



Red Hat Quay 3.6

配置 Red Hat Quay

使用配置选项自定义 Red Hat Quay

Red Hat Quay 3.6 配置 Red Hat Quay

使用配置选项自定义 Red Hat Quay

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律通告

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Configure_Red_Hat_Quay.ent file |.

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 配置入门	6
1.1. QUAY 3.6 的配置更新	6
1.1.1. 新配置字段	6
1.1.2. 弃用的配置字段	6
1.2. 编辑配置文件	6
1.3. 在独立部署中配置文件的位置	7
1.4. 使用命令行和 API 在 OPENSIFT 上配置 QUAY	7
1.4.1. 确定 QuayRegistry 端点和 secret	7
1.4.2. 下载现有配置	8
1.4.3. 使用配置捆绑包配置自定义 SSL 证书	10
1.5. 最小配置	11
1.5.1. 最小配置文件示例	11
1.5.2. 本地存储	12
1.5.3. 云存储	12
第 2 章 配置字段	13
2.1. 所需的配置字段	13
2.2. 自动化选项	13
2.3. 可选配置字段	13
2.4. 常规必填字段	14
2.5. 数据库配置	15
2.5.1. 数据库 URI	15
2.5.2. 数据库连接参数	15
2.5.2.1. PostgreSQL SSL 连接参数	15
2.5.2.2. MySQL SSL 连接参数	16
2.6. 镜像存储	16
2.6.1. 镜像存储功能	16
2.6.2. 镜像存储配置字段	16
2.6.3. 本地存储	17
2.6.4. OCS/NooBaa	17
2.6.5. Ceph / RadosGW Storage / Hitachi HCP	18
2.6.6. AWS S3 存储	18
2.6.7. Google 云存储	18
2.6.8. Azure 存储	18
2.6.9. Swift 存储	19
2.7. REDIS 配置字段	19
2.7.1. 构建日志	19
2.7.2. 用户事件	20
2.7.3. redis 配置示例	20
2.8. 标签过期选项	20
2.9. 预配置 QUAY 以实现自动化	21
2.9.1. 允许 API 创建第一个用户	21
2.9.2. 启用常规 API 访问	21
2.9.3. 添加超级用户	22
2.9.4. 限制用户创建	22
2.9.5. 推荐的自动化配置	22
2.9.6. 使用初始配置部署 Operator	22
2.9.7. 使用 API 创建第一个用户	22
2.9.7.1. 调用 API	23
2.9.7.2. 使用 OAuth 令牌	23
2.9.7.2.1. 创建机构	24

2.9.7.2.2. 获取机构详情	24
2.10. 基本配置	25
2.11. SSL 配置字段	26
2.11.1. 配置 SSL	27
2.12. 在 RED HAT QUAY CONTAINER 中添加 TLS 证书	28
2.12.1. 在 Red Hat Quay 中添加 TLS 证书	28
2.13. LDAP 配置字段	28
2.13.1. LDAP 配置示例	30
2.14. 镜像配置字段	30
2.15. 安全扫描程序配置字段	30
2.16. OCI 和 HELM 配置	32
2.17. 操作日志配置字段	32
2.17.1. 操作日志存储配置	32
2.17.2. 操作日志轮转和归档配置	34
2.18. 构建日志	35
2.19. DOCKERFILE 构建触发器字段	35
2.19.1. GitHub 构建触发器	36
2.19.2. Bitbucket 构建触发器	37
2.19.3. GitLab 构建触发器	37
2.20. OAUTH 配置	37
2.20.1. GitHub OAuth	38
2.20.2. Google OAuth	39
2.21. 配置嵌套软件仓库	39
2.22. 向 QUAY 添加其他 OCI 介质类型	40
2.23. 邮件配置	40
2.24. 用户配置字段	41
2.25. RECAPTCHA 配置	43
2.26. ACI 配置	43
2.27. JWT 配置	43
2.28. 应用程序令牌	44
2.29. 其他字段	44
2.30. 旧配置字段	46
第 3 章 环境变量	48
3.1. GEO-REPLICATION	48
3.2. 数据库连接池	48
3.3. HTTP 连接计数	48
3.4. WORKER 计数变量	49
第 4 章 使用配置工具在 OPENSIFT 中重新配置 QUAY	50
4.1. 访问配置编辑器	50
4.1.1. 检索配置编辑器凭证	50
4.1.2. 登录到配置编辑器	51
4.1.3. 更改配置	52
4.2. 在 UI 中监控重新配置	53
4.2.1. QuayRegistry 资源	53
4.2.2. 事件	55
4.3. 在重新配置后访问更新的信息	56
4.3.1. 在 UI 中访问更新的配置工具凭证	56
4.3.2. 在 UI 中访问更新的 config.yaml	56
第 5 章 QUAY OPERATOR 组件	58
5.1. 使用受管组件	58
5.2. 将非受管组件用于依赖项	59

