Red Hat Quay 3.11

Red Hat Quay Operator の機能

Red Hat Quay Operator の高度な機能

Last Updated: 2024-04-05

Red Hat Quay 3.11 Red Hat Quay Operator の機能

Red Hat Quay Operator の高度な機能

法律上の通知

概要

Red Hat Quay Operator の高度な機能

目次

第1章 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 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。

1.1. FIPS コンプライアンスの有効化

以下の手順を使用して、Red Hat Quay デプロイメントで FIPS コンプライアンスを有効にします。

前提条件

- Red Hat Quay のスタンドアロンデプロイメントを実行している場合、Red Hat Enterprise Linux (RHEL) デプロイメントがバージョン 8 以降であり、FIPS が有効である。
- Red Hat Quay を OpenShift Container Platform にデプロイしている場合、OpenShift Container Platform がバージョン 4.10 以降である。
- Red Hat Quay のバージョンが 3.5.0 以降である。
- IBM Power または IBM Z クラスター上の OpenShift Container Platform で Red Hat Quay を使用している場合:
  - OpenShift Container Platform バージョン 4.14 以降
  - Red Hat Quay バージョン 3.10 以降
- Red Hat Quay デプロイメントの管理者権限がある。

手順

- Red Hat Quay の **config.yaml** ファイルで、**FEATURE_FIPS** 設定フィールドを **true** に設定します。以下に例を示します。

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

FEATURE_FIPS を **true** に設定すると、Red Hat Quay は FIPS 準拠のハッシュ関数を使用して実行されます。

第2章 コンソールでのモニタリングおよびアラート

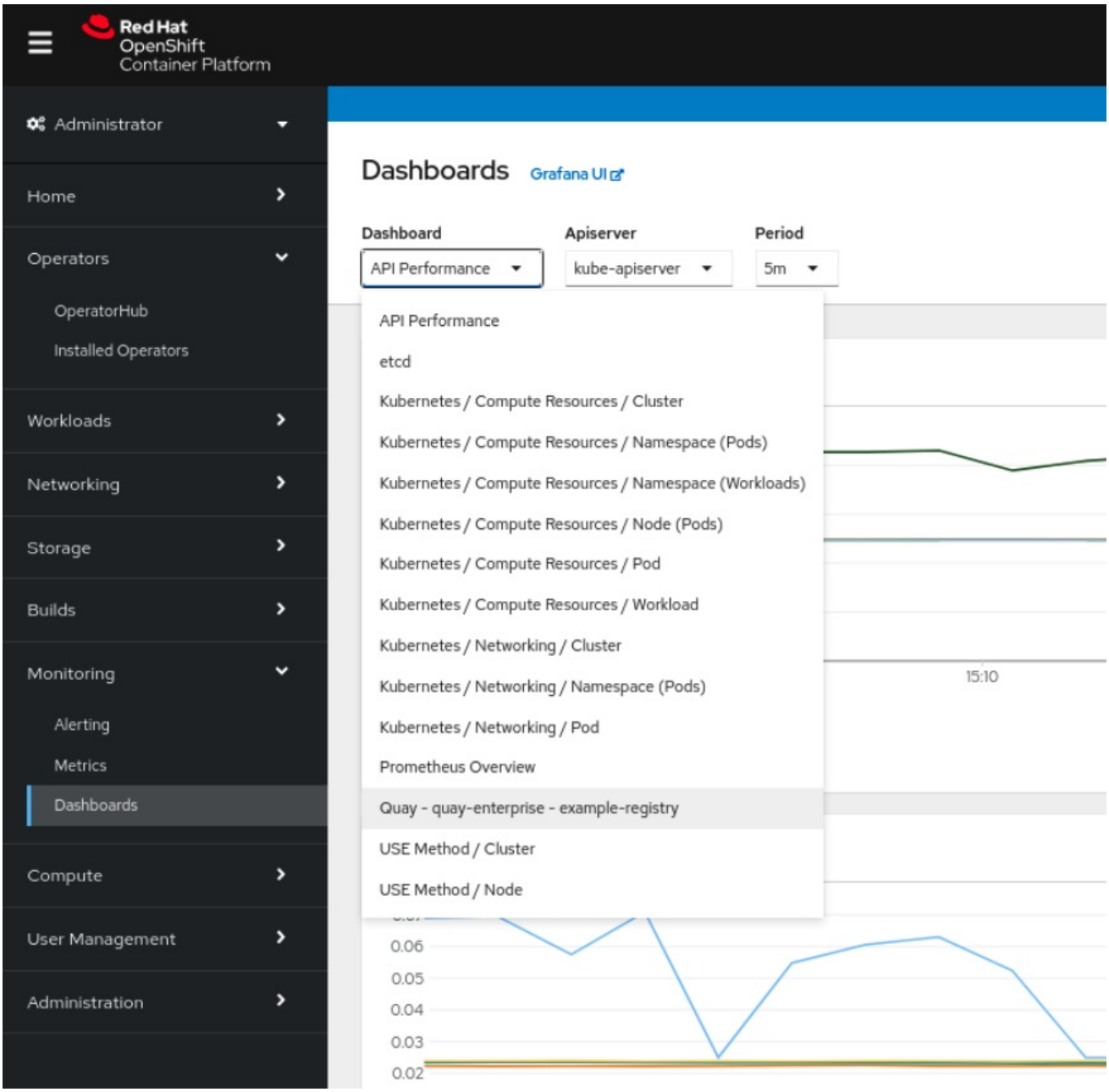
Red Hat Quay では、OpenShift Container Platform コンソール内から Red Hat Quay Operator を使用してデプロイされたインスタンスのモニタリングがサポートされています。新規のモニタリング機能には、Grafana ダッシュボード、個別のメトリクスへのアクセス、**Quay** Pod を頻繁に再起動するために通知するアラートなどが含まれます。

注記

モニタリング機能を有効にするには、Red Hat Quay Operator が **All Namespaces** モードでインストールされている必要があります。

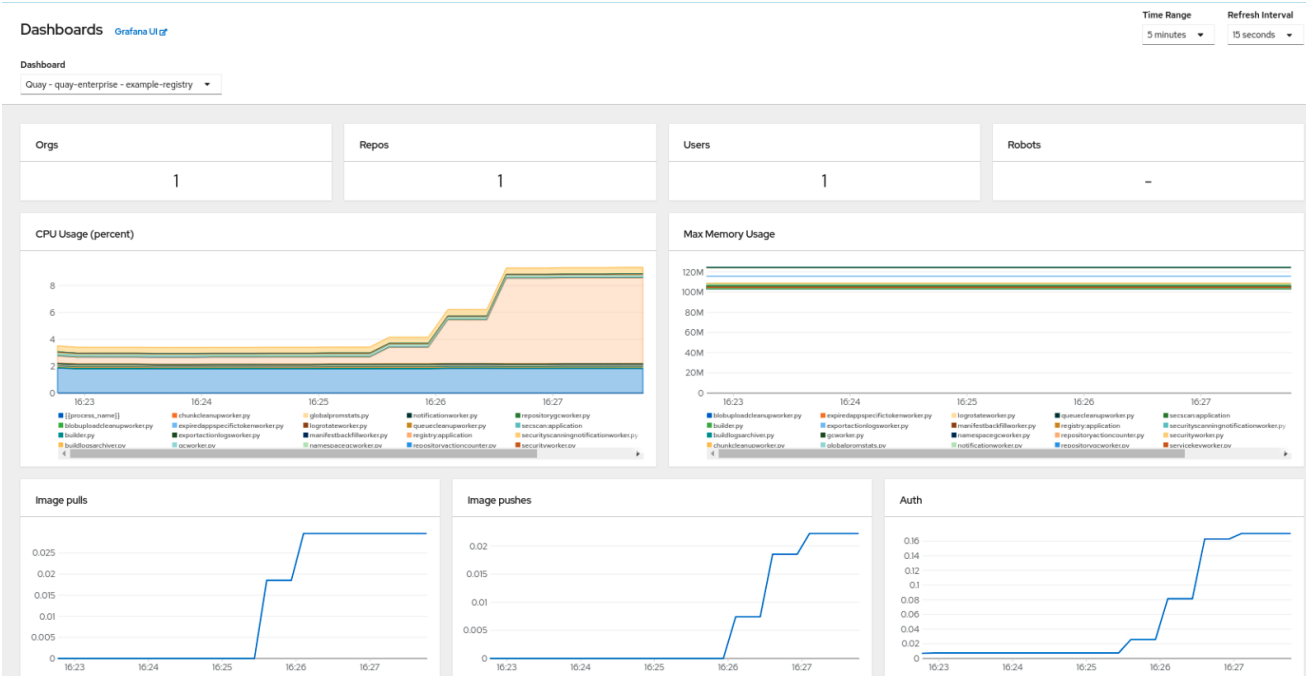
2.1. ダッシュボード

OpenShift Container Platform コンソールで、**Monitoring → Dashboards** をクリックし、目的の Red Hat Quay レジストリーインスタンスのダッシュボードを検索します。



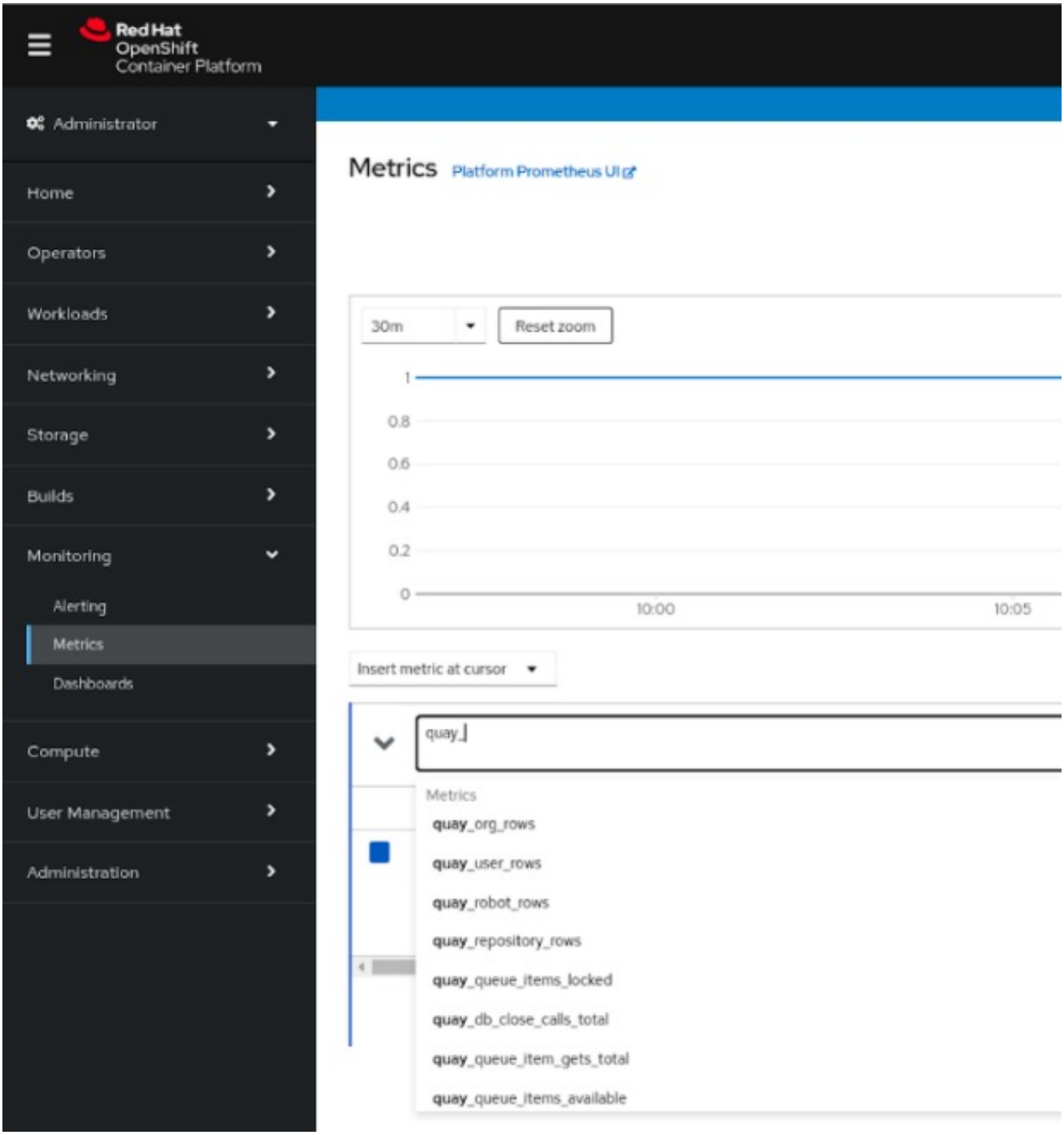
ダッシュボードには、次を含む統計情報が表示されます。

- **Organizations、Repositories、Users、Robot accounts** の数
- CPU の使用率
- 最大メモリー使用量
- プル、プッシュ、認証要求のレート
- API 要求のレート
- 待機時間

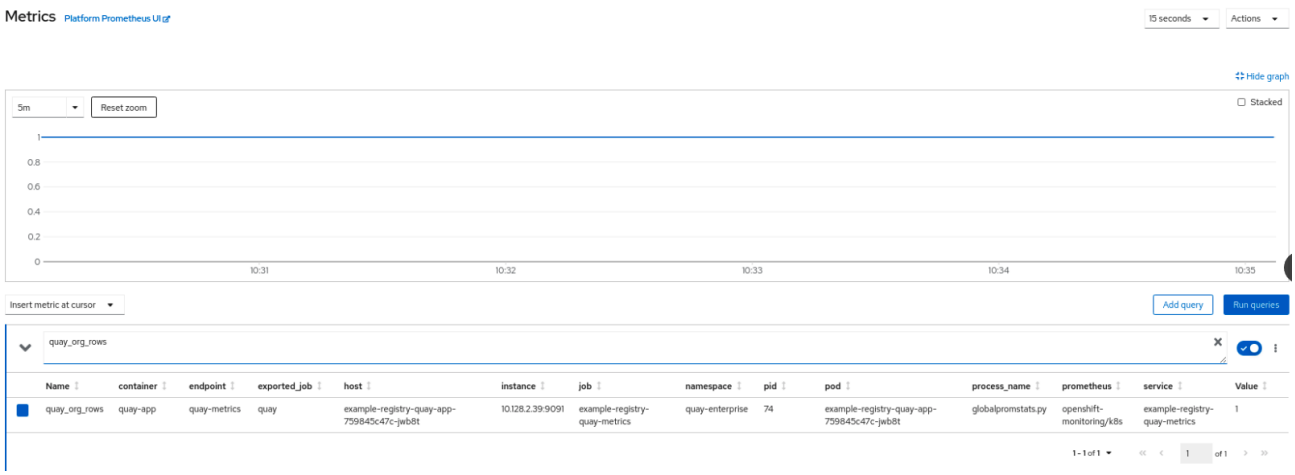


2.2. メトリクス

UI で **Monitoring → Metrics** にアクセスすると、Red Hat Quay ダッシュボードの背後にある基礎となるメトリクスを表示できます。**Expression** フィールドに **quay_** を入力し、利用可能なメトリクスのリストを表示します。



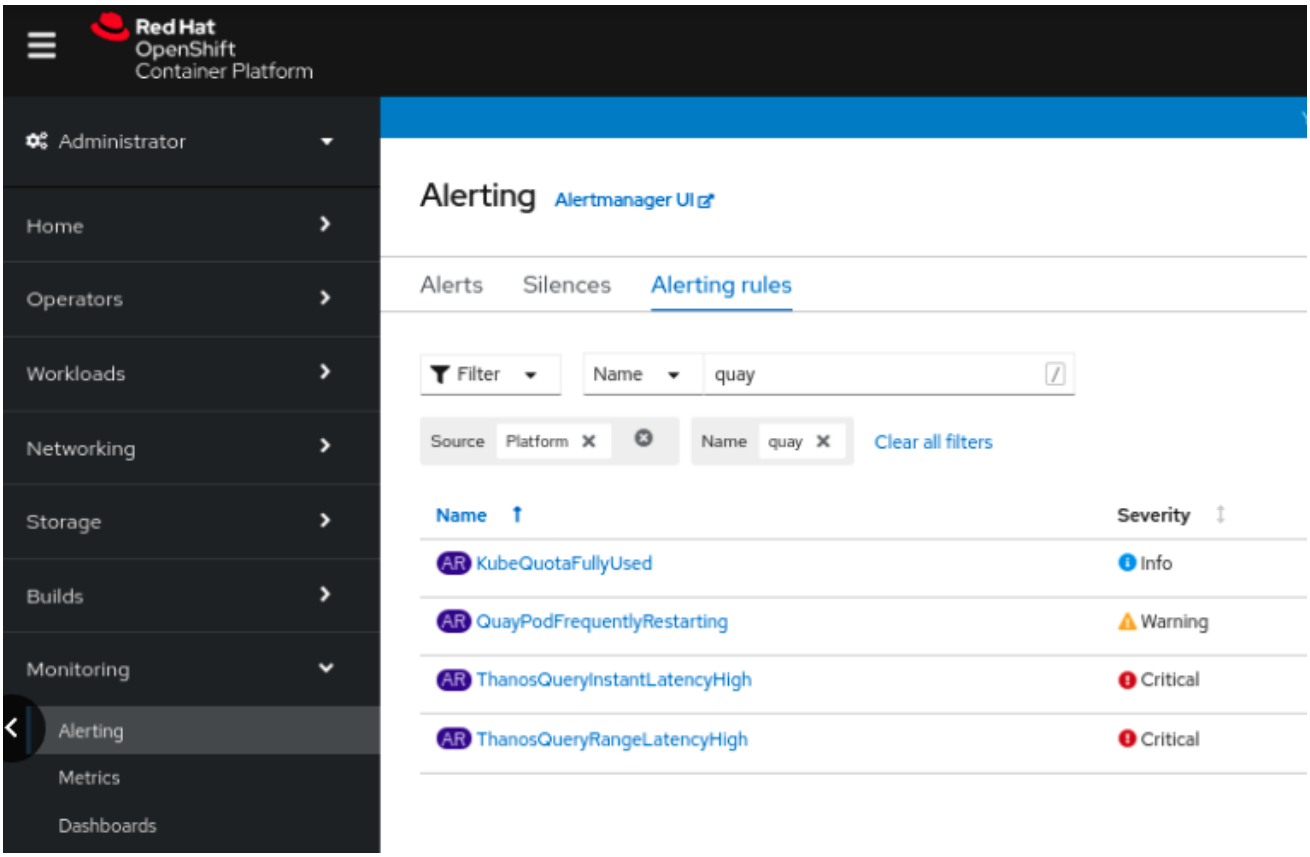
quay_org_rows など、サンプルメトリックを選択します。



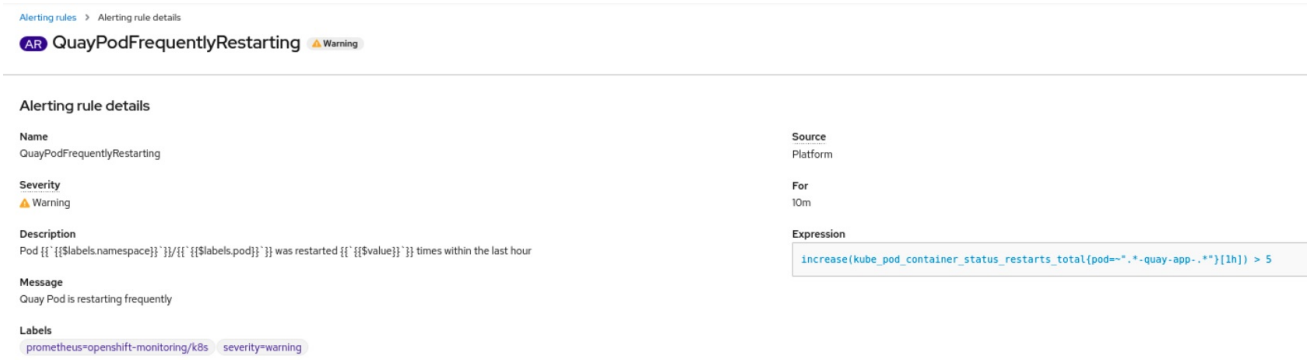
このメトリクスは、レジストリー内の組織の数を示します。ダッシュボードにも直接表示されます。

2.3. アラート

Quay Pod が頻繁に再起動する場合はアラートが発生します。アラートを設定するには、コンソール UI の **Monitoring** → **Alerting** で **Alerting** ルールタブにアクセスし、Quay 固有のアラートを検索します。



