



Red Hat Quay 3.11

配置 Red Hat Quay

使用配置选项自定义 Red Hat Quay

使用配置选项自定义 Red Hat Quay

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

摘要

配置 Red Hat Quay

目录

第 1 章 RED HAT QUAY 配置入门	4
第 2 章 RED HAT QUAY 配置声明器	5
2.1. RED HAT QUAY 3.11 的配置更新	5
2.2. 编辑配置文件	6
2.3. 独立部署中配置文件的位置	6
2.4. 最小配置	6
第 3 章 配置字段	9
3.1. 所需的配置字段	9
3.2. 自动化选项	9
3.3. 可选的配置字段	9
3.4. 常规必填字段	10
3.5. 数据库配置	11
3.6. 镜像存储	13
3.7. REDIS 配置字段	18
3.8. MODELCACHE 配置选项	21
3.9. 标签过期配置字段	22
3.10. 配额管理配置字段	22
3.11. 代理缓存配置字段	23
3.12. 机器人帐户配置字段	24
3.13. 预配置 RED HAT QUAY 用于自动化	24
3.14. 基本配置字段	29
3.15. SSL 配置字段	30
3.16. 将 TLS 证书添加到 RED HAT QUAY CONTAINER	33
3.17. LDAP 配置字段	34
3.18. 镜像配置字段	38
3.19. 安全扫描程序配置字段	39
3.20. HELM 配置字段	41
3.21. 开放容器项目配置字段	42
3.22. 未知介质类型	42
3.23. 操作日志配置字段	43
3.24. 构建日志配置字段	46
3.25. DOCKERFILE 构建触发器字段	47
3.26. 构建管理器配置字段	49
3.27. OAUTH 配置字段	51
3.28. OIDC 配置字段	53
3.29. 嵌套存储库配置字段	55
3.30. QUAYINTEGRATION 配置字段	56
3.31. 邮件配置字段	56
3.32. 用户配置字段	57
3.33. RECAPTCHA 配置字段	60
3.34. ACI 配置字段	61
3.35. JWT 配置字段	61
3.36. 应用程序令牌配置字段	62
3.37. 其它配置字段	62
3.38. 旧配置字段	66
3.39. 用户界面 V2 配置字段	66
3.40. IPV6 配置字段	68
3.41. 品牌配置字段	68
3.42. 会话超时配置字段	69

第 4 章 环境变量	71
4.1. GEO-REPLICATION	71
4.2. 数据库连接池	71
4.3. HTTP 连接数	72
4.4. WORKER 数量变量	72
4.5. 调试变量	73
第 5 章 CLAIR 安全扫描程序	75
5.1. CLAIR 配置概述	75

第 1 章 RED HAT QUAY 配置入门

Red Hat Quay 可以通过独立、独立配置或使用 OpenShift Container Platform 上的 Red Hat Quay Operator 部署。

如何创建、检索、更新和删除 Red Hat Quay 配置会因您使用的部署类型而异。但是，对于任一部署类型，核心配置选项都相同。核心配置主要通过 **config.yaml** 文件设置，但也可以使用配置 API 进行设置。

对于 Red Hat Quay 的独立部署，您必须在启动 registry 前提供最低所需的配置参数。启动 Red Hat Quay registry 的最低要求可在 "Retrieving the current configuration" 部分找到。

如果使用 Red Hat Quay Operator 在 OpenShift Container Platform 上安装 Red Hat Quay，则不需要提供配置参数，因为 Red Hat Quay Operator 提供了用于部署 registry 的默认信息。

使用所需配置部署 Red Hat Quay 后，您应该从部署中检索并保存。完整配置包含重启或升级系统时可能需要的额外生成的值。

第 2 章 RED HAT QUAY 配置声明器

对于基于独立和基于 Operator 的 Red Hat Quay 部署，某些功能和配置参数都不会被主动使用或实施。因此，功能标记（如启用或禁用某些功能）以及没有明确记录或要求的配置参数。红帽支持文档应小心进行修改。未使用的功能或参数可能无法完全测试、支持或与 Red Hat Quay 兼容。修改未使用的功能参数可能会导致您的部署出现意外问题或中断。

有关在独立部署中配置 Red Hat Quay 的详情，请参考 [高级 Red Hat Quay 配置](#)

有关配置 Red Hat Quay Operator 部署的详情，请参考在 [OpenShift Container Platform 上配置 Red Hat Quay](#)

2.1. RED HAT QUAY 3.11 的配置更新

以下小节详细介绍了 Red Hat Quay 3.11 中添加的新配置字段。

2.1.1. 团队同步配置字段

通过 OIDC 功能为团队同步添加了以下配置字段：

表 2.1. 团队同步配置字段

字段	类型	描述
PREFERRED_GROUP_CLAIM_NAME	字符串	OIDC 令牌有效负载中的密钥名称，其中包含有关用户组成员资格的信息。

团队同步示例 YAML 配置

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

2.1.2. AWS S3 STS 部署的配置字段

为 Red Hat Quay 配置 AWS STS 时添加了以下配置字段。这些字段用于为您的部署配置 AWS S3 存储。

表 2.2. AWS S3 STS 配置字段

字段	类型	描述
.sts_role_arn	字符串	为 Red Hat Quay 配置 AWS STS 时需要的唯一 Amazon 资源名称 (ARN)。
.sts_user_access_key	字符串	为 Red Hat Quay 配置 AWS STS 时生成的 AWS S3 用户访问密钥。

<code>.sts_user_secret_key</code>	字符串	为 Red Hat Quay 配置 AWS STS 时生成的 AWS S3 用户 secret 密钥。
-----------------------------------	-----	---

AWS S3 STS 示例 YAML 配置

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

2.2. 编辑配置文件

要部署独立的 Red Hat Quay 实例，您必须提供最少的配置信息。最小配置的要求可在 "Red Hat Quay minimal configuration" 中找到。

提供必填字段后，您可以验证您的配置。如果有任何问题，将突出显示它们。



注意

可以使用配置 API 来验证配置，但这要求以配置模式启动 **Quay** 容器。

要在本地部署配置工具，[请参阅开始使用 Red Hat Quay](#)，并按照"配置 Red Hat Quay"的说明进行操作。

要使更改生效，必须重启 registry。

2.3. 独立部署中配置文件的位置

对于独立的 Red Hat Quay 部署，启动 Red Hat Quay registry 时必须指定 **config.yaml** 文件。此文件位于配置卷中。例如，在以下命令部署 Red Hat Quay 时，配置文件位于 **\$QUAY/config/config.yaml**：

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -v $QUAY/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1
```

2.4. 最小配置

独立部署 Red Hat Quay 需要以下配置选项：

- 服务器主机名
- HTTP 或 HTTPS

- 身份验证类型，例如数据库或轻量级目录访问协议(LDAP)
- 用于加密数据的 secret 密钥
- 镜像存储
- 元数据的数据库
- 用于构建日志和用户事件的 Redis
- 标签过期选项

2.4.1. 最小配置文件示例

以下示例显示了使用本地存储进行镜像的最小配置文件示例：

```
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. 本地存储

只有在部署 registry 进行 *概念验证* 时，才建议将本地存储用于镜像。

在配置本地存储时，启动 registry 时会在命令行中指定存储。

以下命令将本地目录 `$QUAY/storage` 映射到容器中的 `datastorage` 路径：

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  -v $QUAY/storage:/datastorage:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1
```

2.4.3. 云存储

存储配置在[镜像存储](#)部分中详细介绍。对于某些用户，比较 Google Cloud Platform 和本地存储配置之间的区别可能很有用。例如，以下 YAML 提供了 Google Cloud Platform 存储配置：

`$QUAY/config/config.yaml`

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

当使用云存储启动 registry 时，命令行不需要配置。例如：

```
$ sudo podman run -d --rm -p 80:8080 -p 443:8443 \
  --name=quay \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.11.1
```

第 3 章 配置字段

本节介绍了部署 Red Hat Quay 时必需和可选配置字段。

3.1. 所需的配置字段

配置 Red Hat Quay 所需的字段包括在以下部分：

- [常规必填字段](#)
- [镜像存储](#)
- [元数据的数据库](#)
- [用于构建日志和用户事件的 Redis](#)
- [标签过期选项](#)

3.2. 自动化选项

以下小节描述了 Red Hat Quay 部署的可用自动化选项：

- [预配置 Red Hat Quay 用于自动化](#)
- [使用 API 创建第一个用户](#)

3.3. 可选的配置字段

Red Hat Quay 的可选字段可在以下部分中找到：

- [基本配置](#)
- [SSL](#)
- [LDAP](#)
- [存储库镜像](#)
- [配额管理](#)
- [安全扫描程序](#)
- [Helm](#)
- [操作日志](#)
- [构建日志](#)
- [Dockerfile 构建](#)
- [OAuth](#)
- [配置嵌套软件仓库](#)
- [在 Quay 中添加其他 OCI 介质类型](#)

- [Mail](#)
- [User](#)
- [reCAPTCHA](#)
- [ACI](#)
- [JWT](#)
- [应用程序令牌](#)
- [其它](#)
- [用户界面 v2](#)
- [IPv6 配置字段](#)
- [旧选项](#)

3.4. 常规必填字段

下表描述了 Red Hat Quay 部署所需的配置字段：

表 3.1. 常规必填字段

字段	类型	描述
AUTHENTICATION_TYPE (Required)	字符串	用于凭证身份验证的身份验证引擎。 值： 其中一个 数据库 、 LDAP 、 JWT 、 Keystone 、 OIDC 默认： 数据库
PREFERRED_URL_SCHEME (Required)	字符串	访问 Red Hat Quay 时使用的 URL 方案。 值： 以下之一 http 、 https Default: http
SERVER_HOSTNAME (Required)	字符串	可以访问 Red Hat Quay 的 URL，不包括网络协议。 例如： quay-server.example.com

字段	类型	描述
DATABASE_SECRET_KEY (Required)	字符串	用于加密数据库中敏感字段的密钥。设定后永远不会更改这个值，否则所有依赖字段（如存储库镜像用户名和密码配置）都无效。这个值由 Red Hat Quay Operator for Operator 部署自动设置。对于独立部署，管理员可以使用 Open SSL 或类似的工具提供自己的密钥。密钥长度不应超过 63 个字符。
SECRET_KEY (Required)	字符串	用于加密会话 Cookie 和用于正确解释用户会话所需的 CSRF 令牌的密钥。设置时不应更改该值。应该在所有 Red Hat Quay 实例中保留。如果没有在所有实例间持久保留，则可能会出现与会话持久性相关的登录失败和其他错误。
SETUP_COMPLETE (Required)	布尔值	这是软件早期版本中保留的工件，目前 必须使用 true 值指定。

3.5. 数据库配置

本节论述了 Red Hat Quay 部署可用的数据库配置字段。

3.5.1. 数据库 URI

使用 Red Hat Quay 时，使用所需的 **DB_URI** 字段配置与数据库的连接。

下表描述了 **DB_URI** 配置字段：

表 3.2. 数据库 URI

字段	类型	描述
DB_URI (必需)	字符串	用于访问数据库的 URI，包括任何凭据。 DB_URI 字段示例： <code>postgresql://quayuser:quaypass@quay-server.example.com:5432/quay</code>

3.5.2. 数据库连接参数

可选的连接参数由 **DB_CONNECTION_ARGS** 参数配置。**DB_CONNECTION_ARGS** 下定义的部分键值对是通用的，另一些则特定于数据库。

下表描述了数据库连接参数：

表 3.3. 数据库连接参数

字段	类型	描述
DB_CONNECTION_ARGS	对象	数据库的可选连接参数，如超时和 SSL/TLS。
.autorollback	布尔值	是否使用线程本地连接。应该始终为 true
.threadlocals	布尔值	是否使用自动滚动连接。应该始终为 true

3.5.2.1. PostgreSQL SSL/TLS 连接参数

使用 SSL/TLS 时，配置取决于您要部署的数据库。以下示例显示了 PostgreSQL SSL/TLS 配置：

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

sslmode 选项决定是否或具有什么优先级，安全 SSL/TLS TCP/IP 连接将与服务器协商。有六个模式：

表 3.4. SSL/TLS 选项

模式	描述
disable	您的配置只尝试非 SSL/TLS 连接。
allow	您的配置首先尝试非 SSL/TLS 连接。失败时，尝试 SSL/TLS 连接。
首选 (默认)	您的配置首先尝试 SSL/TLS 连接。失败时，尝试非 SSL/TLS 连接。
require	您的配置只尝试 SSL/TLS 连接。如果存在 root CA 文件，它会以与指定 verify-ca 相同的方式验证证书。
verify-ca	您的配置只尝试 SSL/TLS 连接，并验证服务器证书是否由可信证书颁发机构(CA)发布。
verify-full	只尝试 SSL/TLS 连接，并验证服务器证书是否由可信 CA 发布，并且请求的服务器主机名是否与证书中的匹配。

有关 PostgreSQL 的有效参数的更多信息，请参阅 [Database Connection Control Functions](#)。

3.5.2.2. MySQL SSL/TLS 连接参数

以下示例显示了 MySQL SSL/TLS 配置示例：

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

有关 MySQL 的有效连接参数的信息，请访问 [使用类似URI的字符串或键-值对连接到服务器](#)。

3.6. 镜像存储

本节详细介绍了 Red Hat Quay 提供的镜像存储功能和配置字段。

3.6.1. 镜像存储功能

下表描述了 Red Hat Quay 的镜像存储功能：

表 3.5. 存储配置功能

字段	类型	描述
FEATURE_REPO_MIRROR	布尔值	如果设置为 true，请启用存储库镜像。 默认：false
FEATURE_PROXY_STORAGE	布尔值	是否通过 NGINX 代理存储中的所有直接下载 URL。 默认：false
FEATURE_STORAGE_REPLICATION	布尔值	是否在存储引擎之间自动复制。 默认：false

3.6.2. 镜像存储配置字段

下表描述了 Red Hat Quay 的镜像存储配置字段：

表 3.6. 存储配置字段

字段	类型	描述
DISTRIBUTED_STORAGE_CONFIG (Required)	对象	在 Red Hat Quay 中使用的存储引擎的配置。每个键代表存储引擎的唯一标识符。该值由一个元组 (key, value) 组成，组成一个描述存储引擎参数的对象。 Default: []

字段	类型	描述
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS (Required)	字符串数组	存储引擎列表（按 DISTRIBUTED_STORAGE_CONFIG 的 ID），其镜像应完全复制到所有其他存储引擎。
DISTRIBUTED_STORAGE_PREFERENCE (Required)	字符串数组	首选存储引擎（按 DISTRIBUTED_STORAGE_CONFIG 中的 ID）。首选引擎意味着首先检查拉取和镜像被推送到它。 默认：false
MAXIMUM_LAYER_SIZE	字符串	镜像层允许的最大值。 Pattern: ^[0-9]+(G M)\$ 示例:100G Default: 20G

3.6.3. 本地存储

以下 YAML 显示了使用本地存储配置示例：

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

3.6.4. OpenShift Container Storage/NooBaa

以下 YAML 显示了使用 OpenShift Container Storage/NooBaa 实例的示例配置：

