Red Hat Quay 3.11

Red Hat Quay アーキテクチャー

Red Hat Quay アーキテクチャー

Last Updated: 2024-04-05

Red Hat Quay 3.11 Red Hat Quay アーキテクチャー

Red Hat Quay アーキテクチャー

法律上の通知

概要

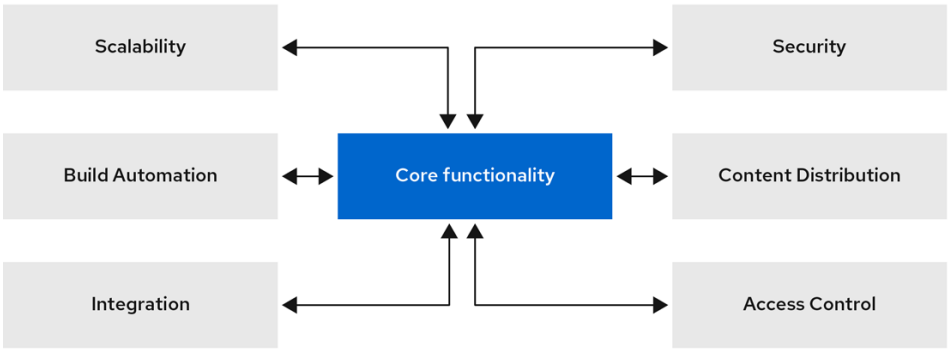
Red Hat Quay アーキテクチャー

目次

第1章 RED HAT QUAY の概要

Red Hat Quay は、企業向けの分散型で可用性の高いコンテナーイメージレジストリーです。

Red Hat Quay コンテナーレジストリープラットフォームは、安全なストレージ、配布、アクセス制御、地理的レプリケーション、リポジトリーのミラーリング、およびあらゆるインフラストラクチャー上のコンテナーとクラウドネイティブのアーティファクトのガバナンスを提供します。スタンドアロンコンポーネントまたは OpenShift Container Platform の Operator として利用でき、オンプレミスまたはパブリッククラウドにデプロイ可能です。



このガイドでは、Red Hat Quay をデプロイする際に使用するアーキテクチャーパターンを説明します。また、サイジングのガイダンスとデプロイメントの前提条件に加えて、Red Hat Quay レジストリーの高可用性を確保するためのベストプラクティスも説明します。

1.1. スケーラビリティーと高可用性 (HA)

Red Hat Quay で使用されるコードベースは、Red Hat によってホストされる高可用性コンテナーイメージレジストリーである Quay.io で使用されるコードベースと同じです。Quay.io と Red Hat Quay は、マルチテナント SaaS ソリューションを提供します。その結果、導入環境がオンプレミスであってもパブリッククラウド上であっても、確信を持って、高可用性を備えた大規模なデプロイメントを行うことができます。

1.2. コンテンツ配信

Red Hat Quay のコンテンツ配布機能には次のものが含まれます。

リポジトリーのミラーリング

Red Hat Quay リポジトリーのミラーリングを使用すると、Red Hat Quay および他のコンテナーレジストリー (JFrog Artifactory、Harbor、Sonatype Nexus Repository など) からのイメージを Red Hat Quay クラスターに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。

geo レプリケーション

Red Hat Quay の geo レプリケーションを使用すると、地理的に分散した複数の Red Hat Quay デプロイメントを、クライアントまたはユーザーの観点から単一のレジストリーとして機能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。

切断された環境またはエアギャップ環境でのデプロイメント

Red Hat Quay は、次のいずれか 2 つの方法で、切断された環境にデプロイできます。

- Red Hat Quay と Clair はインターネットに接続されており、エアギャップされた OpenShift Container Platform クラスターは、ファイアウォールのホワイトリストに明示的に登録したホールを介して Red Hat Quay レジストリーにアクセスします。
- 2 つの独立した Red Hat Quay および Clair インストールを使用します。1 つのインストールはインターネットに接続されており、もう 1 つは切断された環境またはファイアウォールで囲まれた環境内にあります。イメージおよび脆弱性データは、オフラインメディアを使用して、接続環境から切断環境に手動で転送されます。

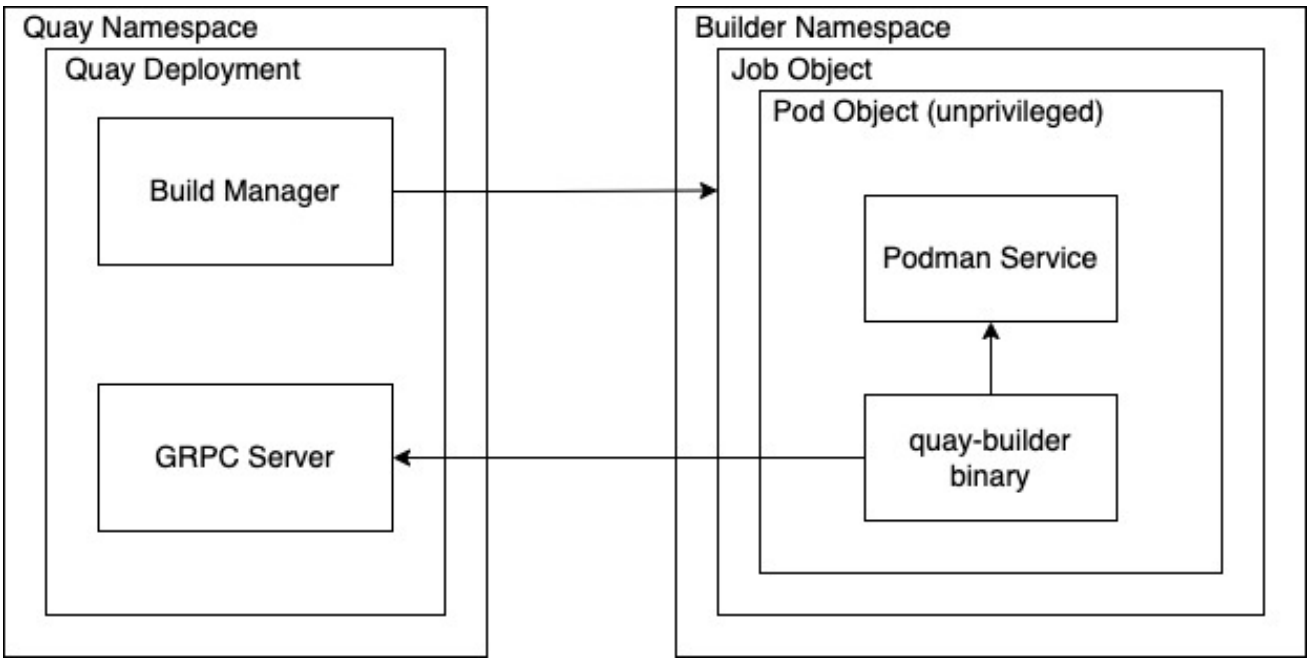
1.3. ビルドの自動化

Red Hat Quay は、OpenShift Container Platform または Kubernetes プラットフォーム上の一連のワーカーノードを使用した Dockerfile の構築をサポートします。GitHub webhook などのビルドトリガーを設定することで、新しいコードがコミットされたときに自動的に新しいバージョンのリポジトリーを構築できます。

Red Hat Quay 3.7 より前は、Red Hat Quay は Pod によって起動された仮想マシンで Podman コマンドを実行していました。仮想プラットフォーム上でビルドを実行するには、ネストされた仮想化を有効にする必要がありますが、これは Red Hat Enterprise Linux (RHEL) や OpenShift Container Platform には搭載されていません。その結果、ビルドはベアメタルクラスター上で実行する必要があり、リソースの使用が非効率になってしまいます。Red Hat Quay 3.7 では、この要件が削除され、仮想化プラットフォームまたはベアメタルプラットフォームで実行されている OpenShift Container Platform クラスター上でビルドを実行できるようになりました。

1.4. RED HAT QUAY の拡張ビルドアーキテクチャー

以下のイメージは、拡張ビルド機能の想定される設計フローとアーキテクチャーを示しています。



この機能拡張により、ビルドマネージャーは最初に **Job Object** を作成します。次に、**Job Object** は **quay-builder-image** を使用して Pod を作成します。 **quay-builder-image** には、**quay-builder binary** サービスおよび **Podman** サービスが含まれます。作成された Pod は **unprivileged** として実行されます。次に、**quay-builder binary** は、ステータスを通知してビルドマネージャーからビルド情報を取得しつつ、イメージをビルドします。

1.5. インテグレーション

Red Hat Quay は、ほぼすべての Git 互換システムと統合できます。Red Hat Quay は、GitHub、GitLab、または BitBucket の自動設定を提供し、ユーザーがコンテナー化されたソフトウェアを継続的に構築して提供できるようにします。

1.5.1. REST API

Red Hat Quay には、完全な OAuth 2、RESTful API が同梱されています。RESTful API には次の利点があります。

- URL (例: **https://quay-server.example.com/api/v1**) を使用して各 Red Hat Quay インスタンスのエンドポイントから利用できます。
- ユーザーがブラウザーを介してエンドポイントに接続し、Swagger で使用できる検出エンドポイントによって提供される Red Hat Quay 設定を **GET**、**DELETE**、**POST**、および **PUT** できるようにします。
- API は URL (**https://quay-server.example.com/api/v1** など) によって呼び出すことができ、JSON オブジェクトをペイロードとして使用します。

1.6. セキュリティー

Red Hat Quay は、コンテンツガバナンスとセキュリティーが 2 つの主要なフォーカスエリアである実際のエンタープライズユースケース向けに構築されます。

Red Hat Quay のコンテンツガバナンスとセキュリティーには、Clair を介した組み込みの脆弱性スキャンが含まれています。

1.6.1. TLS/SSL 設定

Red Hat Quay レジストリーの SSL/TLS は、設定ツール UI または設定バンドルで設定できます。データベース、イメージストレージ、および Redis への SSL/TSL 接続も、設定ツールを使用して指定できます。

データベース内および実行時の機密フィールドは、自動的に暗号化されます。ミラー操作中に、HTTPS を要求し、Red Hat Quay レジストリーの証明書を検証することもできます。

