ 目录中创建一个新的 AWS credentials.yaml，并包含 Operator 创建的 config.yaml 文件中的 access_key 和 secret_key ：**

```
$ touch credentials.yaml
```

```
$ grep -i DISTRIBUTED_STORAGE_CONFIG -A10 /tmp/quay-backup/operator-quay-
config-yaml-backup.yaml
```

```
$ cat > ~/.aws/credentials << EOF
[default]
aws_access_key_id = ACCESS_KEY_FROM_QUAY_CONFIG
aws_secret_access_key = SECRET_KEY_FROM_QUAY_CONFIG
EOF
```



注意

如果 aws cli 没有从 '~/.aws/credentials 文件 自动收集 access_key 和 secret_key，您可以通过运行 aws 配置并手动输入凭证来配置 它们。

21. **记录 NooBaa 的公开可用端点：**

```
$ oc get route s3 -n openshift-storage -o yaml -o jsonpath="{.spec.host}{'\n'}"
```

22. 将备份数据同步到 NooBaa 后端存储：

```
$ aws s3 sync --no-verify-ssl --endpoint-url https://NOOBAA_PUBLIC_S3_ROUTE
/tmp/quay-backup/bucket-backup/* s3://QUAY_DATASTORE_BUCKET_NAME
```

23. 将 Operator 扩展至 1 个 pod：

```
$ oc scale –replicas=1 deployment quay-operator.v3.6.4 -n openshift-operators
```

Operator 使用所提供的自定义配置捆绑包，并协调所有 secret 和部署。OpenShift Container Platform 上新的 Red Hat Quay 部署应包含旧部署具有的所有信息。您应能够拉取所有镜像。

第 19 章 配置工件类型

作为 Red Hat Quay 管理员，您可以通过 FEATURE_GENERAL_OCI_SUPPORT、ALLOWED_OCI_ARTIFACT_TYPES 和 IGNORE_UNKNOWN_MEDIATYPES 配置字段来配置 Open Container Initiative (OCI)工件类型。

以下 Open Container Initiative (OCI)工件类型默认内置在 Red Hat Quay 中，并通过 FEATURE_GENERAL_OCI_SUPPORT 配置字段启用：

字段	介质类型	支持的内容类型
Helm	application/vnd.cncf.helm.config.v1+json	application/tar+gzip, application/vnd.cncf.helm.chart.content.v1.tar+gzip
Cosign	application/vnd.oci.image.config.v1+json	application/vnd.dev.cosign.simplesigning.v1+json,application/vnd.dsse.envelope.v1+json
SPDX	application/vnd.oci.image.config.v1+json	text/spdx, text/spdx+xml, text/spdx+json
Syft	application/vnd.oci.image.config.v1+json	application/vnd.syft+json
CycloneDX	application/vnd.oci.image.config.v1+json	application/vnd.cyclonedx,application/vnd.cyclonedx+xml,application/vnd.cyclonedx+json
in-toto	application/vnd.oci.image.config.v1+json	application/vnd.in-toto+json
Unknown	application/vnd.cncf.openpolicyagent.policy.layer.v1+rego	application/vnd.cncf.openpolicyagent.policy.layer.v1+rego,application/vnd.cncf.openpolicyagent.data.layer.v1+json

另外，Red Hat Quay 使用 *ZStandard* 或 *zstd* 来缩小容器镜像的大小或其他相关工件。zstd 有助于优化存储并改进使用容器镜像的传输速度。

使用以下步骤配置对默认和实验性 OCI 介质类型的支持。

19.1. 配置 OCI 工件类型

使用以下步骤配置默认嵌入在 Red Hat Quay 中的工件类型。

先决条件

- 您有 Red Hat Quay 管理员特权。

流程

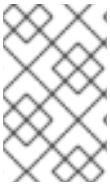
- 在 Red Hat Quay config.yaml 文件中，通过将 FEATURE_GENERAL_OCI_SUPPORT 字段设置为 true 来启用对常规 OCI 支持的支持。例如：

```
FEATURE_GENERAL_OCI_SUPPORT: true
```

将 FEATURE_GENERAL_OCI_SUPPORT 设置为 true 时，Red Hat Quay 用户现在可以将默认工件类型的推送和拉取到其 Red Hat Quay 部署中。

19.2. 配置额外的工件类型

使用以下步骤为您的 Red Hat Quay 部署配置额外的特定工件类型。



注意

使用 ALLOWED_OCI_ARTIFACT_TYPES 配置字段，您可以限制 Red Hat Quay registry 接受哪些工件类型。如果您希望 Red Hat Quay 部署接受所有工件类型，请参阅"配置未知介质类型"。

先决条件

- 您有 Red Hat Quay 管理员特权。

流程

- 添加 ALLOWED_OCI_ARTIFACT_TYPES 配置字段，以及配置和层类型：

```
FEATURE_GENERAL_OCI_SUPPORT: true
ALLOWED_OCI_ARTIFACT_TYPES:
  <oci config type 1>:
  - <oci layer type 1>
  - <oci layer type 2>

  <oci config type 2>:
  - <oci layer type 3>
  - <oci layer type 4>
```

例如，您可以通过在 config.yaml 文件中添加以下内容来添加单性镜像格式(SIF)支持：

```
ALLOWED_OCI_ARTIFACT_TYPES:
  application/vnd.oci.image.config.v1+json:
  - application/vnd.dev.cosign.simplesigning.v1+json
  application/vnd.cncf.helm.config.v1+json:
```

```
- application/tar+gzip
application/vnd.sylabs.sif.config.v1+json:
- application/vnd.sylabs.sif.layer.v1+tar
```

