Red Hat Quay 3.11

Red Hat Quay のトラブルシューティング

Red Hat Quay のトラブルシューティング

Last Updated: 2024-04-05

Red Hat Quay 3.11 Red Hat Quay のトラブルシューティング

Red Hat Quay のトラブルシューティング

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Red Hat Quay のトラブルシューティング

目次

はじめに

Red Hat は、Red Hat Quay デプロイメントのデータを収集するための管理者ツールを提供します。このデータを使用して、Red Hat Quay デプロイメントのトラブルシューティングを自分で行うことも、サポートチケットを提出することもできます。

第1章 サポート

本書で説明されている手順、または Red Hat Quay 全般で問題が発生した場合は、Red Hat カスタマーポータル にアクセスしてください。カスタマーポータルでは、以下を行うことができます。

- Red Hat 製品に関するアーティクルおよびソリューションを対象とした Red Hat ナレッジベースの検索またはブラウズ。
- Red Hat サポートに対するサポートケースの送信。
- その他の製品ドキュメントへのアクセス。

デプロイメントの問題を特定するには、Red Hat Quay デバッグツールを使用するか、デプロイメントのヘルスエンドポイントをチェックして問題に関する情報を取得します。デプロイメントに関する正常性情報をデバッグまたは取得すれば、Red Hat ナレッジベースで解決策を検索したり、サポートチケットを提出したりできます。

このドキュメントを改善するための提案がある場合、またはエラーを見つけた場合は、Jira issue を **ProjectQuay** プロジェクトに送信してください。セクション名や Red Hat Quay のバージョンなどの具体的な詳細を入力します。

1.1. RED HAT ナレッジベースについて

Red Hat ナレッジベース は、お客様が Red Hat の製品やテクノロジーを最大限に活用できるようにするための豊富なコンテンツを提供します。Red Hat ナレッジベースは、Red Hat 製品のインストール、設定、および使用に関する記事、製品ドキュメント、および動画で設定されています。さらに、既知の問題に対する解決策を検索でき、それぞれに根本原因の簡潔な説明と修復手順が記載されています。

Red Hat Quay サポートチームは、一般的な問題の解決策を詳しく説明した Red Hat Quay の Consolidate トラブルシューティング記事 も保守しています。これは、ユーザーがさまざまな問題に効果的かつ効率的に対処できるようにする、進化するドキュメントです。

1.2. RED HAT ナレッジベースの検索

Red Hat Quay の問題が発生した場合には、初期検索を実行して、Red Hat ナレッジベースにソリューションがすでに存在しているかどうかを確認できます。

前提条件

- Red Hat カスタマーポータルのアカウントがある。

手順

1. Red Hat カスタマーポータル にログインします。
2. 主な Red Hat カスタマーポータルの検索フィールドに、問題に関連する入力キーワードおよび文字列を入力します。たとえば、以下を入力します。
   - Red Hat Quay コンポーネント (**database** など)
   - 関連する手順 (**installation** など)
   - 明示的な失敗に関連する警告、エラーメッセージ、およびその他の出力
3. **Search** をクリックします。

4. **Red Hat Quay**製品フィルターを選択します。
5. コンテンツタイプフィルターで **ナレッジベース** を選択します。

1.3. サポートケースの送信

前提条件

- Red Hat カスタマーポータルのアカウントがある。
- Red Hat の標準またはプレミアムサブスクリプションがある。

手順

1. Red Hat Customer Portal にログインし、**Open a support case**を選択します。
2. **Troubleshoot** タブを選択します。
3. **概要**には、問題の簡潔で説明的な概要と、確認されている現象および予想される動作の詳細情報を入力します。
4. Red Hat ナレッジベースで推奨されるソリューション一覧を確認してください。この一覧に上げられているソリューションは、報告しようとしている問題に適用される可能性があります。推奨されている記事で問題が解決されない場合は、次の手順に進んでください。
5. **Product** で、**Red Hat Quay** を選択します。
6. 使用している Red Hat Quay のバージョンを選択します。
7. **Continue** をクリックします。
8. オプション: ドラッグアンドドロップ、貼り付け、または参照してファイルをアップロードします。これは、Red Hat Quay デプロイメントから収集されたデバッグログである可能性があります。
9. **Get support** をクリックしてチケットを提出します。

第2章 RED HAT QUAY をデバッグモードで実行する

Red Hat は、サポートケースの作成時にデバッグ情報を収集することを推奨します。Red Hat Quay をデバッグモードで実行すると、管理者がさまざまな問題に関する詳細情報を見つけるのに役立つ詳細ログが提供されます。デバッグモードを有効にすると、エラーを再現し、geo レプリケーションデプロイメント、Operator デプロイメント、スタンドアロン Red Hat Quay デプロイメント、オブジェクトストレージの問題などの解決策を検証するプロセスが高速化されます。さらに、Red Hat サポートが根本原因分析を実行するのにも役立ちます。

2.1. デバッグモードでのスタンドアロン RED HAT QUAY デプロイメントの実行

Red Hat Quay をデバッグモードで実行すると、管理者がさまざまな問題に関する詳細情報を見つけるのに役立つ詳細ログが提供されます。デバッグモードを有効にすると、エラーを再現し、解決策を検証するプロセスが高速化されます。

Red Hat Quay のスタンドアロンデプロイメントをデバッグモードで実行するには、次の手順を使用します。

手順

1. 以下のコマンドを入力して、スタンドアロンの Red Hat Quay デプロイメントをデバッグモードで実行します。

```
$ podman run -p 443:8443 -p 80:8080 -e DEBUGLOG=true  -v /config:/conf/stack -v
/storage:/datastorage -d {productrepo}/{quayimage}:{productminv}
```

2. デバッグログを表示するには、以下のコマンドを入力します。

```
$ podman logs quay
```

2.2. デバッグモードでの RED HAT QUAY OPERATOR の実行

次の手順を使用して、Red Hat Quay Operator をデバッグモードで実行します。

手順

1. 次のコマンドを入力して、**QuayRegistry** カスタムリソース定義を編集します。

```
$ oc edit quayregistry <quay_registry_name> -n <quay_namespace>
```

2. **QuayRegistry** を更新して、以下のパラメーターを追加します。

```
spec:
  - kind: quay
    managed: true
    overrides:
      env:
      - name: DEBUGLOG
        value: "true"
```