1.6.2. Clair

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

1.6.3. Red Hat Quay Operator のセキュリティー

Red Hat Quay Operator を使用して Red Hat Quay をデプロイすると、**tls** コンポーネントはデフォルトで **managed** 対象に設定され、OpenShift Container Platform の認証局を使用して HTTPS エンドポイントの作成と TLS 証明書のローテーションが行われます。

tls コンポーネントを **unmanaged** に設定すると、パススルールートにカスタム証明書を提供できますが、証明書のローテーションはユーザーが行います。

1.6.4. 完全に分離されたビルド

Red Hat Quay は、ベアメタルビルダーと仮想ビルダーの両方を使用する Dockerfile の構築をサポートするようになりました。

ベアメタルワーカーノードを使用することにより、各ビルドは一時的な仮想マシンで実行し、ビルド実行中の分離とセキュリティーが確保されます。これにより、不正なペイロードに対する最良の保護が提供されます。

コンテナー内で直接ビルドを実行すると、仮想マシンを使用する場合と同じように分離されませんが、それでも十分な保護が提供されます。

1.6.5. ロールベースのアクセス制御

Red Hat Quay は、ユーザーおよび自動化ツールによる読み取り、書き込み、および管理アクセスに対するきめ細かい権限により、組織およびチームごとにレジストリーのコンテンツを完全に分離します。

1.7. 最近追加された機能

最新の機能、機能拡張、非推奨、既知の問題は、Red Hat Quay リリースノート を参照してください。

第2章 RED HAT QUAY の前提条件

Red Hat Quay をデプロイする前に、イメージストレージ、データベース、および Redis をプロビジョニングする必要があります。

2.1. イメージストレージバックエンド

Red Hat Quay は、すべてのバイナリーブロブをストレージバックエンドに格納します。

ローカルストレージ

Red Hat Quay はローカルストレージと連携できますが、バイナリー BLOB の耐久性は保証できないため、これは概念実証またはテストセットアップにのみ使用してください。

HA ストレージのセットアップ

Red Hat Quay HA デプロイメントの場合は、次のような HA イメージストレージを提供する必要があります。

- **Red Hat OpenShift Data Foundation**(旧称 Red Hat OpenShift Container Storage) は、コンテナー向けのソフトウェア定義ストレージです。OpenShift Container Platform のデータおよびストレージサービスプラットフォームとして設計された Red Hat OpenShift Data Foundation は、チームがクラウド全体でアプリケーションを迅速かつ効率的に開発およびデプロイできるように支援します。詳細は、https://www.redhat.com/en/technologies/cloud-computing/openshift-data-foundation を参照してください。

- **Ceph Object Gateway** (RADOS Gateway とも呼ばれます) は、Red Hat Quay が必要とするオブジェクトストレージを提供できるストレージソリューションの例です。Ceph Storage を高可用性ストレージバックエンドとして使用する方法は、Quay の高可用性ガイド を参照してください。Red Hat Ceph Storage と HA セットアップの詳細は、Red Hat Ceph Storage アーキテクチャーガイド を参照してください。

geo レプリケーション

ローカルストレージは geo レプリケーションには使用できないため、サポートされているオンプレミスまたはクラウドベースのオブジェクトストレージソリューションをデプロイする必要があります。ローカライズされたイメージストレージは各リージョンで提供され、最も近くにある利用可能なストレージエンジンからイメージプルが提供されます。コンテナーイメージのプッシュは、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、バックグラウンドで他のストレージエンジンに複製されます。これには、すべてのリージョンからイメージストレージにアクセスできる必要があります。

2.1.1. サポートされているイメージストレージエンジン

Red Hat Quay は、次のオンプレミスストレージタイプをサポートします。

- Ceph/Rados RGW
- OpenStack Swift
- Red Hat OpenShift Data Foundation 4 (NooBaa 経由)

Red Hat Quay は、次のパブリッククラウドストレージエンジンをサポートします。

- Amazon Web Services (AWS) S3
- Google Cloud Storage

- Azure Blob Storage

2.1.2. サポート対象外のイメージストレージエンジン

現在、Hitachi HCP はサポートされていません。S3 の実装はそれぞれ異なるため、以前は Hitachi HCP で問題が発生しました。Ceph/RADOS ドライバーが使用されている場合、Hitachi HCP は動作する可能性がありますが、Red Hat Quay ですべてのシナリオで適切に動作することを保証できないため、こちらはサポートされません。

2.2. データベースバックエンド

Red Hat Quay は、設定情報を **config.yaml** ファイルに保存します。ユーザー情報、ロボットアカウント、チーム、権限、組織、イメージ、タグ、マニフェストなどのレジストリーメタデータは、データベースバックエンド内に保存されます。必要に応じて、ログを ElasticSearch にプッシュできます。PostgreSQL は Red Hat Quay と Clair の両方で使用できるため、データベースバックエンドとして推奨されます。

データベースバックエンドとして MySQL および MariaDB を使用するサポートは、Red Hat Quay 3.6 リリース以降非推奨となっていましたが、Red Hat Quay の今後のバージョンで削除されます。それまでは、MySQL は サポートマトリックス に従って引き続きサポートされますが、追加機能や明示的なテスト対象範囲は提供されません。Red Hat Quay Operator は、データベースを管理している場合に、PostgreSQL デプロイメントのみをサポートします。MySQL を使用する場合は、それを手動でデプロイし、データベースコンポーネントを **managed: false** に設定する必要があります。

Red Hat Quay を高可用性 (HA) 設定でデプロイするには、データベースサービスが高可用性向けにプロビジョニングされている必要があります。Red Hat Quay がパブリッククラウドインフラストラクチャー上で実行している場合は、クラウドプロバイダーが提供する PostgreSQL サービスを使用することを推奨しますが、MySQL もサポートされています。

Geo レプリケーションには、すべてのリージョンからアクセス可能な共有データベースが 1 つ必要です。

2.3. REDIS

Red Hat Quay は Redis キャッシュ内にビルダーログを保存します。保存されるデータは一時的なものであるため、Redis はステートフルであっても高可用性を必要としません。

Redis に障害が発生すると、ビルドログ、ビルダー、ガベージコレクターサービスにアクセスできなくなります。また、ユーザーイベントも利用できなくなります。

Red Hat Software Collections、またはその他の任意のソースから Redis イメージを使用できます。

第3章 RED HAT QUAY インフラストラクチャー

Red Hat Quay は、オンプレミスまたはパブリッククラウドの両方の、物理インフラストラクチャーまたは仮想インフラストラクチャーで実行できます。デプロイメントは、次のように単純なものから大規模なものまで多岐にわたります。

- 開発者ノートブックでのオールインワンセットアップ
- 仮想マシンまたは OpenShift Container Platform での高可用性
- 複数のアベイラビリティーゾーンおよびリージョンにまたがった地理的分散

3.1. スタンドアロンホストでの RED HAT QUAY の実行

Ansible または別の自動化スイートを使用して、スタンドアロンデプロイメントプロセスを自動化できます。すべてのスタンドアロンホストには、有効な Red Hat Enterprise Linux (RHEL) サブスクリプションが必要です。

概念実証の導入

Red Hat Quay は、イメージストレージ、コンテナー化されたデータベース、Redis、およびオプションで Clair セキュリティースキャンを備えたマシン上で実行されます。

可用性の高いセットアップ

Red Hat Quay と Clair は、複数のホストのコンテナー内で実行されます。**systemd** ユニットを使用すると、失敗時の再起動、通常の再起動を確実に行うことができます。
スタンドアロンのホストで高可用性をセットアップするには、低レベルの TCP ロードバランサーや TLS を終端できるアプリケーションロードバランサーなど、ロードバランサーを用意する必要があります。

3.2. OPENSHIFT での RED HAT QUAY の実行

Red Hat Quay Operator for OpenShift Container Platform は、以下の機能を提供します。

- カスタマイズオプションを使用した Red Hat Quay の自動デプロイメントと管理
- Red Hat Quay とそのすべての依存関係の管理
- 自動スケーリングおよび更新
- GitOps、モニタリング、アラート、ロギングなどの既存の OpenShift Container Platform プロセスとの統合
- Red Hat OpenShift Data Foundation (ODF) Operator の一部として、マルチクラウドオブジェクトゲートウェイ (NooBaa) によってサポートされる、可用性が制限されたオブジェクトストレージの提供。このサービスには追加のサブスクリプションは必要ありません。
- ODF Operator が提供するスケールアウトされた高可用性オブジェクトストレージ。このサービスには追加のサブスクリプションが必要です。

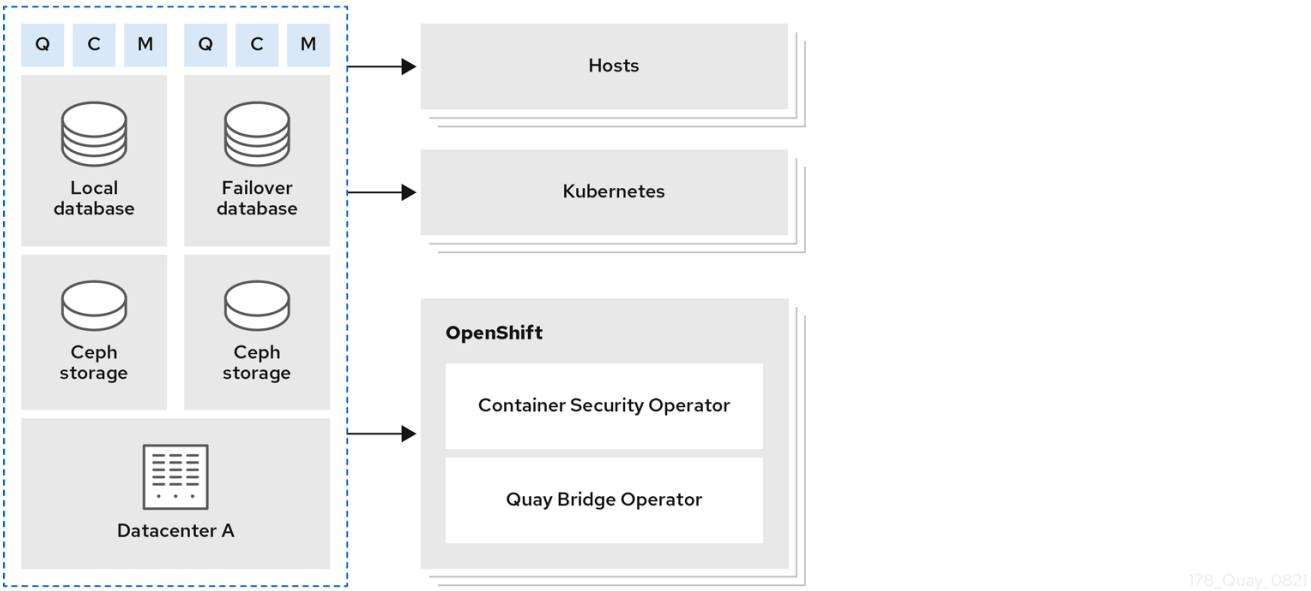
Red Hat Quay は、OpenShift Container Platform インフラストラクチャーノード上で実行できます。そのため、それ以上のサブスクリプションは必要ありません。OpenShift Container Platform 上で Red Hat Quay を実行すると、次の利点があります。