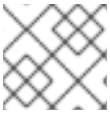
注意

当添加不默认配置的 OCI 工件类型时，Red Hat Quay 管理员还需要手动添加对 Cosign 和 Helm 的支持。

现在，用户可以为其 Red Hat Quay registry 标记 SIF 镜像。

19.3. 配置未知介质类型

使用以下步骤为您的 Red Hat Quay 部署启用所有工件类型。



注意

启用此字段后，您的 Red Hat Quay 部署接受所有工件类型。

先决条件

- 您有 Red Hat Quay 管理员特权。

流程

1. 将 IGNORE_UNKNOWN_MEDIATYPES 配置字段添加到 Red Hat Quayconfig.yaml 文件中：

```
IGNORE_UNKNOWN_MEDIATYPES: true
```

启用此字段后，您的 Red Hat Quay 部署接受未知和未识别的工件类型。

第 20 章 RED HAT QUAY 垃圾回收

Red Hat Quay 包括自动和持续镜像垃圾回收。垃圾回收通过删除占用大量磁盘空间的对象（如悬停或未标记的镜像、存储库和 blob）来有效地将资源用于活跃对象，包括层和清单。Red Hat Quay 执行的垃圾回收可减少机构环境中的停机时间。

20.1. 实践中的 RED HAT QUAY 垃圾回收

目前，所有垃圾回收都会意外地发生，没有命令来手动运行垃圾回收。Red Hat Quay 提供了跟踪不同垃圾回收 worker 状态的指标。

对于命名空间和存储库垃圾回收，会根据对应队列的大小跟踪进度。命名空间和存储库垃圾回收工作需要全局锁定。因此，由于性能的原因，一次只能运行一个 worker。

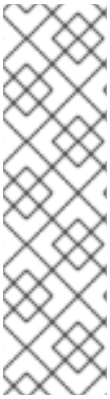
注意

Red Hat Quay 在命名空间和存储库之间共享 blob，以节省磁盘空间。例如，如果同一镜像被推送 10 次，则仅存储该镜像的一个副本。

标签可以与已在 Red Hat Quay 中的某个位置存储的不同镜像共享其层。在这种情况下，Blob 将保留在存储中，因为删除共享 Blob 会使其他镜像不可用。

blob 过期时间独立于时间机器。如果您将标签推送到 Red Hat Quay，并且时间机器被设置为 0 秒，然后您立即删除标签，垃圾回收会删除与该标签相关的标签和所有标签，但不会删除 blob 存储，直到达到 blob 过期时间为止。

垃圾回收标记的镜像的工作方式与命名空间或仓库上的垃圾回收不同。对于标记的镜像，垃圾回收工作者不会强制搜索带有不活跃或过期标签的存储库进行清理。垃圾回收 worker 的每个实例都会获取存储库锁定，这会为每个存储库生成一个 worker。



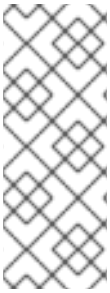
注意

- 在 Red Hat Quay 中，不活跃或已过期的标签是没有标签的清单，因为最后一个标签已被删除或已过期。清单存储有关如何组成镜像并存储到每个标签的数据库的信息。当删除标签并且满足 Time Machine 中的分配时间时，Red Hat Quay 垃圾回收了没有连接到 registry 中任何其他清单的 Blob。如果特定的 blob 连接到清单，则会在存储中保留它，并只删除它与要删除的清单的连接。
- 过期的镜像将在分配的时间后消失，但仍然存储在 Red Hat Quay 中。镜像被完全删除或收集的时间取决于机构的 Time Machine 设置。垃圾回收的默认时间为 14 天，除非另有指定。在此时间之前，标签可以指向过期或删除的镜像。

对于每种类型的垃圾回收，Red Hat Quay 为每个垃圾回收 worker 删除的每个表的行数提供指标。下图显示了 Red Hat Quay 如何监控具有相同指标的垃圾回收示例：

20.1.1. 测量存储回收

Red Hat Quay 无法跟踪垃圾回收释放的空间量。目前，这的最佳指示器是通过检查提供的指标中已删除多少个 Blob。



注意

Red Hat Quay 指标中的 UploadedBlob 表跟踪与存储库关联的各种 Blob。上传 Blob 时，它不会在 PUSH_TEMP_TAG_EXPIRATION_SEC 参数指定的时间之前收集垃圾回收。这是为了避免预先删除作为持续推送一部分的 Blob。例如，如果将垃圾回收设置为经常运行，并且标签在不到一小时的 span 中被删除，则相关的 Blob 可能不会立即清理。相反，假设由 PUSH_TEMP_TAG_EXPIRATION_SEC 参数指定的时间已通过，则相关的 Blob 将由同一存储库上的另一个过期标签触发。

20.2. 垃圾回收配置字段

以下配置字段可用于自定义垃圾收集的内容，以及发生垃圾回收的频率：

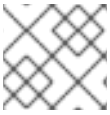
Name	描述	模式
FEATURE_GARBAGE_COLLECTION	是否为镜像标签启用垃圾回收。默认值为 **true**。	布尔值
FEATURE_NAMESPACE_GARBAGE_COLLECTION	是否为命名空间启用垃圾回收。默认值为 **true**。	布尔值

