Red Hat Quay 3.4

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: 2021-11-16

Red Hat Quay 3.4 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

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

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、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 の機能は次のとおりです。

- 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(multiarch)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 は、内部と外部の両方で数多くのコアコンポーネントで構成されます。

1.1.1. 内部コンポーネント

- **Quay (container registry)**: Pod の複数のコンポーネントで構成される quay コンテナーをサービスとして実行します。
- **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

- **Database**: Red Hat Quay で、（イメージストレージ用にではなく）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。
- **Redis (key-value store)**: ライブビルダーログおよび Red Hat Quay チュートリアルを保存します。
- **Cloud storage**: サポートされるデプロイメントでは、以下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。

- **Public cloud storage**: パブリッククラウド環境では、Amazon S3（AWS 用）または Google Cloud Storage（Google Cloud 用）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。
- **Private cloud storage**: 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、以下の仕様にて、単一のマシン（物理または仮想）で実稼働以外の目的でデプロイできます。

2.1. 前提条件

- **Red Hat Enterprise Linux (RHEL)**: Downloads page から、最新の Red Hat Enterprise Linux 8 サーバーメディアを取得し、Product Documentation for Red Hat Enterprise Linux 8 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。
- **CPUs**: 2 つ以上の仮想 CPU
- **RAM**: 4GB 以上
- **Disk space**: 必要なディスク領域は、レジストリーのストレージのニーズによって異なります。テストシステムには約 30 GB のディスク領域があれば十分です（以下は内訳です）。
  - 10GB 以上: オペレーティングシステム(Red Hat Enterprise Linux Server)用のディスク領域
  - 10GB 以上: docker ストレージ用のディスク容量（3 つのコンテナーを実行するために）
  - 10GB 以上: Quay ローカルストレージ用のディスク領域（CEPH またはその他のローカルストレージに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。

2.1.1. Pod 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必要があります。Podman および関連テクノロジーの詳細は、Building, running, and managing Linux containers on Red Hat Enterprise Linux 8 を参照してください。

2.2. RHEL サーバーの設定

2.2.1. Red Hat Enterprise Linux サーバーのインストールおよび登録

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、またはServer plus GUI（デスクトップが必要な場合）を実行できます。How to register and subscribe a system... で説明されているように、RHEL サーバーシステムを登録し、サブスクライブします。以下のコマンドは、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

Podman を、システムにインストールされていない場合は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

Red Hat Container Registry Authentication で説明されているように、quay コンテナーをプルできるように、**registry.redhat.io** への認証をセットアップします。これは、イメージが quay.io でホストされる以前の Red Hat Quay リリースとは異なることに注意してください。

たとえば、レジストリーにログインすることができます。

```
$ sudo podman login registry.redhat.io
Username: <username>
Password: <password>
```

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が実行されている場合、ローカルシステムの外部にある Red Hat Quay 設定ツール（ポート 8443）およびアプリケーション（ポート 8080 および 443）にアクセスするには、以下のコマンドを実行します（特定のゾーンでポートを開くには、各コマンドに **--zone=<yourzone>** を追加します）。

```
# firewall-cmd --permanent --add-port=8443/tcp
# firewall-cmd --permanent --add-port=8080/tcp
# firewall-cmd --permanent --add-port=443/tcp
# firewall-cmd --reload
```