- **Zero to Hero:** Red Hat Quay および関連コンポーネントのデプロイが簡素化されているため、すぐに製品を使い始めることができます。
- **スケーラビリティー:** クラスターのコンピューティング機能を使用して、実際の負荷に基づいた自動スケーリングを通じて需要を管理します。
- **簡素化されたネットワーキング:** OpenShift Container Platform TLS 証明書とルートを使用した HTTPS 経由で保護されたロードバランサーとトラフィック入力の自動プロビジョニング。
- **宣言的な設定管理:** GitOps に適したライフサイクル管理のために CustomResource オブジェクトに格納された設定。
- **再現性:** Red Hat Quay と Clair のレプリカの数に関係なく一貫性。
- **OpenShift 統合:** OpenShift Container Platform のモニタリングおよびアラート機能を使用して、単一クラスター上の複数の Red Hat Quay デプロイメントを管理するための追加サービス。

3.3. スタンドアロン RED HAT QUAY と OPENSHIFT CONTAINER PLATFORM の統合

Red Hat Quay Operator は、OpenShift Container Platform 上で実行される Red Hat Quay のシームレスなデプロイメントと管理を保証しますが、Red Hat Quay をスタンドアロンモードで実行し、実行されている場所に関係なく、1 つまたは複数の OpenShift Container Platform クラスターにコンテンツを提供することもできます。

スタンドアロン Red Hat Quay と OpenShift Container Platform の統合



以下のような、Red Hat Quay のスタンドアロンおよび Operator ベースのデプロイメントを OpenShift Container Platform と統合するために、いくつかの Operator が利用可能です。

Red Hat Quay Cluster Security Operator

Red Hat Quay の脆弱性スキャン結果を OpenShift Container Platform コンソールに中継します

Red Hat Quay Bridge Operator

Red Hat Quay と OpenShift Container Platform を OpenShift Container Platform Builds および ImageStreams と組み合わせて使用することで、シームレスな統合とユーザーエクスペリエンスを確保します

3.4. RED HAT OPENSHIFT 導入用のミラーレジストリー

Red Hat Openshift 導入用のミラーレジストリーは、Red Hat Quay の小規模バージョンであり、切断されたインストール用の OpenShift Container Platform の必要なコンテナーイメージをミラーリングするためのターゲットとして使用できます。

OpenShift Container Platform の切断されたデプロイメントの場合に、クラスターのインストールを実行するためにコンテナーレジストリーが必要です。このようなクラスターで実稼働レベルのレジストリーサービスを実行するには、別のレジストリーデプロイメントを作成して最初のクラスターをインストールする必要があります。**Red Hat Openshift 導入用のミラーレジストリー**は、このニーズに対応し、すべての OpenShift Container Platform サブスクリプションに含まれています。これは、OpenShift コンソールの **ダウンロード** ページからダウンロードできます。

Red Hat Openshift 導入用のミラーレジストリーを使用すると、ユーザーは、 **mirror-registry** コマンドラインインターフェイス (CLI) ツールを使用して、Red Hat Quay の小規模バージョンとその必要なコンポーネントをインストールできます。**Red Hat Openshift 導入用のミラーレジストリー**は、事前設定されたローカルストレージとローカルデータベースを使用して自動的にデプロイされます。また、このレジストリーには、自動生成されたユーザー認証情報とアクセス許可も含まれており、単一の入力セットを使用するだけで開始でき、追加の設定を選択する必要はありません。

Red Hat Openshift のミラーレジストリーは、事前に決定されたネットワーク設定を提供し、成功時にデプロイされたコンポーネントの認証情報とアクセス URL を報告します。完全修飾ドメイン名 (FQDN) サービス、スーパーユーザー名とパスワード、カスタム TLS 証明書などのオプションの設定入力のセットも少しだけ含まれています。これにより、ユーザーはコンテナーレジストリーを利用できるため、制限されたネットワーク環境で OpenShift Container Platform を実行するときに、すべての OpenShift Container Platform リリースコンテンツのオフラインミラーを簡単に作成できます。

Red Hat Openshift 導入用のミラーレジストリーは、リリースイメージや Operator イメージなど、切断された OpenShift Container Platform クラスターのインストールに必要なイメージのホスティングに限定されます。ローカルストレージを使用します。お客様が作成したコンテンツは、**Red Hat Openshift のミラーレジストリー** でホストしないでください。

Red Hat Quay とは異なり、**Red Hat OpenShift のミラーレジストリー**は高可用性レジストリーではありません。ローカルファイルシステムストレージのみがサポートされます。クラスターのグループの更新時にクラスターが複数あると単一障害点を生み出す可能性があるため、複数のクラスターで **Red Hat Openshift のミラーレジストリー** を使用することは推奨しません。**Red Hat OpenShift のミラーレジストリー** を使用して、OpenShift Container Platform コンテンツを他のクラスターに提供できる Red Hat Quay などの実稼働グレードの高可用性レジストリーをホストできるクラスターをインストールすることを推奨します。

詳細は、**Red Hat OpenShift のミラーレジストリー** を使用したミラーレジストリーの作成 を参照してください。

3.5. 複数のレジストリーと比較した単一レジストリー

多くのユーザーは、複数の個別のレジストリーを実行することを検討しています。Red Hat Quay で推奨されるアプローチは、単一の共有レジストリーを持つことです。

- 開発イメージと実稼働イメージを明確に分離する場合、またはコンテンツの出所によって明確に分離する場合 (サードパーティーのイメージを内部のものと区別するなど) は、組織とリポジトリーをロールベースのアクセス制御 (RBAC) と組み合わせて目的の分離を実現するために使用できます。

- イメージレジストリーが企業環境の重要なコンポーネントであることを前提として、個別のデプロイメントを使用して、レジストリーソフトウェアの新しいバージョンへのアップグレードをテストする場合があります。Red Hat Quay Operator は、パッチリリースおよびマイナーまたはメジャー更新のレジストリーを更新します。つまり、複雑な手順はすべて自動化されているため、アップグレードをテストするためにレジストリーのインスタンスを複数プロビジョニングする必要はありません。

- Red Hat Quay では、デプロイするクラスターごとに個別のレジストリーを用意する必要はありません。Red Hat Quay は Quay.io で大規模に動作することが証明されており、何千ものクラスターにコンテンツを提供できます。

- 複数のデータセンターにデプロイしている場合でも、単一の Red Hat Quay インスタンスを使用して、物理的に近い複数のデータセンターにコンテンツを提供したり、HA 機能とロードバランサーを使用してデータセンター全体に拡張したりできます。または、Red Hat Quay の geo レプリケーション機能を使用して、物理的に離れたデータセンター間で拡張することもできます。これには、グローバルロードバランサーまたは DNS ベースの geo-aware ロードバランシングのプロビジョニングが必要です。

- 複数の異なるレジストリーを実行することが適切なシナリオの 1 つは、レジストリーごとに異なる設定を指定する場合です。

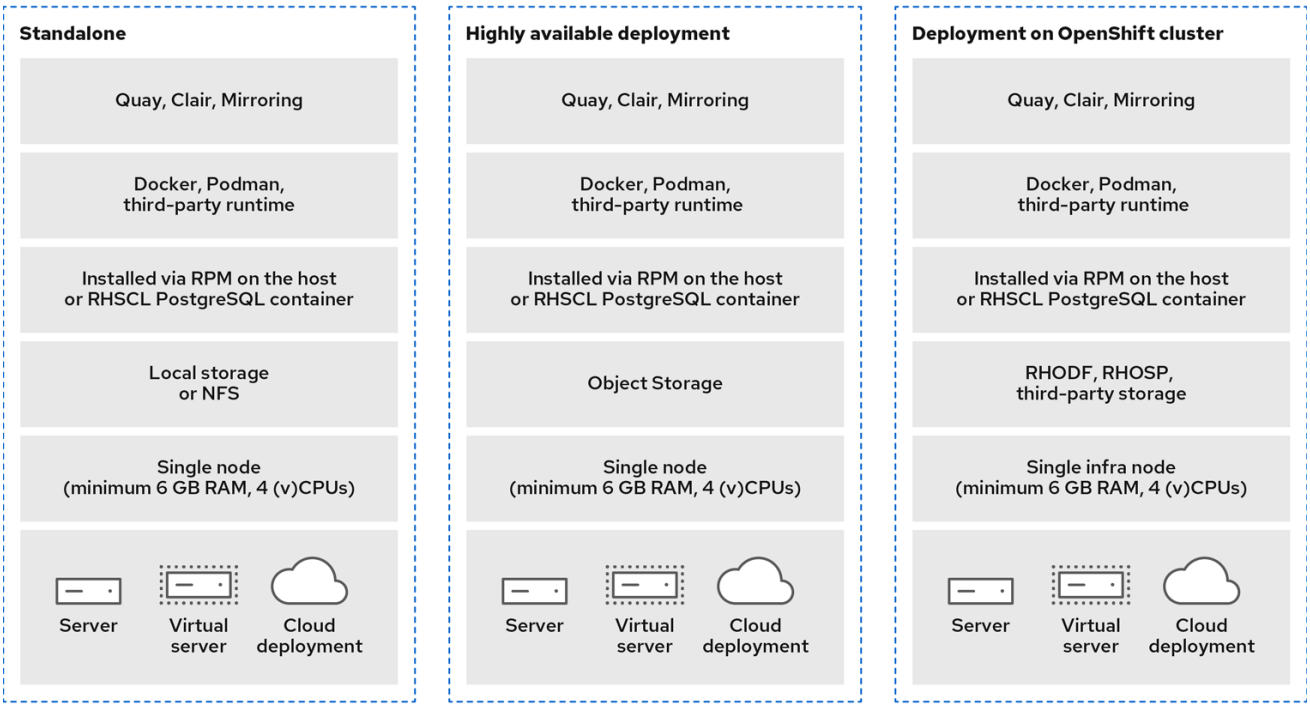
要約すると、共有レジストリーを実行すると、ストレージ、インフラストラクチャー、および運用コストを節約できますが、特定の状況では専用のレジストリーが必要になる場合があります。