QuayPodFrequentlyRestarting ルールの詳細を選択し、アラートを設定します。



第3章 CLAIR セキュリティースキャナー

3.1. CLAIR 脆弱性データベース

Clair は、次の脆弱性データベースを使用して、イメージの問題を報告します。

- Ubuntu Oval データベース
- Debian Security Tracker
- Red Hat Enterprise Linux (RHEL) Oval データベース
- SUSE Oval データベース
- Oracle Oval データベース
- アルパイン SecDB データベース
- VMware Photon OS データベース
- Amazon Web Services (AWS) UpdateInfo
- Open Source Vulnerability (OSV) Database

Clair がさまざまなデータベースでセキュリティーマッピングを行う方法は、Claircore Severity Mapping を参照してください。

3.1.1. Clair の Open Source Vulnerability (OSV) データベースに関する情報

Open Source Vulnerability (OSV) は、オープンソースソフトウェアのセキュリティー脆弱性の追跡と管理に重点を置いた脆弱性データベースおよび監視サービスです。

OSV は、オープンソースプロジェクトにおける既知のセキュリティー脆弱性の包括的かつ最新のデータベースを提供します。ソフトウェア開発で使用されるライブラリー、フレームワーク、その他のコンポーネントを含む、幅広いオープンソースソフトウェアを対象としています。対象エコシステムの完全なリストについては、定義されているエコシステム を参照してください。

Clair は、Open Source Vulnerability (OSV) データベースを通じて、**golang**、**java**、および **ruby** エコシステムの脆弱性とセキュリティー情報も報告します。

開発者や組織は、OSV を活用することで、使用するオープンソースコンポーネントのセキュリティー脆弱性をプロアクティブに監視して対処できるため、プロジェクトにおけるセキュリティー違反やデータ漏洩のリスクを軽減できます。

OSV の詳細は、OSV Web サイト を参照してください。

3.2. OPENSHIFT CONTAINER PLATFORM の CLAIR

OpenShift Container Platform 上の Red Hat Quay デプロイメントで Clair v4 (Clair) をセットアップするには、Red Hat Quay Operator を使用することが推奨されます。デフォルトでは、Red Hat Quay Operator は、Clair デプロイメントを Red Hat Quay デプロイメントとともにインストールまたはアップグレードし、Clair を自動的に設定します。

3.3. 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ている。

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。

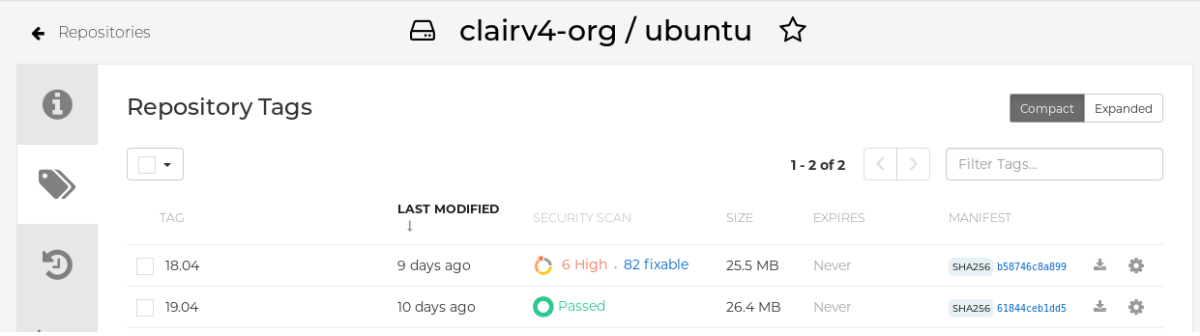
```
$ sudo podman tag docker.io/library/ubuntu:20.04 <quay-server.example.com>/<user-name>/ubuntu:20.04
```

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
```

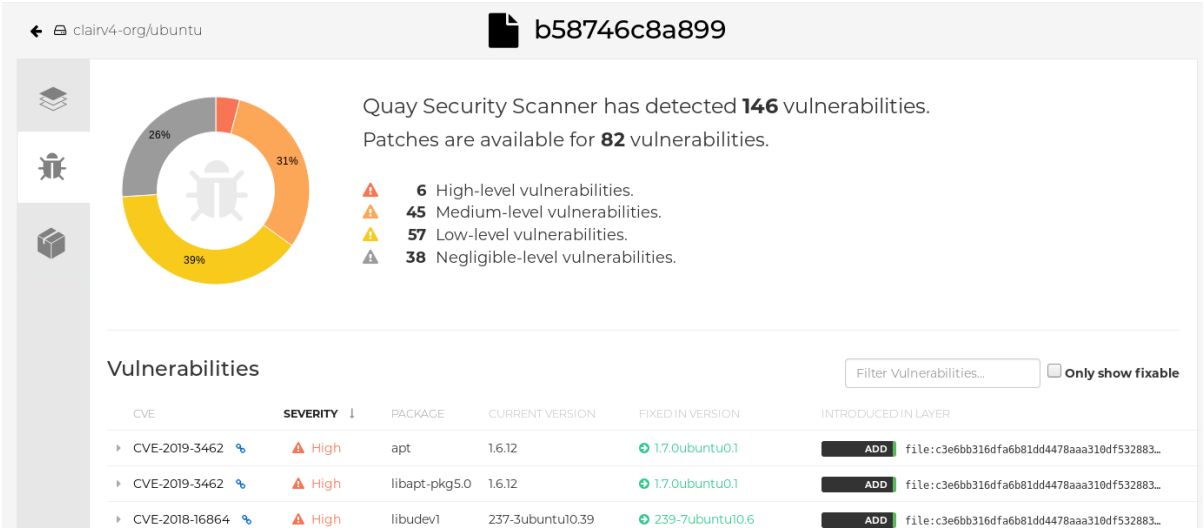
4. UI から Red Hat Quay デプロイメントにログインします。
5. リポジトリー名 (**quayadmin/ubuntu** など) をクリックします。
6. ナビゲーションウィンドウで、**Tags** をクリックします。

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



注記

場合によっては、Clair はイメージに関する重複レポートを表示します (例: **ubi8/nodejs-12** または **ubi8/nodejs-16**)。これは、同じ名前の脆弱性が異なるパッケージに存在するために発生します。この動作は Clair 脆弱性レポートで予期されており、バグとしては扱われません。

3.4. 高度な CLAIR 設定

次のセクションの手順を使用して、Clair の詳細設定を行います。

3.4.1. アンマネージド Clair 設定

Red Hat Quay ユーザーは、Red Hat Quay OpenShift Container Platform Operator を使用してアンマネージド Clair 設定を実行できます。この機能により、ユーザーはアンマネージド Clair データベースを作成したり、アンマネージドデータベースなしでカスタム Clair 設定を実行したりできます。

アンマネージド Clair データベースにより、Red Hat Quay オペレーターは、Operator の複数のインスタンスが同じデータベースと通信する必要がある地理的に複製された環境で作業できます。アンマネージド Clair データベースは、ユーザーがクラスターの外部に存在する高可用性 (HA) Clair データベースを必要とする場合にも使用できます。

3.4.1.1. アンマネージド Clair データベースを使用したカスタム Clair 設定の実行

次の手順を使用して、Clair データベースをアンマネージドに設定します。

手順

- Quay Operator で、**QuayRegistry** カスタムリソースの **clairpostgres** コンポーネントを **managed: false** に設定します。

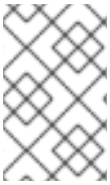
```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: quay370
spec:
  configBundleSecret: config-bundle-secret
  components:
```

```
- kind: objectstorage
  managed: false
- kind: route
  managed: true
- kind: tls
  managed: false
- kind: clairpostgres
  managed: false
```

3.4.1.2. アンマネージド Clair データベースを使用したカスタム Clair データベースの設定

OpenShift Container Platform 上の Red Hat Quay では、ユーザーが独自の Clair データベースを指定できます。

次の手順を使用して、カスタム Clair データベースを作成します。



注記

次の手順では、SSL/TLS 証明書を使用して Clair をセットアップします。SSL/TSL 証明書を使用して Clair をセットアップしない同様の手順を表示するには、マネージド Clair 設定を使用したカスタム Clair データベースの設定を参照してください。

手順

1. 次のコマンドを入力して、**clair-config.yaml** を含む Quay 設定バンドルシークレットを作成します。

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

Clair config.yaml ファイルの例

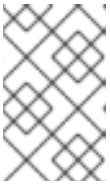
```
indexer:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslrootcert=/run/certs/rds-ca-2019-root.pem sslmode=verify-ca
    layer_scan_concurrency: 6
    migrations: true
    scanlock_retry: 11
log_level: debug
matcher:
    connstring: host=quay-server.example.com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

注記

- データベース証明書は、**clair-config.yaml** の Clair アプリケーション Pod の **/run/certs/rds-ca-2019-root.pem** の下にマウントされます。**clair-config.yaml** を設定するときに指定する必要があります。
- **clair-config.yaml** の例は、OpenShift 設定の Clair にあります。

2. **clair-config.yaml** ファイルをバンドルシークレットに追加します。次に例を示します。

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



注記

更新すると、提供された **clair-config.yaml** ファイルが Clair Pod にマウントされます。提供されていないフィールドには、Clair 設定モジュールを使用してデフォルトが自動的に入力されます。

3. **Build History** ページでコミットをクリックするか、**oc get pods -n <namespace>** を実行して、Clair Pod のステータスを確認できます。以下に例を示します。

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

出力例

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

3.4.2. マネージド Clair データベースを使用したカスタム Clair 設定の実行

場合によっては、マネージド Clair データベースを使用してカスタム Clair 設定を実行します。これは、以下のシナリオで役に立ちます。

- ユーザーが特定のアップデーターリソースを無効にする場合。
- ユーザーが非接続環境で Red Hat Quay を実行している場合。非接続環境での Clair の実行の詳細は、非接続環境での Clair を参照してください。

注記

- 非接続環境で Red Hat Quay を実行している場合は、**clair-config.yaml** の **airgap** パラメーターを **true** に設定する必要があります。
- 非接続環境で Red Hat Quay を実行している場合は、すべてのアップデーターコンポーネントを無効にする必要があります。

3.4.2.1. Clair データベースをマネージドに設定する

次の手順を使用して、Clair データベースをマネージドに設定します。

手順

- Quay Operator で、**QuayRegistry** カスタムリソースの **clairpostgres** コンポーネントを **managed: true** に設定します。

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

3.4.2.2. マネージド Clair 設定を使用したカスタム Clair データベースの設定

OpenShift Container Platform 上の Red Hat Quay では、ユーザーが独自の Clair データベースを指定できます。

次の手順を使用して、カスタム Clair データベースを作成します。

手順

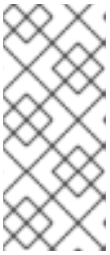
1. 次のコマンドを入力して、**clair-config.yaml** を含む Quay 設定バンドルシークレットを作成します。

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

Clair config.yaml ファイルの例

```
indexer:
    connstring: host=quay-server.example.com port=5432 dbname=quay user=quayrdsdb password=quayrdsdb sslmode=disable
```

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



注記

- データベース証明書は、**clair-config.yaml** の Clair アプリケーション Pod の **/run/certs/rds-ca-2019-root.pem** の下にマウントされます。**clair-config.yaml** を設定するときに指定する必要があります。
- **clair-config.yaml** の例は、OpenShift 設定の Clair にあります。

2. **clair-config.yaml** ファイルをバンドルシークレットに追加します。次に例を示します。

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

注記

- 更新すると、提供された **clair-config.yaml** ファイルが Clair Pod にマウントされます。提供されていないフィールドには、Clair 設定モジュールを使用してデフォルトが自動的に入力されます。

3. **Build History** ページでコミットをクリックするか、**oc get pods -n <namespace>** を実行して、Clair Pod のステータスを確認できます。以下に例を示します。

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

出力例

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

3.4.3. 非接続環境での Clair

注記

現在、非接続環境での Clair のデプロイは、IBM Power および IBM Z ではサポートされていません。

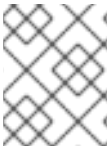
Clair は、**updater** と呼ばれる一連のコンポーネントを使用して、さまざまな脆弱性データベースからのデータのフェッチと解析を処理します。updater はデフォルトで、脆弱性データをインターネットから直接プルし、すぐに使用できるように設定されています。ただし、ユーザーによっては、Red Hat Quay を非接続環境、またはインターネットに直接アクセスできない環境で実行する必要がある場合があります。Clair は、ネットワーク分離を考慮したさまざまな種類の更新ワークフローを使用することで、非接続環境をサポートします。これは **clairctl** コマンドラインインターフェイスツールを使用して機能します。このツールは、オープンホストを使用してインターネットから updater データを取得し、そのデータを隔離されたホストにセキュアに転送してから、隔離されたホスト上の updater データを Clair にインポートします。

非接続環境で Clair をデプロイするには、このガイドを使用してください。

重要

既知の問題 PROJQUAY-6577 により、Red Hat Quay Operator はカスタマイズされた Clair **config.yaml** ファイルを適切に処理しません。そのため、現在、次の手順は機能しません。

Operator を利用してフィールドに入力するのではなく、ユーザー自身が Clair の設定全体を最初から作成する必要があります。これを行うには、Procedure to enable Clair scanning of images in disconnected environments の手順に従ってください。



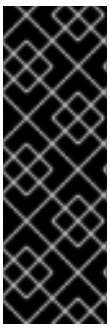
注記

現在、Clair エンリッチメントデータは CVSS データです。エンリッチメントデータは現在、オフライン環境ではサポートされていません。

Clair アップデーターの詳細は、Clair アップデーターを参照してください。

3.4.3.1. 非接続の OpenShift Container Platform クラスターで Clair をセットアップする

以下の手順を使用して、非接続の OpenShift Container Platform クラスターに OpenShift Container Platform でプロビジョニングされた Clair Pod をセットアップします。



重要

既知の問題 PROJQUAY-6577 により、Red Hat Quay Operator はカスタマイズされた Clair **config.yaml** ファイルを適切に処理しません。そのため、現在、次の手順は機能しません。

Operator を利用してフィールドに入力するのではなく、ユーザー自身が Clair の設定全体を最初から作成する必要があります。これを行うには、Procedure to enable Clair scanning of images in disconnected environments の手順に従ってください。

3.4.3.1.1. OpenShift Container Platform デプロイメント用の clairctl コマンドラインユーティリティーツールのインストール

以下の手順を使用して、OpenShift Container Platform デプロイメント用の **clairctl** CLI ツールをインストールします。

手順

1. 以下のコマンドを入力して、Clair デプロイメント用の **clairctl** プログラムを OpenShift Container Platform クラスターにインストールします。

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

注記

非公式ですが、**clairctl** ツールをダウンロードできます。

2. **clairctl** ファイルの権限を設定して、ユーザーが実行できるようにします。次に例を示します。

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

3.4.3.1.2. OpenShift Container Platform での Clair デプロイメントの Clair 設定シークレットの取得とデコード

以下の手順を使用して、OpenShift Container Platform 上の OpenShift Container Platform でプロビジョニングされた Clair インスタンスの設定シークレットを取得してデコードします。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。

手順

1. 次のコマンドを入力して、設定シークレットを取得してデコードし、それを Clair 設定 YAML に保存します。

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

2. **disable_updaters** および **airgap** パラメーターが **true** に設定されるように、**clair-config.yaml** ファイルを更新します。次に例を示します。

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

3.4.3.1.3. 接続された Clair インスタンスからアップデータバンドルをエクスポートする

次の手順を使用して、インターネットにアクセスできる Clair インスタンスから更新プログラムバンドルをエクスポートします。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。

- Clair 設定シークレットを取得してデコードし、Clair の **config.yaml** ファイルに保存している。
- Clair の **config.yaml** ファイルで、**disable_updaters** および **airgap** パラメーターが **true** に設定されている。

手順

- インターネットにアクセスできる Clair インスタンスから、設定ファイルで **clairctl** CLI ツールを使用して、アップデーターバンドルをエクスポートします。以下に例を示します。

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

3.4.3.1.4. 非接続の OpenShift Container Platform クラスター内の Clair データベースへのアクセスの設定

以下の手順を使用して、非接続の OpenShift Container Platform クラスター内の Clair データベースへのアクセスを設定します。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。
- Clair 設定シークレットを取得してデコードし、Clair の **config.yaml** ファイルに保存している。
- Clair の **config.yaml** ファイルで、**disable_updaters** および **airgap** パラメーターが **true** に設定されている。
- インターネットにアクセスできる Clair インスタンスからアップデーターバンドルをエクスポートしている。

手順

1. CLI ツール **oc** を使用して、Clair データベースサービスを特定します。次に例を示します。

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

出力例

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

2. Clair データベースポートを転送して、ローカルマシンからアクセスできるようにします。以下に例を示します。

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

3. Clair の **config.yaml** ファイルを更新します。次に例を示します。

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

(1) 複数の **connstring** フィールドの **host** の値を **localhost** に置き換えます。

(2) **rhel-repository-scanner** パラメーターの詳細は、「Common Product Enumeration 情報へのリポジトリーのマッピング」を参照してください。

(3) **rhel_containersscanner** パラメーターの詳細は、「Common Product Enumeration へのリポジトリーのマッピング」を参照してください。

3.4.3.1.5. 非接続の OpenShift Container Platform クラスターへのアップデーターバンドルのインポート

以下の手順を使用して、アップデーターバンドルを非接続の OpenShift Container Platform クラスターにインポートします。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。
- Clair 設定シークレットを取得してデコードし、Clair の **config.yaml** ファイルに保存している。
- Clair の **config.yaml** ファイルで、**disable_updaters** および **airgap** パラメーターが **true** に設定されている。
- インターネットにアクセスできる Clair インスタンスからアップデーターバンドルをエクスポートしている。
- アップデーターバンドルを非接続環境に転送している。

手順

- CLI ツール **clairctl** を使用して、アップデーターバンドルを OpenShift Container Platform によってデプロイされた Clair データベースにインポートします。以下に例を示します。

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

3.4.3.2. 非接続の OpenShift Container Platform クラスター用の Clair の自己管理デプロイメントをセットアップする

以下の手順を使用して、非接続の OpenShift Container Platform クラスター用の Clair の自己管理デプロイメントをセットアップします。

重要

既知の問題 PROJQUAY-6577 により、Red Hat Quay Operator はカスタマイズされた Clair **config.yaml** ファイルを適切に処理しません。そのため、現在、次の手順は機能しません。

Operator を利用してフィールドに入力するのではなく、ユーザー自身が Clair の設定全体を最初から作成する必要があります。これを行うには、Procedure to enable Clair scanning of images in disconnected environments の手順に従ってください。

3.4.3.2.1. OpenShift Container Platform で自己管理 Clair デプロイメント用の clairctl コマンドラインユーティリティーツールをインストールする

以下の手順を使用して、OpenShift Container Platform に自己管理 Clair デプロイメント用の **clairctl** CLI ツールをインストールします。

手順

1. **podman cp** コマンドを使用して、自己管理の Clair デプロイメント用の **clairctl** プログラムをインストールします。以下に例を示します。

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

2. **clairctl** ファイルの権限を設定して、ユーザーが実行できるようにします。次に例を示します。

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

3.4.3.2.2. 非接続の OpenShift Container Platform クラスター用の自己管理 Clair コンテナーをデプロイする

以下の手順を使用して、非接続の OpenShift Container Platform クラスター用の自己管理 Clair コンテナーをデプロイします。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。

手順

1. 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/clairv4/config/
```

2. **disable_updaters** パラメーターを **true** に設定して Clair 設定ファイルを作成します。次に例を示します。

```
---
indexer:
  airgap: true
---
```

```
matcher:
  disable_updaters: true
---
```

3. 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 -it --rm --name clairv4 \
-p 8081:8081 -p 8088:8088 \
-e CLAIR_CONF=/clair/config.yaml \
-e CLAIR_MODE=combo \
-v /etc/clairv4/config:/clair:Z \
registry.redhat.io/quay/clair-rhel8:v3.11.0
```

3.4.3.2.3. 接続された Clair インスタンスからアップデータバンドルをエクスポートする

次の手順を使用して、インターネットにアクセスできる Clair インスタンスから更新プログラムバンドルをエクスポートします。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。
- Clair をデプロイしている。
- Clair の **config.yaml** ファイルで、**disable_updaters** および **airgap** パラメーターが **true** に設定されている。