2.2.5. IP アドレスおよび命名サービス

Red Hat Quay でコンポーネントコンテナーを設定し、それらが相互に通信できるようにする方法は多数あります。

- **コンテナーの IP アドレスの使用: podman inspect** でコンテナーの IP アドレスを判別し、接続文字列を指定する際に設定ツールでこれらの値を使用できます。以下に例を示します。

  ```
  $ 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。

- **命名サービスの使用:** 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。2 つのコンテナーが同じポートを使用したい場合、ポートの競合の影響を受けます。したがって、このオプションは推奨されません。
- **ポートマッピングの設定**: ポートマッピングを使用して ホストでポートを公開し、これらのポートをホスト 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、後続の例で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。この例では、**quay-server** の IP アドレスは **192.168.1.112**です。

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

コンポーネント	ポートマッピング	Address
Quay	**-p 8080:8080**	http://quay-server:8080
Postgres for Quay	**-p 5432:5432**	quay-server:5432
Redis	**-p 6379:6379**	quay-server:6379
Postgres for Clair V4	**-p 5433:5432**	quay-server:5433
Clair v4	**-p 8081:8080**	http://quay-server:8081

2.3. データベースの設定

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あり、とくに高可用性の設定については Postgres が推奨されます。または、Postgres について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

この概念実証のシナリオでは、ローカルファイルシステム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。

- ここでは $QUAY 変数で示されるインストールフォルダーで、データベースデータのディレクトリーを作成し、パーミッションを適切に設定します。

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

- Podman を使用して、データベースデータのボリューム定義と共にユーザー名、パスワード、データベース名、ポートを指定して Postgres コンテナーを実行します。

```
$ 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-10:1
```

- Quay で必要な 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'
```

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

ポートとパスワードを指定して Redis コンテナーを実行するには、podman を使用します。

```
$ sudo podman run -d --rm --name redis \
  -p 6379:6379 \
  -e REDIS_PASSWORD=strongpassword \
  registry.redhat.io/rhel8/redis-5:1
```

2.5. RED HAT QUAY の設定

Red Hat Quay サービスを実行する前に、レジストリー設定、データベース、および Redis 接続パラメーターを含むすべてのコンポーネントの詳細を含む設定ファイルを生成する必要があります。設定ファイルを生成するには、**quayconfig** ユーザーのパスワード（この例では **secret**）を指定して **config** モードで quay コンテナーを実行します 。

```
$ sudo podman run --rm -it --name quay_config -p 8080:8080 registry.redhat.io/quay/quay-rhel8:v3.4.6 config secret
```

ブラウザーを使用して **http://quay-server:8080** で設定ツールのユーザーインターフェースにアクセスします（ **quay-server** ホスト名を **hosts** ファイルに設定していることを前提とします）。ユーザー名 **quayconfig** およびパスワード **secret** （または上記の podman run コマンドで指定した値）を使用してログインします。

2.5.1. Red Hat Quay のセットアップ

設定エディターで、以下の情報を入力します。

- 基本設定
- サーバー構成
- データベース

- Redis

2.5.1.1. 基本設定

基本的な設定設定で、レジストリーのタイトルおよびレジストリーの短いタイトルフィールドに入力します（または、指定されている場合はデフォルト値を使用できます）。

2.5.1.2. サーバー構成

レジストリーにアクセスできるネットワーク上の場所について HTTP ホストおよびポートを指定します（この例では、**quay-server:8080**）。

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。本書の手順に従って概念実証用のシステムをデプロイする場合は、以下の値を入力します。

- **Database Type:** Postgres
- **Database Server:** quay-server: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。本書の手順に従って概念実証用のシステムをデプロイする場合は、以下の値を指定します。

- **Redis Hostname:** quay-server
- **Redis port:** 6379 (default)

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、すべての必須フィールドが有効であり、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。

設定が有効になったら、設定ファイルをダウンロードし、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。

2.6. RED HAT QUAY をデプロイする

2.6.1. 前提条件

- Quay データベースおよび Redis サーバーが実行中である。
- 有効な設定バンドルを生成している。

- 設定エディターの実行に使用した 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

設定バンドルを展開し、Quay がこれを使用できるようにします。以下に例を示します。

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

この概念実証用のデプロイメントでは、ローカルファイルシステムを使用してレジストリーイメージを保存します。

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

2.6.4. Red Hat Quay レジストリーのデプロイ

Podman を使用して quay コンテナーを実行し、設定データ用の適切なボリュームとイメージデータのローカルストレージを指定します。

```
$ sudo podman run -d --rm -p 8080:8080  \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -v $QUAY/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.4.6
```