第4章 RED HAT QUAY オンプレミスのデプロイ

次の図は、次の種類のデプロイメントのオンプレミス設定の例を示しています。

- スタンドアロンの概念実証
- 複数のホストでの高可用性デプロイメント
- Red Hat Quay Operator を使用した OpenShift Container Platform クラスターへのデプロイメント

オンプレミスの設定例

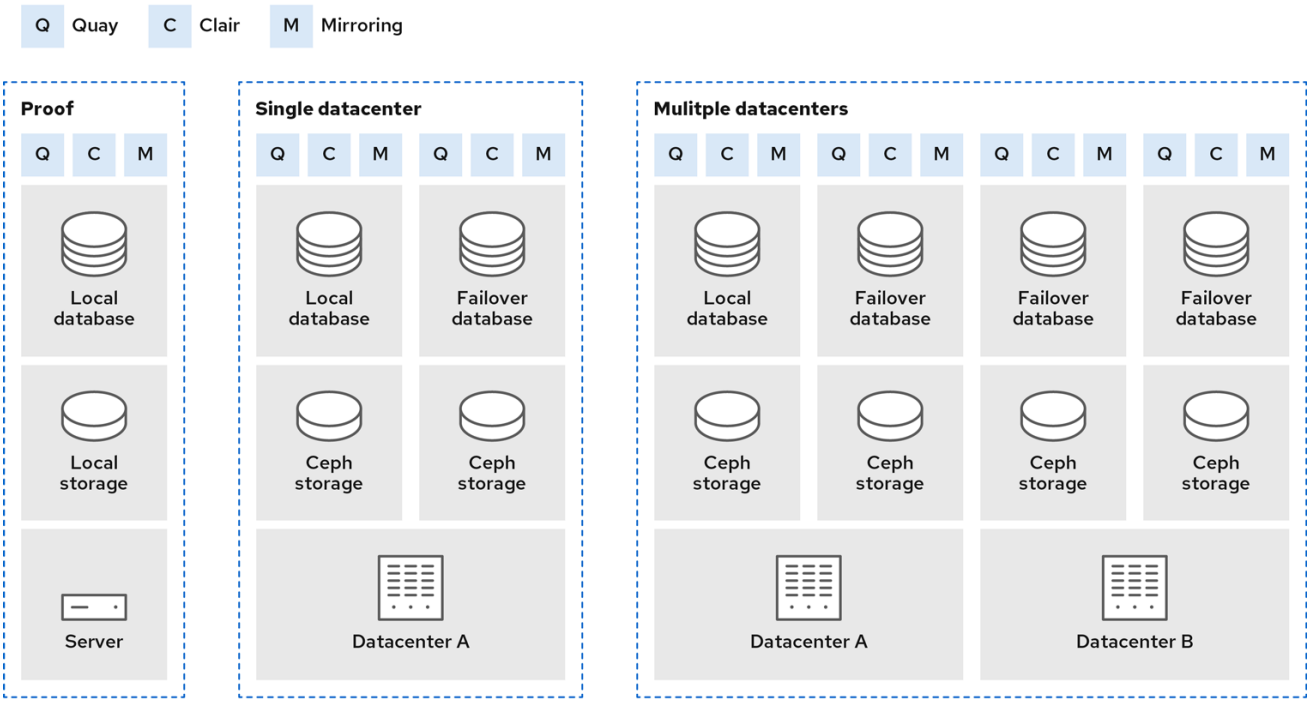


178_Quay_0821

4.1. RED HAT QUAY デプロイメントの例

以下の図は、Red Hat Quay の 3 つの可能なデプロイメントを示しています。

デプロイメントの例



178_Quay_0821

概念実証

Red Hat Quay、Clair、ミラーリングを単一ノードで実行し、ローカルイメージストレージとローカルデータベースを使用する

単一のデータセンター

高可用性 Red Hat Quay、Clair、およびミラーリングを複数のノードで実行し、HA データベースとイメージストレージを使用する

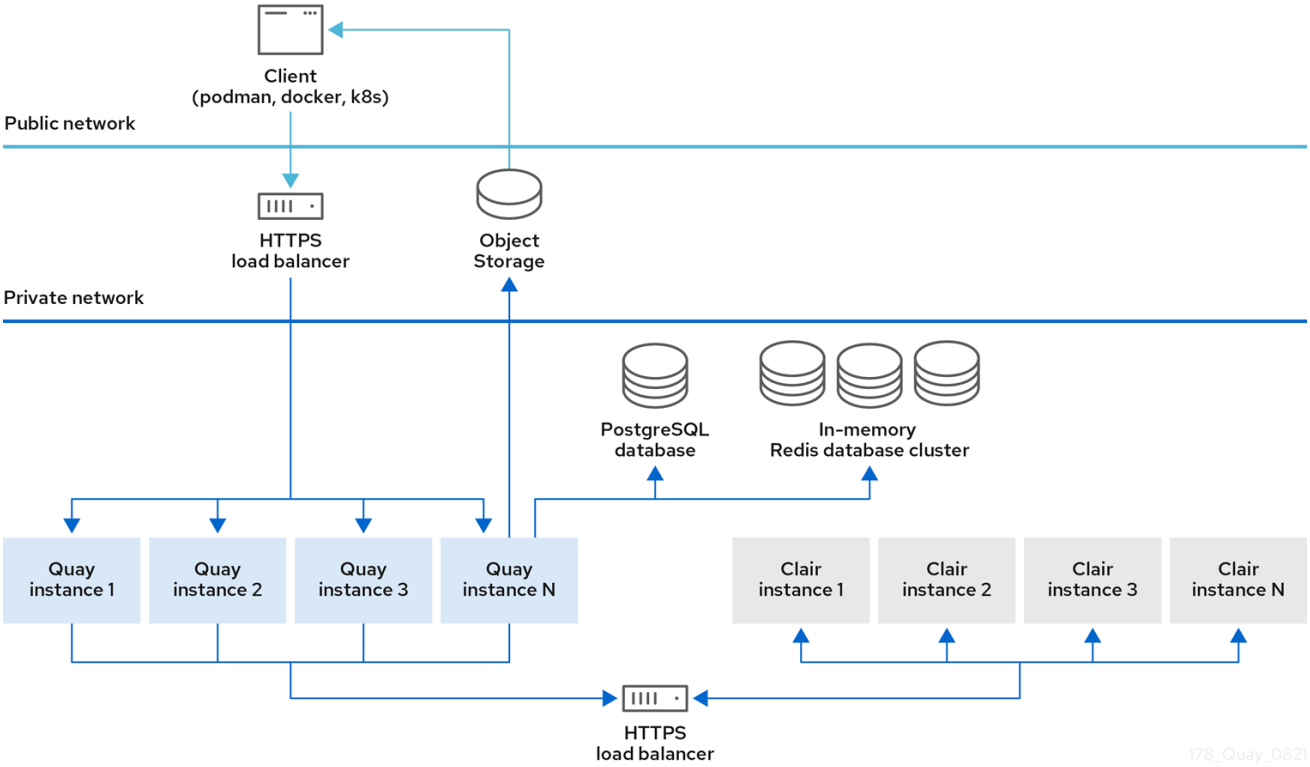
複数のデータセンター

高可用性 Red Hat Quay、Clair、およびミラーリングを複数のデータセンターの複数のノードで実行し、HA データベースとイメージストレージを使用する

4.2. RED HAT QUAY デプロイメントトポロジー

以下の図は、Red Hat Quay デプロイメントトポロジーの概要を示しています。

Red Hat Quay デプロイメントトポロジー

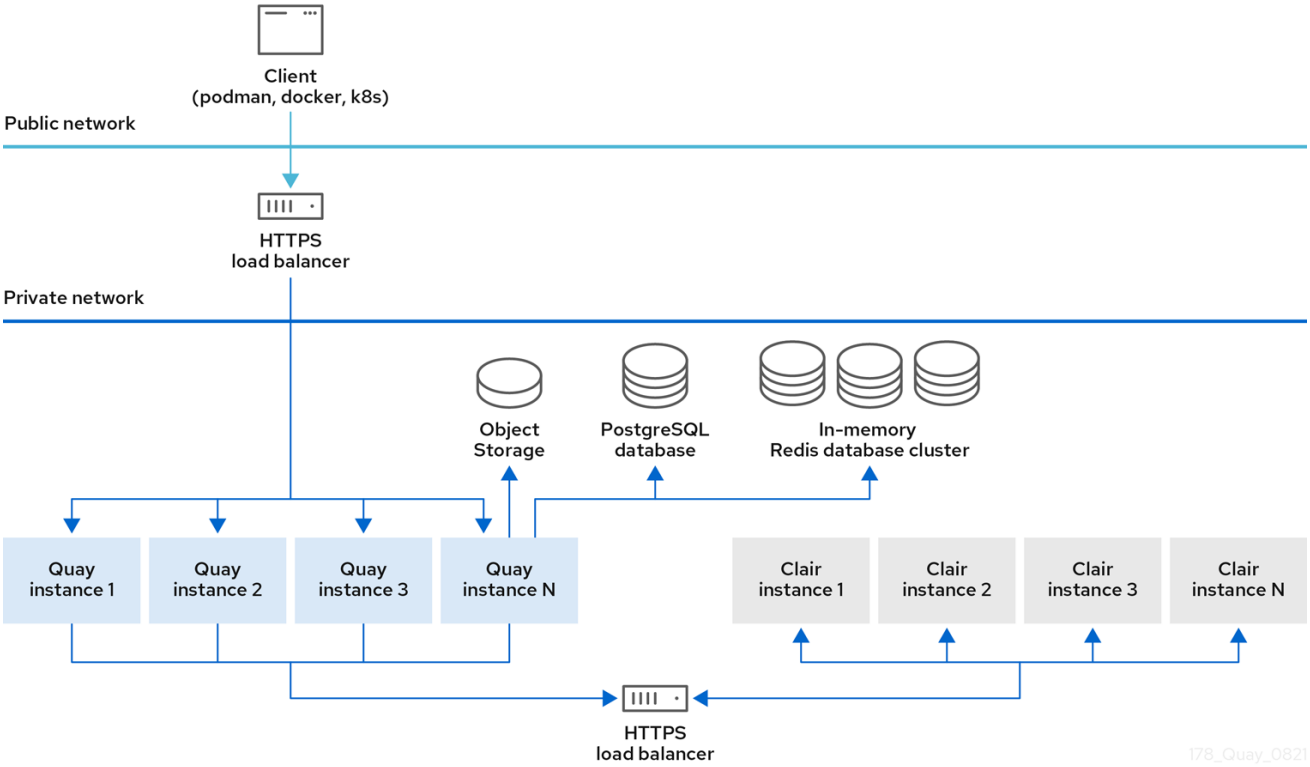


このデプロイメントでは、パブリック Red Hat Quay エンドポイントによって、すべてのプッシュ、ユーザーインターフェイス、および API リクエストを受け取ります。プルは **object storage** から直接提供されます。

