Red Hat Quay 3

Red Hat Quay のデプロイ - 高可用性 (HA)

Red Hat Quay HA のデプロイ

Last Updated: 2024-02-03

Red Hat Quay 3 Red Hat Quay のデプロイ - 高可用性 (HA)

Red Hat Quay HA のデプロイ

法律上の通知

概要

Red Hat Quay の HA 環境でのデプロイ

目次

はじめに

Red Hat Quay は、エンタープライズレベルの品質の高いコンテナーレジストリー製品です。Quay を使用してコンテナーを構築、保存し、企業内のサーバーにデプロイします。

この手順では、可用性が高く、エンタープライズレベルの品質の Red Hat Quay 設定をデプロイする方法を説明します。

第1章 RED HAT QUAY の機能

Red Hat Quay は、新機能とソフトウェア更新を伴って定期的にリリースされます。Red Hat Quay のデプロイメントでは次の機能を使用できますが、リストはすべてを網羅しているわけではありません。

- 高可用性
- Geo レプリケーション
- リポジトリーのミラーリング
- Docker v2、スキーマ 2 (マルチアーキテクチャー) のサポート
- 継続的インテグレーション
- Clair によるセキュリティースキャン
- カスタムログローテーション
- ダウンタイムなしのガベージコレクション
- 24 時間 365 日のサポート

最新の機能情報については 、Red Hat Quay リリースノート を確認してください。

第2章 RED HAT QUAY のサポート

Red Hat Quay は、以下のサポートを提供します。

- 複数の認証およびアクセス方法
- 複数のストレージバックエンド
- **Quay**、**Clair**、およびストレージバックエンドコンテナーのカスタム証明書
- アプリケーションレジストリー
- 異なるコンテナーイメージタイプ

2.1. アーキテクチャー

Red Hat Quay には、内部と外部の両方のコアコンポーネントがいくつか含まれています。

アーキテクチャーの詳細については、Red Hat Quay アーキテクチャー ガイドを参照してください。

2.1.1. 内部コンポーネント

Red Hat Quay には、以下の内部コンポーネントが含まれています。

- **Quay (コンテナーレジストリー)**: Pod 内の複数のコンポーネントで設定されるサービスとして **Quay** コンテナーを実行します。
- **Clair**: コンテナーイメージで脆弱性の有無をスキャンし、修正を提案します。

2.1.2. 外部コンポーネント

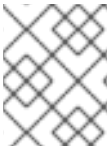
Red Hat Quay には、以下の外部コンポーネントが含まれています。

- **Database**: Red Hat Quay で、プライマリーメタデータストレージとして使用されます。これはイメージストレージ用ではないことに注意してください。
- **Redis (キー/値のストア)**: ライブビルダーログと Red Hat Quay チュートリアルを保存します。ガベージコレクションに必要なロックメカニズムも含まれます。
- **クラウドストレージ**: サポートされているデプロイメントでは、次のストレージタイプのいずれかを使用する必要があります。
  - **パブリッククラウドストレージ**: パブリッククラウド環境では、Amazon Web Services の Amazon S3 や Google Cloud の Google Cloud Storage などのクラウドプロバイダーのオブジェクトストレージを使用する必要があります。
  - **プライベートクラウドストレージ**: プライベートクラウドでは、Ceph RADOS や OpenStack Swift などの S3 または Swift 準拠のオブジェクトストアが必要です。

警告

実稼働環境の設定にローカルにマウントされたディレクトリーのストレージエンジンを使用しないでください。マウントされた NFS ボリュームはサポートされません。ローカルストレージは、Red Hat Quay のテストのみのインストールを対象としています。

第3章 RED HAT QUAY (高可用性) の準備



注記

この手順では、Red Hat Quay の高可用性と実稼働レベルの品質でデプロイを設定する方法を説明します。

3.1. 前提条件

以下で、Red Hat Quay 高可用性デプロイメントを始める前に知っておく必要のある内容をいくつかご紹介します。

- データベースサービスには、Postgres または MySQL のいずれかを使用できます。今回は、Clair のセキュリティースキャンのサポートに必要な機能が含まれているため、データベースとして Postgres が選択されました。オプションには以下が含まれます。
    - Crunchy Data 社の PostgreSQL Operator。Red Hat では直接サポートしていませんが、Crunchy Data 社が提供する CrunchDB Operator は、Red Hat Quay で使用できます。このアプローチを取る場合は、Crunchy Data 社とサポート契約を結び、使用方法の指導やオペレーターやデータベースに関する問題について、Crunchy Data 社と直接やり取りする必要があります。
    - 組織がすでに高可用性 (HA) データベースを使用している場合には、そのデータベースを Red Hat Quay で使用できます。サードパーティーのデータベースやその他のコンポーネントのサポートの詳細は、Red Hat Quay サポートポリシー を参照してください。
- Ceph Object Gateway (RADOS Gateway とも呼ばれる) は、Red Hat Quay が必要とするオブジェクトストレージを提供可能な製品の一例です。Red Hat Quay の設定で Geo レプリケーションを行う場合は、Ceph Object Gateway またはその他のサポート対象のオブジェクトストレージが必要です。クラウドインストールの場合は、以下のいずれかのクラウドオブジェクトストレージを使用できます。
    - Amazon S3 (Red Hat Quay 用の S3 バケットポリシーの設定は、S3 IAM Bucket Policy を参照してください。
    - Azure Blob Storage
    - Google Cloud Storage
    - Ceph Object Gateway
    - OpenStack Swift
    - CloudFront + S3
    - NooBaa S3 ストレージ
- この例では、haproxy サーバーを使用していますが、お客様の環境に合ったプロキシーサービスを使用できます。
- システムの数: この手順では、以下のタスクが割り当てられた 7 つのシステム (物理または仮想) を使用します。
    - **A: db01: ロードバランサーとデータベース**: haproxy のロードバランサーおよび Postgres のデータベースを動作させます。これらのコンポーネントはそれ自体が高可用ではなく、独自のロードバランサーやプロダクションデータベースの設定方法を紹介するためのもので

あることに注意してください。

- **B: quay01、quay02、quay03: Quay および Randis** Quay および Randis のサービスを実行するために、3 つ (またはそれ以上) のシステムが割り当てられています。
- **C: ceph01, ceph02, ceph03, ceph04, ceph05: Ceph** 3 つ (またはそれ以上) のシステムがストレージ向けに Ceph サービスを提供します。クラウドにデプロイする場合は、前述のクラウドストレージ機能を利用できます。この手順では、追加で Ansible (ceph05) 用のシステムと、Ceph Object Gateway (ceph04) 用のシステムを使用します。

