Red Hat Quay 3.8

概念実証 (実稼働以外) 向けの Red Hat Quay のデプロイ

Red Hat Quay のデプロイ

Last Updated: 2024-01-16

Red Hat Quay 3.8 概念実証 (実稼働以外) 向けの Red Hat Quay のデプロイ

Red Hat Quay のデプロイ

法律上の通知

概要

Red Hat Quay を使い始める

目次

はじめに

Red Hat Quay は、コンテナーイメージをビルドして、保護し、これを提供するためのエンタープライズ品質のレジストリーです。この手順では、概念実証 (実稼働以外) 向けの Red Hat Quay のデプロイ方法について説明します。

第1章 概要

Red Hat Quay には、以下の機能が含まれています。

- 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

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

1.1. 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。

1.1.1. 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。

1.1.2. 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 のテスト専用のインストールに使用されることが意図されています。

第2章 RED HAT QUAY の使用開始

Red Hat Quay レジストリーは、実稼働以外の目的で、物理または仮想のいずれかの単一のマシンにデプロイできます。

2.1. 前提条件

- Red Hat Enterprise Linux (RHEL) 8
  - Red Hat Enterprise Linux (RHEL) 8 の最新バージョンを入手するには、Red Hat Enterprise Linux の ダウンロード を参照してください。
  - インストール手順は、Red Hat Enterprise Linux 8 の製品ドキュメント を参照してください。
- Red Hat への有効なサブスクリプション
- 2 つ以上の仮想 CPU
- 4 GB 以上の RAM
- テストシステムに約 30 GB のディスク容量。次のように分類できます。
  - Red Hat Enterprise Linux (RHEL) オペレーティングシステム用に約 10 GB のディスク容量。
  - 3 つのコンテナーを実行するための Docker ストレージ用に約 10 GB のディスク容量。
  - Red Hat Quay ローカルストレージ用に約 10 GB のディスク容量。

注記

CEPH またはその他のローカルストレージでは、より多くのメモリーが必要になる場合があります。

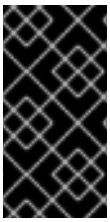
サイジングについての詳細は Quay 3.x Sizing Guidlines を参照してください。

注記

Red Hat Enterprise Linux (RHEL) 8 は、Red Hat Quay 3.8 の高可用性で本番環境の品質のデプロイメントに推奨されます。RHEL 7 は Red Hat Quay 3.8 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。

2.1.1. 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。podman および関連技術の詳細は、Red Hat Enterprise Linux 8 のコンテナーの構築、実行、および管理 を参照してください。



重要

システムに Podman がインストールされていない場合は、同等の Docker コマンドを使用できる可能性がありますが、これは、推奨しません。Docker は Red Hat Quay 3.8 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。Podman は、Red Hat Quay 3.8 の高可用性と本番環境品質が必要なデプロイメントに推奨されます。

2.2. RED HAT QUAY の概念実証デプロイメントのための RED HAT ENTERPRISE LINUX の準備

以下の手順を使用して、Red Hat Quay の概念実証デプロイメント用に Red Hat Enterprise Linux (RHEL) を設定します。

2.2.1. RHEL サーバーのインストールおよび登録

以下の手順を使用して、Red Hat Quay の概念実証デプロイメント用に Red Hat Enterprise Linux (RHEL) サーバーを設定します。

手順

1. 最新の RHEL 8 サーバーをインストールします。最小インストール (シェルアクセスのみ) を実行するか、Server plus GUI (デスクトップが必要な場合) を実行できます。
2. Red Hat Subscription-Manager を使用して RHEL システムを Red Hat Customer Portal に登録およびサブスクライブする方法 の説明に従って、RHEL サーバーシステムを登録およびサブスクライブします。
3. 以下のコマンドを入力して、システムを登録し、利用可能なサブスクリプションを一覧表示します。利用可能な RHEL サーバーのサブスクリプションを選択し、プール ID に割り当て、最新のソフトウェアにアップグレードします。

```
# subscription-manager register --username=<user_name> --password=<password>
# subscription-manager refresh
# subscription-manager list --available
# subscription-manager attach --pool=<pool_id>
# yum update -y
```

2.2.2. podman のインストール

次の手順を使用して、Podman をインストールします。

手順

- 次のコマンドを入力して、Podman をインストールします。

```
$ sudo yum install -y podman
```

- または、コンテナーソフトウェアパッケージの完全なセットをプルする **container-tools** モジュールをインストールできます。

```
$ sudo yum module install -y container-tools
```

2.2.3. レジストリー認証

以下の手順を使用して、Red Hat Quay の概念実証のためにレジストリーを認証します。

手順

1. Red Hat Container Registry の認証 手順に従って、**registry.redhat.io** への認証を設定します。認証を設定すると、**Quay** コンテナーをプルできます。

注記

これは、イメージが Quay.io でホストされていた以前のバージョンの Red Hat Quay とは異なります。

2. 次のコマンドを入力して、レジストリーにログインします。

```
$ sudo podman login registry.redhat.io
```

ユーザー名 と **パスワード** を入力するよう求められます。

2.2.4. ファイアウォールの設定

システムでファイアウォールを実行している場合は、Red Hat Quay へのアクセスが許可されるルールを追加する必要がある場合があります。次の手順を使用して、概念実証のデプロイメントのためにファイアウォールを設定します。

手順

- 必要なコマンドは、システムにマップしたポートによって異なります。次に例を示します。

```
$ firewall-cmd --permanent --add-port=80/tcp
$ firewall-cmd --permanent --add-port=443/tcp
$ firewall-cmd --permanent --add-port=5432/tcp
$ firewall-cmd --permanent --add-port=5433/tcp
$ firewall-cmd --permanent --add-port=6379/tcp
$ firewall-cmd --reload
```

2.2.5. IP アドレスおよび命名サービス

相互に通信できるように Red Hat Quay でコンポーネントコンテナーを設定するには、いくつかの方法があります。次に例を示します。

- **コンテナーの IP アドレスの使用。podman inspect** を使用してコンテナーの IP アドレスを特定し、接続文字列を指定するときに設定ツールで値を使用できます。次に例を示します。

```
$ sudo podman inspect -f "{{.NetworkSettings.IPAddress}}" postgresql-quay
```

この方法は、コンテナーの IP アドレスが再起動後に変更されるためにホストの再起動の影響を受けます。

- **ネームサービスの使用。**デプロイメントをコンテナーの再起動後も存続させたい場合は、つまり IP アドレスが変更されることとなりますが、ネーミングサービスを実装できます。たとえば、dnsname プラグインは、コンテナーが名前で相互を解決できるように使用します。

- **ホストネットワークの使用。podman run** コマンドを **--net=host** オプションと共に使用してから、アドレスを設定に指定する際にホストでコンテナーポートを使用できます。このオプションは、2 つのコンテナーが同じポートを使用する必要がある場合に、ポートの競合の影響を受けやすくなります。この方法は、推奨しません。

- **ポートマッピングの設定。**ポートマッピングを使用してホスト上のポートを公開し、これらのポートをホストの IP アドレスまたはホスト名と組み合わせて使用できます。

本書では、ポートマッピングを使用し、ホストシステムの静的 IP アドレスを使用することを前提としています。導入全体を通して、**quay-sever.example.com** が **192.168.1.112** IP アドレスで使用されます。この情報は、**/etc/hosts** ファイルで確立されます。たとえば、次のようになります。

```
$ cat /etc/hosts
```

出力例:

```
192.168.1.112   quay-server.example.com
```

表2.1 ポートマッピングの概念実証のサンプル

コンポーネント	ポートマッピング	アドレス
Quay	**-p 80:8080 -p 443:8443**	http://quay-server.example.com
Postgres for Quay	**-p 5432:5432**	quay-server.example.com:5432
Redis	**-p 6379:6379**	quay-server.example.com:6379
Postgres for Clair V4	**-p 5433:5432**	quay-server.example.com:5433
Clair V4	**-p 8081:8080**	http://quay-server.example.com:8081

2.3. データベースの設定

Red Hat Quay には、メタデータを保存するためのデータベースが必要です。Postgres は本書全体で使用され、高可用性設定での使用が推奨されています。また、以下で説明されている設定と同様の方法で MySQL を使用できます。

2.3.1. Postgres のセットアップ

Red Hat Quay の概念実証では、ローカルファイルシステム上のディレクトリーを使用してデータベースデータを永続化します。

手順

1. ここでは **$QUAY** 変数で示されているインストールフォルダーに、次のコマンドを入力して、データベースデータ用のディレクトリーを作成します。

```
$ mkdir -p $QUAY/postgres-quay
```

2. 次のコマンドを入力して、適切な権限を設定します。

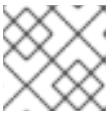
```
$ setfacl -m u:26:-wx $QUAY/postgres-quay
```

3. データベースデータのボリューム定義を使用して、ユーザー名、パスワード、およびデータベース名とポートを指定して、**Postgres** コンテナーを起動します。

```
$ sudo podman run -d --rm --name postgresql-quay \
  -e POSTGRESQL_USER=quayuser \
  -e POSTGRESQL_PASSWORD=quaypass \
  -e POSTGRESQL_DATABASE=quay \
  -e POSTGRESQL_ADMIN_PASSWORD=adminpass \
  -p 5432:5432 \
  -v $QUAY/postgres-quay:/var/lib/pgsql/data:Z \
  registry.redhat.io/rhel8/postgresql-10
```

4. 次のコマンドを実行して、Postgres **pg_trgm** モジュールがインストールされていることを確認します。

```
$ sudo podman exec -it postgresql-quay /bin/bash -c 'echo "CREATE EXTENSION IF NOT EXISTS pg_trgm" | psql -d quay -U postgres'
```