4.3. ストレージプロキシーを使用した RED HAT QUAY デプロイメントトポロジー

以下の図は、ストレージプロキシーが設定された Red Hat Quay デプロイメントトポロジーの概要を示しています。

ストレージプロキシーを使用した Red Hat Quay デプロイメントトポロジー



ストレージプロキシーが設定されている場合、すべてのトラフィックはパブリック Red Hat Quay エンドポイントを通過します。

第5章 パブリッククラウドでの RED HAT QUAY の実行

Red Hat Quay は、パブリッククラウド上で実行できます。これはスタンドアロンモード、または OpenShift Container Platform 自体がパブリッククラウド上にデプロイされた場所のいずれかになります。テスト済みおよびサポート対象の設定の全一覧は、Red Hat Quay **テスト済みのインテグレーションマトリクス** (https://access.redhat.com/articles/4067991) を参照してください。

推奨事項: Red Hat Quay がパブリッククラウドで実行されている場合は、Red Hat Quay バックエンドサービスにパブリッククラウドサービスを使用して、適切な HA とスケーラビリティーを確保する必要があります。

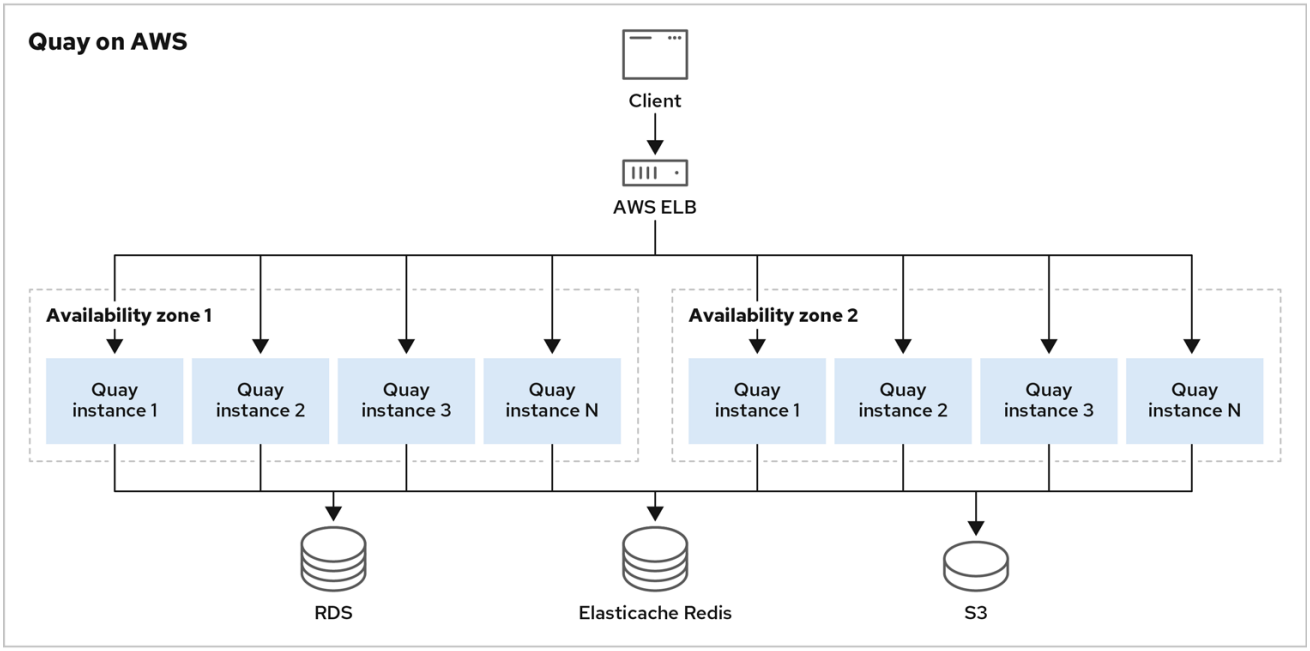
5.1. AMAZON WEB SERVICES での RED HAT QUAY の実行

Red Hat Quay が Amazon Web Services (AWS) 上で実行されている場合は、次の機能を使用できます。

- AWS Elastic Load Balancer
- AWS S3 (hot) blob ストレージ
- AWS RDS データベース
- AWS ElastiCache Redis
- EC2 仮想マシンの推奨: M3.Large または M4.XLarge

次の図は、AWS 上で実行されている Red Hat Quay の概要を示しています。

AWS の Red Hat Quay



5.2. MICROSOFT AZURE での RED HAT QUAY の実行

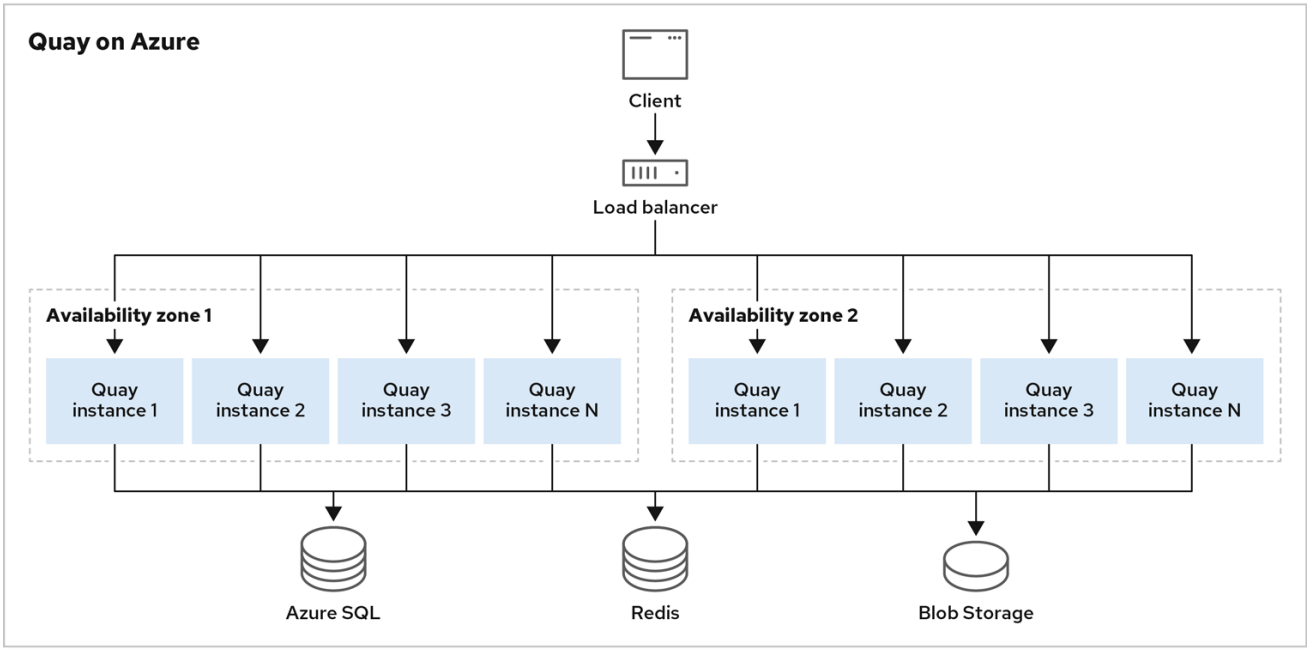
Red Hat Quay が Microsoft Azure 上で実行されている場合は、次の機能を使用できます。

- 高可用性 PostgreSQL などの Azure マネージドサービス

- Azure Blob Storage はホットストレージである必要がある
  - Azure Cool Storage は Red Hat Quay では利用できない
- Redis の Azure キャッシュ

次の図は、Microsoft Azure 上で実行される Red Hat Quay の概要を示しています。

Microsoft Azure の Red Hat Quay



178_Quay_0821

第6章 RED HAT QUAY によるコンテンツ配信

Red Hat Quay のコンテンツ配信機能は以下のとおりです。

- リポジトリーのミラーリング
- geo レプリケーション
- エアギャップ環境でのデプロイメント

6.1. リポジトリーのミラーリング

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

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

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

ミラーリング機能を使用するには、次のアクションを実行する必要があります。

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

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。

6.1.1. リポジトリーミラーリングの使用

次のリストは、Red Hat Quay リポジトリーのミラーリングの機能と制限を示しています。

- リポジトリーのミラーリングでは、リポジトリー全体をミラーリングしたり、同期するイメージを選択的に制限したりできます。フィルターは、コンマ区切りのタグのリスト、タグの範囲、または Unix シェルスタイルのワイルドカードを使用してタグを識別するその他の手段に基づくことができます。詳細は、ワイルドカード のドキュメントを参照してください。
- ミラーリングされたリポジトリーとして設定した場合は、そのリポジトリーに他のイメージを手動で追加できません。
- ミラーリングされたリポジトリーは設定したリポジトリーとタグに基づいているため、リポジトリーとタグのペアで表されるコンテンツのみが保持されます。たとえば、タグを変更してリポジトリー内の一部のイメージが一致しなくなると、それらのイメージは削除されます。

