Red Hat Quay 3.7

Red Hat Quay のアップグレード

Red Hat Quay のアップグレード

Last Updated: 2023-10-27

Red Hat Quay 3.7 Red Hat Quay のアップグレード

Red Hat Quay のアップグレード

法律上の通知

概要

Red Hat Quay のアップグレード

目次

第1章 アップグレードの概要

Red Hat Quay のアップグレード手順は、使用しているインストールの種類によって異なります。

Red Hat Quay Operator は、Red Hat Quay クラスターをデプロイし、管理する簡単な方法を提供します。これは、Red Hat Quay の OpenShift へのデプロイで推奨の手順です。

Quay Operator を使用した Quay のアップグレードの項で説明されているように、Red Hat Quay Operator のアップグレードは、Operator Lifecycle Manager (OLM) を使用する必要があります。

Red Hat Quay および Clair の概念実証または高可用性のインストールをアップグレードする手順は、スタンドアロンでのアップグレードに記載されています。

重要

Red Hat Quay は、以前の z-stream バージョン (3.7.2 → 3.7.1 など) へのロールバックまたはダウングレードのみをサポートします。以前の y-stream バージョン (3.7.0 → 3.6.0) へのロールバックはサポートされていません。これは、Red Hat Quay の更新に、Red Hat Quay の新しいバージョンにアップグレードするときに適用されるデータベーススキーマのアップグレードが含まれている可能性があるためです。データベーススキーマのアップグレードは、下位互換性があるとは見なされません。

第2章 QUAY OPERATOR のアップグレードの概要

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、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、デプロイする Quay のバージョンを設定するフィールドがありません。Operator は単一バージョンの全コンポーネントをデプロイする方法のみを認識します。このスキームは、すべてのコンポーネントが適切に機能するように、また Kubernetes 上の複数バージョンの Quay のライフサイクルを管理する方法に関する Operator の複雑さを軽減するために選択されています。

2.1. OPERATOR LIFECYCLE MANAGER

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** を作成すると、OLM は新規バージョンが利用可能になると常に Quay Operator を自動的にアップグレードします。

警告

Quay Operator が Operator Lifecycle Manager からインストールされると、自動または手動アップグレードをサポートするように設定できます。このオプションは、インストール時に Quay Operator の **Operator Hub** ページに表示されます。また、これは **approvalStrategy** フィールドで Quay Operator **Subscription** オブジェクトで確認できます。**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 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。

2.2. QUAY OPERATOR のアップグレード

インストールされた Operator を OpenShift にアップグレードする一般的な方法は、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。以前のマイナーバージョンのみをアップグレードする通常のアップグレードに対するこの例外により、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。次のアップグレードパスがサポートされています。

1. 3.3.z → 3.6.z
2. 3.4.z → 3.6.z
3. 3.4.z → 3.7.z
4. 3.5.z → 3.7.z

3.7 へのアップグレードを希望する Quay のスタンドアロンデプロイメントのユーザーは、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。

2.2.1. Quay のアップグレード

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、3.4 → 3.5) に更新するには、Quay Operator の更新チャネルを変更する必要があります。

3.4.2 → 3.4.3 などの **z** ストリームのアップグレードの場合、更新は、ユーザーが最初にインストール時に選択した major-minor 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、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。これにより、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 Quay の自動更新が行われます。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。

2.2.2. 3.3.z または 3.4.z から 3.6 に直接アップグレードする際の注意事項

2.2.2.1. エッジルーティングを有効にしてアップグレード

- 以前は、エッジルーティングを有効にして 3.3.z バージョンの Red Hat Quay を実行している場合、ユーザーは 3.4.z バージョンの Red Hat Quay にアップグレードできませんでした。これは、Red Hat Quay 3.6 のリリースで解決されました。
- 3.3.z から 3.6 にアップグレードするときに、Red Hat Quay 3.3.z デプロイメントで **tls.termination** が **none** に設定されている場合は、TLS エッジ終端を使用して HTTPS に変更され、デフォルトのクラスターワイルドカード証明書が使用されます。以下に例を示します。

```
apiVersion: redhatcop.redhat.io/v1alpha1
kind: QuayEcosystem
metadata:
  name: quay33
spec:
  quay:
    imagePullSecretName: redhat-pull-secret
    enableRepoMirroring: true
    image: quay.io/quay/quay:v3.3.4-2
    ...
    externalAccess:
      hostname: quayv33.apps.devcluster.openshift.com
      tls:
        termination: none
    database:
...
```