注記

Quay コンテナーには **pg_trgm** モジュールが必要です。

2.4. REDIS の設定

Redis は、ライブビルダーログおよび Red Hat Quay チュートリアル用に Quay によって使用される key-value ストアです。

2.4.1. Redis のセットアップ

以下の手順を使用して、Red Hat Quay の概念実証用の **Redis** コンテナーをデプロイします。

手順

- 次のコマンドを入力して、ポートとパスワードを指定して **Redis** コンテナーを起動します。

```
$ sudo podman run -d --rm --name redis \
  -p 6379:6379 \
  -e REDIS_PASSWORD=strongpassword \
  registry.redhat.io/rhel8/redis-6
```

2.5. RED HAT QUAY の設定

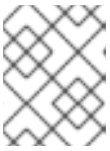
次の手順を使用して、レジストリー設定、データベース、Redis 接続パラメーターなど、すべてのコンポーネントの詳細を示す設定ファイルを生成します。

手順

1. 設定ファイルを生成するには、次のコマンドを入力して、**Quay** コンテナーを **設定** モードで実行します。パスワード (文字列 **secret** など) を指定する必要があります。

```
$ sudo podman run --rm -it --name quay_config -p 80:8080 -p 443:8443
registry.redhat.io/quay/quay-rhel8:v3.8.15 config secret
```

2. ブラウザーを使用して、**http://quay-server.example.com** で設定ツールのユーザーインターフェイスにアクセスします。

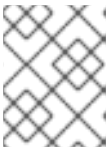


注記

本書では、**quay-server.example.com** ホスト名を **/etc/hosts** ファイルに設定しています。

3. ユーザー名とパスワードを指定してログイン

4. Red Hat Quay の設定 のステップ 1 で設定したユーザー名とパスワードでログインします。



注記

この手順に従った場合、ユーザー名は **quayconfig** で、パスワードは **secret** です。

2.5.1. Red Hat Quay のセットアップ

Red Hat Quay 設定エディターで、次の認証情報を入力する必要があります。

- 基本設定
- サーバー設定
- データベース
- Redis

2.5.1.1. 基本設定

Basic Configuration の下で、**Registry Title** フィールドと **Registry Title Short** フィールドに入力します。デフォルト値が設定されていればそのデフォルト値を使用できます。

2.5.1.2. サーバー設定

Server Hostname の下で、ネットワーク上でレジストリーにアクセスできる場所の HTTP ホストとポートを指定します。

本書の手順に従っている場合は、**quay-server.example.com** を入力します。

2.5.1.3. データベース

Database セクションで、Red Hat Quay がメタデータを保存するために使用するデータベースの接続の詳細を指定します。

このドキュメントの手順に従って概念実証システムをデプロイした場合は、次の値を入力します。

- **Database Type:** Postgres
- **Database Server:** quay-server.example.com:5432
- **Username:** quayuser
- **Password:** quaypass
- **Database Name:** quay

2.5.1.4. Redis

Redis の key-value ストアは、リアルタイムイベントとビルドログを保管するために使用されます。

このドキュメントの手順に従って概念実証システムをデプロイした場合は、**Redis** セクションに次の認証情報を入力します。

- **Redis Hostname:** quay-server.example.com
- **Redis port:** 6379 (デフォルト)
- **Redis password:** strongpassword

2.5.2. 設定の検証およびダウンロード

すべての必須フィールドが設定されたら、**Validate Configuration Changes** ボタンを選択して、設定を検証します。エラーが報告される場合は、設定が有効となり、Red Hat Quay がデータベースおよび Redis サーバーに接続できるまで、設定の編集を継続します。

検証後、**Configuration** ファイルをダウンロードします。設定エディターを実行している **Quay** コンテナーを停止します。

2.6. RED HAT QUAY のデプロイ

2.6.1. 前提条件

- Red Hat Quay データベースが実行されている。
- Redis サーバーが実行されている。
- 有効な設定ファイルを生成している。
- 設定エディターを実行していた **Quay** コンテナーを停止している。

2.6.2. 設定フォルダーの準備

以下の手順を使用して、Red Hat Quay 設定フォルダーを準備します。

手順

1. Red Hat Quay 設定バンドルをコピーするディレクトリーを作成します。

```
$ mkdir $QUAY/config
```

2. 生成された Red Hat Quay 設定バンドルをディレクトリーにコピーします。

```
$ cp ~/Downloads/quay-config.tar.gz ~/config
```

3. ディレクトリーに移動します。

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

4. Red Hat Quay 設定バンドルを解凍します。

```
$ tar xvf quay-config.tar.gz
```

2.6.3. イメージデータ用のローカルストレージの準備

次の手順を使用して、レジストリーイメージを保存するローカルファイルシステムを設定します。

手順

1. 次のコマンドを入力して、レジストリーイメージを保存するローカルディレクトリーを作成します。

```
$ mkdir $QUAY/storage
```

2. レジストリーイメージを保存するディレクトリーを設定します。

```
$ setfacl -m u:1001:-wx $QUAY/storage
```

2.6.4. Red Hat Quay レジストリーのデプロイ

1. 次の手順を使用して、**Quay** レジストリーコンテナーをデプロイします。

2. 次のコマンドを入力して、設定データ用の適切なボリュームとイメージデータ用のローカルストレージを指定して、**Quay** レジストリーコンテナーを起動します。

```
$ 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:v3.8.15
```

2.7. RED HAT QUAY の使用

以下の手順を実行すると、ユーザーインターフェイスを使用して、新しい組織およびリポジトリーを作成し、既存のリポジトリーを検索および参照できます。手順 3 の後に、コマンドラインインターフェイスを使用してレジストリーと対話し、イメージのプルおよびプッシュを実行できます。

1. ブラウザーを使用して、**http://quay-server.example.com** で Red Hat Quay レジストリーのユーザーインターフェイスにアクセスします (**quay-server.example.com** を **/etc/hosts** ファイルのホスト名として設定していることを前提とします)。

2. **Create Account** をクリックし、ユーザーを追加します (例: **quayadmin** とパスワード **password**)。

3. コマンドラインで、レジストリーにログインします。

```
$ sudo podman login --tls-verify=false quay-server.example.com
Username: quayadmin
Password: password
Login Succeeded!
```

2.7.1. イメージのプッシュおよびプル

1. Red Hat Quay レジストリーからイメージのプッシュおよびプルをテストするには、まず外部レジストリーからサンプルイメージをプルします。

```
$ sudo podman pull busybox
Trying to pull docker.io/library/busybox...
Getting image source signatures
Copying blob 4c892f00285e done
Copying config 22667f5368 done
Writing manifest to image destination
Storing signatures
22667f53682a2920948d19c7133ab1c9c3f745805c14125859d20cede07f11f9
```

2. **podman images** コマンドを使用して、ローカルコピーを表示します。

```
$ sudo podman images
REPOSITORY                      TAG      IMAGE ID       CREATED         SIZE
docker.io/library/busybox       latest   22667f53682a   14 hours ago    1.45 MB
...
```

3. このイメージにタグを付け、これを Red Hat Quay レジストリーにプッシュできるようにします。

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

4. 次に、イメージを Red Hat Quay レジストリーにプッシュします。この手順の後に、ブラウザーを使用して、リポジトリーでタグ付けされたイメージを確認できます。

```
$ sudo podman push --tls-verify=false quay-server.example.com/quayadmin/busybox:test
Getting image source signatures
Copying blob 6b245f040973 done
Copying config 22667f5368 done
Writing manifest to image destination
Storing signatures
```

5. コマンドラインからイメージへのアクセスをテストするには、まずイメージのローカルコピーを削除します。

```
$ sudo podman rmi quay-server.example.com/quayadmin/busybox:test
Untagged: quay-server.example.com/quayadmin/busybox:test
```

6. 今度は Red Hat Quay レジストリーからイメージを再度プルします。

```
$ sudo podman pull --tls-verify=false quay-server.example.com/quayadmin/busybox:test
```

```
Trying to pull quay-server.example.com/quayadmin/busybox:test...
Getting image source signatures
Copying blob 6ef22a7134ba [--------------------------------------] 0.0b / 0.0b
Copying config 22667f5368 done
Writing manifest to image destination
Storing signatures
22667f53682a2920948d19c7133ab1c9c3f745805c14125859d20cede07f11f9
```

第3章 RED HAT QUAY の高度なデプロイメント

以下のセクションを使用して、高度な Red Hat Quay を設定します。

3.1. SSL/TLS の使用

自己署名証明書 で Red Hat Quay を設定するには、認証局 (CA) を作成し、必要なキーおよび証明書ファイルを生成する必要があります。

注記

以下の例では、**/etc/hosts** ファイルにエントリーを追加するなど、DNS または別の命名メカニズムを使用してサーバーホスト名 **quay-server.example.com** を設定していることを前提としています。

```
$ cat /etc/hosts
...
192.168.1.112   quay-server.example.com
```

3.1.1. 認証局の作成と証明書への署名

次の手順を使用して、**ssl.cert** および **ssl.key** という名前の証明書ファイルとプライマリーキーファイルを作成します。

3.1.1.1. 認証局の作成

認証局 (CA) を作成するには、次の手順を使用します。

手順

1. 次のコマンドを入力して、ルート CA キーを生成します。

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

2. 次のコマンドを入力して、ルート CA 証明書を生成します。

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

3. サーバーのホスト名など、証明書の要求に組み込まれる情報を入力します。以下に例を示します。

```
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
```