- 指定されたロボットだけが、ミラーリングされたリポジトリーにイメージをプッシュすることができ、リポジトリーに設定されたロールベースのアクセスコントロール権限に優先します。
- ミラーリングは、障害時にロールバックする **または**、ベストエフォートベースで実行するように設定できます。
- ミラーリングされたリポジトリーでは、**read** 権限を持つユーザーはリポジトリーからイメージをプルできますが、イメージをリポジトリーにプッシュすることはできません。
- ミラーリングされたリポジトリーの設定の変更は、Red Hat Quay ユーザーインターフェイスで、作成したミラーリングされたリポジトリーの **Repositories → Mirrors** タブを使用して実行できます。
- イメージは一定の間隔で同期されますが、必要に応じて同期することもできます。

6.1.2. リポジトリーのミラーリングの推奨事項

リポジトリーミラーリングのベストプラクティスには次のようなものがあります。

- リポジトリーミラーリング Pod は任意のノードで実行できます。これは、Red Hat Quay がすでに実行されているノードでミラーリングを実行できることを意味します。
- リポジトリーのミラーリングは、データベースでスケジュールされ、一括して実行されます。その結果、リポジトリーワーカーは各リポジトリーミラー設定ファイルをチェックし、次の同期が必要なタイミングを読み取ります。ミラーワーカーが増えると、より多くのリポジトリーを同時にミラーリングできるようになります。たとえば、10 個のミラーワーカーを実行すると、ユーザーは 10 個のミラーリング operator を並行して実行できることになります。ミラー設定が 10 個あるワーカーが 2 つしかない場合に、実行できる Operator は 2 つのみです。
- ミラーリング Pod の最適な数は、次の条件によって異なります。
  - ミラーリングされるリポジトリーの合計数
  - リポジトリー内のイメージやタグの数と変更の頻度
  - 並列バッチ処理
    たとえば、タグが 100 個あるリポジトリーをミラーリングしている場合に、このミラーは 1 つのワーカーで完了されます。ユーザーは、並行してミラーリングするリポジトリーの数を検討し、それに基づいてワーカーの数を決定する必要があります。

    同じリポジトリー内の複数のタグを並行してミラーリングすることはできません。

6.1.3. ミラーリングのイベント通知

リポジトリーミラーリングには、3 つの通知イベントがあります。

- リポジトリーミラーリングの開始
- リポジトリーのミラーリングの成功
- リポジトリーミラーリングの失敗

イベントは各リポジトリーの **Settings** タブ内で設定でき、電子メール、Slack、Quay UI、Webhook などの既存の通知方法がすべてサポートされています。

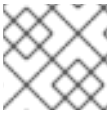
6.1.4. ミラーリング API

Red Hat Quay API を使用して、リポジトリーミラーリングを設定できます。

ミラーリング API

詳細は、Red Hat Quay API ガイド に記載されています。

6.2. 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。

6.2.1. 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。

6.2.2. 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

- 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。

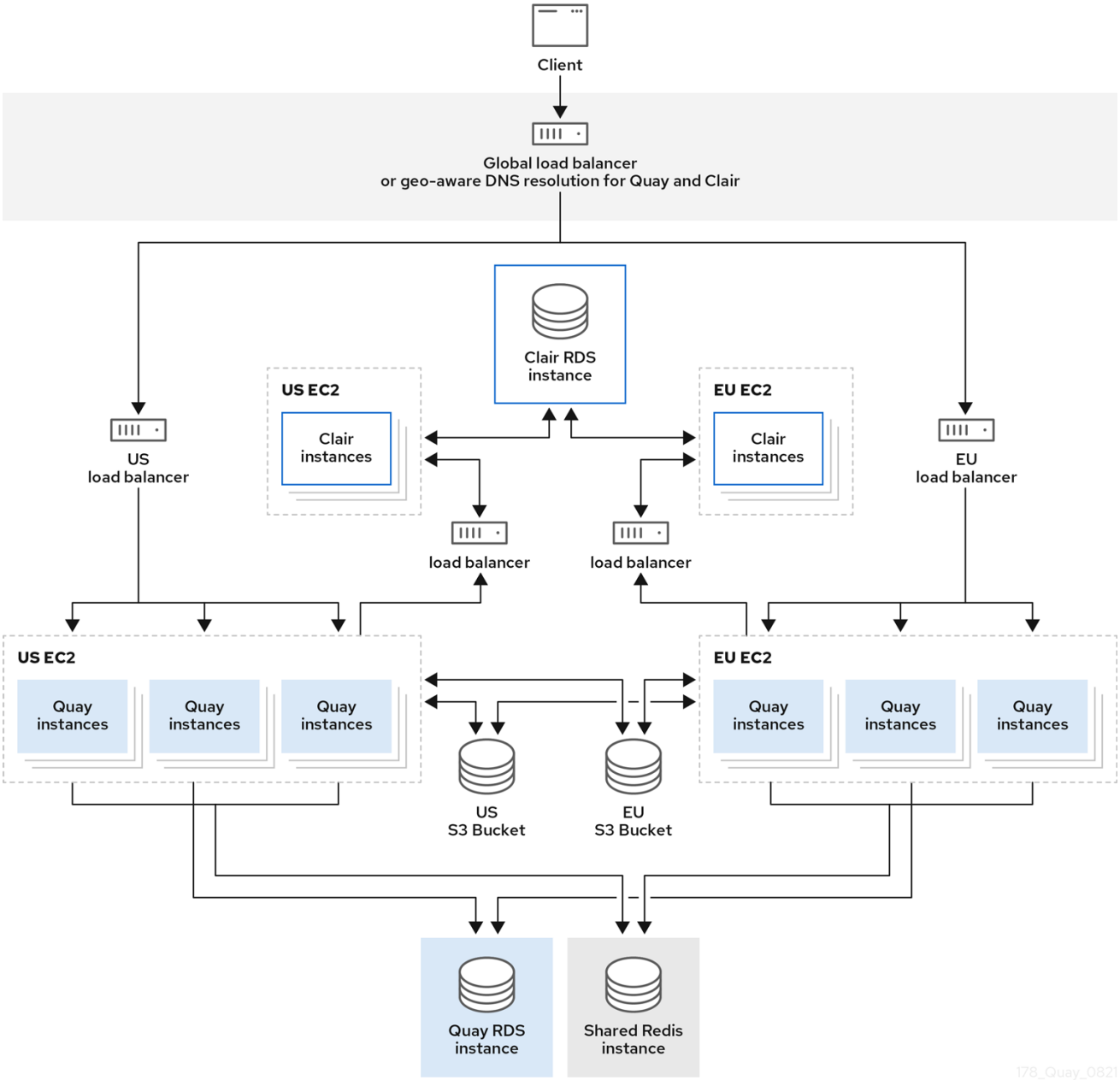
6.2.3. スタンドアロン Red Hat Quay を使用した Geo レプリケーション

次の図では、Red Hat Quay が 2 つの別々のリージョンでスタンドアロンで実行されており、共通のデータベースと共通の Redis インスタンスが使用されています。ローカライズされたイメージストレージは各リージョンで提供され、最も近くにある利用可能なストレージエンジンからイメージプルが提供されます。コンテナーイメージのプッシュは、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、バックグラウンドで他のストレージエンジンに複製されます。

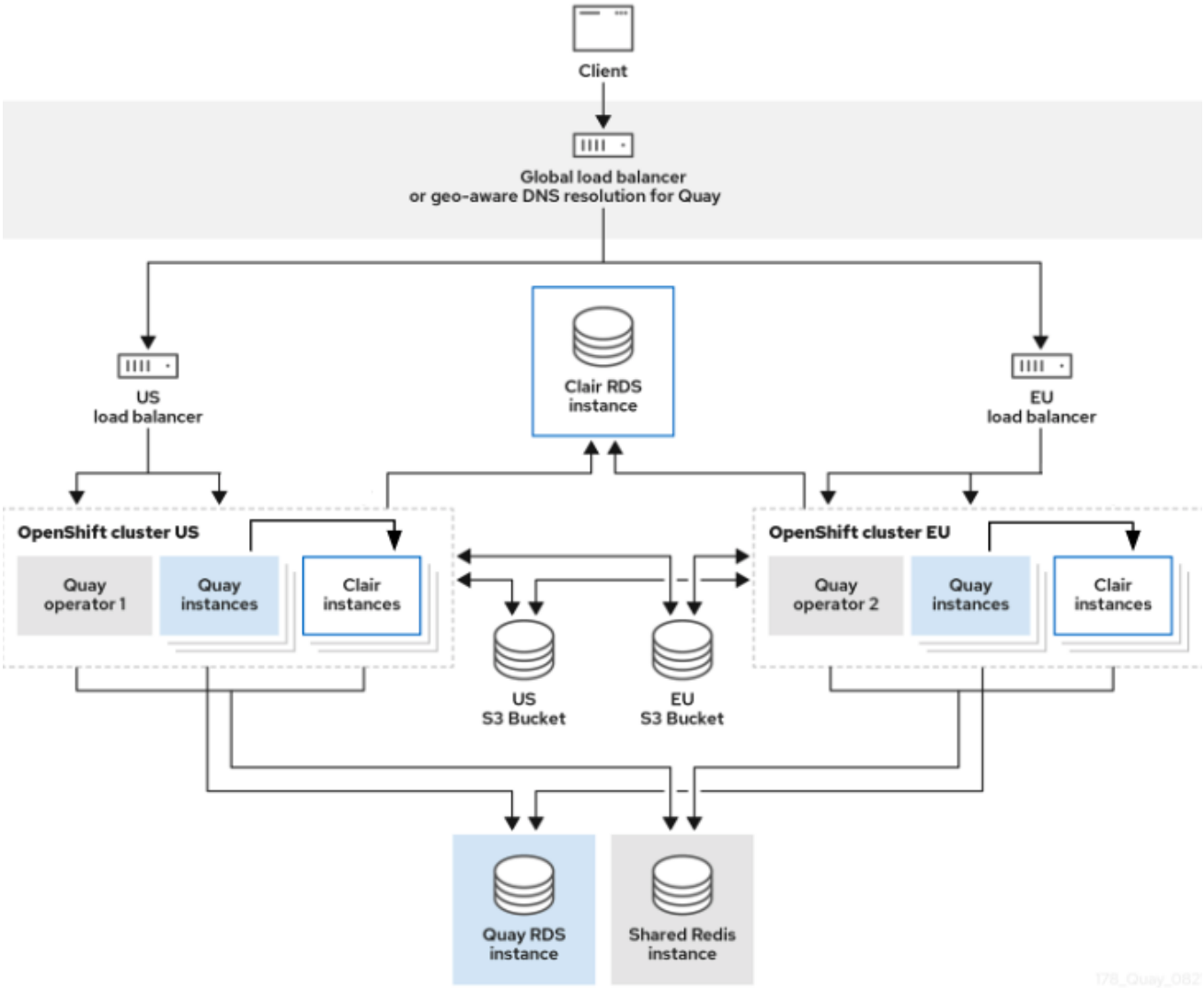
注記

1 つのクラスター (たとえば、US クラスター) で Clair に障害が発生した場合、米国のユーザーには Red Hat Quay で 2 番目のクラスター (EU) の脆弱性レポートが表示されません。これは、すべての Clair インスタンスの状態が同じであるためです。Clair に障害が発生した場合、通常はクラスター内の問題が原因です。