Name	描述	模式
FEATURE_REPOSITORY_GARBAGE_COLLECTION	是否为软件仓库启用垃圾回收。默认值为 **true**。	布尔值
GARBAGE_COLLECTION_FREQUENCY	垃圾回收 worker 运行的频率（以秒为单位）。仅影响垃圾回收工作程序。默认值为 30 秒。	字符串
PUSH_TEMP_TAG_EXPIRATION_SEC	上传后不会收集 blob 的秒数。此功能可防止垃圾回收清理尚未引用的 Blob，但仍然被用作持续推送的一部分。	字符串
TAG_EXPIRATION_OPTIONS	有效标签过期值列表。	字符串
DEFAULT_TAG_EXPIRATION	时间机器的标签过期时间。	字符串
CLEAN_BLOB_UPLOAD_FOLDER	从 S3 多部分上传中自动清理过时的 Blob。默认情况下，每小时清理超过 2 天的 Blob 文件。	布尔值 + **默认： true**

20.3. 禁用垃圾回收

镜像标签、命名空间和存储库的垃圾回收功能存储在 config.yaml 文件中。这些功能默认为 true。

在个别情况下，您可能想要禁用垃圾回收，例如控制执行垃圾回收的时间。您可以通过将 GARBAGE_COLLECTION 功能设置为 false 来禁用垃圾回收。禁用后，不会删除悬停或未标记的镜像、存储库、命名空间、层和清单。这可能会增加环境的停机时间。



注意

没有命令来手动运行垃圾回收。相反，您将禁用，然后重新启用垃圾回收功能。

20.4. 垃圾回收和配额管理

Red Hat Quay 在 3.7 中引入了配额管理。通过配额管理，用户可以通过建立配置的存储配额限制来报告存储消耗并包含 registry 增长。

从 Red Hat Quay 3.7 开始，垃圾回收会回收在删除后分配给镜像、存储库和 Blob 的内存。由于垃圾回收功能在删除后回收内存，因此环境磁盘空间中存储的内容与配额管理作为总消耗报告之间存在差异。当前没有解决此问题的方法。

20.5. 实践中的垃圾回收

使用以下步骤检查 Red Hat Quay 日志，以确保垃圾回收正常工作。

流程

1. **输入以下命令来确保垃圾回收正常工作：**

```
$ sudo podman logs <container_id>
```

输出示例：

```
gcworker stdout | 2022-11-14 18:46:52,458 [63] [INFO] [apscheduler.executors.default] Job "GarbageCollectionWorker._garbage_collection_repos (trigger: interval[0:00:30], next run at: 2022-11-14 18:47:22 UTC)" executed successfully
```

2. **删除镜像标签。**

3. **输入以下命令来确保标签已被删除：**

```
$ podman logs quay-app
```

输出示例：

```
gunicorn-web stdout | 2022-11-14 19:23:44,574 [233] [INFO] [gunicorn.access] 192.168.0.38 - - [14/Nov/2022:19:23:44 +0000] "DELETE /api/v1/repository/quayadmin/busybox/tag/test HTTP/1.0" 204 0 "http://quay-server.example.com/repository/quayadmin/busybox?tab=tags" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

20.6. RED HAT QUAY 垃圾回收指标

以下指标显示垃圾回收删除了多少个资源。这些指标显示垃圾回收 worker 运行的次数，以及删除命名空间、存储库和 Blob 的数量。

指标名称	描述
quay_gc_iterations_total	GCWorker 的迭代数

指标名称	描述
quay_gc_namespaces_purged_total	NamespaceGCWorker 清除的命名空间数量
quay_gc_repos_purged_total	RepositoryGCWorker 或 NamespaceGCWorker 清除的软件仓库数
quay_gc_storage_blobs_deleted_total	已删除的存储 Blob 数量

指标输出示例

```
# TYPE quay_gc_iterations_created gauge
quay_gc_iterations_created{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 1.6317823190189714e+09
...

# HELP quay_gc_iterations_total number of iterations by the GCWorker
# TYPE quay_gc_iterations_total counter
quay_gc_iterations_total{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
...

# TYPE quay_gc_namespaces_purged_created gauge
quay_gc_namespaces_purged_created{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 1.6317823190189433e+09
...

# HELP quay_gc_namespaces_purged_total number of namespaces purged by the NamespaceGCWorker
# TYPE quay_gc_namespaces_purged_total counter
quay_gc_namespaces_purged_total{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
....

# TYPE quay_gc_repos_purged_created gauge
quay_gc_repos_purged_created{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 1.631782319018925e+09
...

# HELP quay_gc_repos_purged_total number of repositories purged by the RepositoryGCWorker or NamespaceGCWorker
# TYPE quay_gc_repos_purged_total counter
quay_gc_repos_purged_total{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
...

# TYPE quay_gc_storage_blobs_deleted_created gauge
quay_gc_storage_blobs_deleted_created{host="example-registry-quay-app-6df87f7b66-9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 1.6317823190189059e+09
```

```
...