3. デバッグを有効にして Red Hat Quay Operator を再起動したら、レジストリーからイメージをプルしてみてください。それでも遅い場合は、すべての **Quay** Pod からファイルにすべての dog をダンプし、ファイルで詳細を確認します。

第3章 RED HAT QUAY のログ情報

を使用してログ情報を取得すると、コンテナーまたは Pod で実行されているアプリケーションの管理、監視、トラブルシューティングにさまざまな方法で役立ちます。ログ情報の取得が重要である理由には、次のようなものがあります。

- **デバッグとトラブルシューティング**: ログはアプリケーション内で何が起こっているかについての洞察を提供し、開発者やシステム管理者が問題を特定して解決できるようにします。ログメッセージを分析すると、アプリケーションの実行中に発生する可能性のあるエラー、例外、警告、または予期しない動作を特定できます。

- **パフォーマンス監視**: ログの監視は、アプリケーションとそのコンポーネントのパフォーマンスを追跡するのに役立ちます。応答時間、リクエスト率、リソース使用率などの指標を監視すると、需要に合わせてアプリケーションを最適化およびスケーリングするのに役立ちます。

- **セキュリティー分析**: 潜在的なセキュリティー違反を監査および検出するには、ログが不可欠です。ログを分析することで、不審なアクティビティー、不正アクセスの試み、または異常な動作を特定でき、セキュリティーの脅威の検出と対応に役立ちます。

- **ユーザー行動の追跡**: 場合によっては、ログを使用してユーザーのアクティビティーや行動を追跡できます。これは、ユーザーアクションの追跡が監査やコンプライアンスの目的に役立つ可能性がある、機密データを扱うアプリケーションにとって特に重要です。

- **キャパシティプランニング**: ログデータを使用してリソースの使用パターンを把握し、キャパシティプランニングに役立てることができます。ログを分析することで、ピーク使用期間を特定し、リソースのニーズを予測して、それに応じてインフラストラクチャーを最適化できます。

- **エラー分析**: エラーが発生した場合、ログはエラーに至るまでに何が起こったのかについての貴重なコンテキストを提供できます。これは、問題の根本原因を理解し、デバッグプロセスを容易にするのに役立ちます。

- **デプロイメントの検証**: デプロイメントプロセス中のログ記録は、アプリケーションが正しく起動しているかどうか、すべてのコンポーネントが期待どおりに機能しているかどうかを検証するのに役立ちます。

- **継続的インテグレーション/継続的デプロイメント (CI/CD)**: CI/CD パイプラインでは、チームが各段階の成功または失敗を監視できるように、ビルドとデプロイメントのステータスをキャプチャーするためにログ記録が不可欠です。

3.1. RED HAT QUAY のログ情報の取得

ログ情報は、geo レプリケーションデプロイメント、スタンドアロンデプロイメント、Operator デプロイメントなど、あらゆるタイプの Red Hat Quay デプロイメントについて取得できます。ミラーリングされたリポジトリーのログ情報も取得できます。これは、認証と認可の問題、およびオブジェクトストレージの問題のトラブルシューティングに役立ちます。必要なログ情報を取得したら、Red Hat ナレッジベース で解決策を検索するか、Red Hat サポートチームにサポートチケットを提出できます。

Red Hat Quay デプロイメントのログを取得するには、次の手順を使用します。

手順

- OpenShift Container Platform で Red Hat Quay Operator を使用している場合は、次のコマンドを入力してログを表示します。

```
$ oc logs <quay_pod_name>
```

- スタンドアロン Red Hat Quay デプロイメントを使用している場合は、次のコマンドを入力します。

```
$ podman logs <quay_container_name>
```

出力例

```
...
gunicorn-web stdout | 2023-01-20 15:41:52,071 [205] [DEBUG] [app] Starting request:
urn:request:0d88de25-03b0-4cf9-b8bc-87f1ac099429 (/oauth2/azure/callback) {'X-
Forwarded-For': '174.91.79.124'}
...
```

3.2. 詳細ログの検査

Red Hat Quay には詳細なログはありませんが、次の手順を使用すると、データベース Pod またはコンテナーの詳細なステータスチェックを取得できます。

手順

1. 次のコマンドを入力して、詳細なデータベースログを調べます。
   a. OpenShift Container Platform で Red Hat Quay Operator を使用している場合は、次のコマンドを入力します。

```
$ oc logs <quay_pod_name> --previous
```

```
$ oc logs <quay_pod_name> --previous -c <container_name>
```

```
$ oc cp <quay_pod_name>:/var/lib/pgsql/data/userdata/log/*
/path/to/desired_directory_on_host
```

   b. 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場合は、次のコマンドを入力します。

```
$ podman logs <quay_container_name> --previous
```

```
$ podman logs <quay_container_name> --previous -c <container_name>
```

```
$ podman cp <quay_container_name>:/var/lib/pgsql/data/userdata/log/*
/path/to/desired_directory_on_host
```

第4章 RED HAT QUAY の設定情報

設定 YAML を確認すると、Red Hat Quay の設定に関連するさまざまな問題を特定して解決するのに役立ちます。設定 YAML を確認すると、次の問題に対処するのに役立ちます。

- **間違った設定パラメーター**: データベースが期待どおりに機能していない場合、またはパフォーマンスの問題が発生している場合は、設定パラメーターに問題がある可能性があります。設定 YAML をチェックすることで、管理者は、必要なパラメーターがすべて正しく設定されており、データベースの意図した設定と一致していることを確認できます。

- **リソース制限**: 設定 YAML では、メモリーや CPU 制限など、データベースのリソース制限を指定する場合があります。データベースがリソースの制約に遭遇している場合、または他のサービスとの競合が発生している場合、これらの制限を調整すると、リソースの割り当てが最適化され、全体的なパフォーマンスが向上することがあります。

- **接続の問題**: 設定 YAML のネットワーク設定が正しくないと、アプリケーションとデータベース間の接続の問題が発生する可能性があります。ネットワーク設定が適切に行われていることを確認すると、接続と通信に関連する問題を解決できます。

- **データストレージとパス**: 設定 YAML には、データとログを保存するためのパスが含まれる場合があります。パスが正しく設定されていない場合、またはパスにアクセスできない場合、データベースでデータの読み取りまたは書き込み中にエラーが発生して、操作上の問題が発生する可能性があります。

- **認証とセキュリティー**: 設定 YAML には、ユーザー名、パスワード、アクセス制御などの認証設定が含まれる場合があります。これらの設定を確認することは、データベースのセキュリティーを維持して、許可されたユーザーのみがアクセスできるようにするために非常に重要です。

