Red Hat Quay 3

Configure Red Hat Quay

Customizing Red Hat Quay using configuration options

Last Updated: 2024-08-19

Red Hat Quay 3 Configure Red Hat Quay

Customizing Red Hat Quay using configuration options

Legal Notice

Abstract

Configure Red Hat Quay

Table of Contents

CHAPTER 1. GETTING STARTED WITH RED HAT QUAY CONFIGURATION

Red Hat Quay can be deployed by an independent, standalone configuration, or by using the Red Hat Quay Operator on OpenShift Container Platform.

How you create, retrieve, update, and validate the Red Hat Quay configuration varies depending on the type of deployment you are using. However, the core configuration options are the same for either deployment type. Core configuration is primarily set through a **config.yaml** file, but can also be set by using the configuration API.

For standalone deployments of Red Hat Quay, you must supply the minimum required configuration parameters before the registry can be started. The minimum requirements to start a Red Hat Quay registry can be found in the "Retrieving the current configuration" section.

If you install Red Hat Quay on OpenShift Container Platform using the Red Hat Quay Operator, you do not need to supply configuration parameters because the Red Hat Quay Operator supplies default information to deploy the registry.

After you have deployed Red Hat Quay with the desired configuration, you should retrieve, and save, the full configuration from your deployment. The full configuration contains additional generated values that you might need when restarting or upgrading your system.

CHAPTER 2. RED HAT QUAY CONFIGURATION DISCLAIMER

With both standalone and Operator-based deployments of Red Hat Quay certain features and configuration parameters are not actively used or implemented. As a result, feature flags, such as those that enable or disable certain features, and configuration parameters that are not explicitly documented or requested for documentation by Red Hat Support, should only be modified with caution. Unused features or parameters might not be fully tested, supported, or compatible with Red Hat Quay. Modifying unused features parameters might lead to unexpected issues or disruptions with your deployment.

For information about configuring Red Hat Quay in standalone deployments, see Advanced Red Hat Quay configuration

For information about configuring Red Hat Quay Operator deployments, see Configuring Red Hat Quay on OpenShift Container Platform

2.1. CONFIGURATION UPDATES FOR RED HAT QUAY 3.12

The following sections detail new configuration fields added in Red Hat Quay 3.12.

2.1.1. Registry auto-pruning configuration fields

The following configuration fields have been added to Red Hat Quay auto-pruning feature:

Field	Type	Description
NOTIFICATION_TASK_RUN_MINIMUM_INTERVAL_MINUTES	Integer	The interval, in minutes, that defines the frequency to re-run notifications for expiring images. **Default: 300**
DEFAULT_NAMESPACE_AUTOPRUNE_POLICY	Object	The default organization-wide auto-prune policy.
.method: number_of_tags	Object	The option specifying the number of tags to keep.
.value: <integer>	Integer	When used with **method: number_of_tags**, denotes the number of tags to keep. For example, to keep two tags, specify **2**.
.method: creation_date	Object	The option specifying the duration of which to keep tags.

.value: <integer>	Integer	When used with **creation_date**, denotes how long to keep tags. Can be set to seconds (**s**), days (**d**), months (**m**), weeks (**w**), or years (**y**). Must include a valid integer. For example, to keep tags for one year, specify **1y**.
AUTO_PRUNING_DEFAULT_POLICY_POLL_PERIOD	Integer	The period in which the auto-pruner worker runs at the registry level. By default, it is set to run one time per day (one time per 24 hours). Value must be in seconds.

2.1.2. OAuth access token reassignment configuration field

The following configuration field has been added for reassigning OAuth access tokens:

Field	Type	Description
FEATURE_ASSIGN_OAUTH_TOKEN	Boolean	Allows organization administrators to assign OAuth tokens to other users.

Example OAuth access token reassignment YAML

```
# ...
FEATURE_ASSIGN_OAUTH_TOKEN: true
# ...
```

2.1.3. Vulnerability detection notification configuration field

The following configuration field has been added to notify users on detected vulnerabilities based on security level:

Field	Type	Description
NOTIFICATION_MIN_SEVERITY_ON_NEW_INDEX	String	Set minimal security level for new notifications on detected vulnerabilities. Avoids creation of large number of notifications after first index. If not defined, defaults to **High**. Available options include **Critical**, **High**, **Medium**, **Low**, **Negligible**, and **Unknown**.

Example image vulnerability notification YAML

```
NOTIFICATION_MIN_SEVERITY_ON_NEW_INDEX: High
```

2.1.4. OCI referrers API configuration field

The following configuration field allows users to list OCI referrers of a manifest under a repository by using the v2 API:

Field	Type	Description
FEATURE_REFERRERS_API	Boolean	Enables OCI 1.1's referrers API.

Example OCI referrers enablement YAML

```
# ...
FEATURE_REFERRERS_API: true
# ...
```

2.1.5. Disable strict logging configuration field

The following configuration field has been added to address when external systems like Splunk or ElasticSearch are configured as audit log destinations but are intermittently unavailable. When set to **True**, the logging event is logged to the stdout instead.

Field	Type	Description
ALLOW_WITHOUT_STRICT_LOGGING	Boolean	When set to **True**, allows you to use any registry action when you are unable to write to the audit log.

Example strict logging YAML

```
# ...
ALLOW_WITHOUT_STRICT_LOGGING: True
# ...
```

2.1.6. Notification interval configuration field

The following configuration field has been added to enhance Red Hat Quay notifications:

Field	Type	Description

NOTIFICATION_TASK_RUN_MINIMUM_INTERVAL_MINUTES	Integer	The interval, in minutes, that defines the frequency to re-run notifications for expiring images. By default, this field is set to notify Red Hat Quay users of events happening every 5 hours.

Example notification re-run YAML

```
# ...
NOTIFICATION_TASK_RUN_MINIMUM_INTERVAL_MINUTES: 10
# ...
```

2.1.7. Clair indexing layer size configuration field

The following configuration field has been added for the Clair security scanner, which allows Red Hat Quay administrators to set a maximum layer size allowed for indexing.

Field	Type	Description
SECURITY_SCANNER_V4_INDEX_MAX_LAYER_SIZE	String	The maximum layer size allowed for indexing. If the layer size exceeds the configured size, the Red Hat Quay UI returns the following message: **The manifest for this tag has layer(s) that are too large to index by the Quay Security Scanner**. Example: **8G**

2.2. EDITING THE CONFIGURATION FILE

To deploy a standalone instance of Red Hat Quay, you must provide the minimal configuration information. The requirements for a minimal configuration can be found in "Red Hat Quay minimal configuration."

After supplying the required fields, you can validate your configuration. If there are any issues, they will be highlighted.

For changes to take effect, the registry must be restarted.

2.3. LOCATION OF CONFIGURATION FILE IN A STANDALONE DEPLOYMENT

For standalone deployments of Red Hat Quay, the **config.yaml** file must be specified when starting the Red Hat Quay registry. This file is located in the configuration volume. For example, the configuration file is located at **$QUAY/config/config.yaml** when deploying Red Hat Quay by the following command:

```
$ 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:v3.12.1
```

2.4. MINIMAL CONFIGURATION

The following configuration options are required for a standalone deployment of Red Hat Quay:

- Server hostname
- HTTP or HTTPS
- Authentication type, for example, Database or Lightweight Directory Access Protocol (LDAP)
- Secret keys for encrypting data
- Storage for images
- Database for metadata
- Redis for build logs and user events
- Tag expiration options

2.4.1. Sample minimal configuration file

The following example shows a sample minimal configuration file that uses local storage for images:

```
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. Local storage

Using local storage for images is only recommended when deploying a registry for *proof of concept* purposes.

When configuring local storage, storage is specified on the command line when starting the registry.

The following command maps a local directory, **$QUAY/storage** to the **datastorage** path in the container:

```
$ 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:v3.12.1
```

2.4.3. Cloud storage

Storage configuration is detailed in the Image storage section. For some users, it might be useful to compare the difference between Google Cloud Platform and local storage configurations. For example, the following YAML presents a Google Cloud Platform storage configuration:

$QUAY/config/config.yaml

```
DISTRIBUTED_STORAGE_CONFIG:
    default:
        - GoogleCloudStorage
        - access_key: GOOGQIMFB3ABCDEFGHIJKLMN
          bucket_name: quay_bucket
          secret_key: FhDAYe2HeuAKfvZCAGyOioNaaRABCDEFGHIJKLMN
          storage_path: /datastorage/registry
          boto_timeout: 120 (1)
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - default
```

(1) Optional. The time, in seconds, until a timeout exception is thrown when attempting to read from a connection. The default is **60** seconds. Also encompasses the time, in seconds, until a timeout exception is thrown when attempting to make a connection. The default is **60** seconds.

When starting the registry using cloud storage, no configuration is required on the command line. For example:

```
$ 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:v3.12.1
```

CHAPTER 3. CONFIGURATION FIELDS

This section describes the both required and optional configuration fields when deploying Red Hat Quay.

3.1. REQUIRED CONFIGURATION FIELDS

The fields required to configure Red Hat Quay are covered in the following sections:

- General required fields
- Storage for images
- Database for metadata
- Redis for build logs and user events
- Tag expiration options

3.2. AUTOMATION OPTIONS

The following sections describe the available automation options for Red Hat Quay deployments:

- Pre-configuring Red Hat Quay for automation
- Using the API to create the first user

3.3. OPTIONAL CONFIGURATION FIELDS

Optional fields for Red Hat Quay can be found in the following sections:

- Basic configuration
- SSL
- LDAP
- Repository mirroring
- Quota management
- Security scanner
- Helm
- Action log
- Build logs
- Dockerfile build
- OAuth
- Configuring nested repositories

- Adding other OCI media types to Quay
- Mail
- User
- Recaptcha
- ACI
- JWT
- App tokens
- Miscellaneous
- User interface v2
- IPv6 configuration field
- Legacy options

3.4. GENERAL REQUIRED FIELDS

The following table describes the required configuration fields for a Red Hat Quay deployment:

Table 3.1. General required fields

Field	Type	Description
AUTHENTICATION_TYPE (Required)	String	The authentication engine to use for credential authentication. **Values:** One of **Database**, **LDAP**, **JWT**, **Keystone**, **OIDC** **Default: Database**
PREFERRED_URL_SCHEME (Required)	String	The URL scheme to use when accessing Red Hat Quay. **Values:** One of **http**, **https** **Default: http**
SERVER_HOSTNAME (Required)	String	The URL at which Red Hat Quay is accessible, without the scheme. **Example:** **quay-server.example.com**

Field	Type	Description
DATABASE_SECRET_KEY (Required)	String	Key used to encrypt sensitive fields within the database. This value should never be changed once set, otherwise all reliant fields, for example, repository mirror username and password configurations, are invalidated. This value is set automatically by the Red Hat Quay Operator for Operator-based deployments. For standalone deployments, administrators can provide their own key using Open SSL or a similar tool. Key length should not exceed 63 characters.
SECRET_KEY (Required)	String	Key used to encrypt the session cookie and the CSRF token needed for correct interpretation of the user session. The value should not be changed when set. Should be persistent across all Red Hat Quay instances. If not persistent across all instances, login failures and other errors related to session persistence might occur.
SETUP_COMPLETE (Required)	Boolean	This is an artifact left over from earlier versions of the software and currently it **must** be specified with a value of **true**.

3.5. DATABASE CONFIGURATION

This section describes the database configuration fields available for Red Hat Quay deployments.

3.5.1. Database URI

With Red Hat Quay, connection to the database is configured by using the required **DB_URI** field.

The following table describes the **DB_URI** configuration field:

Table 3.2. Database URI

Field	Type	Description
DB_URI (Required)	String	The URI for accessing the database, including any credentials. Example **DB_URI** field: **postgresql://quayuser:quaypass@quay-server.example.com:5432/quay**

3.5.2. Database connection arguments

Optional connection arguments are configured by the **DB_CONNECTION_ARGS** parameter. Some of the key-value pairs defined under **DB_CONNECTION_ARGS** are generic, while others are database specific.

The following table describes database connection arguments:

Table 3.3. Database connection arguments

Field	Type	Description
DB_CONNECTION_ARGS	Object	Optional connection arguments for the database, such as timeouts and SSL/TLS.
.autorollback	Boolean	Whether to use thread-local connections. Should always be **true**
.threadlocals	Boolean	Whether to use auto-rollback connections. Should always be **true**

3.5.2.1. PostgreSQL SSL/TLS connection arguments

With SSL/TLS, configuration depends on the database you are deploying. The following example shows a PostgreSQL SSL/TLS configuration:

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

The **sslmode** option determines whether, or with, what priority a secure SSL/TLS TCP/IP connection will be negotiated with the server. There are six modes:

Table 3.4. SSL/TLS options

Mode	Description
disable	Your configuration only tries non-SSL/TLS connections.
allow	Your configuration first tries a non-SSL/TLS connection. Upon failure, tries an SSL/TLS connection.
prefer (Default)	Your configuration first tries an SSL/TLS connection. Upon failure, tries a non-SSL/TLS connection.
require	Your configuration only tries an SSL/TLS connection. If a root CA file is present, it verifies the certificate in the same way as if verify-ca was specified.
verify-ca	Your configuration only tries an SSL/TLS connection, and verifies that the server certificate is issued by a trusted certificate authority (CA).
verify-full	Only tries an SSL/TLS connection, and verifies that the server certificate is issued by a trusted CA and that the requested server hostname matches that in the certificate.

For more information on the valid arguments for PostgreSQL, see Database Connection Control Functions.

3.5.2.2. MySQL SSL/TLS connection arguments

The following example shows a sample MySQL SSL/TLS configuration:

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

Information on the valid connection arguments for MySQL is available at Connecting to the Server Using URI-Like Strings or Key-Value Pairs.

3.6. IMAGE STORAGE

This section details the image storage features and configuration fields that are available with Red Hat Quay.

3.6.1. Image storage features

The following table describes the image storage features for Red Hat Quay:

Table 3.5. Storage config features

Field	Type	Description
FEATURE_REPO_MIRROR	Boolean	If set to true, enables repository mirroring. **Default: false**
FEATURE_PROXY_STORAGE	Boolean	Whether to proxy all direct download URLs in storage through NGINX. **Default: false**
FEATURE_STORAGE_REPLICATION	Boolean	Whether to automatically replicate between storage engines. **Default: false**

3.6.2. Image storage configuration fields

The following table describes the image storage configuration fields for Red Hat Quay:

Table 3.6. Storage config fields

Field	Type	Description
DISTRIBUTED_STORAGE_CONFIG (Required)	Object	Configuration for storage engine(s) to use in Red Hat Quay. Each key represents an unique identifier for a storage engine. The value consists of a tuple of (key, value) forming an object describing the storage engine parameters. **Default: []**
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS (Required)	Array of string	The list of storage engine(s) (by ID in **DISTRIBUTED_STORAGE_CONFIG**) whose images should be fully replicated, by default, to all other storage engines.

Field	Type	Description
DISTRIBUTED_STORAGE_PREFERENCE (Required)	Array of string	The preferred storage engine(s) (by ID in **DISTRIBUTED_STORAGE_CONFIG**) to use. A preferred engine means it is first checked for pulling and images are pushed to it. **Default: false**
MAXIMUM_LAYER_SIZE	String	Maximum allowed size of an image layer. **Pattern: ^[0-9]+(G|M)$** **Example: 100G** **Default: 20G**

3.6.3. Local storage

The following YAML shows a sample configuration using local storage:

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

3.6.4. OpenShift Container Storage/NooBaa

The following YAML shows a sample configuration using an OpenShift Container Storage/NooBaa instance:

```
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
      maximum_chunk_size_mb: 100 (1)
      server_side_assembly: true (2)
```

1. Defines the maximum chunk size, in MB, for the final copy. Has no effect if **server_side_assembly** is set to **false**.
2. Whether Red Hat Quay should try and use server side assembly and the final chunked copy instead of client assembly. Defaults to **true**.

3.6.5. Ceph/RadosGW storage

The following examples show two possible YAML configurations when using Ceph/RadosGW.

Example A: Using RadosGW with the radosGWStorage driver

```
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
      maximum_chunk_size_mb: 100 (1)
      server_side_assembly: true (2)
```

1. Defines the maximum chunk size in MB for the final copy. Has no effect if **server_side_assembly** is set to **false**.
2. Whether Red Hat Quay should try and use server side assembly and the final chunked copy instead of client assembly. Defaults to **true**.

Example B: Using RadosGW with general s3 access

```
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
      maximum_chunk_size_mb: 100 (2)
      server_side_assembly: true (3)
```

1. Used for general s3 access. Note that general s3 access is not strictly limited to Amazon Web Services (AWS) s3, and can be used with RadosGW or other storage services. For an example of general s3 access using the AWS S3 driver, see "AWS S3 storage".
2. Defines the maximum chunk size in MB for the final copy. Has no effect if **server_side_assembly** is set to **false**.

3 Whether Red Hat Quay should try and use server side assembly and the final chunked copy instead of client assembly. Defaults to **true**.

3.6.6. AWS S3 storage

The following YAML shows a sample configuration using AWS S3 storage.

```
# ...
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
      s3_region: <region> 2
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - default
# ...
```

1 The **S3Storage** storage driver should only be used for AWS S3 buckets. Note that this differs from general S3 access, where the RadosGW driver or other storage services can be used. For an example, see "Example B: Using RadosGW with general S3 access".

2 Optional. The Amazon Web Services region. Defaults to **us-east-1**.

3.6.6.1. AWS STS S3 storage

The following YAML shows an example configuration for using Amazon Web Services (AWS) Security Token Service (STS) with Red Hat Quay on OpenShift Container Platform configurations.

```
# ...
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
      s3_region: <region> 4
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - default
# ...
```

1 The unique Amazon Resource Name (ARN).

2 The generated AWS S3 user access key.

3 The generated AWS S3 user secret key.

4 Optional. The Amazon Web Services region. Defaults to **us-east-1**.

3.6.7. Google Cloud Storage

The following YAML shows a sample configuration using Google Cloud Storage:

```
DISTRIBUTED_STORAGE_CONFIG:
    googleCloudStorage:
        - GoogleCloudStorage
        - access_key: GOOGQIMFB3ABCDEFGHIJKLMN
          bucket_name: quay-bucket
          secret_key: FhDAYe2HeuAKfvZCAGyOioNaaRABCDEFGHIJKLMN
          storage_path: /datastorage/registry
          boto_timeout: 120 1
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
    - googleCloudStorage
```

1 Optional. The time, in seconds, until a timeout exception is thrown when attempting to read from a connection. The default is **60** seconds. Also encompasses the time, in seconds, until a timeout exception is thrown when attempting to make a connection. The default is **60** seconds.

3.6.8. Azure Storage

The following YAML shows a sample configuration using Azure Storage:

```
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 The **endpoint_url** parameter for Azure storage is optional and can be used with Microsoft Azure Government (MAG) endpoints. If left blank, the **endpoint_url** will connect to the normal Azure region.

As of Red Hat Quay 3.7, you must use the Primary endpoint of your MAG Blob service. Using the Secondary endpoint of your MAG Blob service will result in the following error: **AuthenticationErrorDetail:Cannot find the claimed account when trying to GetProperties for the account whusc8-secondary**.

3.6.9. Swift storage

The following YAML shows a sample configuration using Swift storage:

```
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 object storage

The following YAML shows a sample configuration using Nutanix object storage.

```
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 object storage

The following YAML shows a sample configuration using IBM Cloud object storage.

```
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 Optional. Recommended to be set to **100mb**.

3.6.12. NetApp ONTAP S3 object storage

The following YAML shows a sample configuration using NetApp ONTAP S3.

```
DISTRIBUTED_STORAGE_CONFIG:
  local_us:
  - RadosGWStorage
  - access_key: <access_key>
    bucket_name: <bucket_name>
    hostname: <host_url_address>
    is_secure: true
    port: <port>
    secret_key: <secret_key>
    storage_path: /datastorage/registry
    signature_version: v4
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS:
- local_us
DISTRIBUTED_STORAGE_PREFERENCE:
- local_us
```

3.7. REDIS CONFIGURATION FIELDS

This section details the configuration fields available for Redis deployments.

3.7.1. Build logs

The following build logs configuration fields are available for Redis deployments:

Table 3.7. Build logs configuration

Field	Type	Description
BUILDLOGS_REDIS (Required)	Object	Redis connection details for build logs caching.
.host (Required)	String	The hostname at which Redis is accessible. **Example:** **quay-server.example.com**
.port (Required)	Number	The port at which Redis is accessible. **Example:** **6379**
.password	String	The password to connect to the Redis instance. **Example:** **strongpassword**

Field	Type	Description
.ssl (Optional)	Boolean	Whether to enable TLS communication between Redis and Quay. Defaults to false.

3.7.2. User events

The following user event fields are available for Redis deployments:

Table 3.8. User events config

Field	Type	Description
USER_EVENTS_REDIS (Required)	Object	Redis connection details for user event handling.
.host (Required)	String	The hostname at which Redis is accessible. **Example:** **quay-server.example.com**
.port (Required)	Number	The port at which Redis is accessible. **Example:** **6379**
.password	String	The password to connect to the Redis instance. **Example:** **strongpassword**
.ssl	Boolean	Whether to enable TLS communication between Redis and Quay. Defaults to false.
.ssl_keyfile (Optional)	String	The name of the key database file, which houses the client certificate to be used. **Example:** **ssl_keyfile: /path/to/server/privatekey.pem**
.ssl_certfile (Optional)	String	Used for specifying the file path of the SSL certificate. **Example:** **ssl_certfile: /path/to/server/certificate.pem**

Field	Type	Description
.ssl_cert_reqs (Optional)	String	Used to specify the level of certificate validation to be performed during the SSL/TLS handshake. **Example:** **ssl_cert_reqs: CERT_REQUIRED**
.ssl_ca_certs (Optional)	String	Used to specify the path to a file containing a list of trusted Certificate Authority (CA) certificates. **Example:** **ssl_ca_certs: /path/to/ca_certs.pem**
.ssl_ca_data (Optional)	String	Used to specify a string containing the trusted CA certificates in PEM format. **Example:** **ssl_ca_data: <certificate>**
.ssl_check_hostname (Optional)	Boolean	Used when setting up an SSL/TLS connection to a server. It specifies whether the client should check that the hostname in the server's SSL/TLS certificate matches the hostname of the server it is connecting to. **Example:** **ssl_check_hostname: true**

3.7.3. Example Redis configuration

The following YAML shows a sample configuration using Redis with optional SSL/TLS fields:

```
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>
```

NOTE

If your deployment uses Azure Cache for Redis and **ssl** is set to **true**, the port defaults to **6380**.

3.8. MODELCACHE CONFIGURATION OPTIONS

The following options are available on Red Hat Quay for configuring ModelCache.

3.8.1. Memcache configuration option

Memcache is the default ModelCache configuration option. With Memcache, no additional configuration is necessary.

3.8.2. Single Redis configuration option

The following configuration is for a single Redis instance with optional read-only replicas:

```
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. Clustered Redis configuration option

Use the following configuration for a clustered Redis instance:

```
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. TAG EXPIRATION CONFIGURATION FIELDS

The following tag expiration configuration fields are available with Red Hat Quay:

Table 3.9. Tag expiration configuration fields

Field	Type	Description
FEATURE_GARBAGE_COLLECTION	Boolean	Whether garbage collection of repositories is enabled. **Default:** True
TAG_EXPIRATION_OPTIONS (Required)	Array of string	If enabled, the options that users can select for expiration of tags in their namespace. **Pattern:** **^[0-9]+(w|m|d|h|s)$**
DEFAULT_TAG_EXPIRATION (Required)	String	The default, configurable tag expiration time for time machine. **Pattern:** **^[0-9]+(w|m|d|h|s)$** **Default: 2w**
FEATURE_CHANGE_TAG_EXPIRATION	Boolean	Whether users and organizations are allowed to change the tag expiration for tags in their namespace. **Default:** True
FEATURE_AUTO_PRUNE	Boolean	When set to **True**, enables functionality related to the auto-pruning of tags. **Default: False**
NOTIFICATION_TASK_RUN_MINIMUM_INTERVAL_MINUTES	Integer	The interval, in minutes, that defines the frequency to re-run notifications for expiring images. **Default: 300**
DEFAULT_NAMESPACE_AUTOPRUNE_POLICY	Object	The default organization-wide auto-prune policy.
.method: number_of_tags	Object	The option specifying the number of tags to keep.
.value: <integer>	Integer	When used with **method: number_of_tags**, denotes the number of tags to keep. For example, to keep two tags, specify **2**.

Field	Type	Description
.creation_date	Object	The option specifying the duration of which to keep tags.
.value: <integer>	Integer	When used with **creation_date**, denotes how long to keep tags. Can be set to seconds (**s**), days (**d**), months (**m**), weeks (**w**), or years (**y**). Must include a valid integer. For example, to keep tags for one year, specify **1y**.
AUTO_PRUNING_DEFAULT_POLICY_POLL_PERIOD	Integer	The period in which the auto-pruner worker runs at the registry level. By default, it is set to run one time per day (one time per 24 hours). Value must be in seconds.

3.9.1. Example tag expiration configuration

The following YAML example shows you a sample tag expiration configuration.

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

3.9.2. Registry-wide auto-prune policies examples

The following YAML examples show you registry-wide auto-pruning examples by both number of tags and creation date.

Example registry auto-prune policy by number of tags

```
# ...
DEFAULT_NAMESPACE_AUTOPRUNE_POLICY:
  method: number_of_tags
  value: 10 (1)
# ...
```

(1) In this scenario, ten tags remain.

Example registry auto-prune policy by creation date