# HELP quay_gc_storage_blobs_deleted_total number of storage blobs deleted
# TYPE quay_gc_storage_blobs_deleted_total counter
quay_gc_storage_blobs_deleted_total{host="example-registry-quay-app-6df87f7b66-
9tfn6",instance="",job="quay",pid="208",process_name="secscan:application"} 0
...
```

第 21 章 使用 V2 UI

21.1. V2 用户界面配置

重要

- 这个 UI 目前处于 beta 阶段，可能会有变化。在其当前状态中，用户只能创建、查看和删除机构、存储库和镜像标签。
- 使用旧 UI 时，超时会话将要求用户在弹出窗口中再次输入密码。使用新的 UI 时，用户将返回到主页面，并需要输入其用户名和密码凭证。这是一个已知问题，并将在以后的新 UI 版本中解决。
- 在传统 UI 和新 UI 之间如何报告镜像清单大小时，有一个差异。在传统 UI 中，镜像清单以兆字节为单位报告。v2 UI 使用标准定义兆字节(MB)来报告镜像清单大小。

流程

1. 登录您的部署。
2. 在部署的导航窗格中，您可以选择在 Current UI 和 New UI 之间切换。点击切换按钮将其设置为 New UI，然后点 Use Beta Environment，例如：

21.1.1. 使用 v2 UI 创建新机构

先决条件

- 您已将部署切换为使用 v2 UI。

使用以下步骤使用 v2 UI 创建机构。

流程

1. 单击导航窗格中的 Organization。
2. 单击 Create Organization。
3. 输入 组织名称，例如 testorg。
4. 点 Create。

现在，您的示例组织应在 Organizations 页面下填充。

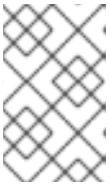
21.1.2. 使用 v2 UI 删除机构

使用以下步骤使用 v2 UI 删除机构。

流程

1. 在 Organizations 页面上，选择您要删除的机构的名称，如 testorg。

2. 点 More Actions 下拉菜单。
3. 点 Delete。



注意

在 Delete 页面中，有一个 搜索 输入框。有了此框，用户可以搜索特定的组织，以确保它们被正确地安排删除。例如，如果用户正在删除 10 个机构，并希望确保删除了一个特定的机构，他们可以使用 搜索 输入框确认机构标记为删除。

4. 通过在框中输入 confirm 确认您要永久删除组织。
5. 点 Delete。
删除后，您将返回到 Organizations 页面。

注意

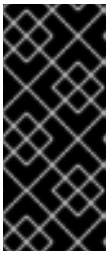
您可以通过选择多个机构，然后点 More Actions → Delete 一次删除多个机构。

21.1.3. 使用 v2 UI 创建新存储库

使用以下步骤使用 v2 UI 创建存储库。

流程

1. 单击导航窗格上的 Repositories。
2. 单击 Create Repository。
3. 选择一个命名空间，如 quayadmin，然后输入 Repository name，如 testrepo。



重要

不要在您的仓库名称中使用以下词语：* build * trigger * tag

当这些词语用于存储库名称时，用户无法访问存储库，且无法永久删除存储库。尝试删除这些软件仓库会返回以下错误： Failed to delete repository <repository_name>, HTTP404 - Not Found。

4. 点 Create。
现在，您的示例存储库应在 Repositories 页面下填充。

21.1.4. 使用 v2 UI 删除存储库

先决条件

- 您已创建了一个存储库。

流程

1. 在 v2 UI 的 Repositories 页面中，单击您要删除的镜像的名称，如 quay/admin/busybox。
2. 点 More Actions 下拉菜单。

3. 点 Delete。

注意

如果需要，您可以点击 Make Public 或 Make Private。

4. 在框中键入 confirm，然后单击 Delete。
5. 删除后，您将返回到 Repositories 页面。

21.1.5. 将镜像推送到 v2 UI

使用以下步骤将镜像推送到 v2 UI。

流程

1. 从外部 registry 拉取示例镜像：

```
$ podman pull busybox
```

2. 标记镜像：

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

3. 将镜像推送到 registry：

