Red Hat Quay 3.11

Red Hat Quay の設定

設定オプションを使用した Red Hat Quay のカスタマイズ

Last Updated: 2024-04-12

Red Hat Quay 3.11 Red Hat Quay の設定

設定オプションを使用した Red Hat Quay のカスタマイズ

法律上の通知

概要

Red Hat Quay の設定

目次

第1章 RED HAT QUAY 設定の開始

Red Hat Quay は、独立したスタンドアロン設定でデプロイすることも、OpenShift Container Platform の Red Hat Quay Operator を使用してデプロイすることもできます。

Red Hat Quay 設定を作成、取得、更新、および検証する方法は、使用しているデプロイメントのタイプによって異なります。ただし、コア設定オプションはどちらのデプロイメントタイプでも同じです。コア設定は主に **config.yaml** ファイルを通じて設定されますが、設定 API を使用して設定することもできます。

Red Hat Quay のスタンドアロンデプロイメントの場合は、レジストリーを開始する前に、最低限必要な設定パラメーターを指定する必要があります。Red Hat Quay レジストリーを開始するための最小要件は、「現在の設定の取得」セクションに記載されています。

Red Hat Quay Operator を使用して OpenShift Container Platform に Red Hat Quay をインストールする場合、Red Hat Quay Operator はレジストリーをデプロイするためのデフォルト情報を提供するため、設定パラメーターを指定する必要はありません。

目的の設定で Red Hat Quay をデプロイしたら、デプロイから完全な設定を取得して保存する必要があります。完全な設定には、システムの再始動またはアップグレード時に必要になる可能性がある追加の生成値が含まれています。

第2章 RED HAT QUAY 設定の免責事項

一部の機能と設定パラメーターは、Red Hat Quay のスタンドアロンデプロイメントでも Operator ベースのデプロイメントでも、積極的には使用または実装されていません。そのため、一部の機能を有効または無効にするフラグなどの機能フラグや、Red Hat サポートによって明示的に文書化されていない、または文書化が要求されていない設定パラメーターは、慎重に変更する必要があります。未使用の機能やパラメーターは、完全にテストまたはサポートされていない場合や、Red Hat Quay との互換性がない場合があります。未使用の機能パラメーターを変更すると、予期しない問題やデプロイメントの停止が発生する可能性があります。

スタンドアロンデプロイメントでの Red Hat Quay の設定については、Red Hat Quay の高度な設定 を参照してください。

Red Hat Quay Operator デプロイメントの設定については、OpenShift Container Platform での Red Hat Quay の設定 を参照してください。

2.1. RED HAT QUAY 3.11 の設定更新

以下のセクションでは、Red Hat Quay 3.11 で追加された新しい設定フィールドについて詳しく説明します。

2.1.1. チーム同期の設定フィールド

OIDC 機能によるチーム同期のために、次の設定フィールドが追加されました。

表2.1 チーム同期の設定フィールド

フィールド	型	説明
推奨グループクレーム名	String	ユーザーのグループメンバーシップに関する情報を保持する OIDC トークンペイロード内のキー名。

チーム同期の例 YAML 設定

```
# ...
PREFERRED_GROUP_CLAIM_NAME: <example_claim_name>
# ...
```

2.1.2. AWS S3 STS デプロイメントの設定フィールド

AWS STS for Red Hat Quay を設定する際に、次の設定フィールドが追加されました。これらのフィールドは、デプロイメント用に AWS S3 ストレージを設定するときに使用されます。

表2.2 AWS S3 STS 設定フィールド

フィールド	型	説明
.sts_role_arn	String	AWS STS for Red Hat Quay を設定するときに必要な一意の Amazon Resource Name (ARN)。

.sts_ ユーザーアクセスキー	String	AWS STS for Red Hat Quay を設定するときに必要な、生成された AWS S3 ユーザー秘密鍵。
.sts_ ユーザーシークレットキー	String	Red Hat Quay の AWS STS を設定するときに必要な、生成された AWS S3 ユーザー秘密鍵。

AWS S3 STS の例 YAML 設定

```
# ...
DISTRIBUTED_STORAGE_CONFIG:
  default:
   - STSS3Storage
   - sts_role_arn: <role_arn>
     s3_bucket: <s3_bucket_name>
     storage_path: <storage_path>
     sts_user_access_key: <s3_user_access_key>
     sts_user_secret_key: <s3_user_secret_key>
# ...
```

2.2. 設定ファイルの編集

Red Hat Quay のスタンドアロンインスタンスをデプロイするには、最小限の設定情報を提供する必要があります。最小設定の要件は、「Red Hat Quay の最小設定」に記載されています。

必須フィールドに入力したら、設定を検証できます。問題がある場合は強調表示されます。

注記

設定 API を使用して設定を検証することもできますが、検証するには **Quay** コンテナーを設定モードで起動する必要があります。

設定ツールをローカルにデプロイするには、Red Hat Quay のスタートガイド を参照し、「Red Hat Quay の設定」までの手順を実行してください。

変更を有効にするには、レジストリーを再起動する必要があります。

2.3. スタンドアロンデプロイメントにおける設定ファイルの場所

Red Hat Quay のスタンドアロンデプロイメントの場合、Red Hat Quay レジストリーを開始するときに **config.yaml** ファイルを指定する必要があります。このファイルは設定ボリュームにあります。たとえば、次のコマンドで Red Hat Quay をデプロイする場合、設定ファイルは **$QUAY/config/config.yaml** にあります。

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -v $QUAY/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.0
```

2.4. 最小設定

Red Hat Quay のスタンドアロンデプロイメントには、以下の設定オプションが必要です。

- サーバーのホスト名
- HTTP または HTTPS
- 認証タイプ (データベースや LDAP (Lightweight Directory Access Protocol) など)
- データ暗号化用の秘密鍵
- イメージのストレージ
- メタデータ用のデータベース
- ビルドログおよびユーザーイベント用の Redis
- タグの有効期限オプション

2.4.1. 最小設定ファイルの例

次の例は、イメージにローカルストレージを使用する最小限の設定ファイルの例を示しています。

```
AUTHENTICATION_TYPE: Database
BUILDLOGS_REDIS:
    host: quay-server.example.com
    password: strongpassword
    port: 6379
    ssl: false
DATABASE_SECRET_KEY: 0ce4f796-c295-415b-bf9d-b315114704b8
DB_URI: postgresql://quayuser:quaypass@quay-server.example.com:5432/quay
DEFAULT_TAG_EXPIRATION: 2w
DISTRIBUTED_STORAGE_CONFIG:
    default:
        - LocalStorage
        - storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - default
PREFERRED_URL_SCHEME: http
SECRET_KEY: e8f9fe68-1f84-48a8-a05f-02d72e6eccba
SERVER_HOSTNAME: quay-server.example.com
SETUP_COMPLETE: true
TAG_EXPIRATION_OPTIONS:
    - 0s
    - 1d
    - 1w
    - 2w
    - 4w
USER_EVENTS_REDIS:
    host: quay-server.example.com
    port: 6379
    ssl: false
```

2.4.2. ローカルストレージ

イメージへのローカルストレージの使用は、**概念実証** の目的のためにレジストリーをデプロイする場合に限り推奨されます。

ローカルストレージを設定する場合は、レジストリーの起動時にコマンドラインでストレージを指定します。

次のコマンドは、ローカルディレクトリー **$QUAY/storage** をコンテナー内の **datastorage** ストレージパスにマップします。

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -v $QUAY/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.0
```

2.4.3. クラウドストレージ

ストレージの設定は、イメージストレージ セクションを参照してください。一部のユーザーにとっては、Google Cloud Platform とローカルストレージ設定の違いを比較すると役立つ場合があります。たとえば、次の YAML は Google Cloud Platform のストレージ設定を表しています。

$QUAY/config/config.yaml

```
DISTRIBUTED_STORAGE_CONFIG:
    default:
        - GoogleCloudStorage
        - access_key: GOOGQIMFB3ABCDEFGHIJKLMN
          bucket_name: quay_bucket
          secret_key: FhDAYe2HeuAKfvZCAGyOioNaaRABCDEFGHIJKLMN
          storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - default
```

クラウドストレージを使用してレジストリーを起動する場合は、コマンドラインでの設定が必要ありません。以下に例を示します。

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.0
```

第3章 設定フィールド

このセクションでは、Red Hat Quay のデプロイ時に必須および任意の設定フィールドの両方を説明します。

3.1. 必須の設定フィールド

Red Hat Quay の設定で必須のフィールドは、以下のセクションで説明されています。

- 一般的な必須フィールド
- イメージのストレージ
- メタデータ用のデータベース
- ビルドログおよびユーザーイベント用の Redis
- タグの有効期限オプション

3.2. 自動化オプション

以下のセクションでは、Red Hat Quay デプロイメントで利用可能な自動化オプションを説明します。

- 自動化のための Red Hat Quay の事前設定
- API を使用した最初のユーザーの作成

3.3. 任意の設定フィールド

Red Hat Quay の任意のフィールドは、以下のセクションで説明します。

- Basic configuration
- SSL
- LDAP
- リポジトリーのミラーリング
- クォータ管理
- セキュリティースキャナー
- Helm
- アクションログ
- ビルドログ
- Dockerfile ビルド
- OAuth
- ネストされたリポジトリーの設定

- その他の OCI メディアタイプの Quay への追加
- メール
- ユーザー
- reCAPTCHA
- ACI
- JWT
- アプリケーショントークン
- その他
- ユーザーインターフェイス v2
- IPv6 設定フィールド
- レガシーオプション

3.4. 一般的な必須フィールド

以下の表は、Red Hat Quay デプロイメントの必須設定フィールドを説明しています。

表3.1 一般的な必須フィールド

フィールド	型	説明
AUTHENTICATION_TYPE (必須)	String	認証情報の認証に使用する認証エンジン。 **値:** **Database**、**LDAP**、**JWT**、**Keystone**、**OIDC** のいずれか。 **デフォルト: Database**
PREFERRED_URL_SCHEME (必須)	String	Red Hat Quay へのアクセスに使用する URL スキーム。 **値:** **http**, **https** のいずれか。 **デフォルト: http**
SERVER_HOSTNAME (必須)	String	スキームなしで Red Hat Quay にアクセスできる URL。 **例:** **quay-server.example.com**

フィールド	型	説明
DATABASE_SECRET_KEY (必須)	String	データベース内で機密フィールドを暗号化するのに使用されるキー。この値は、一度設定したら変更しないでください。変更すると、リポジトリーのミラーユーザー名やパスワード設定など、すべての信頼できるフィールドが無効になります。
SECRET_KEY (必須)	String	ユーザーセッションを正しく解釈するために必要なセッション Cookie と CSRF トークンを暗号化するために使用されるキー。設定時に値を変更しないでください。すべての Red Hat Quay インスタンスで永続的である必要があります。すべてのインスタンスで永続的でない場合、ログインの失敗やセッションの永続性に関連するその他のエラーが発生する可能性があります。
SETUP_COMPLETE (必須)	Boolean	これは、以前のバージョンのソフトウェアからそのまま残っているアーティファクトです。現時点では値を **true** に指定する **必要があります**。

3.5. データベースの設定

このセクションでは、Red Hat Quay デプロイメントで利用可能なデータベース設定フィールドを説明します。

3.5.1. データベース URI

Red Hat Quay では、必要な **DB_URI** フィールドを使用してデータベースへの接続を設定します。

以下の表は **DB_URI** 設定フィールドを説明します。

表3.2 データベース URI

フィールド	型	説明

フィールド	型	説明
DB_URI (必須)	String	認証情報を含む、データベースにアクセスするための URI。 **DB_URI** フィールドの例: **postgresql://quayuser:quaypass@quay-server.example.com:5432/quay**

3.5.2. データベース接続引数

オプションの接続引数は、**DB_CONNECTION_ARGS** パラメーターで設定されます。**DB_CONNECTION_ARGS** で定義されたキーと値のペアの一部は汎用的なものも、データベース固有のものもあります。

以下の表は、データベース接続引数を説明します。

表3.3 データベース接続引数

フィールド	型	説明
DB_CONNECTION_ARGS	Object	タイムアウトや SSL/TLS などのデータベースの任意の接続引数。
.autorollback	Boolean	スレッドローカル接続を使用するかどうか。 常に **true** にする必要があります。
.threadlocals	Boolean	自動ロールバック接続を使用するかどうか。 常に **true** にする必要があります。

3.5.2.1. PostgreSQL SSL/TLS 接続引数

SSL/TLS では、設定はデプロイするデータベースによって異なります。次の例は、PostgreSQL SSL/TLS 設定を示しています。

```
DB_CONNECTION_ARGS:
  sslmode: verify-ca
  sslrootcert: /path/to/cacert
```

sslmode オプションは、セキュアな SSL/TLS TCP/IP 接続がサーバーにネゴシエートされるかどうか、その優先度を決定します。モードは 6 つあります。

表3.4 SSL/TLS オプション

モード	説明
disable	この設定では、非 SSL/TLS 接続のみが試行されます。
allow	設定では、まず非 SSL/TLS 接続が試行されます。失敗すると、SSL/TLS 接続が試行されます。
prefer (デフォルト)	設定では、まず SSL/TLS 接続が試行されます。失敗すると、非 SSL/TLS 接続が試行されます。
require	設定では SSL/TLS 接続のみが試行されます。ルート CA ファイルが存在する場合は、verify-ca が指定されているのと同じ方法で証明書が検証されます。
verify-ca	設定では SSL/TLS 接続のみが試行され、サーバー証明書が信頼された認証局 (CA) によって発行されたかどうかが検証されます。
verify-full	SSL/TLS 接続のみを試行し、サーバー証明書が信頼できる CA によって発行されていること、および要求されたサーバーのホスト名が証明書内のホスト名と一致することが確認されます。

PostgreSQL の有効な引数の詳細は、Database Connection Control Functions を参照してください。

3.5.2.2. MySQL SSL/TLS 接続引数

次の例は、MySQL SSL/TLS 設定のサンプルを示しています。

```
DB_CONNECTION_ARGS:
  ssl:
    ca: /path/to/cacert
```

MySQL の有効な接続引数に関する情報は、Connecting to the Server Using URI-Like Strings or Key-Value Pairs を参照してください。

3.6. イメージストレージ

このセクションでは、Red Hat Quay で利用可能なイメージストレージ機能と設定フィールドを説明します。

3.6.1. イメージストレージ機能

以下の表は、Red Hat Quay のイメージストレージ機能を説明しています。

表3.5 ストレージ設定機能

フィールド	型	説明
FEATURE_REPO_MIRROR	Boolean	true に設定されている場合は、リポジトリーのミラーリングを有効にします。 **デフォルト:false**
FEATURE_PROXY_STORAGE	Boolean	NGINX を使用してストレージ内のすべての直接ダウンロード URL をプロキシーするかどうか。 **デフォルト:false**
FEATURE_STORAGE_REPLICATION	Boolean	ストレージエンジン間で自動的にレプリケートするかどうか。 **のデフォルト:false**

3.6.2. イメージストレージ設定フィールド

以下の表は、Red Hat Quay のイメージストレージ設定フィールドを説明しています。

表3.6 ストレージ設定フィールド

フィールド	型	説明
DISTRIBUTED_STORAGE_CONFIG (必須)	Object	Red Hat Quay で使用するストレージエンジンの設定。各キーは、ストレージエンジンの一意の ID を表します。この値は、ストレージエンジンパラメーターを記述するオブジェクトのタプル (キー、値) で設定されます。 **デフォルト:[]**
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS (必須)	Array of string	イメージをデフォルトで他のすべてのストレージエンジンに対して完全にレプリケートする必要のあるストレージエンジンの一覧 (**DISTRIBUTED_STORAGE_CONFIG** の ID 別)。

フィールド	型	説明
DISTRIBUTED_STORAGE_PREFERENCE (必須)	Array of string	使用する優先ストレージエンジン（**DISTRIBUTED_STORAGE_CONFIG**の ID 別）。優先エンジンとは、プルする必要がないかを先にチェックしてから、イメージがプッシュされることを意味します。 **デフォルト:false**
MAXIMUM_LAYER_SIZE	String	イメージレイヤーの最大許容サイズ。 **パターン**: **^[0-9]+(G|M)$** **例**: **100G** **デフォルト:20G**

3.6.3. ローカルストレージ

以下の YAML は、ローカルストレージを使用した設定のサンプルを示しています。

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
    - LocalStorage
    - storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - default
```