2.7. RED HAT QUAY の使用

ブラウザーを使用して、**quay-server:8080** で Red Hat Quay レジストリーのユーザーインターフェースにアクセスします（ **quay-server** ホスト名を **hosts** ファイルに設定していることを前提とします）。Create Userを選択し、ユーザーを追加します（例: **quayadmin**、パスワードは **password**）。

ユーザーインターフェースを使用して、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るようになりました。または、コマンドラインインターフェー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。

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:8080
Username: quayadmin
Password:
Login Succeeded!
```

2.7.1. イメージのプッシュおよびプル

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
```

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
...
```

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:8080/quayadmin/busybox:test
```

ここで、イメージをRed Hat Quayのレジストリにプッシュします。

```
$ sudo podman push --tls-verify=false quay-server:8080/quayadmin/busybox:test
Getting image source signatures
Copying blob 6b245f040973 done
Copying config 22667f5368 done
Writing manifest to image destination
Storing signatures
```

この時点で、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。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:8080/quayadmin/busybox:test
Untagged: quay-server:8080/quayadmin/busybox:test
```

これで、イメージをもう一度プルします。今回は Red Hat Quay レジストリーから実行されます。

```
$ sudo podman pull --tls-verify=false quay-server:8080/quayadmin/busybox:test
Trying to pull quay-server:8080/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

3.1. 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。

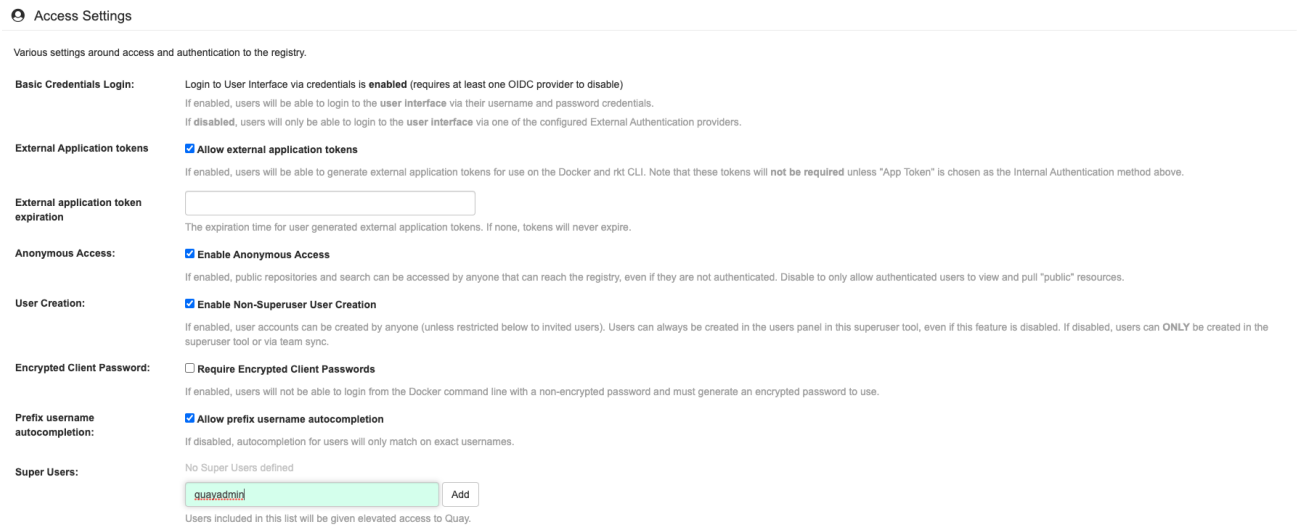
- ユーザーの管理
- 組織の管理
- サービスキーの管理
- 変更ログの閲覧
- 使用状況ログのクエリー
- グローバルに表示されるユーザーメッセージの作成