```
$ podman push quay-server.example.com/quayadmin/busybox:test
```

4. 导航到 v2 UI 上的 Repositories 页面，并确保您的镜像已被正确推送。
5. 您可以选择镜像标签来检查安全详情，然后导航到 Security Report 页面。

21.1.6. 使用 v2 UI 删除镜像

使用以下步骤通过 v2 UI 删除镜像。

先决条件

- 您已将镜像推送到 registry。

流程

1. 在 v2 UI 的 Repositories 页面中，单击您要删除的镜像的名称，如 quay/admin/busybox。
2. 点 More Actions 下拉菜单。
3. 点 Delete。

注意

如果需要，您可以点击 Make Public 或 Make Private。

4. 在框中键入 confirm，然后单击 Delete。
5. 删除后，您将返回到 Repositories 页面。

21.1.7. 使用 Red Hat Quay v2 UI 创建新团队

使用以下步骤使用 Red Hat Quay v2 UI 创建新团队。

先决条件

- 您已创建了一个带有存储库的组织。

流程

1. 在 Red Hat Quay v2 UI 上，点机构的名称。
2. 在您的机构页面中，点 Teams 和 membership。
3. 点 Create new team 复选框。
4. 在 Create team 弹出窗口中，为您的新团队提供一个名称。
5. 可选。为您的新团队提供描述。
6. 单击 Proceed。此时会出现一个新的弹出窗口。
7. 可选。将此团队添加到存储库，并将权限设置为 Read、Write、Admin 或 None 之一。
8. 可选。添加团队成员或机器人帐户。要添加团队成员，请输入其 Red Hat Quay 帐户的名称。
9. 检查并填写信息，然后点 Review and Finish。新团队会出现在 Teams 和 membership 页面下。在这里，您可以点 kebab 菜单并选择以下选项之一：
    - 管理团队成员.在此页面上，您可以查看所有成员、团队成员、机器人帐户或被邀请的用户。您还可以通过单击 Add new member 来添加新团队成员。
    - 设置存储库权限。在本页中，您可以将存储库权限设置为 Read、Write、Admin 或 None 之一。
    - 删除。此弹出窗口允许您通过单击 Delete 来删除团队。
10. 可选。您可以点击以下选项之一来显示有关团队、成员和协作者的更多信息：
    - 团队视图.此菜单显示所有团队名称、成员数量、存储库数量以及每个团队的角色。
    - 成员视图.此菜单显示团队成员的所有用户名，以及他们所属的团队，以及用户的存储库权限。
    - collaborators View.此菜单显示存储库协作器。collaborator 是不属于机构中的任何团队的用户，但他们对属于该组织的一个或多个存储库具有直接权限。

21.1.8. 使用 v2 UI 创建机器人帐户

使用以下步骤，使用 v2 UI 创建机器人帐户。

流程

1. 在 v2 UI 上，单击 Organizations。
2. 单击您要为其创建机器人帐户的组织名称，如 test-org。
3. 点 Robot accounts 选项卡 →Create robot account。
4. 在 Provide a name for your robot account 框中，输入名称，如 robot1。
5. 可选。如果需要，可以使用以下选项：
   a. 将机器人添加到团队。
   b. 将机器人添加到存储库。
   c. 调整机器人的权限。
6. 在 Review and finish 页面中，查看您提供的信息，然后点 Review and finish。此时会出现以下警报： Successfully created robot account with robot name: <organization_name> + <robot_name>.
   或者，如果您试图创建与另一个机器人帐户相同的机器人帐户，您可能会收到以下错误消息： Error create robot account。
7. 可选。您可以单击 Expand 或 Collapse 以显示有关机器人帐户的描述性信息。
8. 可选。您可以点 kebab 菜单 → Set repository 权限来更改机器人帐户的权限。显示以下消息： Successfully updated repository 权限。
9. 可选。要删除您的机器人帐户，请选中机器人帐户的复选框，然后单击回收站图标。此时会出现弹出窗口。在文本框中键入 confirm，然后单击 Delete。或者，您可以点 kebab 菜单 → Delete。显示以下消息： Successfully deleted robot account.

21.1.8.1. 使用 Red Hat Quay v2 UI 批量管理机器人帐户存储库访问权限

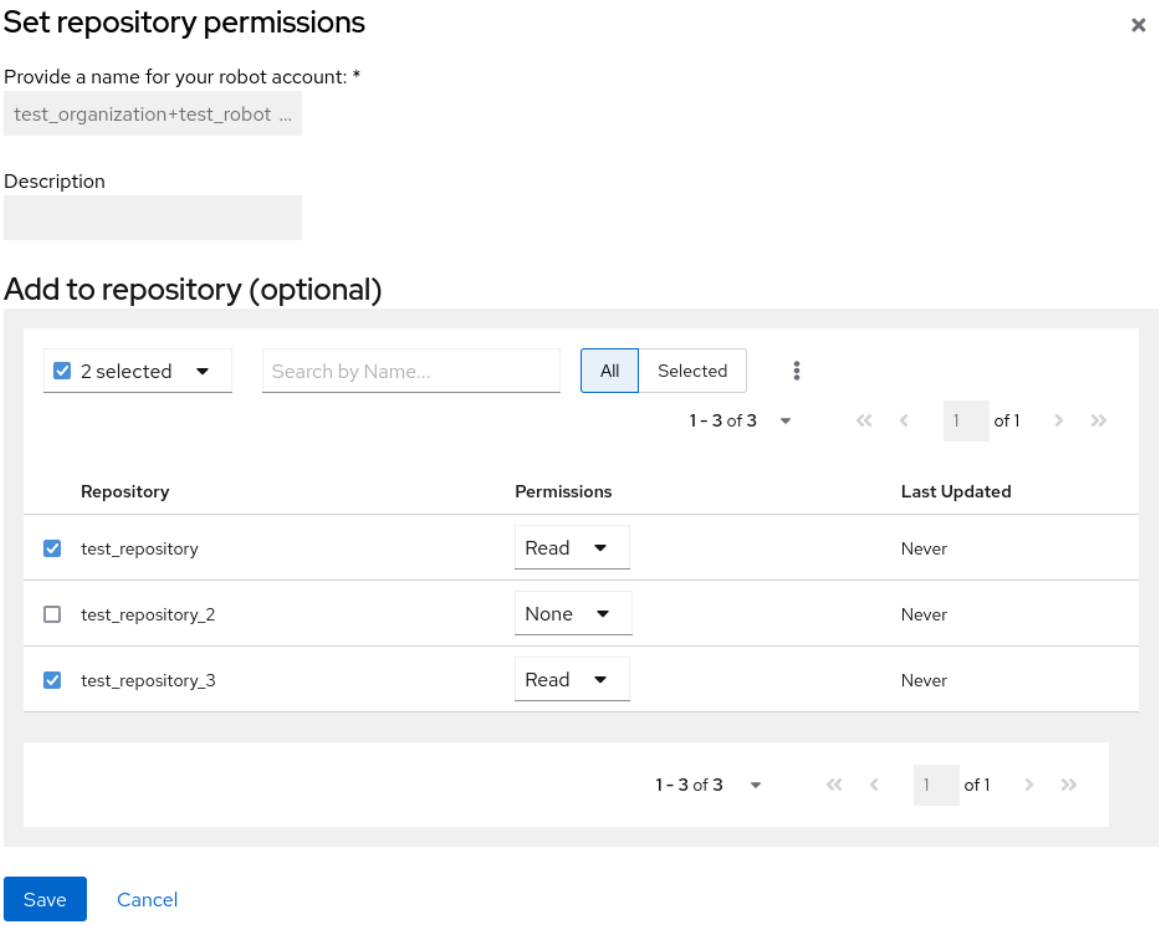
使用以下步骤，使用 Red Hat Quay v2 UI 在批量、机器人帐户存储库访问中。

先决条件

- 您已创建了机器人帐户。
- 您已在单个机构中创建多个软件仓库。

流程

1. 在 Red Hat Quay v2 UI 登录页面中，单击导航窗格中的 Organizations。
2. 在 Organizations 页面上，选择具有多个存储库的组织的名称。单个机构下的存储库数量可在 Repo Count 列下找到。
3. 在您的组织页面中，单击 Robot 帐户。
4. 对于将添加到多个存储库的机器人帐户，点 kebab 图标 → Set repository 权限。
5. 在 Set repository permissions 页面上，选中机器人帐户要添加到的存储库的框。例如：



6. 设置机器人帐户的权限，如 None、Read、Write、Admin。
7. 单击保存。显示 Success alert: Successfully updated repository 权限的警报会出现在 Set repository permissions 页面中，确认更改。
8. 返回到 Organizations → Robot accounts 页面。现在，您的机器人帐户的 Repositories 列显示机器人帐户已添加到的存储库数量。

21.1.9. 使用 Red Hat Quay v2 UI 创建默认权限