```
DISTRIBUTED_STORAGE_CONFIG:
  rhocsStorage:
    - RHOCSSStorage
    - 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 ❶
server_side_assembly: true ❷

```

- ❶ 定义最终副本的最大块大小（以 MB 为单位）。如果 **server_side_assembly** 设为 **false**，则无效。
- ❷ Red Hat Quay 是否应该尝试使用服务器端装配和最终块的副本，而不是客户端装配。默认值为 **true**。

3.6.5. Ceph/RadosGW 存储

以下示例显示了在使用 Ceph/RadosGW 时可能的两个 YAML 配置。

示例 A：使用带有 **radosGWStorage** 驱动程序的 RadosGWW

```

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 ❶
      server_side_assembly: true ❷

```

- ❶ 定义最终副本的最大块大小（以 MB 为单位）。如果 **server_side_assembly** 设为 **false**，则无效。
- ❷ Red Hat Quay 是否应该尝试使用服务器端装配和最终块的副本，而不是客户端装配。默认值为 **true**。

示例 B：使用带有常规 s3 访问的 RadosGW

```

DISTRIBUTED_STORAGE_CONFIG:
  s3Storage: ❶
    - 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 ❷
      server_side_assembly: true ❸

```

- ❶ 用于常规 s3 访问。请注意，一般的 s3 访问不严格仅限于 Amazon Web Services (AWS) s3，可用于 RadosGW 或其他存储服务。有关使用 AWS S3 驱动程序的常规 s3 访问示例，请参阅"AWS S3 存储"。
- ❷ 定义最终副本的最大块大小（以 MB 为单位）。如果 **server_side_assembly** 设为 **false**，则无效。
- ❸

- 3 Red Hat Quay 是否应该尝试使用服务器端装配和最终块的副本，而不是客户端装配。默认值为 **true**。

3.6.6. AWS S3 存储

以下 YAML 显示了使用 AWS S3 存储的示例配置。

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

- 1 **S3Storage** 存储驱动程序应该仅用于 AWS S3 存储桶。请注意，这与常规 S3 访问不同，可以使用 RadosGW 驱动程序或其他存储服务。例如，请参阅"示例 B: 使用 RadosGW 带有常规 S3 访问"。

3.6.6.1. AWS STS S3 存储

以下 YAML 显示了在 OpenShift Container Platform 配置中使用 Amazon Web Services (AWS)安全令牌服务(STS)的示例配置。

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

- 1 唯一的 Amazon 资源名称(ARN)。
- 2 生成的 AWS S3 用户访问密钥。
- 3 生成的 AWS S3 用户 secret 密钥。

3.6.7. Google Cloud Storage

以下 YAML 显示了使用 Google Cloud Storage 的示例配置：

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

3.6.8. Azure Storage

以下 YAML 显示了使用 Azure Storage 的示例配置：

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

1 Azure 存储的 `endpoint_url` 参数是可选的，可用于 Microsoft Azure Government (MAG) 端点。如果留空，则 `endpoint_url` 将连接到普通的 Azure 区域。

从 Red Hat Quay 3.7 开始，您必须使用 MAG Blob 服务的主端点。使用 MAG Blob 服务的 Secondary 端点将导致以下错误：**AuthenticationErrorDetail:Cannot find the claimed account when trying to GetProperties for the account whusc8-secondary.**

3.6.9. Swift 存储

以下 YAML 显示了使用 Swift 存储的配置示例：

```
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 对象存储

以下 YAML 显示了使用 Nutanix 对象存储的示例配置。

```
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 对象存储

以下 YAML 显示了使用 IBM Cloud 对象存储的示例配置。

```
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 可选。建议设置为 **100mb**。

3.7. REDIS 配置字段

本节详细介绍了 Redis 部署可用的配置字段。

3.7.1. 构建日志

以下构建日志配置字段可用于 Redis 部署：

表 3.7. 构建日志配置

字段	类型	描述
BUILDLOGS_REDIS (Required)	对象	构建日志缓存的 redis 连接详情。
.host (必需)	字符串	可以访问 Redis 的主机名。 示例： quay-server.example.com
.port (必需)	Number	可以访问 Redis 的端口。 示例： 6379
.password	字符串	用于连接到 Redis 实例的密码。 示例： strongpassword
.SSL (可选)	布尔值	是否启用 Redis 和 Quay 之间的 TLS 通信。默认为false。

3.7.2. 用户事件

Redis 部署提供了以下用户事件字段：

表 3.8. 用户事件配置

字段	类型	描述
USER_EVENTS_REDIS (Required)	对象	用户事件处理的 redis 连接详情。
.host (必需)	字符串	可以访问 Redis 的主机名。 示例： quay-server.example.com
.port (必需)	Number	可以访问 Redis 的端口。 示例： 6379
.password	字符串	用于连接到 Redis 实例的密码。 示例： strongpassword
.ssl	布尔值	是否启用 Redis 和 Quay 之间的 TLS 通信。默认为false。

字段	类型	描述
<code>.ssl_keyfile</code> (可选)	字符串	密钥数据库文件的名称，其中包含要使用的客户端证书。 示例： ssl_keyfile: /path/to/server/privatekey.pem
<code>.ssl_certfile</code> (可选)	字符串	用于指定 SSL 证书的文件路径。 示例： ssl_certfile: /path/to/server/certificate.pem
<code>.ssl_cert_reqs</code> (可选)	字符串	用于指定在 SSL/TLS 握手期间要执行的证书验证级别。 示例： ssl_cert_reqs: CERT_REQUIRED
<code>.ssl_ca_certs</code> (可选)	字符串	用于指定包含可信证书颁发机构 (CA) 证书列表的文件的路径。 示例： ssl_ca_certs: /path/to/ca_certs.pem
<code>.ssl_ca_data</code> (Optional)	字符串	用于指定包含 PEM 格式的可信 CA 证书的字符串。 示例： ssl_ca_data: <certificate>
<code>.ssl_check_hostname</code> (可选)	布尔值	在设置到服务器的 SSL/TLS 连接时使用。它指定客户端是否应该检查服务器的 SSL/TLS 证书中的主机名是否与它所连接的服务器的主机名匹配。 示例： ssl_check_hostname: true

3.7.3. Redis 配置示例

以下 YAML 显示了使用带有可选 SSL/TLS 字段的 Redis 的示例配置：

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



```

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

```



注意

如果您的部署对 Redis 使用 Azure Cache，并且 **ssl** 设为 **true**，则端口默认为 **6380**。

3.8. MODELCACHE 配置选项

在 Red Hat Quay 中提供了用于配置 ModelCache 的选项。

3.8.1. memcache 配置选项

memcache 是默认的 ModelCache 配置选项。使用 Memcache 时，不需要额外的配置。

3.8.2. 单个 Redis 配置选项

以下配置适用于具有可选只读副本的单个 Redis 实例：

```

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. 集群 Redis 配置选项

对集群的 Redis 实例使用以下配置：

```

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. 标签过期配置字段

Red Hat Quay 提供了以下标签过期配置字段：

表 3.9. 标签过期配置字段

字段	类型	描述
FEATURE_GARBAGE_COLLECTION	布尔值	是否启用仓库的垃圾回收。 Default: True
TAG_EXPIRATION_OPTIONS (Required)	字符串数组	如果启用，用户可以选择在其命名空间中为标签过期的选项。 Pattern: ^[0-9]+(w m d h s)\$
DEFAULT_TAG_EXPIRATION (Required)	字符串	时间机器的默认可配置标签过期时间。 Pattern: ^[0-9]+(w m d h s)\$ Default: 2w
FEATURE_CHANGE_TAG_EXPIRATION	布尔值	用户和机构是否允许更改命名空间中标签的标签过期时间。 Default: True
FEATURE_AUTO_PRUNE	布尔值	当设置为 True 时，启用与自动运行标签相关的功能。 Default: False

3.9.1. 标签过期配置示例

以下 YAML 显示了标签过期配置示例：

```

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

```

3.10. 配额管理配置字段

表 3.10. 配额管理配置

字段	类型	描述
FEATURE_QUOTA_MANAGEMENT	布尔值	为配额管理功能启用配置、缓存和验证。 **Default:** `False`
DEFAULT_SYSTEM_REJECT_QUOTA_BYTES	字符串	为所有机构启用系统默认配额拒绝字节允许。 默认情况下，不会设置限制。
QUOTA_BACKFILL	布尔值	启用配额回填工作程序来计算预先存在的 Blob 的大小。 默认: True
QUOTA_TOTAL_DELAY_SECONDS	字符串	启动配额回填的时间延迟。滚动部署可能会导致总部署不正确。此字段 必须 设置为 超过滚动部署完成所需的时间。 默认 : 1800
PERMANENTLY_DELETE_TAGS	布尔值	启用与从时间窗中删除标签相关的功能。 默认 : False
RESET_CHILD_MANIFEST_EXPIRATION	布尔值	重置以子清单为目标的临时标签的过期。将这个功能设置为 True 时，子清单会立即收集垃圾回收。 默认 : False

3.10.1. 配额管理配置示例

以下 YAML 是启用配额管理时推荐的配置。

配额管理 YAML 配置

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

3.11. 代理缓存配置字段

表 3.11. 代理配置

字段	类型	描述
FEATURE_PROXY_CACHE	布尔值	使 Red Hat Quay 能够通过上游 registry 的缓存作为拉取(pull)。 默认 : false

3.12. 机器人帐户配置字段

表 3.12. 机器人帐户配置字段

字段	类型	描述
ROBOTS_DISALLOW	布尔值	当设置为 true 时，机器人帐户将阻止所有交互，以及创建 Default: False

3.13. 预配置 RED HAT QUAY 用于自动化

Red Hat Quay 支持多种启用自动化的配置选项。用户可以在部署前配置这些选项，以减少与用户界面交互的需求。

3.13.1. 允许 API 创建第一个用户

要创建第一个用户，用户需要将 **FEATURE_USER_INITIALIZE** 参数设置为 **true**，并调用 `/api/v1/user/initialize` API。与需要现有机构中 OAuth 应用生成的 OAuth 令牌的所有其他 registry API 调用不同，API 端点不需要身份验证。

在部署 Red Hat Quay 后，用户可以使用 API 创建如 **quayadmin** 的用户，只要没有创建其他用户。如需更多信息，[请参阅使用 API 创建第一个用户](#)。

3.13.2. 启用常规 API 访问

用户应将 **BROWSER_API_CALLS_XHR_ONLY** 配置选项设置为 **false**，以允许对 Red Hat Quay registry API 的常规访问。

3.13.3. 添加超级用户

部署 Red Hat Quay 后，用户可以创建用户，并授予第一个具有完整权限的用户管理员特权。用户可以使用 **SUPER_USER** 配置对象提前配置完整的权限。例如：