各システムには、以下の属性が必要です。

- **Red Hat Enterprise Linux (RHEL)**8: ダウンロードページ から、最新の Red Hat Enterprise Linux 8 サーバーメディアを取得し、Red Hat Enterprise Linux 9 の製品ドキュメント のインストール手順に従います。
  - **Valid Red Hat Subscription**: 有効な Red Hat Enterprise Linux 8 サーバーのサブスクリプションを設定します。
  - **CPU**: 2 つ以上の仮想 CPU
  - **RAM**: A と B の各システムに 4GB、C の各システムに 8GB
  - **ディスクスペース**: A システム、B システムそれぞれに約 20GB のディスクスペース (OS 用に 10GB、docker ストレージ用に 10GB)。C システムには、少なくとも 30GB のディスクスペース (必要なコンテナーストレージに応じてそれ以上) が必要です。

3.2. PODMAN の使用

本書では、コンテナーを作成し、デプロイするために Podman を使用します。システムで Podman が利用可能ではない場合は、同等の docker コマンドを使用できる必要があります。Podman および関連テクノロジーの詳細は、Red Hat Enterprise Linux 8 コンテナーの構築、実行、および管理 を参照してください。

注記

Podman は、Red Hat Quay の高可用性と実稼働環境品質が必要なデプロイメントに強く推奨されます。Docker は Red Hat Quay 3 でテストされておらず、将来のリリースで非推奨になる予定です。

3.3. HAPROXY ロードバランサーと POSTGRESQL データベースのセットアップ

次の手順を使用して、HAProxy ロードバランサーと PostgreSQL データベースをセットアップします。

前提条件

- Podman または Docker CLI がインストールされている。

手順

1. 最初の 2 つのシステム **q01** と **q02** に、HAProxy ロードバランサーと PostgreSQL データベースをインストールします。これにより、他のシステムで実行されている次のサービスのアクセスポイントおよびロードバランサーとして HAProxy が設定されます。

- Red Hat Quay (B システムではポート 80 および 443)
- Redis (B システムではポート 6379)
- RADOS (C システムではポート 7480)

1. SELinux ですべての HAProxy ポートを開き、ファイアウォールで選択した HAProxy ポートを開きます。

```
# setsebool -P haproxy_connect_any=on
# firewall-cmd --permanent --zone=public --add-port=6379/tcp --add-port=7480/tcp
success
# firewall-cmd --reload
success
```

1. **/etc/haproxy/haproxy.cfg** を設定して、Red Hat Quay、Redis、および Ceph RADOS サービスを提供するシステムとポートを指すようにします。以下は、デフォルトと追加されたフロントエンドおよびバックエンド設定の例です。

```
#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
    mode                    tcp
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 3000

#---------------------------------------------------------------------
# main frontend which proxys to the backends
#---------------------------------------------------------------------

frontend  fe_http *:80
    default_backend             be_http
frontend  fe_https *:443
    default_backend             be_https
frontend fe_redis *:6379
   default_backend              be_redis
frontend  fe_rdgw *:7480
    default_backend             be_rdgw
backend be_http
    balance     roundrobin
    server quay01 quay01:80 check
```

```
    server quay02 quay02:80 check
    server quay03 quay03:80 check
backend be_https
    balance     roundrobin
    server quay01 quay01:443 check
    server quay02 quay02:443 check
    server quay03 quay03:443 check
backend be_rdgw
    balance     roundrobin
    server ceph01 ceph01:7480 check
    server ceph02 ceph02:7480 check
    server ceph03 ceph03:7480 check
backend be_redis
server quay01 quay01:6379 check inter 1s
server quay02 quay02:6379 check inter 1s
server quay03 quay03:6379 check inter 1s
```

新しい **haproxy.cfg** ファイルが配置されたら、次のコマンドを入力して HAProxy サービスを再起動します。

```
# systemctl restart haproxy
```

2. 次のコマンドを入力して、PostgreSQL データベースのフォルダーを作成します。

```
$ mkdir -p /var/lib/pgsql/data
```

3. **/var/lib/pgsql/data** フォルダーに次の権限を設定します。

```
$ chmod 777 /var/lib/pgsql/data
```

4. 次のコマンドを入力して、PostgreSQL データベースを起動します。

```
$ sudo podman run -d --name postgresql_database \
    -v /var/lib/pgsql/data:/var/lib/pgsql/data:Z  \
    -e POSTGRESQL_USER=quayuser -e POSTGRESQL_PASSWORD=quaypass \
    -e POSTGRESQL_DATABASE=quaydb -p 5432:5432 \
    registry.redhat.io/rhel8/postgresql-13:1-109
```

注記

コンテナーのデータは、ホストシステムの **/var/lib/pgsql/data** ディレクトリーに保存されます。

5. 次のコマンドを入力して、使用可能な拡張機能をリスト表示します。

```
$ sudo podman exec -it postgresql_database /bin/bash -c 'echo "SELECT * FROM
pg_available_extensions" | /opt/rh/rh-postgresql96/root/usr/bin/psql'
```

出力例

```
   name    | default_version | installed_version |          comment
-----------+-----------------+-------------------+---------------------------------------
 adminpack | 1.0             |                   | administrative functions for PostgreSQL
```

```
...
```

6. 次のコマンドを入力して、**pg_trgm** 拡張機能を作成します。

```
$ sudo podman exec -it postgresql_database /bin/bash -c 'echo "CREATE EXTENSION IF
NOT EXISTS pg_trgm;" | /opt/rh/rh-postgresql96/root/usr/bin/psql -d quaydb'
```

7. 次のコマンドを入力して、**pg_trgm** が作成されたことを確認します。

```
$ sudo podman exec -it postgresql_database /bin/bash -c 'echo "SELECT * FROM
pg_extension" | /opt/rh/rh-postgresql96/root/usr/bin/psql'
```

出力例

```
 extname | extowner | extnamespace | extrelocatable | extversion | extconfig | extcondition
---------+----------+--------------+----------------+------------+-----------+--------------
 plpgsql |       10 |           11 | f              | 1.0        |           |
 pg_trgm |       10 |         2200 | t              | 1.3        |           |
(2 rows)
```

8. Postgres ユーザー **quayuser** の権限を変更し、**superuser** ロールを付与して、ユーザーにデータベースへの無制限のアクセスを許可します。

```
$ sudo podman exec -it postgresql_database /bin/bash -c 'echo "ALTER USER quayuser
WITH SUPERUSER;" | /opt/rh/rh-postgresql96/root/usr/bin/psql'
```

出力例

```
ALTER ROLE
```

9. システムで firewalld サービスがアクティブになっている場合は、次のコマンドを実行して、ファイアウォール経由で PostgreSQL ポートを使用できるようにします。

```
# firewall-cmd --permanent --zone=trusted --add-port=5432/tcp
```

```
# firewall-cmd --reload
```

10. オプション: **postgres** CLI パッケージがインストールされていない場合は、次のコマンドを入力してインストールします。

```
# yum install postgresql -y
```

11. **psql** コマンドを使用して、PostgreSQL データベースへの接続をテストします。

注記

サービスにリモートでアクセスできることを確認するには、リモートシステムで次のコマンドを実行します。

```
# psql -h localhost quaydb quayuser
```