5.2.1. 使用现有的 Postgres 数据库	59
5.2.2. NooBaa 非受管存储	59
5.2.3. 禁用 Horizontal Pod Autoscaler	60
5.3. 在 KUBERNETES 上部署时添加证书	60
5.4. 使用 OPERATOR 配置 OCI 和 HELM	61
5.5. 卷大小覆盖	62
第 6 章 使用配置 API	63
6.1. 检索默认配置	63
6.2. 检索当前配置	63
6.3. 使用 API 验证配置	64
6.4. 确定必填字段	64
第 7 章 使用配置工具	66
7.1. 自定义 SSL 证书 UI	66
7.2. 基本配置	66
7.2.1. 联系信息	66
7.3. 服务器配置	67
7.3.1. 服务器配置选择	67
7.3.2. TLS 配置	67
7.4. 数据库配置	68
7.4.1. PostgreSQL 配置	68
7.5. 数据一致性	69
7.6. 时间机器配置	69
7.7. REDIS 配置	70
7.8. 存储库镜像配置	70
7.9. REGISTRY 存储配置	70
7.9.1. 启用存储复制	70
7.9.2. 存储引擎	71
7.9.2.1. 本地存储	71
7.9.2.2. Amazon S3 存储	71
7.9.2.3. Azure blob 存储	72
7.9.2.4. Google 云存储	72
7.9.2.5. Ceph 对象网关(RADOS)存储	73
7.9.2.6. OpenStack(Swift)存储配置	73
7.9.2.7. CloudFront + Amazon S3 存储配置	74
7.10. 操作日志配置	75
7.10.1. 操作日志存储配置	75
7.10.1.1. 数据库操作日志存储	75
7.10.1.2. Elasticsearch 操作日志存储	75
7.10.2. 操作日志轮转和归档	76
7.11. 安全扫描程序配置	76
7.12. 应用程序 REGISTRY 配置	77
7.13. 电子邮件配置	77
7.14. 内部验证配置	77
7.14.1. LDAP	78
7.14.2. Keystone (OpenStack 身份)	78
7.14.3. JWT 自定义身份验证	79
7.14.4. 外部应用程序令牌	79
7.15. 外部验证(OAUTH)配置	80
7.15.1. GitHub (企业) 身份验证	80
7.15.2. Google 身份验证	80
7.16. 访问设置配置	81

7.17. DOCKERFILE 构建支持	81
7.17.1. GitHub (企业) 构建触发器	82
7.17.2. Bitbucket 构建触发器	82
7.17.3. GitLab 构建触发器	83

第 1 章 配置入门

Red Hat Quay 可以作为独立部署，或使用 Operator 部署到现有的 OpenShift 集群上。您用来创建、检索、更新和验证 Red Hat Quay 配置的方法稍有不同，具体取决于您使用的部署类型。但是，核心配置选项对于所有类型的部署都基本相同，可以操作这些选项：

- 通过编辑 config.yaml 文件直接编辑 **config.yaml** 文件。请参阅 [编辑配置文件的](#)部分。
- 以编程方式使用配置 API。请参阅使用 [配置 API](#) 的部分。
- 在视觉上，使用配置工具 UI。请参阅使用 [配置工具](#) 的部分。

您可以使用 Operator 在 OpenShift 上安装 Quay，而无需提供任何初始配置，因为 Operator 提供了不合理的默认值来部署 registry。但是，对于独立部署，您必须提供最小配置级别，然后才能启动 registry。最小的要求可以使用 [配置 API](#) 来决定，并记录在 [部分](#)