```
# ...
DEFAULT_NAMESPACE_AUTOPRUNE_POLICY:
  method: creation_date
  value: 1y
# ...
```

3.10. QUOTA MANAGEMENT CONFIGURATION FIELDS

Table 3.10. Quota management configuration

Field	Type	Description
FEATURE_QUOTA_MANAGEMENT	Boolean	Enables configuration, caching, and validation for quota management feature. **Default:** `False`
DEFAULT_SYSTEM_REJECT_QUOTA_BYTES	String	Enables system default quota reject byte allowance for all organizations. By default, no limit is set.
QUOTA_BACKFILL	Boolean	Enables the quota backfill worker to calculate the size of pre-existing blobs. **Default**: **True**
QUOTA_TOTAL_DELAY_SECONDS	String	The time delay for starting the quota backfill. Rolling deployments can cause incorrect totals. This field **must** be set to a time longer than it takes for the rolling deployment to complete. **Default**: **1800**
PERMANENTLY_DELETE_TAGS	Boolean	Enables functionality related to the removal of tags from the time machine window. **Default**: **False**
RESET_CHILD_MANIFEST_EXPIRATION	Boolean	Resets the expirations of temporary tags targeting the child manifests. With this feature set to **True**, child manifests are immediately garbage collected. **Default**: **False**

3.10.1. Example quota management configuration

The following YAML is the suggested configuration when enabling quota management.

Quota management YAML configuration

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

3.11. PROXY CACHE CONFIGURATION FIELDS

Table 3.11. Proxy configuration

Field	Type	Description
FEATURE_PROXY_CACHE	Boolean	Enables Red Hat Quay to act as a pull through cache for upstream registries. **Default: false**

3.12. ROBOT ACCOUNT CONFIGURATION FIELDS

Table 3.12. Robot account configuration fields

Field	Type	Description
ROBOTS_DISALLOW	Boolean	When set to **true**, robot accounts are prevented from all interactions, as well as from being created **Default: False**

3.13. PRE-CONFIGURING RED HAT QUAY FOR AUTOMATION

Red Hat Quay supports several configuration options that enable automation. Users can configure these options before deployment to reduce the need for interaction with the user interface.

3.13.1. Allowing the API to create the first user

To create the first user, users need to set the **FEATURE_USER_INITIALIZE** parameter to **true** and call the **/api/v1/user/initialize** API. Unlike all other registry API calls that require an OAuth token generated by an OAuth application in an existing organization, the API endpoint does not require authentication.

Users can use the API to create a user such as **quayadmin** after deploying Red Hat Quay, provided no other users have been created. For more information, see Using the API to create the first user .

3.13.2. Enabling general API access

Users should set the **BROWSER_API_CALLS_XHR_ONLY** configuration option to **false** to allow general access to the Red Hat Quay registry API.

3.13.3. Adding a superuser

After deploying Red Hat Quay, users can create a user and give the first user administrator privileges with full permissions. Users can configure full permissions in advance by using the **SUPER_USER** configuration object. For example:

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

3.13.4. Restricting user creation

After you have configured a superuser, you can restrict the ability to create new users to the superuser group by setting the **FEATURE_USER_CREATION** to **false**. For example:

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

3.13.5. Enabling new functionality in Red Hat Quay 3

To use new Red Hat Quay 3 functions, enable some or all of the following features:

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

3.13.6. Suggested configuration for automation

The following **config.yaml** parameters are suggested for automation:

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

3.13.7. Deploying the Red Hat Quay Operator using the initial configuration

Use the following procedure to deploy Red Hat Quay on OpenShift Container Platform using the initial configuration.

Prerequisites

- You have installed the **oc** CLI.

Procedure

1. Create a secret using the configuration file:

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

2. Create a **quayregistry.yaml** file. Identify the unmanaged components and reference the created secret, for example:

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

3. Deploy the Red Hat Quay registry:

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

Next Steps

- Using the API to create the first user

3.13.8. Using the API to create the first user

Use the following procedure to create the first user in your Red Hat Quay organization.

Prerequisites

- The config option **FEATURE_USER_INITIALIZE** must be set to **true**.
- No users can already exist in the database.

PROCEDURE

This procedure requests an OAuth token by specifying **"access_token": true**.

1. Open your Red Hat Quay configuration file and update the following configuration fields:

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

2. Stop the Red Hat Quay service by entering the following command:

```
$ sudo podman stop quay
```

3. Start the Red Hat Quay service by entering the following command:

```
$ 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. Run the following **CURL** command to generate a new user with a username, password, email, and access token:

```
$ 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}'
```

If successful, the command returns an object with the username, email, and encrypted password. For example:

```
{"access_token":"6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED",
"email":"quayadmin@example.com","encrypted_password":"1nZMLH57RIE5UGdL/yYpDOHL
qiNCgimb6W9kfF8MjZ1xrfDpRyRs9NUnUuNuAitW","username":"quayadmin"} #
gitleaks:allow
```

If a user already exists in the database, an error is returned:

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

If your password is not at least eight characters or contains whitespace, an error is returned:

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

5. Log in to your Red Hat Quay deployment by entering the following command:

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

Example output

```
Login Succeeded!
```

3.13.8.1. Using the OAuth token

After invoking the API, you can call out the rest of the Red Hat Quay API by specifying the returned OAuth code.

Prerequisites

- You have invoked the **/api/v1/user/initialize** API, and passed in the username, password, and email address.

Procedure

- Obtain the list of current users by entering the following command:

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

Example output:


```
{
   "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
      }
   ]
}
```


In this instance, the details for the **quayadmin** user are returned as it is the only user that has been created so far.

3.13.8.2. Using the API to create an organization

The following procedure details how to use the API to create a Red Hat Quay organization.

Prerequisites

- You have invoked the **/api/v1/user/initialize** API, and passed in the username, password, and email address.
- You have called out the rest of the Red Hat Quay API by specifying the returned OAuth code.

Procedure

1. To create an organization, use a POST call to **api/v1/organization/** endpoint:

```
$ 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"}'
```

Example output:

```
"Created"
```

2. You can retrieve the details of the organization you created by entering the following command:

```
$ 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
```

Example output:

```
{
    "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. BASIC CONFIGURATION FIELDS

Table 3.13. Basic configuration

Field	Type	Description
REGISTRY_TITLE	String	If specified, the long-form title for the registry. Displayed in frontend of your Red Hat Quay deployment, for example, at the sign in page of your organization. Should not exceed 35 characters. **Default:** **Red Hat Quay**
REGISTRY_TITLE_SHORT	String	If specified, the short-form title for the registry. Title is displayed on various pages of your organization, for example, as the title of the tutorial on your organization's **Tutorial** page. **Default:** **Red Hat Quay**
CONTACT_INFO	Array of String	If specified, contact information to display on the contact page. If only a single piece of contact information is specified, the contact footer will link directly.
[0]	String	Adds a link to send an e-mail. **Pattern:** **^mailto:(.)+$** **Example:** **mailto:support@quay.io**
[1]	String	Adds a link to visit an IRC chat room. **Pattern:** **^irc://(.)+$** **Example:** **irc://chat.freenode.net:6665/quay**
[2]	String	Adds a link to call a phone number. **Pattern:** **^tel:(.)+$** **Example:** **tel:+1-888-930-3475**

Field	Type	Description
[3]	String	Adds a link to a defined URL. **Pattern:** **^http(s)?://(.)+$** **Example:** **https://twitter.com/quayio**

3.15. SSL CONFIGURATION FIELDS

Table 3.14. SSL configuration

Field	Type	Description
PREFERRED_URL_SCHEME	String	One of **http** or **https**. Note that users only set their **PREFERRED_URL_SCHEME** to **http** when there is no TLS encryption in the communication path from the client to Quay. Users must set their **PREFERRED_URL_SCHEME `to `https** when using a TLS-terminating load balancer, a reverse proxy (for example, Nginx), or when using Quay with custom SSL certificates directly. In most cases, the **PREFERRED_URL_SCHEME** should be **https**. **Default: http**
SERVER_HOSTNAME (Required)	String	The URL at which Red Hat Quay is accessible, without the scheme **Example:** **quay-server.example.com**

Field	Type	Description
SSL_CIPHERS	Array of String	If specified, the nginx-defined list of SSL ciphers to enabled and disabled Example: [**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	If specified, nginx is configured to enabled a list of SSL protocols defined in the list. Removing an SSL protocol from the list disables the protocol during Red Hat Quay startup. Example: **['TLSv1','TLSv1.1','TLSv1.2', `TLSv1.3`]`**

Field	Type	Description
SESSION_COOKIE_SECURE	Boolean	Whether the **secure** property should be set on session cookies **Default:** False **Recommendation:** Set to True for all installations using SSL

3.15.1. Configuring SSL

1. Copy the certificate file and primary key file to your configuration directory, ensuring they are named **ssl.cert** and **ssl.key** respectively:

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

2. Edit the **config.yaml** file and specify that you want Quay to handle TLS:

   config.yaml

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

3. Stop the **Quay** container and restart the registry

3.16. ADDING TLS CERTIFICATES TO THE RED HAT QUAY CONTAINER

To add custom TLS certificates to Red Hat Quay, create a new directory named **extra_ca_certs/** beneath the Red Hat Quay config directory. Copy any required site-specific TLS certificates to this new directory.

3.16.1. Add TLS certificates to Red Hat Quay

1. View certificate to be added to the container

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

2. Create certs directory and copy certificate there

```
$ 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. Obtain the **Quay** container's **CONTAINER ID** with **podman ps**:

```
$ sudo podman ps
CONTAINER ID  IMAGE                           COMMAND               CREATED
STATUS            PORTS
5a3e82c4a75f  <registry>/<repo>/quay:v3.12.1 "/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. Restart the container with that ID:

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

5. Examine the certificate copied into the container namespace:

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

3.17. LDAP CONFIGURATION FIELDS

Table 3.15. LDAP configuration

Field	Type	Description
AUTHENTICATION_TYPE (Required)	String	Must be set to **LDAP**.
FEATURE_TEAM_SYNCING	Boolean	Whether to allow for team membership to be synced from a backing group in the authentication engine (OIDC, LDAP, or Keystone). **Default: true**
FEATURE_NONSUPERUSER_TEAM_SYNCING_SETUP	Boolean	If enabled, non-superusers can setup team syncrhonization. **Default: false**
LDAP_ADMIN_DN	String	The admin DN for LDAP authentication.
LDAP_ADMIN_PASSWD	String	The admin password for LDAP authentication.

Field	Type	Description
LDAP_ALLOW_INSECURE_FALLBACK	Boolean	Whether or not to allow SSL insecure fallback for LDAP authentication.
LDAP_BASE_DN	Array of String	The base DN for LDAP authentication.
LDAP_EMAIL_ATTR	String	The email attribute for LDAP authentication.
LDAP_UID_ATTR	String	The uid attribute for LDAP authentication.
LDAP_URI	String	The LDAP URI.
LDAP_USER_FILTER	String	The user filter for LDAP authentication.
LDAP_USER_RDN	Array of String	The user RDN for LDAP authentication.
LDAP_SECONDARY_USER_RDNS	Array of String	Provide Secondary User Relative DNs if there are multiple Organizational Units where user objects are located.
TEAM_RESYNC_STALE_TIME	String	If team syncing is enabled for a team, how often to check its membership and resync if necessary. **Pattern:** **^[0-9]+(w|m|d|h|s)$** **Example:** **2h** **Default:** **30m**
LDAP_SUPERUSER_FILTER	String	Subset of the **LDAP_USER_FILTER** configuration field. When configured, allows Red Hat Quay administrators the ability to configure Lightweight Directory Access Protocol (LDAP) users as superusers when Red Hat Quay uses LDAP as its authentication provider. With this field, administrators can add or remove superusers without having to update the Red Hat Quay configuration file and restart their deployment. This field requires that your **AUTHENTICATION_TYPE** is set to **LDAP**.

Field	Type	Description
GLOBAL_READONLY_SUPER_USERS	String	When set, grants users of this list read access to all repositories, regardless of whether they are public repositories. Only works for those superusers defined with the **LDAP_SUPERUSER_FILTER** configuration field.
LDAP_RESTRICTED_USER_FILTER	String	Subset of the **LDAP_USER_FILTER** configuration field. When configured, allows Red Hat Quay administrators the ability to configure Lightweight Directory Access Protocol (LDAP) users as restricted users when Red Hat Quay uses LDAP as its authentication provider. This field requires that your **AUTHENTICATION_TYPE** is set to **LDAP**.
FEATURE_RESTRICTED_USERS	Boolean	When set to **True** with **LDAP_RESTRICTED_USER_FILTER** active, only the listed users in the defined LDAP group are restricted. **Default: False**
LDAP_TIMEOUT	Integer	Specifies the time limit, in seconds, for LDAP operations. This limits the amount of time an LDAP search, bind, or other operation can take. Similar to the **-l** option in **ldapsearch**, it sets a client-side operation timeout. **Default: 10**
LDAP_NETWORK_TIMEOUT	Integer	Specifies the time limit, in seconds, for establishing a connection to the LDAP server. This is the maximum time Red Hat Quay waits for a response during network operations, similar to the **-o nettimeout** option in **ldapsearch**. **Default: 10**

3.17.1. LDAP configuration references

Use the following references to update your **config.yaml** file with the desired LDAP settings.

3.17.1.1. Basic LDAP configuration

Use the following reference for a basic LDAP configuration.