Geo レプリケーションのアーキテクチャー



6.2.4. Red Hat Quay Operator を使用した Geo レプリケーション



上記の例では、Red Hat Quay Operator は、共通のデータベースと共通の Redis インスタンスを使用して、2 つの別々のリージョンにデプロイされています。ローカライズされたイメージストレージは各リージョンで提供され、最も近くにある利用可能なストレージエンジンからイメージプルが提供されます。コンテナーイメージのプッシュは、Quay インスタンスの推奨ストレージエンジンに書き込まれ、次にバックグラウンドで他のストレージエンジンに複製されます。

Operator は現在、Clair セキュリティースキャナーとそのデータベースを別々に管理しているため、Geo レプリケーションの設定を活用して、Clair データベースを管理しないようにできます。代わりに、外部共有データベースが使用されます。Red Hat Quay と Clair は、PostgreSQL のいくつかのプロバイダーとベンダーをサポートします。これらは Red Hat Quay 3.x test matrix にあります。さらに、Operator は、デプロイメントに挿入できるカスタム Clair 設定もサポートします。これにより、ユーザーは外部データベースの接続認証情報を使用して Clair を設定できます。

6.2.5. 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. GEO レプリケーションと比較したリポジトリーミラーリング

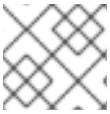
Red Hat Quay の geo レプリケーションは、データベースが共有されている間に、2 つ以上の異なるストレージバックエンド間でイメージストレージのバックエンドデータ全体をミラーリングします (たとえば、2 つの異なる blob ストレージエンドポイントを持つ 1 つの Red Hat Quay レジストリー)。geo レプリケーションの主なユースケースには、次のようなものがあります。

- 地理的に分散する設定向けのバイナリー Blob へのアクセス速度を上げる
- イメージのコンテンツがリージョン全体で同じであることを保証する

リポジトリーのミラーリングは、選択されたリポジトリー (またはリポジトリーのサブセット) をあるレジストリーから別のレジストリーに同期します。レジストリーは固有であり、各レジストリーには個別のデータベースおよび個別のイメージストレージがあります。

ミラーリングの主な使用例は次のとおりです。

- 異なるデータセンターまたはリージョンでの独立したレジストリーのデプロイメント。ここでは、全コンテンツの特定サブセットがデータセンター/リージョン間で共有されることになっています。
- 外部レジストリーからローカル Red Hat Quay デプロイメントへの選択された (許可リスト化された) アップストリームリポジトリーの自動同期またはミラーリング。



注記

リポジトリーのミラーリングと Geo レプリケーションを同時に使用できます。

表6.1 Red Hat Quay リポジトリーのミラーリングと geo レプリケーションの比較

機能と性能	geo レプリケーション	リポジトリーのミラーリング
設計されている機能	グローバルに共有されるレジストリー	独立した異なるレジストレーション
レプリケーションまたはミラーリングがまだ完了していない場合はどうなるか?	リモートコピーを使用する (遅くなります)	イメージが表示されない
両地域のすべてのストレージバックエンドへのアクセスが必要か?	はい (すべての Red Hat Quay ノード)	いいえ (個別のストレージ)
ユーザーは両方のサイトのイメージを同じリポジトリーにプッシュできるか?	はい	いいえ

機能と性能	geo レプリケーション	リポジトリーのミラーリング
すべてのレジストリーの内容と設定がすべての地域で同一であるかどうか (共有データベース)?	はい	いいえ
ユーザーはミラーリングする個別の namespace またはリポジトリーを選択できるか?	いいえ	はい
ユーザーは同期ルールにフィルターを適用できるか?	いいえ	はい
各地域で許可されている個々の/異なるロールベースのアクセス制御設定があるか?	いいえ	はい

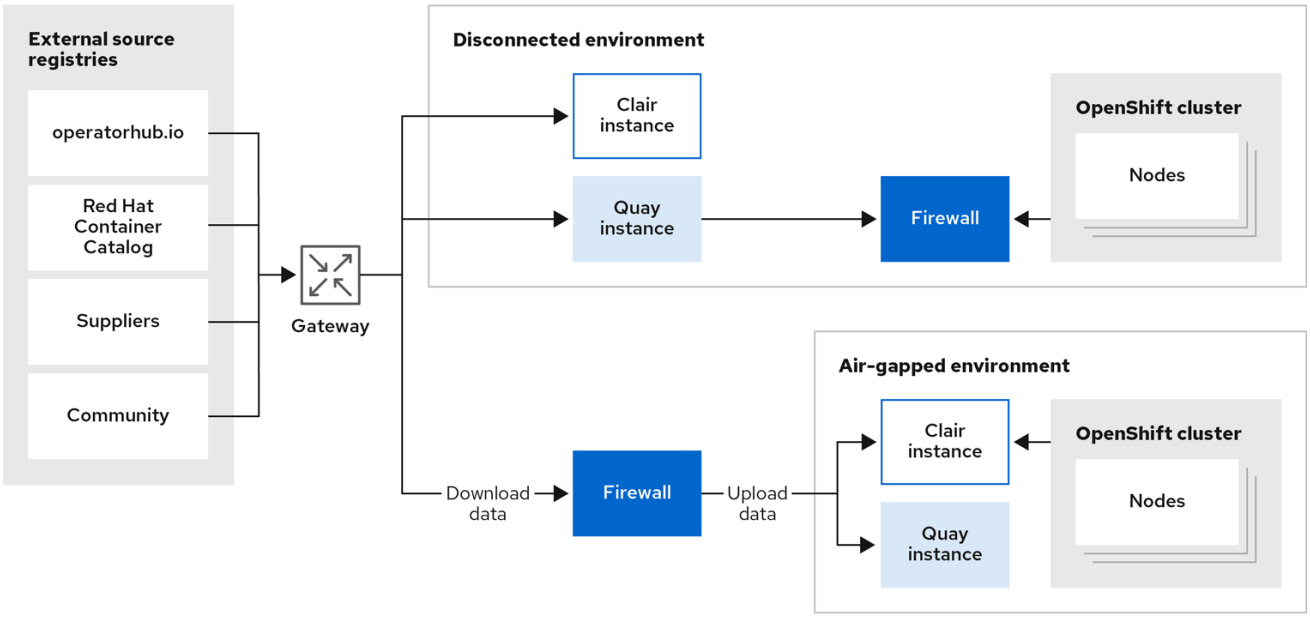
6.4. エアギャップまたは切断されたデプロイメント

次の図では、図の上部のデプロイメントでは、インターネットに接続された Red Hat Quay と Clair が示されており、エアギャップされた OpenShift Container Platform クラスターは、ファイアウォールの明示的なホワイトリストに登録されたホールを介して Red Hat Quay レジストリーにアクセスしています。

図の下側のデプロイメントでは、Red Hat Quay と Clair がファイアウォール内で動作しており、イメージや CVE データはオフラインメディアを使用してターゲットシステムに転送されています。データは、インターネットに接続されている別の Red Hat Quay および Clair デプロイメントからエクスポートされます。

次の図は、Red Hat Quay と Clair がエアギャップまたは切断された環境にどのようにデプロイされるかを示しています。

切断された環境またはエアギャップ環境の Red Hat Quay および Clair


External source registries
operatorhub.io
Red Hat Container Catalog
Suppliers
Community
Gateway
Disconnected environment
Clair instance
Quay instance
Firewall
OpenShift cluster
Nodes
Air-gapped environment
Download data
Firewall
Upload data
Clair instance
Quay instance
OpenShift cluster
Nodes
178_Quay_0821

第7章 RED HAT QUAY のサイジングとサブスクリプション

Red Hat Quay のスケーラビリティはその重要な強みの 1 つであり、単一のコードベースで次のような幅広い規模のデプロイメントをサポートします。

- 単一の開発マシン上での概念実証のデプロイメント
- 数十台の Kubernetes クラスターにコンテンツを配信できる約 2,000 ユーザーの中規模デプロイメント
- 世界中の数千の Kubernetes クラスターにサービスを提供できる Quay.io などのハイエンドデプロイメント

サイズ調整は、ユーザー、イメージ、同時プルおよびプッシュの数など、さまざまな要因に大きく依存するため、標準的なサイジングの推奨値はありません。

以下は、Red Hat Quay を実行するシステムの最小要件です (コンテナー/Pod インスタンスごと)。

- **Quay:** 最小 6 GB; 8 GB、さらに 2 つの vCPU を推奨
- **Clair:** 2 GB RAM と 2 つ以上の vCPU を推奨
- **Storage:** : 推奨 30 GB
- **NooBaa:** 最小 2 GB、1 vCPU (**objectstorage** コンポーネントが Operator によって選択された場合)
- **Clair database:** セキュリティーメタデータには最低 5 GB が必要

Red Hat Quay のステートレスコンポーネントはスケールアウトできますが、これによりステートフルバックエンドサービスの負荷が高くなります。

7.1. RED HAT QUAY のサンプルサイジング

次の表は、概念実証、中規模、およびハイエンドのデプロイメントのおおよそのサイズを示しています。デプロイメントが同じメトリックで適切に実行されるかどうかは、以下に示されていない多くの要因によって決まります。