3.1.1.2. 証明書に署名する

証明書に署名するには、次の手順を使用します。

手順

1. 次のコマンドを入力してサーバーキーを生成します。

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

2. 次のコマンドを入力して、署名リクエストを生成します。

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

3. サーバーのホスト名など、証明書の要求に組み込まれる情報を入力します。以下に例を示します。

```
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. 以下のようにサーバーのホスト名を指定して、設定ファイルの **openssl.cnf** を作成します。

   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. 設定ファイルを使用して、証明書 **ssl.cert** を生成します。

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

3.1.2. Red Hat Quay UI を使用した SSL/TLS の設定

Red Hat Quay UI を使用して SSL/TLS を設定するには、次の手順を実行します。

コマンドラインインターフェイスを使用して SSL を設定するには、コマンドラインインターフェイスを使用した SSL/TLS の設定を参照してください。

前提条件

- 認証局を作成し、証明書に署名しました。

手順

1. **Quay** コンテナーを設定モードで起動します。

```
$ sudo podman run --rm -it --name quay_config -p 80:8080 -p 443:8443
registry.redhat.io/quay/quay-rhel8:v3.8.15 config secret
```

2. **Server Configuration** セクションで、**Red Hat Quay handles TLS**を選択します。前に作成した証明書ファイルと秘密キーファイルをアップロードし、**Server Hostname** 証明書の作成時に使用された値と一致することを確認します。

3. 更新された設定を検証およびダウンロードします。

4. 次のコマンドを入力して、**Quay** コンテナーを停止し、レジストリーを再起動します。

```
$ 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:v3.8.15
```

3.1.3. コマンドラインインターフェイスを使用した SSL の設定

コマンドラインインターフェイスを使用して SSL/TLS を設定するには、次の手順を実行します。

前提条件

- 認証局を作成し、証明書に署名しました。

手順

1. 証明書ファイルとプライマリーキーファイルを設定ディレクトリーにコピーして、それぞれ **ssl.cert** と **ssl.key** という名前が付けられていることを確認します。

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

2. 次のコマンドを入力して、**$QUAY/config** ディレクトリーに移動します。

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

3. **config.yaml** ファイルを編集し、Red Hat Quay が TLS/SSL を処理するように指定します。

config.yaml

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

4. オプション: 次のコマンドを入力して、rootCA.pem ファイルの内容を ssl.cert ファイルの末尾に追加します。

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

5. 次のコマンドを入力して、**Quay** コンテナーを停止します。

```
$ sudo podman stop quay
```

6. 次のコマンドを入力してレジストリーを再起動します。

```
$ 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:v3.8.15
```

3.1.4. コマンドラインを使用した SSL 設定のテスト

- **podman login** コマンドを使用して、SSL が有効になっている Quay レジストリーへのログインを試みます。

```
$ sudo podman login quay-server.example.com
Username: quayadmin
Password:

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
```

- Podman は自己署名証明書を信頼しません。回避策として、**--tls-verify** オプションを使用します。

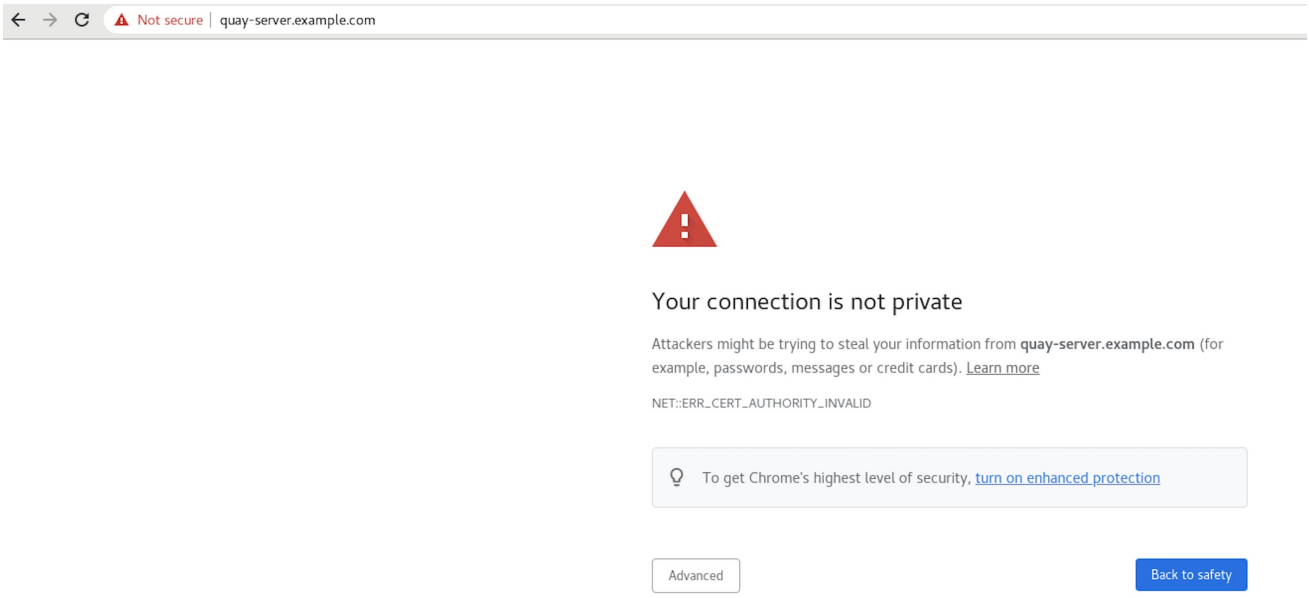
```
$ sudo podman login --tls-verify=false quay-server.example.com
Username: quayadmin
Password:

Login Succeeded!
```

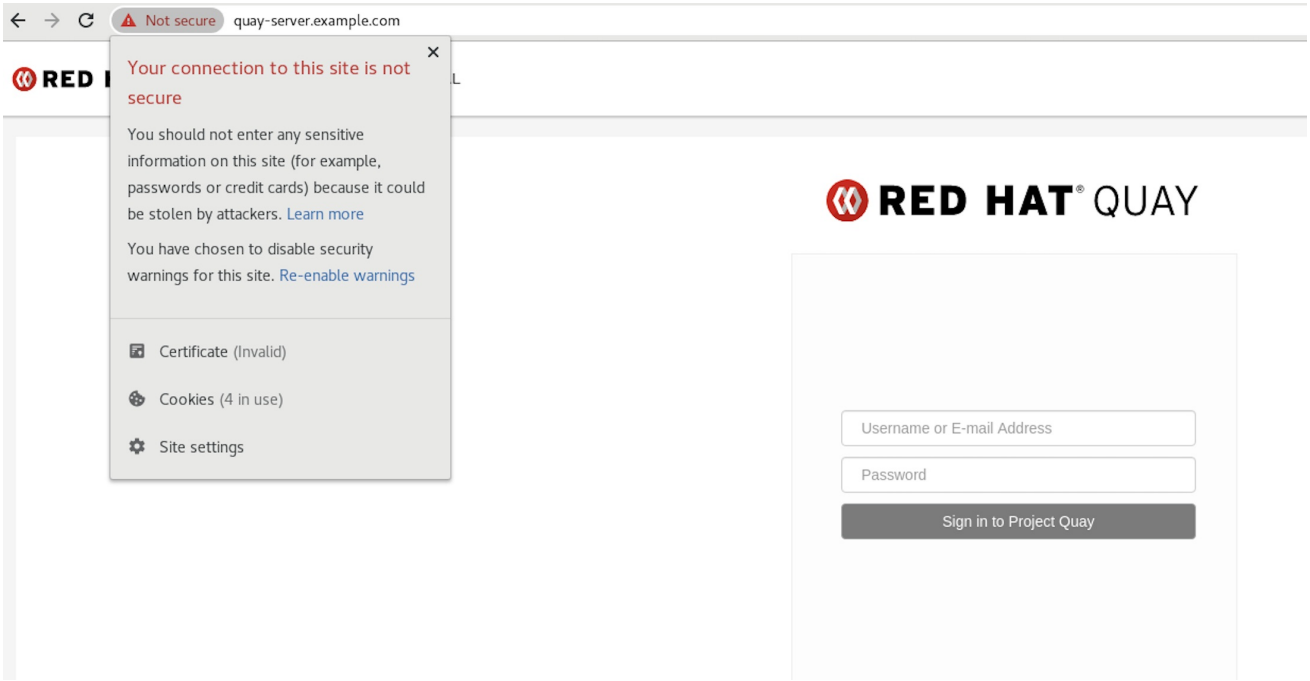
ルート認証局 (CA) を信頼するように Podman を設定する方法は、後続のセクションで説明します。

3.1.5. ブラウザーを使用した SSL 設定のテスト

Quay レジストリーへのアクセスを試みると (この場合は **https://quay-server.example.com**)、ブラウザーは潜在的なリスクについて警告します。



画面にログインすると、ブラウザーは接続が安全ではないことを通知します。



ルート認証局 (CA) を信頼するようにシステムを設定する方法は、後続のセクションで説明します。

3.1.6. 認証局を信頼するように podman を設定する

Podman は、**/etc/containers/certs.d/** および **/etc/docker/certs.d/** の 2 つのパスを使用して CA ファイルを見つけます。

- ルート CA ファイルをこれらの場所のいずれかにコピーし、サーバーのホスト名によって判別されるパスを使用して、**ca.crt** ファイルに名前を付けます。

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

- または、Docker を使用している場合は、ルート CA ファイルを同等の Docker ディレクトリーにコピーします。

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

レジストリーにログインする際に、**--tls-verify=false** オプションを使用する必要がなくなります。

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

Username: quayadmin
Password:
Login Succeeded!
```