默认权限定义在创建存储库时应自动授予的权限，除了存储库的默认创建者之外。根据创建存储库的用户分配权限。

使用以下步骤使用 Red Hat Quay v2 UI 创建默认权限。

流程

1. 点机构的名称。
2. 单击 Default permissions。
3. 单击 创建默认权限。此时会出现切换 drawer。
4. 选择 Anyone 或 Specific 用户，在创建存储库时创建默认权限。
   a. 如果选择 Anyone，则必须提供以下信息：
      - 应用到。搜索、邀请或添加用户/机器/团队。

- 权限.将权限设置为 Read,Write, 或 Admin 之一。

b. 如果选择特定用户，则必须提供以下信息：

- 存储库创建者.提供用户或机器人帐户。
- 应用到。提供用户名、机器人帐户或团队名称。
- 权限.将权限设置为 Read,Write, 或 Admin 之一。

5. 点 Create default permissions。此时会出现确认框，返回以下警报： Successfully created default permissions for creator。

21.1.10. v2 UI 的组织设置

使用以下步骤使用 v2 UI 更改您的机构设置。

流程

1. 在 v2 UI 上，单击 Organizations。
2. 单击您要为其创建机器人帐户的组织名称，如 test-org。
3. 点 Settings 选项卡。
4. 可选。输入与机构关联的电子邮件地址。
5. 可选。将 Time Machine 功能的分配时间设置为以下之一：
   - 1 周
   - 1 个月
   - 1 年
   - Never
6. 点击 Save。

21.1.11. 使用 v2 UI 查看镜像标签信息

使用以下步骤通过 v2 UI 查看镜像标签信息。

流程

1. 在 v2 UI 上，单击 Repositories。
2. 点存储库的名称，例如 quayadmin/busybox。
3. 单击标签的名称，例如 test。您会进入标签的 Details 页面。该页面显示以下信息：
   - Name
   - 软件仓库
   - 摘要

- 安全漏洞
- 创建
- modified
- Size
- 标签
- 如何获取镜像标签

4. 可选。点 Security Report 查看标签的漏洞。您可以扩展公告列以打开 CVE 数据。
5. 可选。点 Packages 查看标签的软件包。
6. 单击存储库的名称，如 busybox，以返回到 Tags 页面。
7. 可选。将鼠标悬停在 Pull 图标上，以显示获取标签的方法。
8. 选中标签框或多个标签，单击 Actions 下拉菜单，然后单击 Delete 以删除该标签。在弹出框中单击 Delete 来确认删除。

21.1.12. 使用 v2 UI 调整存储库设置

使用以下步骤使用 v2 UI 调整存储库的各种设置。

流程

1. 在 v2 UI 上，单击 Repositories。
2. 点存储库的名称，例如 quayadmin/busybox。
3. 点 Settings 选项卡。
4. 可选。单击 User and robot permissions。您可以通过单击 权限 下的下拉菜单选项来调整用户或机器人帐户的设置。您可以将设置更改为 Read、Write 或 Admin。
5. 可选。单击 Events 和 notification。您可以通过单击 Create Notification 来创建事件和通知。可用的事件选项如下：
   - 推送到存储库
   - 发现软件包漏洞
   - 镜像构建失败
   - 镜像构建已排队
   - 镜像构建已启动
   - 镜像构建成功
   - 镜像构建已取消
     然后，发出通知。可用的选项如下：
   - 电子邮件通知

- Flowdock 团队通知
- HipChat Room 通知
- Slack 通知
- Webhook POST
  选择事件选项和通知方法后，包括一个 Room ID #, a Room Notification Token，然后点 Submit。

6. 可选。单击 Repository visibility。您可以通过单击 Make Public 使存储库为私有或公共存储库。
7. 可选。单击 Delete repository。您可以通过单击 Delete Repository 来删除存储库。

21.2. 查看 RED HAT QUAY 标签历史记录

使用以下步骤查看 Red Hat Quay v2 UI 上的标签历史记录。

流程

1. 在 Red Hat Quay v2 UI 仪表板中，单击导航窗格中的 Repositories。
2. 单击具有镜像标签的存储库的名称。
3. 单击 Tag History。在这个页面中，您可以执行以下操作：
   - 根据标签名称搜索
   - 选择日期范围
   - 查看标签更改
   - 查看标签修改日期及其更改的时间

21.3. 在 RED HAT QUAY V2 UI 中添加和管理标签

Red Hat Quay 管理员可以按照以下流程为标签添加和管理标签标签。

流程

1. 在 Red Hat Quay v2 UI 仪表板中，单击导航窗格中的 Repositories。
2. 单击具有镜像标签的存储库的名称。
3. 单击镜像的 kebab 菜单，再选择 Edit labels。
4. 在 Edit labels 窗口中，点 Add new label。
5. 使用 key=value 格式输入镜像标签的标签，如 com.example.release-date=2023-11-14。

注意

当无法使用 key=value 格式时返回以下错误：Invalid label format，必须为由 = 分隔的键值。

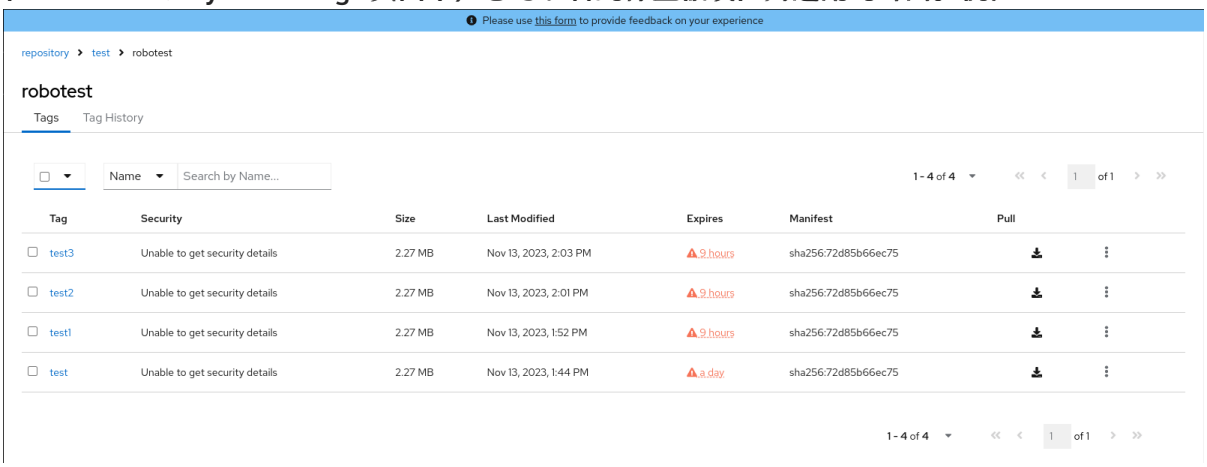
6. 单击框的空格来添加标签。
7. 可选。添加第二个标签。
8. 点 Save labels 将标签保存到镜像标签。返回以下通知： Created labels successfully。
9. 可选。点标签上的同一镜像标签菜单 kebab → Edit labels → X 将其删除；或者，您可以编辑文本。点 Save labels。标签现已被删除或编辑。

21.4. 在 RED HAT QUAY V2 UI 中设置标签过期

Red Hat Quay 管理员可以为存储库中的特定标签设置过期日期。这有助于自动清理旧的或未使用的标签，有助于减少存储空间。