3.1.1. UI を使用したスーパーユーザーの Quay への追加

Quay レジストリーが実行中である場合はこれを停止し、コンテナーを設定モードで再起動し、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 \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.4.6 config secret
```

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 を押します。



設定バンドルを検証し、ダウンロードしてから、config 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。

3.1.2. 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

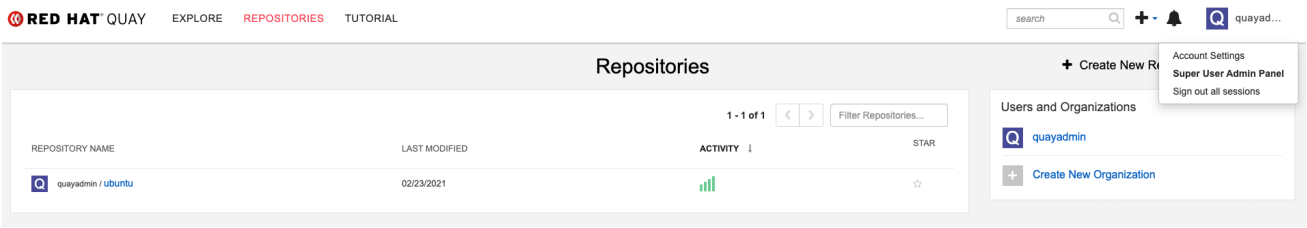
config.yaml ファイルを直接編集してスーパーユーザーを追加することもできます。スーパーユーザーアカウントのリストは、フィールド**SUPER_USERS**に配列として格納されます。

$QUAY/config/config.yaml

```
SERVER_HOSTNAME: quay-server:8080
SETUP_COMPLETE: true
SUPER_USERS:
  - quayadmin
...
```

3.1.3. スーパーユーザー管理パネルへのアクセス

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、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. CLAIR V4 のデプロイ

Clair はイメージのコンテンツを解析し、コンテンツに影響する脆弱性を報告するアプリケーションです。これは静的分析によって実行され、ランタイム時には実行されません。Clair の分析は 3 つの部分に分かれています。

- **Indexing:** インデックス作成はマニフェストの Clair への送信から開始します。受信時に、Clair はレイヤーを取得し、その内容をスキャンし、IndexReport という中間表現を返します。マニフェストは、Clair のコンテナーイメージの表示です。Clair は OCI マニフェストとレイヤーがコンテンツ対応であることを活かして、重複作業を削減します。マニフェストがインデックス化されると、IndexReport は後で取得できるように永続化されます。

- **Matching:** マッチングは IndexReport を取り、レポートが表すマニフェストに影響を与える脆弱性を相関させます。Clair は継続的に新しいセキュリティーデータを取り込み、matcher への要求は IndexReport の最新の脆弱性分析を提供します。

- **Notifications:** Clair は通知サービスを実装します。新しい脆弱性が発見されると、通知サービスは、これらの脆弱性がインデックスされたマニフェストに影響するかどうかを判断します。その後、通知機能は設定に応じてアクションを実行します。

3.2.1. Clair 用の個別データベースのデプロイ

Clair には Postgres データベースが必要です。Quay が Postgres を使用している場合に Quay と Clair 間で共通データベースを共有できますが、この例では個別の Clair 固有のデータベースがデプロイされます。

この概念実証のシナリオでは、ローカルファイルシステム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。

- ここでは $QUAY 変数で示されるインストールフォルダーで、Clair データベースデータのディレクトリーを作成し、パーミッションを適切に設定します。

```
$ mkdir -p $QUAY/postgres-clairv4
$ setfacl -m u:26:-wx $QUAY/postgres-clairv4
```

- Podman を使用し、データベースデータのボリューム定義と共にユーザー名、パスワード、データベース名、ポートを指定して Postgres コンテナーを実行します。標準のPostgresポートである**5432**はQuayのデプロイメントですでに使用されているので、別のポート（この例では**5433**）を公開します。

```
$ sudo podman run -d --rm --name postgresql-clairv4 \
  -e POSTGRESQL_USER=clairuser \
  -e POSTGRESQL_PASSWORD=clairpass \
  -e POSTGRESQL_DATABASE=clair \
  -e POSTGRESQL_ADMIN_PASSWORD=adminpass \
  -p 5433:5432 \
  -v $QUAY/postgres-clairv4:/var/lib/pgsql/data:Z \
  registry.redhat.io/rhel8/postgresql-10:1
```

- Clair で必要な Postgres **uuid-ossp** 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-clairv4 /bin/bash -c 'echo "CREATE EXTENSION IF NOT EXISTS \"uuid-ossp\"" | psql -d clair -U postgres'
```