3.1.7. 認証局を信頼するようにシステムを設定

認証局を信頼するようにシステムを設定するには、次の手順を使用します。

手順

1. 次のコマンドを入力して、**rootCA.pem** ファイルをシステム全体の統合トラストストアにコピーします。

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

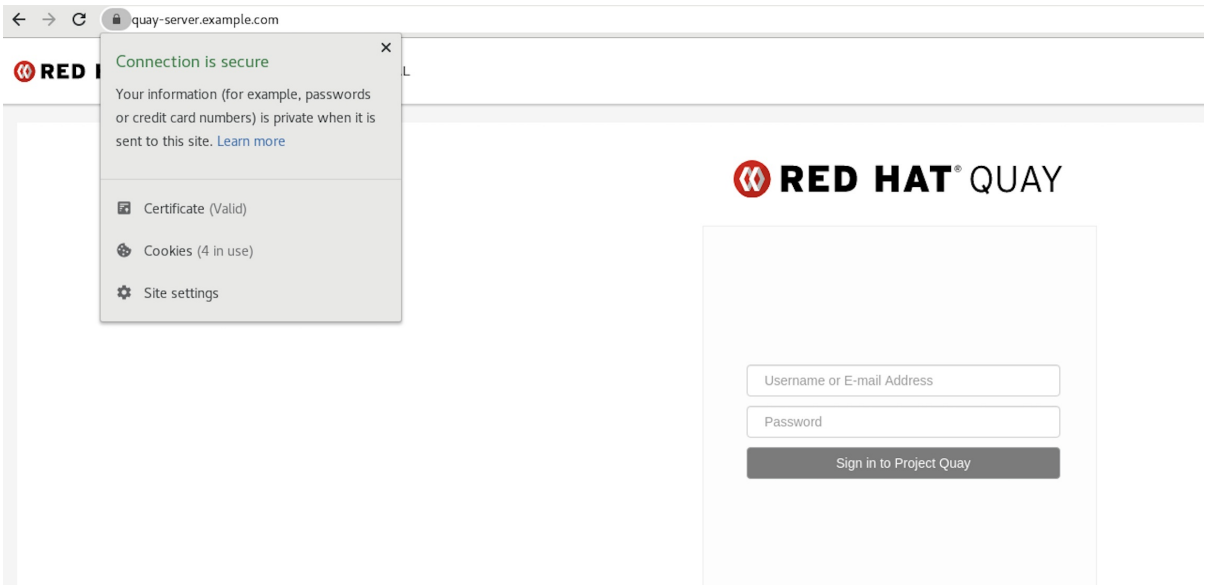
2. 次のコマンドを入力して、システム全体のトラストストア設定を更新します。

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

3. オプション:**trust list** コマンドを使用して、**Quay** サーバーが設定されていることを確認できます。

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

https://quay-server.example.com でレジストリーを参照すると、接続が安全であることを示すロックアイコンが表示されます。



4. **rootCA.pem** ファイルをシステム全体の信頼から削除するには、ファイルを削除して設定を更新します。

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

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

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

詳細は、RHEL 8 のドキュメントの 共有システム証明書の使用 を参照してください。

3.2. RED HAT QUAY スーパーユーザー

superuser は、以下を実行することができる拡張された特権を持つ Quay ユーザーアカウントです。

- ユーザーの管理
- 組織の管理
- サービスキーの管理
- 変更ログの閲覧
- 使用状況ログのクエリー
- グローバルに表示されるユーザーメッセージの作成

3.2.1. UI を使用したスーパーユーザーの Quay への追加

本セクションでは、Quay UI を使用してスーパーユーザーを追加する方法を説明します。コマンドラインインターフェイスを使用してスーパーユーザーを追加するには、以下のセクションを参照してください。

1. **Quay** コンテナーを接続モードで起動し、既存の設定をボリュームとして読み込みます。

```
$ sudo podman run --rm -it --name quay_config \
  -p 8080:8080 \
  -p 443:8443 \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.8.15 config secret
```

2. UI の **Access Settings** セクションで、**Super Users** フィールドにユーザーの名前 (この例では **quayadmin**) を入力し、**Add** をクリックします。

3. **configuration** ファイルを検証し、ダウンロードしてから、設定モードで実行されている **Quay** コンテナーを終了します。**config.yaml** ファイルを設定ディレクトリーに展開し、**Quay** コンテナーをレジストリーモードで再起動します。

```
$ 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:v3.8.15
```

3.2.2. スーパーユーザーを追加するための config.yaml ファイルの編集

config.yaml ファイルを直接編集してスーパーユーザーを追加することもできます。スーパーユーザーアカウントの一覧は、**SUPER_USERS** フィールドの配列として保存されます。

- コンテナーレジストリーが実行中の場合はこれを停止し、**SUPER_USERS** 配列を **config.yaml** ファイルに追加します。

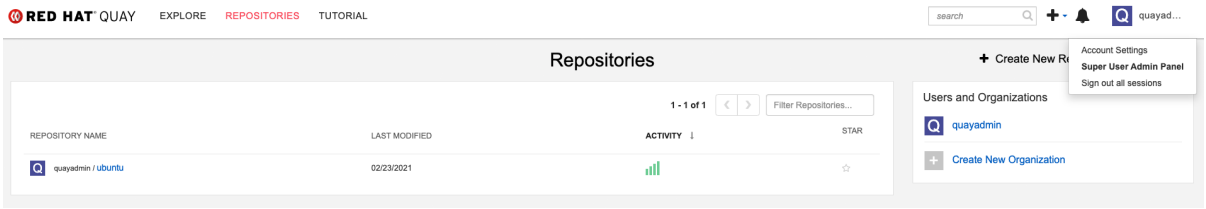
```
SERVER_HOSTNAME: quay-server.example.com
SETUP_COMPLETE: true
SUPER_USERS:
  - quayadmin
...
```

3.2.3. スーパーユーザー管理パネルへのアクセス

1. Quay レジストリーを再起動します。

```
$ 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:v3.8.15
```

2. Super User Admin パネルにアクセスするには、UI の右上にある現在のユーザーの名前またはアバターをクリックします。ユーザーがスーパーユーザーとして追加されている場合は、Super User Admin Panel というドロップダウンリストに追加のアイテムが表示されます。



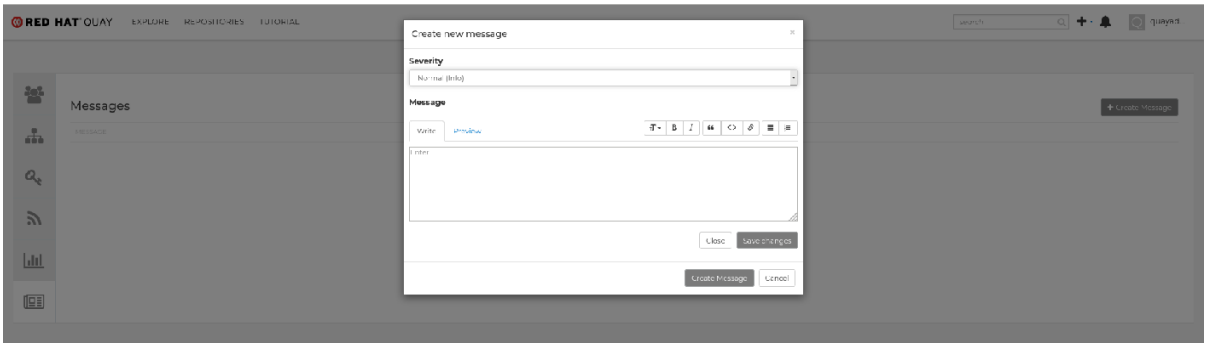
3.2.3.1. グローバルに表示されるユーザーメッセージの作成

Superuser Admin Panel を使用すると、組織の **Normal**、**Warning**、または **Error** メッセージを作成できます。

1. UI の右上でユーザー名をクリックします。**Super User Admin Panel** を選択します。

2. Red Hat Quay Management ページの、左側のペインで **Globally visible user messages** をクリックします。

3. **Create Message** をクリックし、**Normal**、**Warning**、および **Error** メッセージタイプを含むドロップダウンメニューを表示します。



4. メッセージの入力は、**Click to set message** を選択してから、**Create Message** をクリックして実行します。

メッセージの削除には、**Options** をクリックし、続いて **Delete Message** をクリックします。

3.3. リポジトリーのミラーリング

3.3.1. リポジトリーのミラーリング

Red Hat Quay リポジトリーミラーリングを使用すると、外部コンテナーレジストリー (または別のローカルレジストリー) から Red Hat Quay クラスターにイメージをミラーリングできます。リポジトリーミラーリングを使用すると、リポジトリー名とタグに基づいてイメージを Red Hat Quay に同期できます。

リポジトリーのミラーリングが有効になっている Red Hat Quay クラスターから、以下を実行できます。

- 外部のレジストリーからミラーリングするリポジトリーを選択する
- 外部レジストリーにアクセスするための認証情報を追加する
- 同期する特定のコンテナーイメージリポジトリー名とタグを特定する
- リポジトリーが同期される間隔を設定する
- 同期の現在の状態を確認する