2.2.2.2. サブジェクト別名のないカスタム TLS 証明書/キーペアを使用したアップグレード

Red Hat Quay 3.3.4 から Red Hat Quay 3.6 に直接アップグレードするときに、サブジェクト代替名 (SAN) なしで独自の TLS 証明書/キーペアを使用しているお客様には問題があります。Red Hat Quay 3.6 へのアップグレード中、デプロイメントはブロックされ、Quay TLS 証明書に SAN が必要であることを示す Quay Operator Pod ログからのエラーメッセージが表示されます。

可能であれば、SAN 内の正しいホスト名を使用して TLS 証明書を再生成する必要があります。考えられる回避策には、アップグレード後に **quay-app**、**quay-upgrade**、**quay-config-editor** Pod で環境変数を定義して、CommonName のマッチングを有効にすることが含まれます。

```
GODEBUG=x509ignoreCN=0
```

GODEBUG=x509ignoreCN=0 フラグは、SAN が存在しない場合に、X.509 証明書の CommonName フィールドをホスト名として扱うという従来の動作を有効にします。ただし、この回避策は再デプロイメント後も持続しないため、推奨しません。

2.2.2.3. Quay Operator を使用して 3.3.z または 3.4.z から 3.6 にアップグレードする場合の Clair v4 の設定

OpenShift 上の新しい Red Hat Quay デプロイメントに Clair v4 をセットアップするには、Quay Operator を使用することが強く推奨されます。デフォルトでは、Quay Operator は、Red Hat Quay のデプロイとともに Clair のデプロイメントをインストールまたはアップグレードし、Clair のセキュリティースキャンを自動的に設定します。

OpenShift で Clair v4 をセットアップする手順は、Red Hat Quay OpenShift デプロイメントでの Clair のセットアップ を参照してください。

2.2.3. 3.3.z から 3.6 にアップグレードする際の Swift 設定

Red Hat Quay 3.3.z から 3.6.z にアップグレードすると、ユーザーが **Switch auth v3 requires tenant_id (string) in os_options** エラーを受け取る場合があります。回避策として、**DISTRIBUTED_STORAGE_CONFIG** を手動で更新して、**os_options** パラメーターおよび **tenant_id** パラメーターを追加できます。

```
DISTRIBUTED_STORAGE_CONFIG:
  brscale:
  - SwiftStorage
  - auth_url: http://****/v3
    auth_version: "3"
    os_options:
      tenant_id: ****
      project_name: ocp-base
      user_domain_name: Default
    storage_path: /datastorage/registry
    swift_container: ocp-svc-quay-ha
    swift_password: *****
    swift_user: *****
```

2.2.4. Operator の更新チャネルの変更

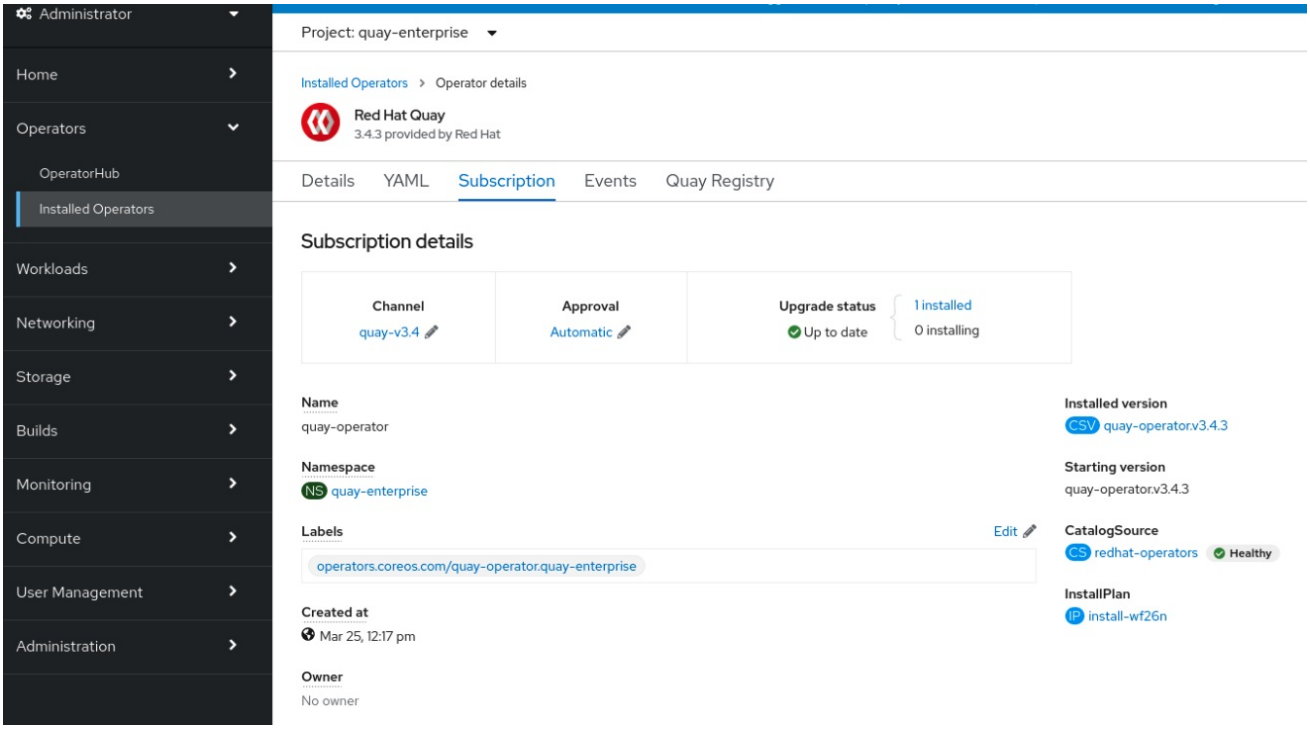
インストールされた Operator のサブスクリプションは、Operator の更新を追跡して受け取るために使用される更新チャネルを指定します。Quay Operator をアップグレードして新規チャネルからの更新の追跡および受信を開始するには、インストールされた Quay Operator の **Subscription** タブで更新チャ