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

3.13.4. 限制用户创建

配置超级用户后，您可以通过将 **FEATURE_USER_CREATION** 设置为 **false** 来限制创建新用户到超级用户组的能力。例如：

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

3.13.5. 在 Red Hat Quay 3.11 中启用新功能

要使用新的 Red Hat Quay 3.11 功能，请启用以下一些或所有功能：

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

3.13.6. 建议自动化配置

建议对自动化使用以下 **config.yaml** 参数：

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

3.13.7. 使用初始配置部署 Red Hat Quay Operator

使用以下步骤使用初始配置在 OpenShift Container Platform 上部署 Red Hat Quay。

先决条件

- 已安装 **oc** CLI。

流程

1. 使用配置文件创建 secret：

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

2. 创建 **quayregistry.yaml** 文件。识别非受管组件并引用创建的 secret，例如：

```
apiVersion: quay.redhat.com/v1
```

```
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  configBundleSecret: init-config-bundle-secret
```

- 部署 Red Hat Quay registry :

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

后续步骤

- 使用 [API 创建第一个用户](#)

3.13.8. 使用 API 创建第一个用户

使用以下步骤在 Red Hat Quay 组织中创建第一个用户。

先决条件

- 配置选项 `FEATURE_USER_INITIALIZE` 必须设置为 `true`。
- 数据库中不能已存在任何用户。



流程

此流程通过指定 `"access_token": true` 来请求 OAuth 令牌。

- 打开 Red Hat Quay 配置文件并更新以下配置字段 :

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

- 输入以下命令停止 Red Hat Quay 服务 :

```
$ sudo podman stop quay
```

- 输入以下命令启动 Red Hat Quay 服务 :

```
$ 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}
```

- 运行以下 **CURL** 命令以使用用户名、密码、电子邮件和访问令牌生成新用户 :

```
$ 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}'
```

如果成功，命令会返回带有用户名、电子邮件和加密密码的对象。例如：

```
{
  "access_token": "6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED",
  "email": "quayadmin@example.com",
  "encrypted_password": "1nZMLH57RIE5UGdL/yYpDOHLqiNCgimb6W9kfF8MjZ1xrfDpRyRs9NUuUuNuAitW",
  "username": "quayadmin"
} #
gitlaks:allow
```

如果用户已在数据库中，则返回错误：

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

如果您的密码至少不是八个字符或包含空格，则返回错误：

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

5. 输入以下命令登录到您的 Red Hat Quay 部署：

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

输出示例

```
Login Succeeded!
```

3.13.8.1. 使用 OAuth 令牌

调用 API 后，您可以通过指定返回的 OAuth 代码来调用 Red Hat Quay API 的其余部分。

先决条件

- 您已调用 `/api/v1/user/initialize` API，并传递用户名、密码和电子邮件地址。

流程

- 输入以下命令来获取当前用户列表：

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

输出示例：

```
{
  "users": [
    {
      "kind": "user",
      "name": "quayadmin",
      "username": "quayadmin",
      "email": "quayadmin@example.com",
      "verified": true,
      "avatar": {
        "name": "quayadmin",
        "hash":
```

```

    "3e82e9cbf62d25dec0ed1b4c66ca7c5d47ab9f1f271958298dea856fb26adc4c",
    "color": "#e7ba52",
    "kind": "user"
  },
  "super_user": true,
  "enabled": true
}
]
}

```

在本实例中，**quayadmin** 用户的详细信息返回，因为它是目前创建的唯一用户。

3.13.8.2. 使用 API 创建机构

以下过程详细介绍了如何使用 API 创建 Red Hat Quay 组织。

先决条件

- 您已调用 `/api/v1/user/initialize` API，并传递用户名、密码和电子邮件地址。
- 您已通过指定返回的 OAuth 代码调用了其余的 Red Hat Quay API。

流程

1. 要创建机构，请使用对 `api/v1/organization/` 端点的 POST 调用：

```

$ 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"}'

```

输出示例：

```
"Created"
```

2. 您可以输入以下命令检索您创建的机构的详情：

```

$ 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

```

输出示例：

```

{
  "name": "testorg",
  "email": "testorg@example.com",
  "avatar": {
    "name": "testorg",
    "hash": "5f113632ad532fc78215c9258a4fb60606d1fa386c91b141116a1317bf9c53c8",
    "color": "#a55194",
    "kind": "user"
  },
  "is_admin": true,
  "is_member": true,

```



```

"teams": {
  "owners": {
    "name": "owners",
    "description": "",
    "role": "admin",
    "avatar": {
      "name": "owners",
      "hash":
"6f0e3a8c0eb46e8834b43b03374ece43a030621d92a7437beb48f871e90f8d90",
      "color": "#c7c7c7",
      "kind": "team"
    },
    "can_view": true,
    "repo_count": 0,
    "member_count": 1,
    "is_synced": false
  }
},
"ordered_teams": [
  "owners"
],
"invoice_email": false,
"invoice_email_address": null,
"tag_expiration_s": 1209600,
"is_free_account": true
}

```

3.14. 基本配置字段

表 3.13. 基本配置

字段	类型	描述
REGISTRY_TITLE	字符串	如果指定了，registry 的长格式标题。在 Red Hat Quay 部署的 frontend 中显示，例如，在您的机构的登录页面中。不应超过 35 个字符。 默认： Red Hat Quay
REGISTRY_TITLE_SHORT	字符串	如果指定了，registry 的简短标题。标题显示在您的机构的各种页面上，例如，作为您机构的 Tutorial 页面上的教程的标题。 默认： Red Hat Quay
CONTACT_INFO	字符串数组	如果指定，请联系页面中要显示的联系信息。如果只指定单一的联系信息，请联系页脚将直接链接。

字段	类型	描述
[0]	字符串	<p>添加链接以发送电子邮件。</p> <p>Pattern: <code>^mailto: (.)+\$</code> 示例： <code>mailto:support@quay.io</code></p>
[1]	字符串	<p>添加链接以访问 IRC chat room。</p> <p>Pattern: <code>^irc://(.)+\$</code> Example: irc://chat.freenode.net:6665/quay</p>
[2]	字符串	<p>添加链接来调用电话号码。</p> <p>Pattern: <code>^tel: (.)+\$</code> 示例： <code>tel:+1-888-930-3475</code></p>
[3]	字符串	<p>向定义的 URL 添加链接。</p> <p>Pattern: <code>^http (s)?://(.)+\$</code> Example: https://twitter.com/quayio</p>

3.15. SSL 配置字段

表 3.14. SSL 配置

字段	类型	描述
----	----	----

字段	类型	描述
PREFERRED_URL_SCHEME	字符串	<p>http 或 https 之一。请注意，当用户从客户端到 Quay 的通信路径中没有 TLS 加密时，用户仅将 PREFERRED_URL_SCHEME 设置为 http。</p> <p>在使用 TLS 终止负载均衡器、反向代理（如 Nginx）或者使用带有自定义 SSL 证书的 Quay 时，用户必须将其 PREFERRED_URL_SCHEME 设置为 https。在大多数情况下，PREFERRED_URL_SCHEME 应该为 https。</p> <p>默认：http</p>
SERVER_HOSTNAME (Required)	字符串	<p>可以访问 Red Hat Quay 的 URL，不包括网络协议。</p> <p>例如： quay-server.example.com</p>

字段	类型	描述
SSL_CIPHERS	字符串数组	<p>如果指定了，用于启用和禁用的 SSL 密码的 nginx 定义列表</p> <p>示例：</p> <pre>[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,EC-DHE-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,aes128-SHA256,aes128-GCM-sha256,aes256-GCM-SHA256,AES256-SHA256,AES128-SHA,AES256-SHA,AES,!3DES",!aNULL,!eNULL,!EXPORT,DES,!RC4,MD5,MD5,!PSK,!aECDH,!EDH-DSS-DES-CBC3-SHA,!EDH-RSA-DES-CBC3-SHA,!KRB5-DES-CBC3-SHA]</pre>

字段	类型	描述
SSL_PROTOCOLS	字符串数组	<p>如果指定了，nginx 被配置为启用列表中定义的 SSL 协议列表。从列表中删除 SSL 协议会禁用 Red Hat Quay 启动期间的协议。</p> <p>示例： <code>['TLSv1','TLSv1.1','TLSv1.2','TLSv1.3']</code></p>
SESSION_COOKIE_SECURE	布尔值	<p>对于使用 SSL 的所有安装，是否应在会话 Cookie 中设置 secure 属性</p> <p>默认值： False</p> <p>Recommendation: Set to True</p>

3.15.1. 配置 SSL

1. 将证书文件和主密钥文件复制到您的配置目录中，确保它们分别命名为 **ssl.cert** 和 **ssl.key**：

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

2. 编辑 **config.yaml** 文件并指定您希望 Quay 处理 TLS：

config.yaml

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

3. 停止 **Quay** 容器并重启 registry

3.16. 将 TLS 证书添加到 RED HAT QUAY CONTAINER

要将自定义 TLS 证书添加到 Red Hat Quay，请在 Red Hat Quay config 目录下创建一个名为 **extra_ca_certs/** 的新目录。将任何所需的特定于站点的 TLS 证书复制到这个新目录中。

3.16.1. 将 TLS 证书添加到 Red Hat Quay

1. 查看要添加到容器中的证书

```
$ cat storage.crt
-----BEGIN CERTIFICATE-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGS1b3DQEBCwUAMD0xCzAJBgNV
[...]
-----END CERTIFICATE-----
```

2. 创建 certs 目录并复制证书

```
$ 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. 使用 podman ps 获取 Quay 容器的 CONTAINER ID :

```
$ sudo podman ps
CONTAINER ID      IMAGE                                COMMAND                                             CREATED
STATUS           PORTS
5a3e82c4a75f     <registry>/<repo>/quay:v3.11.1 "/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. 使用该 ID 重启容器 :

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

5. 检查复制到容器命名空间中的证书 :

```
$ sudo podman exec -it 5a3e82c4a75f cat /etc/ssl/certs/storage.pem
-----BEGIN CERTIFICATE-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGS1b3DQEBCwUAMD0xCzAJBgNV
```

3.17. LDAP 配置字段

表 3.15. LDAP 配置

字段	类型	描述
AUTHENTICATION_TYPE (Required)	字符串	必须设置为 LDAP 。
FEATURE_TEAM_SYNCING	布尔值	是否允许团队成员资格从身份验证引擎 (OIDC、LDAP 或 Keystone) 中的后备组同步。 Default: true

字段	类型	描述
FEATURE_NONSUPERUSER_TEAM_SYNCING_SETUP	布尔值	如果启用，非超级用户可以设置团队同步。 Default: false
LDAP_ADMIN_DN	字符串	用于 LDAP 验证的管理 DN。
LDAP_ADMIN_PASSWD	字符串	LDAP 身份验证的 admin 密码。
LDAP_ALLOW_INSECURE_FALLBACK	布尔值	是否允许 SSL 不安全的回退进行 LDAP 身份验证。
LDAP_BASE_DN	字符串数组	LDAP 身份验证的基本 DN。
LDAP_EMAIL_ATTR	字符串	LDAP 身份验证的电子邮件属性。
LDAP_UID_ATTR	字符串	LDAP 身份验证的 uid 属性。
LDAP_URI	字符串	LDAP URI。
LDAP_USER_FILTER	字符串	LDAP 身份验证的用户过滤器。
LDAP_USER_RDN	字符串数组	用于 LDAP 身份验证的用户 RDN。
TEAM_RESYNC_STALE_TIME	字符串	如果为团队启用了团队同步，需要检查其成员资格和重新同步。 Pattern: ^[0-9]+(w m d h s)\$ 示例： 2h Default: 30m
LDAP_SUPERUSER_FILTER	字符串	LDAP_USER_FILTER 配置字段的子集。配置后，当 Red Hat Quay 使用 LDAP 作为其身份验证提供程序时，Red Hat Quay 管理员可以将轻量级目录访问协议 (LDAP) 用户配置为超级用户。 使用此字段，管理员可以在不更新 Red Hat Quay 配置文件并重启其部署的情况下添加或删除超级用户。 此字段要求您将 AUTHENTICATION_TYPE 设置为 LDAP 。

字段	类型	描述
LDAP_RESTRICTED_USER_FILTER	字符串	<p>LDAP_USER_FILTER 配置字段的子集。配置后，当 Red Hat Quay 使用 LDAP 作为其身份验证提供程序时，Red Hat Quay 管理员可以将轻量级目录访问协议 (LDAP) 用户配置为受限的用户。</p> <p>此字段要求您将 AUTHENTICATION_TYPE 设置为 LDAP。</p>
LDAP_TIMEOUT	整数	<p>决定与轻量级目录访问协议(LDAP)服务器建立连接的最大时间段（以秒为单位）。</p> <p>默认：10</p>
LDAP_NETWORK_TIMEOUT	整数	<p>定义 Red Hat Quay 在网络操作过程中等待轻量级目录访问协议(LDAP)服务器的响应的最长时间（以秒为单位）。</p> <p>默认：10</p>

3.17.1. LDAP 配置参考

使用以下引用，使用所需 LDAP 设置更新 **config.yaml** 文件。

3.17.1.1. 基本 LDAP 配置

使用以下引用进行基本 LDAP 配置。

```

---
AUTHENTICATION_TYPE: LDAP 1
---
LDAP_ADMIN_DN: uid=<name>,ou=Users,o=<organization_id>,dc=
<example_domain_component>,dc=com 2
LDAP_ADMIN_PASSWD: ABC123 3
LDAP_ALLOW_INSECURE_FALLBACK: false 4
LDAP_BASE_DN: 5
  - o=<organization_id>
  - dc=<example_domain_component>
  - dc=com
LDAP_EMAIL_ATTR: mail 6
LDAP_UID_ATTR: uid 7
LDAP_URI: ldap://<example_url>.com 8
LDAP_USER_FILTER: (memberof=cn=developers,ou=Users,dc=<domain_name>,dc=com) 9
LDAP_USER_RDN: 10
  - ou=<example_organization_unit>
  - o=<organization_id>
  - dc=<example_domain_component>
  - dc=com

```


- 1 必需。必须设置为 **LDAP**。
- 2 必需。用于 LDAP 验证的管理 DN。
- 3 必需。LDAP 身份验证的 admin 密码。
- 4 必需。是否允许 SSL/TLS 不安全的 LDAP 身份验证回退。
- 5 必需。LDAP 身份验证的基本 DN。
- 6 必需。LDAP 身份验证的电子邮件属性。
- 7 必需。LDAP 身份验证的 UID 属性。
- 8 必需。LDAP URI。
- 9 必需。LDAP 身份验证的用户过滤器。
- 10 必需。用于 LDAP 身份验证的用户 RDN。

3.17.1.2. LDAP 受限用户配置

对 LDAP 受限用户配置使用以下引用。

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

- 1 将指定用户为受限用户。

3.17.1.3. LDAP 超级用户配置参考

使用以下引用 LDAP 超级用户配置。

■

```

# ...
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>) ❶
LDAP_USER_RDN:
  - ou=<example_organization_unit>
  - o=<organization_id>
  - dc=<example_domain_component>
  - dc=com
# ...