- **プラグインと拡張機能の設定**: 一部のデータベースは、機能を強化する拡張機能またはプラグインをサポートしています。これらのプラグインが正しく設定されていないか、正しく読み込まれていない場合は、問題が発生する可能性があります。設定 YAML を確認すると、プラグイン設定に関する問題の特定に役立ちます。

- **レプリケーションと高可用性の設定**: クラスター化またはレプリケートされたデータベースセットアップでは、設定 YAML でレプリケーション設定と高可用性設定を定義できます。設定が正しくないと、データの不整合やシステムの不安定が発生する可能性があります。

- **バックアップと回復のオプション**: 設定 YAML には、データのバックアップの実行方法と障害発生時のデータの回復方法を指定するバックアップと回復のオプションが含まれる場合があります。これらの設定を検証すると、データの安全性と回復プロセスの成功を保証できます。

設定 YAML を確認することで、Red Hat Quay 管理者は、データベースに依存するアプリケーションやサービスに重大な中断を引き起こす前に、これらの問題を検出して解決できます。

4.1. RED HAT QUAY の設定情報の取得

スタンドアロン、Operator、geo レプリケーションのデプロイメントを含む、あらゆるタイプの Red Hat Quay デプロイメントの設定情報を取得できます。設定情報を取得すると、認証と認可、データベース、オブジェクトストレージ、リポジトリーミラーリングに関する問題の解決に役立ちます。必要な設定情報を取得したら、**config.yaml** ファイルを更新するか、Red Hat ナレッジベース でソリューションを検索するか、Red Hat サポートチームでサポートチケットを作成してください。

手順

1. Red Hat Quay Operator デプロイメントに関する設定情報を取得するには、**oc exec**、**oc cp**、または **oc rsync** を使用できます。

   a. **oc exec** コマンドを使用するには、次のコマンドを入力します。

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

   このコマンドは、**config.yaml** ファイルをターミナルに直接返します。

   b. **oc copy** コマンドを使用するには、次のコマンドを入力します。

   ```
   $ oc cp <quay_pod_name>:/conf/stack/config.yaml /tmp/config.yaml
   ```

   この情報をデバイスに表示するには、次のコマンドを入力します。

   ```
   $ cat /tmp/config.yaml
   ```

   c. **oc rsync** コマンドを使用するには、次のコマンドを入力します。

   ```
   oc rsync <quay_pod_name>:/conf/stack/ /tmp/local_directory/
   ```

   この情報をデバイスに表示するには、次のコマンドを入力します。

   ```
   $ cat /tmp/local_directory/config.yaml
   ```

   出力例

   ```
   DISTRIBUTED_STORAGE_CONFIG:
   local_us:
   - RHOCSStorage
   - access_key: redacted
     bucket_name: lht-quay-datastore-68fff7b8-1b5e-46aa-8110-c4b7ead781f5
     hostname: s3.openshift-storage.svc.cluster.local
     is_secure: true
     port: 443
     secret_key: redacted
     storage_path: /datastorage/registry
   DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS:
   - local_us
   DISTRIBUTED_STORAGE_PREFERENCE:
   - local_us
   ```

2. スタンドアロン Red Hat Quay デプロイメントの設定情報を取得するには、**podman cp** または **podman exec** を使用できます。

   a. **podman copy** コマンドを使用するには、次のコマンドを入力します。

   ```
   $ podman cp <quay_container_id>:/conf/stack/config.yaml /tmp/local_directory/
   ```

   この情報をデバイスに表示するには、次のコマンドを入力します。

   ```
   $ cat /tmp/local_directory/config.yaml
   ```

b. **podman exec** を使用するには、次のコマンドを入力します。

```
$ podman exec -it <quay_container_id> cat /conf/stack/config.yaml
```

出力例

```
BROWSER_API_CALLS_XHR_ONLY: false
ALLOWED_OCI_ARTIFACT_TYPES:
    application/vnd.oci.image.config.v1+json:
        - application/vnd.oci.image.layer.v1.tar+zstd
    application/vnd.sylabs.sif.config.v1+json:
        - application/vnd.sylabs.sif.layer.v1+tar
AUTHENTICATION_TYPE: Database
AVATAR_KIND: local
BUILDLOGS_REDIS:
    host: quay-server.example.com
    password: strongpassword
    port: 6379
DATABASE_SECRET_KEY: 05ee6382-24a6-43c0-b30f-849c8a0f7260
DB_CONNECTION_ARGS: {}
---
```

4.2. データベース設定情報の取得

次の手順を使用して、データベースに関する設定情報を取得できます。

警告

PostgreSQL データベースとの対話は潜在的に破壊的です。Red Hat Quay サポートスペシャリストの助けを借りて次の手順を実行することを強く推奨します。

手順

- OpenShift Container Platform で Red Hat Quay Operator を使用している場合は、次のコマンドを入力します。

```
$ oc exec -it <database_pod> -- cat /var/lib/pgsql/data/userdata/postgresql.conf
```

- 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場合は、次のコマンドを入力します。

```
$ podman exec -it <database_container> cat /var/lib/pgsql/data/userdata/postgresql.conf
```

第5章 RED HAT QUAY デプロイメントでのヘルスチェックの実行

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

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

5.1. RED HAT QUAY ヘルスチェックエンドポイント

重要

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

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

表5.1 ヘルスチェックエンドポイント

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

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

5.2. RED HAT QUAY ヘルスチェックエンドポイントへの移動

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

手順

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

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

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

第6章 RED HAT QUAY コンポーネントのトラブルシューティング

このドキュメントは、Red Hat Quay 内の特定のコンポーネントのトラブルシューティングに焦点を当てており、発生する可能性のある問題を解決するための対象を絞ったガイダンスを提供します。このリソースはシステム管理者、operator、開発者向けに設計されており、Red Hat Quay の個々のコンポーネントに関連する問題の診断とトラブルシューティングを支援することを目的としています。

以下の手順に加えて、Red Hat Quay をデバッグモードで実行し、ログ情報を取得して、設定情報を取得し、エンドポイントでヘルスチェックを実行することで、Red Hat Quay コンポーネントのトラブルシューティングを行うこともできます。

次の手順を使用すると、コンポーネントの一般的な問題のトラブルシューティングを行うことができます。その後、Red Hat ナレッジベース でソリューションを検索したり、Red Hat サポートチームにサポートチケットを提出したりできます。