使用初始配置部署 Quay 后，您应该从正在运行的系统中检索并保存完整的配置，因为它可能包含额外的、生成值，在重新启动或升级系统时，您会在未来需要用到的值。

1.1. QUAY 3.6 的配置更新

1.1.1. 新配置字段

- **FEATURE_EXTENDED_REPOSITORY_NAMES**：支持嵌套存储库和扩展仓库名称。此更改允许使用某些 OpenShift Container Platform 用例所需的存储库名称 `/`。如需更多信息，请参阅 [配置嵌套软件仓库](#)
- **FEATURE_USER_INITIALIZE**：如果设置为 true，可以通过 API `/api/v1/user/initialize` 创建第一个用户帐户。如需更多信息，请参阅 [预配置 Quay 实现自动化](#)
- **ALLOWED_OCI_ARTIFACT_TYPES**：Helm、cosign 和 ztsd 压缩方案工件默认构建在 Red Hat Quay 3.6 中。对于默认不支持的任何其他 OCI 介质类型，您可以将它们添加到 Quay **config.yaml** 中的 **ALLOWED_OCI_ARTIFACT_TYPES** 配置中，如需更多信息，请参阅 [将其他 OCI 介质类型添加到 Quay](#) 中
- **CREATE_PRIVATE_REPO_ON_PUSH**：Registry 用户现在在其 config.yaml 中将 **CREATE_PRIVATE_REPO_ON_PUSH** 设置为 **True** 或 **False**。
- **CREATE_NAMESPACE_ON_PUSH**：现在可被配置为自动创建该机构。

1.1.2. 弃用的配置字段

- **FEATURE_HELM_OCI_SUPPORT**：这个选项已弃用，并将在以后的 Red Hat Quay 版本中删除。在 Red Hat Quay 3.6 中，默认支持 Helm 工件，并包括在 **FEATURE_GENERAL_OCI_SUPPORT** 属性中。用户不再需要更新其 config.yaml 文件来启用支持。

1.2. 编辑配置文件

在独立模式下部署 registry 需要最小配置 - 请参阅 [第 1.5 节“最小配置”](#) 部分。

在 registry 启动时验证配置文件，在输出中会突出显示任何问题：

可以使用配置 API 来验证配置，但这需要以 config 模式启动 Quay 容器

要使更改生效，需要重启 registry。

1.3. 在独立部署中配置文件的位置

对于独立部署，必须在启动 Quay registry 时指定 **config.yaml** 文件。此文件位于配置卷中，因此在以下示例中，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:v3.6.8
```

1.4. 使用命令行和 API 在 OPENSIFT 上配置 QUAY

部署后，您可以通过编辑 Quay 配置捆绑包 secret **spec.configBundleSecret** 配置 Quay 应用程序，您也可以更改 QuayRegistry 资源的 **spec.components** 对象中的组件的受管状态。

Operator 不会监视 **spec.configBundleSecret** 资源是否有更改，因此建议对新的 **Secret** 资源进行配置更改，并且 **spec.configBundleSecret** 字段已更新，以反映更改。如果与新配置存在问题，则简单地将 **spec.configBundleSecret** 的值恢复到旧的 **Secret**。

更改配置的步骤涉及：

1. 确定当前的端点和 secret
2. 如果已在 OpenShift 上部署了 Red Hat Quay，请下载现有配置捆绑包
3. 创建或更新 **config.yaml** 配置文件
4. 装配 Quay 所需的任何 SSL 证书或服务所需的自定义 SSL 证书
5. 使用配置文件和任何证书创建新配置捆绑包 secret
6. 创建或更新 registry，引用新的配置捆绑包 secret，并指定任何覆盖来管理组件
7. 监控部署以确保成功完成并且配置更改生效

另外，您可以使用配置编辑器 UI 来配置 Quay 应用，如 [第 4 章 使用配置工具在 OpenShift 中重新配置 Quay](#) 部分所述。

1.4.1. 确定 QuayRegistry 端点和 secret

您可以使用 **oc describe quayregistry** 或 **oc get quayregistry -o yaml** 来检查 QuayRegistry 资源，以确定当前的端点和 secret：

```
$ oc get quayregistry example-registry -n quay-enterprise -o yaml

apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  ...
  name: example-registry
  namespace: quay-enterprise
```

```

...
spec:
  components:
    ...
    configBundleSecret: example-registry-quay-config-bundle-fjpnm
status:
  configEditorCredentialsSecret: example-registry-quay-config-editor-credentials-kk55dc7299
  configEditorEndpoint: https://example-registry-quay-config-editor-quay-
enterprise.apps.docs.quayteam.org
  currentVersion: 3.6.0
  lastUpdated: 2021-09-21 11:18:13.285192787 +0000 UTC
  registryEndpoint: https://example-registry-quay-quay-enterprise.apps.docs.quayteam.org
  unhealthyComponents: {}

```

相关字段为：

- **registryEndpoint** : registry的 URL，用于浏览器访问 registry UI，以及 registry API 端点
- **configBundleSecret**: config bundle secret，其中包含 **config.yaml** 文件和任何 SSL 证书
- **configEditorEndpoint** : 配置编辑器工具的 URL，用于浏览器访问配置工具，以及配置 API
- **configEditorCredentialsSecret**: 包含用户名的 secret（通常是 **quayconfig**）以及配置编辑器工具的密码

要确定配置编辑器工具的用户名和密码：

1. 检索 secret：