```

- ❶ 将指定的用户配置为超级用户。

3.18. 镜像配置字段

表 3.16. 镜像配置

字段	类型	描述
FEATURE_REPO_MIRROR	布尔值	启用或禁用存储库镜像 默认值：false
REPO_MIRROR_INTERVAL	Number	检查存储库镜像候选者之间的秒数 默认值：30
REPO_MIRROR_SERVER_HOSTNAME	字符串	将 SERVER_HOSTNAME 替换为镜像的目标。 Default: None Example: openshift-quay-service
REPO_MIRROR_TLS_VERIFY	布尔值	在镜像过程中需要 HTTPS 并验证 Quay registry 的证书。 默认：false

字段	类型	描述
REPO_MIRROR_ROLLBACK	布尔值	当设置为 true 时，存储库会在镜像失败后回滚。 默认 : false

3.19. 安全扫描程序配置字段

表 3.17. 安全扫描程序配置

字段	类型	描述
FEATURE_SECURITY_SCANNER	布尔值	启用或禁用安全扫描程序 默认值 : false
FEATURE_SECURITY_NOTIFICATIONS	布尔值	如果启用了安全扫描程序，请打开或关闭安全通知 Default: false
SECURITY_SCANNER_V4_REINDEX_THRESHOLD	字符串	此参数用于确定在重新索引以来具有之前失败或自上次索引后更改状态的清单前要等待的最短时间（以秒为单位）。数据从 manifestsecuritystatus 表中的 last_indexed datetime 计算。这个参数用于避免在每次索引运行时尝试重新索引每个失败的清单。re-index 的默认时间为 300 秒。
SECURITY_SCANNER_V4_ENDPOINT	字符串	V4 安全扫描程序的端点 Pattern: <code>^http(s)?://(.)+\$</code> 示例 : http://192.168.99.101:6060
SECURITY_SCANNER_V4_PSK	字符串	为 Clair 生成预共享密钥(PSK)
SECURITY_SCANNER_ENDPOINT	字符串	V2 安全扫描程序的端点 Pattern: <code>^http(s)?://(.)+\$</code> 示例 : http://192.168.99.100:6060

字段	类型	描述
SECURITY_SCANNER_INDEXING_INTERVAL	整数	此参数用于确定安全扫描程序中索引间隔之间的秒数。触发索引时，Red Hat Quay 将查询其数据库以获取由 Clair 索引的清单。这包括尚未索引的清单，以及之前失败的索引清单。 默认值：30
FEATURE_SECURITY_SCANNER_NOTIFY_ON_NEW_INDEX	布尔值	是否允许发送有关新推送的漏洞的通知。 默认:True
SECURITY_SCANNER_V4_MANIFEST_CLEANUP	布尔值	Red Hat Quay 垃圾回收程序会删除没有由其他标签或清单引用的清单。 默认*: True

3.19.1. 使用 Clair v4 重新索引

当 Clair v4 索引清单时，结果应该是确定的。例如，同一清单应生成相同的索引报告。在扫描程序更改前，这为 true，因为使用不同的扫描程序会生成与报告返回的特定清单相关的不同信息。因此，Clair v4 会公开索引引擎(/indexer/api/v1/index_state)的状态表示，以确定扫描程序配置是否已更改。

Red Hat Quay 通过在解析到 Quay 数据库时将其保存到索引报告中来利用这个索引状态。如果因为清单之前扫描以来此状态已更改，Red Hat Quay 会在定期索引过程中尝试重新索引该清单。

默认情况下，此参数设置为 30 秒。如果用户希望索引过程更频繁地运行，例如，如果他们不希望等待 30 秒在推送新标签后查看 UI 中的安全扫描结果，则用户可能会缩短时间。如果用户希望对请求模式进行更多控制，以及要在 Red Hat Quay 数据库上执行的数据库操作的模式，用户也可以更改参数。

3.19.2. 安全扫描程序配置示例

以下 YAML 是启用安全扫描程序功能时推荐的配置。

安全扫描程序 YAML 配置

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

3.20. HELM 配置字段

表 3.18. Helm 配置字段

字段	类型	描述
FEATURE_GENERAL_OCI_SUPPORT	布尔值	启用对 OCI 工件的支持。 默认值：True

以下 Open Container Initiative (OCI)工件类型默认内置在 Red Hat Quay 中，并通过 FEATURE_GENERAL_OCI_SUPPORT 配置字段启用：

字段	介质类型	支持的内容类型
Helm	application/vnd.cncf.helm.config.v1+json	application/tar+gzip, application/vnd.cncf.helm.chart.content.v1.tar+gzip
Cosign	application/vnd.oci.image.config.v1+json	application/vnd.dev.cosign.simplesigning.v1+json,application/vnd.dsse.envelope.v1+json
SPDX	application/vnd.oci.image.config.v1+json	text/spdx, text/spdx+xml, text/spdx+json
Syft	application/vnd.oci.image.config.v1+json	application/vnd.syft+json
CycloneDX	application/vnd.oci.image.config.v1+json	application/vnd.cyclonedx,application/vnd.cyclonedx+xml,application/vnd.cyclonedx+json
in-toto	application/vnd.oci.image.config.v1+json	application/vnd.in-toto+json
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. 配置 Helm

以下 YAML 是启用 Helm 时的示例配置。

Helm YAML 配置

```
FEATURE_GENERAL_OCI_SUPPORT: true
```

3.21. 开放容器项目配置字段

表 3.19. 其他 OCI 工件配置字段

字段	类型	描述
ALLOWED_OCI_ARTIFACT_TYPES	对象	允许的 OCI 工件 mimetypes 和关联的层类型集合。

3.21.1. 配置额外的工件类型

可以使用 **ALLOWED_OCI_ARTIFACT_TYPES** 配置字段将默认不支持的其他 OCI 工件类型添加到您的 Red Hat Quay 部署中。

使用以下引用添加额外的 OCI 工件类型：

OCI 工件类型配置

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

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

例如，您可以通过将以下内容添加到 **config.yaml** 文件中来添加单数(SIF)支持：

OCI 工件类型配置示例

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



注意

当添加不默认配置的 OCI 工件类型时，Red Hat Quay 管理员还需要手动添加对 cosign 和 Helm 的支持。

3.22. 未知介质类型

表 3.20. 未知介质类型配置字段

字段	类型	描述
IGNORE_UNKNOWN_MEDIATYPES	布尔值	启用后，允许容器 registry 平台对支持的工件类型忽略特定的限制，并接受任何未识别或未知的介质类型。 默认：false

3.22.1. 配置未知介质类型

当启用未知或未识别的介质类型时，以下 YAML 是示例配置。

未知介质类型 YAML 配置

```
IGNORE_UNKNOWN_MEDIATYPES: true
```

3.23. 操作日志配置字段

3.23.1. 操作日志存储配置

表 3.21. 操作日志存储配置

字段	类型	描述
FEATURE_LOG_EXPORT	布尔值	是否允许导出操作日志。 默认值：True
LOGS_MODEL	字符串	指定处理日志数据的首选方法。 值：一个数据库 <code>,transition_reads_both_writes,elasticsearch, mvapich</code> Default: database
LOGS_MODEL_CONFIG	对象	用于操作日志的日志模型配置。

- LOGS_MODEL_CONFIG [object]：操作日志的日志模型配置。
 - elasticsearch_config [object]: Elasticsearch 集群配置。
 - access_key [string]: Elasticsearch 用户（或 AWS ES 的 IAM 密钥）。
 - 示例：**some_string**

- **Host** [string]: Elasticsearch 集群端点。
 - 示例 : **host.elasticsearch.example**
- **index_prefix** [string]: Elasticsearch 的索引前缀。
 - 示例:**logentry_**
- **index_settings** [object]: Elasticsearch 的索引设置
- **use_ssl** [boolean] : 将 ssl 用于 Elasticsearch。默认值为 **True**。
 - 示例:**True**
- **SECRET_KEY** [字符串] : Elasticsearch 密码 (或 AWS ES 的 IAM secret) 。
 - 示例 : **some_secret_string**
- **AWS_REGION** [string]: Amazon Web 服务区域。
 - 示例:**us-east-1**
- **端口** [number]: Elasticsearch 集群端点端口。
 - 示例 : **1234**
- **kinesis_stream_config** [object]: AWS Kinesis Stream 配置。
 - **aws_secret_key** [string]: AWS secret key。
 - 示例 : **some_secret_key**
 - **stream_name** [string]: Kinesis 流将操作日志发送到。
 - 示例 : **logentry-kinesis-stream**
 - **aws_access_key** [string]: AWS 访问密钥。
 - 示例 : **some_access_key**
 - **retries** [number] : 在单个请求中尝试次数。
 - 示例 : **5**
 - **read_timeout** [number] : 从连接读取时超时前的秒数。
 - 示例 : **5**
 - **max_pool_connections** [number] : 连接池中要保留的最大连接数。
 - 示例 : **10**
 - **AWS_REGION** [string]: AWS region。
 - 示例:**us-east-1**
 - **connect_timeout** [number] : 尝试进行连接时超时前的秒数。
 - 示例 : **5**

- **producer** [string]: 如果日志记录到 Elasticsearch, 则日志制作者。
 - **Enum**: kafka, elasticsearch, kinesis_stream
 - **示例** : **kafka**
- **kafka_config** [object]: Kafka 集群配置。
 - **topic** [string]: Kafka topic 将日志条目发布到。
 - **示例** : **logentry**
 - **bootstrap_servers** [array]: 从中引导客户端的 Kafka 代理列表。
 - **max_block_seconds** [number]: 在 **send ()** 期间最大要阻止的秒数, 因为缓冲区已满或元数据不可用。
 - **示例** : **10**
- **producer** [字符串]: **mvapich**
- **mvapich_config** [object]: Splunk 操作日志的日志模型配置或 Splunk 集群配置。
 - **Host** [string]: Splunk 集群端点。
 - **port** [integer]: Splunk 管理集群端点端口。
 - **bearer_token** [string]: Splunk 的 bearer 令牌。
 - **verify_ssl** [boolean]: 启用(**True**)或为 HTTPS 连接禁用 TLS/SSL 验证。
 - **index_prefix** [string]: Splunk 的索引前缀。
 - **ssl_ca_path** [字符串]: 指向包含用于 SSL 验证的证书颁发机构(CA)的单个 .pem 文件的相对容器路径。

3.23.2. 操作日志轮转和归档配置

表 3.22. 操作日志轮转和归档配置

字段	类型	描述
FEATURE_ACTION_LOG_ROTATION	布尔值	启用日志轮转和归档会将所有超过 30 天的日志移到存储。 默认 : false
ACTION_LOG_ARCHIVE_LOCATION	字符串	如果启用了操作日志存档, 则在其中放置存档数据的存储引擎。 示例 : s3_us_east

字段	类型	描述
ACTION_LOG_ARCHIVE_PATH	字符串	如果启用了操作日志存档，则存储中要放置存档数据的路径。 示例：archive/actionlogs
ACTION_LOG_ROTATION_THRESHOLD	字符串	轮转日志的时间间隔。 示例：30d

3.23.3. 操作日志审计配置

表 3.23. 审计日志配置字段

字段	类型	描述
ACTION_LOG_AUDIT_LOGINS	布尔值	当设置为 True 时，请跟踪高级事件，如登录和注销、UI 和使用 Docker 进行常规用户、机器人帐户以及特定于应用的令牌帐户。 默认：True