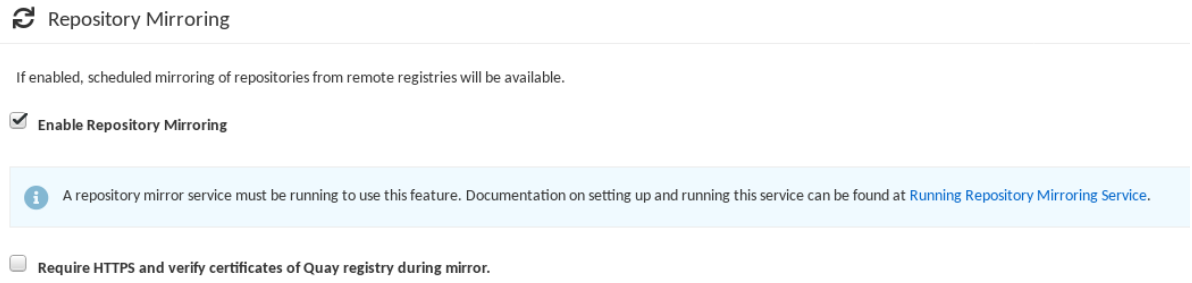
ミラーリング機能を使用するには、次のアクションを実行する必要があります。

- Red Hat Quay 設定ファイルでリポジトリーのミラーリングを有効にする
- リポジトリーミラーリングワーカーを実行する
- ミラーリングされたリポジトリーを作成します

すべてのリポジトリーのミラーリング設定は、設定ツール UI または Red Hat Quay API を使用して実行できます。

3.3.2. 設定のミラーリング UI

1. 設定モードで **Quay** コンテナーを起動し、Enable Repository Mirroring チェックボックスを選択します。HTTPS 通信を必要とし、ミラーリング時に証明書を検証する必要がある場合は、HTTPS を選択し、証明書の検証のチェックボックスを選択します。



2. **configuration** を検証し、ダウンロードしてから、更新された設定ファイルを使用してレジストリーモードで Quay を再起動します。

3.3.3. ミラーリングワーカー

次の手順を使用して、リポジトリーミラーリングワーカーを開始します。

手順

- **/root/ca.crt** 証明書を使用して TLS 通信を設定していない場合は、次のコマンドを入力し、**repomirror** オプションを指定して、**Quay** Pod を開始します。

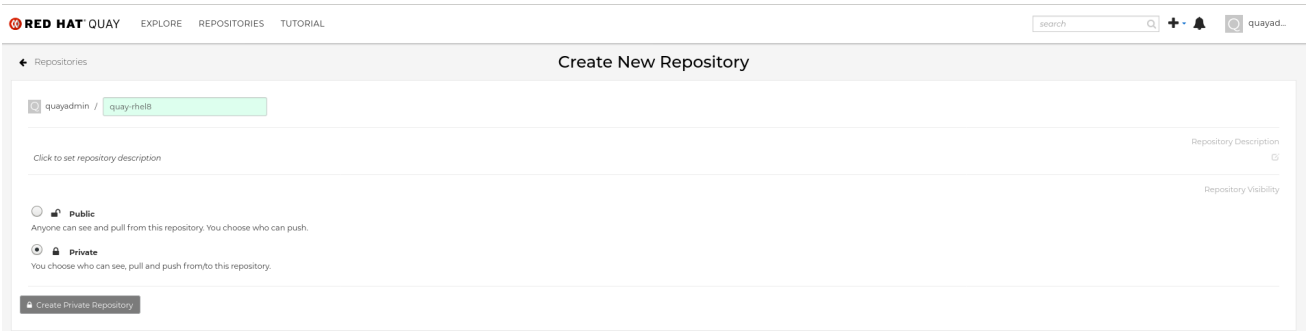
```
$ sudo podman run -d --name mirroring-worker \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.8.15 repomirror
```

- **/root/ca.crt** 証明書を使用して TLS 通信を設定している場合は、次のコマンドを入力して、リポジトリーミラーリングワーカーを開始します。

```
$ 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:v3.8.15 repomirror
```

3.3.4. ミラーリングされたリポジトリーの作成

外部コンテナーレジストリーからリポジトリーをミラーリングする場合は、新しいプライベートリポジトリーを作成する必要があります。通常、同じ名前がターゲットリポジトリーとして使用されます (例: **quay-rhel8**)。



3.3.4.1. リポジトリーのミラーリングの設定

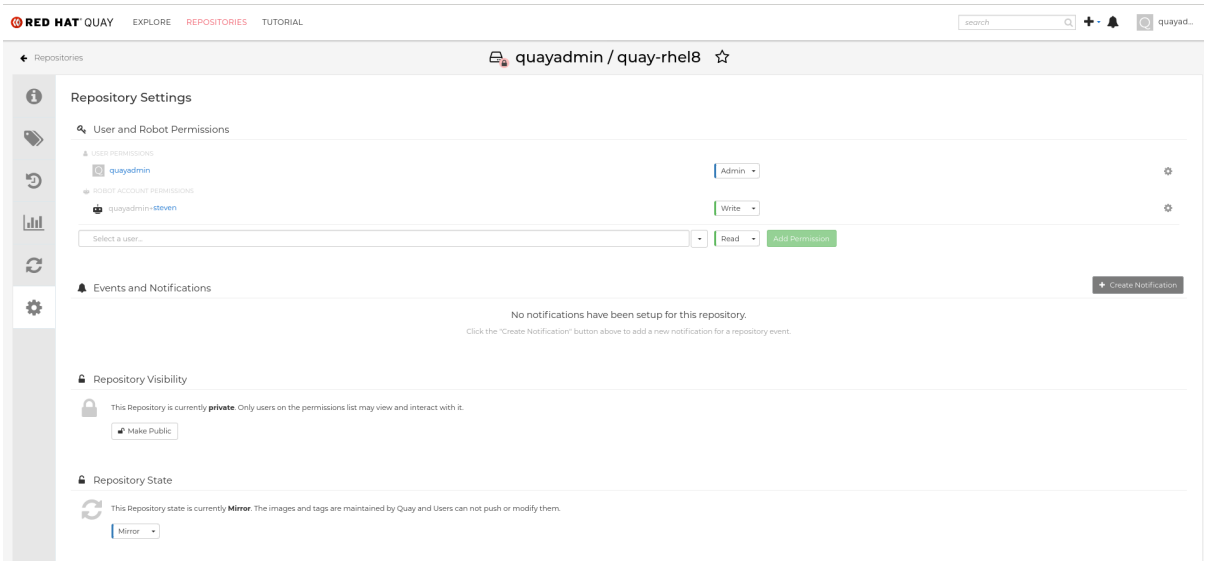
ミラーリングされたリポジトリーの設定を調整するには、次の手順を使用します。

前提条件

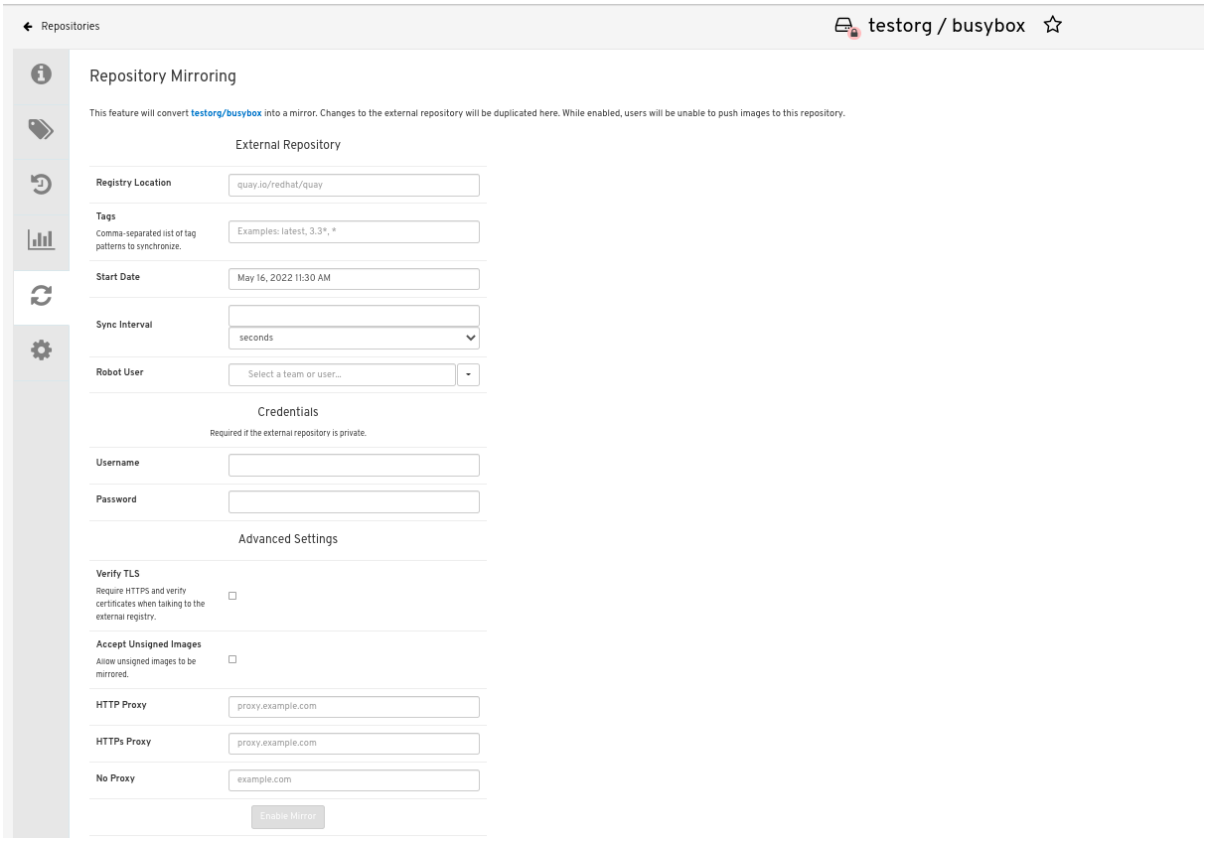
- Red Hat Quay 設定ファイルでリポジトリーミラーリングを有効にしました。
- ミラーリングワーカーをデプロイしました。