出力例

```
Password for user test:
psql (9.2.23, server 9.6.5)
WARNING: psql version 9.2, server version 9.6.
         Some psql features might not work.
Type "help" for help.

test=> \q
```

3.4. CEPH の設定

この Red Hat Quay の設定では、以下のように 3 ノードの Ceph クラスターを作成し、他にもいくつかのサポートノードを用意します。

- ceph01、ceph02、ceph03 - Ceph Monitor、Ceph Manager、Ceph OSD の各ノード
- ceph04 - Ceph RGW ノード
- ceph05 - Ceph Ansible 管理ノード

Ceph ノードのインストールの詳細は、Red Hat Enterprise Linux への Red Hat Ceph Storage のインストール を参照してください。

Ceph ストレージクラスターを設定したら、Ceph Object Gateway (RADOS ゲートウェイとも呼ばれる) を作成します。詳細は、Ceph Object Gateway のインストール を参照してください。

3.4.1. 各 Ceph ノードのインストール

ceph01、ceph02、ceph03、ceph04、および ceph05 で、以下の作業を行います。

1. Red Hat Ceph Storage をインストールするための要件 で、Ceph ノード設定の前提条件を確認します。特に以下を確認します。
   - OSD ノードで RAID コントローラー を使用するかどうかを決定します。
   - Ceph のネットワーク設定 に別のクラスターネットワークを使用するかどうかを決定します。
2. OSD ストレージを準備 (ceph01、ceph02、ceph03 のみ) します。3 つの OSD ノード (ceph01、ceph02、ceph03) で OSD ストレージを設定します。サポート対象のストレージタイプの詳細は、表 3.2 の OSD Ansible 設定を参照してください。これは、後で Ansible 設定に入力します。この例では、OSD ノードのそれぞれに、OS とは別のフォーマットされていないブロックデバイス (**/dev/sdb**) が 1 つ設定されています。メタルにインストールする場合は、この目的用にマシンに別のハードドライブを追加するとよいでしょう。
3. RHEL 7 インストールガイド に記載されている通り、Red Hat Enterprise Linux Server エディションをインストールします。
4. Red Hat Ceph Storage Nodes の登録 で説明したように、各 Ceph ノードを登録してサブスクライブします。ここでは、必要な repo をサブスクライブする方法を説明します。

```
# subscription-manager repos --disable=*
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

```
# subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-rpms
# subscription-manager repos --enable=rhel-7-server-rhceph-3-osd-rpms
# subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
```

5. 各ノードに root 権限のある ansible ユーザーを作成します。任意の名前を選んでください。以下に例を示します。

```
# USER_NAME=ansibleadmin
# useradd $USER_NAME -c "Ansible administrator"
# passwd $USER_NAME
New password: *********
Retype new password: *********
# cat << EOF >/etc/sudoers.d/admin
admin ALL = (root) NOPASSWD:ALL
EOF
# chmod 0440 /etc/sudoers.d/$USER_NAME
```

3.4.2. Ceph Ansible ノード (ceph05) の設定

Ceph Ansible ノード (ceph05) にログインし、以下のように設定します。この手順を実行するには、ceph01、ceph02、ceph03 の各ノードが稼働している必要があります。

1. Ansible ユーザーのホームディレクトリーに、ceph-ansible Playbook で作成した一時的な値を保存するディレクトリーを作成します。

```
# USER_NAME=ansibleadmin
# sudo su - $USER_NAME
[ansibleadmin@ceph05 ~]$ mkdir ~/ceph-ansible-keys
```

2. ansible ユーザーにパスワードレスの ssh を有効にします。ceph05 で ssh-keygen を実行してから (パスフレーズは空のまま)、ceph01、ceph02、ceph03 システムの Ansible ユーザーにも、ssh-copy-id を実行して繰り返し、公開鍵をコピーします。

```
# USER_NAME=ansibleadmin
# sudo su - $USER_NAME
[ansibleadmin@ceph05 ~]$ ssh-keygen
[ansibleadmin@ceph05 ~]$ ssh-copy-id $USER_NAME@ceph01
[ansibleadmin@ceph05 ~]$ ssh-copy-id $USER_NAME@ceph02
[ansibleadmin@ceph05 ~]$ ssh-copy-id $USER_NAME@ceph03
[ansibleadmin@ceph05 ~]$ exit
#
```

3. ceph-ansible パッケージをインストールします。

```
# yum install ceph-ansible
```

4. この 2 つのディレクトリーの間にシンボリックを作成します。

```
# ln -s /usr/share/ceph-ansible/group_vars \
    /etc/ansible/group_vars
```

5. 修正する Ceph サンプル yml ファイルのコピーを作成します。

```
# cd /usr/share/ceph-ansible
# cp group_vars/all.yml.sample group_vars/all.yml
# cp group_vars/osds.yml.sample group_vars/osds.yml
# cp site.yml.sample site.yml
```

6. コピーした group_vars/all.yml ファイルを編集します。詳細は、表 3.1 の Ansible の一般的な設定を参照してください。以下に例を示します。

```
ceph_origin: repository
ceph_repository: rhcs
ceph_repository_type: cdn
ceph_rhcs_version: 3
monitor_interface: eth0
public_network: 192.168.122.0/24
```

ご使用のネットワーク機器やアドレス範囲は、異なる場合があります。

7. コピーした **group_vars/osds.yml** ファイルを編集します。詳細は、表 3.2 の OSD Ansible 設定を参照してください。この例では、各 OSD ノードの 2 番目のディスクデバイス (**/dev/sdb**) がデータとジャーナルの両方のストレージに使用されています。

```
osd_scenario: collocated
devices:
  - /dev/sdb
dmcrypt: true
osd_auto_discovery: false
```

8. **etc/ansible/hosts** インベントリーファイルを編集して、Ceph ノードを Ceph モニター、OSD、マネージャーノードとして識別します。この例では、各ノードでストレージデバイスも識別されています。

```
[mons]
ceph01
ceph02
ceph03

[osds]
ceph01 devices="[ '/dev/sdb' ]"
ceph02 devices="[ '/dev/sdb' ]"
ceph03 devices="[ '/dev/sdb' ]"

[mgrs]
ceph01 devices="[ '/dev/sdb' ]"
ceph02 devices="[ '/dev/sdb' ]"
ceph03 devices="[ '/dev/sdb' ]"
```

9. この行を **/etc/ansible/ansible.cfg** ファイルに追加して、Ansible Playbook の各実行結果の出力を Ansible ユーザーのホームディレクトリーに保存します。

```
retry_files_save_path = ~/
```

10. Ansible が、設定したすべての Ceph ノードに、Ansible ユーザーとして接続できることを確認します。

```
# USER_NAME=ansibleadmin
# sudo su - $USER_NAME
[ansibleadmin@ceph05 ~]$ ansible all -m ping
ceph01 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ceph02 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ceph03 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
[ansibleadmin@ceph05 ~]$
```

11. ceph-ansible Playbook を (Ansible のユーザーとして) 実行します。