流程

1. 在 Red Hat Quay v2 UI 仪表板中，单击导航窗格中的 Repositories。
2. 单击具有镜像标签的存储库的名称。
3. 单击镜像的 kebab 菜单，然后选择 Change expiration。
4. 可选。或者，您可以通过点击多个标签框来批量添加过期日期，然后选择 Actions → Set expiration。
5. 在 Change Tags Expiration 窗口中，设置一个过期日期，指定星期几、月份、月份和年的日期。例如，2023 年 11 月 15 日星期三。或者，您也可以单击日历按钮并手动选择日期。
6. 设置时间，例如 2:30 PM。
7. 点 Change Expiration 确认日期和时间。返回以下通知：成功将 标签测试的过期时间设置为 2023 年 11 月 15 日，2:26 PM。
8. 在 Red Hat Quay v2 UI Tags 页面中，您可以看到标签被设置为过期的时间。例如：



21.5. 在 RED HAT QUAY V2 UI 上选择颜色首选项

在使用 v2 UI 时，用户可以在 light 和 dark 模式间切换。此功能还包括自动模式选择，它根据用户的浏览器首选项在 light 或 dark 模式之间进行选择。

使用以下步骤在自动、light 和 dark 模式间切换。

流程

1. 登录到您的 Red Hat Quay 存储库。
2. 在导航窗格中，点您的用户名，例如 quayadmin。
3. 在 Appearance 下，选择 Light theme、Dark 主题和 Device-based theme。基于主题的设备会根据浏览器的颜色首选项设置模式。

21.6. 查看 RED HAT QUAY V2 UI 中的使用日志

Red Hat Quay 日志可以提供有关 Red Hat Quay registry 使用的方式的有价值的信息。可以通过 v2 UI 上的机构、存储库或命名空间来查看日志。

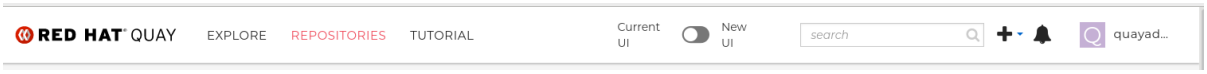
流程

1. 登录到您的 Red Hat Quay registry。
2. 导航到您作为管理员的 Organization、repository 或 namespace。
3. 点 Logs。

4. 可选。通过将日期添加到 From 和 To 框来设置用于查看日志条目的日期范围。
5. 可选。点 Export 来导出日志。您必须输入电子邮件地址或以 http:// 或 https:// 开头的有效回调 URL。根据存在多少个日志，此过程可能需要一小时。

21.7. 启用旧的 UI

1. 在导航窗格中，您可以选择在 Current UI 和 New UI 之间切换。点击切换按钮，来将它设为 Current UI。



第 22 章 在 RED HAT QUAY 部署上执行健康检查

健康检查机制旨在评估系统、服务或组件的健康和功能。健康检查有助于确保一切正常工作，并可用于在潜在问题成为严重问题之前识别潜在问题。通过监控系统的健康状况，Red Hat Quay 管理员可以针对地域复制部署、Operator 部署、独立 Red Hat Quay 部署、对象存储问题等方面解决异常或潜在的故障。执行健康检查有助于降低遇到故障排除场景的可能性。

通过提供有关系统当前状态的宝贵信息，健康检查机制可以在诊断问题方面扮演角色。通过将健康检查结果与预期的基准测试或预定义的阈值进行比较，可以更快地识别 deviations 或 anomalies。

22.1. RED HAT QUAY 健康检查端点

重要

此处包含的任何外部网站的链接仅为方便用户而提供。红帽没有审阅链接的内容，并不对其内容负责。包含到外部网站的任何链接并不意味着红帽认可该网站或其实体、产品或服务。您同意红帽对因您使用（或依赖）外部网站或内容而导致的任何损失或费用不承担任何责任。

Red Hat Quay 有几个健康检查端点。下表显示了健康检查、描述、端点和示例输出。

表 22.1. 健康检查端点

健康检查	描述	端点	输出示例
实例	实例 端点获取特定 Red Hat Quay 实例的完整状态。返回带有以下键值对 的字典 ： auth,database,disk_space,registry_gunicorn,service_key, 和 web_gunicorn。 返回指示 200 的健康检查响应的数字，这表示实例处于健康状态，或者 503，这表示您的部署有问题。	https://{quay-ip-endpoint}/health/instance 或 https://{quay-ip-endpoint}/health	{"data":{"services":{"auth":true,"database":true,"disk_space":true,"registry_gunicorn":true,"service_key":true,"web_gunicorn":true}},"status_code":200}
endtoend	端到端端点对 Red Hat Quay 实例的所有服务进行检查。使用以下的键值对返回 字典 ： auth、数据库、redis、存储。返回指示 200 的健康检查响应的数字，这表示实例处于健康状态，或者 503，这表示您的部署有问题。	https://{quay-ip-endpoint}/health/endtoend	{"data":{"services":{"auth":true,"database":true,"redis":true,"storage":true}},"status_code":200}
warning	警告 端点对警告进行检查。为以下内容返回一个带有键值对的字典：disk_space_warning。返回指示 200 的健康检查响应的数字，这表示实例处于健康状态，或者 503，这表示您的部署有问题。	https://{quay-ip-endpoint}/health/warning	{"data":{"services":{"disk_space_warning":true}},"status_code":503}

22.2. 导航到 RED HAT QUAY 健康检查端点

使用以下步骤导航到 实例 端点。对于 端到端 和警告 端点，这个过程可以重复。

流程

1. 在 Web 浏览器中，导航到 https://{quay-ip-endpoint}/health/instance。
2. 您使用健康实例页面，它会返回类似如下的信息：