手順

1. Settings タブで、Repository State を **Mirror** に設定します。



2. Mirror タブで、タグ、スケジューリング、およびアクセス情報と共に外部レジストリーに接続するための情報を入力します。



3. 必要に応じて、以下のフィールドに詳細を入力します。

- **Registry Location**: ミラーリングする外部リポジトリー (例: **registry.redhat.io/quay/quay-rhel8**)。
- **Tags**: このフィールドは必須です。個別のタグまたはタグパターンのコンマ区切りの一覧を入力できます。(詳細は、**タグパターン** のセクションを参照してください。)
- **Start Date**: ミラーリングが開始する日付。現在の日時がデフォルトで使用されます。
- **Sync Interval**: デフォルトで 24 時間ごとの同期に設定されます。これは時間または日に基づいて変更できます。

- **Robot User**: 新しい robot アカウントを作成するか、既存の robot アカウントを選択してミラーリングを実行します。
- **Username**: ミラーリングするリポジトリーを保持する外部レジストリーにアクセスするためのユーザー名。
- **Password**: ユーザー名に関連付けられたパスワード。パスワードにはエスケープ文字 (\) を必要とする文字を含めることができないことに注意してください。

3.3.4.2. 詳細設定

Advanced Settings セクションでは、次のオプションを使用して SSL/TLS とプロキシーを設定できます。

- **Verify TLS:** ターゲットのリモートレジストリーと通信するときに HTTPS を要求し、証明書を検証する場合は、このオプションを選択します。
- **Accept Unsigned Images:** このオプションを選択すると、署名されていないイメージをミラーリングできます。
- **HTTP Proxy:** ターゲットのリモートレジストリーと通信するときに HTTPS を要求し、証明書を検証する場合は、このオプションを選択します。
- **HTTPS PROXY:** プロキシーサーバーが必要な場合は、リモートサイトにアクセスするために必要な HTTPS プロキシーサーバーを特定します。
- **No Proxy:** プロキシーを必要としない場所のリスト。

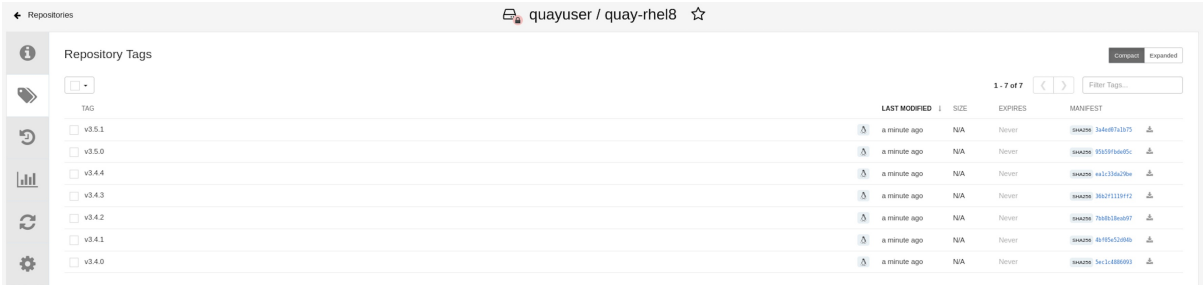
3.3.4.3. 今すぐ同期する

ミラーリング操作を開始するには、次の手順を使用します。

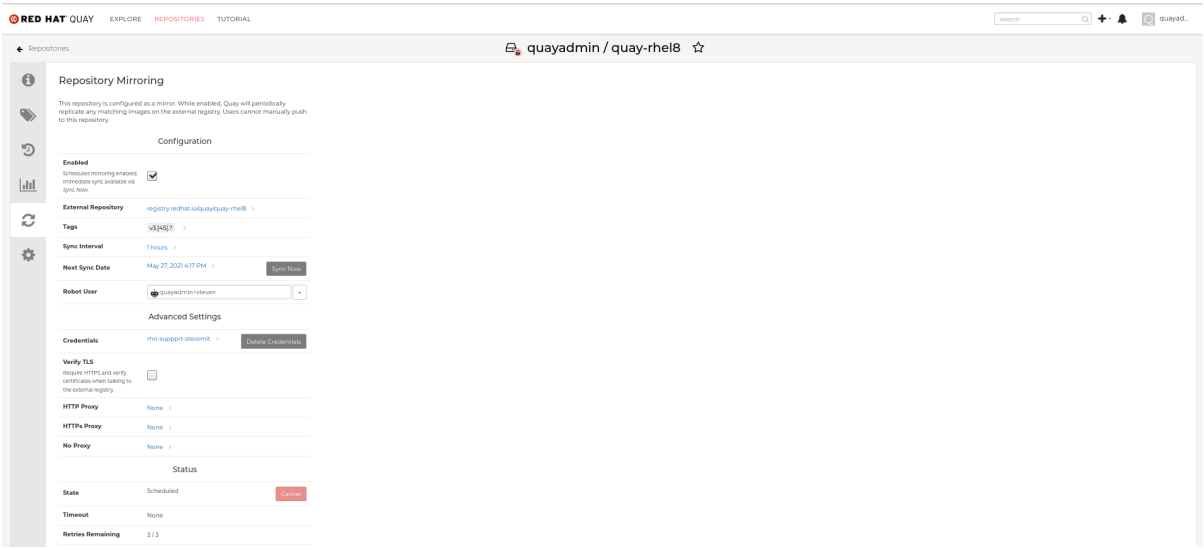
手順

- 即時にミラーリング操作を実行するには、リポジトリーの Mirroring タブで Sync Now ボタンを押します。ログは、Usage Logs タブで利用できます。

ミラーリングが完了すると、イメージは Tags タブに表示されます。



以下は、完了した Repository Mirroring 画面の例です。



3.3.5. タグパターンのミラーリング

少なくとも 1 つのタグを入力する必要があります。次の表は、考えられるイメージタグのパターンを示しています。

3.3.5.1. パターン構文

パターン	説明
*	すべての文字に一致します。
?	任意の単一文字に一致します。
[seq]	**seq** の任意の文字と一致します。
[!seq]	**seq** にない文字と一致します。

3.3.5.2. タグのパターン例

パターン例	マッチの例
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

3.4. RED HAT QUAY の CLAIR

Clair v4 (Clair) は、静的コード分析を活用してイメージコンテンツを解析し、コンテンツに影響を与える脆弱性を報告するオープンソースアプリケーションです。Clair は Red Hat Quay にパッケージ化されており、スタンドアロンと Operator デプロイメントの両方で使用できます。エンタープライズ環境に合わせてコンポーネントを個別にスケーリングできる、非常にスケーラブルな設定で実行できます。

3.4.1. スタンドアロンの Red Hat Quay デプロイメントでの Clair のセットアップ

スタンドアロンの Red Hat Quay デプロイメントの場合、Clair を手動でセットアップできます。

手順

1. Red Hat Quay インストールディレクトリーに、Clair データベースデータ用の新しいディレクトリーを作成します。

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

2. 次のコマンドを入力して、**postgres-clairv4** ファイルに適切な権限を設定します。

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

3. 次のコマンドを入力して、Clair Postgres データベースをデプロイします。

```
$ 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 5433:5433 \
  -v /home/<user-name>/quay-poc/postgres-clairv4:/var/lib/pgsql/data:Z \
  {postgresimage}
```

4. Clair デプロイメント用に Postgres **uuid-ossp** モジュールをインストールします。

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

出力例

```
CREATE EXTENSION
```

注記

Clair では、**uuid-ossp** 拡張機能を Postgres データベースに追加する必要があります。適切な権限を持つユーザーの場合、拡張機能を作成すると、Clair によって自動的に追加されます。ユーザーが適切な権限を持っていない場合は、Clair を開始する前に拡張機能を追加する必要があります。

拡張機能が存在しない場合、クレアが起動しようとすると、次のエラーが表示されます。**ERROR: Please load the "uuid-ossp" extension.(SQLSTATE 42501)**.