手順

- インターネットにアクセスできる Clair インスタンスから、設定ファイルで **clairctl** CLI ツールを使用して、アップデーターバンドルをエクスポートします。以下に例を示します。

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

3.4.3.2.4. 非接続の OpenShift Container Platform クラスター内の Clair データベースへのアクセスの設定

以下の手順を使用して、非接続の OpenShift Container Platform クラスター内の Clair データベースへのアクセスを設定します。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。
- Clair をデプロイしている。
- Clair の **config.yaml** ファイルで、**disable_updaters** および **airgap** パラメーターが **true** に設定されている。
- インターネットにアクセスできる Clair インスタンスからアップデーターバンドルをエクスポートしている。

手順

1. CLI ツール **oc** を使用して、Clair データベースサービスを特定します。次に例を示します。

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

出力例

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

2. Clair データベースポートを転送して、ローカルマシンからアクセスできるようにします。以下に例を示します。

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

3. Clair の **config.yaml** ファイルを更新します。次に例を示します。

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

(1) 複数の **connstring** フィールドの **host** の値を **localhost** に置き換えます。

(2) **rhel-repository-scanner** パラメーターの詳細は、「Common Product Enumeration 情報へのリポジトリーのマッピング」を参照してください。

(3) **rhel_containersscanner** パラメーターの詳細は、「Common Product Enumeration へのリポジトリーのマッピング」を参照してください。

3.4.3.2.5. 非接続の OpenShift Container Platform クラスターへのアップデーターバンドルのインポート

以下の手順を使用して、アップデーターバンドルを非接続の OpenShift Container Platform クラスターにインポートします。

前提条件

- **clairctl** コマンドラインユーティリティーツールをインストールしている。

- Clair をデプロイしている。
- Clair の **config.yaml** ファイルで、**disable_updaters** および **airgap** パラメーターが **true** に設定されている。
- インターネットにアクセスできる Clair インスタンスからアップデーターバンドルをエクスポートしている。
- アップデーターバンドルを非接続環境に転送している。

手順

- CLI ツール **clairctl** を使用して、アップデーターバンドルを OpenShift Container Platform によってデプロイされた Clair データベースにインポートします。

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

3.4.4. Common Product Enumeration へのリポジトリーのマッピング

注記

現在、Common Product Enumeration へのリポジトリーのマッピングは、IBM Power および IBM Z ではサポートされていません。

Clair の Red Hat Enterprise Linux (RHEL) スキャナーは、Common Product Enumeration (CPE) ファイルに依存して、RPM パッケージを対応するセキュリティーデータにマッピングし、マッチングする結果を生成します。これらのファイルは製品セキュリティーによって所有され、毎日更新されます。

スキャナーが RPM を適切に処理するには、CPE ファイルが存在するか、ファイルへのアクセスが許可されている必要があります。ファイルが存在しないと、コンテナーイメージにインストールされている RPM パッケージはスキャンされません。

表3.1 Clair CPE マッピングファイル

CPE	JSON マッピングファイルへのリンク
repos2cpe	Red Hat Repository-to-CPE JSON
names2repos	Red Hat Name-to-Repos JSON

非接続の Clair インストール用のデータベースに CVE 情報をアップロードするだけでなく、マッピングファイルをローカルで使用できるようにする必要もあります。

- スタンドアロン Red Hat Quay および Clair デプロイメントの場合は、マッピングファイルを Clair Pod に読み込む必要があります。
- OpenShift Container Platform デプロイメント上の Red Hat Quay の場合、Clair コンポーネントを **unmanaged** に設定する必要があります。次に、Clair を手動でデプロイメントし、マッピングファイルのローカルコピーを読み込むように設定する必要があります。

3.4.4.1. Common Product Enumeration サンプル設定へのリポジトリーのマッピング

Clair 設定の **repo2cpe_mapping_file** フィールドと **name2repos_mapping_file** フィールドを使用して、CPE JSON マッピングファイルを含めます。以下に例を示します。

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

詳細は、OVAL セキュリティーデータをインストール済みの RPM と正確にマッチングする方法 を参照してください。

第4章 インフラストラクチャーノードへの RED HAT QUAY のデプロイ

デフォルトでは、Red Hat Quay Operator を使用してレジストリーをデプロイする際に **Quay** 関連の Pod は任意のワーカーノードに配置されます。マシンセットを使用してインフラストラクチャーコンポーネントのみをホストするようにノードを設定する方法の詳細は、インフラストラクチャーマシンセットの作成 を参照してください。

OpenShift Container Platform マシンセットリソースを使用してインフラノードをデプロイしていない場合、このドキュメントのセクションでは、インフラストラクチャー目的でノードに手動でラベルを付けてテイントする方法を示します。手動またはマシンセットを使用してインフラストラクチャーノードを設定したら、ノードセレクターおよび容認を使用してこれらのノードに対する **Quay** Pod の配置を制御できます。

4.1. インフラストラクチャー用ノードへのラベルとテインとの追加

次の手順を実行して、インフラストラクチャー用のノードにラベルとテインとを追加します。

1. 次のコマンドを入力して、マスターノードとワーカーノードを表示します。この例では、3 つのマスターノードと 6 つのワーカーノードがあります。

```
$ oc get nodes
```

出力例

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

2. 次のコマンドを入力して、インフラストラクチャーで使用する 3 つのワーカーノードにラベルを付けます。

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

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

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

3. ここで、クラスター内のノードをリストすると、最後の 3 つのワーカーノードには **infra** ロールが与えられます。以下に例を示します。

```
$ oc get nodes
```

例

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

4. ワーカーノードに **infra** ロールが割り当てられている場合、ユーザーのワークロードが誤ってインフラノードに割り当てられる可能性があります。これを回避するには、infra ノードにテイントを適用し、制御する Pod に容認を追加します。以下に例を示します。

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

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

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

4.2. ノードセレクターと容認を使用したプロジェクト作成

ノードセレクターと容認を持つプロジェクトを作成するには、次の手順を実行します。

注記

次の手順は、インストールされている Red Hat Quay Operator と、デプロイメントの作成時に使用した namespace を削除することによっても完了できます。その後、次のアノテーションを付けて新しいリソースを作成できます。

手順

1. 次のコマンドを入力して、Red Hat Quay がデプロイされている namespace と次のアノテーションを編集します。

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

出力例

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

2. 次のコマンドを入力して、使用可能な Pod のリストを取得します。

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

出力例

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

3. 次のコマンドを入力して、使用可能な Pod を削除します。

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

出力例

```
pod "example-registry-clair-app-5744dd64c9-9d5jt" deleted
pod "example-registry-clair-app-5744dd64c9-fg86n" deleted
pod "example-registry-clair-postgres-845b47cd88-vdchz" deleted
pod "example-registry-quay-app-64cbc5bcf-8zvgc" deleted
pod "example-registry-quay-app-64cbc5bcf-pvlz6" deleted
pod "example-registry-quay-app-upgrade-8gspn" deleted
pod "example-registry-quay-database-784d78b6f8-2vkml" deleted
```

```
pod "example-registry-quay-mirror-d5874d8dc-fmknp" deleted
pod "example-registry-quay-mirror-d5874d8dc-t4mff" deleted
pod "example-registry-quay-redis-79848898cb-6qf5x" deleted
```

Pod を削除すると、Pod は自動的に再稼働し、専用のインフラストラクチャーノードでスケジュールされます。

4.3. OPENSHIFT CONTAINER PLATFORM 上の RED HAT QUAY を特定の NAMESPACE にインストールする

以下の手順を使用して、OpenShift Container Platform 上の Red Hat Quay を特定の namespace にインストールします。

- Red Hat Quay Operator を特定の namespace にインストールするには、次のコマンドのように、適切なプロジェクト namespace を明示的に指定する必要があります。
  次の例では、**quay-registry** namespace を使用します。これにより、**quay-operator** Pod が 3 つのインフラストラクチャーノードのいずれかに配置されます。以下に例を示します。

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

出力例

```
NAME                                        READY   STATUS    RESTARTS   AGE   IP            NODE

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

4.4. RED HAT QUAY レジストリーの作成

Red Hat Quay レジストリーを作成するには、次の手順を実行します。

- 次のコマンドを入力して、Red Hat Quay レジストリーを作成します。次に、デプロイメントが **ready** としてマークされるまで待ちます。次の例では、インフラストラクチャー目的でラベルを付けた 3 つのノード上でのみスケジュールされていることがわかります。

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

出力例

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

```
example-registry-quay-redis-5688ddcdb6-ndp4t          1/1     Running     0          5m56s   10.130.2.78   user1-jcnp6-worker-d-m9gg4.c.quay-devel.internal
quay-operator.v3.4.1-6f6597d8d8-bd4dp                 1/1     Running     0          22m     10.131.0.16   user1-jcnp6-worker-d-h5tv2.c.quay-devel.internal
```

4.5. RED HAT QUAY OPERATOR が単一の NAMESPACE にインストールされている場合のモニタリングの有効化

注記

現在、Red Hat Quay Operator が単一の namespace にインストールされている場合のモニタリングの有効化は、IBM Power および IBM Z ではサポートされていません。

Red Hat Quay Operator が単一の namespace にインストールされている場合、モニタリングコンポーネントは **unmanaged** に設定されます。モニタリングを設定するには、OpenShift Container Platform でユーザー定義の namespace に対してモニタリングを有効にする必要があります。

詳細は、OpenShift Container Platform ドキュメントの モニタリングスタックの設定 および ユーザー定義プロジェクトのモニタリングの有効化 を参照してください。

以下のセクションでは、OpenShift Container Platform ドキュメントに基づいて Red Hat Quay のモニタリングを有効にする方法を示します。

4.5.1. クラスターモニタリング設定マップの作成

次の手順を使用して、**cluster-monitoring-config ConfigMap** オブジェクトが存在するかどうかを確認します。

手順

1. 次のコマンドを入力して、**cluster-monitoring-config** ConfigMap オブジェクトが存在するかどうかを確認します。

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

出力例

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

2. オプション: **ConfigMap** オブジェクトが存在しない場合は、YAML マニフェストを作成します。次の例では、ファイルの名前は **cluster-monitoring-config.yaml** です。

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

3. オプション: **ConfigMap** オブジェクトが存在しない場合は、**ConfigMap** オブジェクトを作成します。

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

出力例

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

4. 次のコマンドを実行して、**ConfigMap** オブジェクトが存在することを確認します。

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

出力例

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

4.5.2. ユーザー定義のワークロード監視 ConfigMap オブジェクトの作成

次の手順を使用して、**user-workload-monitoring-config ConfigMap** オブジェクトが存在するかどうかを確認します。

手順

1. 次のコマンドを入力して、**user-workload-monitoring-config ConfigMap** オブジェクトが存在するかどうかを確認します。

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

出力例

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

2. **ConfigMap** オブジェクトが存在しない場合は、YAML マニフェストを作成します。次の例では、ファイルの名前は **user-workload-monitoring-config.yaml** です。

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

3. オプション: 次のコマンドを入力して、**ConfigMap** オブジェクトを作成します。

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

出力例

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

4.5.3. ユーザー定義プロジェクトのモニタリングの有効化

ユーザー定義プロジェクトの監視を有効にするには、次の手順を実行します。

手順

1. 次のコマンドを入力して、ユーザー定義プロジェクトの監視が実行されているかどうかを確認します。

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

出力例

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

2. 次のコマンドを入力して、**cluster-monitoring-config ConfigMap** を編集します。

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

3. **config.yaml** ファイルで **enableUserWorkload: true** を設定して、クラスター上のユーザー定義プロジェクトの監視を有効にします。

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

4. 次のコマンドを入力してファイルを保存し、変更を適用して、適切な Pod が実行されていることを確認します。

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

出力例

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

4.5.4. Red Hat Quay メトリックを公開するための Service オブジェクトの作成

以下の手順を使用して **Service** オブジェクトを作成し、Red Hat Quay メトリクスを公開します。

手順

1. Service オブジェクトの YAML ファイルを作成します。

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

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

2. 次のコマンドを入力して、**Service** オブジェクトを作成します。

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

出力例

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

4.5.5. ServiceMonitor オブジェクトの作成

ServiceMonitor リソースを作成してメトリックを取得するように OpenShift Monitoring を設定するには、次の手順を使用します。

手順

1. **ServiceMonitor** リソースの YAML ファイルを作成します。

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

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

```
  selector:
    matchLabels:
      quay-component: monitoring
EOF
```

2. 次のコマンドを入力して、**ServiceMonitor** リソースを作成します。

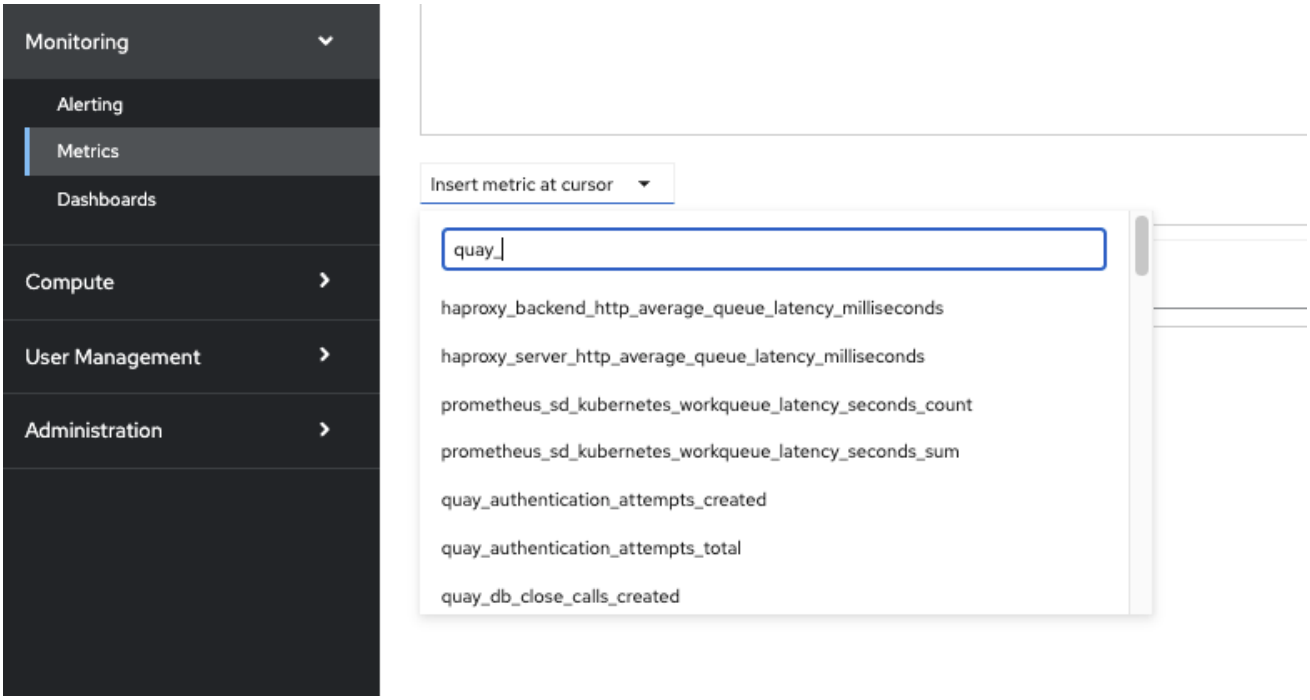
```
$ oc apply -f quay-service-monitor.yaml
```

出力例

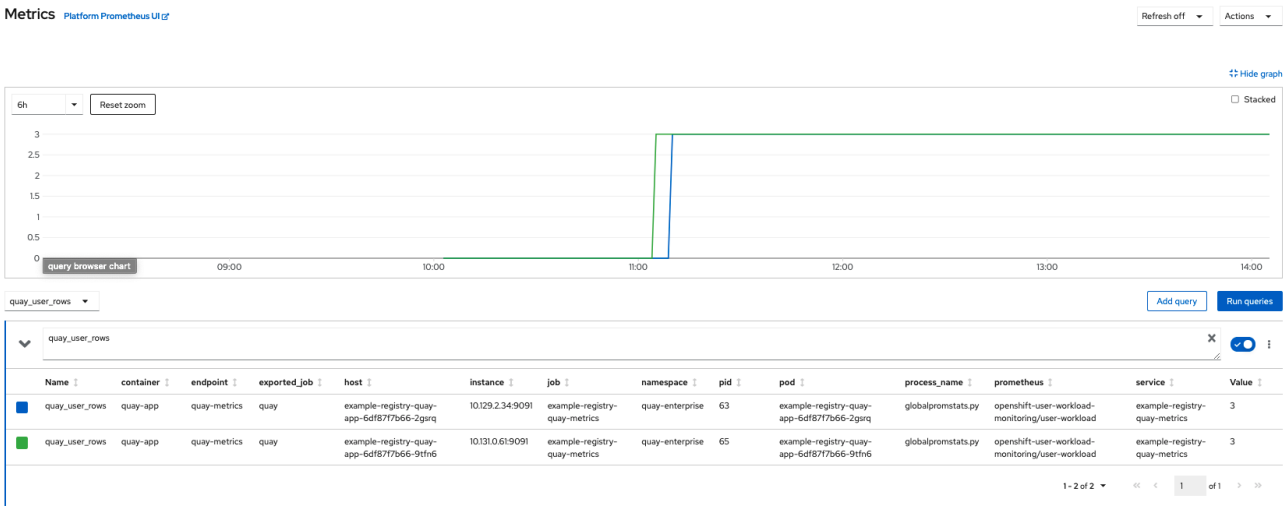
```
servicemonitor.monitoring.coreos.com/example-registry-quay-metrics-monitor created
```

4.5.6. OpenShift Container Platform でのメトリックの表示

OpenShift Container Platform コンソールの **Monitoring → Metrics** からメトリックにアクセスできます。式フィールドに **quay_** と入力すると、使用可能なメトリックのリストが表示されます。



たとえば、レジストリーにユーザーを追加した場合は、**quay-users_rows** メトリックを選択します。



4.6. 管理ストレージのサイズ変更

OpenShift Container Platform に Red Hat Quay をデプロイすると、次の 3 つの異なる永続ボリューム要求 (PVC) がデプロイされます。

- PostgreSQL 13 レジストリー用の PVC
- Clair PostgreSQL 13 レジストリー用の PVC
- NooBaa をバックエンドストレージとして使用する PVC

注記

Red Hat Quay と NooBaa の間の接続は、OpenShift Container Platform の S3 API および ObjectBucketClaim API を通じて行われます。Red Hat Quay は、その API グループを利用して NooBaa にバケットを作成し、アクセスキーを取得して、すべてを自動的にセットアップします。バックエンド (NooBaa) 側では、そのバケットはバッキングストア内に作成されます。そのため、NooBaa PVC は Red Hat Quay Pod にマウントまたは接続されません。

PostgreSQL 13 の PVC および Clair PostgreSQL 13 の PVC のデフォルトサイズは 50 GiB に設定されています。以下の手順を使用して、OpenShift Container Platform コンソールでこれらの PVC のストレージを拡張できます。

注記

次の手順は、Red Hat OpenShift Data Foundation での 永続ボリューム要求の拡張 と共通しています。

4.6.1. Red Hat Quay の PostgreSQL 13 PVC のサイズ変更

PostgreSQL 13 の PVC および Clair PostgreSQL 13 の PVC のサイズを変更するには、次の手順を使用します。

前提条件

- OpenShift Container Platform のクラスター管理者権限がある。

手順

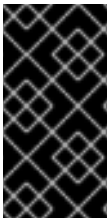
1. OpenShift Container Platform コンソールにログインし、**Storage → Persistent Volume Claims** を選択します。
2. PostgreSQL 13 または Clair PostgreSQL 13 に必要な **PersistentVolumeClaim** を選択します (例: **example-registry-quay-postgres-13**)。
3. **Action** メニューから **Expand PVC** を選択します。
4. 永続ボリューム要求 (PVC) の新しいサイズを入力し、**Expand** を選択します。
   数分後、拡張されたサイズが PVC の **Capacity** フィールドに反映されます。

4.7. デフォルトの OPERATOR イメージのカスタマイズ

注記

現在、デフォルトの Operator イメージのカスタマイズは、IBM Power および IBM Z ではサポートされていません。

特定の状況では、Red Hat Quay Operator が使用するデフォルトのイメージをオーバーライドすると便利な場合があります。これを行うには、Red Hat Quay Operator **ClusterServiceVersion** で 1 つ以上の環境変数を設定します。



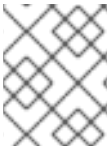
重要

このメカニズムの使用は、実稼働 Red Hat Quay 環境ではサポートされておらず、開発またはテストの目的でのみ使用することが強く推奨されます。Red Hat Quay Operator でデフォルト以外のイメージを使用する場合は、デプロイメントが正しく機能するという保証はありません。

4.7.1. 環境変数

以下の環境変数は、Red Hat Quay Operator でコンポーネントイメージをオーバーライドするために使用されます。

環境変数	コンポーネント
RELATED_IMAGE_COMPONENT_QUAY	**base**
RELATED_IMAGE_COMPONENT_CLAIR	**clair**
RELATED_IMAGE_COMPONENT_POSTGRES	**postgres** および **clair** データベース
RELATED_IMAGE_COMPONENT_REDIS	**redis**



注記

オーバーライドされたイメージは、タグ (:latest) ではなく、マニフェスト (@sha256:) によって参照する **必要があります**。

4.7.2. 実行中のオペレータへのオーバーライドの適用

Red Hat Quay Operator が Operator Lifecycle Manager (OLM) を通じてクラスターにインストールされている場合、マネージドコンポーネントのコンテナーイメージは、**ClusterServiceVersion** オブジェクトを変更すると簡単にオーバーライドできます。

以下の手順を使用して、実行中の Red Hat Quay Operator にオーバーライドを適用します。

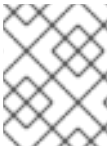
手順

1. **ClusterServiceVersion** オブジェクトは、クラスター内で実行中の Operator を Operator Lifecycle Manager が表現したものです。Kubernetes UI または **kubectl/oc** CLI ツールを使用して、Red Hat Quay Operator の **ClusterServiceVersion** を見つけます。以下に例を示します。

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

2. UI、**oc edit**、または別の方法を使用して、Red Hat Quay **ClusterServiceVersion** を変更して、オーバーライドイメージを指す上記の環境変数を含めます。
JSONPath: **spec.install.spec.deployments[0].spec.template.spec.containers[0].env**

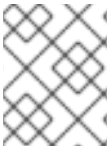
```
- name: RELATED_IMAGE_COMPONENT_QUAY
  value:
quay.io/projectquay/quay@sha256:c35f5af964431673f4ff5c9e90bdf45f19e38b8742b5903d41c
10cc7f6339a6d
- name: RELATED_IMAGE_COMPONENT_CLAIR
  value:
quay.io/projectquay/clair@sha256:70c99feceb4c0973540d22e740659cd8d616775d3ad1c169
8ddf71d0221f3ce6
- name: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