```
[ansibleadmin@ceph05 ~]$ cd /usr/share/ceph-ansible/
[ansibleadmin@ceph05 ~]$ ansible-playbook site.yml
```

この時点で、Ansible Playbook が Ceph ノードをチェックし、要求したサービス向けに設定します。失敗した場合は、必要な修正を行い、コマンドを再実行してください。

12. 3 つの Ceph ノード (ceph01、ceph02、または ceph03) のいずれかにログインし、Ceph クラスターの正常性を確認します。

```
# ceph health
HEALTH_OK
```

13. 同じノードで、rados を使用してモニタリングが機能していることを確認します。

```
# ceph osd pool create test 8
# echo 'Hello World!' > hello-world.txt
# rados --pool test put hello-world hello-world.txt
# rados --pool test get hello-world fetch.txt
# cat fetch.txt
Hello World!
```

3.4.3. Ceph Object Gateway のインストール

Ansible システム (cef05) で、Ceph Storage クラスターへの Ceph Object Gateway を設定します (最終的には cef04 で実行します)。詳細は、Ceph Object Gateway のインストール を参照してください。

3.5. REDIS の設定

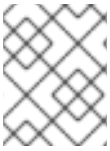
3 台の Red Hat Quay システム (quay01、quay02、quay03) のそれぞれに Red Hat Enterprise Linux 8 サーバーがインストールされている状態で、以下のように Redis サービスをインストールして起動します。

1. **Redis をインストール/デプロイ**: 3 台あるそれぞれの quay0* システムで、Redis をコンテナーとして実行します。

```
# mkdir -p /var/lib/redis
# chmod 777 /var/lib/redis
# sudo podman run -d -p 6379:6379 \
    -v /var/lib/redis:/var/lib/redis/data:Z \
    registry.redhat.io/rhel8/redis-5
```

2. **redis の接続性を確認**: **telnet** コマンドを使用して、redis サービスへの接続性をテストできます。MONITOR と入力して (サービスの監視を開始し)、QUIT で終了します。

```
# yum install telnet -y
# telnet 192.168.122.99 6379
Trying 192.168.122.99...
Connected to 192.168.122.99.
Escape character is '^]'.
MONITOR
+OK
+1525703165.754099 [0 172.17.0.1:43848] "PING"
QUIT
+OK
Connection closed by foreign host.
```



注記

Podman の使用とコンテナーの再起動については、このドキュメントの前半にある Podman の使用を参照してください。

第4章 RED HAT QUAY の設定

Red Hat Quay サービスをコンテナーとして実行する前に、同じ **Quay** コンテナーを使用して Red Hat Quay のデプロイに必要な設定ファイル (**config.yaml**) を作成する必要があります。これを実行するには、**config** 引数とパスワード (ここで my-secret-password を置き換える) を **Quay** コンテナーに渡します。後に、そのパスワードを使用して設定ツールに **quayconfig** としてログインします。

その例を以下で紹介します。

1. **セットアップモードで quay を起動**: 最初の quay ノードで、以下を実行します。