6.1. RED HAT QUAY データベースのトラブルシューティング

Red Hat Quay で使用される PostgreSQL データベースには、コンテナーイメージとその管理に関連するさまざまな種類の情報が保存されます。PostgreSQL データベースが保存する情報の重要な部分には以下が含まれます。

- **イメージのメタデータ**: データベースには、イメージ名、バージョン、作成タイムスタンプ、イメージを所有するユーザーまたは組織など、コンテナーイメージに関連付けられたメタデータが保存されます。この情報により、レジストリー内のコンテナーイメージを簡単に識別および整理できます。

- **イメージタグ**: Red Hat Quay を使用すると、ユーザーはコンテナーイメージにタグを割り当てることができ、便利なラベル付けとバージョン管理が可能になります。PostgreSQL データベースは、イメージタグとそれに対応するイメージマニフェスト間のマッピングを維持し、ユーザーが提供されたタグに基づいてコンテナーイメージの特定のバージョンを取得できるようにします。

- **イメージレイヤー**: コンテナーイメージは複数のレイヤーで設定され、個別のオブジェクトとして保存されます。データベースには、順序、チェックサム、サイズなど、これらのレイヤーに関する情報が記録されます。このデータは、コンテナーイメージの効率的な保存と取得にとって非常に重要です。

- **ユーザーおよび組織のデータ**: Red Hat Quay はユーザーと組織の管理をサポートしており、ユーザーがコンテナーイメージへのアクセスを認証および管理できるようにします。PostgreSQL データベースには、ユーザー名、電子メールアドレス、認証トークン、アクセス許可などのユーザーと組織の情報が保存されます。

- **レポジトリー情報**: Red Hat Quay は、コンテナーイメージをリポジトリーに編成します。リポジトリーは、関連するイメージをグループ化するための論理ユニットとして機能します。データベースは、名前、説明、表示設定、アクセス制御情報などのリポジトリーデータを維持し、ユーザーがリポジトリーを効果的に管理および共有できるようにします。

- **イベントログ**: Red Hat Quay は、イメージ管理とリポジトリーの操作に関連するさまざまなイベントとアクティビティーを追跡します。イメージのプッシュ、プル、削除、リポジトリーの変更を含むこれらのイベントログは PostgreSQL データベースに保存されて、監査証跡が提供されます。このため、管理者はシステムアクティビティーを監視および分析できます。

このセクションの内容では、次の手順について説明します。

- **デプロイメントのタイプの確認**: データベースが仮想マシン上のコンテナーとしてデプロイされているか、OpenShift Container Platform 上の Pod としてデプロイされているかを確認します。
- **コンテナーまたは Pod のステータスの確認**: デプロイメントタイプに基づいた特定のコマンドを使用して、**データベース** Pod またはコンテナーのステータスを確認します。
- **データベースコンテナーまたは Pod のログの調査**: データベース Pod またはコンテナーのログ (さまざまなデプロイメントタイプのコマンドを含む) にアクセスして調査します。
- **Red Hat Quay とデータベース Pod 間の接続を確認する**: 関連するコマンドを使用して、Red Hat Quay と **database** Pod 間の接続を確認します。
- **データベース設定の確認**: デプロイメントタイプに基づいて、さまざまなレベル (OpenShift Container Platform または PostgreSQL レベル) でデータベース設定を確認します。
- **リソース割り当ての確認**: ディスク使用量やその他のリソース使用量を含む、Red Hat Quay デプロイメントのリソース割り当てを監視します。
- **Red Hat Quay データベースとの対話** データベースにアクセスしてクエリーを実行するコマンドなど、PostgreSQL データベースと対話する方法を学びます。

6.1.1. Red Hat Quay データベースの問題のトラブルシューティング

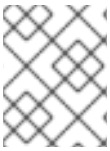
PostgreSQL データベースのトラブルシューティングを行うには、次の手順を使用します。

6.1.1.1. Red Hat Quay データベースとの対話

PostgreSQL データベースと対話するには、次の手順を使用します。

警告

PostgreSQL データベースとの対話は潜在的に破壊的です。Red Hat Quay サポートスペシャリストの助けを借りて次の手順を実行することを強く推奨します。



注記

PostgreSQL データベースとの対話は、認可と認証の問題のトラブルシューティングにも使用できます。

手順

1. Red Hat Quay データベースを実行します。
   a. 次のコマンドを入力して、OpenShift Container Platform 上の Red Hat Quay データベース Pod を実行します。

```
$ oc exec -it <quay_database_pod> -- psql
```

b. 次のコマンドを入力して、スタンドアロンデプロイメントで Red Hat Quay データベースを実行します。

```
$ sudo podman exec -it <quay_container_name> /bin/bash
```

2. PostgreSQL シェルに入ります。

警告

PostgreSQL データベースとの対話は潜在的に破壊的です。Red Hat Quay サポートスペシャリストの助けを借りて次の手順を実行することを強く推奨します。

a. Red Hat Quay Operator を使用している場合は、次のコマンドを入力して PostgreSQL シェルに入ります。

```
$ oc rsh <quay_pod_name> psql -U your_username -d your_database_name
```

b. スタンドアロン Red Hat Quay デプロイメントを使用している場合は、次のコマンドを入力して PostgreSQL シェルに入ります。

```
bash-4.4$ psql -U your_username -d your_database_name
```

6.1.1.2. CrashLoopBackOff 状態のトラブルシューティング

crashloopbackoff 状態のトラブルシューティングを行うには、次の手順を使用します。

手順

1. コンテナーまたは Pod が **crashloopbackoff** 状態にある場合は、次のコマンドを入力できます。

a. 次のコマンドを入力して、Red Hat Quay Operator をスケールダウンします。

```
$ oc scale deployment/quay-operator.v3.8.z  --replicas=0
```

出力例

```
deployment.apps/quay-operator.v3.8.z scaled
```

b. 次のコマンドを入力して、Red Hat Quay データベースをスケールダウンします。

```
$ oc scale deployment/<quay_database> --replicas=0
```