注記

これは Operator レベルで実行されるため、すべての **QuayRegistry** はこれらの同じオーバーライドを使用してデプロイされています。

4.8. AWS S3 CLOUDFRONT



注記

現在、AWS S3 CloudFront の使用は、IBM Power および IBM Z ではサポートされていません。

バックエンドレジストリーストレージに AWS S3 Cloudfront を使用している場合は、次の手順を使用します。

手順

1. 次のコマンドを入力してレジストリーキーを指定します。

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

第5章 RED HAT QUAY ビルドの機能強化

Red Hat Quay ビルドは仮想化プラットフォーム上で実行できます。以前のビルド設定を実行するための下位互換性も利用できます。

5.1. RED HAT QUAY ビルドの制限

特権のないコンテキストで Red Hat Quay でビルドを実行すると、以前のビルド戦略で機能していた一部のコマンドが失敗する可能性があります。ビルド戦略を変更しようとすると、ビルドのパフォーマンスの問題と信頼性の問題が発生する可能性があります。

コンテナーでビルドを直接実行しても、仮想マシンを使用する場合と同じように分離されることはありません。ビルド環境を変更すると、以前は機能していたビルドが失敗する可能性もあります。

5.2. OPENSHIFT CONTAINER PLATFORM を使用した RED HAT QUAY ビルダー環境の作成

このセクションの手順では、OpenShift Container Platform で Red Hat Quay 仮想ビルダー環境を作成する方法を説明します。

5.2.1. OpenShift Container Platform TLS コンポーネント

tls コンポーネントを使用すると、TLS 設定を制御できます。

注記

TLS コンポーネントが Operator によって管理されている場合、Red Hat Quay 3.11 はビルダーをサポートしません。

tls を **unmanaged** に設定する場合は、独自の **ssl.cert** ファイルと **ssl.key** ファイルを提供します。このとき、クラスターでビルダーをサポートする場合は、Quay ルートとビルダールート名の両方を証明書の SAN リストに追加するか、ワイルドカードを使用する必要があります。

ビルダールートを追加するには、次の形式を使用します。

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

5.2.2. OpenShift Container Platform ビルダー向けの Red Hat Quay の使用

ビルダーには SSL/TLS 証明書が必要です。SSL/TLS 証明書の詳細は Red Hat Quay コンテナーへの TLS 証明書の追加 を参照してください。

Amazon Webservice (AWS) S3 ストレージを使用している場合は、ビルダーを実行する前に、AWS コンソールでストレージバケットを変更する必要があります。必要なパラメーターは、次のセクションの AWS S3 ストレージバケットの変更を参照してください。

5.2.2.1. 仮想ビルダー向けの OpenShift Container Platform の準備

以下の手順を使用して、Red Hat Quay 仮想ビルダー用に OpenShift Container Platform を準備します。

注記

- この手順は、クラスターがすでにプロビジョニングされており、Quay Operator が実行されていることを前提とします。
- この手順は、OpenShift Container Platform で仮想 namespace を設定するためのものです。

手順

1. クラスター管理者アカウントを使用して Red Hat Quay クラスターにログインします。
2. 次のコマンドを実行して、仮想ビルダーが実行される新しいプロジェクト (**virtual-builders** など) を作成します。

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

3. 次のコマンドを入力して、ビルドの実行に使用されるプロジェクトに **ServiceAccount** を作成します。

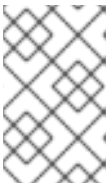
```
$ oc create sa -n virtual-builders quay-builder
```

4. 作成したサービスアカウントに編集権限を付与して、ビルドを実行できるようにします。

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

5. 次のコマンドを入力して、Quay ビルダーに **anyuid scc** 権限を付与します。

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



注記

このアクションには、クラスター管理者特権が必要です。非特権ビルドまたはルートレスビルドを機能させるには、ビルダーを Podman ユーザーとして実行する必要があるため、これが必要です。

6. Quay ビルダーサービスアカウントのトークンを取得します。
   a. OpenShift Container Platform 4.10 以前のバージョンを使用している場合は、以下のコマンドを入力します。

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

   b. OpenShift Container Platform 4.11 以降を使用している場合は、以下のコマンドを入力します。

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

注記

トークンの有効期限が切れたら、新しいトークンを要求する必要があります。必要に応じて、カスタムの有効期限を追加することもできます。たとえば、トークンを 2 週間保持するには、**--duration 20160m** と指定します。

出力例

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IldfQUJkaDVmb3ltTHZ0dGZMYjhIWnYxZTQzN2dJVEJxcDJscldSdEUtYWsifQ...
```

7. 次のコマンドを入力してビルダールートを決定します。

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

出力例

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

8. 次のコマンドを入力して、拡張子が .crt の自己署名 SSL/TLS 証明書を生成します。

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

出力例

```
ca.crt
```

9. 次のコマンドを入力して、**ca.crt** ファイルの名前を **extra_ca_cert_build_cluster.crt** に変更します。

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

10. **Console** で設定バンドルのシークレットを見つけ、**Actions → Edit Secret** を選択して、適切なビルダー設定を追加します。

```
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- <superusername>
FEATURE_USER_CREATION: false
FEATURE_QUOTA_MANAGEMENT: true
FEATURE_BUILD_SUPPORT: True
BUILDMAN_HOSTNAME: <sample_build_route> 1
BUILD_MANAGER:
  - ephemeral
  - ALLOWED_WORKER_COUNT: 1
```

```
ORCHESTRATOR_PREFIX: buildman/production/
JOB_REGISTRATION_TIMEOUT: 3600 2
ORCHESTRATOR:
  REDIS_HOST: <sample_redis_hostname> 3
  REDIS_PASSWORD: ""
  REDIS_SSL: false
  REDIS_SKIP_KEYSPACE_EVENT_SETUP: false
EXECUTORS:
  - EXECUTOR: kubernetesPodman
    NAME: openshift
    BUILDER_NAMESPACE: <sample_builder_namespace> 4
    SETUP_TIME: 180
    MINIMUM_RETRY_THRESHOLD: 0
    BUILDER_CONTAINER_IMAGE: <sample_builder_container_image> 5
    # Kubernetes resource options
    K8S_API_SERVER: <sample_k8s_api_server> 6
    K8S_API_TLS_CA: <sample_crt_file> 7
    VOLUME_SIZE: 8G
    KUBERNETES_DISTRIBUTION: openshift
    CONTAINER_MEMORY_LIMITS: 300m 8
    CONTAINER_CPU_LIMITS: 1G 9
    CONTAINER_MEMORY_REQUEST: 300m 10
    CONTAINER_CPU_REQUEST: 1G 11
    NODE_SELECTOR_LABEL_KEY: ""
    NODE_SELECTOR_LABEL_VALUE: ""
    SERVICE_ACCOUNT_NAME: <sample_service_account_name>
    SERVICE_ACCOUNT_TOKEN: <sample_account_token> 12
```

1. ビルドルートは、Open Shift Operators namespace の名前で **oc get route -n** を実行することにより取得されます。ルートの最後にポートを指定する必要があり、**quayregistry-cr-name-quay-builder-ocp-namespace.ocp-domain-name:443** の形式を使用する必要があります。

2. **JOB_REGISTRATION_TIMEOUT** パラメーターの設定が低すぎると、**failed to register job to build manager: rpc error: code = Unauthenticated desc = Invalid build token: Signature has expired** エラーが発生する可能性があります。このパラメーターは少なくとも 240 に設定することを推奨します。

3. Redis ホストにパスワードまたは SSL/TLS 証明書がある場合は、それに応じて更新する必要があります。

4. 仮想ビルダーの namespace の名前と一致するように設定します (例: **virtual-builders**)。

5. 早期アクセスの場合、**BUILDER_CONTAINER_IMAGE** は現在 **quay.io/projectquay/quay-builder:3.7.0-rc.2** です。ただし、早期アクセス期間中に変更される可能性があります。これが発生した場合は、顧客に警告が表示されます。

6. **K8S_API_SERVER** は、**oc cluster-info** を実行して取得します。

7. カスタム CA 証明書を手動で作成して追加する必要があります (例 **K8S_API_TLS_CA: /conf/stack/extra_ca_certs/build_cluster.crt**)。

8. 指定しないと、デフォルトは **5120Mi** です。

9. 仮想ビルドの場合は、クラスターに十分なリソースがあることを確認する必要がありま

10 指定しないと、デフォルトは **3968Mi** です。

11 指定しないと、デフォルトは **500m** です。

12 **oc create sa** の実行時に取得します。

設定サンプル

```
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- quayadmin
FEATURE_USER_CREATION: false
FEATURE_QUOTA_MANAGEMENT: true
FEATURE_BUILD_SUPPORT: True
BUILDMAN_HOSTNAME: example-registry-quay-builder-quay-
enterprise.apps.docs.quayteam.org:443
BUILD_MANAGER:
  - ephemeral
  - ALLOWED_WORKER_COUNT: 1
    ORCHESTRATOR_PREFIX: buildman/production/
    JOB_REGISTRATION_TIMEOUT: 3600
    ORCHESTRATOR:
      REDIS_HOST: example-registry-quay-redis
      REDIS_PASSWORD: ""
      REDIS_SSL: false
      REDIS_SKIP_KEYSPACE_EVENT_SETUP: false
    EXECUTORS:
      - EXECUTOR: kubernetesPodman
        NAME: openshift
        BUILDER_NAMESPACE: virtual-builders
        SETUP_TIME: 180
        MINIMUM_RETRY_THRESHOLD: 0
        BUILDER_CONTAINER_IMAGE: quay.io/projectquay/quay-builder:3.7.0-rc.2
        # Kubernetes resource options
        K8S_API_SERVER: api.docs.quayteam.org:6443
        K8S_API_TLS_CA: /conf/stack/extra_ca_certs/build_cluster.crt
        VOLUME_SIZE: 8G
        KUBERNETES_DISTRIBUTION: openshift
        CONTAINER_MEMORY_LIMITS: 1G
        CONTAINER_CPU_LIMITS: 1080m
        CONTAINER_MEMORY_REQUEST: 1G
        CONTAINER_CPU_REQUEST: 580m
        NODE_SELECTOR_LABEL_KEY: ""
        NODE_SELECTOR_LABEL_VALUE: ""
        SERVICE_ACCOUNT_NAME: quay-builder
        SERVICE_ACCOUNT_TOKEN:
"eyJhbGciOiJSUzI1NiIsImtpZCI6IldfQUJkaDVmb3ltTHZ0dGZMYjhIWnYxZTQzN2dJVEJxcDJs
cldSdEUtYWsifQ"
```

5.2.2.2. SSL/TLS 証明書の手動追加

設定ツールの既知の問題のため、ビルダーを適切に実行するには、カスタム SSL/TLS 証明書を手動で追加する必要があります。次の手順を使用して、カスタム SSL/TLS 証明書を手動で追加します。

SSL/TLS 証明書の作成の詳細は、Red Hat Quay コンテナーへの TLS 証明書の追加 を参照してください。

5.2.2.2.1. 証明書の作成と署名

SSL/TLS 証明書を作成して署名するには、次の手順を実行します。

手順

- 認証局を作成し、証明書に署名します。詳細は、認証局の作成と証明書への署名 を参照してください。

  openssl.cnf

```
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = example-registry-quay-quay-enterprise.apps.docs.quayteam.org (1)
DNS.2 = example-registry-quay-builder-quay-enterprise.apps.docs.quayteam.org (2)
```

(1) Red Hat Quay レジストリーの URL の **alt_name** を含める必要があります。

(2) **BUILDMAN_HOSTNAME** の **alt_name**

  サンプルコマンド

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

5.2.2.2.2. TLS の管理対象外への設定

次の手順を使用して、**king:tls** を管理対象外に設定します。

手順

1. Red Hat Quay Registry YAML で、**kind: tls** を **managed: false** に設定します。

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

2. **Events** ページでは、適切な **config.yaml** ファイルを設定するまで変更がブロックされます。以下に例を示します。

```
    - lastTransitionTime: '2022-03-28T12:56:49Z'
      lastUpdateTime: '2022-03-28T12:56:49Z'
      message: >-
        required component `tls` marked as unmanaged, but `configBundleSecret`
        is missing necessary fields
      reason: ConfigInvalid
      status: 'True'
```

5.2.2.2.3. 一時的なシークレットの作成

次の手順を使用して、CA 証明書の一時的なシークレットを作成します。

手順

1. CA 証明書のデフォルトの namespace にシークレットを作成します。

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

2. **ssl.key** ファイルおよび **ssl.cert** ファイルのデフォルトの namespace にシークレットを作成します。

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

5.2.2.2.4. シークレットデータの設定 YAML へのコピー

次の手順を使用して、シークレットデータを **config.yaml** ファイルにコピーします。

手順

1. コンソール UI の **Workloads → Secrets** で新しいシークレットを見つけます。
2. シークレットごとに、YAML ビューを見つけます。

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

```
kind: Secret
apiVersion: v1
metadata:
  name: quay-config-ssl
```

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

3. UI で Red Hat Quay レジストリー設定バンドルのシークレットを見つけるか、以下のようなコマンドを実行してコマンドラインから見つけます。

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

4. OpenShift Container Platform コンソールで、設定バンドルのシークレットの YAML タブを選択し、作成した 2 つのシークレットからデータを追加します。

```
kind: Secret
apiVersion: v1
metadata:
  name: init-config-bundle-secret
  namespace: quay-enterprise
  uid: 4724aca5-bff0-406a-9162-ccb1972a27c1
  resourceVersion: '4383160'
  creationTimestamp: '2022-03-22T12:35:59Z'
...
data:
  config.yaml: >-
    RkVBVFVSRV9VU0VSX0lOSVRJQUxJWkU6IHRydWUKQlJ...
  extra_ca_cert_build_cluster.crt: >-

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

5. **Save** をクリックします。

6. 次のコマンドを入力して、Pod が再起動しているかどうかを確認します。

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

出力例

```
NAME                                        READY   STATUS              RESTARTS   AGE
...
example-registry-quay-app-6786987b99-vgg2v          0/1     ContainerCreating   0          2s
example-registry-quay-app-7975d4889f-q7tvl          1/1     Running             0          5d21h
```

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

7. Red Hat Quay レジストリーが再設定されたら、次のコマンドを入力して、Red Hat Quay アプリの Pod が実行されているかどうかを確認します。

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

出力例

```
example-registry-quay-app-6786987b99-sz6kb            1/1     Running            0          7m45s
example-registry-quay-app-6786987b99-vgg2v            1/1     Running            0          9m1s
example-registry-quay-app-upgrade-lswsn               0/1     Completed          0          6d1h
example-registry-quay-config-editor-77847fc4f5-nsbbv  1/1     Running            0          9m1s
example-registry-quay-database-66969cd859-n2ssm       1/1     Running            0          6d1h
example-registry-quay-mirror-758fc68ff7-5wxlp         1/1     Running            0          8m29s
example-registry-quay-mirror-758fc68ff7-lbl82         1/1     Running            0          8m29s
example-registry-quay-redis-7cc5f6c977-956g8          1/1     Running            0          5d21h
```

8. ブラウザーで、レジストリーエンドポイントにアクセスし、証明書が適切に更新されていることを確認します。以下に例を示します。

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

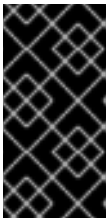
5.2.2.3. UI を使用してビルドトリガーを作成

UI を使用してビルドトリガーを作成するには、次の手順に従います。

手順

1. Red Hat Quay リポジトリーにログインします。
2. **Create New Repository** をクリックして、**testrepo** などの新しいレジストリーを作成します。
3. **Repositories** ページで、ナビゲーションペインの **Builds** タブをクリックします。または、対応する URL を直接使用します。

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



重要

場合によっては、ビルダーでホスト名の解決に問題が発生することがあります。この問題は、ジョブオブジェクトで **default** に設定されている **dnsPolicy** に関連している可能性があります。現在、この問題に対する回避策はありません。これは、Red Hat Quay の将来のバージョンで解決される予定です。

4. **Create Build Trigger→ Custom Git Repository Push** をクリックします。
5. Git リポジトリーのクローン作成に使用する HTTPS または SSH スタイルの URL を入力し、**Continue** をクリックします。以下に例を示します。

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

6. **Tag manifest with the branch or tag name**を確認し、**Continue** をクリックします。
7. トリガーが呼び出されたときにビルドする Dockerfile の場所 (たとえば **/Dockerfile**) を入力し、**Continue** をクリックします。
8. Docker ビルドのコンテキストの場所 (たとえば /) を入力し、**Continue** をクリックします。
9. 必要に応じて、ロボットアカウントを作成します。それ以外の場合は、**Continue** をクリックします。
10. **Continue** をクリックして、パラメーターを確認します。
11. **Builds** ページで、トリガー名の **Options** アイコンをクリックし、**Run Trigger Now** をクリックします。
12. Git リポジトリーからコミット SHA を入力し、**Start Build** をクリックします。
13. ビルドのステータスを確認するには、**Build History** ページで commit をクリックするか、**oc get pods -n virtual-builders** を実行します。以下に例を示します。

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

出力例

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

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

出力例

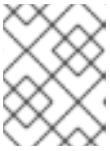
```
NAME                                     READY   STATUS        RESTARTS   AGE
f192fe4a-c802-4275-bcce-d2031e635126-9l2b5-25lg2   1/1     Terminating   0          9s
```

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

出力例

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

14. ビルドが完了したら、ナビゲーションペインのタグで **Tags** のステータスを確認できます。



注記

早期アクセスにより、完全なビルドログとビルドのタイムスタンプは現在利用できません。

5.2.2.4. AWS S3 ストレージバケットの変更

AWS S3 ストレージを使用している場合は、ビルダーを実行する前に、AWS コンソールでストレージバケットを変更する必要があります。

手順

1. s3.console.aws.com で AWS コンソールにログインします。
2. 検索バーで **S3** を検索し、**S3** をクリックします。
3. バケットの名前 (**myawsbucket** など) をクリックします。
4. **Permissions** タブをクリックします。
5. **Cross-origin resource sharing (CORS)**の下に、次のパラメーターを含めます。