```
{"data":{"services":
{"auth":true,"database":true,"disk_space":true,"registry_gunicorn":true,"service_key"
:true,"web_gunicorn":true}},"status_code":200}
```

对于 Red Hat Quay，"status_code": 200 表示实例是健康的。相反，如果您收到 "status_code": 503，则部署有问题。

第 23 章 在旧 UI 上品牌 RED HAT QUAY 部署

您可以通过更改 registry 标题、徽标、页脚镜像以及将用户定向到嵌入在页脚镜像中的网站来品牌 Red Hat Quay 部署的 UI。

流程

1. 更新您的 Red Hat Quay **config.yaml** 文件，以添加以下参数：

```
BRANDING:
  logo: 1
  footer_img: 2
  footer_url: 3
---
REGISTRY_TITLE: 4
REGISTRY_TITLE_SHORT: 5
```

1. 显示在 Red Hat Quay 部署顶部的镜像的 URL。
2. 在 Red Hat Quay 部署的底部出现的镜像的 URL。
3. 点 footer 镜像时，用户将定向到的网站的 URL。
4. registry 的长期标题。这在 Red Hat Quay 部署的前端中显示，例如，在您的机构签名页面中。
5. registry 的简短标题。该标题显示在您的机构的各种页面中，例如，作为您机构的 Tutorial 页面上的 教程 的标题。

2. 重启 Red Hat Quay 部署。重启后，您的 Red Hat Quay 部署会使用新的徽标、页脚镜像和页脚镜像 URL 更新。

第 24 章 SCHEMA FOR RED HAT QUAY 配置

大多数 Red Hat Quay 配置信息都存储在 config.yaml 文件中。Red Hat Quay 配置指南中描述了所有配置选项。