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。

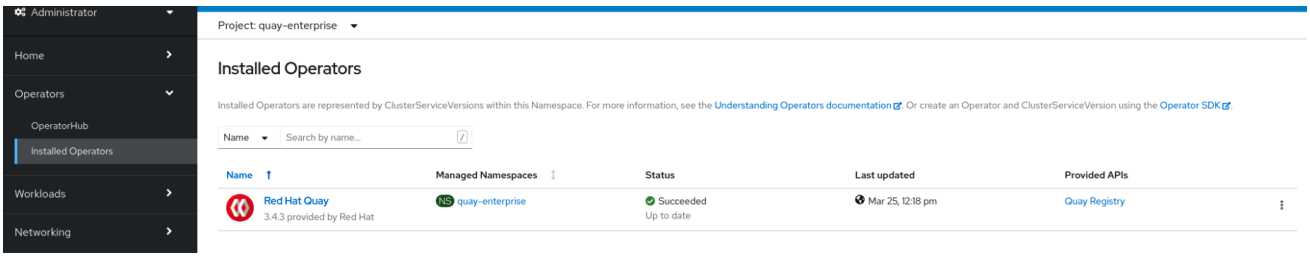
2.2.5. 保留中の Operator アップグレードの手動による承認

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。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 Installed Operators のリストに表示されます。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リスト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 をリスト表示したページに戻り、アップグレードの進捗をモニターします。

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。



2.3. QUAYREGISTRY のアップグレード

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** を探します。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。

2.4. QUAY 3.7 の機能の有効化

2.4.1. クォータ管理設定

クォータ管理は **FEATURE_QUOTA_MANAGEMENT** プロパティーでサポートされるようになり、デフォルトでオフになっています。クォータ管理を有効にするには、**config.yaml** の機能フラグを **true** に設定します。

```
FEATURE_QUOTA_MANAGEMENT: true
```

2.4.2. Red Hat Quay を使用してリモート組織設定をプロキシーする

Red Hat Quay を使用したリモート組織のプロキシーが、**FEATURE_PROXY_CACHE** プロパティーでサポートされるようになりました。プロキシーキャッシュを有効にするには、**confg.yaml** の機能フラグを **true** に設定します。

```
FEATURE_PROXY_CACHE: true
```

2.4.3. Red Hat Quay ビルドの機能強化

ビルドは仮想化プラットフォームで実行できます。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。仮想ビルドを有効にするには、**config.yaml** の機能フラグを **true** に設定します。

```
FEATURE_BUILD_SUPPORT: true
```

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

Geo レプリケーションを使用した Red Hat Quay のデプロイメントが、Operator デプロイメントでサポートされるようになりました。ジオレプリケーションを有効にするには、**config.yaml** の機能フラグを **true** に設定します。

```
FEATURE_STORAGE_REPLICATION: true
```