3.2.2. Clair の Quay 設定

Quay コンテナーが実行中である場合はこれを停止し、これを設定モードで再起動し、既存の設定をボリュームとして読み込みます。

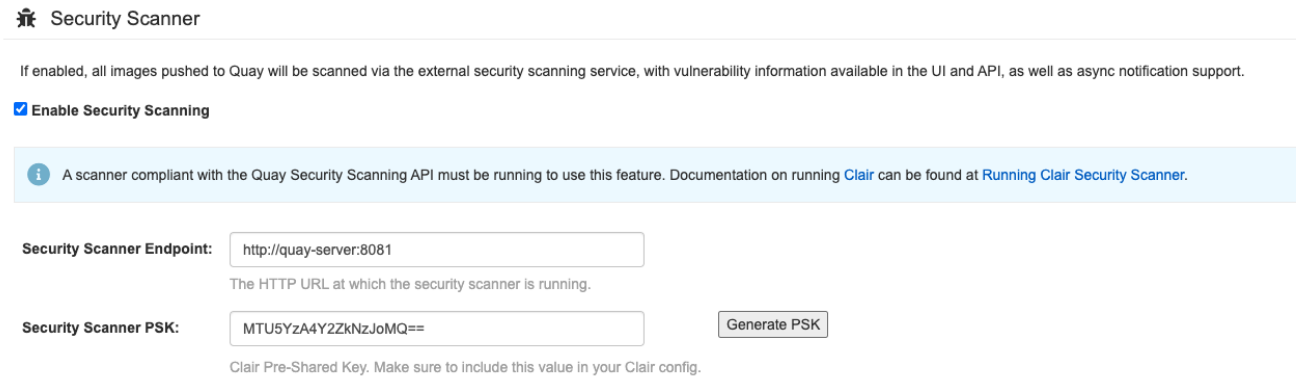
```
$ sudo podman run --rm -it --name quay_config \
  -p 8080:8080 \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.4.6 config secret
```

設定ツールにログインし、UI の Security Scanner セクションでスキャンを有効にします。**quay-server** システムでまだ使用されていないポート（例: **8081**）を使用して、Clair の HTTP エンドポイントを設定します。**Generate PSK** ボタンを使用して Clair の PSK（事前共有キー）を作成します。以下に例を示します。

- Security Scanner Endpoint: **http://quay-server:8081**
- Security Scanner PSK: **MTU5YzA4Y2ZkNzJoMQ==**

スキャナーデータを設定するための UI が以下のイメージに表示されます。

セキュリティースキャナー UI



設定を検証し、ダウンロードしてから、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。設定バンドルを以前と同様に **$QUAY/config** ディレクトリーに展開します。

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

Quay 設定ファイルが更新され、セキュリティースキャナーのフィールドが含まれるようになりました。

$QUAY/config/config.yaml

```
...
FEATURE_SECURITY_NOTIFICATIONS: false
FEATURE_SECURITY_SCANNER: true
...
SECURITY_SCANNER_INDEXING_INTERVAL: 30
SECURITY_SCANNER_V4_ENDPOINT: http://quay-server:8081
SECURITY_SCANNER_V4_PSK: MTU5YzA4Y2ZkNzJoMQ==
SERVER_HOSTNAME: quay-server:8080
...
```

3.2.3. Clair の設定

Clair 設定についての詳細は https://github.com/quay/clair/blob/main/Documentation/reference/config.md を参照してください。以下の例は、概念実証用のデプロイメントで使用する最小の設定を示しています。

/etc/clairv4/config/config.yaml

```
http_listen_addr: :8081
introspection_addr: :8089
log_level: debug
indexer:
  connstring: host=quay-server port=5433 dbname=clair user=clairuser password=clairpass
sslmode=disable
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
matcher:
  connstring: host=quay-server 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 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"
```