3.6.4. OCS/NooBaa

以下の YAML は、Open Container Storage/NooBaa インスタンスを使用した設定例を示しています。

```
DISTRIBUTED_STORAGE_CONFIG:
  rhocsStorage:
    - RHOCSStorage
    - access_key: access_key_here
      secret_key: secret_key_here
      bucket_name: quay-datastore-9b2108a3-29f5-43f2-a9d5-2872174f9a56
      hostname: s3.openshift-storage.svc.cluster.local
      is_secure: 'true'
      port: '443'
      storage_path: /datastorage/registry
```

3.6.5. Ceph/RadosGW ストレージ

次の例は、Ceph/RadosGW を使用する場合に考えられる 2 つの YAML 設定を示しています。

例 A: radosGWStorage ドライバーで RadosGW を使用する

```
DISTRIBUTED_STORAGE_CONFIG:
  radosGWStorage:
    - RadosGWStorage
    - access_key: <access_key_here>
      secret_key: <secret_key_here>
      bucket_name: <bucket_name_here>
      hostname: <hostname_here>
      is_secure: true
      port: '443'
      storage_path: /datastorage/registry
```

例 B: 一般的な s3 アクセスで RadosGW を使用する

```
DISTRIBUTED_STORAGE_CONFIG:
  s3Storage: 1
    - RadosGWStorage
    - access_key: <access_key_here>
      bucket_name: <bucket_name_here>
      hostname: <hostname_here>
      is_secure: true
      secret_key: <secret_key_here>
      storage_path: /datastorage/registry
```

1 一般的な s3 アクセスに使用します。一般的な s3 アクセスは、厳密には Amazon Web Services (AWS) 3 に限定されず、RadosGW または他のストレージサービスでも使用できることに注意してください。AWS S3 ドライバーを使用した一般的な s3 アクセスの例については、「AWS S3 ストレージ」を参照してください。

3.6.6. AWS S3 ストレージ

以下の YAML は、AWS S3 ストレージを使用した設定のサンプルを示しています。

```
# ...
DISTRIBUTED_STORAGE_CONFIG:
  default:
    - S3Storage 1
    - host: s3.us-east-2.amazonaws.com
      s3_access_key: ABCDEFGHIJKLMN
      s3_secret_key: OL3ABCDEFGHIJKLMN
      s3_bucket: quay_bucket
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - s3Storage
# ...
```

1 **S3Storage** ストレージドライバーは、AWS S3 バケットにのみ使用してください。これは、RadosGW ドライバーや他のストレージサービスを使用できる一般的な S3 アクセスとは異なることに注意してください。例については、「例 B: 一般的な S3 アクセスで RadosGW を使用する」を参照してください。

3.6.6.1. AWS STS S3 ストレージ

次の YAML は、OpenShift Container Platform 設定で Amazon Web Services (AWS) Security Token Service (STS) を Red Hat Quay で使用するための設定例を示しています。

```
# ...
DISTRIBUTED_STORAGE_CONFIG:
  default:
   - STSS3Storage
   - sts_role_arn: <role_arn> (1)
     s3_bucket: <s3_bucket_name>
     storage_path: <storage_path>
     sts_user_access_key: <s3_user_access_key> (2)
     sts_user_secret_key: <s3_user_secret_key> (3)
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
   - default
# ...
```

(1) 一意の Amazon Resource Name (ARN)。

(2) 生成された AWS S3 ユーザーアクセスキー。

(3) 生成された AWS S3 ユーザー秘密鍵。

3.6.7. Google Cloud Storage

以下の YAML は、Google Cloud Storage を使用した設定例を示しています。

```
DISTRIBUTED_STORAGE_CONFIG:
   googleCloudStorage:
      - GoogleCloudStorage
      - access_key: GOOGQIMFB3ABCDEFGHIJKLMN
        bucket_name: quay-bucket
        secret_key: FhDAYe2HeuAKfvZCAGyOioNaaRABCDEFGHIJKLMN
        storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
   - googleCloudStorage
```

3.6.8. Azure Storage

以下の YAML は、Azure Storage を使用した設定のサンプルを示しています。

```
DISTRIBUTED_STORAGE_CONFIG:
  azureStorage:
   - AzureStorage
   - azure_account_name: azure_account_name_here
     azure_container: azure_container_here
     storage_path: /datastorage/registry
     azure_account_key: azure_account_key_here
     sas_token: some/path/
     endpoint_url: https://[account-name].blob.core.usgovcloudapi.net (1)
```

```
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
  - azureStorage
```

1 Azure ストレージの **endpoint_url** パラメーターは任意であり、Microsoft Azure Government (MAG) エンドポイントで使用できます。空白のままにすると、**endpoint_url** は通常の Azure リージョンに接続します。

Red Hat Quay 3.7 以降では、MAG Blob サービスのプライマリーエンドポイントを使用する必要があります。MAG Blob サービスのセカンダリーエンドポイントを使用すると、**AuthenticationErrorDetail:Cannot find the claimed account when trying to GetProperties for the account whusc8-secondary** エラーが発生します。

3.6.9. Swift ストレージ:

以下の YAML は、Swift ストレージを使用した設定のサンプルを示しています。

```
DISTRIBUTED_STORAGE_CONFIG:
  swiftStorage:
    - SwiftStorage
    - swift_user: swift_user_here
      swift_password: swift_password_here
      swift_container: swift_container_here
      auth_url: https://example.org/swift/v1/quay
      auth_version: 1
      ca_cert_path: /conf/stack/swift.cert"
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - swiftStorage
```

3.6.10. Nutanix オブジェクトストレージ

以下の YAML は、Nutanix オブジェクトストレージを使用した設定例を示しています。

```
DISTRIBUTED_STORAGE_CONFIG:
  nutanixStorage: #storage config name
    - RadosGWStorage #actual driver
    - access_key: access_key_here #parameters
      secret_key: secret_key_here
      bucket_name: bucket_name_here
      hostname: hostname_here
      is_secure: 'true'
      port: '443'
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE: #must contain name of the storage config
    - nutanixStorage
```

3.6.11. IBM Cloud オブジェクトストレージ

次の YAML は、IBM Cloud オブジェクトストレージを使用したサンプル設定を示しています。

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
  - IBMCloudStorage #actual driver
  - access_key: <access_key_here> #parameters
    secret_key: <secret_key_here>
    bucket_name: <bucket_name_here>
    hostname: <hostname_here>
    is_secure: 'true'
    port: '443'
    storage_path: /datastorage/registry
    maximum_chunk_size_mb: 100mb (1)
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS:
- default
DISTRIBUTED_STORAGE_PREFERENCE:
- default
```

1 オプション: **100mb** に設定することを推奨します。

3.7. REDIS 設定フィールド

このセクションでは、Redis デプロイメントで利用可能な設定フィールドを説明します。

3.7.1. ビルドログ

Redis デプロイメントには、以下のビルドログ設定フィールドを利用できます。

表3.7 ビルドログの設定

フィールド	型	説明
BUILDLOGS_REDIS (必須)	Object	ビルドログキャッシュ用の Redis 接続の詳細。
.host (必須)	String	Redis にアクセスできるホスト名。 **例:** **quay-server.example.com**
.port (必須)	Number	Redis にアクセスできるポート。 **例:** **6379**
.password	String	Redis インスタンスに接続するためのパスワード。 **例:** **strongpassword**
.ssl (オプション)	Boolean	Redis と Quay 間の TLS 通信を有効にするかどうか。デフォルトは false です。

3.7.2. ユーザーイベント

Redis デプロイメントには、以下のユーザーイベントフィールドを使用できます。

表3.8 ユーザーイベント設定

フィールド	型	説明
USER_EVENTS_REDIS (必須)	Object	ユーザーイベント処理の Redis 接続の詳細。
.host (必須)	String	Redis にアクセスできるホスト名。 **例:** **quay-server.example.com**
.port (必須)	Number	Redis にアクセスできるポート。 **例:** **6379**
.password	String	Redis インスタンスに接続するためのパスワード。 **例:** **strongpassword**
.ssl	Boolean	Redis と Quay 間の TLS 通信を有効にするかどうか。デフォルトは false です。
.ssl_keyfile (オプション)	String	使用するクライアント証明書を格納する鍵データベースファイルの名前。 **例:** **ssl_keyfile:/path/to/server/privatekey.pem**
.ssl_certfile (オプション)	String	SSL 証明書のファイルパスを指定するために使用されます。 **例:** **ssl_certfile:/path/to/server/certificate.pem**
.ssl_cert_reqs (オプション)	String	SSL/TLS ハンドシェーク中に実行される証明書検証のレベルを指定するために使用されます。 **例:** **ssl_cert_reqs: CERT_REQUIRED**

フィールド	型	説明
.ssl_ca_certs (オプション)	String	信頼された認証局 (CA) 証明書のリストを含むファイルへのパスを指定するために使用されます。 **例:** **ssl_ca_certs:/path/to/ca_certs.pem**
.ssl_ca_data (オプション)	String	信頼できる CA 証明書を含む文字列を PEM 形式で指定するために使用されます。 **例:** **ssl_ca_data: <certificate>**
.ssl_check_hostname (オプション)	Boolean	サーバーへの SSL/TLS 接続をセットアップするときに使用されます。サーバーの SSL/TLS 証明書のホスト名が、接続先のサーバーのホスト名と一致することをクライアントが確認する必要があるかどうかを指定します。 **例:** **ssl_check_hostname: true**

3.7.3. redis の設定例

次の YAML は、オプションの SSL/TLS フィールドを持つ Redis を使用したサンプル設定を示しています。

```
BUILDLOGS_REDIS:
  host: quay-server.example.com
  password: strongpassword
  port: 6379
  ssl: true


USER_EVENTS_REDIS:
  host: quay-server.example.com
  password: strongpassword
  port: 6379
  ssl: true
  ssl_*: <path_location_or_certificate>
```

注記

デプロイで Azure Cache for Redis を使用し、**ssl** が **true** に設定されていると、ポートはデフォルトで **6380** になります。

3.8. MODELCACHE 設定オプション

以下のオプションは、ModelCache を設定するために Red Hat Quay で利用できます。

3.8.1. memcache 設定オプション

memcache は、デフォルトの ModelCache 設定オプションです。memcache を使用すると、追加の設定は必要ありません。

3.8.2. 単一の Redis 設定オプション

以下の設定は、オプションの読み取り専用レプリカを持つ単一の Redis インスタンス用です。

```
DATA_MODEL_CACHE_CONFIG:
  engine: redis
  redis_config:
    primary:
        host: <host>
        port: <port>
        password: <password if ssl is true>
       ssl: <true | false >
    replica:
        host: <host>
        port: <port>
        password: <password if ssl is true>
       ssl: <true | false >
```

3.8.3. クラスター化された Redis 設定オプション

クラスター化された Redis インスタンスには、次の設定を使用します。

```
DATA_MODEL_CACHE_CONFIG:
  engine: rediscluster
  redis_config:
    startup_nodes:
      - host: <cluster-host>
        port: <port>
    password: <password if ssl: true>
    read_from_replicas: <true|false>
    skip_full_coverage_check: <true | false>
    ssl: <true | false >
```

3.9. タグの有効期限の設定フィールド

以下のタグの有効期限設定フィールドは Red Hat Quay で利用できます。

表3.9 タグの有効期限の設定フィールド

フィールド	型	説明
FEATURE_GARBAGE_COLLECTION	Boolean	リポジトリーのガベージコレクションを有効にするかどうか。 **デフォルト:** True

フィールド	型	説明
TAG_EXPIRATION_OPTIONS (必須)	Array of string	有効にすると、名前空間内のタグの有効期限についてユーザーが選択できるオプション。 **パターン:** **^[0-9]+(w|m|d|h|s)$**
DEFAULT_TAG_EXPIRATION (必須)	String	タイムマシンのデフォルトの設定可能なタグの有効期限。 **パターン:** **^[0-9]+(w|m|d|h|s)$** **デフォルト 2w**
FEATURE_CHANGE_TAG_EXPIRATION	Boolean	ユーザーおよび組織が namespace のタグの有効期限を変更できるかどうか。 **デフォルト:** True
FEATURE_AUTO_PRUNE	Boolean	**True** に設定すると、タグの自動プルーニングに関連する機能が有効になります。 **デフォルト:** **False**

3.9.1. タグの有効期限の設定例

以下の YAML は、タグの有効期限の設定例を示しています。

```
DEFAULT_TAG_EXPIRATION: 2w
TAG_EXPIRATION_OPTIONS:
    - 0s
    - 1d
    - 1w
    - 2w
    - 4w
```

3.10. クォータ管理設定フィールド

表3.10 クォータ管理設定

フィールド	型	説明
FEATURE_QUOTA_MANAGEMENT	Boolean	クォータ管理機能の設定、キャッシュ、検証を有効にします。 **Default:** `False`

フィールド	型	説明
DEFAULT_SYSTEM_REJECT_QUOTA_BYTES	String	すべての組織に対してシステムのデフォルトクォータ拒否バイト許容量を有効にします。 デフォルトでは、制限は設定されていません。
QUOTA_BACKFILL	Boolean	クォータバックフィルワーカーが既存の BLOB のサイズを計算できるようにします。 **デフォルト**: **True**
QUOTA_TOTAL_DELAY_SECONDS	String	クォータバックフィルを開始するまでの遅延時間。ローリングデプロイメントでは、合計が不正確になる可能性があります。このフィールドは、ローリングデプロイメントが完了するまでにかかる時間よりも長い時間を設定する**必要があります**。 **デフォルト**: **1800**
PERMANENTLY_DELETE_TAGS	Boolean	タイムマシンウィンドウからのタグの削除に関連する機能を有効にします。 **デフォルト**:**False**
RESET_CHILD_MANIFEST_EXPIRATION	Boolean	子マニフェストを対象とする一時タグの有効期限をリセットします。この機能を **True** に設定すると、子マニフェストはすぐにガベージコレクションされます。 **デフォルト**:**False**

3.10.1. クォータ管理設定の例

次の YAML は、クォータ管理を有効にするときに推奨される設定です。

クォータ管理 YAML 設定

```
FEATURE_QUOTA_MANAGEMENT: true
FEATURE_GARBAGE_COLLECTION: true
PERMANENTLY_DELETE_TAGS: true
QUOTA_TOTAL_DELAY_SECONDS: 1800
RESET_CHILD_MANIFEST_EXPIRATION: true
```

3.11. プロキシーキャッシュ設定フィールド

表3.11 プロキシー設定

フィールド	型	説明
FEATURE_PROXY_CACHE	Boolean	Red Hat Quay がアップストリームレジストリーのプルスルーキャッシュとして機能できるようにします。 **デフォルト**: **false**

3.12. ロボットアカウント設定フィールド

表3.12 ロボットアカウント設定フィールド

フィールド	型	説明
ROBOTS_DISALLOW	Boolean	**true** に設定すると、ロボットアカウントの作成だけでなく、すべてのインタラクションが禁止されます。 **デフォルト**: **False**

3.13. 自動化のための RED HAT QUAY の事前設定

Red Hat Quay は、自動化を可能にするいくつかの設定オプションをサポートします。ユーザーはデプロイメント前にこれらのオプションを設定して、ユーザーインターフェイスとの対話の必要性を減らすことができます。

3.13.1. API による最初のユーザー作成の許可

最初のユーザーを作成するには、**FEATURE_USER_INITIALIZE** パラメーターを **true** に設定し、**/api/v1/user/initialize** API を呼び出す必要があります。既存の組織の OAuth アプリケーションによって生成された OAuth トークンを必要とする他のすべてのレジストリー API 呼び出しとは異なり、API エンドポイントは認証を必要としません。

他のユーザーが作成されていない場合、ユーザーは Red Hat Quay のデプロイ後に API を使用して **quayadmin** などのユーザーを作成できます。詳細は、API を使用して最初のユーザーを作成する を参照してください。

3.13.2. API 一般アクセスの有効化

Red Hat Quay レジストリー API への一般アクセスを許可するには、**BROWSER_API_CALLS_XHR_ONLY** 設定オプションを **false** に設定する必要があります。

3.13.3. スーパーユーザーの追加

Red Hat Quay をデプロイしたら、ユーザーを作成し、最初のユーザーに完全な権限を持つ管理者権限を付与できます。**SUPER_USER** 設定オブジェクトを使用すると、事前に完全な権限を設定できます。以下に例を示します。

```
# ...
SERVER_HOSTNAME: quay-server.example.com
SETUP_COMPLETE: true
SUPER_USERS:
  - quayadmin
# ...
```

3.13.4. ユーザー作成の制限

スーパーユーザーを設定した後、**FEATURE_USER_CREATION** を **false** に設定することで、新しいユーザーを作成できる権限をスーパーユーザーグループに制限できます。以下に例を示します。