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

5.2.2.5. Google Cloud Platform オブジェクトバケットの変更

注記

現在、Google Cloud Platform オブジェクトバケットの変更は、IBM Power および IBM Z ではサポートされていません。

仮想ビルダーの Cross Origin Resource Sharing (CORS) を設定するには、次の手順を実行します。

注記

CORS 設定がないと、ビルド Dockerfile のアップロードは失敗します。

手順

1. 次のリファレンスを使用して、特定の CORS ニーズに合わせた JSON ファイルを作成します。以下に例を示します。

```
$ cat gcp_cors.json
```

出力例

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

2. 次のコマンドを入力して、GCP ストレージバケットを更新します。

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

出力例

```
Updating
  Completed 1
```

3. 次のコマンドを実行すると、GCP バケットの更新された CORS 設定を表示できます。

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

出力例

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

第6章 GEO レプリケーション

注記

現在、geo レプリケーション機能は IBM Power ではサポートされていません。

geo レプリケーションでは、地理的に分散した複数の Red Hat Quay デプロイメントを、クライアントやユーザーの視点から、単一のレジストリーとして動作させることができます。グローバルに分散された Red Hat Quay のセットアップにおいて、プッシュとプルのパフォーマンスが大幅に向上します。イメージデータはバックグラウンドで非同期的に複製され、クライアントには透過的なフェイルオーバー/リダイレクトが行われます。

geo レプリケーションを使用した Red Hat Quay のデプロイメントは、スタンドアロンおよび Operator デプロイメントでサポートされます。

関連情報

- geo レプリケーション機能のアーキテクチャーの詳細は、アーキテクチャーガイド を参照してください。技術的な図と概要が記載されています。

6.1. GEO レプリケーション機能

- geo レプリケーションが設定されると、コンテナーイメージのプッシュはその Red Hat Quay インスタンスの優先ストレージエンジンに書き込まれます。これは通常、リージョン内で最も近いストレージバックエンドです。
- 最初のプッシュの後、イメージデータはバックグランドで他のストレージエンジンに複製されます。
- レプリケーションロケーションのリストは設定可能で、それらは異なるストレージバックエンドにできます。
- イメージプルでは、プルのパフォーマンスを最大化するために、常に利用可能な最も近いストレージエンジンを使用します。
- レプリケーションがまだ完了していない場合、プルでは代わりにソースストレージバックエンドが使用されます。

6.2. GEO レプリケーションの要件と制約

- geo レプリケーション設定では、Red Hat Quay で、すべてのリージョンが他の全リージョンのオブジェクトストレージに対して読み取りと書き込みができるようにする必要があります。オブジェクトストレージは、他のすべてのリージョンから地理的にアクセスできる必要があります。
- 1 つの geo レプリケーションサイトでオブジェクトストレージシステムに障害が発生した場合に、そのサイトの Red Hat Quay デプロイメントをシャットダウンして、クライアントがグローバルロードバランサーにより、ストレージシステムで問題のない残りのサイトにリダイレクトされるようにする必要があります。そうしないと、クライアントでプルとプッシュの失敗が発生します。
- Red Hat Quay は、接続されたオブジェクトストレージシステムの健全性や可用性を内部的に認識しません。分散システムの健全性を監視し、ストレージのステータスに基づいてトラフィックを別のサイトにルーティングするには、ユーザーがグローバルロードバランサー (LB) を設定

する必要があります。

- geo レプリケーションデプロイメントのステータスを確認するには、**/health/endtoend** チェックポイントを使用する必要があります。このチェックポイントは全体的な健全性の監視に使用されます。**/health/endtoend** エンドポイントを使用してリダイレクトを手動で設定する必要があります。**/health/instance** エンドポイントは、ローカルインスタンスの健全性のみをチェックします。

- 1 つのサイトのオブジェクトストレージシステムが利用できなくなった場合に、残りのサイトの残りのストレージシステム (複数可) に自動的にリダイレクトされません。

- geo レプリケーションは非同期です。サイトが完全に失われると、そのサイトのオブジェクトストレージシステムに保存されていても、障害発生時に残りのサイトに複製されていないデータが失われます。

- 単一のデータベース、つまりすべてのメタデータと Red Hat Quay 設定がすべてのリージョンで共有されます。
  geo レプリケーションはデータベースをレプリケートしません。障害が発生すると、geo レプリケーションが有効になっている Red Hat Quay は別のデータベースにフェイルオーバーしません。

- 1 つの Redis キャッシュは Red Hat Quay のセットアップ全体で共有され、すべての Red Hat Quay Pod からアクセスできる必要があります。

- ストレージバックエンド以外のすべてのリージョンで同じ設定を使用する必要があります。これは、**QUAY_DISTRIBUTED_STORAGE_PREFERENCE** 環境変数を使用して明示的に設定できます。

- geo レプリケーションでは、各リージョンにオブジェクトストレージが必要です。ローカルストレージでは機能しません。

- 各リージョンは、ネットワークパスを必要とする各リージョン内のすべてのストレージエンジンにアクセスできる必要があります。

- また、ストレージプロキシーオプションを使用することもできます。

- ストレージバックエンド全体 (たとえば、すべての BLOB) がレプリケートされます。対照的に、リポジトリーミラーリングはリポジトリーまたはイメージに限定できます。

- すべての Red Hat Quay インスタンスは、通常はロードバランサーを介して同じエントリーポイントを共有する必要があります。

- すべての Red Hat Quay インスタンスは、共通の設定ファイル内で定義されているため、スーパーユーザーの同じセットが含まれる必要があります。

- geo レプリケーションでは、Clair 設定を **unmanaged** に設定する必要があります。アンマネージド Clair データベースにより、Red Hat Quay は geo-replicated environment で作業できます。この環境では、Red Hat Quay Operator の複数のインスタンスが同じデータベースと通信する必要があります。詳細は、Clair の詳細設定 を参照してください。

- geo レプリケーションには SSL/TLS 証明書とキーが必要です。詳細は、Red Hat Quay への接続を保護するための SSL/TSL の使用 を参照してください。

上記の要件を満たすことができない場合は、代わりに 2 つ以上の異なる Red Hat Quay のデプロイメントを使用し、リポジトリーミラーリング機能を利用する必要があります。

6.2.1. OpenShift Container Platform での geo レプリケーションのセットアップ

OpenShift Container Platform で geo レプリケーションを設定するには、次の手順を実行します。

手順

1. Red Hat Quay の postgres インスタンスをデプロイします。
2. 次のコマンドを入力してデータベースにログインします。

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

3. **quay** という名前で Red Hat Quay のデータベースを作成します。以下に例を示します。

```
CREATE DATABASE quay;
```

4. データベース内で pg_trm 拡張機能を有効にします。

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

5. Redis インスタンスをデプロイします。

注記

- クラウドプロバイダーに独自のサービスが含まれている場合は、Redis インスタンスをデプロイする必要がない可能性があります。
- Builder を利用している場合は、Redis インスタンスをデプロイする必要があります。

   a. Redis 用の VM をデプロイします。

   b. Red Hat Quay が実行されているクラスターからアクセスできることを確認します。

   c. ポート 6379/TCP が開いている必要があります。

   d. インスタンス内で Redis を実行します。

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

6. クラスターごとに 1 つずつ、2 つのオブジェクトストレージバックエンドを作成します。1 つのオブジェクトストレージバケットが最初のクラスター (プライマリークラスター) の近くにあり、もう 1 つが 2 番目のクラスター (セカンダリークラスター) の近くにあると理想的です。
7. 環境変数のオーバーライドを使用して、同じ設定バンドルでクラスターをデプロイし、個々のクラスターに適切なストレージバックエンドを選択します。
8. クラスターへの単一のエントリーポイントを提供するように、ロードバランサーを設定します。

6.2.1.1. OpenShift Container Platform 上の Red Hat Quay での geo レプリケーションの設定

OpenShift Container Platform 上の Red Hat Quay の geo レプリケーションを設定するには、次の手順を使用します。

手順

1. クラスター間で共有される **config.yaml** ファイルを作成します。この **config.yaml** ファイルには、一般的な PostgreSQL、Redis、ストレージバックエンドの詳細が含まれています。

   geo レプリケーションの config.yaml ファイル

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

   (1) ルートには適切な **SERVER_HOSTNAME** を使用する必要があり、グローバルロードバランサーのホスト名と一致する必要があります。

   (2) OpenShift Container Platform Operator を使用してデプロイされた Clair インスタンスの設定ファイルを取得するには、Clair 設定の取得 を参照してください。

2. 次のコマンドを入力して、**configBundleSecret** を作成します。

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

3. 各クラスターで、**configBundleSecret** を設定し、**QUAY_DISTRIBUTED_STORAGE_PREFERENCE** 環境変数のオーバーライドを使用して、そのクラスターに適切なストレージを設定します。以下に例を示します。

注記

両方のデプロイメント間の **config.yaml** ファイルは一致する必要があります。一方のクラスターに変更を加える場合は、もう一方のクラスターでも変更する必要があります。

US クラスター QuayRegistry の例

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

注記

SSL/TLS が管理対象であり、ルートが管理対象外であるため、証明書を設定バンドルで直接指定する必要があります。詳細は、TLS およびルートの設定 を参照してください。

ヨーロッパのクラスター

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
```

```
spec:
  configBundleSecret: georep-config-bundle
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: postgres
      managed: false
    - kind: clairpostgres
      managed: false
    - kind: redis
      managed: false
    - kind: quay
      managed: true
      overrides:
        env:
        - name: QUAY_DISTRIBUTED_STORAGE_PREFERENCE
          value: eustorage
    - kind: mirror
      managed: true
      overrides:
        env:
        - name: QUAY_DISTRIBUTED_STORAGE_PREFERENCE
          value: eustorage
```

注記

SSL/TLS が管理対象であり、ルートが管理対象外であるため、証明書を設定バンドルで直接指定する必要があります。詳細は、TLS およびルートの設定 を参照してください。

6.2.2. geo レプリケーションのための複合ストレージ

Red Hat Quay の geo レプリケーションは、異なる複数のレプリケーションターゲットの使用をサポートしています。たとえば、パブリッククラウドの AWS S3 ストレージとオンプレミスの Ceph ストレージを使用します。これは、すべての Red Hat Quay Pod とクラスターノードからすべてのストレージバックエンドへのアクセスを許可するという重要な要件を複雑にします。そのため、以下を使用することを推奨します。

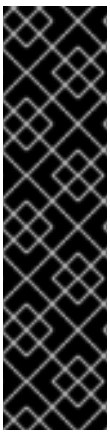
- 内部ストレージの可視性を防ぐための VPN、**または**
- Red Hat Quay が使用する特定のバケットへのアクセスのみを許可するトークンペア

これにより、Red Hat Quay のパブリッククラウドインスタンスがオンプレミスのストレージにアクセスできるようになりますが、ネットワークは暗号化され、保護され、ACL を使用するため、セキュリティー要件が満たされます。

これらのセキュリティー対策を実施できない場合は、2 つの異なる Red Hat Quay レジストリーをデプロイし、geo レプリケーションの代わりにリポジトリーミラーリングを使用することが推奨されます。

6.3. 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

OpenShift Container Platform デプロイメント上の geo レプリケートされた Red Hat Quay をアップグレードするには、次の手順を使用します。



重要

- OpenShift Container Platform デプロイメント上の geo レプリケートされた Red Hat Quay を次の y-stream リリース (例: Red Hat Quay 3.7 → Red Hat Quay 3.8) にアップグレードする場合は、アップグレード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、OpenShift Container Platform デプロイメント上の Red Hat Quay をバックアップすることを強く推奨します。

手順

この手順は、3 つ以上のシステムで Red Hat Quay レジストリーを実行していることを前提としています。この手順では、**System A**、**System B**、および **System C** という名前の 3 つのシステムを想定します。**System A** は、Red Hat Quay Operator がデプロイされるプライマリーシステムとして機能します。

1. システム B およびシステム C で、Red Hat Quay レジストリーをスケールダウンします。これを行うには、自動スケーリングを無効にし、Red Hat Quay、ミラーワーカー、および Clair (マネージドの場合) のレプリカ数をオーバーライドします。次の **quayregistry.yaml** ファイルを参照として使用します。

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

1. **Quay**、**Clair**、および **Mirroring** ワーカーの自動スケーリングを無効にします。
2. データベースおよびオブジェクトストレージにアクセスするコンポーネントのレプリカ数を 0 に設定します。

注記

Red Hat Quay レジストリーがシステム A で実行されている状態を維持する必要があります。システム A の **quayregistry.yaml** ファイルは更新しないでください。

2. **registry-quay-app**、**registry-quay-mirror**、および **registry-clair-app** Pod が消えるまで待機します。以下のコマンドを入力してステータスを確認します。

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

出力例

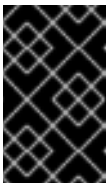
```
quay-operator.v3.7.1-6f9d859bd-p5ftc               1/1     Running     0             12m
quayregistry-clair-postgres-7487f5bd86-xnxpr       1/1     Running     1 (12m ago)   12m
quayregistry-quay-app-upgrade-xq2v6                0/1     Completed   0             12m
quayregistry-quay-redis-84f888776f-hhgms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

3. システム A で、最新の y-stream バージョンへの Red Hat Quay のアップグレードを開始します。これは手動プロセスです。インストールされた Operator のアップグレードの詳細は、インストールされた Operator のアップグレード を参照してください。Red Hat Quay アップグレードパスの詳細は、Red Hat Quay Operator のアップグレード を参照してください。

4. 新しい Red Hat Quay レジストリーがインストールされると、クラスター上で必要なアップグレードが自動的に完了します。その後、新しい Red Hat Quay Pod は、最新の y-stream バージョンで起動します。さらに、新しい **Quay** Pod がスケジュールされ、起動します。

5. Red Hat Quay UI に移動して、更新が適切に機能していることを確認します。

   a. **OpenShift** コンソールで **Operators → Installed Operators** に移動し、**Registry Endpoint** リンクをクリックします。



重要

Red Hat Quay UI が利用可能になるまで、次の手順を実行しないでください。システム A で UI が利用可能になるまで、システム B およびシステム C で Red Hat Quay レジストリーをアップグレードしないでください。

6. システム A で更新が適切に機能したことを確認した後、システム B とシステム C で Red Hat Quay のアップグレードを開始します。Operator のアップグレードにより、Red Hat Quay インストールがアップグレードされ、Pod が再起動します。

注記

データベーススキーマが新しい y-stream インストールに適したものであるため、システム B とシステム C の新しい Pod がすぐに起動します。

6.3.1. OpenShift Container Platform デプロイメント上の Red Hat Quay から geo レプリケートされたサイトを削除する

以下の手順を使用すると、Red Hat Quay 管理者は geo レプリケートされたセットアップ内のサイトを削除できます。

前提条件

- OpenShift Container Platform にログインしている。
- 少なくとも 2 つのサイト (例: **usstorage** と **eustorage**) を使用して Red Hat Quay geo レプリケーションを設定している。
- 各サイトに、独自の組織、リポジトリー、およびイメージタグがある。

手順

1. 次のコマンドを実行して、定義されたすべてのサイト間で BLOB を同期します。

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

警告

Red Hat Quay **config.yaml** ファイルからストレージエンジンを削除する前に、定義されているすべてのサイト間ですべての BLOB が同期されていることを確認する **必要があります**。

このコマンドを実行すると、レプリケーションワーカーによって取得されるレプリケーションジョブが作成されます。レプリケートする必要がある Blob がある場合、スクリプトはレプリケートされる Blob の UUID を返します。このコマンドを複数回実行したときに、スクリプトから返された出力が空であっても、レプリケーションプロセスが完了したことを意味するわけではありません。それはレプリケーションのためにキューに入れる Blob が残っていないことを意味します。レプリケーションにかかる割り当て時間は検出された Blob の数によって異なるため、次のステップに進む前に適切に評価してください。

または、Microsoft Azure などのサードパーティーのクラウドツールを使用して、同期ステータスを確認することもできます。

このステップは次に進む前に完了する必要があります。

2. サイト **usstorage** の Red Hat Quay **config.yaml** ファイルで、**eustorage** サイトの **DISTRIBUTED_STORAGE_CONFIG** エントリーを削除します。
3. 次のコマンドを入力して、**Quay** アプリケーション Pod を識別します。

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

出力例

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

4. 以下のコマンドを入力して、**usstorage** Pod でインタラクティブシェルセッションを開きます。

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

5. 次のコマンドを入力して、**eustorage** サイトを完全に削除します。

重要

次の操作は元に戻すことができません。注意して使用してください。

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

出力例

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

第7章 RED HAT QUAY OPERATOR によって管理される RED HAT QUAY のバックアップおよび復元

OpenShift Container Platform で Red Hat Quay Operator によって管理される場合、このセクション内のコンテンツを使用して Red Hat Quay をバックアップおよび復元します。

7.1. RED HAT QUAY のバックアップ

Postgres SQL イメージで提供されるツールまたは独自のバックアップインフラストラクチャーのいずれかを使用して、データベースのバックアップを定期的に実行する必要があります。Red Hat Quay Operator は、PostgreSQL データベースがバックアップされていることを確認しません。

注記

この手順では、Red Hat Quay PostgreSQL データベースのバックアップについて説明します。Clair PostgreSQL データベースのバックアップについては説明しません。厳密に言えば、Clair PostgreSQL データベースは再作成できるため、バックアップする必要はありません。ゼロから再作成する場合は、Red Hat Quay デプロイメント内のすべてのイメージがスキャンされた後、情報が再入力されるまで待つことになります。このダウンタイム中は、セキュリティーレポートを利用できません。

Clair PostgreSQL データベースのバックアップを検討している場合は、データベースのサイズが Red Hat Quay 内に保存されているイメージの数に依存することを考慮する必要があります。データベースが非常に大きくなる可能性があるためです。

この手順では、Operator を使用して OpenShift Container Platform 上の Red Hat Quay のバックアップを作成する方法について説明します。

前提条件

- Red Hat Quay Operator を使用して、OpenShift Container Platform 上に Red Hat Quay を正常にデプロイしている。ステータス条件 **Available** が **true** に設定されている。
- コンポーネント **quay**、**postgres**、および **objectstorage** が **managed: true** に設定されている。
- コンポーネント **clair** が **managed: true** に設定されている場合、コンポーネント **clairpostgres** も **managed: true** に設定されている (Red Hat Quay v3.7 以降)。

注記

デプロイメントに部分的に管理されていないデータベースまたはストレージコンポーネントが含まれ、PostgreSQL または S3 互換オブジェクトストレージの外部サービスを使用している場合、Red Hat Quay デプロイメントを実行するには、サービスプロバイダーまたはベンダーのドキュメントを参照してデータのバックアップを作成してください。このガイドで説明されているツールは、外部 PostgreSQL データベースまたはオブジェクトストレージのバックアップの開始点として参照できます。

7.1.1. Red Hat Quay 設定のバックアップ

Red Hat Quay の設定をバックアップするには、次の手順を実行します。

手順

1. **QuayRegistry** カスタムリソースをエクスポートしてバックアップするには、次のコマンドを入力します。

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

2. 作成される **quayregistry.yaml** を編集し、ステータスセクションおよび以下のメタデータフィールドを削除します。

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

3. 次のコマンドを入力して、マネージドキーのシークレットをバックアップします。

注記

Red Hat Quay 3.7.0 より前のバージョンを実行している場合は、この手順を省略できます。一部のシークレットは Red Hat Quay の初回デプロイ時に自動的に生成されます。これらは **QuayRegistry** リソースの名前空間で、**<quay_registry_name>-quay_registry_managed_secret_keys** というシークレットに保存されます。

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

4. 作成された **managed_secret_keys.yaml** ファイルを編集し、エントリー **metadata.ownerReferences** を削除します。**managed_secret_keys.yaml** ファイルは、以下のようになります。