2.5. QUAY 3.6 の機能の有効化

2.5.1. コンソールでのモニタリングおよびアラート

OpenShift コンソールでの Quay 3.6 のモニタリングのサポートを使用するには、Operator がすべての namespace でインストールされている必要があります。Operator を特定の namespace にインストールしている場合は、Operator 自体を削除し、アップグレードが実行されたら、これをすべての namespace に対して再インストールします。

2.5.2. OCI および Helm サポート

Helm アーティファクトおよび OCI アーティファクトのサポートが、Red Hat Quay 3.6 でデフォルトで有効にされるようになりました。機能を明示的に有効にする必要がある場合 (機能がデフォルトで有効にされていないバージョンからアップグレードする場合など) は、以下のプロパティーを使用して OCI アーティファクトの使用を有効にするために、Quay デプロイメントを再設定する必要があります。

```
FEATURE_GENERAL_OCI_SUPPORT: true
```

2.6. 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** を新規の **QuayRegistry** に移行するには、以下の手順を実行します。

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. 新規 **QuayRegistry** の **status.registryEndpoint** が設定された後に、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. すべてが正しく動作したと確信できたら、**QuayEcosystem** を削除できます。Kubernetes のガベージコレクションがすべての古いリソースをクリーンアップします。

2.6.1. 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。

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 Postgres データベースを管理していた場合は、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移行されます。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。

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

Quay Operator は、**QuayEcosystem** コンポーネントの移行に失敗したり、サポートされていない場合は、ログや **status.conditions** にエラーを報告します。Kubernetes リソースを採用する必要がなく、必要な値がすべて Quay の **config.yaml** に提供されているため、すべてのアンマネージドコンポーネントは正常に移行されるはずです。

データベース

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。

第3章 スタンドアロンアップグレード

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。以前のマイナーバージョンのみをアップグレードする通常のアップグレードに対するこの例外により、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。次のアップグレードパスがサポートされています。

1. 3.3.z → 3.6.z
2. 3.4.z → 3.6.z
3. 3.4.z → 3.7.z
4. 3.5.z → 3.7.z

Quay Operator を介してアップグレードするユーザーについては、Quay Operator をアップグレードして Quay をアップグレード を参照してください。

このドキュメントでは、各アップグレードに必要な手順を説明します。現在のバージョンを決定し、現在のバージョンから順に、目標とするバージョンへとステップを踏んで進めていきます。

- 3.6.z から 3.7.z へのアップグレード
- 3.5.z から 3.7.z へのアップグレード
- 3.4.z から 3.7.z へのアップグレード
- 3.3.z から 3.7.z へのアップグレード
- 3.5.z から 3.6.z へのアップグレード
- 3.4.z から 3.6.z へのアップグレード
- 3.3.z から 3.6.z へのアップグレード
- 3.4.z から 3.5.z へのアップグレード
- 3.3.4 から 3.4.z へのアップグレード
- 3.2.2 から 3.3.4 へのアップグレード

- 3.1.3 から 3.2.2 へのアップグレード
- 3.0.5 から 3.1.3 へのアップグレード
- 2.9.5 から 3.0.5 へのアップグレード

個々のリリースの機能に関する情報は、Red Hat Quay リリースノート を参照してください。

手動アップグレードの一般的な手順は、以下のとおりです。

1. Quay および Clair コンテナーを停止する
2. データベースとイメージストレージをバックアップする (任意ではあるが推奨)
3. 新バージョンのイメージを使用して Clair を起動する
4. Clair が接続を受け入れる準備ができるまで待ってから、新しいバージョンの Quay を起動する

3.1. イメージへのアクセス

Quay 3.4.0 以降のイメージは registry.redhat.io および registry.access.redhat.com から入手でき、Red Hat コンテナーレジストリーの認証 で説明されているように認証が設定されています。

Quay 3.3.4 以前のイメージは quay.io から入手可能で、認証は、Accessing Red Hat Quay without a CoreOS login で説明されている通りに設定されています。

3.2. 3.6.Z から 3.7.Z へのアップグレード

3.2.1. ターゲットイメージ

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.8.8
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.8.8
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10
- **Redis:** registry.redhat.io/rhel8/redis-6

3.3. 3.5.Z から 3.7.Z へのアップグレード

3.3.1. ターゲットイメージ

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.8.8
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.8.8
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10
- **Redis:** registry.redhat.io/rhel8/redis-6

3.4. 3.4.Z から 3.7.Z へのアップグレード

3.4.1. ターゲットイメージ

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.8.8
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.8.8
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10
- **Redis:** registry.redhat.io/rhel8/redis-6