```
# ...
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- quayadmin
FEATURE_USER_CREATION: false
# ...
```

3.13.5. Red Hat Quay 3.11 での新機能の有効化

新しい Red Hat Quay 3.11 の機能を使用するには、次の機能の一部またはすべてを有効にします。

```
# ...
FEATURE_UI_V2: true
FEATURE_UI_V2_REPO_SETTINGS: true
FEATURE_AUTO_PRUNE: true
ROBOTS_DISALLOW: false
# ...
```

3.13.6. 自動化の推奨設定

自動化には、以下の **config.yaml** パラメーターが推奨されます。

```
# ...
FEATURE_USER_INITIALIZE: true
BROWSER_API_CALLS_XHR_ONLY: false
SUPER_USERS:
- quayadmin
FEATURE_USER_CREATION: false
# ...
```

3.13.7. 初期設定を使用した Red Hat Quay Operator のデプロイ

以下の手順に従って、初期設定を使用して OpenShift Container Platform に Red Hat Quay をデプロイします。

前提条件

前提条件

- **oc** CLI がインストールされている。

手順

1. 設定ファイルを使用してシークレットを作成します。

```
$ oc create secret generic -n quay-enterprise --from-file config.yaml=./config.yaml init-config-bundle-secret
```

2. **quayregistry.yaml** ファイルを作成します。以下のように、管理対象外のコンポーネントを特定し、作成されたシークレットを参照します。

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  configBundleSecret: init-config-bundle-secret
```

3. Red Hat Quay レジストリーをデプロイします。

```
$ oc create -n quay-enterprise -f quayregistry.yaml
```

次のステップ

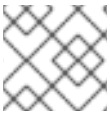
- API を使用した最初のユーザーの作成

3.13.8. API を使用した最初のユーザーの作成

以下の手順に従って、Red Hat Quay 組織で最初のユーザーを作成します。

前提条件

- 設定オプション **FEATURE_USER_INITIALIZE** は **true** に設定する。
- データベースにユーザーが存在していない。



手順

この手順では、**"access_token": true** を指定して OAuth トークンを要求します。

1. Red Hat Quay 設定ファイルを開き、以下の設定フィールドを更新します。

```
FEATURE_USER_INITIALIZE: true
SUPER_USERS:
    - quayadmin
```

2. 次のコマンドを入力して、Red Hat Quay サービスを停止します。

```
$ sudo podman stop quay
```

3. 次のコマンドを入力して、Red Hat Quay サービスを開始します。

```
$ sudo podman run -d -p 80:8080 -p 443:8443 --name=quay -v $QUAY/config:/conf/stack:Z
-v $QUAY/storage:/datastorage:Z {productrepo}/{quayimage}:{productminv}
```

4. 次の **CURL** コマンドを実行して、ユーザー名、パスワード、電子メール、およびアクセストークンを使用して新しいユーザーを生成します。

```
$ curl -X POST -k  http://quay-server.example.com/api/v1/user/initialize --header 'Content-
Type: application/json' --data '{ "username": "quayadmin", "password":"quaypass12345",
"email": "quayadmin@example.com", "access_token": true}'
```

成功すると、このコマンドはユーザー名、メール、および暗号化されたパスワードが含まれるオブジェクトを返します。以下に例を示します。

```
{"access_token":"6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED",
"email":"quayadmin@example.com","encrypted_password":"1nZMLH57RIE5UGdL/yYpDOHLqiNCgimb6W9kfF8MjZ1xrfDpRyRs9NUnUuNuAitW","username":"quayadmin"} #
gitleaks:allow
```

データベースにユーザーが存在している場合は、エラーが返されます。

```
{"message":"Cannot initialize user in a non-empty database"}
```

パスワードが 8 文字以上でない場合や、空白が含まれている場合には、エラーが返されます。

```
{"message":"Failed to initialize user: Invalid password, password must be at least 8
characters and contain no whitespace."}
```

5. 以下のコマンドを入力して、Red Hat Quay デプロイメントにログインします。

```
$ sudo podman login -u quayadmin -p quaypass12345 http://quay-server.example.com --tls-
verify=false
```

出力例

```
Login Succeeded!
```

3.13.8.1. OAuth トークンの使用

API の呼び出し後に、返される OAuth コードを指定して残りの Red Hat Quay API を呼び出すことができます。

前提条件

- **/api/v1/user/initialize** API を呼び出し、ユーザー名、パスワード、およびメールアドレスに渡している。

手順

- 以下のコマンドを入力して、現在のユーザーの一覧を取得します。

```
$ curl -X GET -k -H "Authorization: Bearer
6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED" https://example-registry-quay-
quay-enterprise.apps.docs.quayteam.org/api/v1/superuser/users/
```

出力例:

```
{
   "users": [
      {
         "kind": "user",
         "name": "quayadmin",
         "username": "quayadmin",
         "email": "quayadmin@example.com",
         "verified": true,
         "avatar": {
            "name": "quayadmin",
            "hash":
"3e82e9cbf62d25dec0ed1b4c66ca7c5d47ab9f1f271958298dea856fb26adc4c",
            "color": "#e7ba52",
            "kind": "user"
         },
         "super_user": true,
         "enabled": true
      }
   ]
}
```

このインスタンスでは、これまで作成した唯一のユーザーであるため、**quayadmin** ユーザーの詳細が返されます。

3.13.8.2. API を使用した組織の作成

以下の手順では、API を使用して Red Hat Quay 組織を作成する方法を説明します。

前提条件

- **/api/v1/user/initialize** API を呼び出し、ユーザー名、パスワード、およびメールアドレスに渡している。
- 返された OAuth コードを指定して、残りの Red Hat Quay API を呼び出している。

手順

1. 組織を作成するには、**api/v1/organization/** エンドポイントへの POST 呼び出しを使用します。

```
$ curl -X POST -k --header 'Content-Type: application/json' -H "Authorization: Bearer
6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED" https://example-registry-quay-
quay-enterprise.apps.docs.quayteam.org/api/v1/organization/ --data '{"name": "testorg",
"email": "testorg@example.com"}'
```

出力例:

```
"Created"
```

2. 以下のコマンドを入力して、作成した組織の詳細を取得できます。

```
$ curl -X GET -k --header 'Content-Type: application/json' -H "Authorization: Bearer
6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED" https://min-registry-quay-quay-
enterprise.apps.docs.quayteam.org/api/v1/organization/testorg
```

出力例:


```
{
    "name": "testorg",
    "email": "testorg@example.com",
    "avatar": {
        "name": "testorg",
        "hash": "5f113632ad532fc78215c9258a4fb60606d1fa386c91b141116a1317bf9c53c8",
        "color": "#a55194",
        "kind": "user"
    },
    "is_admin": true,
    "is_member": true,
    "teams": {
        "owners": {
            "name": "owners",
            "description": "",
            "role": "admin",
            "avatar": {
                "name": "owners",
                "hash":
"6f0e3a8c0eb46e8834b43b03374ece43a030621d92a7437beb48f871e90f8d90",
                "color": "#c7c7c7",
                "kind": "team"
            },
            "can_view": true,
            "repo_count": 0,
            "member_count": 1,
            "is_synced": false
        }
    },
    "ordered_teams": [
        "owners"
    ],
    "invoice_email": false,
    "invoice_email_address": null,
    "tag_expiration_s": 1209600,
    "is_free_account": true
}
```


3.14. 基本設定フィールド

表3.13 Basic configuration

フィールド	型	説明

フィールド	型	説明
REGISTRY_TITLE	String	指定されている場合、レジストリーの長いタイトル。Red Hat Quay デプロイメントのフロントエンド (組織のサインインページなど) に表示されます。35 文字を超えないようにしてください。 **デフォルト:** **Red Hat Quay**
REGISTRY_TITLE_SHORT	String	指定されている場合は、省略形式のレジストリーのタイトル。タイトルは、組織のさまざまなページに表示されます。たとえば、組織の Tutorial ページのチュートリアルのタイトルとして表示されます。 **デフォルト:** **Red Hat Quay**
CONTACT_INFO	Array of string	指定されている場合は、連絡先ページに表示される連絡先情報。連絡先が 1 つしか指定されていない場合は、連絡先のフッターが直接リンクされます。
[0]	String	電子メールを送信するためのリンクを追加します。 **パターン:** **^mailto:(.)+$** **例:** **mailto:support@quay.io**
[1]	String	IRC チャットルームにアクセスするためのリンクを追加します。 **パターン:** **^irc://(.)+$** **例:** **irc://chat.freenode.net:6665/quay**
[2]	String	電話番号を呼び出すリンクを追加します。 **パターン:** **^tel:(.)+$** **例:** **tel:+1-888-930-3475**

フィールド	型	説明
[3]	String	定義された URL へのリンクを追加します。 **パターン:** **^http(s)?://(.)+$** **Example:** **https://twitter.com/quayio**

3.15. SSL 設定フィールド

表3.14 SSL 設定

フィールド	型	説明
PREFERRED_URL_SCHEME	String	**http** または **https** のいずれか。クライアントから Quay への通信パスに TLS 暗号化がない場合に限り、ユーザーは **PREFERRED_URL_SCHEME** を **http** に設定することに注意してください。 TLS を終了するロードバランサー、リバースプロキシー (Nginx など) を使用する場合、またはカスタム SSL 証明書で Quay を直接使用する場合、ユーザーは **PREFERRED_URL_SCHEME** を https に設定する必要があります。ほとんどの場合、**PREFERRED_URL_SCHEME** は **https** である必要があります。 **デフォルト: http**
SERVER_HOSTNAME (必須)	String	スキームなしで Red Hat Quay にアクセスできる URL。 **例:** **quay-server.example.com**

フィールド	型	説明
SSL_CIPHERS	Array of string	指定した場合、nginx で定義された SSL 暗号のリストが有効または無効になります。 **例:** [**ECDHE-RSA-AES128-GCM-SHA256**, **ECDHE-ECDSA-AES128-GCM-SHA256**, **ECDHE-RSA-AES256-GCM-SHA384**, **ECDHE-ECDSA-AES256-GCM-SHA384**, **DHE-RSA-AES128-GCM-SHA256**, **DHE-DSS-AES128-GCM-SHA256**, **kEDH+AESGCM**, **ECDHE-RSA-AES128-SHA256**, **ECDHE-ECDSA-AES128-SHA256**, **ECDHE-RSA-AES128-SHA**, **ECDHE-ECDSA-AES128-SHA**, **ECDHE-RSA-AES256-SHA384**, **ECDHE-ECDSA-AES256-SHA384**, **ECDHE-RSA-AES256-SHA**, **ECDHE-ECDSA-AES256-SHA**, **DHE-RSA-AES128-SHA256**, **DHE-RSA-AES128-SHA**, **DHE-DSS-AES128-SHA256**, **DHE-RSA-AES256-SHA256**, **DHE-DSS-AES256-SHA**, **DHE-DSS-AES256-SHA**, **AES128-GCM-SHA256**, **AES256-GCM-SHA384**, **AES128-SHA256**, **AES256-SHA256**, **AES128-SHA**, **AES256-SHA**, **AES**, **!3DES''**, **!aNULL**, **!eNULL**, **!EXPORT**, **DES**, **!RC4**, **MD5**, **!PSK**, **!aECDH**, **!EDH-DSS-DES-CBC3-SHA**, **!EDH-RSA-DES-CBC3-SHA**, **!KRB5-DES-CBC3-SHA**]
SSL_PROTOCOLS	Array of string	指定されている場合、nginx は、リストで定義される SSL プロトコルのリストを有効にするように設定されます。リストから SSL プロトコルを削除すると、Red Hat Quay の起動時にそのプロトコルが無効になります。 **例:** **['TLSv1','TLSv1.1','TLSv1.2', `TLSv1.3**]`

フィールド	型	説明
SESSION_COOKIE_SECURE	Boolean	セッションクッキーに **secure** プロパティーを設定するべきであるかどうか。 **デフォルト:** False **推奨:** SSL を使用するインストールの場合はすべて、True に n 設定します。

3.15.1. SSL の設定

1. 証明書ファイルとプライマリーキーファイルを設定ディレクトリーにコピーして、それぞれ **ssl.cert** と **ssl.key** という名前が付けられていることを確認します。

```
$ cp ~/ssl.cert $QUAY/config
$ cp ~/ssl.key $QUAY/config
$ cd $QUAY/config
```

2. **config.yaml** ファイルを編集し、Quay で TLS を処理できるように指定します。

   config.yaml

```
...
SERVER_HOSTNAME: quay-server.example.com
...
PREFERRED_URL_SCHEME: https
...
```

3. **Quay** コンテナーを停止し、レジストリーを再起動します。

3.16. RED HAT QUAY コンテナーへの TLS 証明書の追加

カスタム TLS 証明書を Red Hat Quay に追加するには、Red Hat Quay の config ディレクトリーの下に **extra_ca_certs/** という名前の新しいディレクトリーを作成します。必要なサイト固有の TLS 証明書をこの新しいディレクトリーにコピーします。

3.16.1. TLS 証明書を Red Hat Quay に追加

1. コンテナーに追加される証明書を表示します。

```
$ cat storage.crt
-----BEGIN CERTIFICATE-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
[...]
-----END CERTIFICATE-----
```

2. certs ディレクトリーを作成し、そこに証明書をコピーします。

```
$ mkdir -p quay/config/extra_ca_certs
$ cp storage.crt quay/config/extra_ca_certs/
$ tree quay/config/
├── config.yaml
├── extra_ca_certs
│   ├── storage.crt
```

3. **Quay** コンテナーの **CONTAINER ID** を **podman ps** で取得します。

```
$ sudo podman ps
CONTAINER ID  IMAGE                           COMMAND               CREATED
STATUS             PORTS
5a3e82c4a75f  <registry>/<repo>/quay:v3.11.0 "/sbin/my_init"  24 hours ago  Up
18 hours  0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp, 443/tcp  grave_keller
```

4. その ID でコンテナーを再起動します。

```
$ sudo podman restart 5a3e82c4a75f
```

5. コンテナーの名前空間にコピーされた証明書を調べます。

```
$ sudo podman exec -it 5a3e82c4a75f cat /etc/ssl/certs/storage.pem
-----BEGIN CERTIFICATE-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
```

3.17. LDAP 設定フィールド

表3.15 LDAP の設定

フィールド	型	説明
AUTHENTICATION_TYPE (必須)	String	**LDAP** に設定する必要があります。
FEATURE_TEAM_SYNCING	Boolean	認証エンジン (OIC、LDAP または Keystone) のバッキンググループからのチームメンバーシップの同期を許可するかどうか。 **デフォルト:true**
FEATURE_NONSUPERUSER_TEAM_SYNCING_SETUP	Boolean	有効にすると、スーパーユーザー以外のユーザーもチームの同期を設定できます。 **デフォルト:false**
LDAP_ADMIN_DN	String	LDAP 認証の管理 DN。
LDAP_ADMIN_PASSWD	String	LDAP 認証の管理パスワード。

フィールド	型	説明
LDAP_ALLOW_INSECURE_FALLBACK	Boolean	LDAP 認証で SSL の非セキュアなフォールバックを許可するかどうか。
LDAP_BASE_DN	Array of string	LDAP 認証のベース DN。
LDAP_EMAIL_ATTR	String	LDAP 認証のメール属性。
LDAP_UID_ATTR	String	LDAP 認証の uid 属性。
LDAP_URI	String	LDAP URI。
LDAP_USER_FILTER	String	LDAP 認証のユーザーフィルター。
LDAP_USER_RDN	Array of string	LDAP 認証のユーザー RDN。
TEAM_RESYNC_STALE_TIME	String	チームでチーム同期が有効になっている場合は、そのメンバーシップを確認し、必要に応じて再同期する頻度。 **パターン:** **^0-9+(w|m|d|h|s)$** **例:** **2h** **デフォルト:** **30m**
LDAP_SUPERUSER_FILTER	String	**LDAP_USER_FILTER** 設定フィールドのサブセット。設定すると、Red Hat Quay 管理者は、Red Hat Quay が認証プロバイダーとして LDAP を使用する場合に、Lightweight Directory Access Protocol (LDAP) ユーザーをスーパーユーザーとして設定できます。 このフィールドを使用すると、管理者は Red Hat Quay 設定ファイルを更新してデプロイメントを再起動することなく、スーパーユーザーを追加または削除できます。 このフィールドでは、**AUTHENTICATION_TYPE** が **LDAP** に設定されている必要があります。