5. 実行中の場合は、**Quay** コンテナーを停止し、設定モードで再始動して、既存の設定をボリュームとしてロードします。

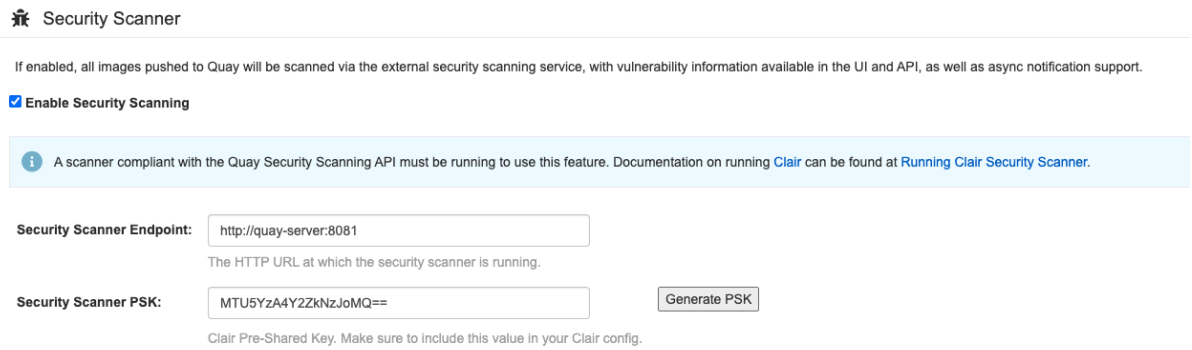
```
$ sudo podman run --rm -it --name quay_config \
  -p 80:8080 -p 443:8443 \
  -v $QUAY/config:/conf/stack:Z \
  {productrepo}/{quayimage}:{productminv} config secret
```

6. 設定ツールにログインし、UI の **Security Scanner** セクションで **Enable Security Scanning** をクリックします。

7. **quay-server** システムでまだ使用されていないポート (**8081** など) を使用して、Clair の HTTP エンドポイントを設定します。

8. **Generate PSK** ボタンを使用して、事前共有キー (PSK) を作成します。

セキュリティースキャナー UI



9. Red Hat Quay の **config.yaml** ファイルを検証してダウンロードし、設定エディターを実行している **Quay** コンテナーを停止します。

10. 新しい設定バンドルを Red Hat Quay インストールディレクトリーに解凍します。次に例を示します。

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

11. Clair 設定ファイル用のフォルダーを作成します。次に例を示します。

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

12. Clair 設定フォルダーに移動します。

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

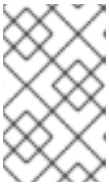
13. 以下のように、Clair 設定ファイルを作成します。

```
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
  run: ""
  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"
```

Clair の設定形式について詳しくは、Clair 設定リファレンス を参照してください。

14. コンテナーイメージを使用して Clair を起動し、作成したファイルから設定にマウントします。

```
$ 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:v3.8.15
```



注記

複数の Clair コンテナーを実行することもできます。ただし、単一のコンテナーを超えるデプロイシナリオでは、Kubernetes や OpenShift Container Platform などのコンテナーオーケストレーターを使用することを強く推奨します。

3.4.2. Clair のテスト

以下の手順を使用して、スタンドアロンの Red Hat Quay デプロイメントまたは OpenShift Container Platform Operator ベースのデプロイメントで Clair をテストします。

前提条件

- Clair コンテナーイメージをデプロイしました。

手順

1. 次のコマンドを入力して、サンプルイメージをプルします。

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

2. 次のコマンドを入力して、レジストリーにイメージをタグ付けします。

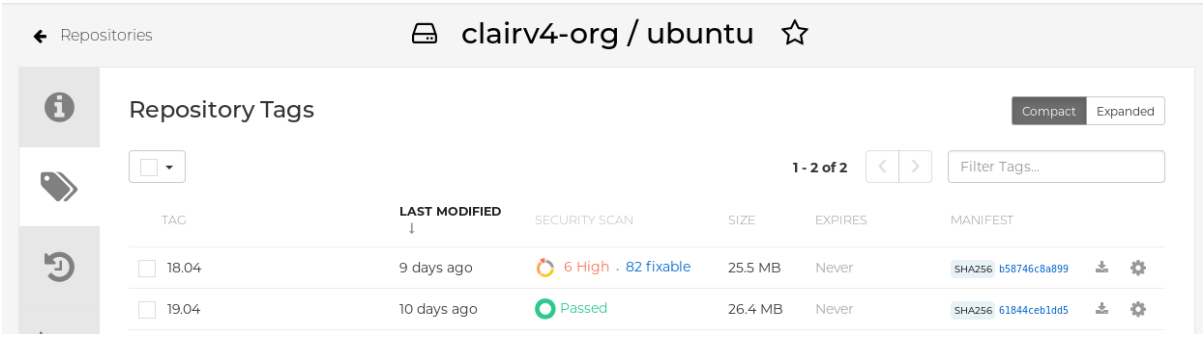
```
$ sudo podman tag docker.io/library/ubuntu:20.04 <quay-server.example.com>/<user-name>/ubuntu:20.04
```

3. 以下のコマンドを入力して、イメージを Red Hat Quay レジストリーにプッシュします。

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

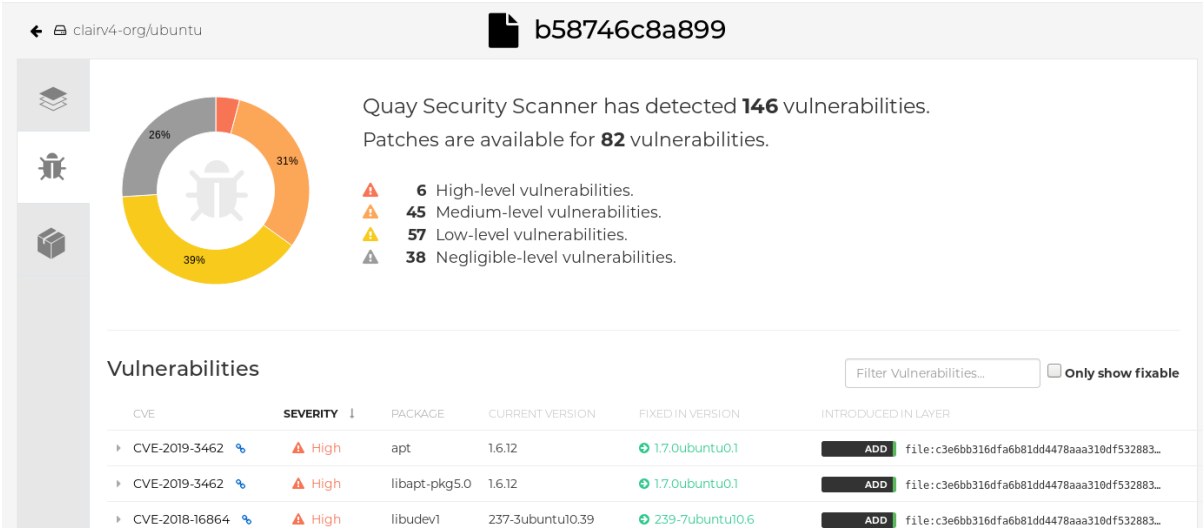
4. UI から Red Hat Quay デプロイメントにログインします。
5. リポジトリー名 (**quayadmin/ubuntu** など) をクリックします。
6. ナビゲーションウィンドウで、**Tags** をクリックします。

レポートの概要



7. イメージレポート (例: **45 medium**) をクリックして、より詳細なレポートを表示します。

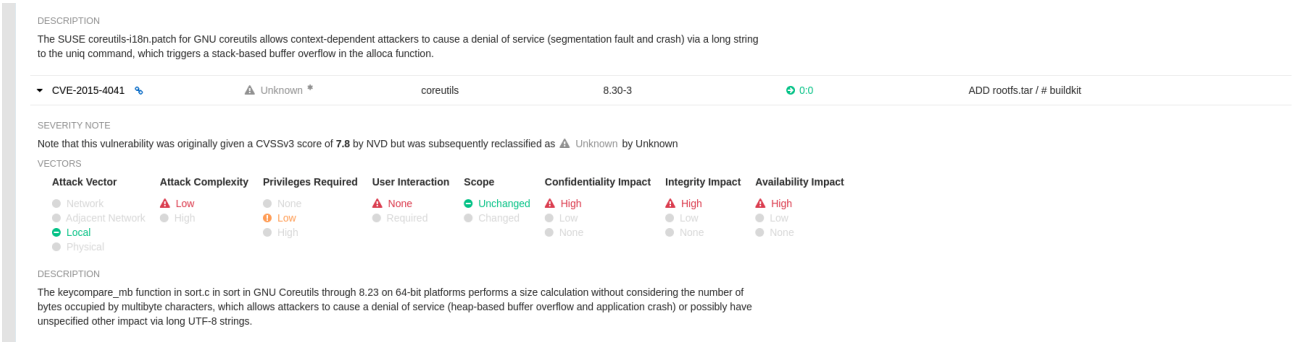
レポートの詳細



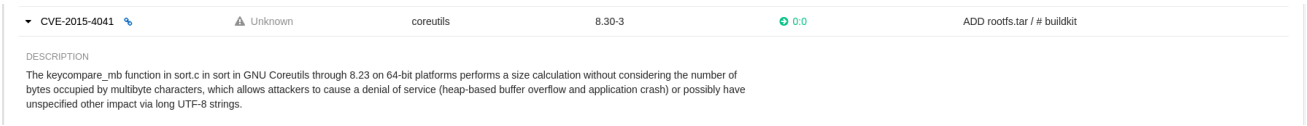
3.4.3. 脆弱性情報データベース (NVD: National Vulnerability Database) の CVE 評価

Clair v4.2 の時点で、Common Vulnerability Scoring System (CVSS) 強化データが Red Hat Quay UI で表示できるようになりました。さらに、Clair v4.2 は、検出された脆弱性について National Vulnerability Database から CVSS スコアを追加します。

今回の変更により、脆弱性の CVSS スコアがディストリビューションスコアの 2 レベル以内である場合、Red Hat Quay UI はデフォルトでディストリビューションのスコアを提示します。以下はその例です。



これは以前のインターフェイスとは異なり、以下の情報のみを表示します。



関連情報

- Red Hat Quay の Clair による脆弱性レポート

3.5. コンテナーの起動

--restart オプションは podman で完全にサポートされていないため、Podman を使用した systemd へのコンテナーの移植 で説明されているように、**podman** を systemd サービスとして設定できます。

3.5.1. Podman での systemd ユニットファイルの使用

デフォルトで、Podman は既存のコンテナーまたは Pod のユニットファイルを生成します。**podman**

generate systemd --new を使用して、移植可能な別の systemd ユニットファイルを生成できます。**--new** フラグでは、コンテナーの作成、起動、削除を行うユニットファイルを生成するように Podman に指示します。