3.5. 3.3.Z から 3.7.Z へのアップグレード

Red Hat Quay 3.3 から 3.7 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、最初に 3.3 から 3.6 にアップグレードしてから、3.7 にアップグレードする必要があります。詳細については、3.3.z から 3.6.z へのアップグレード を参照してください。

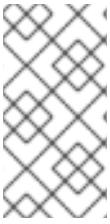
3.6. 3.5.Z から 3.6.Z へのアップグレード

3.6.1. ターゲットイメージ

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.6.0
- **Clair:** registry.redhat.io/quay/clair-rhel8:v.3.60
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10
- **Redis:** registry.redhat.io/rhel8/redis-6

3.7. 3.4.Z から 3.6.Z へのアップグレード

+



注記

Red Hat Quay 3.6 は、3.4.z 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。以前のマイナーバージョンのみをアップグレードする通常のアップグレードに対するこの例外により、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。

3.4.z から Red Hat Quay 3.6 にアップグレードするには、以前のバージョンの Red Hat Quay へのダウングレードをサポートしないデータベースの移行が必要です。この移行を行う前に、データベースをバックアップしてください。

また、ユーザーは、3.4.z からアップグレードするときに、古い Clair v2 を置き換えるために完全に新しい Clairv4 インスタンスを設定する必要があります。Clair v4 の設定手順については、OpenShift 以外の Red Hat Quay デプロイメントでの Clair のセットアップ を参照してください。

3.7.1. ターゲットイメージ

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.6.0
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.6.0
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10

- **Redis:** registry.redhat.io/rhel8/redis-6

3.8. 3.3.Z から 3.6.Z へのアップグレード

+

注記

Red Hat Quay 3.6 は、3.3.z 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。以前のマイナーバージョンのみをアップグレードする通常のアップグレードに対するこの例外により、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。

3.3.z から Red Hat Quay 3.6.z にアップグレードするには、以前のバージョンの Red Hat Quay へのダウングレードをサポートしないデータベースの移行が必要です。この移行を行う前に、データベースをバックアップしてください。

また、ユーザーは、3.3.z からアップグレードするときに、古い Clair v2 を置き換えるために完全に新しい Clairv4 インスタンスを設定する必要があります。Clair v4 の設定手順については、OpenShift 以外の Red Hat Quay デプロイメントでの Clair のセットアップ を参照してください。

3.8.1. ターゲットイメージ

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.6.0
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.6.0
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10
- **Redis:** registry.redhat.io/rhel8/redis-6

3.8.2. 3.3.z から 3.6 にアップグレードする際の Swift 設定

Red Hat Quay 3.3.z から 3.6.z にアップグレードすると、ユーザーが **Switch auth v3 requires tenant_id (string) in os_options** エラーを受け取る場合があります。回避策として、**DISTRIBUTED_STORAGE_CONFIG** を手動で更新して、**os_options** パラメーターおよび **tenant_id** パラメーターを追加できます。

```
DISTRIBUTED_STORAGE_CONFIG:
  brscale:
  - SwiftStorage
  - auth_url: http://****/v3
    auth_version: "3"
    os_options:
      tenant_id: ****
      project_name: ocp-base
      user_domain_name: Default
    storage_path: /datastorage/registry
    swift_container: ocp-svc-quay-ha
    swift_password: *****
    swift_user: *****
```

3.9. 3.4.Z から 3.5.7 へのアップグレード

3.9.1. ターゲットイメージ

- **Quay:** registry.redhat.io/quay/quay-rhel8:v3.5.7
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.5.7
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10:1
- **Redis:** registry.redhat.io/rhel8/redis-5

3.10. 3.3.Z から 3.4.6 へのアップグレード

Quay 3.4 にアップグレードするには、データベースの移行が必要ですが、データベースを移行すると、以前のバージョンの Quay にダウングレードできません。この移行を行う前に、データベースをバックアップしてください。

3.10.1. ターゲットイメージ

- **Quay:**registry.redhat.io/quay/quay-rhel8:v3.4.6
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.4.6
- **PostgreSQL:** registry.redhat.io/rhel8/postgresql-10
- **Redis:** registry.redhat.io/rhel8/redis-5

3.11. 3.2.Z から 3.3.4 へのアップグレード

3.11.1. ターゲットイメージ

- **Quay:** quay.io/redhat/quay:v3.3.4
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.3.4
- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7

3.12. 3.1.Z から 3.2.2 へのアップグレード

クラスターで Red Hat Quay 3.1.z が稼働しており、クラスターを 3.2.2 にアップグレードするには、クラスター全体をダウンさせ、設定を少し変更してから 3.2.2 バージョンで、起動し直す必要があります。