3.24. 构建日志配置字段

表 3.24. 构建日志配置字段

字段	类型	描述
FEATURE_READER_BUILD_LOGS	布尔值	如果设置为 true，则构建日志可以被对存储库具有读取访问权限的用户 读取 ，而不仅仅是 写入 访问或 admin 访问权限。 Default: False
LOG_ARCHIVE_LOCATION	字符串	在 DISTRIBUTED_STORAGE_CONFIG 中定义的存储位置，用于放置归档的构建日志。 示例：s3_us_east
LOG_ARCHIVE_PATH	字符串	配置的存储引擎下的路径，在其中将归档的构建日志放在 .JSON 格式。 示例：archive/buildlogs

3.25. DOCKERFILE 构建触发器字段

表 3.25. Dockerfile 构建支持

字段	类型	描述
FEATURE_BUILD_SUPPORT	布尔值	是否支持 Dockerfile 构建。 Default: False
SUCCESSIVE_TRIGGER_FAILURE_DISABLE_THRESHOLD	Number	如果没有设置 None ，则构建触发器前可能会出现连续失败次数。 默认：100
SUCCESSIVE_TRIGGER_INTERNAL_ERROR_DISABLE_THRESHOLD	Number	如果没有设置 None ，则构建触发器前可能出现的连续内部错误数量 默认为 5

3.25.1. GitHub 构建触发器

表 3.26. GitHub 构建触发器

字段	类型	描述
FEATURE_GITHUB_BUILD	布尔值	是否支持 GitHub 构建触发器。 默认值：False
GITHUB_TRIGGER_CONFIG	对象	使用 GitHub Enterprise 进行构建触发器的配置。
.GITHUB_ENDPOINT (Required)	字符串	GitHub Enterprise 的端点。 示例：https://github.com/
.API_ENDPOINT	字符串	要使用的 GitHub Enterprise API 的端点。必须覆盖 github.com 。 示例：https://api.github.com/
.CLIENT_ID (Required)	字符串	此 Red Hat Quay 实例注册的客户端 ID；这不能与 GITHUB_LOGIN_CONFIG 共享。

字段	类型	描述
.CLIENT_SECRET (Required)	字符串	此 Red Hat Quay 实例的注册的客户端 secret。

3.25.2. Bitbucket 构建触发器

表 3.27. Bitbucket 构建触发器

字段	类型	描述
FEATURE_BITBUCKET_BUILD	布尔值	是否支持 Bitbucket 构建触发器。 默认值： False
BITBUCKET_TRIGGER_CONFIG	对象	使用 BitBucket 进行构建触发器的配置。
.CONSUMER_KEY (Required)	字符串	此 Red Hat Quay 实例注册的使用者密钥（客户端 ID）。
.CONSUMER_SECRET (Required)	字符串	此 Red Hat Quay 实例注册的消费者 secret（客户端 secret）。

3.25.3. GitLab 构建触发器

表 3.28. GitLab 构建触发器

字段	类型	描述
FEATURE_GITLAB_BUILD	布尔值	是否支持 GitLab 构建触发器。 默认值： False
GITLAB_TRIGGER_CONFIG	对象	使用 Gitlab 进行构建触发器的配置。
.GITLAB_ENDPOINT (Required)	字符串	运行 Gitlab Enterprise 的端点。
.CLIENT_ID (Required)	字符串	此 Red Hat Quay 实例的注册的客户端 ID。

字段	类型	描述
<code>.CLIENT_SECRET</code> (Required)	字符串	此 Red Hat Quay 实例的注册的客户端 secret。

3.26. 构建管理器配置字段

表 3.29. 构建管理器配置字段

字段	类型	描述
<code>ALLOWED_WORKER_COUNT</code>	字符串	定义每个 Red Hat Quay pod 实例化多少个 Build Worker。通常设置为 1 。
<code>ORCHESTRATOR_PREFIX</code>	字符串	定义添加到所有 Redis 密钥的唯一前缀。这可用于将 Orchestrator 值与其他 Redis 键隔离。
<code>REDIS_HOST</code>	对象	Redis 服务的主机名。
<code>REDIS_PASSWORD</code>	字符串	在 Redis 服务中进行身份验证的密码。
<code>REDIS_SSL</code>	布尔值	定义您的 Redis 连接是否使用 SSL/TLS。
<code>REDIS_SKIP_KEYSPACE_EVENT_SETUP</code>	布尔值	默认情况下，Red Hat Quay 不会在运行时设置关键事件所需的关键空间事件。为此，请将 REDIS_SKIP_KEYSPACE_EVENT_SETUP 设置为 false 。
<code>EXECUTOR</code>	字符串	启动此类型的可执行文件定义。有效值为 kubernetes 和 ec2 。
<code>BUILDER_NAMESPACE</code>	字符串	将进行 Red Hat Quay 构建的 Kubernetes 命名空间。
<code>K8S_API_SERVER</code>	对象	发生 Builds 的 OpenShift Container Platform 集群的 API 服务器的主机名。
<code>K8S_API_TLS_CA</code>	对象	为 Quay 应用构建集群的 CA 证书的 Quay 容器中的文件路径，以便在发出 API 调用时信任。

字段	类型	描述
KUBERNETES_DISTRIBUTION	字符串	指明正在使用的 Kubernetes 类型。有效值为 openshift 和 k8s 。
CONTAINER_*	对象	定义每个构建 Pod 的资源请求和限值。
NODE_SELECTOR_*	对象	定义应调度 构建 Pod 的节点选择器标签名称-值对。
CONTAINER_RUNTIME	对象	指定 Builder 是否应该运行 docker 或 podman 。使用红帽的 quay-builder 镜像的客户应将其设置为 podman 。
SERVICE_ACCOUNT_NAME/SERVICE_ACCOUNT_TOKEN	对象	定义 构建 Pod 将要使用的服务帐户名称或令牌。
QUAY_USERNAME/QUAY_PASSWORD	对象	定义拉取 WORKER_IMAGE 字段中指定的 Red Hat Quay 构建 worker 镜像所需的 registry 凭证。客户应提供一个 Red Hat Service Account 凭证，如 https://access.redhat.com/RegistryAuthentication 文章中的针对 registry.redhat.io 的"创建 Registry 服务账户"部分。
WORKER_IMAGE	对象	Red Hat Quay Builder 镜像的镜像引用。 registry.redhat.io/quay/quay-builder
WORKER_TAG	对象	需要的 Builder 镜像的标签。最新版本为 3.11。
BUILDER_VM_CONTAINER_IMAGE	对象	对包含运行每个 Red Hat Quay 构建所需的内部虚拟机的完整引用 (registry.redhat.io/quay/quay-builder-qemu-rhcos:3.11)。

字段	类型	描述
SETUP_TIME	字符串	指定在 Build Manager 中尚未注册自身时构建超时的秒数。默认值为 500 秒。尝试重启三次的构建。如果构建在三次尝试后没有注册自己，则被视为失败。
MINIMUM_RETRY_THRESHOLD	字符串	此设置与多个可执行文件一起使用。它指示在选择不同的可执行文件前尝试启动构建的次数。设置为 0 表示对于构建作业需要的次数没有限制。这个值应该保持非常小（三个或更少），以确保在基础架构故障期间快速进行故障转移。必须为此设置指定一个值。例如， Kubernetes 被设置为第一个 executor， EC2 设置为第二个 executor。如果您希望最后一次尝试始终在 EC2 上执行作业，而不是在 EC2 上执行，您可以将 Kubernetes executor 的 MINIMUM_RETRY_THRESHOLD 设置为 1 ，EC2 的 MINIMUM_RETRY_THRESHOLD 设置为 0 （如果未设置则为 0 ）。在这种情况下，Kubernetes 的 MINIMUM_RETRY_THRESHOLD retries_remaining (1) 将评估为 False ，因此回退到配置的第二个 executor。
SSH_AUTHORIZED_KEYS	对象	在 ignition 配置中要引导的 SSH 密钥列表。这允许使用其他密钥通过 SSH 连接到 EC2 实例或 QEMU 虚拟机(VM)。

3.27. OAUTH 配置字段

表 3.30. OAuth 字段

字段	类型	描述
DIRECT_OAUTH_CLIENTID_WHITELIST	字符串数组	允许在没有用户批准的情况下执行直接 OAuth 批准的 Quay 管理的应用程序的客户端 ID 列表。

3.27.1. GitHub OAuth 配置字段

表 3.31. GitHub OAuth 字段

字段	类型	描述
FEATURE_GITHUB_LOGIN	布尔值	是否支持 GitHub 登录 **默认 : False
GITHUB_LOGIN_CONFIG	对象	使用 GitHub (企业) 作为外部登录提供程序的配置。
.ALLOWED_ORGANIZATIONS	字符串数组	GitHub (Enterprise)组织的名称列入白名单以与 ORG_RESTRICT 选项配合使用。
.API_ENDPOINT	字符串	要使用的 GitHub (Enterprise) API 的端点。必须覆盖 github.com 示例 : https://api.github.com/
.CLIENT_ID (Required)	字符串	此 Red Hat Quay 实例注册的客户端 ID ; 不能与 GITHUB_TRIGGER_CONFIG 共享。 示例 : 0e8dbe15c7630b6780
.CLIENT_SECRET (Required)	字符串	此 Red Hat Quay 实例注册的客户端 secret。 示例 : e4a58ddd3d7408b7aec109e85564a0d153d3e846
.GITHUB_ENDPOINT (Required)	字符串	GitHub (Enterprise)的端点。 示例 : https://github.com/
.ORG_RESTRICT	布尔值	如果为 true, 则只有机构白名单中的用户可以使用这个供应商登录。

3.27.2. Google OAuth 配置字段

表 3.32. Google OAuth 字段

字段	类型	描述
FEATURE_GOOGLE_LOGIN	布尔值	是否支持 Google 登录。 **Default: False

字段	类型	描述
GOOGLE_LOGIN_CONFIG	对象	使用 Google 进行外部身份验证的配置。
.CLIENT_ID (Required)	字符串	此 Red Hat Quay 实例的注册的客户端 ID。 示例： 0e8dbe15c4c7630b6780
.CLIENT_SECRET (Required)	字符串	此 Red Hat Quay 实例注册的客户端 secret。 示例： e4a58ddd3d7408b7aec109e85564a0d153d3e846

3.28. OIDC 配置字段

表 3.33. OIDC 字段

字段	类型	描述
<string>_LOGIN_CONFIG (Required)	字符串	包含 OIDC 配置设置的父键。通常，OIDC 供应商的名称，如 AZURE_LOGIN_CONFIG ，但接受任何任意字符串。
.CLIENT_ID (Required)	字符串	此 Red Hat Quay 实例的注册的客户端 ID。 示例： 0e8dbe15c4c7630b6780
.CLIENT_SECRET (Required)	字符串	此 Red Hat Quay 实例注册的客户端 secret。 示例： e4a58ddd3d7408b7aec109e85564a0d153d3e846
.DEBUGLOG	布尔值	是否启用调试。
.LOGIN_BINDING_FIELD	字符串	当内部授权设置为 LDAP 时使用。Red Hat Quay 读取此参数，并尝试使用此用户名为用户搜索 LDAP 树。如果存在，它会自动创建到该 LDAP 帐户的链接。

<code>.LOGIN_SCOPES</code>	对象	添加 Red Hat Quay 用来与 OIDC 供应商通信的其他范围。
<code>.OIDC_ENDPOINT_CUSTOM_PARAMS</code>	字符串	支持 OIDC 端点上的自定义查询参数。支持以下端点： authorization_endpoint 、 token_endpoint 和 user_endpoint 。
<code>.OIDC_ISSUER</code>	字符串	允许用户定义要验证的签发者。例如，JWT 令牌容器是一个称为 <code>iss</code> 的参数，它定义谁发出了令牌。默认情况下，这从 <code>.well-known/openid/configuration</code> 端点读取，该端点由每个 OIDC 供应商公开。如果此验证失败，则没有登录。
<code>.OIDC_SERVER</code> (Required)	字符串	用于身份验证的 OIDC 服务器地址。 : https://sts.windows.net/6c878.../
<code>.PREFERRED_USERNAME_CLAIM_NAME</code>	字符串	将首选用户名设置为令牌中的参数。
<code>.SERVICE_ICON</code>	字符串	更改登录屏幕上的图标。
<code>.SERVICE_NAME</code> (Required)	字符串	正在验证的服务名称。 示例：Microsoft Entra ID
<code>.VERIFIED_EMAIL_CLAIM_NAME</code>	字符串	用于验证用户电子邮件地址的声明名称。
<code>.PREFERRED_GROUP_CLAIM_NAME</code>	字符串	OIDC 令牌有效负载中的密钥名称，其中包含有关用户组成员资格的信息。

.OIDC_DISABLE_USER_ENDPOINT	布尔值	<p>是否允许或禁用 <code>/userinfo</code> 端点。如果使用 Azure Entra ID，则必须将此字段设置为 true，因为 Azure 从令牌中获取用户信息，而不是调用 <code>/userinfo</code> 端点。</p> <p>默认：false</p>
-----------------------------	-----	---

3.28.1. OIDC 配置

以下示例显示了 OIDC 配置示例。

OIDC 配置示例

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

3.29. 嵌套存储库配置字段

在 `FEATURE_EXTENDED_REPOSITORY_NAMES` 属性下添加了对嵌套存储库路径名称的支持。此可选配置默认添加到 `config.yaml` 中。启用允许在存储库名称中使用 `/`。