```
---
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
LDAP_SECONDARY_USER_RDNS: (11)
    - ou=<example_organization_unit_one>
    - ou=<example_organization_unit_two>
    - ou=<example_organization_unit_three>
    - ou=<example_organization_unit_four>
```

(1) Required. Must be set to **LDAP**.

(2) Required. The admin DN for LDAP authentication.

(3) Required. The admin password for LDAP authentication.

(4) Required. Whether to allow SSL/TLS insecure fallback for LDAP authentication.

(5) Required. The base DN for LDAP authentication.

(6) Required. The email attribute for LDAP authentication.

(7) Required. The UID attribute for LDAP authentication.

(8) Required. The LDAP URI.

(9) Required. The user filter for LDAP authentication.

(10) Required. The user RDN for LDAP authentication.

(11) Optional. Secondary User Relative DNs if there are multiple Organizational Units where user objects are located.

3.17.1.2. LDAP restricted user configuration

Use the following reference for an LDAP restricted user configuration.

```
# ...
AUTHENTICATION_TYPE: LDAP
# ...
FEATURE_RESTRICTED_USERS: true (1)
# ...
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>) (2)
LDAP_USER_RDN:
  - ou=<example_organization_unit>
  - o=<organization_id>
  - dc=<example_domain_component>
  - dc=com
# ...
```

1. Must be set to **true** when configuring an LDAP restricted user.
2. Configures specified users as restricted users.

3.17.1.3. LDAP superuser configuration reference

Use the following reference for an LDAP superuser configuration.

```
# ...
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 Configures specified users as superusers.

3.18. MIRRORING CONFIGURATION FIELDS

Table 3.16. Mirroring configuration

Field	Type	Description
FEATURE_REPO_MIRROR	Boolean	Enable or disable repository mirroring **Default:** **false**
REPO_MIRROR_INTERVAL	Number	The number of seconds between checking for repository mirror candidates **Default:** 30
REPO_MIRROR_SERVER_HOSTNAME	String	Replaces the **SERVER_HOSTNAME** as the destination for mirroring. **Default:** None **Example**: **openshift-quay-service**
REPO_MIRROR_TLS_VERIFY	Boolean	Require HTTPS and verify certificates of Quay registry during mirror. **Default:** **false**
REPO_MIRROR_ROLLBACK	Boolean	When set to **true**, the repository rolls back after a failed mirror attempt. **Default**: **false**

3.19. SECURITY SCANNER CONFIGURATION FIELDS

Table 3.17. Security scanner configuration

Field	Type	Description
FEATURE_SECURITY_SCANNER	Boolean	Enable or disable the security scanner **Default: false**
FEATURE_SECURITY_NOTIFICATIONS	Boolean	If the security scanner is enabled, turn on or turn off security notifications **Default: false**
SECURITY_SCANNER_V4_REINDEX_THRESHOLD	String	This parameter is used to determine the minimum time, in seconds, to wait before re-indexing a manifest that has either previously failed or has changed states since the last indexing. The data is calculated from the **last_indexed datetime** in the **manifestsecuritystatus** table. This parameter is used to avoid trying to re-index every failed manifest on every indexing run. The default time to re-index is 300 seconds.
SECURITY_SCANNER_V4_ENDPOINT	String	The endpoint for the V4 security scanner **Pattern: ^http(s)?://(.)+$** **Example: http://192.168.99.101:6060**
SECURITY_SCANNER_V4_PSK	String	The generated pre-shared key (PSK) for Clair
SECURITY_SCANNER_ENDPOINT	String	The endpoint for the V2 security scanner **Pattern: ^http(s)?://(.)+$** **Example: http://192.168.99.100:6060**

Field	Type	Description
SECURITY_SCANNER_INDEXING_INTERVAL	Integer	This parameter is used to determine the number of seconds between indexing intervals in the security scanner. When indexing is triggered, Red Hat Quay will query its database for manifests that must be indexed by Clair. These include manifests that have not yet been indexed and manifests that previously failed indexing. **Default:** 30
FEATURE_SECURITY_SCANNER_NOTIFY_ON_NEW_INDEX	Boolean	Whether to allow sending notifications about vulnerabilities for new pushes. **Default**: **True**
SECURITY_SCANNER_V4_MANIFEST_CLEANUP	Boolean	Whether the Red Hat Quay garbage collector removes manifests that are not referenced by other tags or manifests. **Default**: **True**
NOTIFICATION_MIN_SEVERITY_ON_NEW_INDEX	String	Set minimal security level for new notifications on detected vulnerabilities. Avoids creation of large number of notifications after first index. If not defined, defaults to **High**. Available options include **Critical**, **High**, **Medium**, **Low**, **Negligible**, and **Unknown**.
SECURITY_SCANNER_V4_INDEX_MAX_LAYER_SIZE	String	The maximum layer size allowed for indexing. If the layer size exceeds the configured size, the Red Hat Quay UI returns the following message: **The manifest for this tag has layer(s) that are too large to index by the Quay Security Scanner**. **Example**: **8G**

3.19.1. Re-indexing with Clair v4

When Clair v4 indexes a manifest, the result should be deterministic. For example, the same manifest should produce the same index report. This is true until the scanners are changed, as using different scanners will produce different information relating to a specific manifest to be returned in the report.

Because of this, Clair v4 exposes a state representation of the indexing engine (**/indexer/api/v1/index_state**) to determine whether the scanner configuration has been changed.

Red Hat Quay leverages this index state by saving it to the index report when parsing to Quay's database. If this state has changed since the manifest was previously scanned, Red Hat Quay will attempt to re-index that manifest during the periodic indexing process.

By default this parameter is set to 30 seconds. Users might decrease the time if they want the indexing process to run more frequently, for example, if they did not want to wait 30 seconds to see security scan results in the UI after pushing a new tag. Users can also change the parameter if they want more control over the request pattern to Clair and the pattern of database operations being performed on the Red Hat Quay database.

3.19.2. Example security scanner configuration

The following YAML is the suggested configuration when enabling the security scanner feature.

Security scanner YAML configuration

```
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
SECURITY_SCANNER_V4_INDEX_MAX_LAYER_SIZE: 8G
...
```

3.20. HELM CONFIGURATION FIELDS

Table 3.18. Helm configuration fields

Field	Type	Description
FEATURE_GENERAL_OCI_SUPPORT	Boolean	Enable support for OCI artifacts. **Default:** True

The following Open Container Initiative (OCI) artifact types are built into Red Hat Quay by default and are enabled through the **FEATURE_GENERAL_OCI_SUPPORT** configuration field:

Field	Media Type	Supported content types
Helm	**application/vnd.cncf.helm.config.v1+json**	**application/tar+gzip**, **application/vnd.cncf.helm.chart.content.v1.tar+gzip**

Field	Media Type	Supported content types
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**
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. Configuring Helm

The following YAML is the example configuration when enabling Helm.

Helm YAML configuration

```
FEATURE_GENERAL_OCI_SUPPORT: true
```

3.21. OPEN CONTAINER INITIATIVE CONFIGURATION FIELDS

Table 3.19. Additional OCI artifact configuration field

Field	Type	Description
FEATURE_REFERRERS_API	Boolean	Enables OCI 1.1's referrers API.

Example OCI referrers enablement YAML

```
# ...
FEATURE_REFERRERS_API: True
# ...
```

3.22. UNKNOWN MEDIA TYPES

Table 3.20. Unknown media types configuration field

Field	Type	Description
IGNORE_UNKNOWN_MEDIATYPES	Boolean	When enabled, allows a container registry platform to disregard specific restrictions on supported artifact types and accept any unrecognized or unknown media types. Default: **false**

3.22.1. Configuring unknown media types

The following YAML is the example configuration when enabling unknown or unrecognized media types.

Unknown media types YAML configuration

```
IGNORE_UNKNOWN_MEDIATYPES: true
```

3.23. ACTION LOG CONFIGURATION FIELDS

3.23.1. Action log storage configuration

Table 3.21. Action log storage configuration

Field	Type	Description
FEATURE_LOG_EXPORT	Boolean	Whether to allow exporting of action logs. Default: **True**
LOGS_MODEL	String	Specifies the preferred method for handling log data. Values: One of **database**, **transition_reads_both_writes_es**, **elasticsearch**, **splunk** Default: **database**
LOGS_MODEL_CONFIG	Object	Logs model config for action logs.

Field	Type	Description
ALLOW_WITHOUT_STRICT_LOGGING	Boolean	When set to **True**, if the external log system like Splunk or ElasticSearch is intermittently unavailable, allows users to push images normally. Events are logged to the stdout instead. **Default:** False

3.23.1.1. Elasticsearch configuration fields

The following fields are available when configuring Elasticsearch for Red Hat Quay.

- **LOGS_MODEL_CONFIG** [object]: Logs model config for action logs.
  - **elasticsearch_config** [object]: Elasticsearch cluster configuration.
    - **access_key** [string]: Elasticsearch user (or IAM key for AWS ES).
      - Example: **some_string**
    - **host** [string]: Elasticsearch cluster endpoint.
      - Example: **host.elasticsearch.example**
    - **index_prefix** [string]: Elasticsearch's index prefix.
      - Example: **logentry_**
    - **index_settings** [object]: Elasticsearch's index settings
    - **use_ssl** [boolean]: Use ssl for Elasticsearch. Defaults to **True**.
      - Example: **True**
    - **secret_key** [string]: Elasticsearch password (or IAM secret for AWS ES).
      - Example: **some_secret_string**
    - **aws_region** [string]: Amazon web service region.
      - Example: **us-east-1**
    - **port** [number]: Elasticsearch cluster endpoint port.
      - Example: **1234**
  - **kinesis_stream_config** [object]: AWS Kinesis Stream configuration.
    - **aws_secret_key** [string]: AWS secret key.
      - Example: **some_secret_key**
    - **stream_name** [string]: Kinesis stream to send action logs to.

- Example: **logentry-kinesis-stream**

- **aws_access_key** [string]: AWS access key.
  - Example: **some_access_key**
- **retries** [number]: Max number of attempts made on a single request.
  - Example: **5**
- **read_timeout** [number]: Number of seconds before timeout when reading from a connection.
  - Example: **5**
- **max_pool_connections** [number]: The maximum number of connections to keep in a connection pool.
  - Example: **10**
- **aws_region** [string]: AWS region.
  - Example: **us-east-1**
- **connect_timeout** [number]: Number of seconds before timeout when attempting to make a connection.
  - Example: **5**

- **producer** [string]: Logs producer if logging to Elasticsearch.
  - **enum**: kafka, elasticsearch, kinesis_stream
  - Example: **kafka**
- **kafka_config** [object]: Kafka cluster configuration.
  - **topic** [string]: Kafka topic to publish log entries to.
    - Example: **logentry**
  - **bootstrap_servers** [array]: List of Kafka brokers to bootstrap the client from.
  - **max_block_seconds** [number]: Max number of seconds to block during a **send()**, either because the buffer is full or metadata unavailable.
    - Example: **10**

3.23.1.2. Splunk configuration fields

The following fields are available when configuring Splunk for Red Hat Quay.

- **producer** [string]: **splunk**. Use when configuring Splunk.
- **splunk_config** [object]: Logs model configuration for Splunk action logs or the Splunk cluster configuration.
  - **host** [string]: Splunk cluster endpoint.

- **port** [integer]: Splunk management cluster endpoint port.
- **bearer_token** [string]: The bearer token for Splunk.
- **verify_ssl** [boolean]: Enable (**True**) or disable (**False**) TLS/SSL verification for HTTPS connections.
- **index_prefix** [string]: Splunk's index prefix.
- **ssl_ca_path** [string]: The relative container path to a single **.pem** file containing a certificate authority (CA) for SSL validation.

Example Splunk configuration

```
# ...
LOGS_MODEL: splunk
LOGS_MODEL_CONFIG:
   producer: splunk
   splunk_config:
      host: http://<user_name>.remote.csb
      port: 8089
      bearer_token: <bearer_token>
      url_scheme: <http/https>
      verify_ssl: False
      index_prefix: <splunk_log_index_name>
      ssl_ca_path: <location_to_ssl-ca-cert.pem>
# ...
```

3.23.1.3. Splunk HEC configuration fields

The following fields are available when configuring Splunk HTTP Event Collector (HEC) for Red Hat Quay.