警告

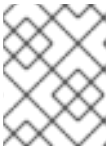
この手順で DATABASE_SECRET_KEY の値を設定したら、絶対に変更しないでください。変更すると、既存のロボットアカウントや API トークンなどは使用できなくなります。Quay で使用するためには、新しいロボットアカウントと API トークンを作成する必要があります。

1. Red Hat Quay クラスターの全ホストのサービスを停止します。
2. データベースの秘密鍵として使用するランダムなデータを生成します。以下に例を示します。

```
$ openssl rand -hex 48
2d023adb9c477305348490aa0fd9c
```

3. **config.yaml** ファイルに新しい DATABASE_SECRET_KEY フィールドを追加します。以下に例を示します。

```
DATABASE_SECRET_KEY: "2d023adb9c477305348490aa0fd9c"
```



注記

OpenShift のインストールでは、**config.yaml** ファイルはシークレットとして保存されます。

4. **Quay** コンテナーを 1 つ起動し、3.2.2 への移行を完了します。
5. 移行が完了したら、すべてのノードで同じ **config.yaml** が利用可能であることを確認し、それらのノードで新しい quay 3.2.2 サービスを起動します。
6. quay-builder と Clair の 3.0.z バージョンを起動して、クラスターに戻すコンテナーのインスタンスを置き換えます。

3.12.1. ターゲットイメージ

- **Quay:** quay.io/redhat/quay:v3.2.2
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.8.8
- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7

3.13. 3.0.Z から 3.1.3 へのアップグレード

3.13.1. ターゲットイメージ

- **Quay:** quay.io/redhat/quay:v3.1.3
- **Clair:** registry.redhat.io/quay/clair-rhel8:v3.8.8

- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7

3.14. 2.9.5 から 3.0.5 へのアップグレード

2.9.5 から 3.0.5 へのアップグレードでは、Red Hat Quay がダウンしている状態で全アップグレードを行う (同時アップグレード) か、Red Hat Quay を数分間だけダウンさせて、アップグレードの大部分は Red Hat Quay が稼働している状態で行う (バックグラウンドアップグレード) かを選択できます。

処理する必要のあるタグの数によっては、バックグラウンドアップグレードの実行に時間がかかる場合があります。ただし、合計ダウンタイムは短くなります。バックグラウンドアップグレードの欠点は、アップグレードが完了するまで最新の機能にアクセスできないことです。クラスターは、アップグレードが完了するまで、Quay v3 コンテナーから v2 互換モードで実行されます。

3.14.1. アップグレードの概要

Red Hat Quay 2.y.z クラスターで作業を開始する場合は、以下の手順に従います。最新の Red Hat Quay 3.x バージョンにアップグレードする前に、ここ で説明されているように、まずそのクラスターを 3.0.5 に移行する必要があります。クラスターで 3.0.5 が実行されたら、各マイナーバージョンに順番にアップグレードすることで、最新の 3.x バージョンにアップグレードできます。以下に例を示します。

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

Red Hat Quay 2.y.z から 3.0 へのアップグレードを開始する前に、次の点に注意してください。

- **同期アップグレード**: 同期アップグレードの場合は、小規模なインストールであれば、ダウンタイムの想定時間は合計 1 時間未満となっています。小規模なインストールの場合は、コンテナーイメージのタグが数千以下であると想定してください。このサイズのインストールでは、計画ダウンタイムが 2 時間程度に抑えられるはずです。Red Hat Quay サービス全体がこの期間、停止しているので、何百万ものタグが含まれるレジストリーで同期アップグレードを試行する場合はダウンタイムが数日間に及ぶ場合があります。
- **バックグラウンドアップグレード**: バックグラウンドアップグレード (互換性モードのアップグレードとも呼ばれる) の場合は、シャットダウンが短時間で行われた後に Red Hat Quay クラスターのアップグレードがバックグラウンドで実行されます。大規模な Red Hat Quay レジストリーの場合、これには数週間の時間がかかる可能性がありますが、アップグレード中には、クラスターは引き続き v2 モードで動作します。参考までに、ある Red Hat Quay v3 のアップグレードでは、6 台のマシンで約 3000 万個のタグを処理するのに 4 日かかりました。
- **完了時に完全な機能**: Docker バージョン 2、スキーマ 2 の変更に伴う機能 (異なるアーキテクチャーのコンテナーのサポートなど) にアクセスするには、移行がすべて完了している必要があります。その他の v3 機能は、切り替え後すぐに利用できます。
- **アップグレードの完了**: アップグレードが完了したら、新機能が利用可能になるように Red Hat Quay **config.yaml** ファイルで **V3_UPGRADE_MODE: complete** を設定する必要があります。すべての新しい Red Hat Quay v3 のインストールには、自動的にこの設定がされています。