出力例

```
deployment.apps/<quay_database> scaled
```

c. 次のコマンドを入力して、Red Hat Quay データベースを編集します。

> **警告**
>
> PostgreSQL データベースとの対話は潜在的に破壊的です。Red Hat Quay サポートスペシャリストの助けを借りて次の手順を実行することを強く推奨します。

```
$ oc edit deployment <quay_database>
```

```
...
 template:
   metadata:
     creationTimestamp: null
     labels:
       quay-component: <quay_database>
       quay-operator/quayregistry: quay-operator.v3.8.z
   spec:
     containers:
     - env:
       - name: POSTGRESQL_USER
         value: postgres
       - name: POSTGRESQL_DATABASE
         value: postgres
       - name: POSTGRESQL_PASSWORD
         value: postgres
       - name: POSTGRESQL_ADMIN_PASSWORD
         value: postgres
       - name: POSTGRESQL_MAX_CONNECTIONS
         value: "1000"
       image: registry.redhat.io/rhel8/postgresql-
10@sha256:a52ad402458ec8ef3f275972c6ebed05ad64398f884404b9bb8e3010c5c95291

       imagePullPolicy: IfNotPresent
       name: postgres
       command: ["/bin/bash", "-c", "sleep 86400"] 1
...
```

1 この行を同じインデントに追加します。

出力例

```
deployment.apps/<quay_database> edited
```

d. **<quay_database>** 内で次のコマンドを実行します。

```
$ oc exec -it <quay_database> -- cat /var/lib/pgsql/data/userdata/postgresql/logs/*
/path/to/desired_directory_on_host
```

6.1.1.3. Red Hat Quay とデータベース Pod 間の接続を確認する

次の手順を使用して、Red Hat Quay とデータベース Pod 間の接続を確認します。

手順

1. Red Hat Quay とデータベース Pod 間の接続を確認します。
   a. OpenShift Container Platform で Red Hat Quay Operator を使用している場合は、次のコマンドを入力します。

```
$ oc exec -it _quay_pod_name_ -- curl -v telnet://<database_pod_name>:5432
```

   b. 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場合は、次のコマンドを入力します。

```
$ podman exec -it <quay_container_name >curl -v
telnet://<database_container_name>:5432
```

6.1.1.4. リソース割り当ての確認

リソースの割り当てを確認するには、次の手順を実行します。

手順

1. 実行中のコンテナーのリストを取得します。
2. Red Hat Quay デプロイメントのディスク使用量を監視します。
   a. OpenShift Container Platform で Red Hat Quay Operator を使用している場合は、次のコマンドを入力します。

```
$ oc exec -it <quay_database_pod_name> -- df -ah
```

   b. 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場合は、次のコマンドを入力します。

```
$ podman exec -it <quay_database_conatiner_name> df -ah
```

3. 他のリソースの使用状況を監視します。
   a. 次のコマンドを入力して、Red Hat Quay Operator デプロイメントでのリソース割り当てを確認します。

```
$ oc adm top pods
```

   b. 次のコマンドを入力して、Red Hat Quay のスタンドアロンデプロイメント上の特定の Pod のステータスを確認します。

```
$ podman pod stats <pod_name>
```

   c. 次のコマンドを入力して、Red Hat Quay のスタンドアロンデプロイメント上の特定のコンテナーのステータスを確認します。

```
$ podman stats <container_name>
```

以下の情報が返されます。

- CPU %.前回の測定以降のコンテナーによる CPU 使用率のパーセンテージ。この値は、利用可能な CPU リソースのコンテナーの共有を表します。
- MEM USAGE / LIMIT: コンテナーの現在のメモリー使用量とそのメモリー制限が続きます。値は **current_usage/memory_limit** の形式で表示されます。たとえば、**300.4MiB/7.795GiB** は、コンテナーが現在 7.795 ギガバイトの制限のうち 300.4 メガバイトのメモリーを使用していることを示します。
- MEM %: メモリー制限に対するコンテナーによるメモリー使用量の割合。
- NET I/O: コンテナーのネットワーク I/O (入力/出力) 統計。コンテナーがネットワーク上で送受信したデータの量が表示されます。値は、**transmitted_bytes** / **received_bytes** の形式で表示されます。
- BLOCK I/O: コンテナーのブロック I/O (入力/出力) 統計。これは、コンテナーによって使用されるブロックデバイス (ディスクなど) に対して読み書きされるデータの量を表します。値は **read_bytes/write_bytes** の形式で表示されます。

6.1.2. Red Hat Quay スタンドアロンデプロイメントでのスーパーユーザーのパスワードのリセット

スーパーユーザーのパスワードをリセットするには、次の手順を使用します。

前提条件

- Red Hat Quay スーパーユーザーを作成している。
- Python 3.9 がインストールされている。
- Python 用の **pip** パッケージマネージャーがインストールされている。
- **pip** 用の **bcrypt** パッケージがインストールされている。

手順

1. 次のコマンドを入力して、Python 3.9 の **bcrypt** パッケージを使用して、安全なハッシュ化されたパスワードを生成します。

```
$ python3.9 -c 'import bcrypt; print(bcrypt.hashpw(b"newpass1234",
bcrypt.gensalt(12)).decode("utf-8"))'
```

出力例

```
$2b$12$T8pkgtOoys3G5ut7FV1She6vXlYgU.6TeoGmbbAVQtN8X8ch4knKm
```

2. 次のコマンドを入力して、Red Hat Quay コンテナーレジストリーのコンテナー ID を表示します。

```
$ sudo podman ps -a
```

出力例

```
CONTAINER ID  IMAGE                                COMMAND         CREATED         STATUS
PORTS                   NAMES
70560beda7aa  registry.redhat.io/rhel8/redis-5:1        run-redis       2 hours ago     Up 2 hours
ago     0.0.0.0:6379->6379/tcp                  redis
8012f4491d10  registry.redhat.io/quay/quay-rhel8:v3.8.2  registry        3 minutes ago   Up 8
seconds ago   0.0.0.0:80->8080/tcp, 0.0.0.0:443->8443/tcp  quay
8b35b493ac05  registry.redhat.io/rhel8/postgresql-10:1   run-postgresql  39 seconds ago  Up
39 seconds ago  0.0.0.0:5432->5432/tcp                  postgresql-quay
```

3. 次のコマンドで、**postgresql** コンテナーイメージの対話型シェルを実行します。

```
$ sudo podman exec -it 8b35b493ac05 /bin/bash
```

4. データベース、ユーザー名、およびホストアドレスを指定して、**quay** PostgreSQL データベースサーバーに再入力します。

```
bash-4.4$ psql -d quay -U quayuser -h 192.168.1.28 -W
```

5. パスワードを紛失したスーパーユーザー管理者の **password_hash** を更新します。

```
quay=> UPDATE public.user SET password_hash =
'$2b$12$T8pkgtOoys3G5ut7FV1She6vXlYgU.6TeoGmbbAVQtN8X8ch4knKm' where
username = 'quayadmin';
```