- **producer** [string]: **splunk_hec**. Use when configuring Splunk HEC.
- **splunk_hec_config** [object]: Logs model configuration for Splunk HTTP event collector action logs configuration.
  - **host** [string]: Splunk cluster endpoint.
  - **port** [integer]: Splunk management cluster endpoint port.
  - **hec_token** [string]: HEC token for Splunk.
  - **url_scheme** [string]: The URL scheme for access the Splunk service. If Splunk is behind SSL/TLS, must be **https**.
  - **verify_ssl** [boolean]: Enable (**true**) or disable (**false**) SSL/TLS verification for HTTPS connections.
  - **index** [string]: The Splunk index to use.
  - **splunk_host** [string]: The host name to log this event.
  - **splunk_sourcetype** [string]: The name of the Splunk **sourcetype** to use.

```
# ...
LOGS_MODEL: splunk
LOGS_MODEL_CONFIG:
  producer: splunk_hec
  splunk_hec_config: 1
    host: prd-p-aaaaaq.splunkcloud.com 2
    port: 8088 3
    hec_token: 12345678-1234-1234-1234-1234567890ab 4
    url_scheme: https 5
    verify_ssl: False 6
    index: quay 7
    splunk_host: quay-dev 8
    splunk_sourcetype: quay_logs 9
# ...
```

3.23.2. Action log rotation and archiving configuration

Table 3.22. Action log rotation and archiving configuration

Field	Type	Description
FEATURE_ACTION_LOG_ROTATION	Boolean	Enabling log rotation and archival will move all logs older than 30 days to storage. Default: **false**
ACTION_LOG_ARCHIVE_LOCATION	String	If action log archiving is enabled, the storage engine in which to place the archived data. Example:: **s3_us_east**
ACTION_LOG_ARCHIVE_PATH	String	If action log archiving is enabled, the path in storage in which to place the archived data. Example: **archives/actionlogs**
ACTION_LOG_ROTATION_THRESHOLD	String	The time interval after which to rotate logs. Example: **30d**

3.23.3. Action log audit configuration

Table 3.23. Audit logs configuration field

Field	Type	Description
ACTION_LOG_AUDIT_LOGINS	Boolean	When set to **True**, tracks advanced events such as logging into, and out of, the UI, and logging in using Docker for regular users, robot accounts, and for application-specific token accounts. Default: **True**

3.24. BUILD LOGS CONFIGURATION FIELDS

Table 3.24. Build logs configuration fields

Field	Type	Description
FEATURE_READER_BUILD_LOGS	Boolean	If set to true, build logs can be read by those with **read** access to the repository, rather than only **write** access or **admin** access. Default: **False**
LOG_ARCHIVE_LOCATION	String	The storage location, defined in **DISTRIBUTED_STORAGE_CONFIG**, in which to place the archived build logs. Example: **s3_us_east**
LOG_ARCHIVE_PATH	String	The path under the configured storage engine in which to place the archived build logs in **.JSON** format. Example: **archives/buildlogs**

3.25. DOCKERFILE BUILD TRIGGERS FIELDS

Table 3.25. Dockerfile build support

Field	Type	Description
FEATURE_BUILD_SUPPORT	Boolean	Whether to support Dockerfile build. Default: **False**

Field	Type	Description
SUCCESSIVE_TRIGGER_FAILURE_DISABLE_THRESHOLD	Number	If not set to **None**, the number of successive failures that can occur before a build trigger is automatically disabled. **Default: 100**
SUCCESSIVE_TRIGGER_INTERNAL_ERROR_DISABLE_THRESHOLD	Number	If not set to **None**, the number of successive internal errors that can occur before a build trigger is automatically disabled **Default: 5**

3.25.1. GitHub build triggers

Table 3.26. GitHub build triggers

Field	Type	Description
FEATURE_GITHUB_BUILD	Boolean	Whether to support GitHub build triggers. **Default: False**
GITHUB_TRIGGER_CONFIG	Object	Configuration for using GitHub Enterprise for build triggers.
.GITHUB_ENDPOINT (Required)	String	The endpoint for GitHub Enterprise. **Example: https://github.com/**
.API_ENDPOINT	String	The endpoint of the GitHub Enterprise API to use. Must be overridden for **github.com**. **Example**: **https://api.github.com/**
.CLIENT_ID (Required)	String	The registered client ID for this Red Hat Quay instance; this cannot be shared with **GITHUB_LOGIN_CONFIG**.

Field	Type	Description
.CLIENT_SECRET (Required)	String	The registered client secret for this Red Hat Quay instance.

3.25.2. BitBucket build triggers

Table 3.27. BitBucket build triggers

Field	Type	Description
FEATURE_BITBUCKET_BUILD	Boolean	Whether to support Bitbucket build triggers. **Default:** False
BITBUCKET_TRIGGER_CONFIG	Object	Configuration for using BitBucket for build triggers.
.CONSUMER_KEY (Required)	String	The registered consumer key (client ID) for this Red Hat Quay instance.
.CONSUMER_SECRET (Required)	String	The registered consumer secret (client secret) for this Red Hat Quay instance.

3.25.3. GitLab build triggers

Table 3.28. GitLab build triggers

Field	Type	Description
FEATURE_GITLAB_BUILD	Boolean	Whether to support GitLab build triggers. **Default:** False
GITLAB_TRIGGER_CONFIG	Object	Configuration for using Gitlab for build triggers.
.GITLAB_ENDPOINT (Required)	String	The endpoint at which Gitlab Enterprise is running.

Field	Type	Description
.CLIENT_ID (Required)	String	The registered client ID for this Red Hat Quay instance.
.CLIENT_SECRET (Required)	String	The registered client secret for this Red Hat Quay instance.

3.26. BUILD MANAGER CONFIGURATION FIELDS

Table 3.29. Build manager configuration fields

Field	Type	Description
ALLOWED_WORKER_COUNT	String	Defines how many Build Workers are instantiated per Red Hat Quay pod. Typically set to **1**.
ORCHESTRATOR_PREFIX	String	Defines a unique prefix to be added to all Redis keys. This is useful to isolate Orchestrator values from other Redis keys.
REDIS_HOST	Object	The hostname for your Redis service.
REDIS_PASSWORD	String	The password to authenticate into your Redis service.
REDIS_SSL	Boolean	Defines whether or not your Redis connection uses SSL/TLS.
REDIS_SKIP_KEYSPACE_EVENT_SETUP	Boolean	By default, Red Hat Quay does not set up the keyspace events required for key events at runtime. To do so, set **REDIS_SKIP_KEYSPACE_EVENT_SETUP** to **false**.
EXECUTOR	String	Starts a definition of an Executor of this type. Valid values are **kubernetes** and **ec2**.
BUILDER_NAMESPACE	String	Kubernetes namespace where Red Hat Quay Builds will take place.

Field	Type	Description
K8S_API_SERVER	Object	Hostname for API Server of the OpenShift Container Platform cluster where Builds will take place.
K8S_API_TLS_CA	Object	The filepath in the **Quay** container of the Build cluster's CA certificate for the **Quay** application to trust when making API calls.
KUBERNETES_DISTRIBUTION	String	Indicates which type of Kubernetes is being used. Valid values are **openshift** and **k8s**.
CONTAINER_*	Object	Define the resource requests and limits for each **build** pod.
NODE_SELECTOR_*	Object	Defines the node selector label name-value pair where **build** Pods should be scheduled.
CONTAINER_RUNTIME	Object	Specifies whether the Builder should run **docker** or **podman**. Customers using Red Hat's **quay-builder** image should set this to **podman**.
SERVICE_ACCOUNT_NAME/SERVICE_ACCOUNT_TOKEN	Object	Defines the Service Account name or token that will be used by **build** pods.
QUAY_USERNAME/QUAY_PASSWORD	Object	Defines the registry credentials needed to pull the Red Hat Quay build worker image that is specified in the **WORKER_IMAGE** field. Customers should provide a Red Hat Service Account credential as defined in the section "Creating Registry Service Accounts" against registry.redhat.io in the article at https://access.redhat.com/RegistryAuthentication.
WORKER_IMAGE	Object	Image reference for the Red Hat Quay Builder image. registry.redhat.io/quay/quay-builder

Field	Type	Description
WORKER_TAG	Object	Tag for the Builder image desired. The latest version is 3.
BUILDER_VM_CONTAINER_IMAGE	Object	The full reference to the container image holding the internal VM needed to run each Red Hat Quay Build. (**registry.redhat.io/quay/quay-builder-qemu-rhcos:3**).
SETUP_TIME	String	Specifies the number of seconds at which a Build times out if it has not yet registered itself with the Build Manager. Defaults at **500** seconds. Builds that time out are attempted to be restarted three times. If the Build does not register itself after three attempts it is considered failed.

Field	Type	Description
MINIMUM_RETRY_THRESHOLD	String	This setting is used with multiple Executors. It indicates how many retries are attempted to start a Build before a different Executor is chosen. Setting to **0** means there are no restrictions on how many tries the build job needs to have. This value should be kept intentionally small (three or less) to ensure failovers happen quickly during infrastructure failures. You must specify a value for this setting. For example, **Kubernetes** is set as the first executor and **EC2** as the second executor. If you want the last attempt to run a job to always be executed on EC2 and not Kubernetes, you can set the Kubernetes executor's **MINIMUM_RETRY_THRESHOLD** to **1** and EC2's **MINIMUM_RETRY_THRESHOLD** to **0** (defaults to **0** if not set). In this case, the Kubernetes' **MINIMUM_RETRY_THRESHOLD retries_remaining(1)** would evaluate to **False**, therefore falling back to the second executor configured.
SSH_AUTHORIZED_KEYS	Object	List of SSH keys to bootstrap in the **ignition** config. This allows other keys to be used to SSH into the EC2 instance or QEMU virtual machine (VM).

3.27. OAUTH CONFIGURATION FIELDS

Table 3.30. OAuth fields

Field	Type	Description
DIRECT_OAUTH_CLIENTID_WHITELIST	Array of String	A list of client IDs for **Quay-managed** applications that are allowed to perform direct OAuth approval without user approval.

Field	Type	Description
FEATURE_ASSIGN_OAUTH_TOKEN	Boolean	Allows organization administrators to assign OAuth tokens to other users.

3.27.1. GitHub OAuth configuration fields

Table 3.31. GitHub OAuth fields

Field	Type	Description
FEATURE_GITHUB_LOGIN	Boolean	Whether GitHub login is supported **Default: **False**
GITHUB_LOGIN_CONFIG	Object	Configuration for using GitHub (Enterprise) as an external login provider.
.ALLOWED_ORGANIZATIONS	Array of String	The names of the GitHub (Enterprise) organizations whitelisted to work with the ORG_RESTRICT option.
.API_ENDPOINT	String	The endpoint of the GitHub (Enterprise) API to use. Must be overridden for github.com **Example:** **https://api.github.com/**
.CLIENT_ID (Required)	String	The registered client ID for this Red Hat Quay instance; cannot be shared with **GITHUB_TRIGGER_CONFIG**. **Example:** **0e8dbe15c4c7630b6780**
.CLIENT_SECRET (Required)	String	The registered client secret for this Red Hat Quay instance. **Example:** **e4a58ddd3d7408b7aec109e85564a0d153d3e846**

Field	Type	Description
.GITHUB_ENDPOINT (Required)	String	The endpoint for GitHub (Enterprise). **Example**: https://github.com/
.ORG_RESTRICT	Boolean	If true, only users within the organization whitelist can login using this provider.

3.27.2. Google OAuth configuration fields

Table 3.32. Google OAuth fields

Field	Type	Description
FEATURE_GOOGLE_LOGIN	Boolean	Whether Google login is supported. **Default: **False**
GOOGLE_LOGIN_CONFIG	Object	Configuration for using Google for external authentication.
.CLIENT_ID (Required)	String	The registered client ID for this Red Hat Quay instance. **Example: 0e8dbe15c4c7630b6780**
.CLIENT_SECRET (Required)	String	The registered client secret for this Red Hat Quay instance. **Example: e4a58ddd3d7408b7aec109e85564a0d153d3e846**

3.28. OIDC CONFIGURATION FIELDS

Table 3.33. OIDC fields

Field	Type	Description

<string>_LOGIN_CONFIG (Required)	String	The parent key that holds the OIDC configuration settings. Typically the name of the OIDC provider, for example, **AZURE_LOGIN_CONFIG**, however any arbitrary string is accepted.
.CLIENT_ID (Required)	String	The registered client ID for this Red Hat Quay instance. **Example:** **0e8dbe15c4c7630b6780**
.CLIENT_SECRET (Required)	String	The registered client secret for this Red Hat Quay instance. **Example:** **e4a58ddd3d7408b7aec109e85564a0d153d3e846**
.DEBUGLOG	Boolean	Whether to enable debugging.
.LOGIN_BINDING_FIELD	String	Used when the internal authorization is set to LDAP. Red Hat Quay reads this parameter and tries to search through the LDAP tree for the user with this username. If it exists, it automatically creates a link to that LDAP account.
.LOGIN_SCOPES	Object	Adds additional scopes that Red Hat Quay uses to communicate with the OIDC provider.
.OIDC_ENDPOINT_CUSTOM_PARAMS	String	Support for custom query parameters on OIDC endpoints. The following endpoints are supported: **authorization_endpoint**, **token_endpoint**, and **user_endpoint**.