- **http_listen_addr** は、Quay 設定ツールで指定した Clair HTTP エンドポイントのポート（この場合は**:8081**）に設定されます 。
- 認証にはQuayの設定ツールで生成したClairの事前共有鍵（PSK）が使用され、**iss**フィールドで指定した発行者は**quay**に設定されます。

3.2.4. Clair の実行

podman runコマンドを使用してClairコンテナを実行し、設定ツールで指定したHTTPエンドポイントポート（ここでは**8081**）を公開します。

```
sudo podman run -d --rm --name clairv4 \
  -p 8081:8081 -p 8089:8089 \
  -e CLAIR_CONF=/clair/config.yaml -e CLAIR_MODE=combo \
  -v /etc/clairv4/config:/clair:Z \
  registry.redhat.io/quay/clair-rhel8:v3.4.6
```

次に、スキャナー設定が含まれる更新された設定ファイルを使用して Quay コンテナーを再起動します。

```
$ sudo podman run -d --rm -p 8080:8080  \
   --name=quay \
   -v $QUAY/config:/conf/stack:Z \
   -v $QUAY/storage:/datastorage:Z \
   registry.redhat.io/quay/quay-rhel8:v3.4.6
```

3.2.5. Clair セキュリティースキャンの使用

コマンドラインで、レジストリーにログインします。

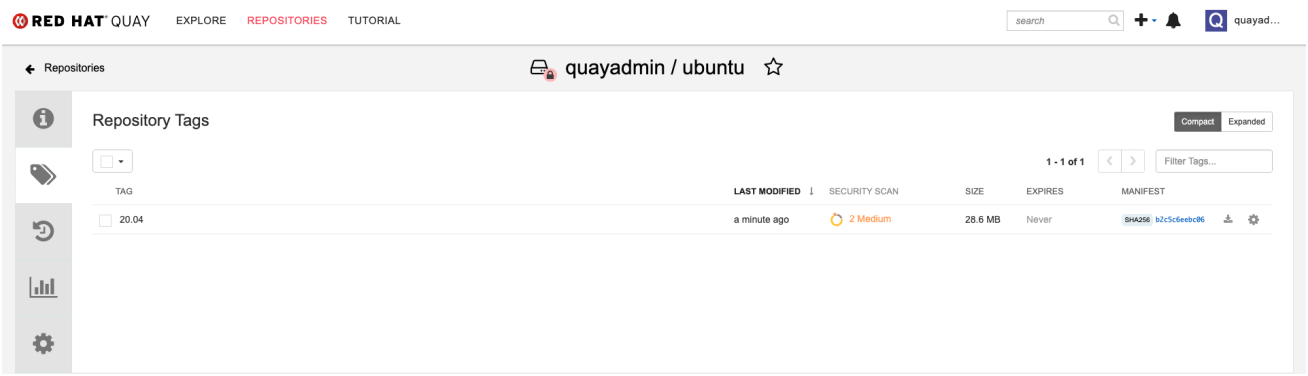
```
$ sudo podman login --tls-verify=false quay-server:8080
Username: quayadmin
Password:
Login Succeeded!
```