```
# sudo podman run --rm -it --name quay_config -p 8080:8080 registry.redhat.io/quay/quay-rhel8:v3.10.3 config my-secret-password
```

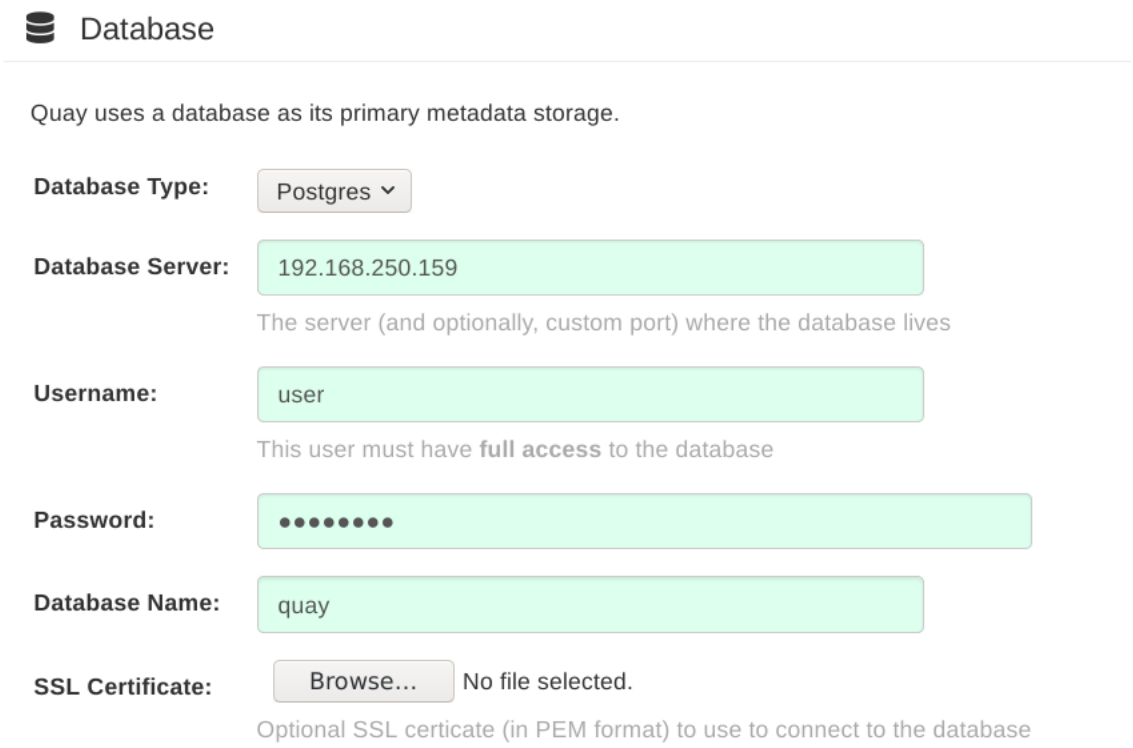
2. **ブラウザーを開く**: quay の設定ツールが起動したら、ブラウザーで、設定ツールを実行中のシステムの URL とポート 8080 を開きます (例: http://myquay.example.com:8080)。ユーザー名とパスワードの入力を求められます。

3. **quayconfig としてログイン**: プロンプトが表示されたら、**quayconfig** のユーザー名とパスワード (**podman run** コマンドラインで入力したもの) を入力します。

4. **必要な項目を入力**: 既存の設定バンドルをマウントせずに設定ツールを起動すると、初期設定セッションが起動します。設定セッションでは、デフォルト値が自動的に入力されます。次のステップでは、残りの必須項目を入力する方法を説明します。

5. **データベースを特定**: 初期設定では、Red Hat Quay で使用するデータベースの種類と場所について、以下の情報を追加する必要があります。

   - **データベースの種類**: MySQL または PostgreSQL を選択してください。基本的な例では MySQL を使用し、高可用性の Red Hat Quay on OpenShift の例では PostgreSQL を使用します。
   - **データベースサーバー**: データベースの IP アドレスまたはホスト名、(3306 と異なる場合はポート番号も) 指定します。
   - **ユーザー名**: データベースへの完全なアクセス権が割り当てられたユーザーを指定します。
   - **パスワード**: 選択したユーザーに割り当てたパスワードを入力します。
   - **データベース名**: データベースサーバーの起動時に割り当てたデータベース名を入力します。
   - **SSL 証明書**: 実稼働環境では、データベースに接続するための SSL 証明書を用意する必要があります。
     次の図は、Red Hat Quay が使用するデータベースを示す画面の例です。



6. **Redis のホスト名、サーバー設定を指定し、他に必要な設定を追加**: 設定完了までに他に追加可能な設定は以下のとおりです。Red Hat Quay の高可用性デプロイメントの他の設定

   - 基本的なテスト設定では、Redis のホスト名を特定するだけで十分です。ただし、この手順の最後で説明するように、Clair スキャンやリポジトリーミラーリングなどの他の機能も追加できます。

   - 高可用性および OpenShift の設定では、共有ストレージやシステム間のセキュアな通信などの機能を実現するのに、さらに多くの設定が必要になります (以下に記載)。
     ここでは、考慮する必要のある設定について説明します。

   - **カスタム SSL 証明書**: Red Hat Quay で使用するカスタムまたは自己署名の SSL 証明書をアップロードします。詳細は、SSL を使用した Red Hat Quay への接続の保護 を参照してください。これは、高可用性の場合に推奨しています。

     重要

     基本デプロイメントと高可用性デプロイメントの両方で SSL 証明書の使用を推奨します。SSL を使用しないことにした場合は、セキュアでないレジストリーのテスト で説明されているように、新しい Red Hat Quay 設定をセキュアでないレジストリーとして使用するように、コンテナークライアントを設定する必要があります。

   - **基本的な設定**: 会社のロゴをアップロードして、Red Hat Quay のレジストリーをリブランディングします。

   - **サーバー設定**: Red Hat Quay サービスに到達するためのホスト名または IP アドレス、TLS の指定 (実稼働インストールでは推奨)。サーバーのホスト名は、Red Hat Quay の全デプロイメントに必要です。TLS 終端は 2 つの異なる方法で実行できます。

     - インスタンス自体で実行する。すべての TLS トラフィックが **Quay** コンテナー内の nginx サーバーによって制御されます (推奨される方法)。

    - ロードバランサーで実行する。これは、推奨されません。ロードバランサーで TLS 設定が正しく行われないと、Red Hat Quay へのアクセスが失われる可能性があります。

- **データ整合性の設定**: パフォーマンスと可用性を向上させるために、ロギングの一貫性保証の設定を緩和するかどうかを選択します。

- **タイムマシン**: 古いイメージタグを一定期間リポジトリーに残すことができ、ユーザーがタグの有効期限を独自に選択できるようにします。

- **redis**: Red Hat Quay が使用する redis サービスに接続するためのホスト名または IP アドレス (およびオプションのパスワード) を指定します。

- **リポジトリーミラーリング**: Enable Repository Mirroring チェックボックスを選択します。この機能を有効にすると、Red Hat Quay クラスターに、リモートレジストリーから選択したリポジトリーをミラーリングするリポジトリーを作成できます。リポジトリーミラーリングを有効にする前に、この手順で後述するように、リポジトリーミラーリングワーカーを起動します。

- **レジストリーストレージ**: ストレージの場所を特定します。クラウドとローカルの各種ストレージオプションがあります。高可用性にはリモートストレージが必要です。Red Hat Quay 高可用ストレージの例を使用している場合は、Ceph ストレージの場所を特定します。OpenShift では、この例では Amazon S3 のストレージを使用しています。

- **アクションログの保存設定**: アクションログは、デフォルトでは Red Hat Quay データベースに保存されます。大量のアクションログがある場合は、それらのログを Elasticsearch に転送して、後で検索や分析を行うことができます。これを行うには、アクションログストレージの設定 で説明されているように、アクションログストレージの値を Elasticsearch に変更し、関連する設定を行います。

- **アクションログのローテーションおよびアーカイブ**: 30 日以上前のログをストレージに移動するログローテーションを有効にして、ストレージ領域を指定します。

- **セキュリティースキャナー**: セキュリティースキャナのエンドポイントと認証キーを選択して、セキュリティースキャンを有効にします。イメージスキャンを行うように Clair を設定するには、Clair のセットアップ および Clair の設定 を参照してください。これは、高可用性の場合に推奨しています。

- **アプリケーションレジストリー**: Kubernetes のマニフェストや Helm のチャートなどを含む、追加のアプリケーションレジストリーを有効にします (App Registry の仕様 を参照)。

- **rkt 変換**: **rkt fetch** が Red Hat Quay レジストリーからイメージを取得して使用できるようにします。GPG2 の公開鍵および秘密鍵が必要です。このフィールドは非推奨です。

- **電子メール**: 通知やユーザーパスワードのリセットに使用する電子メールを有効にします。

- **内部認証**: レジストリーのデフォルト認証をローカルデータベースから LDAP、Keystone (OpenStack)、JWT Custom Authentication、External Application Token に変更します。

- **外部認証 (OAuth)**: これを有効にして GitHub または GitHub Enterprise によるレジストリーへの認証を許可します。

- **Google 認証**: これを有効にして、Google によるレジストリーへの認証を許可します。

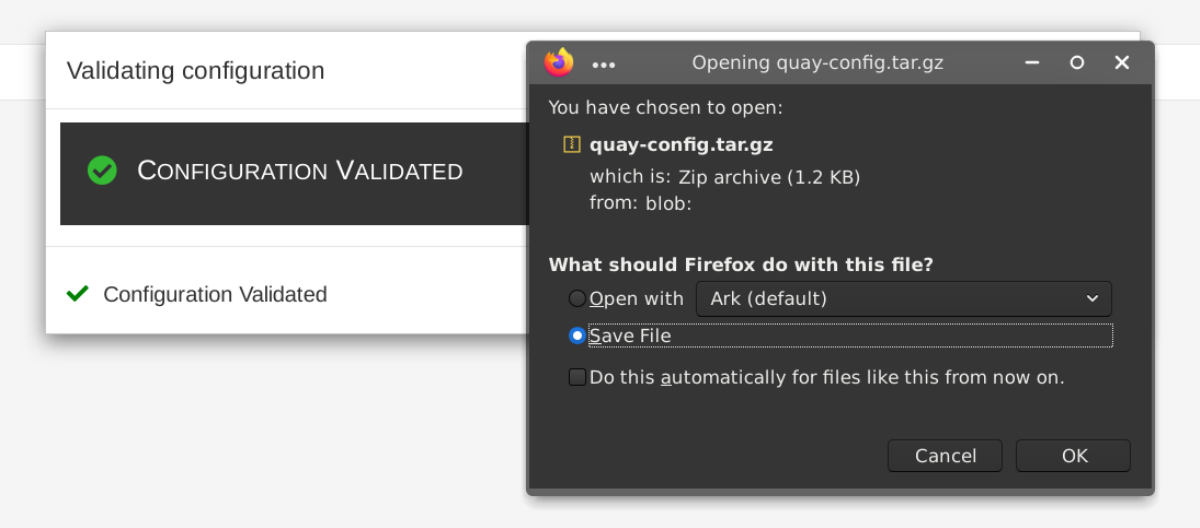
- **アクセス設定**: 基本的なユーザー名/パスワード認証がデフォルトで有効になっています。有効にできる他の認証タイプには、外部アプリケーショントークン (docker や rkt コマンドで使用されるユーザー生成トークン)、匿名アクセス (レジストリーにアクセスできる人なら誰でもアクセスできるようにする)、ユーザー作成 (ユーザーに自分のアカウントを作成

させる)、暗号化クライアントパスワード (暗号化パスワードを含むコマンドラインユーザーアクセスを必要とする)、接頭辞ユーザー名の自動補完 (自動補完時にユーザー名を正確に一致させる必要がないように無効化する) などがあります。

- **レジストリープロトコル設定**: **Restrict V1 Push Support** のチェックボックスをチェックしたままにし、Docker V1 プロトコルのプッシュへのアクセスを制限します。Red Hat は Docker V1 プッシュプロトコルを有効にしないように推奨していますが、有効にする場合は、有効にする namespace を明示的にホワイトリスト化する必要があります。

- **Dockerfile ビルドサポート**: ビルドして Red Hat Quay にプッシュする Dockerfile をユーザーが送信できるようにします。これは、マルチテナント環境では推奨されません。

7. **変更内容の検証**: **Validate Configuration Changes** を選択します。検証が成功すると、次のような Download Configuration モーダルが表示されます。



8. **設定のダウンロード**: **Download Configuration** ボタンを選択し、後で Red Hat Quay の起動に使用するために、tarball (**quay-config.tar.gz**) をローカルディレクトリーに保存します。

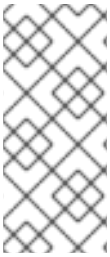
この時点で、Red Hat Quay 設定ツールを終了し、ブラウザーを閉じます。次に、最初の Red Hat Quay ノードをインストールするシステムに、この tarball ファイルをコピーします。基本的なインストールでは、同じシステム上で Red Hat Quay を実行しているだけの場合もあります。

第5章 RED HAT QUAY のデプロイ

Red Hat Quay サービスをクラスター内のノードにデプロイするには、設定ファイルの作成に使用したものと同じ **Quay** コンテナーを使用します。違いは以下のとおりです。

- 設定ファイルやデータが保存されているディレクトリーを指定する。
- **--sysctl net.core.somaxconn=4096** を指定してコマンドを実行する。
- **config** オプションやパスワードは使用しない。

基本的な設定であれば、1つのノードにデプロイできますが、高可用性には、3 つ以上のノード (たとえば quay01、quay02、quay03) が必要になります。



注記

作成された Red Hat Quay サービスは、通常のポート 8080 および SSL ポート 8443 をリッスンします。これは、それぞれ標準のポート 80 と 443 をリッスンしていた、Red Hat Quay の以前のリリースとは異なります。本書では、8080 と 8443 をそれぞれホストの標準ポート 80 と 443 にマッピングしています。本書の残りの部分では、この方法でポートをマッピングしたとの前提で進めていきます。

手順は以下のとおりです。

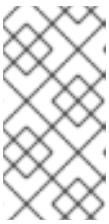
1. **ディレクトリーを作成**: 2 つのディレクトリーを作成して、ホストの設定情報やデータを保存します。以下に例を示します。