1. 以下のように、実行中の Red Hat Quay レジストリーから systemd ユニットファイルを作成します。

```
$ sudo podman generate systemd --new --files --name redis
$ sudo podman generate systemd --new --files --name postgresql-quay
$ sudo podman generate systemd --new --files --name quay
$ sudo podman generate systemd --new --files --name postgresql-clairv4
$ sudo podman generate systemd --new --files --name clairv4
```

2. **/usr/lib/systemd/system** にユニットファイルをコピーして root ユーザーとしてインストールします。

```
$ sudo cp -Z container-redis.service /usr/lib/systemd/system
$ sudo cp -Z container-postgresql-quay.service /usr/lib/systemd/system
$ sudo cp -Z container-quay.service /usr/lib/systemd/system
$ sudo cp -Z container-postgresql-clairv4.service /usr/lib/systemd/system
$ sudo cp -Z container-clairv4.service /usr/lib/systemd/system
```

3. systemd マネージャーの設定を再読み込みするには、次のコマンドを実行します。

```
$ sudo systemctl daemon-reload
```

4. サービスを有効にし、システムの起動時に起動します。

```
$ sudo systemctl enable --now container-redis.service
$ sudo systemctl enable --now container-postgresql-quay.service
$ sudo systemctl enable --now container-quay.service
$ sudo systemctl enable --now container-postgresql-clairv4.service
$ sudo systemctl enable --now container-clairv4.service
```

3.5.2. サービスの起動、停止、およびステータスのチェック

1. Quay コンポーネントのステータスを確認します。

```
$ sudo systemctl status container-redis.service
$ sudo systemctl status container-postgresql-quay.service
$ sudo systemctl status container-quay.service
$ sudo systemctl status container-postgresql-clairv4.service
$ sudo systemctl status container-clairv4.service
```

2. Quay コンポーネントサービスを停止するには、以下を実行します。

```
$ sudo systemctl stop container-redis.service
$ sudo systemctl stop container-postgresql-quay.service
$ sudo systemctl stop container-quay.service
$ sudo systemctl stop container-postgresql-clairv4.service
$ sudo systemctl stop container-clairv4.service
```

3. Quay コンポーネントサービスを起動するには、以下を実行します。

```
$ sudo systemctl start container-redis.service
$ sudo systemctl start container-postgresql-quay.service
$ sudo systemctl start container-quay.service
$ sudo systemctl start container-postgresql-clairv4.service
$ sudo systemctl start container-clairv4.service
```

3.5.3. 再起動後の再起動のテスト

サービスを設定して有効にしたら、システムを再起動します。システムを再起動したら、**podman ps** を使用して Quay コンポーネントのすべてのコンテナーが再起動されていることを確認します。

```
$ sudo podman ps -a
CONTAINER ID  IMAGE                                COMMAND         CREATED         STATUS
PORTS                 NAMES
4e87c7889246  {postgresimage}    run-postgresql  19 seconds ago  Up 18 seconds ago
0.0.0.0:5432->5432/tcp  postgresql-quay
b8fbac1920d4  registry.redhat.io/rhel8/redis-6:1-110)          run-redis       19 seconds ago  Up 18
seconds ago  0.0.0.0:6379->6379/tcp  redis
d959d5bf7a24  {postgresimage}    run-postgresql  18 seconds ago  Up 18 seconds ago  0.0.0.0:5433-
>5432/tcp  postgresql-clairv4
e75ff8651dbd  registry.redhat.io/quay/clair-rhel8:v3.4.0                  18 seconds ago  Up 17 seconds
ago  0.0.0.0:8081->8080/tcp  clairv4
```

この場合、**Quay** コンテナー自体は起動できませんでした。これは、セキュリティースキャンが Quay で有効にされている場合に、起動時に Clair への接続を試みるためです。ただし、Clair は初期化を完了せず、接続を受け入れることができないため、結果として Quay はすぐに終了します。この問題を解決するには、以下のセクションにあるように、Quay サービスを Clair サービスの依存関係を持つように設定する必要があります。

3.5.4. Clair の Quay の依存関係の設定

Quay の **systemd** サービスファイルで、**After=container-clairv4.service** を設定して、**[Unit]** セクションに Clair サービスの依存関係を設定します。Clair コンテナーの初期化に時間を指定するには **[Service]** セクションに遅延を追加します (**RestartSec=30** など)。以下は、Clair の依存関係を設定した後に変更された Quay ファイルの例です。

/usr/lib/systemd/system/container-quay.service

```
# container-quay.service
# autogenerated by Podman 2.0.5
# Tue Feb 16 17:02:26 GMT 2021

[Unit]
Description=Podman container-quay.service
Documentation=man:podman-generate-systemd(1)
Wants=network.target
After=container-clairv4.service

[Service]
Environment=PODMAN_SYSTEMD_UNIT=%n
Restart=on-failure
RestartSec=30
ExecStartPre=/bin/rm -f %t/container-quay.pid %t/container-quay.ctr-id
ExecStart=/usr/bin/podman run --conmon-pidfile %t/container-quay.pid --cidfile %t/container-
```

```
quay.ctr-id --cgroups=no-conmon -d --rm -p 8080:8080 --name=quay -v
/home/user1/quay/config:/conf/stack:Z -v /home/user1/quay/storage:/datastorage:Z
registry.redhat.io/quay/quay-rhel8:v3.4.0
ExecStop=/usr/bin/podman stop --ignore --cidfile %t/container-quay.ctr-id -t 10
ExecStopPost=/usr/bin/podman rm --ignore -f --cidfile %t/container-quay.ctr-id
PIDFile=%t/container-quay.pid
KillMode=none
Type=forking

[Install]
WantedBy=multi-user.target default.target
```

Quay サービス設定を更新したら、サーバーを再起動して **podman ps** をすぐに実行します。

```
$ sudo podman ps -a
CONTAINER ID  IMAGE                                COMMAND         CREATED         STATUS
PORTS                 NAMES
4e87c7889246  {postgresimage}    run-postgresql  29 seconds ago  Up 28 seconds ago
0.0.0.0:5432->5432/tcp  postgresql-quay
b8fbac1920d4  registry.redhat.io/rhel8/redis-6:1-110)          run-redis       29 seconds ago  Up 28
seconds ago  0.0.0.0:6379->6379/tcp  redis
d959d5bf7a24  {postgresimage}    run-postgresql  28 seconds ago  Up 28 seconds ago  0.0.0.0:5433-
>5432/tcp  postgresql-clairv4
e75ff8651dbd  registry.redhat.io/quay/clair-rhel8:v3.4.0                  28 seconds ago  Up 27 seconds
ago  0.0.0.0:8081->8080/tcp  clairv4
```

最初は **Quay** コンテナーは利用できませんが、**RestartSec** の遅延の期限が切れると、起動するはずです。

```
$ sudo podman ps -a
CONTAINER ID  IMAGE                                COMMAND         CREATED         STATUS
PORTS                 NAMES
4e87c7889246  {postgresimage}    run-postgresql  35 seconds ago  Up 34 seconds ago
0.0.0.0:5432->5432/tcp  postgresql-quay
ab9f0e6ad7c3  registry.redhat.io/quay/quay-rhel8:v3.4.0   registry        3 seconds ago   Up 2 seconds
ago   0.0.0.0:8080->8080/tcp  quay
b8fbac1920d4  registry.redhat.io/rhel8/redis-6:1-110)          run-redis       35 seconds ago  Up 34
seconds ago  0.0.0.0:6379->6379/tcp  redis
d959d5bf7a24  {postgresimage}    run-postgresql  34 seconds ago  Up 34 seconds ago  0.0.0.0:5433-
>5432/tcp  postgresql-clairv4
e75ff8651dbd  registry.redhat.io/quay/clair-rhel8:v3.4.0                  34 seconds ago  Up 33 seconds
ago  0.0.0.0:8081->8080/tcp  clairv4
```

Quay コンテナーの **CREATED** フィールドには、サービス定義に設定されるように作成時間に 30 秒の差異が示されます。

quay-server.example.com で Red Hat Quay レジストリーにログインし、すべてが正常に再起動されていることを確認します。

3.6. 連邦情報処理標準 (FIPS) の準備と準拠

米国国立標準技術研究所 (NIST) によって開発された連邦情報処理標準 (FIPS) は、特に銀行、医療、公共部門などの高度に規制された分野で、機密データを保護および暗号化するために高く評価されていると見なされています。Red Hat Enterprise Linux (RHEL) および OpenShift Container Platform は FIPS

モード を提供することで FIPS 標準をサポートします。このモードでは、システムは **openssl** などの特定の FIPS 検証済み暗号モジュールの使用のみを許可します。これにより、FIPS への準拠が保証されます。

Red Hat Quay は、Red Hat Quay バージョン 3.5.0 からの FIPS 対応の RHEL および OpenShift Container Platform 環境での実行をサポートします。

第4章 次のステップ

本書では、概念実証用の Red Hat Quay を設定し、デプロイする方法を説明します。実稼働環境へのデプロイに関する詳細は、Red Hat Quay のデプロイ - 高可用性 (HA) を参照してください。

Red Hat Quay の使用ガイドでは、以下の方法について説明しています。

- ユーザーおよびリポジトリーの追加
- タグの使用
- ビルドワーカーを使用した Dockerfile の自動ビルド
- GitLab ビルドトリガー
- リポジトリーイベントの通知の追加

Red Hat Quay の管理ガイドでは、以下の方法について説明しています。

- SSL および TLS の使用
- Clair によるセキュリティースキャンの有効化
- リポジトリーミラーリングの使用
- LDAP 認証の設定
- ストレージの Georeplication の使用