フィールド	型	説明
LDAP_RESTRICTED_USER_FILTER	String	**LDAP_USER_FILTER** 設定フィールドのサブセット。設定すると、Red Hat Quay 管理者は、Red Hat Quay が認証プロバイダーとして LDAP を使用する場合に、Lightweight Directory Access Protocol (LDAP) ユーザーを制限付きユーザーとして設定できます。 このフィールドでは、**AUTHENTICATION_TYPE** が **LDAP** に設定されている必要があります。
LDAP_TIMEOUT	整数	Lightweight Directory Access Protocol (LDAP) サーバーへの接続を確立するのに許可される最大時間を秒単位で指定します。 **デフォルト:10**
LDAP_NETWORK_TIMEOUT	整数	Red Hat Quay がネットワーク操作中に Lightweight Directory Access Protocol (LDAP) サーバーからの応答を待つ最大時間を秒単位で定義します。 **デフォルト:10**

3.17.1. LDAP 設定リファレンス

次の参照を使用して、**config.yaml** ファイルを必要な LDAP 設定で更新します。

3.17.1.1. 基本的な LDAP 設定

基本的な LDAP 設定については、以下を参照してください。

```
---
AUTHENTICATION_TYPE: LDAP 1
---
LDAP_ADMIN_DN: uid=<name>,ou=Users,o=<organization_id>,dc=
<example_domain_component>,dc=com 2
LDAP_ADMIN_PASSWD: ABC123 3
LDAP_ALLOW_INSECURE_FALLBACK: false 4
LDAP_BASE_DN: 5
    - o=<organization_id>
    - dc=<example_domain_component>
    - dc=com
LDAP_EMAIL_ATTR: mail 6
LDAP_UID_ATTR: uid 7
```

```
LDAP_URI: ldap://<example_url>.com 8
LDAP_USER_FILTER: (memberof=cn=developers,ou=Users,dc=<domain_name>,dc=com) 9
LDAP_USER_RDN: 10
  - ou=<example_organization_unit>
  - o=<organization_id>
  - dc=<example_domain_component>
  - dc=com
```

1. 必須。**LDAP** に設定する必要があります。
2. 必須。LDAP 認証の管理 DN。
3. 必須。LDAP 認証の管理パスワード。
4. 必須。LDAP 認証に SSL/TLS の安全でないフォールバックを許可するかどうか。
5. 必須。LDAP 認証のベース DN。
6. 必須。LDAP 認証のメール属性。
7. 必須。LDAP 認証の UID 属性。
8. 必須。LDAP URI。
9. 必須。LDAP 認証のユーザーフィルター。
10. 必須。LDAP 認証のユーザー RDN。

3.17.1.2. LDAP 制限付きユーザー設定

LDAP 制限ユーザー設定については、次のリファレンスを使用してください。

```
# ...
AUTHENTICATION_TYPE: LDAP
# ...
LDAP_ADMIN_DN: uid=<name>,ou=Users,o=<organization_id>,dc=
<example_domain_component>,dc=com
LDAP_ADMIN_PASSWD: ABC123
LDAP_ALLOW_INSECURE_FALLBACK: false
LDAP_BASE_DN:
  - o=<organization_id>
  - dc=<example_domain_component>
  - dc=com
LDAP_EMAIL_ATTR: mail
LDAP_UID_ATTR: uid
LDAP_URI: ldap://<example_url>.com
LDAP_USER_FILTER: (memberof=cn=developers,ou=Users,o=<example_organization_unit>,dc=
<example_domain_component>,dc=com)
LDAP_RESTRICTED_USER_FILTER: (<filterField>=<value>) 1
LDAP_USER_RDN:
  - ou=<example_organization_unit>
  - o=<organization_id>
  - dc=<example_domain_component>
  - dc=com
# ...
```

-

1 指定されたユーザーを制限されたユーザーとして設定します。

3.17.1.3. LDAP スーパーユーザー設定リファレンス

LDAP スーパーユーザー設定については、次のリファレンスを使用してください。

```
# ...
AUTHENTICATION_TYPE: LDAP
# ...
LDAP_ADMIN_DN: uid=<name>,ou=Users,o=<organization_id>,dc=
<example_domain_component>,dc=com
LDAP_ADMIN_PASSWD: ABC123
LDAP_ALLOW_INSECURE_FALLBACK: false
LDAP_BASE_DN:
    - o=<organization_id>
    - dc=<example_domain_component>
    - dc=com
LDAP_EMAIL_ATTR: mail
LDAP_UID_ATTR: uid
LDAP_URI: ldap://<example_url>.com
LDAP_USER_FILTER: (memberof=cn=developers,ou=Users,o=<example_organization_unit>,dc=
<example_domain_component>,dc=com)
LDAP_SUPERUSER_FILTER: (<filterField>=<value>) (1)
LDAP_USER_RDN:
    - ou=<example_organization_unit>
    - o=<organization_id>
    - dc=<example_domain_component>
    - dc=com
# ...
```

1 指定されたユーザーをスーパーユーザーとして設定します。

3.18. 設定フィールドのミラーリング

表3.16 ミラーリング設定

フィールド	型	説明
FEATURE_REPO_MIRROR	Boolean	リポジトリーミラーリングを有効または無効にします。 **デフォルト:false**
REPO_MIRROR_INTERVAL	Number	次にリポジトリーミラー候補をチェックするまでの秒数。 **デフォルト:**30

フィールド	型	説明
REPO_MIRROR_SERVER_HOSTNAME	String	**SERVER_HOSTNAME** は、ミラーリングの宛先に置き換えます。 **デフォルト:** None **例**: **openshift-quay-service**
REPO_MIRROR_TLS_VERIFY	Boolean	HTTPS を必要とし、ミラー時に Quay レジストリーの証明書を検証します。 **デフォルト:** **false**
REPO_MIRROR_ROLLBACK	Boolean	**true** に設定されている場合、リポジトリーはミラー試行の失敗後にロールバックします。 **デフォルト**: **false**

3.19. セキュリティースキャナー設定フィールド

表3.17 セキュリティースキャナーの設定

フィールド	型	説明
FEATURE_SECURITY_SCANNER	Boolean	セキュリティースキャナー。 **デフォルト:** **false**
FEATURE_SECURITY_NOTIFICATIONS	Boolean	セキュリティースキャナーが有効になっている場合、セキュリティー通知を有効にするか、オフにします **デフォルト値:** **false**

フィールド	型	説明
SECURITY_SCANNER_V4_REINDEX_THRESHOLD	String	このパラメーターは、最終のインデックス作成から状態が変更されたか、以前に失敗したマニフェストのインデックスをもう一度作成するまで待機する最小時間を判断するのに使用します。データは **manifestsecuritystatus** テーブルの **last_indexed datetime** から計算されます。インデックス作成実行時に必ず、失敗したマニフェストのインデックスを再度作成しないように、このパラメーターを使用します。再インデックスのデフォルト時間は 300 秒です。
SECURITY_SCANNER_V4_ENDPOINT	String	V4 セキュリティースキャナーのエンドポイント。 **パターン:** **^http(s)?://(.)+$** **例:** **http://192.168.99.101:6060**
SECURITY_SCANNER_V4_PSK	String	Clair 用に生成された共有キー (PSK)。
SECURITY_SCANNER_ENDPOINT	String	V2 セキュリティースキャナーのエンドポイント。 **パターン:** **^http(s)?://(.)+$** **例:** **http://192.168.99.100:6060**

フィールド	型	説明
SECURITY_SCANNER_INDEXING_INTERVAL	整数	このパラメーターは、セキュリティースキャナーのインデックス作成の間隔 (秒単位) を決定するために使用されます。インデックスのトリガーが発生すると、Red Hat Quay は、Clair でインデックス化する必要のあるマニフェストに対してそのデータベースをクエリーします。これには、インデックス付けされていないマニフェストや、以前にインデックス付けに失敗したマニフェストが含まれます。 **デフォルト:** 30
FEATURE_SECURITY_SCANNER_NOTIFY_ON_NEW_INDEX	Boolean	新しいプッシュの脆弱性に関する通知の送信を許可するかどうか。 **デフォルト**: **True**
SECURITY_SCANNER_V4_MANIFEST_CLEANUP	Boolean	Red Hat Quay ガベージコレクターが、他のタグまたはマニフェストによって参照されていないマニフェストを削除するかどうか。 **デフォルト***: **True**

3.19.1. Clair v4 によるインデックスの再作成

Clair v4 がマニフェストをインデックス化する場合は、結果として、決定論的なものである必要があります。たとえば、同じマニフェストが同じインデックスレポートを生成する必要があります。これは、異なるスキャナーを使用するとレポートで返される特定のマニフェストに関連して異なる情報を生成するため、スキャナーが変更されるまで True となります。そのため、Clair v4 はインデックスエンジン (**/indexer/api/v1/index_state**) の状態表現を公開し、スキャナー設定が変更されたかどうかを判別します。

Red Hat Quay は、Quay のデータベースの解析時にこれをインデックスレポートに保存し、このインデックス状態を活用します。以前にスキャンされてからこの状態が変更された場合、Red Hat Quay は定期的なインデックスプロセス時にマニフェストの再作成を試行します。

デフォルトでは、このパラメーターは 30 秒に設定されています。インデックス作成のプロセスをより頻繁に実行する場合は、時間を短縮します。たとえば、新規タグをプッシュしてから、30 秒待たずに、UI でセキュリティースキャンの結果を表示する場合などです。また、ユーザーは要求パターンを Clair に制御し、Red Hat Quay データベースで実行されるデータベース操作のパターンをより詳細に制御する必要がある場合にパラメーターを変更することもできます。

3.19.2. セキュリティースキャナーの設定の例

次の YAML は、セキュリティースキャナー機能を有効にする場合の推奨設定です。

セキュリティースキャナーの YAML 設定

```
FEATURE_SECURITY_NOTIFICATIONS: true
FEATURE_SECURITY_SCANNER: true
FEATURE_SECURITY_SCANNER_NOTIFY_ON_NEW_INDEX: true
...
SECURITY_SCANNER_INDEXING_INTERVAL: 30
SECURITY_SCANNER_V4_MANIFEST_CLEANUP: true
SECURITY_SCANNER_V4_ENDPOINT: http://quay-server.example.com:8081
SECURITY_SCANNER_V4_PSK: MTU5YzA4Y2ZkNzJoMQ==
SERVER_HOSTNAME: quay-server.example.com
...
```

3.20. HELM 設定フィールド

表3.18 Helm 設定フィールド

フィールド	型	説明
FEATURE_GENERAL_OCI_SUPPORT	Boolean	OCI アーティファクトのサポートを有効にします。 **デフォルト:** True

次の Open Container Initiative (OCI) アーティファクトタイプは、デフォルトで Red Hat Quay に組み込まれており、**FEATURE_GENERAL_OCI_SUPPORT** 設定フィールドを通じて有効になります。

フィールド	メディアタイプ	サポートされているコンテンツタイプ
Helm	**application/vnd.cncf.helm.config.v1+json**	**application/tar+gzip**、**application/vnd.cncf.helm.chart.content.v1.tar+gzip**
Cosign	**application/vnd.oci.image.config.v1+json**	**application/vnd.dev.cosign.simplesigning.v1+json**、**application/vnd.dsse.envelope.v1+json**
SPDX	**application/vnd.oci.image.config.v1+json**	**text/spdx**、**text/spdx+xml**、**text/spdx+json**
Syft	**application/vnd.oci.image.config.v1+json**	**application/vnd.syft+json**
CycloneDX	**application/vnd.oci.image.config.v1+json**	**application/vnd.cyclonedx**、**application/vnd.cyclonedx+xml**、**application/vnd.cyclonedx+json**
In-toto	**application/vnd.oci.image.config.v1+json**	**application/vnd.in-toto+json**

フィールド	メディアタイプ	サポートされているコンテンツタイプ
Unknown	**application/vnd.cncf.openpolicyagent.policy.layer.v1+rego**	**application/vnd.cncf.openpolicyagent.policy.layer.v1+rego、application/vnd.cncf.openpolicyagent.data.layer.v1+json**

3.20.1. Helm の設定

次の YAML は、Helm を有効にする場合の設定例です。

Helm YAML 設定

```
FEATURE_GENERAL_OCI_SUPPORT: true
```

3.21. OPEN CONTAINER INITIATIVE 設定フィールド

表3.19 追加の OCI アーティファクト設定フィールド

フィールド	型	説明
ALLOWED_OCI_ARTIFACT_TYPES	Object	許可される OCI アーティファクト MIME タイプと関連するレイヤータイプのセット。

3.21.1. 追加のアーティファクトタイプの設定

デフォルトでサポートされていない他の OCI アーティファクトタイプは、**ALLOWED_OCI_ARTIFACT_TYPES** 設定フィールドを使用して Red Hat Quay デプロイメントに追加できます。

追加の OCI アーティファクトタイプを追加するには、次のリファレンスを使用します。

OCI アーティファクトタイプの設定

```
FEATURE_GENERAL_OCI_SUPPORT: true
ALLOWED_OCI_ARTIFACT_TYPES:
  <oci config type 1>:
  - <oci layer type 1>
  - <oci layer type 2>

  <oci config type 2>:
  - <oci layer type 3>
  - <oci layer type 4>
```

たとえば、以下を **config.yaml** に追加して Singularity (SIF) サポートを追加できます。

OCI アーティファクトタイプの設定例

```
ALLOWED_OCI_ARTIFACT_TYPES:
```

```
application/vnd.oci.image.config.v1+json:
- application/vnd.dev.cosign.simplesigning.v1+json
application/vnd.cncf.helm.config.v1+json:
- application/tar+gzip
application/vnd.sylabs.sif.config.v1+json:
- application/vnd.sylabs.sif.layer.v1+tar
```

注記

デフォルトで設定されていない OCI アーティファクトタイプを追加する場合、Red Hat Quay 管理者は、必要に応じて cosign と Helm のサポートを手動で追加する必要もあります。

3.22. 不明なメディアタイプ

表3.20 不明なメディアタイプ設定フィールド

フィールド	型	説明
IGNORE_UNKNOWN_MEDIATYPES	Boolean	有効にすると、コンテナーレジストリープラットフォームは、サポートされているアーティファクトタイプに関する特定の制限を無視し、認識されないメディアタイプまたは未知のメディアタイプを受け入れることができます。 **デフォルト:false**

3.22.1. 不明なメディアタイプの設定

次の YAML は、不明なメディアタイプまたは認識されないメディアタイプを有効にする場合の設定例です。

不明なメディアタイプの YAML 設定

```
IGNORE_UNKNOWN_MEDIATYPES: true
```

3.23. アクションログ設定フィールド

3.23.1. アクションログストレージ設定

表3.21 アクションログストレージ設定

フィールド	型	説明
FEATURE_LOG_EXPORT	Boolean	アクションログのエクスポートを許可するかどうか。 **デフォルト:True**

フィールド	型	説明
LOGS_MODEL	String	ログデータを処理するための推奨される方法を指定します。 **値:** **database**、**transition_reads_both_writes_es**、**elasticsearch**、**splunk** のいずれか。 **デフォルト:** **database**
LOGS_MODEL_CONFIG	Object	アクションログのログモデル設定。

- LOGS_MODEL_CONFIG [オブジェクト]: アクションログ用のログモデル設定。
  - elasticsearch_config [オブジェクト]: Elasticsearch クラスターの設定。
    - access_key [文字列] :Elasticsearch のユーザー (AWS ES の場合は IAM キー)。
      - 例: **some_string**
    - host [文字列]: Elasticsearch クラスターのエンドポイント。
      - 例: **host.elasticsearch.example**
    - index_prefix [文字列]。Elasticsearch のインデックスの接頭辞。
      - 例: **logentry_**
    - index_settings [オブジェクト]: Elasticsearch のインデックス設定。
    - use_ssl [ブール値]。Elasticsearch に ssl を使用します。デフォルトは **True** に設定されます。
      - 例: **True**
    - secret_key [文字列] :Elasticsearch のパスワード (AWS ES の場合は IAM シークレット)。
      - 例: **some_secret_string**
    - aws_region [文字列]: Amazon Web サービスの地域。
      - 例: **us-east-1**
    - port [数値]: Elasticsearch クラスターのエンドポイントポート。
      - 例: **1234**
  - kinesis_stream_config [オブジェクト]: AWS Kinesis ストリームの設定。
    - aws_secret_key [文字列]: AWS の秘密鍵。
      - 例: **some_secret_key**