表 3.34. OCI 和嵌套软件仓库配置字段

字段	类型	描述
FEATURE_EXTENDED_REPOSITORY_NAMES	布尔值	<p>启用对嵌套软件仓库的支持</p> <p>默认值：True</p>

OCI 和嵌套存储库配置示例

FEATURE_EXTENDED_REPOSITORY_NAMES: true

3.30. QUAYINTEGRATION 配置字段

以下配置字段可用于 QuayIntegration 自定义资源：

Name	描述	模式
allowlistNamespaces (可选)	要包含的命名空间列表。	Array
clusterid (Required)	与此集群关联的 ID。	字符串
credentialsSecret.key (Required)	包含与 Quay registry 通信的凭据的机密。	对象
denylistNamespaces (可选)	要排除的命名空间列表。	Array
insecureRegistry (Optional)	是否跳过 Quay registry 的 TLS 验证	布尔值
quayHostname (Required)	Quay registry 的主机名。	字符串
scheduledImageStreamImport (Optional)	是否启用镜像流导入。	布尔值

3.31. 邮件配置字段

表 3.35. 邮件配置字段

字段	类型	描述
FEATURE_MAILING	布尔值	是否启用电子邮件 默认值： False

字段	类型	描述
MAIL_DEFAULT_SENDER	字符串	如果指定了，当 Red Hat Quay 发送电子邮件时，电子邮件地址用作 from 。如果没有，则默认为 support@quay.io 示例： support@example.com
MAIL_PASSWORD	字符串	发送电子邮件时要使用的 SMTP 密码
MAIL_PORT	Number	要使用的 SMTP 端口。如果没有指定，则默认为 587。
MAIL_SERVER	字符串	用于发送电子邮件的 SMTP 服务器。仅在 FEATURE_MAILING 设为 true 时才需要。 示例： smtp.example.com
MAIL_USERNAME	字符串	发送电子邮件时要使用的 SMTP 用户名
MAIL_USE_TLS	布尔值	如果指定了，是否使用 TLS 发送电子邮件 默认值： True

3.32. 用户配置字段

表 3.36. 用户配置字段

字段	类型	描述
FEATURE_SUPER_USERS	布尔值	是否支持超级用户 默认值： true
FEATURE_USER_CREATION	布尔值	是否可以创建用户（非超级用户） 默认值： true
FEATURE_USER_LAST_ACCESSED	布尔值	是否记录用户最后一次访问的时间 Default: true

字段	类型	描述
FEATURE_USER_LOG_ACCESS	布尔值	如果设置为 true，用户将有权访问其命名空间的审计日志 Default: false
FEATURE_USER_METADATA	布尔值	是否收集和支持用户元数据 默认值 : false
FEATURE_USERNAME_CONFIRMATION	布尔值	如果设置为 true，用户可以在通过 OpenID Connect (OIDC) 或非数据库内部身份验证供应商（如 LDAP）登录时确认和修改其初始用户名。 默认 : true
FEATURE_USER_RENAME	布尔值	如果设置为 true，用户可以重命名自己的命名空间 Default: false
FEATURE_INVITE_ONLY_USER_CREATION	布尔值	正在创建的用户是否需要被其他用户邀请 默认值 : false
FRESH_LOGIN_TIMEOUT	字符串	新登录需要用户重新输入其密码的时间 示例 : 5m
USERFILES_LOCATION	字符串	用于放置用户上传的文件的存储引擎 ID 示例 : s3_us_east
USERFILES_PATH	字符串	在存储下放置用户上传的文件的路径 示例 : userfiles
USER_RECOVERY_TOKEN_LIFETIME	字符串	恢复用户帐户的令牌长度为有效 Pattern: ^[0-9]+(w m d h s)\$ Default: 30m

字段	类型	描述
FEATURE_SUPERUSERS_FULL_ACCESS	布尔值	授予超级用户从他们没有或具有显式权限的命名空间中其他存储库中读取、写入和删除内容的权限。 Default: False
FEATURE_SUPERUSERS_ORG_CREATION_ONLY	布尔值	是否只允许超级用户创建机构。 Default: False
FEATURE_RESTRICTED_USERS	布尔值	使用 RESTRICTED_USERS_WHITELIST 设置时，受限用户无法在其自己的命名空间中创建机构或内容。例如，对于机构成员资格的普通权限，例如，受限用户仍根据他们所属的团队在机构中具有普通权限。 Default: False
RESTRICTED_USERS_WHITELIST	字符串	当使用 FEATURE_RESTRICTED_USERS: true 设置时，特定用户不包括在 FEATURE_RESTRICTED_USERS 设置中。
GLOBAL_READONLY_SUPER_USERS	字符串	设置后，授予此列表用户对所有存储库的读取访问权限，无论它们是公共存储库。

3.32.1. 用户配置字段参考

使用以下引用，使用所需的配置字段更新 `config.yaml` 文件。

3.32.1.1. FEATURE_SUPERUSERS_FULL_ACCESS 配置参考

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

3.32.1.2. GLOBAL_READONLY_SUPER_USERS 配置参考

```
---
```

```
GLOBAL_READONLY_SUPER_USERS:
```

```
- user1
```

```
---
```

3.32.1.3. FEATURE_RESTRICTED_USERS 配置参考

```
---
```

```
AUTHENTICATION_TYPE: Database
```

```
---
```

```
---
```

```
FEATURE_RESTRICTED_USERS: true
```

```
---
```

3.32.1.4. RESTRICTED_USERS_WHITELIST 配置参考

先决条件

- 在 config.yaml 文件中将 FEATURE_RESTRICTED_USERS 设置为 true。

```
---
```

```
AUTHENTICATION_TYPE: Database
```

```
---
```

```
---
```

```
FEATURE_RESTRICTED_USERS: true
```

```
RESTRICTED_USERS_WHITELIST:
```

```
- user1
```

```
---
```



注意

当设置此字段时，即使将 FEATURE_RESTRICTED_USERS 设置为 true，也可以从存储库创建机构或读取或写入内容。其他用户（如 user2、user3 和 user4）仅限于创建机构、读取或写入内容

3.33. RECAPTCHA 配置字段

表 3.37. reCAPTCHA 配置字段

字段	类型	描述
FEATURE_RECAPTCHA	布尔值	用户登录和恢复需要 Recaptcha 默认值：False
RECAPTCHA_SECRET_KEY	字符串	如果启用了 recaptcha，Recaptcha 服务的 secret 密钥

字段	类型	描述
RECAPTCHA_SITE_KEY	字符串	如果启用了 recaptcha, 则 Recaptcha 服务的站点密钥

3.34. ACI 配置字段

表 3.38. ACI 配置字段

字段	类型	描述
FEATURE_ACI_CONVERSION	布尔值	是否启用到 ACI 默认值: False
GPG2_PRIVATE_KEY_FILENAME	字符串	用于解密 ACI 的私钥的文件名
GPG2_PRIVATE_KEY_NAME	字符串	用于签署 ACI 的私钥的名称
GPG2_PUBLIC_KEY_FILENAME	字符串	用于加密 ACI 的公钥的文件名

3.35. JWT 配置字段

表 3.39. JWT 配置字段

字段	类型	描述
JWT_AUTH_ISSUER	字符串	JWT 用户的端点 Pattern: <code>^http (s)?://(.)+\$</code> 示 例: http://192.168.99.101:6060
JWT_GETUSER_ENDPOINT	字符串	JWT 用户的端点 Pattern: <code>^http (s)?://(.)+\$</code> 示例 : http://192.168.99.101:6060
JWT_QUERY_ENDPOINT	字符串	JWT 查询的端点 Pattern: <code>^http (s)?://(.)+\$</code> 示 例: http://192.168.99.101:6060

字段	类型	描述
JWT_VERIFY_ENDPOINT	字符串	JWT 验证的端点 Pattern: <code>^http (s)?://(.)+\$</code> 示 例: http://192.168.99.101:6060

3.36. 应用程序令牌配置字段

表 3.40. 应用程序令牌配置字段

字段	类型	描述
FEATURE_APP_SPECIFIC_TOKENS	布尔值	如果启用，用户可以创建令牌以供 Docker CLI Default: True
APP_SPECIFIC_TOKEN_EXPIRATION	字符串	外部应用程序令牌的过期时间。 默认 None Pattern: <code>^[0-9]+(w m d h s)\$</code>
EXPIRED_APP_SPECIFIC_TOKEN_GC	字符串	过期的外部应用程序令牌的持续时间将在垃圾回收前保留 默认值：1d

3.37. 其它配置字段

表 3.41. 其它配置字段

字段	类型	描述
ALLOW_PULLS_WITHOUT_STRICT_LOGGING	字符串	如果为 true，即使无法写入 pull audit 日志条目，也会成功拉取。如果数据库处于只读状态，这很有用，此时需要拉取(pull)以继续操作。 Default: False
AVATAR_KIND	字符串	要显示的 avatars，可以是生成的内联（本地）或 Gravatar (gravatar) Values: local, gravatar

字段	类型	描述
BROWSER_API_CALLS_XHR_ONLY	布尔值	如果启用，则只允许被标记为来自浏览器的、由 XHR 发出的 API 调用 默认： True
DEFAULT_NAMESPACE_MAXIMUM_BUILD_COUNT	Number	可以在命名空间中排队的默认构建的最大数量。 Default: None
ENABLE_HEALTH_DEBUG_SECRET	字符串	如果指定，可以向健康端点赋予一个 secret，以便在不作为超级用户进行身份验证时查看完整的调试信息
EXTERNAL_TLS_TERMINATION	布尔值	如果支持 TLS，则设为 true ，但在 Quay 之前的一个层终止。当 Quay 使用自己的 SSL 证书运行并直接接收 TLS 流量时，设置为 false 。
FRESH_LOGIN_TIMEOUT	字符串	新登录需要用户重新输入其密码的时间 示例： 5m
HEALTH_CHECKER	字符串	配置的健康检查 示例： ('RDSAwareHealthCheck' , { 'access_key' : 'foo' , 'secret_key' : 'bar' })
PROMETHEUS_NAMESPACE	字符串	应用到所有公开的 Prometheus metrics Default: quay 的前缀
PUBLIC_NAMESPACES	字符串数组	如果命名空间定义在公共命名空间列表中，则它将在 所有 用户的存储库列表页上显示，无论用户是否是命名空间的成员。通常，这供企业客户用来配置一组 "well-known" 命名空间。
REGISTRY_STATE	字符串	registry 的状态 值： normal 或 read-only

字段	类型	描述
SEARCH_MAX_RESULT_PAGE_COUNT	Number	<p>在用户限制前，用户可以在搜索中分页的最大页面数</p> <p>默认：10</p>
SEARCH_RESULTS_PER_PAGE	Number	<p>按搜索页面按页面返回的结果数</p> <p>默认值：10</p>
V2_PAGINATION_SIZE	Number	<p>V2 registry API 中每个页面返回的结果数</p> <p>默认值：50</p>
WEBHOOK_HOSTNAME_BLACKLIST	字符串数组	<p>验证超过 localhost 时禁止 Webhook 的主机名集合</p>
CREATE_PRIVATE_REPO_ON_PUSH	布尔值	<p>通过推送创建的新存储库是否设置为私有可见性</p> <p>默认值：True</p>
CREATE_NAMESPACE_ON_PUSH	布尔值	<p>新推送到不存在的机构时是否创建它</p> <p>默认：False</p>
NON_RATE_LIMITED_NAMESPACES	字符串数组	<p>如果使用 FEATURE_RATE_LIMITS 启用了速率限制，您可以为需要无限访问的特定命名空间覆盖它。</p>
FEATURE_UI_V2	布尔值	<p>设置后，允许用户尝试 beta UI 环境。</p> <p>Default: True</p>
FEATURE_REQUIRE_TEAM_INVITE	布尔值	<p>在将用户添加到团队时是否需要邀请</p> <p>默认值：True</p>
FEATURE_REQUIRE_ENCRYPTED_BASIC_AUTH	布尔值	<p>未加密的密码（而不是加密令牌）都可用于基本 auth</p> <p>默认值：False</p>

字段	类型	描述
FEATURE_RATE_LIMITS	布尔值	是否启用对 API 和 registry 端点的速率限制。将 FEATURE_RATE_LIMITS 设置为 true 会导致 nginx 将某些 API 调用限制为每秒 30 个。如果没有设置该功能，API 调用会每秒限制为 300 个（有效无限）。 Default: False
FEATURE_FIPS	布尔值	如果设置为 true，Red Hat Quay 将使用 FIPS 兼容哈希功能运行 Default: False
FEATURE_AGGREGATED_LOG_COUNT_RETRIEVAL	布尔值	是否允许检索聚合的日志计数 Default: True
FEATURE_ANONYMOUS_ACCESS	布尔值	是否允许匿名用户浏览和拉取公共存储库 默认值： True
FEATURE_DIRECT_LOGIN	布尔值	用户是否可以直接登录到 UI 默认值： True
FEATURE_LIBRARY_SUPPORT	布尔值	从 Docker 默认值拉取和推送时是否允许"无命名空间"存储库 默认值
FEATURE_PARTIAL_USER_AUTOCOMPLETE	布尔值	如果设置为 true，则自动完成将应用到部分 usernames+ Default: True
FEATURE_PERMANENT_SESSIONS	布尔值	会话是永久的 默认值： True
FEATURE_PUBLIC_CATALOG	布尔值	如果设置为 true， _catalog 端点会返回公共存储库。否则，只能返回私有存储库。 Default: False