```
# mkdir -p /mnt/quay/config
# #optional: if you don't choose to install an Object Store
# mkdir -p /mnt/quay/storage
```

2. **設定ファイルをコピー**: tarball (**quay-config.tar.gz**) を設定ディレクトリーにコピーして展開します。以下に例を示します。

```
# cp quay-config.tar.gz /mnt/quay/config/
# tar xvf quay-config.tar.gz
config.yaml ssl.cert ssl.key
```

3. **Red Hat Quay をデプロイ**: Quay.io への認証を済ませた後 (Red Hat Quay へのアクセス を参照)、以下のように Red Hat Quay をコンテナーとして実行します。



注記

quay コンテナーについて **-e DEBUGLOG=true** を **Quay** の **Podman run** コマンドラインに追加して、デバッグレベルのロギングを有効にします。**-e IGNORE_VALIDATION=true** を追加し、起動プロセスで検証をバイパスします。

```
# sudo podman run --restart=always -p 443:8443 -p 80:8080 \
  --sysctl net.core.somaxconn=4096 \
  --privileged=true \
```

```
-v /mnt/quay/config:/conf/stack:Z \
-v /mnt/quay/storage:/datastorage:Z \
-d registry.redhat.io/quay/quay-rhel8:v3.10.3
```

4. **ブラウザーおよび UI を開く**: **Quay** コンテナーが起動したら、Web ブラウザーに移動し、**Quay** コンテナーを実行するノードの URL を開きます。

5. **Red Hat Quay にログイン**: 設定時に作成したスーパーユーザーアカウントを使用して、ログインして Red Hat Quay が適切に機能していることを確認します。

6. **Red Hat Quay ノードを追加**: この時点で、説明されているように各ノードに移動し、tarball を追加して **Quay** コンテナーを起動することで、この Red Hat Quay クラスターにノードを追加するオプションを選択することができます。

7. **オプション機能を追加**: Clair イメージスキャンやリポジトリーミラーリングなどの Red Hat Quay クラスターに機能を追加するには、次のセクションに進んでください。

5.1. RED HAT QUAY への CLAIR のイメージスキャンの追加

Red Hat Quay のデプロイメントに Clair イメージスキャンを設定してデプロイする方法は、Clair セキュリティースキャン で説明しています。

5.2. RED HAT QUAY へのリポジトリーミラーリングの追加

リポジトリーミラーリングを有効にすると、選択した外部レジストリーのコンテンツと完全に一致するコンテナーイメージのリポジトリーを Red Hat Quay クラスター上に作成し、それらのリポジトリーのコンテンツを定期的なスケジュールで、またはオンデマンドで同期できます。

Red Hat Quay クラスターにリポジトリーミラーリング機能を追加するには、以下の手順に従います。

- リポジトリーミラーリングワーカーを実行します。これを行うには、**repomirror** オプションで quay pod を起動します。
- Red Hat Quay 設定ツールで Enable Repository Mirroring を選択します。
- Red Hat Quay Web UI にログインし、Red Hat Quay でのリポジトリーミラーリング で説明されているように、ミラーリングされたリポジトリーの作成を開始します。

以下の手順は、ブラウザーで Red Hat Quay Setup コンテナーが実行されている OpenShift プラットフォームで Red Hat Quay クラスターがすでに実行されていることを前提としています。

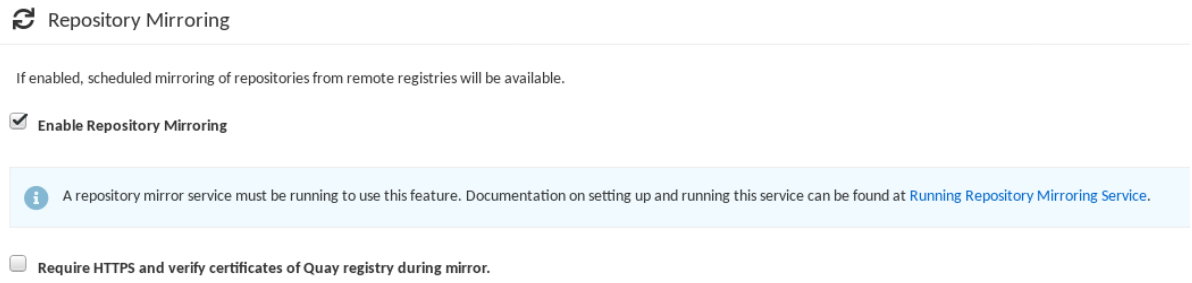
1. **リポジトリーミラーリングワーカーを起動**: **repomirror** モードで **Quay** コンテナーを起動します。この例では、現在 **/root/ca.crt** に保存されている証明書を使用して TLS 通信を設定していることを前提としています。そうでない場合は、コンテナーに **/root/ca.crt** を追加する行を削除します。