- **stream_name** [文字列]: アクションログの送信先となる Kinesis ストリーム。
  - **例**: **logentry-kinesis-stream**
- **aws_access_key** [文字列]: AWS アクセスキー。
  - **例**: **some_access_key**
- **retries** [数値]: 一回のリクエストに対する最大試行回数。
  - **例**: **5**
- **read_timeout** [数値]: 接続の読み込み時にタイムアウトするまでの秒数。
  - **例**: **5**
- **max_pool_connections** [数値]: コネクションプールに保持するコネクションの最大数。
  - **例**: **10**
- **aws_region** [文字列]: AWS のリージョン。
  - **例**: **us-east-1**
- **connect_timeout** [数値]: 接続を試みる際のタイムアウトまでの秒数。
  - **例**: **5**

- **producer** [文字列]: Elasticsearch にロギングする場合は、producer を記録する。
  - **enum**: kafka、elasticsearch、kinesis_stream
  - **例**: **kafka**
- **kafka_config** [オブジェクト]: Kafka クラスターの設定。
  - **topic** [文字列]: ログエントリーを公開する Kafka トピック。
    - **例**: **logentry**
  - **bootstrap_servers** [配列]: クライアントをブートストラップさせる Kafka ブローカーのリスト。
  - **max_block_seconds** [数値]: **send()** の実行中に、バッファーがいっぱいになったり、メタデータが利用できないなどの理由でブロックする最大秒数。
    - **例**: **10**
- **producer** [文字列]: **splunk**
- **splunk_config** [オブジェクト]: Splunk アクションログのログモデル設定または Splunk クラスター設定。
  - **host** [文字列]: Splunk クラスターのエンドポイント。
  - **port** [整数]: Splunk 管理クラスターのエンドポイントポート。
  - **bearer_token** [文字列]: Splunk のベアラートークン。

- **verify_ssl** [ブール値]: HTTPS 接続の TLS/SSL 検証を有効 (**True**) または無効 (**False**) にする。
- **Index_prefix** [文字列]: Splunk のインデックス接頭辞。
- **ssl_ca_path** [文字列]: SSL 検証用の認証局 (CA) を含む単一の **.pem** ファイルへの相対コンテナーパス。

3.23.2. アクションログのローテーションおよびアーカイブ設定

表3.22 アクションログのローテーションおよびアーカイブ設定

フィールド	型	説明
FEATURE_ACTION_LOG_ROTATION	Boolean	ログローテーションおよび archival を有効にすると、30 日以上経過したすべてのログをストレージに移動します。 **デフォルト:false**
ACTION_LOG_ARCHIVE_LOCATION	String	アクションログのアーカイブが有効な場合は、アーカイブされたデータを配置するストレージエンジン。 **例:s3_us_east**
ACTION_LOG_ARCHIVE_PATH	String	アクションログのアーカイブが有効な場合は、アーカイブされたデータを配置するストレージのパス。 **例:archives/actionlogs**
ACTION_LOG_ROTATION_THRESHOLD	String	ログをローテーションする間隔。 **例:30d**

3.23.3. アクションログの監査設定

表3.23 監査ログ設定フィールド

フィールド	型	説明

フィールド	型	説明
ACTION_LOG_AUDIT_LOGINS	Boolean	**True** に設定すると、UI へのログインとログアウトや、通常のユーザー、ロボットアカウント、アプリケーション固有のトークンアカウントによる Docker を使用したログインなど、詳細なイベントが追跡されます。 **デフォルト:True**

3.24. ビルドログ設定フィールド

表3.24 ビルドログ設定フィールド

フィールド	型	説明
FEATURE_READER_BUILD_LOGS	Boolean	true に設定すると、**write** アクセス権や **admin** アクセス権だけでなく、リポジトリーへの **read** アクセス権を持つユーザーもビルドログを読み取ることができます。 **デフォルト:False**
LOG_ARCHIVE_LOCATION	String	アーカイブされたビルドログを配置する、**DISTRIBUTED_STORAGE_CONFIG** で定義されたストレージの場所。 **例:s3_us_east**
LOG_ARCHIVE_PATH	String	アーカイブされたビルドログを **.JSON** 形式で配置する、設定されたストレージエンジンの下のパス。 **例:archives/buildlogs**

3.25. DOCKERFILE ビルドトリガーフィールド

表3.25 Dockerfile ビルドのサポート

フィールド	型	説明
FEATURE_BUILD_SUPPORT	Boolean	Dockerfile ビルドをサポートするかどうか **デフォルト:False**

フィールド	型	説明
SUCCESSIVE_TRIGGER_FAILURE_DISABLE_THRESHOLD	Number	**None** に設定されていない場合、ビルドトリガーが自動的に無効にされる前に、連続で失敗できる回数。 **デフォルト:100**
SUCCESSIVE_TRIGGER_INTERNAL_ERROR_DISABLE_THRESHOLD	Number	**None** に設定されていない場合、ビルドトリガーが自動的に無効にされる前に、連続で発生可能な内部エラーの回数。 **デフォルト:5**

3.25.1. GitHub ビルドトリガー

表3.26 GitHub ビルドトリガー

フィールド	型	説明
FEATURE_GITHUB_BUILD	Boolean	GitHub ビルドトリガーをサポートするかどうか。 **デフォルト:False**
GITHUB_TRIGGER_CONFIG	Object	ビルドトリガーに GitHub Enterprise を使用するための設定。
.GITHUB_ENDPOINT (必須)	String	GitHub Enterprise のエンドポイント。 **例:** https://github.com/
.API_ENDPOINT	String	使用する GitHub Enterprise API のエンドポイント。**github.com** に対してオーバーライドする必要があります。 **例:** https://api.github.com/
.CLIENT_ID (必須)	String	この Red Hat Quay インスタンスの登録済みクライアント ID。これを **GITHUB_LOGIN_CONFIG** と共有することはできません。

フィールド	型	説明
.CLIENT_SECRET (必須)	String	この Red Hat Quay インスタンスの登録されたクライアントシークレット。

3.25.2. Bitbucket ビルドトリガー

表3.27 Bitbucket ビルドトリガー

フィールド	型	説明
FEATURE_BITBUCKET_BUILD	Boolean	Bitbucket ビルドトリガーをサポートするかどうか。 **デフォルト:** False
BITBUCKET_TRIGGER_CONFIG	Object	ビルドトリガーに BitBucket を使用するための設定。
.CONSUMER_KEY (必須)	String	この Red Hat Quay インスタンスの登録済みコンシューマーキー (クライアント ID)。
.CONSUMER_SECRET (必須)	String	この Red Hat Quay インスタンスの登録済みコンシューマーシークレット (クライアントシークレット)。

3.25.3. GitLab ビルドトリガー

表3.28 GitLab ビルドトリガー

フィールド	型	説明
FEATURE_GITLAB_BUILD	Boolean	GitLab ビルドトリガーをサポートするかどうか。 **デフォルト:** False
GITLAB_TRIGGER_CONFIG	Object	ビルドトリガーに Gitlab を使用するための設定。

フィールド	型	説明
.GITLAB_ENDPOINT (必須)	String	Gitlab Enterprise が実行されているエンドポイント。
.CLIENT_ID (必須)	String	この Red Hat Quay インスタンスの登録済みクライアント ID。
.CLIENT_SECRET (必須)	String	この Red Hat Quay インスタンスの登録されたクライアントシークレット。

3.26. ビルドマネージャー設定フィールド

表3.29 ビルドマネージャー設定フィールド

フィールド	型	説明
ALLOWED_WORKER_COUNT	String	Red Hat Quay Pod ごとにインスタンス化される Build Worker の数を定義します。通常は **1** に設定します。
ORCHESTRATOR_PREFIX	String	すべての Redis キーに追加する一意の接頭辞を定義します。これは、オーケストレーターの値を他の Redis キーから分離するのに役立ちます。
REDIS_HOST	Object	Redis サービスのホスト名。
REDIS_PASSWORD	String	Redis サービスへの認証に使用するパスワード。
REDIS_SSL	Boolean	Redis の接続に SSL/TLS を使用するかどうかを定義します。
REDIS_SKIP_KEYSPACE_EVENT_SETUP	Boolean	デフォルトでは、Red Hat Quay はランタイム時のキーイベントに必要なキースペースイベントを設定しません。これを行うには、**REDIS_SKIP_KEYSPACE_EVENT_SETUP** を **false** に設定します。
EXECUTOR	String	このタイプのエグゼキュータの定義を開始します。有効な値は **kubernetes** および **ec2** です。

フィールド	型	説明
BUILDER_NAMESPACE	String	Red Hat Quay のビルドが行われる Kubernetes 名前空間。
K8S_API_SERVER	Object	ビルドが行われる OpenShift Container Platform クラスターの API サーバーのホスト名。
K8S_API_TLS_CA	Object	API 呼び出しの実行時に **Quay** アプリケーションが信頼するビルドクラスターの CA 証明書の **Quay** コンテナーのファイルパス。
KUBERNETES_DISTRIBUTION	String	使用している Kubernetes の種類を示します。有効な値は **openshift** および **k8s** です。
CONTAINER_*	Object	各 **build** Pod のリソース要求および制限を定義します。
NODE_SELECTOR_*	Object	**build** Pod がスケジューリングされるノードセレクターラベル名と値のペアを定義します。
CONTAINER_RUNTIME	Object	ビルダーが **docker** と **podman** のどちらを実行するかを指定します。Red Hat の **quay-builder** イメージを使用しているお客様は、これを **podman** に設定してください。
SERVICE_ACCOUNT_NAME/SERVICE_ACCOUNT_TOKEN	Object	**build** Pod で使用されるサービスアカウント名またはトークンを定義します。
QUAY_USERNAME/QUAY_PASSWORD	Object	**WORKER_IMAGE** フィールドで指定された Red Hat Quay ビルドワーカーイメージをプルするために必要なレジストリー認証情報を定義します。お客様は、registry.redhat.io に対して https://access.redhat.com/RegistryAuthentication の記事のレジストリーサービスアカウントの作成セクションで定義されている Red Hat Service Account の認証情報を提供する必要があります。

フィールド	型	説明
WORKER_IMAGE	Object	Red Hat Quay ビルダーイメージのイメージ参照 (registry.redhat.io/quay/quay-builder)。
WORKER_TAG	Object	希望するビルダーイメージのタグ。最新バージョンは 3.11 です。
BUILDER_VM_CONTAINER_IMAGE	Object	各 Red Hat Quay ビルドの実行に必要な内部仮想マシンを保持するコンテナーイメージの完全な参照 (**registry.redhat.io/quay/quay-builder-qemu-rhcos:3.11**)。
SETUP_TIME	String	ビルドがまだビルドマネージャーに登録されていない場合に、タイムアウトする秒数を指定します。デフォルトは **500** 秒です。タイムアウトしたビルドは、3 回再起動が試みられます。3 回試してもビルドが登録されない場合は、失敗とみなされます。

フィールド	型	説明
MINIMUM_RETRY_THRESHOLD	String	この設定は、複数のエグゼキューターで使用されます。別のエグゼキューターを選択するまでに、ビルドの開始を何回再試行するかを示します。**0** に設定すると、ビルドジョブの試行回数に制限がなくなります。この値を意図的に小さく (3 以下) しておくことで、インフラストラクチャーに障害が発生した際に迅速にフェイルオーバーを行うことができます。この設定には値を指定する必要があります。たとえば、**Kubernetes** を第 1 のエグゼキューター、**EC2** を第 2 のエグゼキューターとして設定します。ジョブ実行の最後の試行を常に Kubernetes ではなく EC2 で実行する場合は、Kubernetes のエグゼキューターの **MINIMUM_RETRY_THRESHOLD** を **1** に、EC2 の **MINIMUM_RETRY_THRESHOLD** を **0** に設定します (設定していない場合はデフォルトで **0** になります)。この場合、Kubernetes の **MINIMUM_RETRY_THRESHOLD retries_remaining(1)** は **False** と評価され、設定された 2 番目のエクゼキュータにフォールバックされます。
SSH_AUTHORIZED_KEYS	Object	**ignition** 設定でブートストラップする SSH 鍵のリスト。これにより、他の鍵を使用して EC2 インスタンスや QEMU 仮想マシン (VM) に SSH 接続できます。

3.27. OAUTH 設定フィールド

表3.30 OAuth フィールド

フィールド	型	説明
DIRECT_OAUTH_CLIENTID_WHITELIST	Array of string	ユーザーの承認なしに直接 OAuth 承認を実行できる **Red Hat Quay 管理** アプリケーションのクライアント ID のリスト。

3.27.1. GitHub OAuth 設定フィールド

表3.31 GitHub OAuth フィールド

フィールド	型	説明
FEATURE_GITHUB_LOGIN	Boolean	GitHub ログインがサポートされるかどうか。 **デフォルト: **False**
GITHUB_LOGIN_CONFIG	Object	外部ログインプロバイダーとして GitHub (Enterprise) を使用するための設定。
.ALLOWED_ORGANIZATIONS	Array of string	ORG_RESTRICT オプションを使用するためにホワイトリスト化された GitHub (Enterprise) 組織の名前。
.API_ENDPOINT	String	使用する GitHub (Enterprise) API のエンドポイント。github.com に対してオーバーライドする必要があります。 **例: https://api.github.com/**
.CLIENT_ID (必須)	String	この Red Hat Quay インスタンスの登録されたクライアント ID。GITHUB_TRIGGER_CONFIG とは共有できません。**GITHUB_TRIGGER_CONFIG**. **例: 0e8dbe15c4c7630b6780**
.CLIENT_SECRET (必須)	String	Red Hat Quay インスタンスの登録クライアントシークレット。 **例: e4a58ddd3d7408b7aec109e85564a0d153d3e846**
.GITHUB_ENDPOINT (必須)	String	GitHub (Enterprise) のエンドポイント。 **例: https://github.com/**
.ORG_RESTRICT	Boolean	true の場合は、組織のホワイトリスト内のユーザーのみが、このプロバイダーを使用してログインできます。

3.27.2. Google OAuth 設定フィールド

表3.32 Google OAuth フィールド

フィールド	型	説明
FEATURE_GOOGLE_LOGIN	Boolean	Google ログインがサポートされるかどうか。 **デフォルト: **False**
GOOGLE_LOGIN_CONFIG	Object	外部認証に Google を使用するための設定。
.CLIENT_ID (必須)	String	この Red Hat Quay インスタンスの登録されたクライアント ID。 **例: 0e8dbe15c4c7630b6780**
.CLIENT_SECRET (必須)	String	Red Hat Quay インスタンスの登録クライアントシークレット。 **例: e4a58ddd3d7408b7aec109e85564a0d153d3e846**

3.28. OIDC 設定フィールド

表3.33 OIDC フィールド

フィールド	型	説明
<文字列>_LOGIN_CONFIG (必須)	String	OIDC 設定を保持する親キー。通常は、OIDC プロバイダーの名前 (**AZURE_LOGIN_CONFIG** など) ですが、任意の文字列も受け入れられます。
.CLIENT_ID (必須)	String	この Red Hat Quay インスタンスの登録されたクライアント ID。 **例: 0e8dbe15c4c7630b6780**
.CLIENT_SECRET (必須)	String	Red Hat Quay インスタンスの登録クライアントシークレット。 **例: e4a58ddd3d7408b7aec109e85564a0d153d3e846**
.DEBUGLOG	Boolean	デバッグを有効にするかどうか。

.LOGIN_BINDING_FIELD	String	内部承認が LDAP に設定されている場合に使用されます。Red Hat Quay はこのパラメーターを読み取り、このユーザー名でユーザーの LDAP ツリーで検索を試みます。存在する場合は、その LDAP アカウントへのリンクが自動的に作成されます。
.LOGIN_SCOPES	Object	Red Hat Quay が OIDC プロバイダーとの通信に使用するスコープを追加します。
.OIDC_ENDPOINT_CUSTOM_PARAMS	String	OIDC エンドポイントでのカスタムクエリーパラメーターのサポートを追加しました。エンドポイント **authorization_endpoint**、**token_endpoint**、および **user_endpoint** がサポートされています。
.OIDC_ISSUER	String	ユーザーが検証する発行者を定義できます。たとえば、JWT トークンは、トークンを発行したユーザーを定義する **iss** と呼ばれるパラメーターをコンテナー化します。デフォルトでは、これはすべての OIDC プロバイダーによって公開される **.well-know/openid/configuration** エンドポイントから読み込まれます。この検証に失敗すると、ログインはありません。
.OIDC_SERVER (必須)	String	認証に使用される OIDC サーバーのアドレス。 **例**: **https://sts.windows.net/6c878…/**
.PREFERRED_USERNAME_CLAIM_NAME	String	優先ユーザー名をトークンのパラメーターに設定します。
.SERVICE_ICON	String	ログイン画面のアイコンを変更します。
.SERVICE_NAME (必須)	String	認証されているサービスの名前。 **例**: **Microsoft Entra ID**