3.38. 旧配置字段

以下字段已弃用或过时。

表 3.42. 旧配置字段

字段	类型	描述
FEATURE_BLACKLISTED_EMAILS	布尔值	如果设置为 true，则在将电子邮件域列入黑名单时不会创建新的用户帐户
BLACKLISTED_EMAIL_DOMAINS	字符串数组	如果 FEATURE_BLACKLISTED_EMAILS 设置为 true 时使用的电子邮件地址域列表为 true 示例： "example.com", "example.org"
BLACKLIST_V2_SPEC	字符串	Red Hat Quay 将响应的 Docker CLI 版本不支持 示例： <1.8.0 Default: <1.6.0
DOCUMENTATION_ROOT	字符串	文档链接的根 URL
SECURITY_SCANNER_V4_NAMESPACE_WHITELIST	字符串	应该启用安全扫描程序的命名空间
FEATURE_RESTRICTED_V1_PUSH	布尔值	如果设置为 true，则只有 V1_PUSH_WHITELIST 中列出的命名空间支持 V1 push Default: True
V1_PUSH_WHITELIST	字符串数组	如果 FEATURE_RESTRICTED_V1_PUSH 设置为 true，则支持 V1 push 的命名空间名称数组
FEATURE_HELM_OCI_SUPPORT	布尔值	启用对 Helm 工件的支持。 默认值： False

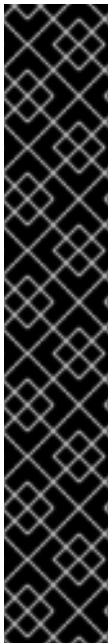
3.39. 用户界面 V2 配置字段

表 3.43. 用户界面 v2 配置字段

字段	类型	描述
FEATURE_UI_V2	布尔值	设置后，允许用户尝试 beta UI 环境。 + 默认值： False
FEATURE_UI_V2_REPO_SETTINGS	布尔值	当设置为 True 时，在 Red Hat Quay v2 UI 中启用存储库设置。 + 默认值： False

3.39.1. v2 用户界面配置

启用 `FEATURE_UI_V2` 后，您可以在用户界面的当前版本和用户界面的新版本间切换。



重要

- 这个 UI 目前处于 **beta** 阶段，可能会有变化。在其当前状态中，用户只能创建、查看和删除机构、存储库和镜像标签。
- 在旧 UI 中运行 Red Hat Quay 时，超时会话将在弹出窗口中再次输入其密码。使用新的 UI 时，用户将返回到主页面，并需要输入其用户名和密码凭证。这是一个已知问题，并将在以后的新 UI 版本中解决。
- 在传统 UI 和新 UI 之间如何报告镜像清单大小时，有一个差异。在传统 UI 中，镜像清单以兆字节为单位报告。在新 UI 中，Red Hat Quay 使用标准定义 megabyte (MB) 来报告镜像清单大小。

流程

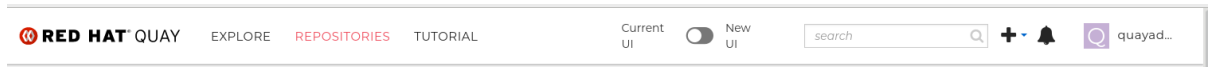
- 在部署的 `config.yaml` 文件中，添加 `FEATURE_UI_V2` 参数并将其设置为 `true`，例如：

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

- 登录到您的 Red Hat Quay 部署。

3.

在 Red Hat Quay 部署的导航窗格中，会给一个可以在 Current UI 和 New UI 之间切换的选项。点击切换按钮将其设置为新 UI，然后点 Use Beta Environment，例如：



3.40. IPV6 配置字段

表 3.44. IPv6 配置字段

字段	类型	描述
FEATURE_LISTEN_IP_VERSION	字符串	启用 IPv4、IPv6 或双栈协议系列。必须正确设置此配置字段，否则 Red Hat Quay 无法启动。 默认：IPv4 其他配置：IPv6、双栈

3.41. 品牌配置字段

表 3.45. 品牌配置字段

字段	类型	Description
品牌	对象	Red Hat Quay UI 中的徽标和 URL 的自定义品牌。
.logo (Required)	字符串	主徽标图像 URL。 标头徽标默认为 205x30 PX。 Web UI 屏幕上的表单徽标默认为 356.5x39.7 PX。 示例： /static/img/quay-horizontal-color.svg
.footer_img	字符串	UI footer 的徽标。默认为 144x34 PX。 示例： /static/img/RedHat.svg

字段	类型	Description
.footer_url	字符串	footer 镜像的链接。 示例： https://redhat.com

3.41.1. Red Hat Quay 品牌配置示例

品牌 config.yaml 示例

BRANDING:

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

3.42. 会话超时配置字段

以下配置字段依赖于相同名称的 Flask API 配置字段。

表 3.46. 会话注销配置字段

字段	类型	描述
PERMANENT_SESSION_LIFETIME	整数	用于设置永久会话的过期日期的时间增量。默认值为 31 天，这会在大约一个月后保持永久会话。 默认：2678400

3.42.1. 会话超时配置示例

以下 YAML 是启用会话生命周期时的建议配置。



重要

不建议更改会话生命周期。在设置会话超时时间时，管理员应了解分配的时间。如果您设置的时间过早，它可能会中断您的工作流。

会话超时 YAML 配置

PERMANENT_SESSION_LIFETIME: 3000

第 4 章 环境变量

Red Hat Quay 支持有限的动态配置环境变量。

4.1. GEO-REPLICATION

除存储后端外，应该使用相同的配置，但可以使用 `QUAY_DISTRIBUTED_STORAGE_PREFERENCE` 环境变量明确配置。

表 4.1. 地域复制配置

变量	类型	描述
<code>QUAY_DISTRIBUTED_STORAGE_PREFERENCE</code>	字符串	首选存储引擎（通过 <code>DISTRIBUTED_STORAGE_CONFIG</code> 中的 ID）。

4.2. 数据库连接池

Red Hat Quay 由许多不同进程组成，它们在同一容器中运行。其中许多进程与数据库交互。

如果启用，与数据库交互的每个进程都将包含一个连接池。这些每个进程的连接池配置为最多维护 20 个连接。在负载过重时，可以为 Red Hat Quay 容器中的每个进程填充连接池。在某些部署和负载下，可能需要分析以确保 Red Hat Quay 不超过配置的数据库的最大连接计数。

随着时间的推移，连接池将释放闲置连接。要立即释放所有连接，Red Hat Quay 需要重启。

可以通过将环境变量 `DB_CONNECTION_POOLING` 设置为 `true` 或 `false` 来切换数据库连接池。

表 4.2. 数据库连接池配置

变量	类型	描述
<code>DB_CONNECTION_POOLING</code>	布尔值	启用或禁用数据库连接池

如果启用了数据库连接池，则可以更改连接池的最大大小。这可以通过以下 `config.yaml` 选项完成：

`config.yaml`

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

4.3. HTTP 连接数

可以使用环境变量指定同时 HTTP 连接的数量。它们可以作为一个整体指定，也可以指定为一个特定的组件。每个进程的默认值为 50 个并行连接。

表 4.3. HTTP 连接计数配置

变量	类型	描述
WORKER_CONNECTION_COUNT	Number	同时 HTTP 连接 默认值：50
WORKER_CONNECTION_COUNT_REGISTRY	Number	同时 HTTP 连接 registry 默认值： WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_WEB	Number	Web UI 同步 HTTP 连接 默认值： WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_SECSCAN	Number	Clair 的同步 HTTP 连接 默认值： WORKER_CONNECTION_COUNT

4.4. WORKER 数量变量

表 4.4. worker 数量变量

变量	类型	描述
WORKER_COUNT	Number	进程数量的通用覆盖
WORKER_COUNT_REGISTRY	Number	指定处理 Quay 容器中 Registry 请求的进程数量 值：8 到 64 之间的整数
WORKER_COUNT_WEB	Number	指定处理容器中的 UI/Web 请求的进程数量 值：2 到 32 之间的整数
WORKER_COUNT_SECSCAN	Number	指定处理容器中的安全扫描（如 Clair）集成的进程数量 值：整数。因为 Operator 为资源请求和限值指定 2 个 vCPU，所以在 2 到 4 之间设置这个值是安全的。但是，如果保证，用户可以运行更多，例如 16 个。

4.5. 调试变量

以下调试变量在 Red Hat Quay 上提供。

表 4.5. 调试配置变量

变量	类型	描述
DEBUGLOG	布尔值	是否启用或禁用调试日志。

变量	类型	描述
USERS_DEBUG	整数 0 或 1 。	<p>用于以明文形式调试 LDAP 操作，包括密码。必须与 DEBUGLOG=TRUE 一起使用。</p>  <p>重要</p> <p>设置 USERS_DEBUG=1 以明文形式公开凭据。调试后，应该从 Red Hat Quay 部署中删除此变量。使用此环境变量生成的日志文件应该被清理，在发送给其他用户前应删除密码。请谨慎使用。</p>

第 5 章 CLAIR 安全扫描程序

5.1. CLAIR 配置概述

Clair 由一个结构化的 YAML 文件配置。每个 Clair 节点都需要指定它将在哪些模式下运行的模式，以及通过 CLI 标志或环境变量到配置文件的路径。例如：

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

或者

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

上述命令会使用相同的配置文件启动两个 Clair 节点。一个运行索引工具，另一个则运行匹配的设施。

如果您以组合模式运行 Clair，则必须在配置中提供索引器、匹配器和通知程序配置块。

5.1.1. 在代理环境中使用 Clair 的信息

如果需要，可以指定 Go 标准库相关的环境变量，例如：

-

HTTP_PROXY

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

-

HTTPS_PROXY.

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

-

SSL_CERT_DIR.

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

如果您使用带有 Clair 更新器 URL 的环境中的代理服务器，您必须识别哪些 URL 需要添加到代理允许列表中，以确保 Clair 可以访问它们。例如：osv updater 需要访问 <https://osv->

`vulnerabilities.storage.googleapis.com` 来获取生态系统数据转储。在这种情况下，URL 必须添加到代理允许列表中。有关 updater URL 的完整列表，请参阅“Clair updater URL”。

您还必须确保将标准 Clair URL 添加到代理允许列表中：

- `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`

在配置代理服务器时，请考虑启用 Clair 和这些 URL 之间的无缝通信所需的任何身份验证要求或特定的代理设置。通过全面记录并解决这些注意事项，您可以确保在通过代理路由更新器流量的同时，有效处理 Clair 功能。

5.1.2. Clair 配置参考

以下 YAML 显示了一个 Clair 配置示例：

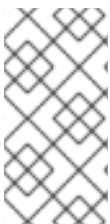
```
http_listen_addr: ""
introspection_addr: ""
log_level: ""
tls: {}
indexer:
  connstring: ""
  scanlock_retry: 0
  layer_scan_concurrency: 5
  migrations: false
  scanner: {}
  airgap: false
matcher:
  connstring: ""
  indexer_addr: ""
  migrations: false
  period: ""
  disable_updaters: false
  update_retention: 2
matchers:
  names: nil
```



```

config: nil
updaters:
  sets: nil
  config: nil
notifier:
  connstring: ""
  migrations: false
  indexer_addr: ""
  matcher_addr: ""
  poll_interval: ""
  delivery_interval: ""
  disable_summary: false
  webhook: null
  amqp: null
  stomp: null
auth:
  psk: nil
trace:
  name: ""
  probability: null
jaeger:
  agent:
    endpoint: ""
  collector:
    endpoint: ""
    username: null
    password: null
  service_name: ""
  tags: nil
  buffer_max: 0
metrics:
  name: ""
  prometheus:
    endpoint: null
  dogstatsd:
    url: ""

```



注意

为了完整，以上 YAML 文件列出了每个键。按照原样使用此配置文件将导致一些选项正常设置它们的默认值。

5.1.3. Clair 常规字段

下表描述了 Clair 部署可用的常规配置字段。

字段	Typhttp_listen _ae	描述
http_listen_addr	字符串	配置公开 HTTP API 的位置。 默认： :6060
introspection_addr	字符串	配置 Clair 的指标和健康端点公开的位置。
log_level	字符串	设置日志记录级别。需要以下字符串之一： debug-color,debug,info,warn,error,fatal,panic
tls	字符串	包含提供 TLS/SSL 和 HTTP/2 的 HTTP API 配置的映射。
.cert	字符串	要使用的 TLS 证书。必须是 full-chain 证书。