サンプルイメージをプルし、これにタグを付け、レジストリーにプッシュします。

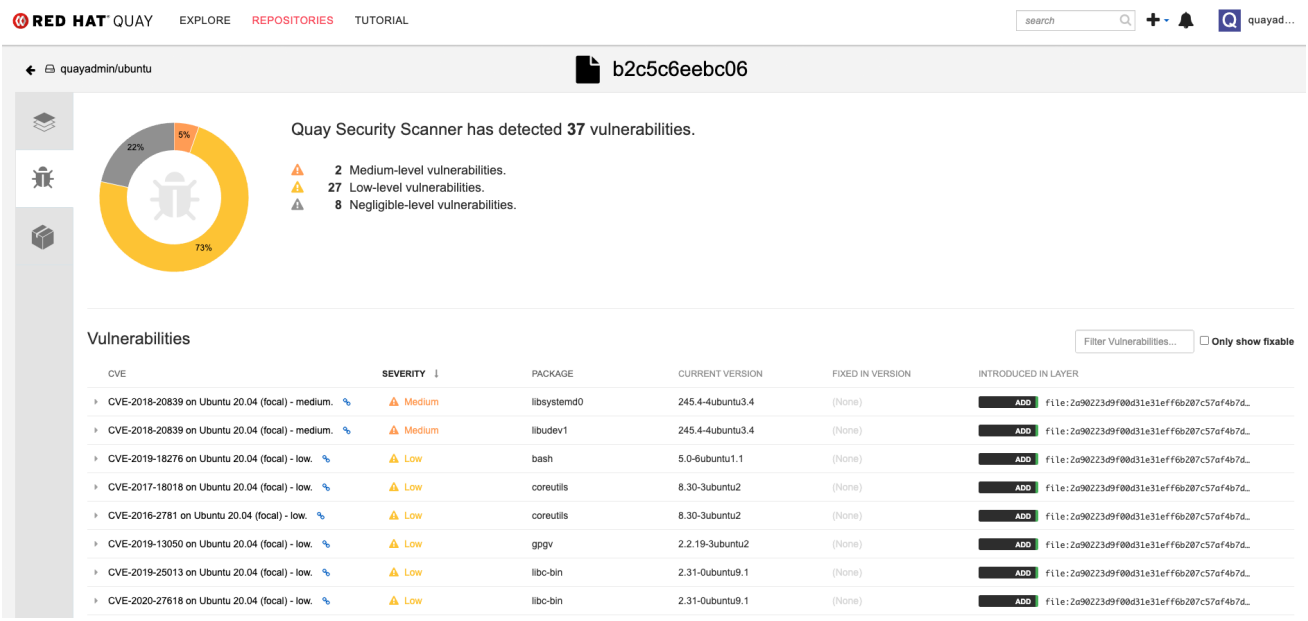
```
$ sudo podman pull ubuntu:20.04
$ sudo podman tag docker.io/library/ubuntu:20.04 quay-server:8080/quayadmin/ubuntu:20.04
$ sudo podman push --tls-verify=false quay-server:8080/quayadmin/ubuntu:20.04
```

以下のイメージが示すように、セキュリティースキャンの結果が Quay UI に表示されます。

スキャンの要約



スキャンの詳細



3.3. 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、Porting containers to systemd using Podman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。

3.3.1. 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。

- 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
```

- **/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
```

- 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
```

- 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
```

3.3.2. サービスの起動、停止、およびステータスのチェック

- 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
```

- 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
```

- 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
```

3.3.3. 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  registry.redhat.io/rhel8/postgresql-10:1    run-postgresql  19 seconds ago  Up 18
seconds ago  0.0.0.0:5432->5432/tcp  postgresql-quay
b8fbac1920d4  registry.redhat.io/rhel8/redis-5:1          run-redis       19 seconds ago  Up 18 seconds
ago  0.0.0.0:6379->6379/tcp  redis
d959d5bf7a24  registry.redhat.io/rhel8/postgresql-10:1    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。

3.3.4. 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  registry.redhat.io/rhel8/postgresql-10:1    run-postgresql  29 seconds ago  Up 28
seconds ago  0.0.0.0:5432->5432/tcp  postgresql-quay
b8fbac1920d4  registry.redhat.io/rhel8/redis-5:1          run-redis       29 seconds ago  Up 28 seconds
ago  0.0.0.0:6379->6379/tcp  redis
d959d5bf7a24  registry.redhat.io/rhel8/postgresql-10:1    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  registry.redhat.io/rhel8/postgresql-10:1    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-5:1          run-redis       35 seconds ago  Up 34 seconds
ago  0.0.0.0:6379->6379/tcp  redis
d959d5bf7a24  registry.redhat.io/rhel8/postgresql-10:1    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:8080 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。

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、『Deploy Red Hat Quay - High Availability』を参照してください。

『Using Red Hat Quay』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
- Use tags
- 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

『Manage Red Hat Quay』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