.VERIFIED_EMAIL_CLAIM_NAME	String	ユーザーの電子メールアドレスを確認するために使用されるクレームの名前。
.推奨グループクレーム名	String	ユーザーのグループメンバーシップに関する情報を保持する OIDC トークンペイロード内のキー名。

3.28.1. OIDC 設定

以下の例は、OIDC 設定のサンプルを示しています。

OIDC 設定の例

```
AZURE_LOGIN_CONFIG:
    CLIENT_ID: <client_id>
    CLIENT_SECRET: <client_secret>
    OIDC_SERVER: <oidc_server_address_>
    DEBUGGING: true
    SERVICE_NAME: Microsoft Entra ID
    VERIFIED_EMAIL_CLAIM_NAME: <verified_email>
    OIDC_ENDPOINT_CUSTOM_PARAMS":
            "authorization_endpoint":
               "some": "param",
```

3.29. ネストされたリポジトリー設定フィールド

ネストされたリポジトリーパス名のサポートが **FEATURE_EXTENDED_REPOSITORY_NAMES** プロパティーに追加されました。このオプションの設定は、デフォルトで config.yaml に追加されます。有効にすると、リポジトリー名で / を使用できます。

表3.34 OCI およびネストされたリポジトリー設定フィールド

フィールド	型	説明
FEATURE_EXTENDED_REPOSITORY_NAMES	Boolean	ネストされたリポジトリーのサポートを有効にします。 **デフォルト:** True

OCI とネストされたリポジトリーの設定例

```
FEATURE_EXTENDED_REPOSITORY_NAMES: true
```

3.30. QUAYINTEGRATION 設定フィールド

QuayIntegration カスタムリソースでは、次の設定フィールドを使用できます。

名前	説明	スキーマ
allowlistNamespaces (オプション)	含める namespace のリスト。	アレイ
clusterID (必須)	このクラスターに関連付けられている ID。	String
credentialsSecret.key (必須)	Quay レジストリーと通信するための認証情報を含む秘密。	Object
denylistNamespaces (オプション)	除外する namespace のリスト。	アレイ
insecureRegistry (オプション)	Quay レジストリーへの TLS 検証をスキップするかどうか	Boolean
quayHostname (必須)	Quay レジストリーのホスト名。	String
scheduledImageStreamImport (オプション)	イメージストリームのインポートを有効にするかどうか。	Boolean

3.31. メール設定フィールド

表3.35 メール設定フィールド

フィールド	型	説明
FEATURE_MAILING	Boolean	メールが有効かどうか **デフォルト: False**
MAIL_DEFAULT_SENDER	String	指定されている場合、Red Hat Quay がメールを送信する際の **from** として使用されるメールアドレス。何も指定されていない場合は、**support@quay.io** にデフォルト設定されます。 **例: support@example.com**
MAIL_PASSWORD	String	メールの送信時に使用する SMTP パスワード。

フィールド	型	説明
MAIL_PORT	Number	使用する SMTP ポート。指定されていない場合は、デフォルトの 587 になります。
MAIL_SERVER	String	メールの送信に使用する SMTP サーバー。FEATURE_MAILING が true に設定されている場合にのみ必要です。 **例: smtp.example.com**
MAIL_USERNAME	String	メールの送信時に使用する SMTP ユーザー名。
MAIL_USE_TLS	Boolean	指定されている場合は、電子メールの送信に TLS を使用するかどうか。 **デフォルト: True**

3.32. ユーザー設定フィールド

表3.36 ユーザー設定フィールド

フィールド	型	説明
FEATURE_SUPER_USERS	Boolean	スーパーユーザーがサポートされるかどうか。 **デフォルト: true**
FEATURE_USER_CREATION	Boolean	ユーザーを作成できるようにするかどうか (スーパーユーザー以外が)。 **デフォルト: true**
FEATURE_USER_LAST_ACCESSED	Boolean	ユーザーが最後にアクセスした時間を記録するかどうか。 **デフォルト: true**

フィールド	型	説明
FEATURE_USER_LOG_ACCESS	Boolean	true に設定すると、ユーザーは namespace の監査ログにアクセスできます **デフォルト:false**
FEATURE_USER_METADATA	Boolean	ユーザーメタデータを収集してサポートするかどうか。 **デフォルト:false**
FEATURE_USERNAME_CONFIRMATION	Boolean	true に設定すると、OpenID Connect (OIDC) または LDAP などのデータベース以外の認証プロバイダーでログインする場合に、初期ユーザー名を確認および変更できます。 **デフォルト:true**
FEATURE_USER_RENAME	Boolean	true に設定すると、ユーザーは独自の namespace の名前を変更できます。 **デフォルト:false**
FEATURE_INVITE_ONLY_USER_CREATION	Boolean	作成するユーザーは別のユーザーから招待を受ける必要があります。 **デフォルト:false**
FRESH_LOGIN_TIMEOUT	String	新規ログイン時にユーザーがパスワードの再入力を要求されるまでの時間。 **例**:**5m**
USERFILES_LOCATION	String	ユーザーがアップロードしたファイルを配置するストレージエンジンの ID。 **例**:**s3_us_east**
USERFILES_PATH	String	ユーザーがアップロードしたファイルを配置するストレージの下のパス。 **例**:**userfiles**

フィールド	型	説明
USER_RECOVERY_TOKEN_LIFETIME	String	ユーザーアカウントを復元するためのトークンが有効な期間。 **パターン**: **^[0-9]+(w|m|d|h|s)$** **デフォルト**: **30m**
FEATURE_SUPERUSERS_FULL_ACCESS	Boolean	スーパーユーザーが所有していない名前空間、または明示的なアクセス許可を持っていない名前空間内の他のリポジトリーからコンテンツを読み取り、書き込み、削除する機能をスーパーユーザーに付与します。 **デフォルト**: **False**
FEATURE_SUPERUSERS_ORG_CREATION_ONLY	Boolean	スーパーユーザーのみに組織の作成を許可するかどうか。 **デフォルト**: **False**
FEATURE_RESTRICTED_USERS	Boolean	**RESTRICTED_USERS_WHITELIST** を設定すると、制限されたユーザーは自分の名前空間で組織やコンテンツを作成できなくなります。通常のアクセス許可は、組織のメンバーシップに適用されます。たとえば、制限付きのユーザーは、所属するチームをもとに、組織内の通常のパーミッションは割り当てられたままです。 **デフォルト**: **False**
RESTRICTED_USERS_WHITELIST	String	**FEATURE_RESTRICTED_USERS: true** を設定すると、特定のユーザーが **FEATURE_RESTRICTED_USERS** 設定から除外されます。
GLOBAL_READONLY_SUPER_USERS	String	設定すると、公開リポジトリーかどうかに関係なく、このリストのユーザーにすべてのリポジトリーへの読み取りアクセスが許可されます。

3.32.1. ユーザー設定フィールドのリファレンス

次の参照を使用して、目的の設定フィールドで **config.yaml** ファイルを更新します。

3.32.1.1. FEATURE_SUPERUSERS_FULL_ACCESS 設定リファレンス

```
---
SUPER_USERS:
- quayadmin
FEATURE_SUPERUSERS_FULL_ACCESS: True
---
```

3.32.1.2. GLOBAL_READONLY_SUPER_USERS 設定リファレンス

```
---
GLOBAL_READONLY_SUPER_USERS:
    - user1
---
```

3.32.1.3. FEATURE_RESTRICTED_USERS 設定リファレンス

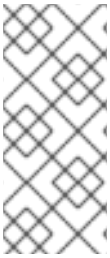
```
---
AUTHENTICATION_TYPE: Database
---
---
FEATURE_RESTRICTED_USERS: true
---
```

3.32.1.4. RESTRICTED_USERS_WHITELIST 設定リファレンス

前提条件

- **FEATURE_RESTRICTED_USERS** は **config.yaml** ファイルで **true** に設定されています。

```
---
AUTHENTICATION_TYPE: Database
---
---
FEATURE_RESTRICTED_USERS: true
RESTRICTED_USERS_WHITELIST:
    - user1
---
```



注記

このフィールドが設定されていると、ホワイトリストに登録されたユーザーは、**FEATURE_RESTRICTED_USERS** が **true** に設定されていても、組織を作成したり、リポジトリーからコンテンツを読み書きしたりできます。**user2**、**user3**、**user4** などの他のユーザーは、組織の作成、コンテンツの読み取りまたは書き込みが制限されています。

3.33. RECAPTCHA 設定フィールド

表3.37 reCAPTCHA 設定フィールド

フィールド	型	説明
FEATURE_RECAPTCHA	Boolean	ユーザーログインおよびリカバリーに Recaptcha が必要かどうか **デフォルト:** False
RECAPTCHA_SECRET_KEY	String	recaptcha が有効になっている場合は、Recaptcha サービスの秘密鍵
RECAPTCHA_SITE_KEY	String	recaptcha が有効になっている場合は、Recaptcha サービスのサイトキー

3.34. ACI 設定フィールド

表3.38 ACI 設定フィールド

フィールド	型	説明
FEATURE_ACI_CONVERSION	Boolean	ACI への変換を有効にするかどうか **デフォルト:** False
GPG2_PRIVATE_KEY_FILENAME	String	ACI の復号化に使用される秘密鍵のファイル名
GPG2_PRIVATE_KEY_NAME	String	ACI に署名するために使用されるプライベートキーの名前
GPG2_PUBLIC_KEY_FILENAME	String	ACI の暗号化に使用する公開鍵のファイル名

3.35. JWT 設定フィールド

表3.39 JWT 設定フィールド

フィールド	型	説明
JWT_AUTH_ISSUER	String	JWT ユーザーのエンドポイント **パターン**: **^http(s)?://(.)+$** **例**: **http://192.168.99.101:6060**

フィールド	型	説明
JWT_GETUSER_ENDPOINT	String	JWT ユーザーのエンドポイント **パターン**: **^http(s)?://(.)+$** **例**: **http://192.168.99.101:6060**
JWT_QUERY_ENDPOINT	String	JWT クエリーのエンドポイント **パターン**: **^http(s)?://(.)+$** **例**: **http://192.168.99.101:6060**
JWT_VERIFY_ENDPOINT	String	JWT 検証のエンドポイント **パターン**: **^http(s)?://(.)+$** **例**: **http://192.168.99.101:6060**

3.36. アプリケーショントークン設定フィールド

表3.40 アプリケーショントークン設定フィールド

フィールド	型	説明
FEATURE_APP_SPECIFIC_TOKENS	Boolean	有効な場合、ユーザーは Docker CLI で使用するトークンを作成できます。 **デフォルト**: True
APP_SPECIFIC_TOKEN_EXPIRATION	String	外部アプリトークンの有効期限。 **デフォルト**: なし **パターン**: **^[0-9]+(w|m|d|h|s)$**
EXPIRED_APP_SPECIFIC_TOKEN_GC	String	期限切れとなった外部アプリケーションがガベージコレクションが行われるまでに留まる期間。 **デフォルト**: 1d

3.37. その他の設定フィールド

表3.41 その他の設定フィールド

フィールド	型	説明

フィールド	型	説明
ALLOW_PULLS_WITHOUT_STRICT_LOGGING	String	true に指定すると、プルの監査ログのエントリーに書き込みできない場合でも、プルは成功します。これは、データベースが読み取り専用の状態にフォールバックし、その間プルを続行する必要がある場合に便利です。 **デフォルト:** False
AVATAR_KIND	String	表示する avatar のタイプ。インライン (local) または Gravatar (gravatar)。 **値:** local、gravatar
BROWSER_API_CALLS_XHR_ONLY	Boolean	有効になっていると、ブラウザーから XHR による呼び出しとしてマークが付けられた API のみが許可されます。 **デフォルト:** True
DEFAULT_NAMESPACE_MAXIMUM_BUILD_COUNT	Number	namespace でキューに入れることができるデフォルトの最大ビルド数です。 **デフォルト:** None
ENABLE_HEALTH_DEBUG_SECRET	String	指定していると、スーパーユーザーとして認証されていない場合に詳細なデバッグ情報を表示するために正常性エンドポイントに指定できるシークレット。
EXTERNAL_TLS_TERMINATION	Boolean	TLS がサポートされていても、Quay の前の層で終了する場合は **true** に設定します。独自の SSL 証明書を使用して Quay を実行しており、TLS トラフィックを直接受信している場合は、**false** に設定します。
FRESH_LOGIN_TIMEOUT	String	新規ログイン時にユーザーがパスワードの再入力を要求されるまでの時間。 **例: 5m**

フィールド	型	説明
HEALTH_CHECKER	String	設定済みのヘルスチェック。 **例**: **('RDSAwareHealthCheck', {'access_key': 'foo', 'secret_key': 'bar'})**
PROMETHEUS_NAMESPACE	String	公開されているすべての Prometheus メトリクスに適用される接頭辞。 **デフォルト**: **quay**
PUBLIC_NAMESPACES	Array of string	namespace がパブリック namespace リストに定義されている場合に、それはユーザーが namespace のメンバーであるかどうかに関係なく、**すべての** ユーザーのリポジトリーリストページに表示されます。一般的には、企業のお客様がよく知られた名前空間のセットを設定する際に使用されます。
REGISTRY_STATE	String	レジストリーの状態 **値**: **normal** または **read-only**
SEARCH_MAX_RESULT_PAGE_COUNT	Number	ユーザーが検索で表示できる最大ページ数。 **デフォルト**: 10
SEARCH_RESULTS_PER_PAGE	Number	検索ページでページごとに返される結果数。 **デフォルト**: 10
V2_PAGINATION_SIZE	Number	V2 レジストリー API において、1 ページあたりに返される結果の数。 **デフォルト**: 50
WEBHOOK_HOSTNAME_BLACKLIST	Array of string	検証時に、ローカルホスト以外に Webhook から禁止するホスト名のセット。

フィールド	型	説明
CREATE_PRIVATE_REPO_ON_PUSH	Boolean	プッシュで作成された新規リポジトリーがプライベート表示に設定されているかどうか。 **デフォルト:** True
CREATE_NAMESPACE_ON_PUSH	Boolean	既存の組織への新規プッシュで namespace を作成するかどうか。 **デフォルト:** False
NON_RATE_LIMITED_NAMESPACES	Array of string	**FEATURE_RATE_LIMITS** を使用してレートの制限が有効で、特定の namespace で無制限のアクセス権が必要な場合に、オーバーライドできます。
FEATURE_UI_V2	Boolean	設定すると、ユーザーはベータ UI 環境を試すことができます。 **デフォルト:** **True**
FEATURE_REQUIRE_TEAM_INVITE	Boolean	ユーザーをチームに追加するときに招待を必要とするかどうか。 **デフォルト:** True
FEATURE_REQUIRE_ENCRYPTED_BASIC_AUTH	Boolean	(暗号化されたトークンではなく) 暗号化されていないパスワードを Basic 認証に使用できるかどうか **デフォルト:** False
FEATURE_RATE_LIMITS	Boolean	API およびレジストリーエンドポイントでレート制限を有効にするかどうか。 FEATURE_RATE_LIMITS を **true** に設定すると、**nginx** は特定の API 呼び出しを 1 秒あたり 30 回に制限します。この機能が設定されていないと、API コールは 1 秒間に 300 回に制限されます (事実上無制限)。 **デフォルト:** False

フィールド	型	説明
FEATURE_FIPS	Boolean	true に設定すると、Red Hat Quay は FIPS 準拠のハッシュ関数を使用して実行されます。 **デフォルト:** False
FEATURE_AGGREGATED_LOG_COUNT_RETRIEVAL	Boolean	集計されたログ数の取得を許可するかどうか。 **デフォルト:** True
FEATURE_ANONYMOUS_ACCESS	Boolean	匿名ユーザーにパブリックリポジトリーの参照とプルを許可するかどうか。 **デフォルト:** True
FEATURE_DIRECT_LOGIN	Boolean	ユーザーが UI に直接ログインできるかどうか **デフォルト:** True
FEATURE_LIBRARY_SUPPORT	Boolean	Docker からプルおよびプッシュするときに名前空間のないリポジトリーを許可するかどうか。 **デフォルト:** True
FEATURE_PARTIAL_USER_AUTOCOMPLETE	Boolean	true に設定すると、オートコンプリートは部分的なユーザー名に適用されます。 **デフォルト:** True
FEATURE_PERMANENT_SESSIONS	Boolean	セッションが永続的かどうか。 **デフォルト:** True
FEATURE_PUBLIC_CATALOG	Boolean	true に設定すると、**_catalog** エンドポイントはパブリックリポジトリーを返します。それ以外では、プライベートリポジトリーのみが返されます。 **デフォルト:** False