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: <quayname>_quay_registry_managed_secret_keys>
  namespace: <quay_namespace>
data:
  CONFIG_EDITOR_PW: <redacted>
  DATABASE_SECRET_KEY: <redacted>
  DB_ROOT_PW: <redacted>
  DB_URI: <redacted>
  SECRET_KEY: <redacted>
  SECURITY_SCANNER_V4_PSK: <redacted>
```

data プロパティーの情報はすべて同じままにする必要があります。

5. 次のコマンドを入力して、現在の **Quay** 設定ファイルをリダイレクトします。

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

6. **Quay** Pod 内にマウントされた **/conf/stack/config.yaml** ファイルをバックアップします。

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

7.1.2. Red Hat Quay デプロイメントのスケールダウン

Red Hat Quay デプロイメントをスケールダウンするには、次の手順を実行します。

重要

この手順は、Red Hat Quay デプロイメントの状態の整合性のあるバックアップを作成するために必要になります。PostgreSQL データベースや S3 互換オブジェクトストレージが外部サービスによって提供されるセットアップ (Red Hat Quay Operator で管理されない) を含め、この手順を省略しないでください。

手順

1. Red Hat Quay デプロイメントのバージョンに応じて、次のいずれかのオプションを使用してデプロイメントをスケールダウンします。

   a. **Operator バージョン 3.7 以降:** Red Hat Quay の自動スケーリングを無効にし、Red Hat Quay、ミラーワーカー、および Clair (管理される場合) のレプリカ数をオーバーライドすることで Red Hat Quay デプロイメントを縮小します。**QuayRegistry** リソースは、以下のようになります。

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

(1) Quay、Clair、ミラーリングワーカーの自動スケーリングを無効にします。

(2) データベースおよびオブジェクトストレージにアクセスするコンポーネントのレプリカ数を 0 に設定します。

b. **Operator バージョン 3.6 以前**: まず Red Hat Quay レジストリーをスケールダウンしてからマネージドの Red Hat Quay リソースをスケールダウンして、Red Hat Quay デプロイメントをスケールダウンします。

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

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

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

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

2. **registry-quay-app**、**registry-quay-mirror**、および **registry-clair-app** Pod (どのコンポーネントを Red Hat Quay Operator がマネージするように設定したかにより異なります) が非表示になるまで待機します。以下のコマンドを実行してステータスを確認できます。

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

出力例:

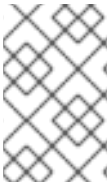
```
$ oc get pod
```

出力例

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

7.1.3. Red Hat Quay マネージドデータベースのバックアップ

Red Hat Quay マネージドデータベースをバックアップするには、次の手順を実行します。



注記

Red Hat Quay デプロイメントが外部のアンマネージド PostgreSQL データベースで設定されている場合は、これらのデータベースの一貫したバックアップを作成する方法についてベンダーのドキュメントを参照してください。

手順

1. Quay PostgreSQL Pod 名を特定します。

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

出力例:

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

2. Quay データベース名を取得します。

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

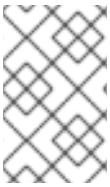
3. バックアップデータベースをダウンロードします。

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

7.1.3.1. Red Hat Quay マネージドオブジェクトストレージのバックアップ

Red Hat Quay マネージドオブジェクトストレージをバックアップするには、次の手順を実行します。このセクションの手順は、以下の設定に適用されます。

- スタンドアロンのマルチクラウドオブジェクトゲートウェイ設定
- OpenShift Data Foundations ストレージでは、Red Hat Quay Operator が ObjectStorageBucketClaim API 経由で S3 オブジェクトストレージバケットをプロビジョニングしている必要があります。



注記

Red Hat Quay デプロイメントが外部 (マネージド外) オブジェクトストレージで設定されている場合は、Quay のストレージバケットのコンテンツのコピーを作成する方法についてベンダーのドキュメントを参照してください。

手順

1. 次のコマンドを入力し、**AWS_ACCESS_KEY_ID** をデコードしてエクスポートします。

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

2. 次のコマンドを入力し、**AWS_SECRET_ACCESS_KEY_ID** をデコードしてエクスポートします。

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

3. 新しいディレクトリーを作成します。

```
$ mkdir blobs
```

注記

AWS コマンドラインユーティリティーの代わりに、rclone または sc3md を使用することもできます。

1. 次のコマンドを入力して、すべての Blob をディレクトリーにコピーします。

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

7.1.4. Red Hat Quay デプロイメントのバックアップのスケーリング

1. Red Hat Quay デプロイメントのバージョンに応じて、次のいずれかのオプションを使用してデプロイメントをスケールアップします。

   a. **Operator バージョン 3.7 以降:** 自動スケーリングを再度有効にし、必要な場合は Quay、ミラーワーカー、および Clair のレプリカオーバーライドを適宜削除して、Red Hat Quay デプロイメントをスケールアップします。**QuayRegistry** リソースは、以下のようになります。

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

   (1) Quay、Clair、ミラーリングワーカーの自動スケーリングの再有効化 (必要に応じて)

   (2) Quay コンポーネントのバックアップをスケーリングするためにレプリカオーバーライドを再削除

   b. **Operator バージョン 3.6 以前の場合:** Red Hat Quay レジストリーをスケールアップして、Red Hat Quay デプロイメントをスケールアップします。

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

2. 次のコマンドを入力して、Red Hat Quay デプロイメントのステータスを確認します。

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

出力例:

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

7.2. RED HAT QUAY の復元

Red Hat Quay オペレーターがデータベースを管理している場合は、次の手順を使用して Red Hat Quay を復元します。これは、Red Hat Quay レジストリーをバックアップした後に実行する必要があります。詳細は、Red Hat Quay のバックアップ を参照してください。

前提条件

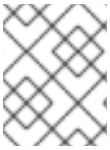
- Red Hat Quay が、Red Hat Quay Operator を使用して OpenShift Container Platform にデプロイされている。
- Red Hat Quay Operator によって管理される Red Hat Quay 設定のバックアップが、Red Hat Quay のバックアップ セクションの手順に従って作成されている。
- Red Hat Quay データベースがバックアップされている。
- Red Hat Quay で使用されるオブジェクトストレージバケットがバックアップされている。
- コンポーネント **quay**、**postgres**、および **objectstorage** が **managed: true** に設定されている。
- コンポーネント **clair** が **managed: true** に設定されている場合、コンポーネント **clairpostgres** も **managed: true** に設定されている (Red Hat Quay v3.7 以降)。
- OpenShift Container Platform クラスターのターゲット namespace で、Red Hat Quay Operator に管理される Red Hat Quay デプロイメントを実行していない。

注記

デプロイメントに部分的に管理されていないデータベースまたはストレージコンポーネントが含まれ、PostgreSQL または S3 互換オブジェクトストレージの外部サービスを使用している場合、Red Hat Quay デプロイメントを実行するには、サービスプロバイダーまたはベンダーのドキュメントを参照して、Red Hat Quay を復元する前にバックアップからデータを復元してください。

7.2.1. バックアップからの Red Hat Quay およびその設定の復元

Red Hat Quay とその設定ファイルをバックアップから復元するには、次の手順を実行します。



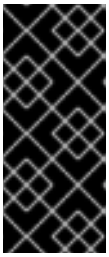
注記

これらの手順では、Red Hat Quay のバックアップ ガイドのプロセスに従い、同じ名前のバックアップファイルを作成していることを前提としています。

手順

1. 次のコマンドを入力して、バックアップされた Red Hat Quay 設定を復元します。

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



重要

エラー **Error from server (AlreadyExists): error when creating "./config-bundle.yaml": secrets "config-bundle-secret" already exists** が発生した場合は、**$ oc delete Secret config-bundle-secret -n <quay-namespace>** を使用して既存リソースを削除し、**$ oc create -f ./config-bundle.yaml** で再作成する必要があります。

2. 次のコマンドを入力して、生成されたキーをバックアップから復元します。

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

3. **QuayRegistry** カスタムリソースを復元します。

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

4. Red Hat Quay デプロイメントのステータスを確認し、これが利用可能になるまで待機します。

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

7.2.2. Red Hat Quay デプロイメントのスケールダウン

Red Hat Quay デプロイメントをスケールダウンするには、次の手順を実行します。

手順

1. Red Hat Quay デプロイメントのバージョンに応じて、次のいずれかのオプションを使用してデプロイメントをスケールダウンします。

a. **Operator バージョン 3.7 以降:** Red Hat Quay の自動スケーリングを無効にし、Red Hat Quay、ミラーワーカー、および Clair (マネージドの場合) のレプリカ数をオーバーライドすることで Quay デプロイメントを縮小します。**QuayRegistry** リソースは、以下のようになります。

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

(1) Quay、Clair、ミラーリングワーカーの自動スケーリングを無効にします。

(2) データベースおよびオブジェクトストレージにアクセスするコンポーネントのレプリカ数を 0 に設定します。

b. **Operator バージョン 3.6 以前:** まず Red Hat Quay レジストリーをスケールダウンしてからマネージドの Red Hat Quay リソースをスケールダウンして、Red Hat Quay デプロイメントをスケールダウンします。

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

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

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

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

2. **registry-quay-app**、**registry-quay-mirror**、および **registry-clair-app** Pod (どのコンポーネントを Red Hat Quay Operator がマネージするように設定したかにより異なります) が非表示になるまで待機します。以下のコマンドを実行してステータスを確認できます。

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

出力例:

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

7.2.3. Red Hat Quay データベースの復元

Red Hat Quay データベースを復元するには、次の手順を実行します。

手順

1. 次のコマンドを入力して、**Quay** データベース Pod を識別します。

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

出力例:

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

2. ローカル環境および Pod にコピーして、バックアップをアップロードします。

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

3. 次のコマンドを入力して、データベースへのリモートターミナルを開きます。

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

4. 次のコマンドを実行して psql を入力します。

```
bash-4.4$ psql
```

5. 以下のコマンドを実行してデータベースをリスト表示できます。

```
postgres=# \l
```

出力例

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

6. 次のコマンドを入力してデータベースを削除します。

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

出力例

```
DROP DATABASE
```

7. postgres CLI を終了して bash-4.4 を再入力します。

```
\q
```

8. PostgreSQL データベースをバックアップデータベースにリダイレクトします。

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

9. 次のコマンドを入力して bash を終了します。

```
sh-4.4$ exit
```

7.2.4. Red Hat Quay オブジェクトストレージデータの復元

Red Hat Quay オブジェクトストレージデータを復元するには、次の手順を実行します。

手順

1. 次のコマンドを入力して、**AWS_ACCESS_KEY_ID** をエクスポートします。

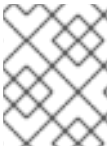
```
$ export AWS_ACCESS_KEY_ID=$(oc get secret -l app=noobaa -n <quay-namespace>  -o
jsonpath='{.items[0].data.AWS_ACCESS_KEY_ID}' |base64 -d)
```

2. 次のコマンドを入力して、**AWS_SECRET_ACCESS_KEY** をエクスポートします。

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

3. 以下のコマンドを実行して、すべての Blob をバケットにアップロードします。

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



注記

AWS コマンドラインユーティリティーの代わりに、rclone または sc3md を使用することもできます。

7.2.5. Red Hat Quay デプロイメントのスケールアップ

1. Red Hat Quay デプロイメントのバージョンに応じて、次のいずれかのオプションを使用してデプロイメントをスケールアップします。

a. **Operator バージョン 3.7 以降:** 自動スケーリングを再度有効にし、必要な場合は Quay、ミラーワーカー、および Clair のレプリカオーバーライドを適宜削除して、Red Hat Quay デプロイメントをスケールアップします。**QuayRegistry** リソースは、以下のようになります。

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

(1) Red Hat Quay、Clair、ミラーリングワーカーの自動スケーリングの再有効化 (必要に応じて)

(2) Red Hat Quay コンポーネントのバックアップをスケーリングするためにレプリカオーバーライドを再削除

b. **Operator バージョン 3.6 以前の場合:** Red Hat Quay レジストリーを再度スケールアップして、Red Hat Quay デプロイメントをスケールアップします。

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

2. Red Hat Quay デプロイメントのステータスを確認します。

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

出力例:

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

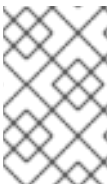
```
reason: HealthChecksPassing
status: 'True'
type: Available
```

第8章 ボリュームサイズのオーバーライド

マネージドコンポーネントにプロビジョニングされるストレージリソースの希望のサイズを指定できます。Clair および PostgreSQL データベースのデフォルトサイズは **50Gi** です。パフォーマンス上の理由がある場合や、ストレージバックエンドにサイズ変更機能がない場合など、十分な容量を事前に選択できるようになりました。

以下の例では、Clair および Quay PostgreSQL データベースのボリュームサイズは **70Gi** に設定されています。

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



注記

clairpostgres コンポーネントのボリュームサイズはオーバーライドできません。これは既知の問題であり、Red Hat Quay の将来のバージョンで修正される予定です。(PROJQUAY-4301)

第9章 コンテナーセキュリティー OPERATOR での POD イメージのスキャン

Container Security Operator (CSO) は、OpenShift Container Platform およびその他の Kubernetes プラットフォームで利用可能な Clair セキュリティースキャナーのアドオンです。CSO を使用すると、ユーザーはアクティブな Pod に関連付けられているコンテナーイメージをスキャンして、既知の脆弱性を見つけることができます。

注記

CSO は、Red Hat Quay と Clair なしでは機能しません。

Container Security Operator (CSO) には、次の機能が含まれています。

- 指定された namespace またはすべての namespace の Pod に関連付けられたコンテナーを監視します。
- コンテナーのソースであるコンテナーレジストリーにクエリーを実行して、脆弱性情報を取得します (イメージのレジストリーが、Clair スキャンを使用する Red Hat Quay レジストリーなどのイメージスキャンをサポートしている場合)。
- Kubernetes API の **ImageManifestVuln** オブジェクトを使用して脆弱性を公開します。

注記

CSO を Kubernetes にインストールする手順を見るには、Container Security OperatorHub.io ページから **Install** ボタンを選択します。

9.1. OPENSHIFT CONTAINER PLATFORM での CONTAINER SECURITY OPERATOR のダウンロードおよび実行

Container Security Operator (CSO) をダウンロードするには、次の手順を使用します。

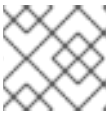
注記

次の手順では、CSO を **marketplace-operators** namespace にインストールします。これにより、OpenShift Container Platform クラスターのすべての namespace で CSO を使用できるようになります。

手順

1. OpenShift Container Platform コンソールページで、**Operators → OperatorHub** を選択し、**Container Security Operator** を検索します。
2. Container Security Operator を選択し、**Install** を選択して **Create Operator Subscription** ページに移動します。
3. 設定 (デフォルトでは、すべての名前空間と自動承認戦略) を確認し、**Subcription** を選択します。しばらくすると、**Installed Operators** 画面に **Container Security** が表示されます。
4. オプション: カスタム証明書を CSO に追加できます。以下の例では、現在のディレクトリーに **quay.crt** という名前の証明書を作成します。次に、次のコマンドを実行して証明書を CSO に追加します。

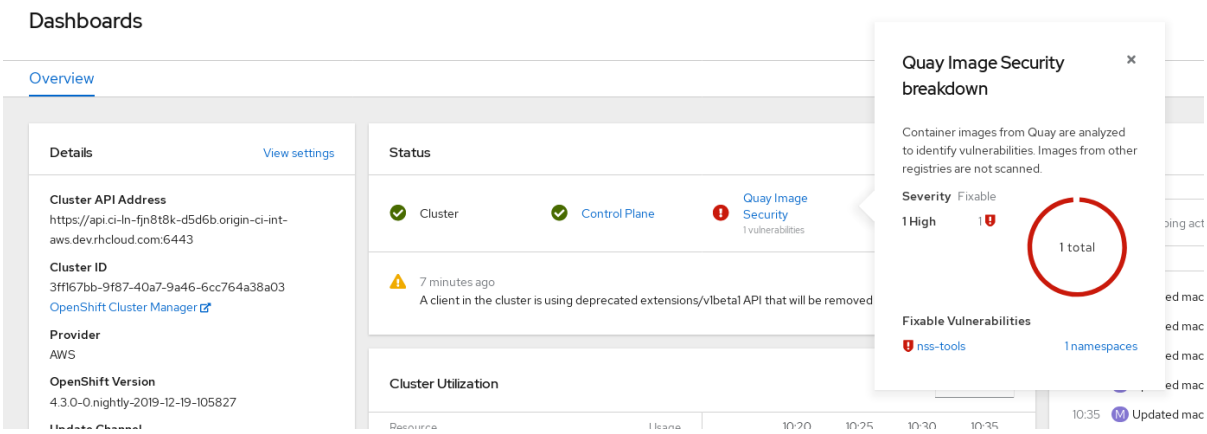
```
$ oc create secret generic container-security-operator-extra-certs --from-file=quay.crt -n
openshift-operators
```

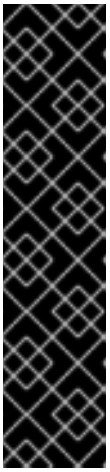


注記

新しい証明書を有効にするには、Operator Pod を再起動する必要があります。

5. **Home → Dashboard** に移動します。**Image Security** へのリンクが status セクションに表示され、これまでに見つかった脆弱性の数の一覧が表示されます。次の図に示すように、リンクを選択するとセキュリティーの内訳が表示されます。



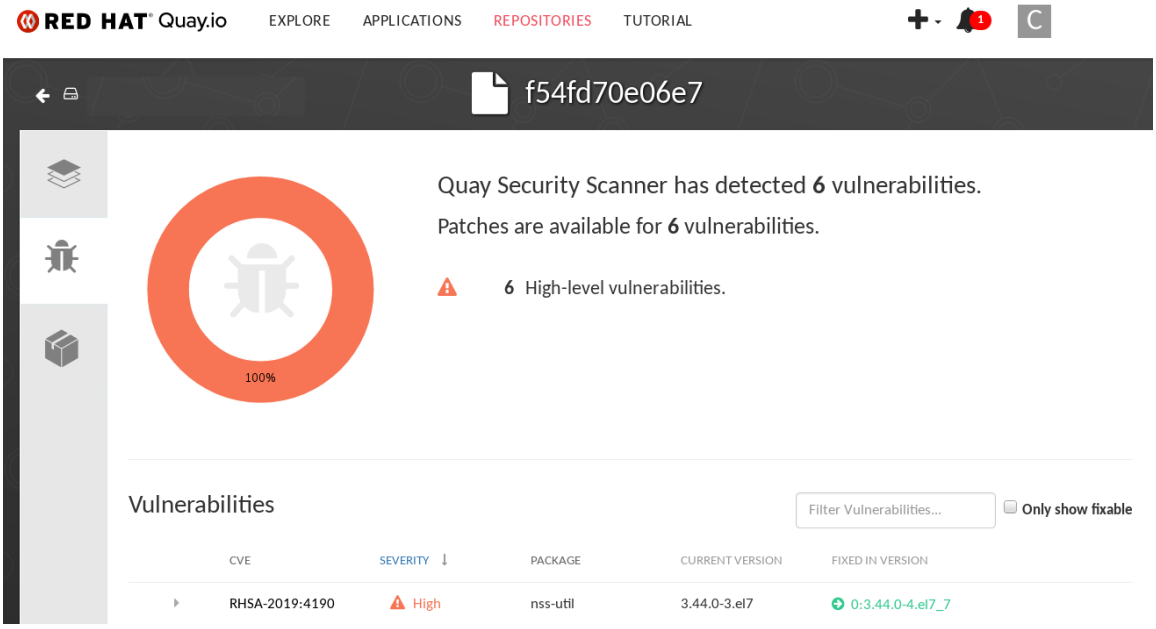


重要

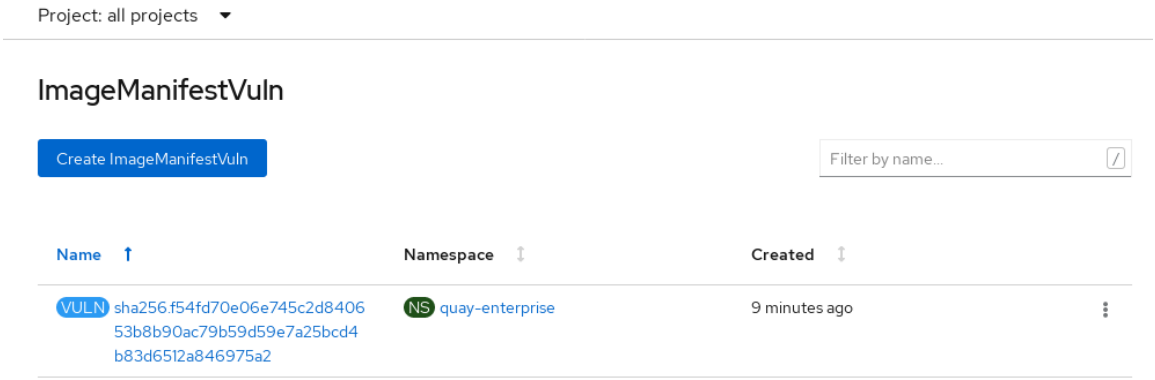
Container Security Operator は現在、Red Hat セキュリティーアドバイザリーの壊れたリンクを提供しています。たとえば、リンク **https://access.redhat.com/errata/RHSA-2023:1842%20https://access.redhat.com/security/cve/CVE-2023-23916** が提供される場合があります。URL 内の **%20** はスペース文字を表しますが、現時点では 2 つの URL が 1 つの不完全な URL に結合されます (例: **https://access.redhat.com/errata/RHSA-2023:1842** と **https://access.redhat.com/security/cve/CVE-2023-23916**。一時的な回避策として、各 URL をブラウザーにコピーして、適切なページに移動できます。これは既知の問題であり、Red Hat Quay の今後のバージョンで修正される予定です。

6. この時点で、検出された脆弱性をフォローするために以下の 2 つのいずれかの操作を実行できます。

   a. 脆弱性へのリンクを選択します。コンテナーのレジストリー、Red Hat Quay、またはコンテナーを取得したその他のレジストリーに移動し、脆弱性に関する情報が表示されます。以下の図は、Quay.io レジストリーから検出された脆弱性の例を示しています。

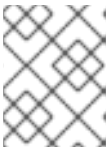


b. namespaces リンクを選択し、**ImageManifestVuln** 画面に移動します。ここでは、選択されたイメージの名前、およびイメージが実行されているすべての namespace を確認できます。以下の図は、特定の脆弱なイメージが 2 つの namespace で実行されていることを示しています。



この手順を実行すると、どのイメージに脆弱性があるか、それらの脆弱性を修正するために何をしなければならないか、およびイメージが実行されたすべての名前空間がわかります。これを知っていると、次のアクションを実行できます。

- イメージを実行しているユーザーに、脆弱性を修正する必要があることを警告します。
- デプロイメント、またはイメージが含まれる Pod を開始したオブジェクトを削除して、イメージの実行を停止します。



注記

Pod を削除した場合、ダッシュボードで脆弱性がリセットされるまでに数分かかる場合があります。

9.2. CLI からイメージの脆弱性を問い合わせ

コマンドラインインターフェイス (CLI) からイメージの脆弱性をクエリーするには、次の手順を使用します。

手順

1. 次のコマンドを入力して、検出された脆弱性をクエリーします。

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

出力例

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

2. オプション: 特定の脆弱性の詳細を表示するには、特定の脆弱性とその名前空間を特定し、**oc description** コマンドを使用します。以下の例は、イメージに脆弱性のある RPM パッケージが含まれるアクティブなコンテナーを示しています。

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

出力例