メトリクス	概念実証	中規模	ハイエンド (Quay.io)
デフォルトの Quay コンテナーの数	1	4	15
スケールアウト時の Quay コンテナーの最大数	該当なし	8	30
デフォルトの Clair コンテナーの数	1	3	10
スケールアウト時の Clair コンテナーの最大数	該当なし	6	15
ミラーリング Pod の数 (100 個のリポジトリーをミラーリングする場合)	1	5-10	該当なし

メトリクス	概念実証	中規模	ハイエンド (Quay.io)
データベースのサイジング	2 -4 コア 6-8 GB RAM 10-20 GB ディスク	4-8 コア 6-32 GB RAM 100 GB - 1 TB ディスク	32 コア 244GB 1TB 以上のディスク
オブジェクトストレージのバックエンドのサイジング	10-100 GB	1-20 TB	50 TB 以上 PB まで
Redis のキャッシュサイジング		2 コア 2-4 GB RAM	4 コア 28 GB RAM
基礎となるノードサイジング (物理または仮想)	4 コア 8 GB RAM	4-6 コア 12-16 GB RAM	Quay: 13 コア 56 GB RAM Clair: 2 コア 4 GB RAM

サイジングおよびミラーリングに関連する推奨事項の詳細は、リポジトリーのミラーリング の項を参照してください。

Redis キャッシュのサイジングは、Quay ビルダーを使用する場合にのみ関連します。それ以外の場合は、重要ではありません。

7.2. RED HAT QUAY のサブスクリプション情報

Red Hat Quay は Standard または Premium のサポートを受けることができ、サブスクリプションはデプロイメントに基づいて行われます。

注記

デプロイメントとは、共有データバックエンドを使用する単一の Red Hat Quay レジストリーのインストールを意味します。

Red Hat Quay サブスクリプションでは、次のオプションが利用可能です。

- デプロイできる Pod (Quay、Clair、Builder など) の数に制限はありません。
- Red Hat Quay の Pod は、複数のデータセンターやアベイラビリティーゾーンで実行できます。
- ストレージとデータベースのバックエンドは、複数のデータセンターやアベイラビリティーゾーンにまたがってデプロイできますが、単一の共有ストレージバックエンドと単一の共有データベースバックエンドとしてのみデプロイできます。
- Red Hat Quay は、無制限の数のクラスターまたはスタンドアロンのサーバーのコンテンツを管理できます。

- クライアントは、物理的な場所に関係なく、Red Hat Quay デプロイメントにアクセスできます。
- Red Hat Quay を OpenShift Container Platform インフラストラクチャーノードにデプロイして、サブスクリプション要件を最小限に抑えることができます。
- Container Security Operator (CSO) と Quay Bridge Operator (QBO) は、追加コストなしで OpenShift Container Platform クラスター上で実行できます。

注記

Red Hat Quay の Geo レプリケーションは、各ストレージレプリケーションにサブスクリプションが必要です。しかし、データベースは共有されています。

Red Hat Quay サブスクリプションの購入の詳細は、 Red Hat Quay を参照してください。

7.3. 内部レジストリーあり/なしで RED HAT QUAY を使用する場合

Red Hat Quay は、内部レジストリーを持つ複数の OpenShift Container Platform クラスターの前で外部レジストリーとして使用できます。

Red Hat Quay は、ビルドやデプロイメントのロールアウトを自動化する際に、内部レジストリーの代わりに使用することもできます。**Secret** と **ImageStream** に必要な調整は、OpenShift Container Platform の OperatorHub から起動できる Quay Bridge Operator によって自動化されます。

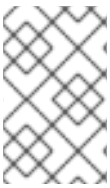
第8章 クォータ管理アーキテクチャー

クォータ管理機能を有効にすると、個々の BLOB サイズがリポジトリーレベルと名前空間レベルで合計されます。たとえば、同じリポジトリー内の 2 つのタグが同じ BLOB を参照している場合、その BLOB のサイズはリポジトリーの合計に対して 1 回だけカウントされます。さらに、マニフェストリストの合計もリポジトリーの合計にカウントされます。

重要

マニフェスト一覧の合計はリポジトリーの合計にカウントされるため、Red Hat Quay の以前のバージョンからアップグレードするときに消費される合計クォータは、Red Hat Quay 3.9 では大幅に異なる可能性があります。場合によっては、新しい合計がリポジトリーで以前に設定された制限を超える可能性があります。Red Hat Quay 管理者は、これらの変更を考慮して、リポジトリーに割り当てられたクォータを調整する必要がある場合があります。

クォータ管理機能は、バックフィルワーカーを使用して既存のリポジトリーと名前空間のサイズを計算し、その後プッシュまたはガベージコレクションされたすべてのイメージの合計を加算または減算することによって機能します。さらに、マニフェストがガベージコレクションされるときに、合計からの減算が行われます。



注記

マニフェストがガベージコレクションされるときに合計から減算が発生するため、ガベージコレクションが可能になるまでサイズの計算に遅れが生じます。ガベージコレクションの詳細は、Red Hat Quay ガベージコレクション を参照してください。

以下のデータベーステーブルには、組織内の Red Hat Quay リポジトリーのクォータリポジトリーサイズ、クォータ名前空間サイズ、およびクォータレジストリーサイズ (バイト単位) が保持されます。

- **QuotaRepositorySize**
- **QuotaNameSpaceSize**
- **QuotaRegistrySize**

組織のサイズは、重複しないようにバックフィル作業者によって計算されます。イメージプッシュが初期化されると、ユーザーの組織ストレージが検証され、設定されたクォータ制限を超えているかどうかがチェックされます。イメージプッシュが定義されたクォータ制限を超えると、ソフトチェックまたはハードチェックが発生します。

- ソフトチェックの場合は、ユーザーに通知されます。
- ハードチェックの場合は、プッシュが停止します。

ストレージ消費量が設定済みのクォータ制限内にある場合は、プッシュを続行できます。

イメージマニフェストの削除も同様のフローに従い、関連するイメージタグとマニフェストの間のリンクが削除されます。さらに、イメージマニフェストが削除された後、リポジトリーサイズが再計算され、**QuotaRepositorySize**、**QuotaNameSpaceSize**、および **QuotaRegistrySize** テーブルで更新されます。

第9章 名前空間の自動プルーニングのアーキテクチャー

名前空間の自動プルーニング機能のために、データベーススキーマ内に 2 つの異なるデータベーステーブルが作成されました。1つは **namespaceautoprunepolicy** 用、もう1つは **autoprunetaskstatus** 用です。自動プルーンワーカーが、設定されたポリシーを実行します。

名前空間の自動プルーニングポリシーのデータベーステーブル

namespaceautoprunepolicy データベーステーブルには、単一の名前空間のポリシー設定が保持されます。エントリーは名前空間ごとに1つだけ存在しますが、**namespace_id** ごとに複数の行がサポートされています。**policy** フィールドには、**{method: "creation_date", olderThan: "2w"}** や **{method: "number_of_tags", numTags: 100}** などのポリシーの詳細が保持されます。

表9.1 namespaceautoprunepolicy データベーステーブル

フィールド	型	属性	説明
uuid	character varying (225)	一意、インデックス付き	このポリシーの一意の識別子
namespace_id	Integer	外部キー	ポリシーが属する名前空間
policy	text	JSON	ポリシー設定

自動プルーンタスクステータスのデータベーステーブル

autoprunetaskstatus テーブルには、自動プルーンワーカーによって実行されるタスクが登録されます。タスクは単一の名前空間のコンテキスト内で実行されます。タスクは名前空間ごとに1つだけ存在します。

表9.2 autoprunetaskstatus データベーステーブル

フィールド	型	属性	説明
namespace_id	Integer	外部キー	このタスクが属する名前空間
last_ran_ms	Big Integer (bigint)	Null 許容型、インデックス付き	ワーカーがこの名前空間のポリシーを最後に実行した時刻
status	text	Null 許容型	最後に実行されたタスクの詳細

9.1. 自動プルーンワーカー

次のセクションでは、自動プルーンワーカーに関する情報について詳しく説明します。

9.1.1. 自動プルーンタスクの作成

新しいポリシーが **namespaceautoprunepolicy** データベーステーブルに作成されると、**autoprunetask** テーブルにも行が作成されます。これは同じトランザクションで実行されます。自

動プルーンワーカーは、**autoprunetask** テーブル内のエントリーを使用して、ポリシーを実行する必要がある名前空間を特定します。

9.1.2. 自動プルーンワーカーの実行

自動プルーニングワーカーは、設定されたポリシーを実行する非同期ジョブです。そのワークフローは、**autoprunetask** テーブルの値に基づいています。タスクが開始すると、次のことが起こります。

- 自動プルーンワーカーが、設定された間隔 (デフォルトは 30 秒) で起動します。
- 自動プルーンワーカーが、**autoprunetask** から、**last_ran_ms** および **FOR UPDATE SKIP LOCKED** が最小または null の行を選択します。
    - **last_ran_ms** が null である場合、そのタスクは一度も実行されていません。
    - 最も長い時間実行されていないタスク、または一度も実行されていないタスクが優先されます。
- 自動プルーンワーカーが、**namespaceautoprunepolicy** テーブルからポリシー設定を取得します。
    - ポリシー設定が存在しない場合、この名前空間のエントリーが **autoprunetask** から削除され、手順が直ちに停止します。
- 自動プルーンワーカーが、組織内の全リポジトリーのページ付けされたループを開始します。
    - 自動プルーンワーカーは、**policy.method** に基づいて使用するプルーニング方法を決定します。
- 自動プルーンワーカーが、以前に取得したポリシー設定を使用してプルーニング方法を実行します。
    - タグの数に基づくプルーニングの場合: 自動プルーナーワーカーが、現在アクティブなタグの数を作成日順に取得し、設定された数だけ古いタグを削除します。
    - 日付に基づくプルーニングの場合: 自動プルーナーワーカーが、指定された期間よりも古いアクティブなタグを取得し、返されたタグがすべて削除されます。
- 自動プルーンワーカーが、削除されたタグの監査ログを追加します。
- **autoprunetask** の行が選択された後に、**last_ran_ms** が更新されます。
- 自動プルーンワーカーが終了します。