3.38. レガシー設定フィールド

次のフィールドは非推奨または廃止されました。

表3.42 レガシー設定フィールド

フィールド	型	説明
FEATURE_BLACKLISTED_EMAILS	Boolean	true に設定すると、メールドメインがブラックリストに指定されている場合に新しいユーザーアカウントが作成されません。
BLACKLISTED_EMAIL_DOMAINS	Array of string	FEATURE_BLACKLISTED_EMAILS が true に設定されている場合に使用されるメールアドレスドメインの一覧。 **例**: **"example.com", "example.org"**
BLACKLIST_V2_SPEC	String	Red Hat Quay が V2 は **サポート対象外** であることを示す応答を返す Docker CLI バージョン。 **例**: **<1.8.0** **デフォルト**: **<1.6.0**
DOCUMENTATION_ROOT	String	ドキュメントリンクのルート URL。
SECURITY_SCANNER_V4_NAMESPACE_WHITELIST	String	セキュリティースキャナーを有効にする namespace。
FEATURE_RESTRICTED_V1_PUSH	Boolean	true に設定すると、V1_PUSH_WHITELIST にリストされている名前空間のみが V1 プッシュをサポートします。 **デフォルト**: **True**
V1_PUSH_WHITELIST	Array of string	FEATURE_RESTRICTED_V1_PUSH が true に設定されている場合に V1 push をサポートする namespace 名の配列。
FEATURE_HELM_OCI_SUPPORT	Boolean	Helm アーティファクトのサポートを有効にします。 **デフォルト**: **False**

3.39. ユーザーインターフェイス V2 設定フィールド

表3.43 ユーザーインターフェイス v2 設定フィールド

フィールド	型	説明
FEATURE_UI_V2	Boolean	設定すると、ユーザーはベータ UI 環境を試すことができます。 + **デフォルト:False**
FEATURE_UI_V2_REPO_SETTINGS	Boolean	**True** に設定すると、Red Hat Quay v2 UI でリポジトリー設定が有効になります。 + **デフォルト:False**

3.39.1. v2 ユーザーインターフェイス設定

FEATURE_UI_V2 を有効にすると、現在のバージョンのユーザーインターフェイスと新しいバージョンのユーザーインターフェイスを切り替えることができます。

重要

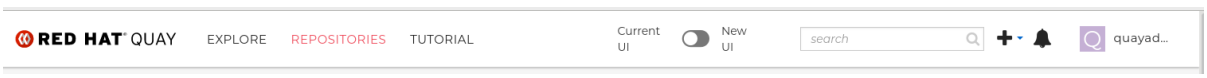
- この UI は現在ベータ版であり、変更される可能性があります。現在の状態では、ユーザーは組織、リポジトリー、およびイメージタグのみを作成、表示、および削除できます。
- 古い UI で Red Hat Quay を実行している場合にセッションがタイムアウトになると、ユーザーはポップアップウィンドウでパスワードを再度入力する必要がありました。新しい UI では、ユーザーはメインページに戻り、ユーザー名とパスワードの認証情報を入力する必要があります。これは既知の問題であり、新しい UI の今後のバージョンで修正される予定です。
- 従来の UI と新しい UI の間で、イメージマニフェストのサイズが報告される方法に違いがあります。従来の UI では、イメージマニフェストはメビバイト単位で報告されていました。新しい UI では、Red Hat Quay はメガバイト (MB) の標準定義を使用して、イメージマニフェストのサイズを報告します。

手順

1. デプロイメントの **config.yaml** ファイルで、**FEATURE_UI_V2** パラメーターを追加し、**true** に設定します。次に例を示します。

```
---
FEATURE_TEAM_SYNCING: false
FEATURE_UI_V2: true
FEATURE_USER_CREATION: true
---
```

2. Red Hat Quay デプロイメントにログインします。
3. Red Hat Quay デプロイメントのナビゲーションペインに、**Current UI** と **New UI** を切り替えるオプションが表示されます。切り替えボタンをクリックして新しい UI に設定し、次に **Use Beta Environment** をクリックします。次に例を示します。



3.40. IPV6 設定フィールド

表3.44 IPv6 設定フィールド

フィールド	型	説明
FEATURE_LISTEN_IP_VERSION	String	IPv4、IPv6、またはデュアルスタックプロトコルファミリーを有効にします。この設定フィールドは正しく設定する必要があります。そうしないと、Red Hat Quay は起動に失敗します。 **デフォルト: IPv4** **追加設定: IPv6、デュアルスタック**

3.41. ブランディング設定フィールド

表3.45 ブランディング設定フィールド

フィールド	型	説明
BRANDING	Object	Red Hat Quay UI のロゴおよび URL のカスタムブランディング
.logo (必須)	String	メインのロゴイメージ URL。 ヘッダーロゴのデフォルトは 205x30 PX です。Web UI の Red Hat Quay サインイン画面のフォームロゴは、デフォルトで 356.5x39.7 PX です。 **例: /static/img/quay-horizontal-color.svg**
.footer_img	String	UI フッターのロゴ。デフォルトは 144x34 ピクセルです。 **例: /static/img/RedHat.svg**
.footer_url	String	フッターイメージへのリンク。 **例: https://redhat.com**

3.41.1. Red Hat Quay ブランディングの設定例

ブランディング config.yaml の例

```
BRANDING:
    logo: https://www.mend.io/wp-content/media/2020/03/5-tips_small.jpg
    footer_img: https://www.mend.io/wp-content/media/2020/03/5-tips_small.jpg
    footer_url: https://opensourceworld.org/
```

3.42. セッションタイムアウト設定フィールド

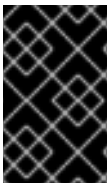
次の設定フィールドは、同じ名前の Flask API 設定フィールドに依存しています。

表3.46 セッションログアウト設定フィールド

フィールド	型	説明
PERMANENT_SESSION_LIFETIME	整数	永続セッションの有効期限を設定するために使用される **timedelta**。デフォルトは 31 日で、永続セッションは約 1 か月間存続します。 **デフォルト:2678400**

3.42.1. 設定セッションタイムアウトの設定例

次の YAML は、セッションの有効期間を有効にする場合の推奨設定です。



重要

セッションの有効期間を変更することは推奨できません。管理者は、セッションタイムアウトを設定するときに、割り当てられた時間を認識する必要があります。時間が早すぎると、ワークフローが中断する可能性があります。

セッションタイムアウトの YAML 設定

```
PERMANENT_SESSION_LIFETIME: 3000
```

第4章 環境変数

Red Hat Quay は、動的に設定する多数の環境変数をサポートします。

4.1. GEO レプリケーション

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。

表4.1 Geo レプリケーションの設定

変数	型	説明
QUAY_DISTRIBUTED_STORAGE_PREFERENCE	String	使用する優先されるストレージ (DISTRIBUTED_STORAGE_CONFIG の ID 別)

4.2. データベース接続プール

Red Hat Quay は、すべてが同じコンテナー内で実行する多くの異なるプロセスで設定されています。これらのプロセスの多くは、データベースと連動しています。

有効にすると、データベースと対話する各プロセスには、コネクションプールが含まれます。これらのプロセスごとのコネクションプールは、最大 20 個の接続を維持するように設定されています。高負荷時には、Red Hat Quay コンテナー内のすべてのプロセスの接続プールを満たすことが可能です。特定のデプロイメントおよび負荷では、Red Hat Quay が設定されたデータベースの最大接続数を超えないようにするための分析が必要になる場合があります。

時間が経つと、接続プールはアイドル接続を解放します。すべての接続をすぐに解除するには、Red Hat Quay の再起動が必要です。

データベース接続プーリングは、環境変数 **DB_CONNECTION_POOLING** を **true** または **false** に設定することで切り替えることができます。

表4.2 データベース接続プールの設定

変数	型	説明
DB_CONNECTION_POOLING	Boolean	データベース接続プールの有効化または無効化

データベース接続プーリングが有効な場合は、接続プールの最大サイズを変更することができます。これは、以下の **config.yaml** オプションを使用して実行できます。

config.yaml

```
...
DB_CONNECTION_ARGS:
  max_connections: 10
...
```

4.3. HTTP 接続回数

環境変数を使用して、HTTP の同時接続数を指定することができます。これらは、全体として、または特定のコンポーネントに対して指定できます。それぞれのデフォルトは、プロセスあたり **50 の** 並列接続です。

表4.3 HTTP 接続数の設定

変数	型	説明
WORKER_CONNECTION_COUNT	Number	同時 HTTP 接続 **デフォルト:** 50
WORKER_CONNECTION_COUNT_REGISTRY	Number	レジストリーの同時 HTTP 接続 **デフォルト:** WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_WEB	Number	Web UI の同時 HTTP 接続 **デフォルト:** WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_SECSCAN	Number	Clair の同時 HTTP 接続 **デフォルト:** WORKER_CONNECTION_COUNT

4.4. ワーカーカウント変数

表4.4 ワーカーカウント変数

変数	型	説明
WORKER_COUNT	Number	プロセス数の汎用上書き
WORKER_COUNT_REGISTRY	Number	**Quay** コンテナー内のレジストリー要求を処理するプロセス数を指定します。 **値: 8** から **64** までの整数
WORKER_COUNT_WEB	Number	コンテナー内の UI/Web リクエストを処理するプロセス数を指定します。 **値: 2** から **32** までの整数

変数	型	説明
WORKER_COUNT_SECSCAN	Number	コンテナー内のセキュリティースキャン (Clair など) の統合を処理するプロセス数を指定します。 **値:** 整数。Operator がリソースの要求と制限に 2 vCPU を指定するため、この値は **2** から **4** に設定するのが安全です。ただし、ユーザーは、保証があれば、それ以上 (たとえば **16**) 実行できます。

4.5. デバッグ変数

次のデバッグ変数は Red Hat Quay で利用できます。

表4.5 デバッグ設定変数

変数	型	説明
DEBUGLOG	Boolean	デバッグログを有効または無効にするかどうか。

変数	型	説明
USERS_DEBUG	integer**0** または **1** のいずれか。	パスワードを含むクリアテキストで LDAP 操作をデバッグするために使用されます。**DEBUGLOG=TRUE** とともに使用する必要があります。 **重要** **USERS_DEBUG=1** を設定すると、認証情報がクリアテキストで公開されます。この変数は、デバッグ後に Red Hat Quay デプロイメントから削除する必要があります。この環境変数を使用して生成されたログファイルは精査する必要があり、他のユーザーに送信する前にパスワードを削除する必要があります。注意して使用してください。

第5章 CLAIR セキュリティースキャナー

5.1. CLAIR 設定の概要

Clair は、構造化された YAML ファイルによって設定されます。各 Clair ノードは、CLI フラグまたは環境変数を使用して、実行するモードと設定ファイルへのパスを指定する必要があります。以下に例を示します。

```
$ clair -conf ./path/to/config.yaml -mode indexer
```

または、以下を実行します。

```
$ clair -conf ./path/to/config.yaml -mode matcher
```

前述のコマンドはそれぞれ、同じ設定ファイルを使用して 2 つの Clair ノードを開始します。1 つはインデックス作成機能を実行し、もう 1 つはマッチング機能を実行します。

Clair を **combo** モードで実行している場合は、設定でインデクサー、matcher、およびノーティファイアー設定ブロックを指定する必要があります。

5.1.1. プロキシー環境での Clair の使用に関する情報

Go 標準ライブラリーが尊重する環境変数は、必要に応じて指定できます。次に例を示します。

- **HTTP_PROXY**

  ```
  $ export http://<user_name>:<password>@<proxy_host>:<proxy_port>
  ```

- **HTTPS_PROXY**

  ```
  $ export https://<user_name>:<password>@<proxy_host>:<proxy_port>
  ```

- **SSL_CERT_DIR**

  ```
  $ export SSL_CERT_DIR=/<path>/<to>/<ssl>/<certificates>
  ```

Clair のアップデーター URL を使用して環境内でプロキシーサーバーを使用している場合は、Clair がスムーズにその URL にアクセスできるように、どの URL をプロキシー許可リストに追加する必要があるかを特定する必要があります。たとえば、**osv** アップデーターは、エコシステムデータダンプを取得するために **https://osv-vulnerabilities.storage.googleapis.com** にアクセスする必要があります。このような場合、URL をプロキシー許可リストに追加する必要があります。アップデーター URL の完全なリストについては、「Clair のアップデーター URL」を参照してください。

また、標準の Clair URL がプロキシー許可リストに追加されていることを確認する必要があります。

- **https://search.maven.org/solrsearch/select**
- **https://catalog.redhat.com/api/containers/**
- **https://access.redhat.com/security/data/metrics/repository-to-cpe.json**
- **https://access.redhat.com/security/data/metrics/container-name-repos-map.json**

プロキシーサーバーを設定するときは、Clair とこれらの URL 間のシームレスな通信を可能にするために必要な認証要件や特定のプロキシー設定を考慮してください。これらの考慮事項をすべて文書化して対処することで、アップデーターのトラフィックをプロキシー経由でルーティングしながら、Clair を効果的に機能させることができます。

5.1.2. Clair 設定リファレンス

次の YAML は、Clair 設定の例を示しています。

```
http_listen_addr: ""
introspection_addr: ""
log_level: ""
tls: {}
indexer:
    connstring: ""
    scanlock_retry: 0
    layer_scan_concurrency: 5
    migrations: false
    scanner: {}
    airgap: false
matcher:
    connstring: ""
    indexer_addr: ""
    migrations: false
    period: ""
    disable_updaters: false
    update_retention: 2
matchers:
    names: nil
    config: nil
updaters:
    sets: nil
    config: nil
notifier:
    connstring: ""
    migrations: false
    indexer_addr: ""
    matcher_addr: ""
    poll_interval: ""
    delivery_interval: ""
    disable_summary: false
    webhook: null
    amqp: null
    stomp: null
auth:
  psk: nil
trace:
    name: ""
    probability: null
    jaeger:
        agent:
            endpoint: ""
        collector:
            endpoint: ""
            username: null
```

```
        password: null
      service_name: ""
      tags: nil
      buffer_max: 0
metrics:
  name: ""
  prometheus:
    endpoint: null
  dogstatsd:
    url: ""
```

注記

上記の YAML ファイルには、万全を期すためにすべてのキーがリストされています。この設定ファイルをそのまま使用すると、一部のオプションがデフォルトで正常に設定されない場合があります。

5.1.3. Clair の一般的なフィールド

次の表では、Clair デプロイメントで使用できる一般的な設定フィールドについて説明します。

フィールド	Typhttp_listen_ae	説明
http_listen_addr	String	HTTP API が公開される場所を設定します。 デフォルト: **:6060**
introspection_addr	String	Clair のメトリクスと正常性エンドポイントが公開される場所を設定します。
log_level	String	ログレベルを設定します。文字列 **debug-color**、**debug**、**info**、**warn**、**error**、**fatal**、**panic** のいずれかが必要です。
tls	String	TLS/SSL および HTTP/2 の HTTP API を提供するための設定を含むマップ。
.cert	String	使用する TLS 証明書。フルチェーン証明書である必要があります。

一般的な Clair フィールドの設定例
次の例は、Clair 設定を示しています。

一般的な Clair フィールドの設定例

```
# ...
http_listen_addr: 0.0.0.0:6060
introspection_addr: 0.0.0.0:8089
log_level: info
# ...
```

5.1.4. Clair インデクサー設定フィールド

次の表では、Clair の **indexer** コンポーネントの設定フィールドについて説明します。

フィールド	型	説明
indexer	Object	Clair インデクサーノード設定を提供します。
.airgap	Boolean	インデクサーとフェッチャーのインターネットへの HTTP アクセスを無効にします。プライベート IPv4 および IPv6 アドレスが許可されます。データベース接続は影響を受けません。
.connstring	String	Postgres 接続文字列。URL または libpq 接続文字列として形式を受け入れます。
.index_report_request_concurrency	整数	レートは、インデックスレポート作成リクエストの数を制限します。これを **0** に設定すると、この値の自動サイズ調整が試みられます。負の値を設定すると、無制限になります。自動サイジングは、使用可能なコア数の倍数です。 同時実行数を超えた場合、API はステータスコード **429** を返します。
.scanlock_retry	整数	秒を表す正の整数。並行インデクサーは、マニフェストスキャンをロックして、上書きを回避します。この値は、待機中のインデクサーがロックをポーリングする頻度をチューニングします。
.layer_scan_concurrency	整数	レイヤーの同時スキャン数を制限する正の整数。インデクサーは、マニフェストのレイヤーを同時にマッチングします。この値は、インデクサーが並行してスキャンするレイヤーの数をチューニングします。