.OIDC_ISSUER	String	Allows the user to define the issuer to verify. For example, JWT tokens container a parameter known as **iss** which defines who issued the token. By default, this is read from the **.well-know/openid/configuration** endpoint, which is exposed by every OIDC provider. If this verification fails, there is no login.
.OIDC_SERVER (Required)	String	The address of the OIDC server that is being used for authentication. **Example: https://sts.windows.net/6c878…/**
.PREFERRED_USERNAME_CLAIM_NAME	String	Sets the preferred username to a parameter from the token.
.SERVICE_ICON	String	Changes the icon on the login screen.
.SERVICE_NAME (Required)	String	The name of the service that is being authenticated. **Example: Microsoft Entra ID**
.VERIFIED_EMAIL_CLAIM_NAME	String	The name of the claim that is used to verify the email address of the user.
.PREFERRED_GROUP_CLAIM_NAME	String	The key name within the OIDC token payload that holds information about the user's group memberships.
.OIDC_DISABLE_USER_ENDPOINT	Boolean	Whether to allow or disable the **/userinfo** endpoint. If using Azure Entra ID, this field must be set to **true** because Azure obtains the user's information from the token instead of calling the **/userinfo** endpoint. **Default: false**

3.28.1. OIDC configuration

The following example shows a sample OIDC configuration.

Example OIDC configuration

```
AUTHENTICATION_TYPE: OIDC
# ...
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_DISABLE_USER_ENDPOINT: true
    OIDC_ENDPOINT_CUSTOM_PARAMS":
            "authorization_endpoint":
               "some": "param",
# ...
```

3.29. NESTED REPOSITORIES CONFIGURATION FIELDS

Support for nested repository path names has been added under the **FEATURE_EXTENDED_REPOSITORY_NAMES** property. This optional configuration is added to the config.yaml by default. Enablement allows the use of / in repository names.

Table 3.34. OCI and nested repositories configuration fields

Field	Type	Description
FEATURE_EXTENDED_REPOSITORY_NAMES	Boolean	Enable support for nested repositories **Default:** True

OCI and nested repositories configuration example

```
FEATURE_EXTENDED_REPOSITORY_NAMES: true
```

3.30. QUAYINTEGRATION CONFIGURATION FIELDS

The following configuration fields are available for the QuayIntegration custom resource:

Name	Description	Schema
allowlistNamespaces (Optional)	A list of namespaces to include.	Array
clusterID (Required)	The ID associated with this cluster.	String

Name	Description	Schema
credentialsSecret.key (Required)	The secret containing credentials to communicate with the Quay registry.	Object
denylistNamespaces (Optional)	A list of namespaces to exclude.	Array
insecureRegistry (Optional)	Whether to skip TLS verification to the Quay registry	Boolean
quayHostname (Required)	The hostname of the Quay registry.	String
scheduledImageStreamImport (Optional)	Whether to enable image stream importing.	Boolean

3.31. MAIL CONFIGURATION FIELDS

Table 3.35. Mail configuration fields

Field	Type	Description
FEATURE_MAILING	Boolean	Whether emails are enabled **Default: False**
MAIL_DEFAULT_SENDER	String	If specified, the e-mail address used as the **from** when Red Hat Quay sends e-mails. If none, defaults to **support@quay.io** **Example: support@example.com**
MAIL_PASSWORD	String	The SMTP password to use when sending e-mails
MAIL_PORT	Number	The SMTP port to use. If not specified, defaults to 587.

Field	Type	Description
MAIL_SERVER	String	The SMTP server to use for sending e-mails. Only required if FEATURE_MAILING is set to true. Example: **smtp.example.com**
MAIL_USERNAME	String	The SMTP username to use when sending e-mails
MAIL_USE_TLS	Boolean	If specified, whether to use TLS for sending e-mails Default: **True**

3.32. USER CONFIGURATION FIELDS

Table 3.36. User configuration fields

Field	Type	Description
FEATURE_SUPER_USERS	Boolean	Whether superusers are supported Default: **true**
FEATURE_USER_CREATION	Boolean	Whether users can be created (by non-superusers) Default: **true**
FEATURE_USER_LAST_ACCESSED	Boolean	Whether to record the last time a user was accessed Default: **true**
FEATURE_USER_LOG_ACCESS	Boolean	If set to true, users will have access to audit logs for their namespace Default: **false**
FEATURE_USER_METADATA	Boolean	Whether to collect and support user metadata Default: **false**

Field	Type	Description
FEATURE_USERNAME_CONFIRMATION	Boolean	If set to true, users can confirm and modify their initial usernames when logging in via OpenID Connect (OIDC) or a non-database internal authentication provider like LDAP. **Default:** **true**
FEATURE_USER_RENAME	Boolean	If set to true, users can rename their own namespace **Default:** **false**
FEATURE_INVITE_ONLY_USER_CREATION	Boolean	Whether users being created must be invited by another user **Default:** **false**
FRESH_LOGIN_TIMEOUT	String	The time after which a fresh login requires users to re-enter their password **Example**: **5m**
USERFILES_LOCATION	String	ID of the storage engine in which to place user-uploaded files **Example**: **s3_us_east**
USERFILES_PATH	String	Path under storage in which to place user-uploaded files **Example**: **userfiles**
USER_RECOVERY_TOKEN_LIFETIME	String	The length of time a token for recovering a user accounts is valid **Pattern**: **^[0-9]+(w|m|d|h|s)$** **Default**: **30m**
FEATURE_SUPERUSERS_FULL_ACCESS	Boolean	Grants superusers the ability to read, write, and delete content from other repositories in namespaces that they do not own or have explicit permissions for. **Default:** **False**

Field	Type	Description
FEATURE_SUPERUSERS_ORG_CREATION_ONLY	Boolean	Whether to only allow superusers to create organizations. Default: **False**
FEATURE_RESTRICTED_USERS	Boolean	When set to **True** with **RESTRICTED_USERS_WHITELIST**: ● All normal users and superusers are restricted from creating organizations or content in their own namespace unless they are allowlisted via **RESTRICTED_USERS_WHITELIST**. ● Restricted users retain their normal permissions within organizations based on team memberships. Default: **False**
RESTRICTED_USERS_WHITELIST	String	When set with **FEATURE_RESTRICTED_USERS: true**, specific users are excluded from the **FEATURE_RESTRICTED_USERS** setting.
GLOBAL_READONLY_SUPER_USERS	String	When set, grants users of this list read access to all repositories, regardless of whether they are public repositories. Only works for those superusers defined with the **SUPER_USERS** configuration field.

3.32.1. User configuration fields references

Use the following references to update your **config.yaml** file with the desired configuration field.

3.32.1.1. FEATURE_SUPERUSERS_FULL_ACCESS configuration reference

```
---
SUPER_USERS:
- quayadmin
```

```
FEATURE_SUPERUSERS_FULL_ACCESS: True
---
```

3.32.1.2. GLOBAL_READONLY_SUPER_USERS configuration reference

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

3.32.1.3. FEATURE_RESTRICTED_USERS configuration reference

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

3.32.1.4. RESTRICTED_USERS_WHITELIST configuration reference

Prerequisites

- **FEATURE_RESTRICTED_USERS** is set to **true** in your **config.yaml** file.

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

NOTE

When this field is set, whitelisted users can create organizations, or read or write content from the repository even if **FEATURE_RESTRICTED_USERS** is set to **true**. Other users, for example, **user2**, **user3**, and **user4** are restricted from creating organizations, reading, or writing content

3.33. RECAPTCHA CONFIGURATION FIELDS

Table 3.37. Recaptcha configuration fields

Field	Type	Description

Field	Type	Description
FEATURE_RECAPTCHA	Boolean	Whether Recaptcha is necessary for user login and recovery **Default:** False
RECAPTCHA_SECRET_KEY	String	If recaptcha is enabled, the secret key for the Recaptcha service
RECAPTCHA_SITE_KEY	String	If recaptcha is enabled, the site key for the Recaptcha service

3.34. ACI CONFIGURATION FIELDS

Table 3.38. ACI configuration fields

Field	Type	Description
FEATURE_ACI_CONVERSION	Boolean	Whether to enable conversion to ACIs **Default:** False
GPG2_PRIVATE_KEY_FILENAME	String	The filename of the private key used to decrypte ACIs
GPG2_PRIVATE_KEY_NAME	String	The name of the private key used to sign ACIs
GPG2_PUBLIC_KEY_FILENAME	String	The filename of the public key used to encrypt ACIs

3.35. JWT CONFIGURATION FIELDS

Table 3.39. JWT configuration fields

Field	Type	Description
JWT_AUTH_ISSUER	String	The endpoint for JWT users **Pattern**: **^http(s)?://(.)+$** **Example**: **http://192.168.99.101:6060**
JWT_GETUSER_ENDPOINT	String	The endpoint for JWT users **Pattern**: **^http(s)?://(.)+$** **Example**: **http://192.168.99.101:6060**

Field	Type	Description
JWT_QUERY_ENDPOINT	String	The endpoint for JWT queries **Pattern**: **^http(s)?://(.)+$** **Example**: **http://192.168.99.101:6060**
JWT_VERIFY_ENDPOINT	String	The endpoint for JWT verification **Pattern**: **^http(s)?://(.)+$** **Example**: **http://192.168.99.101:6060**

3.36. APP TOKENS CONFIGURATION FIELDS

Table 3.40. App tokens configuration fields

Field	Type	Description
FEATURE_APP_SPECIFIC_TOKENS	Boolean	If enabled, users can create tokens for use by the Docker CLI **Default:** True
APP_SPECIFIC_TOKEN_EXPIRATION	String	The expiration for external app tokens. **Default** None **Pattern: ^[0-9]+(w|m|d|h|s)$**
EXPIRED_APP_SPECIFIC_TOKEN_GC	String	Duration of time expired external app tokens will remain before being garbage collected **Default:** 1d

3.37. MISCELLANEOUS CONFIGURATION FIELDS

Table 3.41. Miscellaneous configuration fields

Field	Type	Description

Field	Type	Description
ALLOW_PULLS_WITHOUT_STRICT_LOGGING	String	If true, pulls will still succeed even if the pull audit log entry cannot be written . This is useful if the database is in a read-only state and it is desired for pulls to continue during that time. **Default:** False
AVATAR_KIND	String	The types of avatars to display, either generated inline (local) or Gravatar (gravatar) **Values:** local, gravatar
BROWSER_API_CALLS_XHR_ONLY	Boolean	If enabled, only API calls marked as being made by an XHR will be allowed from browsers **Default:** True
DEFAULT_NAMESPACE_MAXIMUM_BUILD_COUNT	Number	The default maximum number of builds that can be queued in a namespace. **Default:** None
ENABLE_HEALTH_DEBUG_SECRET	String	If specified, a secret that can be given to health endpoints to see full debug info when not authenticated as a superuser
EXTERNAL_TLS_TERMINATION	Boolean	Set to **true** if TLS is supported, but terminated at a layer before Quay. Set to **false** when Quay is running with its own SSL certificates and receiving TLS traffic directly.
FRESH_LOGIN_TIMEOUT	String	The time after which a fresh login requires users to re-enter their password **Example:** **5m**

Field	Type	Description
HEALTH_CHECKER	String	The configured health check **Example:** **('RDSAwareHealthCheck', {'access_key': 'foo', 'secret_key': 'bar'})**
PROMETHEUS_NAMESPACE	String	The prefix applied to all exposed Prometheus metrics **Default:** **quay**
PUBLIC_NAMESPACES	Array of String	If a namespace is defined in the public namespace list, then it will appear on **all** users' repository list pages, regardless of whether the user is a member of the namespace. Typically, this is used by an enterprise customer in configuring a set of "well-known" namespaces.
REGISTRY_STATE	String	The state of the registry Values: **normal** or **read-only**
SEARCH_MAX_RESULT_PAGE_COUNT	Number	Maximum number of pages the user can paginate in search before they are limited **Default:** 10
SEARCH_RESULTS_PER_PAGE	Number	Number of results returned per page by search page **Default:** 10
V2_PAGINATION_SIZE	Number	The number of results returned per page in V2 registry APIs **Default:** 50
WEBHOOK_HOSTNAME_BLACKLIST	Array of String	The set of hostnames to disallow from webhooks when validating, beyond localhost

Field	Type	Description
CREATE_PRIVATE_REPO_ON_PUSH	Boolean	Whether new repositories created by push are set to private visibility **Default:** True
CREATE_NAMESPACE_ON_PUSH	Boolean	Whether new push to a non-existent organization creates it **Default:** False
NON_RATE_LIMITED_NAMESPACES	Array of String	If rate limiting has been enabled using **FEATURE_RATE_LIMITS**, you can override it for specific namespace that require unlimited access.
FEATURE_UI_V2	Boolean	When set, allows users to try the beta UI environment. **Default:** **True**
FEATURE_REQUIRE_TEAM_INVITE	Boolean	Whether to require invitations when adding a user to a team **Default:** True
FEATURE_REQUIRE_ENCRYPTED_BASIC_AUTH	Boolean	Whether non-encrypted passwords (as opposed to encrypted tokens) can be used for basic auth **Default:** False
FEATURE_RATE_LIMITS	Boolean	Whether to enable rate limits on API and registry endpoints. Setting FEATURE_RATE_LIMITS to **true** causes **nginx** to limit certain API calls to 30 per second. If that feature is not set, API calls are limited to 300 per second (effectively unlimited). **Default:** False

Field	Type	Description
FEATURE_FIPS	Boolean	If set to true, Red Hat Quay will run using FIPS-compliant hash functions **Default:** False
FEATURE_AGGREGATED_LOG_COUNT_RETRIEVAL	Boolean	Whether to allow retrieval of aggregated log counts **Default:** True
FEATURE_ANONYMOUS_ACCESS	Boolean	Whether to allow anonymous users to browse and pull public repositories **Default:** True
FEATURE_DIRECT_LOGIN	Boolean	Whether users can directly login to the UI **Default:** True
FEATURE_LIBRARY_SUPPORT	Boolean	Whether to allow for "namespace-less" repositories when pulling and pushing from Docker **Default:** True
FEATURE_PARTIAL_USER_AUTOCOMPLETE	Boolean	If set to true, autocompletion will apply to partial usernames+ **Default:** True
FEATURE_PERMANENT_SESSIONS	Boolean	Whether sessions are permanent **Default:** True
FEATURE_PUBLIC_CATALOG	Boolean	If set to true, the **_catalog** endpoint returns public repositories. Otherwise, only private repositories can be returned. **Default:** False

3.38. LEGACY CONFIGURATION FIELDS

The following fields are deprecated or obsolete.