```
$ sudo podman run -d --name mirroring-worker \
  -v /mnt/quay/config:/conf/stack:Z \
  -v /root/ca.crt:/etc/pki/ca-trust/source/anchors/ca.crt \
  registry.redhat.io/quay/quay-rhel8:v3.10.3 repomirror
```

2. **設定ツールにログイン**: Red Hat Quay Setup Web UI (設定ツール) にログインします。

3. **リポジトリーのミラーリングを有効化**: リポジトリーミラーリングセクションまでスクロールダウンし、Enable Repository Mirroring チェックボックスを選択します (下図参照)。

4. **HTTPS および証明書の検証を選択**: ミラーリング時に HTTPS 通信を要求し、証明書を検証する場合は、このチェックボックスを選択します。



5. **設定を保存**: Save Configuration Changes ボタンを選択します。これで Red Hat Quay クラスターのリポジトリーミラーリングが有効になります。ミラーリングされた独自のコンテナーイメージのリポジトリーを設定する方法は、Red Hat Quay におけるリポジトリーミラーリングを参照してください。

第6章 RED HAT QUAY の使用開始

Red Hat Quay を起動したら、以下を行うことができます。

- Quay のホームページでチュートリアルを選択して、15 分間のチュートリアルを試します。チュートリアルでは、Quay へのログイン、コンテナーの起動、イメージの作成、リポジトリーのプッシュ、リポジトリーの閲覧、リポジトリーのパーミッションの変更などを確認できます。
- Red Hat Quay リポジトリーでの作業については、Red Hat Quay の使用 を参照してください。

第7章 OPENSHIFT CONTAINER PLATFORM 上の RED HAT QUAY の GEO レプリケーションデプロイメントのアップグレード

以下の手順を実行して、geo-replication Red Hat Quay デプロイメントをアップグレードします。

重要

- geo-replication Red Hat Quay デプロイメントを次の y-stream リリース (例: Red Hat Quay 3.7 → Red Hat Quay 3.8) または geo-replication デプロイメントにアップグレードする場合は、アップグレードを実行する前に操作を停止する必要があります。
- y-stream リリースを次のリリースにアップグレードする場合は、アップグレード中にダウンタイムが断続的に発生します。
- アップグレードする前に、Red Hat Quay デプロイメントをバックアップすることが強く推奨されます。

前提条件

- **registry.redhat.io** にログインしている。

手順

この手順では、3 つ以上のシステムで Red Hat Quay サービスを実行していることを前提としています。詳細は、Red Hat Quay の高可用性の準備 を参照してください。

1. Red Hat Quay インスタンスを実行している各システムですべての Red Hat Quay インスタンスのリストを取得します。
   a. システム A で以下のコマンドを入力して、Red Hat Quay インスタンスを表示します。

```
$ sudo podman ps
```

出力例

```
CONTAINER ID  IMAGE                                COMMAND         CREATED
STATUS            PORTS                                NAMES
ec16ece208c0  registry.redhat.io/quay/quay-rhel8:v3.7.0  registry        6 minutes ago  Up
6 minutes ago  0.0.0.0:80->8080/tcp, 0.0.0.0:443->8443/tcp  quay01
```

   b. System B で以下のコマンドを入力して、Red Hat Quay インスタンスを表示します。

```
$ sudo podman ps
```

出力例

```
CONTAINER ID  IMAGE                                COMMAND         CREATED
STATUS            PORTS                                NAMES
7ae0c9a8b37d  registry.redhat.io/quay/quay-rhel8:v3.7.0  registry        5 minutes ago   Up
2 seconds ago   0.0.0.0:82->8080/tcp, 0.0.0.0:445->8443/tcp  quay02
```

c. System C で以下のコマンドを入力して、Red Hat Quay インスタンスを表示します。

```
$ sudo podman ps
```

出力例

```
CONTAINER ID  IMAGE                                COMMAND         CREATED
STATUS            PORTS                                NAMES
e75c4aebfee9  registry.redhat.io/quay/quay-rhel8:v3.7.0  registry        4 seconds ago   Up
4 seconds ago   0.0.0.0:84->8080/tcp, 0.0.0.0:447->8443/tcp  quay03
```

2. 各システムの Red Hat Quay インスタンスをすべて一時的にシャットダウンします。

a. システム A で以下のコマンドを入力して、Red Hat Quay インスタンスをシャットダウンします。

```
$ sudo podman stop ec16ece208c0
```

b. System B で以下のコマンドを入力して、Red Hat Quay インスタンスをシャットダウンします。

```
$ sudo podman stop 7ae0c9a8b37d
```

c. System C で以下のコマンドを入力して、Red Hat Quay インスタンスをシャットダウンします。

```
$ sudo podman stop e75c4aebfee9
```

3. 各システムで、Red Hat Quay 3 などの最新の Red Hat Quay バージョンを取得します。

a. システム A で以下のコマンドを入力して、最新の Red Hat Quay バージョンを取得します。

```
$ sudo podman pull registry.redhat.io/quay/quay-rhel8:v3.8.0
```

b. システム B で以下のコマンドを入力して、最新の Red Hat Quay バージョンを取得します。