フィールド	型	説明
.migrations	Boolean	インデクサーノードがデータベースへの移行を処理するかどうか。
.scanner	String	インデクサー設定。 Scanner を使用すると、設定オプションをレイヤースキャナーに渡すことができます。スキャナーは、そのように設計されていると、構築時にこの設定を渡します。
.scanner.dist	String	特定のスキャナーの名前と値として任意の YAML を持つマップ。
.scanner.package	String	特定のスキャナーの名前と値として任意の YAML を持つマップ。
.scanner.repo	String	特定のスキャナーの名前と値として任意の YAML を持つマップ。

インデクサー設定の例
次の例は、Clair の仮のインデクサー設定を示しています。

インデクサー設定の例

```
# ...
indexer:
  connstring: host=quay-server.example.com port=5433 dbname=clair user=clairuser
password=clairpass sslmode=disable
  scanlock_retry: 10
  layer_scan_concurrency: 5
  migrations: true
# ...
```

5.1.5. Clair matcher 設定フィールド

次の表では、Clair の **matcher** コンポーネントの設定フィールドについて説明します。

注記

matchers 設定フィールドとは異なります。

フィールド	型	説明
matcher	Object	Clair matcher ノード設定を提供します。

フィールド	型	説明
.cache_age	String	レスポンスをキャッシュするように、ユーザーに通知する期間を制御します。
.connstring	String	Postgres 接続文字列。URL または libpq 接続文字列として形式を受け入れます。
.max_conn_pool	整数	データベース接続プールのサイズを制限します。 Clair では、カスタムの接続プールサイズを使用できます。この数は、同時に許可されるアクティブなデータベース接続の数を直接設定します。 このパラメーターは、将来のバージョンでは無視されます。ユーザーは、接続文字列を使用して、これを設定する必要があります。
.indexer_addr	String	matcher は indexer に接続して脆弱性レポートを作成します。このインデクサーの場所は必須です。 デフォルトは **30m** です。
.migrations	Boolean	matcher ノードがデータベースへの移行を処理するかどうか。
.period	String	新しいセキュリティーアドバイザリーの更新頻度を決定します。 デフォルトは **30m** です。
.disable_updaters	Boolean	バックグラウンド更新を実行するかどうか。 デフォルト: **False**

フィールド	型	説明
.update_retention	整数	ガベージコレクションサイクル間で保持する更新操作の数を設定します。これは、データベースサイズの制約に基づいて安全な MAX 値に設定する必要があります。 デフォルトは **10m** です。 **0** 未満の値を指定すると、ガベージコレクションが無効になります。**2** は、更新を通知と比較できるようにするための最小値です。

matcher 設定の例

matcher 設定の例

```
# ...
matcher:
  connstring: >-
    host=<DB_HOST> port=5432 dbname=<matcher> user=<DB_USER> password=D<B_PASS>
    sslmode=verify-ca sslcert=/etc/clair/ssl/cert.pem sslkey=/etc/clair/ssl/key.pem
    sslrootcert=/etc/clair/ssl/ca.pem
  indexer_addr: http://clair-v4/
  disable_updaters: false
  migrations: true
  period: 6h
  update_retention: 2
# ...
```

5.1.6. Clair matchers 設定フィールド

次の表では、Clair の **matchers** コンポーネントの設定フィールドについて説明します。

注記

matcher 設定フィールドとは異なります。

表5.1 matchers 設定フィールド

フィールド	型	説明
matchers	文字列の配列。	ツリー内 **matchers** の設定を提供します。

フィールド	型	説明
.names	String	有効な matchers について matcher ファクトリーに通知する文字列値のリスト。値を **null** に設定すると、matchers のデフォルトのリストが実行されます。次の文字列が受け入れられます。**alpine-matcher**、**aws-matcher**、**debian-matcher**、**gobin**、**java-maven**、**oracle**、**photon**、**python**、**rhel**、**rhel-container-matcher**、**Ruby**、**suse**、**ubuntu-matcher**
.config	String	特定の matcher に設定を提供します。 matchers ファクトリーコンストラクターに提供されるサブオブジェクトを含む matcher の名前をキーとするマップ。以下に例を示します。

matchers 設定の例
次の例は、**alpine**、**aws**、**debian**、**oracle** matchers のみを必要とする仮の Clair デプロイメントを示しています。

matchers 設定の例

```
# ...
matchers:
  names:
  - "alpine-matcher"
  - "aws"
  - "debian"
  - "oracle"
# ...
```

5.1.7. Clair アップデーター設定フィールド

次の表では、Clair の **updaters** コンポーネントの設定フィールドについて説明します。

表5.2 アップデーター設定フィールド

フィールド	型	説明
updaters	Object	matcher の更新マネージャーの設定を提供します。

フィールド	型	説明
.sets	String	どのアップデーターを実行するかを更新マネージャーに通知する値のリスト。 値を **null** に設定すると、アップデーターのデフォルトのセットが、**alpine**、**aws**、**clair.cvss**、**debian**、**oracle**、**photon**、**osv**、**rhel**、**rhcc suse**、**ubuntu** を実行します。 空白のままにすると、アップデーターは実行されません。
.config	String	特定のアップデーターセットに設定を提供します。 アップデーターセットのコンストラクターに提供されるサブオブジェクトを含むアップデーターセットの名前をキーとするマップ。各アップデーターのサブオブジェクトのリストについては、「詳細なアップデーター設定」を参照してください。

アップデーター設定の例
次の設定では、**rhel** セットのみが設定されます。**rhel** アップデーターに固有の **ignore_unpatched** 変数も定義されています。

アップデーター設定の例

```
# ...
updaters:
  sets:
    - rhel
  config:
    rhel:
      ignore_unpatched: false
# ...
```

5.1.8. Clair ノーティファイアー設定フィールド

Clair の一般的なノーティファイアー設定フィールドを以下に示します。

フィールド	型	説明

フィールド	型	説明
notifier	Object	Clair ノーティファイアーノード設定を提供します。
.connstring	String	Postgres 接続文字列。形式を URL または libpq 接続文字列として受け入れます。
.migrations	Boolean	ノーティファイアーノードがデータベースへの移行を処理するかどうか。
.indexer_addr	String	ノーティファイアーはインデクサーに接続して、脆弱性の影響を受けるマニフェストを作成または取得します。このインデクサーの場所は必須です。
.matcher_addr	String	ノーティファイアーは matcher に接続して、更新操作をリストし、差分を取得します。この matcher の場所は必須です。
.poll_interval	String	ノーティファイアーが matcher に更新操作をクエリーする頻度。
.delivery_interval	String	ノーティファイアーが、作成された通知または以前に失敗した通知の配信を試行する頻度。
.disable_summary	Boolean	通知をマニフェストごとに 1 つに要約するかどうかを制御します。

ノーティファイアー設定の例
次の **notifier** スニペットは、最小設定用です。

ノーティファイアー設定の例

```
# ...
notifier:
  connstring: >-
    host=DB_HOST port=5432 dbname=notifier user=DB_USER password=DB_PASS
    sslmode=verify-ca sslcert=/etc/clair/ssl/cert.pem sslkey=/etc/clair/ssl/key.pem
    sslrootcert=/etc/clair/ssl/ca.pem
  indexer_addr: http://clair-v4/
  matcher_addr: http://clair-v4/
  delivery_interval: 5s
  migrations: true
  poll_interval: 15s
  webhook:
```

```
    target: "http://webhook/"
    callback: "http://clair-notifier/notifier/api/v1/notifications"
    headers: ""
  amqp: null
  stomp: null
# ...
```

5.1.8.1. Clair Webhook 設定フィールド

次の Webhook フィールドを Clair ノーティファイアー環境で使用できます。

表5.3 Clair Webhook フィールド

.webhook	Object	Webhook 配信のノーティファイアーを設定します。
.webhook.target	String	Webhook が配信される URL。
.webhook.callback	String	通知を取得できるコールバック URL。この URL に通知 ID が追加されます。 これは通常、Clair ノーティファイアーがホスティングされている場所です。
.webhook.headers	String	ヘッダー名を値のリストに関連付けるマップ。

Webhook 設定の例

Webhook 設定の例

```
# ...
notifier:
# ...
  webhook:
    target: "http://webhook/"
    callback: "http://clair-notifier/notifier/api/v1/notifications"
# ...
```

5.1.8.2. Clair amqp 設定フィールド

次の Advanced Message Queuing Protocol (AMQP) フィールドを Clair ノーティファイアー環境で使用できます。

.amqp	Object	AMQP 配信のノーティファイアーを設定します。 [注記] ==== Clair は独自に AMQP コンポーネントを宣言しません。エクスチェンジまたはキューを使用しようとするすべての試みは、パッシブのみであり、失敗します。ブローカー管理者は、事前にエクスチェンジとキューをセットアップする必要があります。====
.amqp.direct	Boolean	**true** の場合、ノーティファイアーは設定された AMQP ブローカーに個別の通知 (コールバックではない) を配信します。
.amqp.rollup	整数	**amqp.direct** が **true** に設定されている場合、この値は直接配信で送信する通知の数をノーティファイアーに通知します。たとえば、**direct** が **true** に設定され、**amqp.rollup** が **5** に設定されている場合、ノーティファイアーは単一の JSON ペイロードで 5 つ以下の通知をブローカーに配信します。値を **0** に設定すると、実質的に **1** に設定されます。
.amqp.exchange	Object	接続先の AMQP エクスチェンジ。
.amqp.exchange.name	String	接続先のエクスチェンジの名前。
.amqp.exchange.type	String	エクスチェンジのタイプ。通常は、**direct**、**fanout**、**topic**、**headers** のいずれかです。
.amqp.exchange.durability	Boolean	設定されたキューが永続的かどうか。
.amqp.exchange.auto_delete	Boolean	設定されたキューが **auto_delete_policy** を使用するかどうか。
.amqp.routing_key	String	各通知が送信されるルーティングキーの名前。

.amqp.callback	String	**amqp.direct** が **false** に設定されている場合、この URL はブローカーに送信される通知コールバックで提供されます。この URL は、Clair の通知 API エンドポイントを指している必要があります。
.amqp.uris	String	接続先の 1 つ以上の AMQP ブローカーのリスト (優先順位順)。
.amqp.tls	Object	AMQP ブローカーへの TLS/SSL 接続を設定します。
.amqp.tls.root_ca	String	ルート CA を読み取ることができるファイルシステムパス。
.amqp.tls.cert	String	TLS/SSL 証明書を読み取ることができるファイルシステムパス。 [注意] ==== Go **crypto/x509** パッケージに記載されているように、Clair は **SSL_CERT_DIR** も許可します。====
.amqp.tls.key	String	TLS/SSL 秘密鍵を読み取ることができるファイルシステムパス。

AMQP 設定の例
次の例は、Clair の仮の AMQP 設定を示しています。

AMQP 設定の例

```
# ...
notifier:
# ...
  amqp:
    exchange:
        name: ""
        type: "direct"
        durable: true
        auto_delete: false
    uris: ["amqp://user:pass@host:10000/vhost"]
    direct: false
    routing_key: "notifications"
    callback: "http://clair-notifier/notifier/api/v1/notifications"
    tls:
     root_ca: "optional/path/to/rootca"
     cert: "madatory/path/to/cert"
     key: "madatory/path/to/key"
# ...
```

5.1.8.3. Clair STOMP 設定フィールド

次の Simple Text Oriented Message Protocol (STOMP) フィールドを Clair ノーティファイアー環境で使用できます。

.stomp	Object	STOMP 配信のノーティファイアーを設定します。
.stomp.direct	Boolean	**true** の場合、ノーティファイアーは個別の通知 (コールバックではない) を設定済みの STOMP ブローカーに配信します。
.stomp.rollup	整数	**stomp.direct** が **true** に設定されている場合、この値は、1 回の直接配信で送信される通知の数を制限します。たとえば、**direct** が **true** に設定され、**rollup** が **5** に設定されている場合、ノーティファイアーは単一の JSON ペイロードで 5 つ以下の通知をブローカーに配信します。値を **0** に設定すると、実質的に **1** に設定されます。
.stomp.callback	String	**stomp.callback** が **false** に設定されている場合は、通知コールバックで指定された URL がブローカーに送信されます。この URL は、Clair の通知 API エンドポイントを指している必要があります。
.stomp.destination	String	通知を配信する STOMP の宛先。
.stomp.uris	String	接続先の 1 つ以上の STOMP ブローカーのリスト (優先順位順)。
.stomp.tls	Object	STOMP ブローカーへの TLS/SSL 接続を設定しました。
.stomp.tls.root_ca	String	ルート CA を読み取ることができるファイルシステムパス。 [注意] ==== Go **crypto/x509** パッケージに記載されているように、Clair は **SSL_CERT_DIR** も受け入れます。====
.stomp.tls.cert	String	TLS/SSL 証明書を読み取ることができるファイルシステムパス。

.stomp	Object	STOMP 配信のノーティファイアーを設定します。
.stomp.tls.key	String	TLS/SSL 秘密鍵を読み取ることができるファイルシステムパス。
.stomp.user	String	STOMP ブローカーのログインの詳細を設定します。
.stomp.user.login	String	接続に使用する STOMP ログイン。
.stomp.user.passcode	String	接続に使用する STOMP パスコード。

STOMP 設定の例
次の例は、Clair の仮の STOMP 設定を示しています。

STOMP 設定の例

```
# ...
notifier:
# ...
  stomp:
    desitnation: "notifications"
    direct: false
    callback: "http://clair-notifier/notifier/api/v1/notifications"
    login:
      login: "username"
      passcode: "passcode"
    tls:
     root_ca: "optional/path/to/rootca"
     cert: "madatory/path/to/cert"
     key: "madatory/path/to/key"
# ...
```

5.1.9. Clair 認可設定フィールド

Clair では、次の認可設定フィールドを使用できます。

フィールド	型	説明
auth	Object	Clair の外部およびサービス内 JWT ベースの認証を定義します。複数の **auth** 機能が定義されている場合、Clair は 1 つを選択します。現在、複数の機能はサポートされていません。

フィールド	型	説明
.psk	String	事前共有キー認証を定義します。
.psk.key	String	JWT の署名と検証を行うすべての当事者間で配布される、base64 でエンコードされた共有キー。
.psk.iss	String	確認する JWT 発行者のリスト。空のリストは、JWT クレームで任意の発行者を受け入れます。

認可設定の例
次の **authorization** スニペットは、最小設定用です。

認可設定の例

```
# ...
auth:
  psk:
    key: MTU5YzA4Y2ZkNzJoMQ== 1
    iss: ["quay"]
# ...
```

5.1.10. Clair トレース設定フィールド

Clair では、次のトレース設定フィールドを使用できます。

フィールド	型	説明
trace	Object	OpenTelemetry に基づいて分散トレース設定を定義します。
.name	String	トレースが属するアプリケーションの名前。
.probability	整数	トレースが発生する確率。
.jaeger	Object	Jaeger トレースの値を定義します。
.jaeger.agent	Object	Jaeger エージェントへの配信を設定するための値を定義します。
.jaeger.agent.endpoint	String	トレースを送信できる **<host>:<post>** 構文のアドレス。

フィールド	型	説明
.jaeger.collector	Object	Jaeger コレクターへの配信を設定するための値を定義します。
.jaeger.collector.endpoint	String	トレースを送信できる **<host>:<post>** 構文のアドレス。
.jaeger.collector.username	String	Jaeger ユーザー名。
.jaeger.collector.password	String	Jaeger パスワード。
.jaeger.service_name	String	Jaeger に登録されているサービス名。
.jaeger.tags	String	追加のメタデータを提供するキーと値のペア。
.jaeger.buffer_max	整数	保管および分析のために Jaeger バックエンドに送信される前にメモリーにバッファーできるスパンの最大数。

トレース設定の例
次の例は、Clair の仮のトレース設定を示しています。

トレース設定の例

```
# ...
trace:
  name: "jaeger"
  probability: 1
  jaeger:
    agent:
      endpoint: "localhost:6831"
    service_name: "clair"
# ...
```

5.1.11. Clair メトリクス設定フィールド

Clair では、次のメトリクス設定フィールドを使用できます。

フィールド	型	説明
metrics	Object	OpenTelemetry に基づいて分散トレース設定を定義します。
.name	String	使用中のメトリクスの名前。

フィールド	型	説明
.prometheus	String	Prometheus メトリクスエクスポーターの設定。
.prometheus.endpoint	String	メトリクスが提供されるパスを定義します。

メトリクス設定の例
次の例は、Clair の仮のメトリクス設定を示しています。

メトリクス設定の例

```
# ...
metrics:
  name: "prometheus"
  prometheus:
    endpoint: "/metricsz"
# ...
```