常规 Clair 字段的配置示例

以下示例显示了 Clair 配置。

常规 Clair 字段的配置示例

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

5.1.4. Clair indexer 配置字段

下表描述了 Clair 的 indexer 组件的配置字段。

字段	类型	描述
indexer	对象	提供 Clair 索引节点配置。

字段	类型	描述
.airgap	布尔值	为 indexers 和 fetchers 禁用对互联网的 HTTP 访问。允许私有 IPv4 和 IPv6 地址。数据库连接不受影响。
.connstring	字符串	Postgres 连接字符串。接受格式作为 URL 或 libpq 连接字符串。
.index_report_request_concurrency	整数	速率限制索引报告创建请求的数量。把它设置为 0 ，以自动调整这个值。设置负值表示无限。自动大小是可用内核数的倍数。 如果超过并发，API 会返回 429 状态代码。
.scanlock_retry	整数	一个正整数，代表秒。并发索引器锁定在清单扫描上，以避免冲突。这个值调整等待索引器轮询锁定的频率。
.layer_scan_concurrency	整数	正整数限制并发层扫描的数量。Indexers 将同时匹配清单的层。这个值调整索引程序并行扫描的层数。
.migrations	布尔值	索引节点是否处理到其数据库的迁移。
.scanner	字符串	索引器配置。 扫描程序允许将配置选项传递给层扫描程序。如果设计这样做，扫描程序会将此配置传递给它。
.scanner.dist	字符串	带有特定扫描程序名称和任意 YAML 作为值的映射。
.scanner.package	字符串	带有特定扫描程序名称和任意 YAML 作为值的映射。
.scanner.repo	字符串	带有特定扫描程序名称和任意 YAML 作为值的映射。

indexer 配置示例

以下示例显示了 Clair 的 hypothetical indexer 配置。

indexer 配置示例

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

5.1.5. Clair matcher 配置字段

下表描述了 Clair 的 matcher 组件的配置字段。



注意

与 匹配器 配置字段不同。

字段	类型	描述
matcher	对象	提供 Clair 匹配节点配置。
.cache_age	字符串	控制提示用户缓存响应的时长。
.connstring	字符串	Postgres 连接字符串。接受格式作为 URL 或 libpq 连接字符串。
.max_conn_pool	整数	限制数据库连接池大小。 Clair 允许自定义连接池大小。这个数字直接设定同时允许的活跃数据库连接的数量。 以后的版本将忽略此参数。用户应该通过连接字符串进行配置。
.indexer_addr	字符串	匹配者联系索引程序来创建漏洞报告。需要此索引器的位置。 默认值为 30m 。

字段	类型	描述
.migrations	布尔值	匹配节点是否处理到其数据库的迁移。
.period	字符串	决定新安全公告的更新频率。 默认值为 30m 。
.disable_updaters	布尔值	是否运行后台更新。 Default: False
.update_retention	整数	设置在垃圾回收周期之间保留的更新操作数量。这应该根据数据库大小限制设置为安全 MAX 值。 默认值为 10m 。 如果提供了小于 0 的值，则禁用垃圾回收。 2 是最低值，可确保与通知进行比较更新。

matcher 配置示例

matcher 配置示例

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

5.1.6. Clair matchers 配置字段

下表描述了 Clair 的 matchers 组件的配置字段。



注意

与匹配配置字段不同。

表 5.1. matchers 配置字段

字段	类型	描述
matchers	字符串数组	为树内 匹配器 提供配置。
.names	字符串	一个字符串值列表，用于告知匹配者工厂关于启用的匹配者信息。如果值设为 null ，则默认匹配者列表将运行。以下字符串被接受： alpine-matcher,aws-matcher,debian-matcher,gobin,java-maven,oracle,photon,python,rhel,rhel -container-matcher,ruby,suse,suse,ubuntu-matcher
.config	字符串	为特定匹配器提供配置。 一个映射键是包含子对象的 matcher 名称键 ，该对象将提供给匹配者工厂构造器。例如：

matchers 配置示例

以下示例显示了一个假设的 Clair 部署，它只需要 **alpine,aws,debian,oracle matchers**。

matchers 配置示例

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

5.1.7. Clair updaters 配置字段

下表描述了 Clair 的 updaters 组件的配置字段。

表 5.2. 更新器配置字段

字段	类型	描述
updaters	对象	为 matcher 的更新管理器提供配置。
.sets	字符串	<p>一个值列表，告知更新管理器要运行的更新程序。</p> <p>如果值设为 null，则默认更新程序将运行以下内容： alpine,aws,clair.cvss,debian,oracle,photon,osv,rhel,rhccsuse,0.11.0-</p> <p>如果留空，则运行零更新程序。</p>
.config	字符串	<p>为特定更新器集提供配置。</p> <p>由 updater 集名称的映射键包含子对象，该对象将提供给 updater 设置的构造器。有关每个 updater 的子对象列表，请参阅“高级更新器配置”。</p>

updaters 配置示例

在以下配置中，仅配置 rhel 集。也定义了特定于 rhel 更新器的 `ignore_unpatched` 变量。

updaters 配置示例

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

5.1.8. Clair 通知程序配置字段

Clair 的一般通知程序配置字段如下。

字段	类型	描述
通知程序	对象	提供 Clair 通知程序节点配置。
.connstring	字符串	postgres 连接字符串。接受格式为 URL 或 libpq 连接字符串。
.migrations	布尔值	通知节点是否处理到其数据库的迁移。
.indexer_addr	字符串	通知程序联系一个索引程序，以创建或获取受漏洞影响的清单。需要此索引器的位置。
.matcher_addr	字符串	通知程序联系一个匹配程序，以列出更新操作并获得 diffs。需要此匹配器的位置。
.poll_interval	字符串	通知程序将查询匹配者更新操作的频率。
.delivery_interval	字符串	通知程序尝试发送创建或之前失败的通知的频率。
.disable_summary	布尔值	控制是否应将通知总结为每个清单中的一个。

通知程序配置示例

以下 通知程序 片断用于最小配置。

通知程序配置示例

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



```

migrations: true
poll_interval: 15s
webhook:
  target: "http://webhook/"
  callback: "http://clair-notifier/notifier/api/v1/notifications"
  headers: ""
amqp: null
stomp: null
# ...

```

5.1.8.1. Clair Webhook 配置字段

以下 Webhook 字段可用于 Clair 通知程序环境。

表 5.3. Clair Webhook 字段

.webhook	对象	配置用于 Webhook 发送的通知程序。
.webhook.target	字符串	提供 webhook 的 URL。
.webhook.callback	字符串	可以检索通知的回调 URL。通知 ID 将附加到此 URL 中。 这通常是托管 Clair 通知程序的位置。
.webhook.headers	字符串	将标头名称与值列表关联的映射。

Webhook 配置示例

Webhook 配置示例

```

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

```

5.1.8.2. Clair amqp 配置字段

以下高级消息队列协议(AMQP)字段可用于 Clair 通知程序环境。

<code>.amqp</code>	对象	配置用于 AMQP 发送的通知程序。 [NOTE] ==== Clair 不自行声明任何 AMQP 组件。所有尝试使用交换或队列的尝试都是仅被动的，并将失败。代理管理员应提前设置交换和队列。 ===
<code>.amqp.direct</code>	布尔值	如果为 true ，则通知程序会将单个通知（而非回调）发送到配置的 AMQP 代理。
<code>.amqp.rollup</code>	整数	当将 amqp.direct 设置为 true 时，这个值告知通知要直接发送多少通知。例如，如果 direct 设置为 true ，并且 amqp.rollup 设置为 5 ，则通知程序不会在单个 JSON 有效负载向代理中提供 5 个通知。将值设为 0 实际的效果是将其设置为 1 。
<code>.amqp.exchange</code>	对象	要连接的 AMQP 交换。
<code>.amqp.exchange.name</code>	字符串	要连接的交换的名称。
<code>.amqp.exchange.type</code>	字符串	交换的类型。通常，以下之一： 直接 、 发出 、 主题 、 标题 。
<code>.amqp.exchange.durability</code>	布尔值	配置的队列是否持久。
<code>.amqp.exchange.auto_delete</code>	布尔值	配置的队列是否使用 auto_delete_policy 。
<code>.amqp.routing_key</code>	字符串	每个通知发送的路由密钥的名称。
<code>.amqp.callback</code>	字符串	如果 amqp.direct 设置为 false ，则会在发送到代理的通知回调中提供此 URL。此 URL 应指向 Clair 的通知 API 端点。
<code>.amqp.uris</code>	字符串	要连接的一个或多个 AMQP 代理的列表，按优先级顺序连接。

.amqp.tls	对象	配置 TLS/SSL 连接到 AMQP 代理。
.amqp.tls.root_ca	字符串	可以读取 root CA 的文件系统路径。
.amqp.tls.cert	字符串	可读取 TLS/SSL 证书的文件系统路径。 [NOTE] ===== Clair 还允许 SSL_CERT_DIR , 如 Go crypto/x509 软件包所记录。 =====
.amqp.tls.key	字符串	可以读取 TLS/SSL 私钥的文件系统路径。

AMQP 配置示例

以下示例显示了 Clair 的 hypothetical AMQP 配置。

AMQP 配置示例

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

5.1.8.3. Clair STOMP 配置字段

Clair 通知程序环境提供了以下简单文本导向型消息协议(STOMP)字段。

.stomp	对象	为 STOMP 交付配置通知。
.stomp.direct	布尔值	如果为 true ，则通知程序会向配置的 STOMP 代理提供单独的通知（而非回调）。
.stomp.rollup	整数	如果 stomp.direct 设为 true ，则该值限制在单个直接发送中发送的通知数量。例如，如果 direct 设为 true ，并且 滚动 设置为 5 ，则通知程序不会在单个 JSON 有效负载中向代理发送 5 个通知。将值设为 0 实际的效果是将其设置为 1 。
.stomp.callback	字符串	如果 stomp.callback 设置为 false ，则通知回调中提供的 URL 将发送到代理。此 URL 应指向 Clair 的通知 API 端点。
.stomp.destination	字符串	将通知发送到的 STOMP 目的地。
.stomp.uris	字符串	要以优先级顺序连接到的一个或多个 STOMP 代理的列表。
.stomp.tls	对象	配置 TLS/SSL 连接到 STOMP 代理。
.stomp.tls.root_ca	字符串	可以读取 root CA 的文件系统路径。 [NOTE] ===== Clair 还尊重 SSL_CERT_DIR ，如 Go crypto/x509 软件包所记录。 =====
.stomp.tls.cert	字符串	可读取 TLS/SSL 证书的文件系统路径。
.stomp.tls.key	字符串	可以读取 TLS/SSL 私钥的文件系统路径。
.stomp.user	字符串	配置 STOMP 代理的登录详情。
.stomp.user.login	字符串	要连接的 STOMP 登录。
.stomp.user.passcode	字符串	要连接的 STOMP passcode。

STOMP 配置示例

以下示例显示了 Clair 的 hypothetical STOMP 配置。

STOMP 配置示例

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

5.1.9. Clair 授权配置字段

以下授权配置字段可用于 Clair。

字段	类型	描述
auth	对象	定义 Clair 的外部 and 基于服务 JWT 的身份验证。如果定义了多个 auth 机制，Clair 会选择一个。目前，不支持多个机制。
.psk	字符串	定义预共享密钥身份验证。
.psk.key	字符串	在所有方签名和验证 JWT 之间的共享 base64 编码密钥。
.psk.iss	字符串	要验证的 JWT 签发者列表。空列表接受 JWT 声明中的任何签发者。

授权配置示例

以下 授权 片断用于最小配置。

授权配置示例

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

5.1.10. Clair trace 配置字段

以下 trace 配置字段可用于 Clair。

字段	类型	描述
trace	对象	基于 OpenTelemetry 定义分布式追踪配置。
.name	字符串	应用程序跟踪的名称将属于。
.probability	整数	可能会出现 trace 的概率。
.jaeger	对象	为 Jaeger tracing 定义值。
.jaeger.agent	对象	定义用于配置发送到 Jaeger 代理的值。
.jaeger.agent.endpoint	字符串	可提交 trace 的 <host> : <post > 语法中的地址。
.jaeger.collector	对象	为配置发送到 Jaeger 收集器定义值。
.jaeger.collector.endpoint	字符串	可提交 trace 的 <host> : <post > 语法中的地址。
.jaeger.collector.username	字符串	Jaeger 用户名。

字段	类型	描述
.jaeger.collector.password	字符串	Jaeger 密码。
.jaeger.service_name	字符串	在 Jaeger 中注册的服务名称。
.jaeger.tags	字符串	用于提供额外的元数据的键值对。
.jaeger.buffer_max	整数	在发送到 Jaeger 后端以进行存储和分析前可以缓冲的最大 span 数量。

trace 配置示例

以下示例显示了 Clair 的 hypothetical trace 配置。

trace 配置示例

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

5.1.11. Clair 指标配置字段

以下指标配置字段可用于 Clair。

字段	类型	描述
metrics	对象	基于 OpenTelemetry 定义分布式追踪配置。
.name	字符串	使用的指标的名称。
.prometheus	字符串	配置 Prometheus 指标导出器。

字段	类型	描述
<code>.prometheus.endpoint</code>	字符串	定义提供指标的路径。

指标配置示例

以下示例显示了 Clair 的 hypothetical 指标配置。

指标配置示例

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