```
Name:         sha256.ac50e3752...
Namespace:    quay-enterprise
...
Spec:
  Features:
    Name:            nss-util
    Namespace Name:  centos:7
    Version:         3.44.0-3.el7
    Versionformat:   rpm
    Vulnerabilities:
      Description: Network Security Services (NSS) is a set of libraries...
```

第10章 AWS STS FOR RED HAT QUAY の設定

Amazon Web Services (AWS) Security Token Service (STS) のサポートは、スタンドアロンの Red Hat Quay デプロイメントと OpenShift Container Platform 上の Red Hat Quay で利用できます。AWS STS は、AWS アイデンティティーおよびアクセス管理 (IAM) ユーザー、認証したユーザー、または **フェデレーションユーザー** に対して、一時的な制限付き権限の認証情報をリクエストするための Web サービスです。この機能は、Amazon S3 をオブジェクトストレージとして使用するクラスターに役立ち、Red Hat Quay は STS プロトコルを使用して Amazon S3 で認証できます。これにより、クラスターの全体的なセキュリティーが強化され、機密データへのアクセスが適切に認証および承認される際に役立ちます。

AWS STS の設定は、AWS IAM ユーザーの作成、S3 ロールの作成、適切なリソースを含めるための Red Hat Quay **config.yaml** ファイルの設定を必要とする複数の手順からなるプロセスです。

AWS STS for Red Hat Quay を設定するには、次の手順に従います。

10.1. IAM ユーザーの作成

IAM ユーザーを作成するには、次の手順を使用します。

手順

1. Amazon Web Services (AWS) コンソールにログインし、アイデンティティーおよびアクセス管理 (IAM) コンソールに移動します。
2. ナビゲーションウィンドウの **Access management** で **Users** をクリックします。
3. **Create User** をクリックし、以下の情報を入力します。
    a. 有効なユーザー名 (例: **quay-user**) を入力します。
    b. **Permissions options** の場合は、**Add user to group** をクリックします。
4. **review and create** ページで、**Create user** をクリックします。**Users** ページにリダイレクトされます。
5. ユーザー名 (例: **quay-user**) をクリックします。
6. ユーザーの ARN をコピーします (例: **arn:aws:iam::123492922789:user/quay-user**)。
7. 同じページで、 **Security credentials** タブをクリックします。
8. **Access keys** に移動します。
9. **Create access key** をクリックします。
10. **Access key best practices & alternatives**ページで、**Command Line Interface (CLI)** をクリックしてから、確認ボックスをオンにします。**Next** をクリックします。
11. オプション: **Set description tag - optional**ページで、説明を入力します。
12. **Create access key** をクリックします。
13. アクセスキーとシークレットアクセスキーをコピーして保存します。

重要

シークレットアクセスキーを表示またはダウンロードできるのは、このときだけです。後でこれを実行することはできません。ただし、新しいアクセスキーはいつでも作成できます。

14. **Done** をクリックします。

10.2. S3 ロールの作成

次の手順を使用して、AWS STS の S3 ロールを作成します。

前提条件

- IAM ユーザーが作成され、アクセスキーとシークレットアクセスキーが保存されている。

手順

1. まだ移動していない場合は、**Dashboard** をクリックして IAM ダッシュボードに移動します。
2. ナビゲーションペインで、**Access management** の下の **Roles** をクリックします。
3. **Create role** をクリックします。
   - **Custom Trust Policy** をクリックすると、編集可能な JSON ポリシーが表示されます。デフォルトでは、次の情報が表示されます。

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

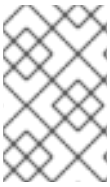
4. **Principal** 設定フィールドに、AWS ARN 情報を追加します。以下に例を示します。

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

5. **Next** をクリックします。
6. **Add permissions** ページで、検索ボックスに **AmazonS3FullAccess** と入力します。チェックボックスをオンにしてそのポリシーを S3 ロールに追加し、**Next** をクリックします。
7. **Name, review, and create**ページで、次の情報を入力します。
   a. ロール名を入力します (例: **example-role**)。
   b. オプション: 説明を追加します。
8. **Create role** ボタンをクリックします。**Roles** ページに移動します。**Role name** の下に、新しく作成された S3 が利用可能になっているはずです。

10.3. AWS STS を使用するための OPENSHIFT CONTAINER PLATFORM 上の RED HAT QUAY の設定

AWS STS を使用するために、OpenShift Container Platform 上の Red Hat Quay の**config.yaml** ファイルを編集するには、次の手順に従います。



注記

OpenShift Container Platform UI を使用する代わりに、OpenShift Container Platform 上の Red Hat Quay の **config.yaml** ファイルを直接編集して再デプロイすることもできます。

前提条件

- Role ARN が設定されている。
- ユーザーアクセスキーが生成されている。
- ユーザーシークレットキーが生成されている。

手順

1. OpenShift Container Platform デプロイメントの **Home** ページで、**Operators → Installed Operators** をクリックします。
2. **Red Hat Quay** をクリックします。
3. **Quay Registry** をクリックし、Red Hat Quay レジストリーの名前をクリックします。
4. **Config Bundle Secret** の下で、レジストリー設定バンドルの名前 (例: **quay-registry-config-bundle-qet56**) をクリックします。
5. 設定バンドルのページで、 **Actions** をクリックしてドロップダウンメニューを表示します。次に、**Edit Secret** をクリックします。
6. **config.yaml** ファイルの **DISTRIBUTED_STORAGE_CONFIG** フィールドを次の情報で更新します。