Table 3.42. Legacy configuration fields

Field	Type	Description
FEATURE_BLACKLISTED_EMAILS	Boolean	If set to true, no new User accounts may be created if their email domain is blacklisted
BLACKLISTED_EMAIL_DOMAINS	Array of String	The list of email-address domains that is used if FEATURE_BLACKLISTED_EMAILS is set to true Example: **"example.com", "example.org"**
BLACKLIST_V2_SPEC	String	The Docker CLI versions to which Red Hat Quay will respond that V2 is **unsupported** Example: **<1.8.0** Default: **<1.6.0**
DOCUMENTATION_ROOT	String	Root URL for documentation links. This field is useful when Red Hat Quay is configured for disconnected environments to set an alternatively, or allowlisted, documentation link.
SECURITY_SCANNER_V4_NAMESPACE_WHITELIST	String	The namespaces for which the security scanner should be enabled
FEATURE_RESTRICTED_V1_PUSH	Boolean	If set to true, only namespaces listed in V1_PUSH_WHITELIST support V1 push Default: **True**
V1_PUSH_WHITELIST	Array of String	The array of namespace names that support V1 push if FEATURE_RESTRICTED_V1_PUSH is set to true
FEATURE_HELM_OCI_SUPPORT	Boolean	Enable support for Helm artifacts. Default: **False**

Field	Type	Description
ALLOWED_OCI_ARTIFACT_TYPES	Object	The set of allowed OCI artifact MIME types and the associated layer types.

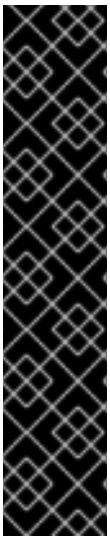
3.39. USER INTERFACE V2 CONFIGURATION FIELDS

Table 3.43. User interface v2 configuration fields

Field	Type	Description
FEATURE_UI_V2	Boolean	When set, allows users to try the beta UI environment. + **Default: False**
FEATURE_UI_V2_REPO_SETTINGS	Boolean	When set to **True**, enables repository settings in the Red Hat Quay v2 UI. + **Default: False**

3.39.1. v2 user interface configuration

With **FEATURE_UI_V2** enabled, you can toggle between the current version of the user interface and the new version of the user interface.



IMPORTANT

- This UI is currently in beta and subject to change. In its current state, users can only create, view, and delete organizations, repositories, and image tags.
- When running Red Hat Quay in the old UI, timed-out sessions would require that the user input their password again in the pop-up window. With the new UI, users are returned to the main page and required to input their username and password credentials. This is a known issue and will be fixed in a future version of the new UI.
- There is a discrepancy in how image manifest sizes are reported between the legacy UI and the new UI. In the legacy UI, image manifests were reported in mebibytes. In the new UI, Red Hat Quay uses the standard definition of megabyte (MB) to report image manifest sizes.

Procedure

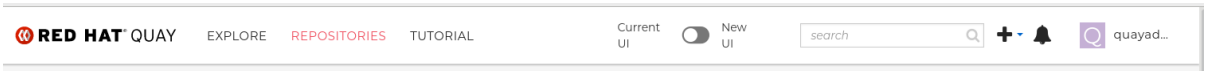
1. In your deployment's **config.yaml** file, add the **FEATURE_UI_V2** parameter and set it to **true**, for example:

```
---
FEATURE_TEAM_SYNCING: false
```

```
FEATURE_UI_V2: true
FEATURE_USER_CREATION: true
---
```

2. Log in to your Red Hat Quay deployment.

3. In the navigation pane of your Red Hat Quay deployment, you are given the option to toggle between **Current UI** and **New UI**. Click the toggle button to set it to new UI, and then click **Use Beta Environment**, for example:



3.40. IPV6 CONFIGURATION FIELD

Table 3.44. IPv6 configuration field

Field	Type	Description
FEATURE_LISTEN_IP_VERSION	String	Enables IPv4, IPv6, or dual-stack protocol family. This configuration field must be properly set, otherwise Red Hat Quay fails to start. **Default: IPv4** **Additional configurations: IPv6, dual-stack**

3.41. BRANDING CONFIGURATION FIELDS

Table 3.45. Branding configuration fields

Field	Type	Description
BRANDING	Object	Custom branding for logos and URLs in the Red Hat Quay UI.
.logo (Required)	String	Main logo image URL. The header logo defaults to 205x30 PX. The form logo on the Red Hat Quay sign in screen of the web UI defaults to 356.5x39.7 PX. **Example: /static/img/quay-horizontal-color.svg**

Field	Type	Description
.footer_img	String	Logo for UI footer. Defaults to 144x34 PX. **Example:** **/static/img/RedHat.svg**
.footer_url	String	Link for footer image. **Example:** **https://redhat.com**

3.41.1. Example configuration for Red Hat Quay branding

Branding config.yaml example

```
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. SESSION TIMEOUT CONFIGURATION FIELD

The following configuration field relies on on the Flask API configuration field of the same name.

Table 3.46. Session logout configuration field

Field	Type	Description
PERMANENT_SESSION_LIFETIME	Integer	A **timedelta** which is used to set the expiration date of a permanent session. The default is 31 days, which makes a permanent session survive for roughly one month. **Default: 2678400**

3.42.1. Example session timeout configuration

The following YAML is the suggest configuration when enabling session lifetime.

IMPORTANT

Altering session lifetime is not recommended. Administrators should be aware of the allotted time when setting a session timeout. If you set the time too early, it might interrupt your workflow.

Session timeout YAML configuration

```
PERMANENT_SESSION_LIFETIME: 3000
```

CHAPTER 4. ENVIRONMENT VARIABLES

Red Hat Quay supports a limited number of environment variables for dynamic configuration.

4.1. GEO-REPLICATION

The same configuration should be used across all regions, with exception of the storage backend, which can be configured explicitly using the **QUAY_DISTRIBUTED_STORAGE_PREFERENCE** environment variable.

Table 4.1. Geo-replication configuration

Variable	Type	Description
QUAY_DISTRIBUTED_STORAGE_PREFERENCE	String	The preferred storage engine (by ID in DISTRIBUTED_STORAGE_CONFIG) to use.

4.2. DATABASE CONNECTION POOLING

Red Hat Quay is composed of many different processes which all run within the same container. Many of these processes interact with the database.

If enabled, each process that interacts with the database will contain a connection pool. These per-process connection pools are configured to maintain a maximum of 20 connections. Under heavy load, it is possible to fill the connection pool for every process within a Red Hat Quay container. Under certain deployments and loads, this might require analysis to ensure that Red Hat Quay does not exceed the configured database's maximum connection count.

Overtime, the connection pools will release idle connections. To release all connections immediately, Red Hat Quay requires a restart.

Database connection pooling can be toggled by setting the environment variable **DB_CONNECTION_POOLING** to **true** or **false**.

Table 4.2. Database connection pooling configuration

Variable	Type	Description
DB_CONNECTION_POOLING	Boolean	Enable or disable database connection pooling

If database connection pooling is enabled, it is possible to change the maximum size of the connection pool. This can be done through the following **config.yaml** option:

config.yaml

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

4.3. HTTP CONNECTION COUNTS

It is possible to specify the quantity of simultaneous HTTP connections using environment variables. These can be specified as a whole, or for a specific component. The default for each is **50** parallel connections per process.

Table 4.3. HTTP connection counts configuration

Variable	Type	Description
WORKER_CONNECTION_COUNT	Number	Simultaneous HTTP connections **Default:** 50
WORKER_CONNECTION_COUNT_REGISTRY	Number	Simultaneous HTTP connections for registry **Default:** WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_WEB	Number	Simultaneous HTTP connections for web UI **Default:** WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_SECSCAN	Number	Simultaneous HTTP connections for Clair **Default:** WORKER_CONNECTION_COUNT

4.4. WORKER COUNT VARIABLES

Table 4.4. Worker count variables

Variable	Type	Description
WORKER_COUNT	Number	Generic override for number of processes
WORKER_COUNT_REGISTRY	Number	Specifies the number of processes to handle Registry requests within the **Quay** container **Values:** Integer between **8** and **64**

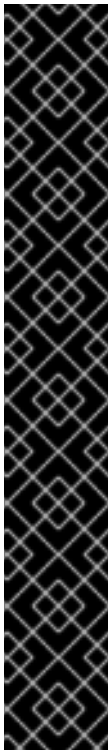
Variable	Type	Description
WORKER_COUNT_WEB	Number	Specifies the number of processes to handle UI/Web requests within the container **Values:** Integer between **2** and **32**
WORKER_COUNT_SECSCAN	Number	Specifies the number of processes to handle Security Scanning (e.g. Clair) integration within the container **Values:** Integer. Because the Operator specifies 2 vCPUs for resource requests and limits, setting this value between **2** and **4** is safe. However, users can run more, for example, **16**, if warranted.

4.5. DEBUG VARIABLES

The following debug variables are available on Red Hat Quay.

Table 4.5. Debug configuration variables

Variable	Type	Description
DEBUGLOG	Boolean	Whether to enable or disable debug logs.

Variable	Type	Description
USERS_DEBUG	Integer. Either **0** or **1**.	Used to debug LDAP operations in clear text, including passwords. Must be used with **DEBUGLOG=TRUE**.  **IMPORTANT** Setting **USERS_DEBUG=1** exposes credentials in clear text. This variable should be removed from the Red Hat Quay deployment after debugging. The log file that is generated with this environment variable should be scrutinized, and passwords should be removed before sending to other users. Use with caution.

CHAPTER 5. CLAIR SECURITY SCANNER

5.1. CLAIR CONFIGURATION OVERVIEW

Clair is configured by a structured YAML file. Each Clair node needs to specify what mode it will run in and a path to a configuration file through CLI flags or environment variables. For example:

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

or

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

The aforementioned commands each start two Clair nodes using the same configuration file. One runs the indexing facilities, while other runs the matching facilities.

If you are running Clair in **combo** mode, you must supply the indexer, matcher, and notifier configuration blocks in the configuration.

5.1.1. Information about using Clair in a proxy environment

Environment variables respected by the Go standard library can be specified if needed, for example:

- **HTTP_PROXY**

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

- **HTTPS_PROXY**.

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

- **SSL_CERT_DIR**

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

- **NO_PROXY**

  ```
  $ export NO_PROXY=<comma_separated_list_of_hosts_and_domains>
  ```

If you are using a proxy server in your environment with Clair's updater URLs, you must identify which URL needs to be added to the proxy allowlist to ensure that Clair can access them unimpeded. For example, the **osv** updater requires access to **https://osv-vulnerabilities.storage.googleapis.com** to fetch ecosystem data dumps. In this scenario, the URL must be added to the proxy allowlist. For a full list of updater URLs, see "Clair updater URLs".

You must also ensure that the standard Clair URLs are added to the proxy allowlist:

- **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**

When configuring the proxy server, take into account any authentication requirements or specific proxy settings needed to enable seamless communication between Clair and these URLs. By thoroughly documenting and addressing these considerations, you can ensure that Clair functions effectively while routing its updater traffic through the proxy.

5.1.2. Clair configuration reference

The following YAML shows an example Clair configuration:

```
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: ""
```

NOTE

The above YAML file lists every key for completeness. Using this configuration file as-is will result in some options not having their defaults set normally.

5.1.3. Clair general fields

The following table describes the general configuration fields available for a Clair deployment.

Field	Typhttp_listen_ae	Description
http_listen_addr	String	Configures where the HTTP API is exposed. Default: **:6060**
introspection_addr	String	Configures where Clair's metrics and health endpoints are exposed.
log_level	String	Sets the logging level. Requires one of the following strings: **debug-color**, **debug**, **info**, **warn**, **error**, **fatal**, **panic**
tls	String	A map containing the configuration for serving the HTTP API of TLS/SSL and HTTP/2.
.cert	String	The TLS certificate to be used. Must be a full-chain certificate.

Example configuration for general Clair fields
The following example shows a Clair configuration.

Example configuration for general Clair fields

```
# ...
```

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

5.1.4. Clair indexer configuration fields

The following table describes the configuration fields for Clair's **indexer** component.

Field	Type	Description
indexer	Object	Provides Clair indexer node configuration.
.airgap	Boolean	Disables HTTP access to the internet for indexers and fetchers. Private IPv4 and IPv6 addresses are allowed. Database connections are unaffected.
.connstring	String	A Postgres connection string. Accepts format as a URL or libpq connection string.
.index_report_request_concurrency	Integer	Rate limits the number of index report creation requests. Setting this to **0** attemps to auto-size this value. Setting a negative value means unlimited. The auto-sizing is a multiple of the number of available cores. The API returns a **429** status code if concurrency is exceeded.
.scanlock_retry	Integer	A positive integer representing seconds. Concurrent indexers lock on manifest scans to avoid clobbering. This value tunes how often a waiting indexer polls for the lock.
.layer_scan_concurrency	Integer	Positive integer limiting the number of concurrent layer scans. Indexers will match a manifest's layer concurrently. This value tunes the number of layers an indexer scans in parallel.
.migrations	Boolean	Whether indexer nodes handle migrations to their database.

Field	Type	Description
.scanner	String	Indexer configuration. Scanner allows for passing configuration options to layer scanners. The scanner will have this configuration pass to it on construction if designed to do so.
.scanner.dist	String	A map with the name of a particular scanner and arbitrary YAML as a value.
.scanner.package	String	A map with the name of a particular scanner and arbitrary YAML as a value.
.scanner.repo	String	A map with the name of a particular scanner and arbitrary YAML as a value.

Example indexer configuration
The following example shows a hypothetical indexer configuration for Clair.

Example indexer configuration

```
# ...
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 configuration fields

The following table describes the configuration fields for Clair's **matcher** component.

NOTE

Differs from **matchers** configuration fields.

Field	Type	Description
matcher	Object	Provides Clair matcher node configuration.

Field	Type	Description
.cache_age	String	Controls how long users should be hinted to cache responses for.
.connstring	String	A Postgres connection string. Accepts format as a URL or libpq connection string.
.max_conn_pool	Integer	Limits the database connection pool size. Clair allows for a custom connection pool size. This number directly sets how many active database connections are allowed concurrently. This parameter will be ignored in a future version. Users should configure this through the connection string.
.indexer_addr	String	A matcher contacts an indexer to create a vulnerability report. The location of this indexer is required. Defaults to **30m**.
.migrations	Boolean	Whether matcher nodes handle migrations to their databases.
.period	String	Determines how often updates for new security advisories take place. Defaults to **30m**.
.disable_updaters	Boolean	Whether to run background updates or not. Default: **False**

Field	Type	Description
.update_retention	Integer	Sets the number of update operations to retain between garbage collection cycles. This should be set to a safe MAX value based on database size constraints. Defaults to **10m**. If a value of less than **0** is provided, garbage collection is disabled. **2** is the minimum value to ensure updates can be compared to notifications.

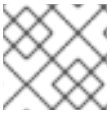
Example matcher configuration

Example matcher configuration

```
# ...
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 configuration fields

The following table describes the configuration fields for Clair's **matchers** component.



NOTE

Differs from **matcher** configuration fields.

Table 5.1. Matchers configuration fields

Field	Type	Description
matchers	Array of strings	Provides configuration for the in-tree **matchers**.

Field	Type	Description
.names	String	A list of string values informing the matcher factory about enabled matchers. If value is set to **null**, the default list of matchers run. The following strings are accepted: **alpine-matcher**, **aws-matcher**, **debian-matcher**, **gobin**, **java-maven**, **oracle**, **photon**, **python**, **rhel**, **rhel-container-matcher**, **ruby**, **suse**, **ubuntu-matcher**
.config	String	Provides configuration to a specific matcher. A map keyed by the name of the matcher containing a sub-object which will be provided to the matchers factory constructor. For example:

Example matchers configuration
The following example shows a hypothetical Clair deployment that only requires only the **alpine**, **aws**, **debian**, **oracle** matchers.

Example matchers configuration

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

5.1.7. Clair updaters configuration fields

The following table describes the configuration fields for Clair's **updaters** component.

Table 5.2. Updaters configuration fields

Field	Type	Description
updaters	Object	Provides configuration for the matcher's update manager.

Field	Type	Description
.sets	String	A list of values informing the update manager which updaters to run. If value is set to **null**, the default set of updaters runs the following: **alpine**, **aws**, **clair.cvss**, **debian**, **oracle**, **photon**, **osv**, **rhel**, **rhcc suse**, **ubuntu** If left blank, zero updaters run.
.config	String	Provides configuration to specific updater sets. A map keyed by the name of the updater set containing a sub-object which will be provided to the updater set's constructor. For a list of the sub-objects for each updater, see "Advanced updater configuration".

Example updaters configuration
In the following configuration, only the **rhel** set is configured. The **ignore_unpatched** variable, which is specific to the **rhel** updater, is also defined.

Example updaters configuration

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

5.1.8. Clair notifier configuration fields

The general notifier configuration fields for Clair are listed below.

Field	Type	Description
notifier	Object	Provides Clair notifier node configuration.
.connstring	String	Postgres connection string. Accepts format as URL, or libpq connection string.

Field	Type	Description
.migrations	Boolean	Whether notifier nodes handle migrations to their database.
.indexer_addr	String	A notifier contacts an indexer to create or obtain manifests affected by vulnerabilities. The location of this indexer is required.
.matcher_addr	String	A notifier contacts a matcher to list update operations and acquire diffs. The location of this matcher is required.
.poll_interval	String	The frequency at which the notifier will query a matcher for update operations.
.delivery_interval	String	The frequency at which the notifier attempts delivery of created, or previously failed, notifications.
.disable_summary	Boolean	Controls whether notifications should be summarized to one per manifest.

Example notifier configuration
The following **notifier** snippet is for a minimal configuration.

Example notifier configuration

```
# ...
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 configuration fields

The following webhook fields are available for the Clair notifier environment.

Table 5.3. Clair webhook fields

.webhook	Object	Configures the notifier for webhook delivery.
.webhook.target	String	URL where the webhook will be delivered.
.webhook.callback	String	The callback URL where notifications can be retrieved. The notification ID will be appended to this URL. This will typically be where the Clair notifier is hosted.
.webhook.headers	String	A map associating a header name to a list of values.

Example webhook configuration

Example webhook configuration

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

5.1.8.2. Clair amqp configuration fields

The following Advanced Message Queuing Protocol (AMQP) fields are available for the Clair notifier environment.

.amqp	Object	Configures the notifier for AMQP delivery. [NOTE] ==== Clair does not declare any AMQP components on its own. All attempts to use an exchange or queue are passive only and will fail. Broker administrators should setup exchanges and queues ahead of time. ====

.amqp.direct	Boolean	If **true**, the notifier will deliver individual notifications (not a callback) to the configured AMQP broker.
.amqp.rollup	Integer	When **amqp.direct** is set to **true**, this value informs the notifier of how many notifications to send in a direct delivery. For example, if **direct** is set to **true**, and **amqp.rollup** is set to **5**, the notifier delivers no more than 5 notifications in a single JSON payload to the broker. Setting the value to **0** effectively sets it to **1**.
.amqp.exchange	Object	The AMQP exchange to connect to.
.amqp.exchange.name	String	The name of the exchange to connect to.
.amqp.exchange.type	String	The type of the exchange. Typically one of the following: **direct**, **fanout**, **topic**, **headers**.
.amqp.exchange.durability	Boolean	Whether the configured queue is durable.
.amqp.exchange.auto_delete	Boolean	Whether the configured queue uses an **auto_delete_policy**.
.amqp.routing_key	String	The name of the routing key each notification is sent with.
.amqp.callback	String	If **amqp.direct** is set to **false**, this URL is provided in the notification callback sent to the broker. This URL should point to Clair's notification API endpoint.
.amqp.uris	String	A list of one or more AMQP brokers to connect to, in priority order.
.amqp.tls	Object	Configures TLS/SSL connection to an AMQP broker.

.amqp.tls.root_ca	String	The filesystem path where a root CA can be read.
.amqp.tls.cert	String	The filesystem path where a TLS/SSL certificate can be read. [NOTE] ==== Clair also allows **SSL_CERT_DIR**, as documented for the Go **crypto/x509** package. ====
.amqp.tls.key	String	The filesystem path where a TLS/SSL private key can be read.

Example AMQP configuration
The following example shows a hypothetical AMQP configuration for Clair.

Example AMQP configuration

```
# ...
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 configuration fields

The following Simple Text Oriented Message Protocol (STOMP) fields are available for the Clair notifier environment.

.stomp	Object	Configures the notifier for STOMP delivery.
.stomp.direct	Boolean	If **true**, the notifier delivers individual notifications (not a callback) to the configured STOMP broker.

.stomp	Object	Configures the notifier for STOMP delivery.
.stomp.rollup	Integer	If **stomp.direct** is set to **true**, this value limits the number of notifications sent in a single direct delivery. For example, if **direct** is set to **true**, and **rollup** is set to **5**, the notifier delivers no more than 5 notifications in a single JSON payload to the broker. Setting the value to **0** effectively sets it to **1**.
.stomp.callback	String	If **stomp.callback** is set to **false**, the provided URL in the notification callback is sent to the broker. This URL should point to Clair's notification API endpoint.
.stomp.destination	String	The STOMP destination to deliver notifications to.
.stomp.uris	String	A list of one or more STOMP brokers to connect to in priority order.
.stomp.tls	Object	Configured TLS/SSL connection to STOMP broker.
.stomp.tls.root_ca	String	The filesystem path where a root CA can be read. [NOTE] ==== Clair also respects **SSL_CERT_DIR**, as documented for the Go **crypto/x509** package. ====
.stomp.tls.cert	String	The filesystem path where a TLS/SSL certificate can be read.
.stomp.tls.key	String	The filesystem path where a TLS/SSL private key can be read.
.stomp.user	String	Configures login details for the STOMP broker.
.stomp.user.login	String	The STOMP login to connect with.

.stomp	Object	Configures the notifier for STOMP delivery.
.stomp.user.passcode	String	The STOMP passcode to connect with.

Example STOMP configuration
The following example shows a hypothetical STOMP configuration for Clair.

Example STOMP configuration

```
# ...
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 authorization configuration fields

The following authorization configuration fields are available for Clair.

Field	Type	Description
auth	Object	Defines Clair's external and intra-service JWT based authentication. If multiple **auth** mechanisms are defined, Clair picks one. Currently, multiple mechanisms are unsupported.
.psk	String	Defines pre-shared key authentication.
.psk.key	String	A shared base64 encoded key distributed between all parties signing and verifying JWTs.
.psk.iss	String	A list of JWT issuers to verify. An empty list accepts any issuer in a JWT claim.

Example authorization configuration
The following **authorization** snippet is for a minimal configuration.

Example authorization configuration

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

5.1.10. Clair trace configuration fields

The following trace configuration fields are available for Clair.

Field	Type	Description
trace	Object	Defines distributed tracing configuration based on OpenTelemetry.
.name	String	The name of the application traces will belong to.
.probability	Integer	The probability a trace will occur.
.jaeger	Object	Defines values for Jaeger tracing.
.jaeger.agent	Object	Defines values for configuring delivery to a Jaeger agent.
.jaeger.agent.endpoint	String	An address in the **<host>:<post>** syntax where traces can be submitted.
.jaeger.collector	Object	Defines values for configuring delivery to a Jaeger collector.
.jaeger.collector.endpoint	String	An address in the **<host>:<post>** syntax where traces can be submitted.
.jaeger.collector.username	String	A Jaeger username.
.jaeger.collector.password	String	A Jaeger password.
.jaeger.service_name	String	The service name registered in Jaeger.

Field	Type	Description
.jaeger.tags	String	Key-value pairs to provide additional metadata.
.jaeger.buffer_max	Integer	The maximum number of spans that can be buffered in memory before they are sent to the Jaeger backend for storage and analysis.

Example trace configuration
The following example shows a hypothetical trace configuration for Clair.

Example trace configuration

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

5.1.11. Clair metrics configuration fields

The following metrics configuration fields are available for Clair.

Field	Type	Description
metrics	Object	Defines distributed tracing configuration based on OpenTelemetry.
.name	String	The name of the metrics in use.
.prometheus	String	Configuration for a Prometheus metrics exporter.
.prometheus.endpoint	String	Defines the path where metrics are served.

Example metrics configuration
The following example shows a hypothetical metrics configuration for Clair.

Example metrics configuration

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