出力例

```
UPDATE 1
```

6. 次のコマンドを入力して、**password_hash** が更新されたことを確認します。

```
quay=> select * from public.user;
```

出力例

```
id | uuid | username | password_hash | email | verified | stripe_id | organization | robot |
invoice_email | invalid_login_attempts | last_invalid_login |removed_tag_expiration_s |
enabled | invoice_email_address | company | family_name | given_name | location |
maximum_queued_builds_count | creation_date | last_accessed
----+--------------------------------------+----------+--------------------------------------------------------------
-------+----------------------+---
-------+----------+-------------+-------+--------------+----------------------+-------------------------
--+-------------------------+------
---+----------------------+---------+-------------+------------+----------+---------------------------+---
------------------------+----------
1 | 73f04ef6-19ba-41d3-b14d-f2f1eed94a4a | quayadmin |
$2b$12$T8pkgtOoys3G5ut7FV1She6vXlYgU.6TeoGmbbAVQtN8X8ch4knKm |
quayadmin@example.com | t | | f | f | f | 0 | 2023-02-23 07:54:39.116485 | 1209600 | t | | | | | |
| 2023-02-23 07:54:39.116492
```

7. 新しいパスワードを使用して Red Hat Quay デプロイメントにログインします。

```
$ sudo podman login -u quayadmin -p newpass1234 http://quay-server.example.com --tls-verify=false
```

出力例

```
Login Succeeded!
```

関連情報

詳細は、Quay のスーパーユーザーパスワードのリセット を参照してください。

6.1.3. Red Hat Quay Operator でのスーパーユーザーのパスワードのリセット

前提条件

- Red Hat Quay スーパーユーザーを作成している。
- Python 3.9 がインストールされている。
- Python 用の **pip** パッケージマネージャーがインストールされている。
- **pip** 用の **bcrypt** パッケージがインストールされている。

手順

1. Red Hat Quay デプロイメントにログインします。
2. OpenShift Container Platform UI で、**Workloads → Secrets** に移動します。
3. Red Hat Quay デプロイメントの名前空間 (例えば、**Project quay** を選択します。
4. PostgreSQL データベースの認証情報を見つけて保存します。
5. 次のコマンドを入力して、Python 3.9 の **bcrypt** パッケージを使用して、安全なハッシュ化されたパスワードを生成します。

```
$ python3.9 -c 'import bcrypt; print(bcrypt.hashpw(b"newpass1234", bcrypt.gensalt(12)).decode("utf-8"))'
```

出力例

```
$2b$12$zoilcTG6XQeAoVuDuIZH0..UpvQEZcKh3V6puksQJaUQupHgJ4.4y
```

6. CLI で、たとえば次のようにデータベースにログインします。

```
$ oc rsh quayuser-quay-quay-database-669c8998f-v9qsl
```

7. 次のコマンドを入力して、データベース、ユーザー名、およびホストアドレスを指定して、**quay** PostgreSQL データベースサーバーへの接続を開きます。

```
sh-4.4$ psql -U quayuser-quay-quay-database -d quayuser-quay-quay-database -W
```

8. 次のコマンドを入力して、現在のユーザーのデフォルトのデータベースに接続します。

```
quay=> \c
```

9. パスワードを紛失したスーパーユーザー管理者の **password_hash** を更新します。

```
quay=> UPDATE public.user SET password_hash =
'$2b$12$zoilcTG6XQeAoVuDuIZH0..UpvQEZcKh3V6puksQJaUQupHgJ4.4y' where
username = 'quayadmin';
```

10. 次のコマンドを入力して、**password_hash** が更新されたことを確認します。

```
quay=> select * from public.user;
```

出力例

```
id | uuid | username | password_hash | email | verified | stripe_id | organization | robot |
invoice_email | invalid_login_attempts | last_invalid_login |removed_tag_expiration_s |
enabled | invoice_email_address | company | family_name | given_name | location |
maximum_queued_builds_count | creation_date | last_accessed
----+--------------------------------------+----------+--------------------------------------------------------
-------+----------------------+---
-------+-----------+--------------+-------+--------------+-----------------------+--------------------------
--+--------------------------+------
---+----------------------+---------+-------------+-----------+----------+---------------------------+---
------------------------+-----------
1 | 73f04ef6-19ba-41d3-b14d-f2f1eed94a4a | quayadmin |
$2b$12$zoilcTG6XQeAoVuDuIZH0..UpvQEZcKh3V6puksQJaUQupHgJ4.4y |
quayadmin@example.com | t | | f | f | f | 0 | 2023-02-23 07:54:39.116485 | 1209600 | t | | | | | |
| 2023-02-23 07:54:39.116492
```

11. OpenShift Container Platform 上の Red Hat Quay UI に移動し、新しい認証情報を使用してログインします。

6.2. RED HAT QUAY 認証のトラブルシューティング

Red Hat Quay に安全にアクセスするには、認証と認可が重要です。これらは同時に、機密のコンテナーイメージを保護し、ユーザー ID を検証して、アクセス制御を実施し、監査と説明責任を促進して、外部 ID プロバイダーとのシームレスな統合を可能にします。認証を優先することで、組織はコンテナーレジストリー環境の全体的なセキュリティーと整合性を強化できます。

Red Hat Quay では次の認証方法がサポートされています。

- **ユーザー名およびパスワード**: ユーザーはユーザー名とパスワードを入力することで認証でき、これらは Red Hat Quay で設定されたユーザーデータベースに対して検証されます。この従来の方法では、ユーザーはアクセスするために認証情報を入力する必要があります。

- **OAuth**: Red Hat Quay は OAuth 認証をサポートしており、ユーザーは Google、GitHub、Keycloak などのサードパーティーのサービスからの認証情報を使用して認証できます。OAuth によりシームレスで統合されたログインエクスペリエンスが可能になり、アカウントを個別に作成する必要がなくなり、ユーザー管理が容易になります。

- **OIDC**: OpenID Connect により、Single Sign-On (SSO) 機能とエンタープライズ ID プロバイダーとの統合が可能になります。OpenID Connect を使用すると、ユーザーは既存の組織の認