```
# ...
DISTRIBUTED_STORAGE_CONFIG:
  default:
```

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

1. AWS STS を設定するときに必要な、一意の Amazon Resource Name (ARN)
2. S3 バケットの名前。
3. データのストレージパス。通常は **/datastorage** です。
4. AWS STS を設定するときに必要な、生成された AWS S3 ユーザーアクセスキー。
5. AWS STS を設定するときに必要な、生成された AWS S3 ユーザーシークレットキー。

7. **Save** をクリックします。

検証

1. リポジトリーにプッシュされるサンプルイメージ (例: **busybox**) にタグを付けます。以下に例を示します。

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

2. 次のコマンドを実行して、サンプルイメージをプッシュします。

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

3. Red Hat Quay registry → **Tags** でイメージをプッシュした Organization に移動して、プッシュが成功したことを確認します。
4. Amazon Web Services (AWS) コンソールに移動し、s3 バケットを探します。
5. s3 バケットの名前をクリックします。
6. **Objects** ページで、**datastorage/** をクリックします。
7. **datastorage/** ページには、次のリソースが表示されます。
   - **sha256/**
   - **uploads/**
     これらのリソースは、プッシュが成功し、AWS STS が適切に設定されていることを示しています。

第11章 QUAY BRIDGE OPERATOR を使用した OPENSHIFT CONTAINER PLATFORM の RED HAT QUAY への統合

Quay Bridge Operator は、統合された OpenShift Container Platform レジストリーの機能を新しい Red Hat Quay レジストリーに複製します。Quay Bridge Operator を使用すると、Red Hat Quay の統合コンテナーレジストリーを OpenShift Container Platform レジストリーに置き換えることができます。

Quay Bridge Operator で有効になる機能は次のとおりです。

- OpenShift Container Platform namespace を Red Hat Quay 組織として同期。
- 各デフォルト namespace サービスアカウント用のロボットアカウントの作成。
- 作成された各ロボットアカウントのシークレットの作成 (各ロボットシークレットを **Mountable Image Pull Secret** としてサービスアカウントに関連付ける)。
- OpenShift Container Platform イメージストリームを Red Hat Quay リポジトリーとして同期。
- Red Hat Quay への出力に ImageStream を使用した新規ビルドの自動再作成。
- ビルド完了後の image stream タグの自動インポート。

以下の手順を使用すると、Red Hat Quay クラスターと OpenShift Container Platform クラスター間の双方向通信を有効にすることができます。

11.1. QUAY BRIDGE OPERATOR 用の RED HAT QUAY のセットアップ

この手順では、専用の Red Hat Quay 組織を作成し、その組織内で作成された新しいアプリケーションから、OpenShift Container Platform の OpenShift Container Platform で使用される OAuth トークンを生成します。

手順

1. WebUI を介して Red Hat Quay にログインします。
2. 外部アプリケーションを設定する組織を選択します。
3. ナビゲーションペインで、**Applications** を選択します。
4. **Create New Application** を選択し、新規アプリケーションの名前を入力します (例: **openshift**)。
5. **OAuth Applications** ページで、アプリケーション (**openshift** など) を選択します。
6. ナビゲーションペインで、**Generate Token** を選択します。
7. 次のフィールドを選択します。
   - **Administer Organization**
   - **Administer Repositories**
   - **Create Repositories**
   - **View all visible repositories**

- **Read/Write to any accessible repositories**
- **Administer User**
- **Read User Information**

8. 割り当てられた権限を確認します。
9. **Authorize Application** を選択し、**Authorize Application** を選択して認証を確認します。
10. 生成されたアクセストークンを保存します。

重要

Red Hat Quay はトークン管理を提供しません。トークンをリスト表示したり、トークンを削除したり、トークンを変更したりすることはできません。生成されたアクセストークンは 1 回だけ表示され、ページを閉じた後に再取得することはできません。

11.2. OPENSHIFT CONTAINER PLATFORM への QUAY BRIDGE OPERATOR のインストール

この手順では、Quay Bridge Operator を OpenShift Container Platform にインストールします。

前提条件

- Red Hat Quay をセットアップし、アクセストークンを取得した。
- クラスター管理者権限のある OpenShift Container Platform 4.6 の環境が準備できている。

手順

1. Web コンソールの **Administrator** パースペクティブを開き、ナビゲーションペインで **Operator → OperatorHub** に移動します。
2. **Quay Bridge Operator** を検索し、**Quay Bridge Operator** のタイトルをクリックして、**Install** をクリックします。
3. インストールするバージョン (たとえば、**stable-3.7**) を選択し、**Install** をクリックします。
4. インストールが完了したら、**View Operator** をクリックして、Quay Bridge Operator の **Details** ページに移動します。または、**Installed Operators → Red Hat Quay Bridge Operator** をクリックして、**Details** ページに移動することもできます。

11.3. OAUTH トークンの OPENSHIFT CONTAINER PLATFORM シークレットの作成

この手順では、以前に取得したアクセストークンを追加して、Red Hat Quay デプロイメントと通信します。アクセストークンは、OpenShift Container Platform 内にシークレットとして保存されます。

前提条件

- Red Hat Quay をセットアップし、アクセストークンを取得している。

- OpenShift Container Platform に OpenShift Container Platform をデプロイしている。
- クラスター管理者権限のある OpenShift Container Platform 4.6 の環境が準備できている。
- OpenShift CLI (oc) がインストールされている。

手順

- **openshift-operators** namespace にアクセストークンを含むシークレットを作成します。

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

11.4. QUAYINTEGRATION カスタムリソースの作成

この手順では、**QuayIntegration** カスタムリソースを作成します。これは、Web コンソールまたはコマンドラインから実行できます。

前提条件

- Red Hat Quay をセットアップし、アクセストークンを取得している。
- OpenShift Container Platform に OpenShift Container Platform をデプロイしている。
- クラスター管理者権限のある OpenShift Container Platform 4.6 の環境が準備できている。
- オプション: OpenShift CLI (oc) をインストールしている。

11.4.1. オプション: CLI を使用して QuayIntegration カスタムリソースを作成する

この手順に従って、コマンドラインを使用して **QuayIntegration** カスタムリソースを作成します。

手順

1. **quay-integration.yaml** を作成します:

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

2. **QuayIntegration** カスタムリソースの最小限のデプロイメントには、次の設定を使用します。

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

1. clusterID の値は、エコシステム全体で一意である必要があります。この値は必須であり、デフォルトは **openshift** です。
2. **credentialsSecret** プロパティーは、以前に作成されたトークンが含まれるシークレットの namespace および名前を参照します。
3. **QUAY_URL** を Red Hat Quay インスタンスのホスト名に置き換えます。
4. Red Hat Quay が自己署名証明書を使用している場合は、プロパティーを **insecureRegistry: true** に設定します。

すべての設定フィールドのリストは、QuayIntegration configuration fields を参照してください。

1. **QuayIntegration** カスタムリソースを作成します。

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

11.4.2. オプション:Web コンソールを使用して QuayIntegration カスタムリソースを作成する

この手順に従って、Web コンソールを使用して **QuayIntegration** カスタムリソースを作成します。

手順

1. Web コンソールの **Administrator** パースペクティブを開き、**Operators → Installed Operators** に移動します。
2. **Red Hat Quay Bridge Operator**をクリックします。
3. Quay Bridge Operator の **Details** ページで、**Quay Integration** API カードの **Create Instance** をクリックします。
4. **Create QuayIntegration** ページで、**Form view** または **YAML view** のいずれかに以下の必須情報を入力します。
   - **Name**: **QuayIntegration** カスタムリソースオブジェクトを参照する名前。
   - **Cluster ID**: このクラスターに関連付けられている ID。この値は、エコシステム全体で一意である必要があります。指定しない場合、デフォルトで **openshift** になります。
   - **Credentials secret** 以前に作成されたトークンを含むシークレットの名前空間と名前を参照します。
   - **Quay hostname**: Quay レジストリーのホスト名。

すべての設定フィールドのリストは、QuayIntegration 設定フィールド を参照してください。

QuayIntegration カスタムリソースが作成されると、OpenShift Container Platform クラスターが Red Hat Quay インスタンスにリンクされます。Red Hat Quay レジストリー内の組織は、OpenShift Container Platform 環境の関連する namespace 用に作成する必要があります。

11.5. QUAY BRIDGE OPERATOR の使用

Quay Bridge Operator を使用するには、次の手順を実行します。

前提条件

- Red Hat Quay Operator をインストールしている。
- クラスター管理者として OpenShift Container Platform にログインしている。
- Red Hat Quay レジストリーにログインしている。
- Quay Bridge Operator をインストールしている。
- **QuayIntegration** カスタムリソースを設定している。

手順

1. 次のコマンドを入力して、**e2e-demo** という新しい OpenShift Container Platform プロジェクトを作成します。

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

2. 新しいプロジェクトを作成すると、Red Hat Quay に新しい組織が作成されます。Red Hat Quay レジストリーに移動し、**openshift_e2e-demo** という名前の新しい組織が作成されたことを確認します。

注記

組織の **openshift** 値は、**QuayIntegration** リソースのクラスター ID で別の値を使用している場合、異なる可能性があります。

3. Red Hat Quay UI で、新しい組織の名前 (例: **openshift_e2e-demo)** をクリックします。
4. ナビゲーションペインで **Robot Accounts** をクリックします。新しいプロジェクトの一部として、次のロボットアカウントが作成されているはずです。
   - **openshift_e2e-demo+deployer**
   - **openshift_e2e-demo+default**
   - **openshift_e2e-demo+builder**
5. 次のコマンドを入力して、該当するロボットアカウントに関連する Docker 設定を含む 3 つのシークレットが作成されたことを確認します。

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

出力例

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

6. 次のコマンドを入力して、**builder** ServiceAccount (SA) に関する詳細情報 (そのシークレット、トークンの有効期限、関連するロールとロールバインディングなど) を表示します。これにより、プロジェクトが Quay Bridge Operator を介して統合されることを確認します。

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

出力例

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

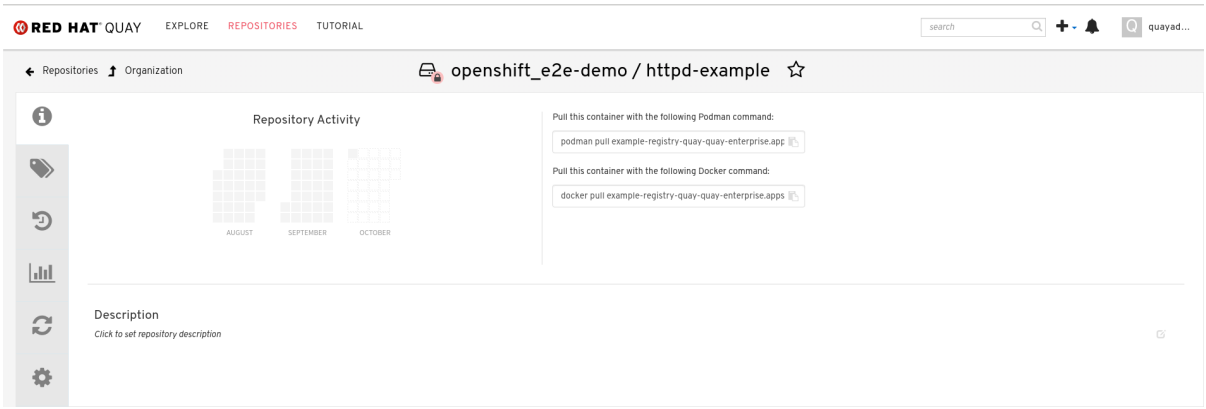
7. 次のコマンドを入力して、**httpd-template** という新しいアプリケーションを作成してデプロイします。

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

出力例

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

このコマンドを実行すると、**BuildConfig**、**ImageStream**、**Service**、**Route**、および **DeploymentConfig** リソースが作成されます。**ImageStream** リソースが作成されると、関連するリポジトリーが Red Hat Quay に作成されます。以下に例を示します。



8. **BuildConfig** の **ImageChangeTrigger** は、**openshift** namespace にある Apache HTTPD イメージが解決されると、新しいビルドをトリガーします。新しいビルドが作成されると、**MutatingWebhookConfiguration** が Red Hat Quay を参照するように出力を自動的に書き換えます。次のコマンドを実行してビルドの出力フィールドをクエリーすることで、ビルドが完了したことを確認できます。

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

出力例

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

9. Red Hat Quay UI で、**openshift_e2e-demo** 組織に移動し、**httpd-example** リポジトリーを選択します。

10. ナビゲーションペインで **Tags** をクリックし、**latest** タグが正常にプッシュされたことを確認します。

11. 次のコマンドを入力して、最新のタグが解決されたことを確認します。

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

出力例

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

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

12. **ImageStream** が解決された後、新しいデプロイメントがトリガーされます。次のコマンドを入力して URL 出力を生成します。

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

出力例

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

13. URL に移動します。サンプル Web ページが表示されれば、デプロイメントは成功しています。

14. 次のコマンドを入力してリソースを削除し、Red Hat Quay リポジトリーをクリーンアップします。

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

注記

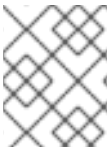
このコマンドは、プロジェクトリソースが削除されるまで待機します。上記のコマンドに **--wait=false** を追加すると、待機を回避できます。

15. コマンドが完了したら、Red Hat Quay リポジトリーに移動し、**openshift_e2e-demo** 組織が使用できなくなっていることを確認します。

関連情報

- ベストプラクティスとして、クライアントとイメージレジストリー間のすべての通信を、セキュアな手段により容易化することが推奨されています。通信には、当事者間の証明書信頼と HTTPS/TLS を利用してください。Red Hat Quay はセキュアでない設定を提供するように設定することもできますが、適切な証明書をサーバー上で利用し、クライアント上で設定することを推奨します。コンテナーランタイムレベルでの証明書の追加と管理については、OpenShift Container Platform のドキュメント を参照してください。

第12章 OPENSHIFT CONTAINER PLATFORM 上の RED HAT QUAY への IPV6 のデプロイ



注記

現在、OpenShift Container Platform 上の Red Hat Quay への IPv6 のデプロイは、IBM Power および IBM Z ではサポートされていません。

OpenShift Container Platform デプロイメント上の Red Hat Quay は、通信事業者の環境やエッジ環境など、IPv6 のみをサポートする場所でサービスを提供できるようになりました。

既知の制限のリストは、IPv6 の制限 を参照してください。

12.1. IPV6 プロトコルファミリーの有効化

Red Hat Quay デプロイメントで IPv6 のサポートを有効にするには、次の手順を使用します。

前提条件

- Red Hat Quay を 3.8 に更新している。
- ホストとコンテナーソフトウェアプラットフォーム (Docker、Podman) を、IPv6 をサポートするように設定している。

手順

1. デプロイメントの **config.yaml** ファイルに **FEATURE_LISTEN_IP_VERSION** パラメーターを追加し、**IPv6** に設定します。次に例を示します。

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

2. Red Hat Quay デプロイメントを起動または再起動します。
3. 次のコマンドを入力して、デプロイが IPv6 をリッスンしていることを確認します。

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

デプロイメントの **config.yaml** で IPv6 を有効にすると、環境が IPv6 を使用するように設定され、IPv6 およびデュアルスタックの制限 によって妨げられない限り、Red Hat Quay のすべての機能を通常どおり使用できます。

警告

環境が IPv4 に設定されていても、**FEATURE_LISTEN_IP_VERSION** 設定フィールドが **IPv6** に設定されていると、Red Hat Quay はデプロイに失敗します。

12.2. IPV6 の制限

- 現在、一般的な Microsoft Azure Blob Storage 設定を使用して Red Hat Quay デプロイメントを設定しようとしても、IPv6 シングルスタック環境では機能しません。Microsoft Azure Blob Storage のエンドポイントは IPv6 をサポートしていないため、この問題に対する回避策はありません。
  詳細は、PROJQUAY-4433 を参照してください。

- 現在、Amazon S3 CloudFront を使用して Red Hat Quay デプロイメントを設定しようとしても、IPv6 シングルスタック環境では機能しません。Amazon S3 CloudFront のエンドポイントは IPv6 をサポートしていないため、この問題に対する回避策はありません。
  詳細は、PROJQUAY-4470 を参照してください。

- 現在、Red Hat Quay が IPv6 シングルスタック環境にデプロイされている場合、Red Hat OpenShift Data Foundations はサポートされません。そのため、Red Hat OpenShift Data Foundation は IPv6 環境で使用できません。この制限は、OpenShift Data Foundations の今後のバージョンで修正される予定です。

- 現在、デュアルスタック (IPv4 および IPv6) のサポートは、Red Hat Quay OpenShift Container Platform デプロイメントでは機能しません。Red Hat Quay 3.8 が OpenShift Container Platform にデプロイされ、デュアルスタックサポートが有効になっている場合は、Red Hat Quay Operator によって生成される Quay Route が IPv4 アドレスのみを生成し、IPv6 アドレスは生成されません。その結果、IPv6 アドレスを持つクライアントは、OpenShift Container Platform 上の Red Hat Quay アプリケーションにアクセスできません。この制限は、OpenShift Container Platform の今後のバージョンで修正される予定です。

第13章 RED HAT QUAY が KUBERNETES にデプロイされている場合のカスタム SSL/TLS 証明書の追加

Kubernetes にデプロイすると、Red Hat Quay は config アセットを保存するボリュームとしてシークレットにマウントします。現在、これによりスーパーユーザーパネルの証明書のアップロード機能が中断されます。

一時的な回避策として、Red Hat Quay の **デプロイ後** に、**base64** でエンコードされた証明書をシークレットに追加できます。

Red Hat Quay が Kubernetes にデプロイされている場合は、次の手順を使用してカスタム SSL/TLS 証明書を追加します。

前提条件

- Red Hat Quay がデプロイされている。
- カスタムの **ca.crt** ファイルがある。

手順

1. 次のコマンドを入力して、SSL/TLS 証明書の内容を Base64 でエンコードします。

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

出力例

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

2. 次の **kubectl** コマンドを入力して、**quay-enterprise-config-secret** ファイルを編集します。

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

3. 証明書のエントリーを追加し、**base64** でエンコードされた完全なストリンガーをエントリーの下に貼り付けます。以下に例を示します。

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

4. **kubectl delete** コマンドを使用して、すべての Red Hat Quay Pod を削除します。以下に例を示します。

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

その後、Red Hat Quay デプロイメントにより、Pod を新しい証明書データに置き換えるスケジュールが自動的に設定されます。

第14章 RED HAT QUAY OPERATOR のアップグレードの概要

注記

現在、Red Hat Quay Operator のアップグレードは、IBM Power および IBM Z ではサポートされていません。

Red Hat Quay Operator は、**シンクロナイズドバージョニング** スキームに従います。つまり、Red Hat Operator の各バージョンは Quay とその管理するコンポーネントに関連付けられます。**QuayRegistry** カスタムリソースには、Red Hat Quay が **deploy** するバージョンを設定するフィールドはありません。Operator は、すべてのコンポーネントを 1 つのバージョンのみデプロイできます。このスキームは、すべてのコンポーネントが適切に連携し、Kubernetes で Red Hat Quay の多数に渡るバージョンのライフサイクルを管理する方法を把握する必要がある Operator の複雑性を軽減するために、選択されました。

14.1. OPERATOR LIFECYCLE MANAGER

Red Hat Quay Operator は Operator Lifecycle Manager (OLM) を使用してインストールし、アップグレードする必要があります。デフォルトの **approvalStrategy: Automatic** で **Subscription** を作成する場合、OLM は新規バージョンが利用可能になると常に Red Hat Quay Operator を自動的にアップグレードします。

警告

Red Hat Quay Operator が Operator Lifecycle Manager によってインストールされている場合、自動または手動のアップグレードをサポートするように設定されていることがあります。このオプションは、インストール時に Red Hat Quay Operator の **OperatorHub** ページに表示されます。これは、Red Hat Quay Operator **Subscription** オブジェクトの **ApprovalStrategy** フィールドでも確認できます。**Automatic** を選択すると、新規 Operator バージョンがリリースされるたびに Red Hat Quay Operator が自動的にアップグレードされます。これが望ましくない場合は、**Manual** 承認ストラテジーを選択する必要があります。

14.2. RED HAT QUAY OPERATOR のアップグレード

インストールされた Operator を OpenShift Container Platform にアップグレードする一般的な方法については、インストールされた Operator のアップグレード を参照してください。

一般的に、Red Hat Quay は以前の (N-1) マイナーバージョンからのアップグレードのみをサポートしています。たとえば、Red Hat Quay 3.0.5 から最新バージョンの 3.5 への直接アップグレードはサポートされていません。代わりに、次のようにアップグレードする必要があります。

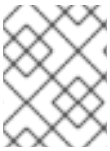
1. 3.0.5 → 3.1.3
2. 3.1.3 → 3.2.2
3. 3.2.2 → 3.3.4
4. 3.3.4 → 3.4.z

5. 3.4.z → 3.5.z

この作業は、必要なデータベースの移行が正しく実行され、適切な順序でアップグレードが行われるようにするために必要です。

場合によっては、Red Hat Quay は、以前の (N-2、N-3) マイナーバージョンからの直接のシングルステップアップグレードをサポートします。これにより、古いリリースを使用している顧客のアップグレード手順が簡素化されます。Red Hat Quay 3.11 では、次のアップグレードパスがサポートされています。

- 3.9.z → 3.11.z
- 3.10.z → 3.11.z



注記

3.8.z から 3.11へのアップグレードはサポートされていません。ユーザーはまず 3.9 または 3.10 にアップグレードし、次に 3.11 にアップグレードする必要があります。

Red Hat Quay のスタンドアロンデプロイメントを使用しているユーザーが 3.11 にアップグレードしたい場合は、スタンドアロンアップグレード ガイドを参照してください。

14.2.1. Red Hat Quay のアップグレード

Red Hat Quay をあるマイナーバージョンから次のマイナーバージョン (たとえば、3.10 → 3.11) に更新するには、Red Hat Quay Operator の更新チャネルを変更する必要があります。

z ストリームのアップグレード (3.10.1 → 3.10.2 など) の場合、更新は、ユーザーがインストール時に最初に選択したメジャー/マイナーチャネルでリリースされます。**z** ストリームのアップグレードを実行する手順は、上記のように **approvalStrategy** によって異なります。承認ストラテジーが **Automatic** に設定されている場合、Red Hat Quay Operator は自動的に最新の **z** ストリームにアップグレードします。この場合、ダウンタイムがほとんどない (またはまったくない) 新しい **z** ストリームへの Red Hat Quay の自動ローリング更新が行われます。それ以外の場合は、インストールを開始する前に更新を手動で承認する必要があります。

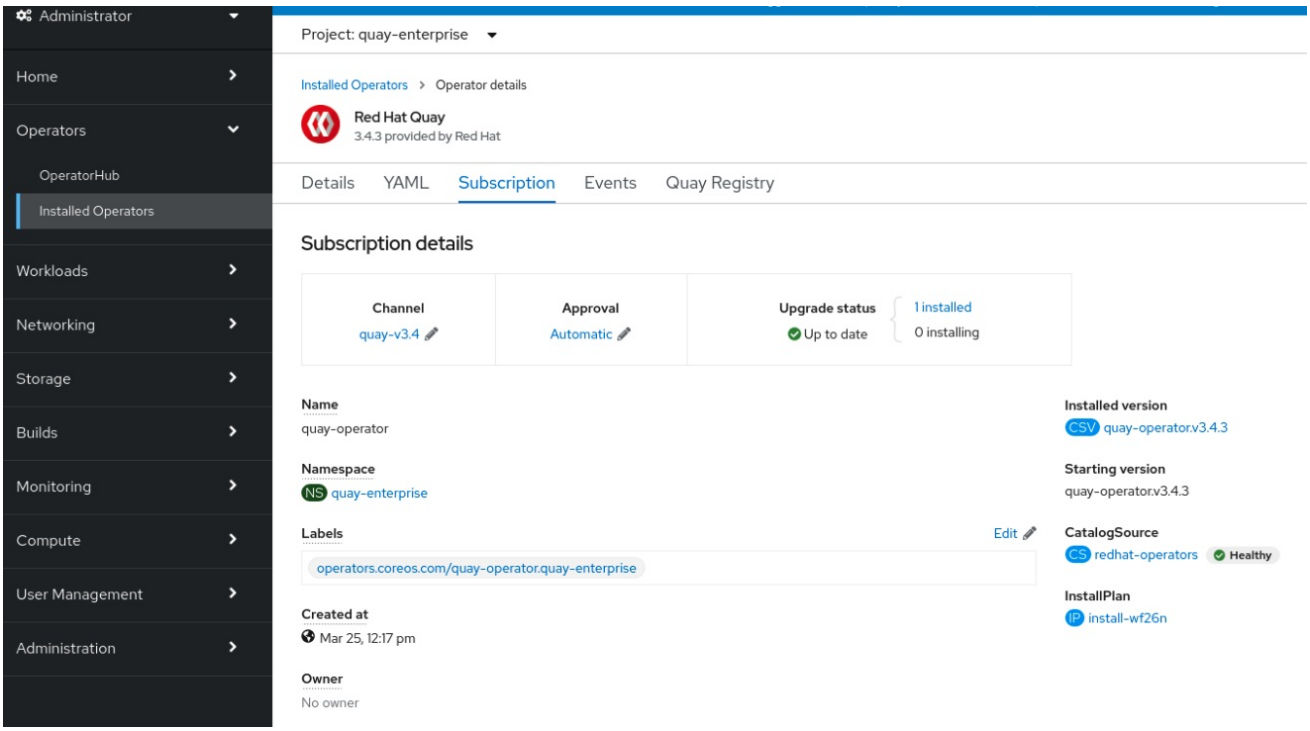
14.2.2. Red Hat Quay Operator の更新チャネルの変更

インストールされた Operator のサブスクリプションは、Operator の更新を追跡して受け取るために使用される更新チャネルを指定します。Red Hat Quay Operator をアップグレードして新規チャネルからの更新の追跡および受信を開始するには、インストールされた Red Hat Quay Operator の **Subscription** タブで更新チャネルを変更します。**Automatic** 承認ストラテジーのあるサブスクリプションの場合、アップグレードは自動的に開始し、インストールされた Operator をリスト表示したページでモニターできます。

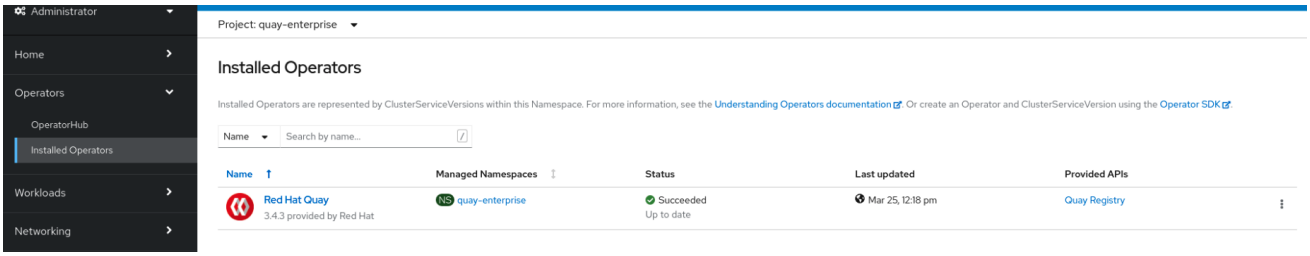
14.2.3. 保留中の Operator アップグレードの手動による承認

インストールされた Operator のサブスクリプションの承認ストラテジーが **Manual** に設定されている場合は、新規の更新が現在の更新チャネルにリリースされると、インストールを開始する前に更新を手動で承認する必要があります。Red Hat Quay Operator に保留中のアップグレードがある場合、このステータスはインストールされたOperator のリストに表示されます。Red Hat Quay Operator の **Subscription** タブで、インストール計画をプレビューし、アップグレードに利用可能なリソースとして一覧表示されるリソースを確認できます。問題がなければ、**Approve** をクリックし、Installed Operators を一覧表示したページに戻り、アップグレードの進捗を監視します。

以下のイメージには、更新 **Channel**、**Approval** ストラテジー、**Upgrade status** および **InstallPlan** などの UI の **Subscription** タブが表示されています。



Installed Operator のリストは、現在の Quay インストールの概要を提供します。



14.3. QUAYREGISTRY リソースのアップグレード

Red Hat Quay Operator を起動すると、監視するように設定されている namespace にある **QuayRegistries** をすぐに検索します。見つかった場合は、次のロジックが使用されます。

- **status.currentVersion** が設定されていない場合は、通常通り調整を行います。
- **status.currentVersion** が Operator のバージョンと等しい場合は、通常通り調整を行います。
- **status.currentVersion** が Operator のバージョンと一致しない場合は、アップグレードできるかどうかを確認します。可能な場合は、アップグレードタスクを実行し、完了後に **status.currentVersion** を Operator のバージョンに設定します。アップグレードできない場合は、エラーを返し、**QuayRegistry** とそのデプロイされた Kubernetes オブジェクトのみを残します。

14.4. QUAYECOSYSTEM のアップグレード

アップグレードは、**QuayEcosystem** API を使用して限られた設定を行っていた旧バージョンの Operator からサポートされています。移行が予期せず行われるようにするには、移行を行うために特別なラベルを **QuayEcosystem** に適用する必要があります。Operator が管理するための新しい

QuayRegistry が作成されますが、古い **QuayEcosystem** は手動で削除されるまで残り、何か問題が発生した場合にロールバックして Quay にアクセスできるようになります。既存の **QuayEcosystem** を新しい **QuayRegistry** に移行するには、次の手順を実行します。

手順

1. **"quay-operator/migrate": "true"** を **QuayEcosystem** の **metadata.labels** に追加します。

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

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

2. **QuayRegistry** が **QuayEcosystem** と同じ **metadata.name** で作成されるまで待機します。**QuayEcosystem** にはラベル **"quay-operator/migration-complete": "true"** のマークが付けられます。

3. 新しい **QuayRegistry** の **status.registryEndpoint** が設定されたら、Red Hat Quay にアクセスし、すべてのデータと設定が正常に移行されたことを確認します。

4. すべてが正常に動作する場合は **QuayEcosystem** を削除でき、Kubernetes ガベージコレクションによって古いリソースがすべてクリーンアップされます。

14.4.1. QuayEcosystem アップグレードを元に戻す

QuayEcosystem から **QuayRegistry** への自動アップグレード時に問題が発生した場合は、以下の手順を実行して **QuayEcosystem** の使用に戻します。

手順

1. UI または **kubectl** のいずれかを使用して **QuayRegistry** を削除します。

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

2. **Route** を使用して外部アクセスを提供していた場合は、UI や **kubectl** を使用して元の **Service** を指すように **Route** を変更します。

注記

QuayEcosystem が PostgreSQL データベースを管理している場合、アップグレードプロセスにより、アップグレードされた Operator が管理する新しい Postgres データベースにデータが以降されます。古いデータベースは変更または削除されませんが、移行が完了すると Quay はこのデータベースを使用しなくなります。データの移行中に問題が発生した場合は、アップグレードプロセスを終了し、データベースをマネージド外コンポーネントとして継続して使用することが推奨されます。

14.4.2. アップグレードでサポートされる QuayEcosystem 設定

QuayEcosystem コンポーネントの移行が失敗するかサポートされていない場合、Red Hat Quay Operator はログと **status.conditions** でエラーを報告します。アンマネージドコンポーネントを移行する場合、Kubernetes リソースを導入する必要がなく、必要な値はすべて Red Hat Quay の **config.yaml** で指定されているため、正常に移行できるはずです。

Database

一時データベースはサポートされません (**volumeSize** フィールドを設定する必要があります)。

Redis

特別な設定は必要ありません。

External Access

パススルー **Route** アクセスのみが自動移行でサポートされます。他の方法には手動移行が必要です。

- ホスト名のない **LoadBalancer**: **QuayEcosystem** にラベル **"quay-operator/migration-complete": "true"** が付けられた後、Kubernetes が **Service** をガベージコレクションしてロードバランサーを削除するのを防ぐため、**QuayEcosystem** を削除する **前** に、既存の **Service** から **metadata.ownerReferences** フィールドを削除します。新規 **Service** は **metadata.name** 形式の **<QuayEcosystem-name>-quay-app** で作成されます。既存の **Service** の **spec.selector** を新しい **Service** の **spec.selector** に合わせて編集することで、古いロードバランサーのエンドポイントへのトラフィックが新しい Pod に誘導されるようになります。これで古い **Service** を管理します。Quay Operator はこれを管理しません。

- カスタムホスト名を持つ **LoadBalancer/NodePort/Ingress**: タイプ **LoadBalancer** の新規 **Service** は **metadata.name** 形式の **<QuayEcosystem-name>-quay-app** で作成されます。新しい **Service** が提供する **status.loadBalancer** エンドポイントを指すように、DNS 設定を変更します。

Clair

特別な設定は必要ありません。

オブジェクトストレージ

QuayEcosystem には管理オブジェクトストレージコンポーネントがないため、オブジェクトストレージには常に管理外のマークが付けられます。ローカルストレージはサポートされません。

リポジトリーのミラーリング

特別な設定は必要ありません。

関連情報

- Red Hat Quay Operator の詳細は、アップストリームの quay-operator プロジェクトを参照してください。