3.14.2. 前提条件

最善の結果が得られるように、以下の前提条件を満たすことを推奨します。

- アップグレードを開始する前に Red Hat Quay データベースをバックアップしておきます (定期的なバックアップを実行するのが一般的なベストプラクティスです)。バックアップは、アップグレードを行うために Red Hat Quay クラスターを停止した直後が適切なタイミングです。
- ストレージをバックアップします (こちらも一般的なベストプラクティス)。
- V3 のアップグレードを開始する前に、現在の Red Hat Quay 2.y.z 設定を最新の 2.9.z バージョン (現時点では 2.9.5) にアップグレードします。これを実行するには、以下を行います。
    - Red Hat Quay クラスターがまだ実行中の間に、ノード 1 つを取り、そのシステムの **Quay** コンテナーを最新の 2.9.z バージョンを実行している **Quay** コンテナーに変更します。
    - すべてのデータベース移行の実行を待機し、データベースを最新の 2.9.z バージョンにします。これには数分から 1 時間かかります。
    - 完了したら、すべての既存ノードの **Quay** コンテナーを、同じ最新の 2.9.z バージョンに置き換えます。新規バージョンの Red Hat Quay クラスター全体で、v3 アップグレードに進むことができます。

3.14.3. アップグレードタイプの選択

同期アップグレード (ダウンタイムでアップグレードを完了) か、バックグラウンドアップグレード (Red Hat Quay の実行中にアップグレードを完了) のいずれかを選択します。これらのメジャーリリースのアップグレードでは、Red Hat Quay クラスターを少なくとも短期間停止する必要があります。

選択したアップグレードのタイプを問わず、ビルダーおよび clair イメージを使用している場合は、Red Hat Quay クラスターが停止している間に、ビルダーおよび Clair を新規イメージにアップグレードする必要があります。

- **Builder**: quay.io/redhat/quay-builder:v3.0.5
- **Clair**: quay.io/redhat/clair-jwt:v3.0.5

これらのイメージはいずれも registry.redhat.io/quay リポジトリーから入手できます。

3.14.4. 同期アップグレードの実行

同期アップグレード (アップグレード中にクラスター全体が停止する) を実行するには、以下を実行します。

1. Quay-builder および Clair コンテナーなど Red Hat Quay クラスター全体で停止します。
2. 以下の設定を全ノードの **config.yaml** ファイルに追加します。
   V3_UPGRADE_MODE: complete
3. 単一ノードで v3 コンテナーをプルおよび起動して、アップグレードが完了するのを待ちます (数分で完了するはずです)。以下のコンテナーのバージョンより新しいものを使用します。
    - **Quay**: quay.io/redhat/quay:v3.0.5
      Quay コンテナーは、Red Hat Quay 2 の場合のように 80 および 443 ではなく、Red Hat Quay 3 のポート 8080 および 8443 で起動することに注意してください。したがって、以下の例のように 8080 および 8443 をそれぞれ 80 および 443 に再マッピングすることを推奨します。

```
# docker run --restart=always -p 80:8080 -p 443:8443 \
  --sysctl net.core.somaxconn=4096 \
  --privileged=true \
  -v /mnt/quay/config:/conf/stack:Z \
  -v /mnt/quay/storage:/datastorage:Z \
  -d quay.io/redhat/quay:v3.0.5
```

4. アップグレードが完了したら、他のすべてのノードで Red Hat Quay 3 コンテナーを起動します。

5. quay-builder と Clair の 3.0.z バージョンを起動して、クラスターに戻すコンテナーのインスタンスを置き換えます。

6. Docker バージョン 2、スキーマ 2 と互換性のあるコンテナーのプッシュおよびプルなど、Red Hat Quay が機能していることを確認します。これには、Windows コンテナーイメージおよび異なるコンピューターアーキテクチャーのイメージ (arm、ppc など) が含まれます。

3.14.5. バックグラウンドアップグレードの実行

バックグラウンドアップグレードは、2 回ほどクラスターを短時間ダウンさせるだけで実行できます。最初のダウンタイム後にクラスターを再起動すると、データベースをバックフィルするため、quay v3 コンテナーは v2 互換性モードで実行します。このバックグラウンドプロセスが完了するまでに時間または数日かかる場合があります。数時間を超えるダウンタイムが問題となるような大規模なインストールを行う場合は、バックグラオウンドアップグレードが推奨されます。