証情報を使用して認証できるため、さまざまなシステムやアプリケーションにわたって統一された認証エクスペリエンスが提供されます。

- **トークンベースの認証**: ユーザーは、Red Hat Quay 内の特定のリソースへのアクセスを許可する固有のトークンを取得できます。トークンは、OAuth や Red Hat Quay ユーザーインターフェイス内で API トークンを生成するなど、さまざまな手段を通じて取得できます。トークンベースの認証は、レジストリーへの自動アクセスまたはプログラムによるアクセスによく使用されます。

- **外部 ID プロバイダー**: Red Hat Quay は、認証の目的で LDAP や AzureAD などの外部認証プロバイダーと統合できます。この統合により、組織は既存の ID 管理インフラストラクチャーを使用できるようになり、一元的なユーザー認証が可能になり、個別のユーザーデータベースの必要性が軽減されます。

6.2.1. 特定のユーザーに対する Red Hat Quay の認証および認可の問題のトラブルシューティング

特定のユーザーの認証および認可の問題をトラブルシューティングするには、次の手順を使用します。

手順

1. Red Hat Quay Pod またはコンテナーで実行します。詳細は、Red Hat Quay データベースとの対話を参照してください。

2. 次のコマンドを入力して、外部認証のすべてのユーザーを表示します。

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

出力例

```
id | user_id | service_id |              service_ident              |              metadata_json
----+---------+------------+-----------------------------------------+-----------------------------------
------
1 |       1 |          3 | testuser0                               | {}
2 |       1 |          8 | PK7Zpg2Yu2AnfUKG15hKNXqOXirqUog6G-oE7OgzSWc |
{"service_username": "live.com#testuser0"}
3 |       2 |          3 | testuser1                               | {}
4 |       2 |          4 | 110875797246250333431                   | {"service_username":
"testuser1"}
5 |       3 |          3 | testuser2                               | {}
6 |       3 |          1 | 26310880                                | {"service_username": "testuser2"}
(6 rows)
```

3. ユーザーが **user** テーブルに挿入されていることを確認します。

```
quay=# select username, email from "user";
```

出力例

```
username  |        email
-----------+----------------------
testuser0 | testuser0@outlook.com
```

```
testuser1  | testuser1@gmail.com
testuser2  | testuser2@redhat.com
(3 rows)
```

6.3. RED HAT QUAY オブジェクトストレージのトラブルシューティング

オブジェクトストレージは、データをオブジェクトと呼ばれる **objects** の単位として管理するデータストレージアーキテクチャーの一種です。データを階層ディレクトリーやファイルに編成する従来のファイルシステムとは異なり、オブジェクトストレージはデータを一意の識別子を持つ独立したエンティティーとして扱います。各オブジェクトには、データ自体と、オブジェクトを説明して効率的な検索を可能にするメタデータが含まれています。

Red Hat Quay は、コンテナーイメージを保存および管理するための基礎となるストレージメカニズムとしてオブジェクトストレージを使用します。コンテナーイメージを個別のオブジェクトとして保存します。各コンテナーイメージは、独自の一意の識別子と関連するメタデータを持つオブジェクトとして扱われます。

6.3.1. Red Hat Quay オブジェクトストレージの問題のトラブルシューティング

Red Hat Quay オブジェクトストレージの問題をトラブルシューティングするには、次のオプションを使用します。

手順

- 次のコマンドを入力して、どのオブジェクトストレージが使用されているかを確認します。

```
$ oc get quayregistry quay-registry-name -o yaml
```

- 使用しているオブジェクトストレージが、テスト済みの統合 ページをチェックして、Red Hat Quay によって正式にサポートされていることを確認してください。
- デバッグモードを有効化します詳細は、デバッグモードでの Red Hat Quay の実行を参照してください。
- **config.yaml** ファイル内のオブジェクトストレージ設定を確認してください。設定が正確であり、オブジェクトストレージプロバイダーが提供する設定と一致していることを確認してください。アクセス認証情報、エンドポイント URL、バケット名とコンテナー名、その他の関連する設定パラメーターなどの情報を確認できます。
- Red Hat Quay がオブジェクトストレージエンドポイントにネットワーク接続されていることを確認します。ネットワーク設定をチェックして、Red Hat Quay とオブジェクトストレージエンドポイント間の通信をブロックする制限がないことを確認します。
- **FEATURE_STORAGE_PROXY** が **config.yaml** ファイルで有効になっている場合は、そのダウンロード URL がアクセス可能かどうかを確認してください。これは、Red Hat Quay のデバッグログで確認できます。以下に例を示します。

```
$ curl -vvv
"https://QUAY_HOSTNAME/_storage_proxy/dhaWZKRjlyO.....Kuhc=/https/quay.hostname.com/quay-
test/datastorage/registry/sha256/0e/0e1d17a1687fa270ba4f52a85c0f0e7958e13d3ded5123c3851a8031a9e55681?
AWSAccessKeyId=xxxx&Signature=xxxxxx4%3D&Expires=1676066703"
```

- Red Hat Quay の外部にあるオブジェクトストレージサービスにアクセスして、問題がデプロイメントに固有のものなのか、基礎となるオブジェクトストレージに固有のものなのかを判断してください。オブジェクトストレージプロバイダーが提供する **aws**、**gsutil**、**s3cmd など** のコマンドラインツールを使用して、バケット、コンテナーのリスト表示、オブジェクトのアップロードとダウンロードなどの基本的な操作を実行できます。これは問題を切り分けるのに役立つ場合があります。

6.4. 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。

6.4.1. Red Hat Quay の geo レプリケーションのトラブルシューティング

Red Hat Quay の geo レプリケーションのトラブルシューティングを行うには、次のセクションを使用してください。

6.4.1.1. バックエンドバケットでのデータレプリケーションの確認

次の手順を使用して、データがすべてのバックエンドバケットに適切にレプリケートされていることを確認します。

前提条件

- **aws** CLI 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、データがすべてのバックエンドバケットにレプリケートされていることを確認します。

```
$ aws --profile quay_prod_s3 --endpoint=http://10.0.x.x:port s3 ls ocp-quay --recursive --human-readable --summarize
```

出力例

```
Total Objects: 17996
Total Size: 514.4 GiB
```

6.4.1.2. バックエンドストレージのステータスを確認する

バックエンドストレージのステータスを確認するには、次のリソースを使用してください。

- **Amazon Web Service Storage (AWS)** AWS Service Health Dashboard で AWS S3 サービスの健全性ステータスを確認します。**aws** CLI または SDK を使用して既知のバケット内のオブジェクトをリストすることにより、S3 へのアクセスを検証します。