```
$ sudo podman pull registry.redhat.io/quay/quay-rhel8:v3.8.0
```

c. システム C で以下のコマンドを入力して、最新の Red Hat Quay バージョンを取得します。

```
$ sudo podman pull registry.redhat.io/quay/quay-rhel8:v3.8.0
```

4. 高可用性 Red Hat Quay デプロイメントの System A で、Red Hat Quay 3 などの新しいイメージバージョンを実行します。

```
# sudo podman run --restart=always -p 443:8443 -p 80:8080 \
  --sysctl net.core.somaxconn=4096 \
  --name=quay01 \
```

```
    -v /mnt/quay/config:/conf/stack:Z \
    -v /mnt/quay/storage:/datastorage:Z \
    -d registry.redhat.io/quay/quay-rhel8:v3.8.0
```

5. 新しい Red Hat Quay コンテナーがシステム A で完全に動作可能になるまで待ちます。コンテナーのステータスは、次のコマンドを実行すると確認できます。

```
$ sudo podman ps
```

出力例

```
CONTAINER ID  IMAGE                                 COMMAND       CREATED        STATUS
PORTS                                 NAMES
70b9f38c3fb4  registry.redhat.io/quay/quay-rhel8:v3.8.0  registry      2 seconds ago  Up 2
seconds ago   0.0.0.0:82->8080/tcp, 0.0.0.0:445->8443/tcp  quay01
```

6. オプション: Red Hat Quay UI に移動して、Red Hat Quay が完全に動作していることを確認します。

7. システム A 上の Red Hat Quay が完全に動作可能であることを確認したら、System B および System C で新しいイメージバージョンを実行します。

   a. 高可用性 Red Hat Quay デプロイメントの System B で、Red Hat Quay 3 などの新しいイメージバージョンを実行します。

```
# sudo podman run --restart=always -p 443:8443 -p 80:8080 \
  --sysctl net.core.somaxconn=4096 \
  --name=quay02 \
  -v /mnt/quay/config:/conf/stack:Z \
  -v /mnt/quay/storage:/datastorage:Z \
  -d registry.redhat.io/quay/quay-rhel8:v3.8.0
```

   b. 高可用性 Red Hat Quay デプロイメントの System C で、Red Hat Quay 3 などの新しいイメージバージョンを実行します。

```
# sudo podman run --restart=always -p 443:8443 -p 80:8080 \
  --sysctl net.core.somaxconn=4096 \
  --name=quay03 \
  -v /mnt/quay/config:/conf/stack:Z \
  -v /mnt/quay/storage:/datastorage:Z \
  -d registry.redhat.io/quay/quay-rhel8:v3.8.0
```

8. 次のコマンドを入力して、システム B およびシステム C のコンテナーのステータスを確認できます。

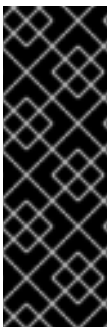
```
$ sudo podman ps
```

第8章 RED HAT QUAY デプロイメントでのヘルスチェックの実行

ヘルスチェックメカニズムは、システム、サービス、またはコンポーネントの正常性と機能を評価するように設計されています。ヘルスチェックは、すべてが正しく機能していることを確認するのに役立ち、重大な問題になる前に潜在的な問題を特定するために使用できます。Red Hat Quay 管理者は、システムの健全性を監視することで、geo レプリケーションのデプロイメント、Operator のデプロイメント、スタンドアロン Red Hat Quay のデプロイメント、オブジェクトストレージの問題といった異常や潜在的な障害に対処できます。ヘルスチェックを行うことで、トラブルシューティングのシナリオが発生する可能性を低減させることができます。

ヘルスチェックメカニズムは、システムの現在の状態に関する貴重な情報を提供することで、問題の診断にロールを果たします。ヘルスチェックの結果を予想されるベンチマークまたは事前定義されたしきい値と比較することで、逸脱や異常をより迅速に特定できます。

8.1. RED HAT QUAY ヘルスチェックエンドポイント



重要

以下に示す外部の Web サイトへのリンクは、お客様の利便性のみを目的として提供しています。Red Hat はリンクの内容を確認しておらず、コンテンツまたは可用性について責任を負わないものとします。外部 Web サイトへのリンクが含まれていても、Red Hat が Web サイトまたはその組織、製品、もしくはサービスを保証することを意味するものではありません。お客様は、外部サイトまたはコンテンツの使用 (または信頼) によって生じる損失または費用について、Red Hat が責任を負わないことに同意するものとします。

Red Hat Quay には、いくつかのヘルスチェックエンドポイントがあります。次の表に、ヘルスチェック、説明、エンドポイント、および出力例を示します。

表8.1 ヘルスチェックエンドポイント

Health check	説明	Endpoint (エンドポイント)	出力例
instance	**instance** エンドポイントは、特定の Red Hat Quay インスタンスのステータス全体を取得します。**auth**、**database**、**disk_space**、**registry_gunicorn**、**service_key**、**web_gunicorn** のキーと値のペアを含む **dict** を返します。インスタンスが正常であることを示す **200**、またはデプロイメントに問題があることを示す **503** のいずれかのヘルスチェック応答を示す数値を返します。	**https://{quay-ip-endpoint}/health/instance** または **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}**

Health check	説明	Endpoint (エンドポイント)	出力例
endtoend	**endtoend** エンドポイントは、Red Hat Quay インスタンスの全サービスのチェックを実行します。**auth**、**database**、**redis**、**storage** のキーと値のペアを含む **dict** を返します。インスタンスが正常であることを示す **200**、またはデプロイメントに問題があることを示す **503** のいずれかのヘルスチェック応答を示す数値を返します。	**https://{quay-ip-endpoint}/health/endtoend**	**{"data":{"services":{"auth":true,"database":true,"redis":true,"storage":true}},"status_code":200}**
warning	**warning** エンドポイントは、警告のチェックを実行します。**disk_space_warning** のキーと値のペアを持つ **dict** を返します。インスタンスが正常であることを示す **200**、またはデプロイメントに問題があることを示す **503** のいずれかのヘルスチェック応答を示す数値を返します。	**https://{quay-ip-endpoint}/health/warning**	**{"data":{"services":{"disk_space_warning":true}},"status_code":503}**

8.2. RED HAT QUAY ヘルスチェックエンドポイントへの移動

次の手順を使用して、**instance** のエンドポイントに移動します。この手順は、**endtoend** および **warning** エンドポイントに対して繰り返すことができます。

手順

1. Web ブラウザーで、**https://{quay-ip-endpoint}/health/instance** に移動します。
2. ヘルスインスタンスページが表示され、以下のような情報が返されます。

```
{"data":{"services":
{"auth":true,"database":true,"disk_space":true,"registry_gunicorn":true,"service_key":true,"web_gunicorn":true}},"status_code":200}
```

Red Hat Quay の場合、**status_code: 200** はインスタンスが正常であることを意味します。逆に、**"status_code": 503** を受け取った場合は、デプロイメントに問題があります。

関連情報