このタイプのアップグレードでは、Red Hat Quay を互換性モードにします。互換性モードでは、**Quay** 3 コンテナーが実行しますが、アップグレードが完了するまで以前のデータモデルで実行します。手順は以下のとおりです。

1. Red Hat Quay 3 コンテナーをすべてのノードにプルします。以下のコンテナーのバージョンより新しいものを使用します。
   quay.io/redhat/quay:v3.0.5

2. Quay-builder および Clair コンテナーなど Red Hat Quay クラスター全体で停止します。

3. 各ノードで **config.yaml** ファイルを編集し、以下のようにアップグレードモードを background に設定します。
   V3_UPGRADE_MODE: background

4. Red Hat Quay 3 コンテナーを単一ノードで起動し、移行が完了するまで待機します (最大で数分かかります)。以下はコマンドの例です。
   Quay コンテナーは、Red Hat Quay 2 の場合のように 80 および 443 ではなく、Red Hat Quay 3 のポート 8080 および 8443 で起動することに注意してください。したがって、以下の例のように 8080 および 8443 をそれぞれ 80 および 443 に再マッピングすることを推奨します。

```
# docker run --restart=always -p 80:8080 -p 443:8443 \
  --sysctl net.core.somaxconn=4096 \
  --privileged=true \
  -v /mnt/quay/config:/conf/stack:Z \
  -v /mnt/quay/storage:/datastorage:Z \
  -d quay.io/redhat/quay:v3.0.5
```

5. その他のすべてのノードで Red Hat Quay 3 コンテナーを起動します。

6. 次の手順に進むのに十分なレポーティングがされるまで (ステータスが 99% に到達するまで)、**/upgradeprogress** API エンドポイントを監視します。たとえば **https://myquay.example.com/upgradeprogress** を表示するか、他のツールを使用して API をクエリーします。

7. バックグラウンドプロセスが十分に終了したら、別のメンテナンス期間をスケジュールする必要があります。

8. 定期メンテナンス時に、Red Hat Quay クラスター全体を停止します。

9. 各ノードで **config.yaml** ファイルを編集し、以下のように、アップグレードモードを **complete** に設定します。

```
V3_UPGRADE_MODE: complete
```

10. Red Hat Quay を 1 つのノードで再び起動し、最終チェックを実行できるようにします。

11. 最終チェックが完了したら、Red Hat Quay v3 を他のすべてのノードでも起動します。

12. quay-builder と Clair の 3.0.z バージョンを起動して、クラスターに戻すコンテナーのインスタンスを置き換えます。

13. Docker バージョン 2、スキーマ 2 と互換性のあるコンテナーのプッシュおよびプルなど、Quay が機能していることを確認します。これには、Windows コンテナーイメージおよび異なるコンピューターアーキテクチャーのイメージ (arm、ppc など) が含まれます。

3.14.6. ターゲットイメージ

- **Quay:** quay.io/redhat/quay:v3.0.5
- **Clair:** quay.io/redhat/clair-jwt:v3.0.5
- **Redis:** registry.access.redhat.com/rhscl/redis-32-rhel7
- **PostgreSQL:** rhscl/postgresql-96-rhel7
- **Builder:** quay.io/redhat/quay-builder:v3.0.5

第4章 QUAY BRIDGE OPERATOR のアップグレード

Quay Bridge Operator (QBO) をアップグレードするには、Subscription タブの Channel Subscription 更新チャンネルを目的のチャンネルに変更します。

QBO をバージョン 3.5 から 3.7 にアップグレードする場合は、いくつかの追加の手順が必要です。

1. 新しい **QuayIntegration** カスタムリソースを作成する必要があります。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。

   upgrade-quay-integration.yaml

   ```
   - apiVersion: quay.redhat.com/v1
     kind: QuayIntegration
     metadata:
       name: example-quayintegration-new
     spec:
       clusterID: openshift (1)
       credentialsSecret:
         name: quay-integration
         namespace: openshift-operators
       insecureRegistry: false
       quayHostname: https://registry-quay-quay35.router-default.apps.cluster.openshift.com
   ```

   (1) **clusterID** が既存の **QuayIntegration** リソースの値と一致することを確認してください。

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

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

3. 古い **QuayIntegration** カスタムリソースを削除します。

4. 古い **mutatingwebhookconfigurations** を削除します。

   ```
   $ oc delete mutatingwebhookconfigurations.admissionregistration.k8s.io quay-bridge-operator
   ```