- **Google Cloud Storage (GCS)**: GCS サービスのステータスは Google Cloud ステータスダッシュボード を確認してください。Google Cloud SDK または GCS クライアントライブラリーを使用して既知のバケット内のオブジェクトをリスト表示し、GCS へのアクセスを確認します。

- **NooBaa**: NooBaa 管理コンソールまたは管理インターフェイスで、健全性またはステータスのインジケーターを確認してください。NooBaa サービスと関連コンポーネントが実行中であり、アクセス可能であることを確認します。NooBaa CLI または SDK を使用して既知のバケット内のオブジェクトをリストし、NooBaa へのアクセスを確認します。

- **Red Hat OpenShift Data Foundation**OpenShift Container Platform コンソールまたは管理インターフェイスで Red Hat OpenShift Data Foundation コンポーネントのステータスを確認してください。Red Hat OpenShift Data Foundation S3 インターフェイスとサービスの可用性を確認します。Red Hat OpenShift Data Foundation サービスが実行中であり、アクセス可能であることを確認します。適切な S3 互換 SDK または CLI を使用して既知のバケット内のオブジェクトをリストすることにより、Red Hat OpenShift Data Foundation S3 へのアクセスを検証します。

- **Ceph**: Ceph モニター、OSD、RGW などの Ceph サービスのステータスを確認します。Ceph クラスターが正常で動作していることを検証します。適切な Ceph オブジェクトストレージ API または CLI を使用して既知のバケット内のオブジェクトをリストすることにより、Ceph オブジェクトストレージへのアクセスを確認します。

- **Azure Blob Storage**: Azure ステータスダッシュボード をチェックして、Azure Blob Storage サービスの正常性ステータスを確認します。Azure CLI または Azure SDK を使用してコンテナーまたはオブジェクトをリスト表示し、Azure Blob Storage へのアクセスを検証します。

- **OpenStack Swift**: OpenStack Status ページをチェックして、OpenStack Swift サービスのステータスを確認します。プロキシーサーバー、コンテナーサーバー、オブジェクトサーバーなどの Swift サービスが実行中であり、アクセス可能であることを確認します。適切な Swift CLI または SDK を使用してコンテナーまたはオブジェクトをリストすることにより、Swift へのアクセスを検証します。

バックエンドストレージのステータスを確認した後、すべての Red Hat Quay インスタンスがすべての s3 ストレージバックエンドにアクセスできることを確認します。

6.5. リポジトリーのミラーリング

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

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

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

- 同期の現在の状態を確認する

ミラーリング機能を使用するには、次のアクションを実行する必要があります。

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

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

6.5.1. リポジトリーのミラーリングのトラブルシューティング

Red Hat Quay のレポジトリーミラーリングのトラブルシューティングを行うには、次のセクションを使用してください。

6.5.1.1. 認証と権限の確認

ミラーリングに使用される認証情報に、ソースと宛先の両方の Red Hat Quay インスタンスに必要なパーミッションおよびアクセス権限があることを確認します。

Red Hat Quay UI で、次の設定を確認します。

- アクセス制御設定。ミラーリング操作を実行するユーザーまたはサービスアカウントに必要な権限があることを確認してください。
- Red Hat Quay レジストリー上のロボットアカウントの権限。

6.6. CLAIR セキュリティースキャナー

6.6.1. Clair の問題のトラブルシューティング

Clair のトラブルシューティングを行うには、次の手順を使用します。

6.6.1.1. イメージの互換性を確認する

Clair を使用している場合は、スキャンしようとしているイメージが Clair でサポートされていることを確認してください。Clair には特定の要件があり、すべてのイメージ形式や設定をサポートしているわけではありません。

詳細は、Clair 脆弱性データベース を参照してください。

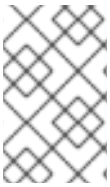
6.6.1.2. Clair アップデーターの許可リストへの登録

プロキシー設定の背後で Clair を使用している場合は、プロキシーまたはファイアウォール設定でアップデータを許可リストに登録する必要があります。アップデーター URL の詳細は、Clair アップデーター URL を参照してください。

6.6.1.3. Clair スキャナーとその依存関係の更新

最新バージョンの Clair セキュリティースキャナーを使用していることを確認してください。古いバージョンには、新しいイメージ形式のサポートがないか、既知の問題が存在する可能性があります。

Clair のバージョンを確認するには、次の手順を使用します。



注記

Clair ログの確認は、Clair ログにアップデータマイクロサービスからのエラーがあるかどうかを確認するためにも使用できます。デフォルトでは、Clair は脆弱性データベースを 30 分ごとに更新します。

手順

1. Clair のバージョンを確認してください。
   a. Red Hat Quay Operator で Clair を実行している場合は、次のコマンドを入力します。

```
$ oc logs clair-pod
```

   b. Red Hat Quay のスタンドアロンデプロイメントを実行し、Clair コンテナーを使用している場合は、次のコマンドを入力します。

```
$ podman logs clair-container
```

出力例

```
"level":"info",
"component":"main",
"version":"v4.5.1",
```

6.6.1.4. Clair のデバッグモードの有効化

デフォルトでは、Clair のデバッグモードが有効になっています。Clair **config.yaml** ファイルを更新することで、Clair のデバッグモードを有効にできます。

Clair のデバッグモードを有効にするには、次の手順を使用します。

手順

1. Clair のデバッグモードを有効にする
   a. Red Hat Quay Operator で Clair を実行している場合は、次のコマンドを入力します。

```
$ oc exec -it clair-pod-name -- cat /clair/config.yaml
```

   b. Red Hat Quay のスタンドアロンデプロイメントを実行し、Clair コンテナーを使用している場合は、次のコマンドを入力します。

```
$ podman exec -it clair-container-name cat /clair/config.yaml
```

2. Clair **config.yaml** ファイルを更新してデバッグを有効にします。

```
http_listen_addr: :8081
introspection_addr: :8088
log_level: debug
```

6.6.1.5. Clair 設定の確認

Clair **config.yaml** ファイルをチェックして、問題を引き起こす可能性のある設定ミスや不一致がないことを確認します。詳細は、Clair 設定の概要 を参照してください。

6.6.1.6. イメージメタデータを検査する

場合によっては、**Unsupported** メッセージが表示されることがあります。これは、スキャナーがイメージから必要なメタデータを抽出できないことを示している可能性があります。イメージメタデータが適切にフォーマットされており、アクセス可能であるかどうかを確認してください。

関連情報

詳細は、Clair のトラブルシューティング を参照してください。