```

$ oc get secret -n quay-enterprise example-registry-quay-config-editor-credentials-
kk55dc7299 -o yaml

apiVersion: v1
data:
  password: SkZwQkVKTUN0a1BUZmp4dA==
  username: cXVheWNvbmZpZw==
kind: Secret

```

2. 解码用户名：

```

$ echo 'cXVheWNvbmZpZw==' | base64 --decode

quayconfig

```

3. 解码密码：

```

$ echo 'SkZwQkVKTUN0a1BUZmp4dA==' | base64 --decode

JFpBEJMCtkPTfjxt

```

1.4.2. 下载现有配置

访问当前配置的方法有很多：

1. 使用配置编辑器端点，为配置编辑器指定用户名和密码：

```
$ curl -k -u quayconfig:JFpBEJMCtkPTfjxt https://example-registry-quay-config-editor-quay-enterprise.apps.docs.quayteam.org/api/v1/config
```

```
{
  "config.yaml": {
    "ALLOW_PULLS_WITHOUT_STRICT_LOGGING": false,
    "AUTHENTICATION_TYPE": "Database",
    ...
    "USER_RECOVERY_TOKEN_LIFETIME": "30m"
  },
  "certs": {
    "extra_ca_certs/service-ca.crt":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURVVENQWptZ0F3SUJBZ0lJRE9k
WFhuUXFjMUF3RFFZSkVWklodmNOQVFFTEJRQXdOakUwTURJR0ExVUUKQXd3cmIzQ
mxibk5vYVdaMExYTmxjblpwWTJVdGMvVnlkbWx1WnkxemFXZHVhWEpBTVRZek1UYzNPRE
V3TXpBZQpGdzB5TVRBNU1UWXdoeIF4TkRKYUZ..."
  }
}
```

2. 使用 config bundle secret

a. 获取 secret 数据：

```
$ oc get secret -n quay-enterprise example-registry-quay-config-bundle-jkfh8 -o
jsonpath='{.data}'
```

```
{
  "config.yaml":
"QUxMT1dfUFVMTFNfV0lUSE9VVF9TVFJJQ1RfTE9HR0lORz0gZmFsc2UKQVUUSEVO
VEIDQVRJT05fVFIQRTogRGF0YWJhc2UKQVZBVEFSX0tJTkQ6IGxvY2FsCkRBVEFCQ
VNFx1NFQ1JFVF9LRVh6IWhhIOEc1VDBNbklkaGxNQzNkTjd3MWR5WWxwVmo0a0R2enl
xZ3l6Ulp5ZjFpODBmWWU3VDUxU1FPZ3hXelpocFlqYlVxNzRkKaDIIVVVEVWpyCkRFR
...
OgotIDJ3CIRFQU1fUkVTVWU5DX1NUQUxFX1RJTUU6IDYwbQpURVNUSU5HOiBmYWx
zZQpVU0VSX1JFQ09WRVJZX1RPS0VOX0xJRkVUSU1FOiAzMG0K",
  "extra_ca_cert_service-ca.crt":
"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURVVENQWptZ0F3SUJBZ0lJR
E9kWFhuUXFjMUF3RFFZSkVWklodmNOQVFFTEJRQXdOakUwTURJR0ExVUUKQXd3
cmIzQmxibk5vYVdaMExYTmxjblpwWTJVdGMvVnlkbWx1WnkxemFXZHVhWEpBTVRZek1
UYzNPREV3TXpBZQpGdzB5TVRBNU1UWXdoeIF4TkRKYUZ3MHI
...
XSW1jaApkQXZTWGpFUnZOZEZzN3pHK1VzTmZwN0ZlQkJVWkY4L2RZnWJCR2Mw
WTVaY0J6bFNjQT09Ci0tLS0tRU5EIEFUIRJRkIDQVRFLS0tLS0K"
}
```

b. 解码数据：

```
$ echo 'QUxMT1dfUFVMTFN...U1FOiAzMG0K' | base64 --decode
```

```
ALLOW_PULLS_WITHOUT_STRICT_LOGGING: false
AUTHENTICATION_TYPE: Database
...
TAG_EXPIRATION_OPTIONS:
- 2w
```

```
TEAM_RESYNC_STALE_TIME: 60m
TESTING: false
USER_RECOVERY_TOKEN_LIFETIME: 30m
```

1.4.3. 使用配置捆绑包配置自定义 SSL 证书

您可以通过创建新的配置捆绑包 secret，在初始部署前或在 OpenShift 上部署 Red Hat Quay 后配置自定义 SSL 证书。如果您要在现有部署中添加证书，则必须在新 config bundle secret 中包含完整的 **config.yaml**，即使您没有进行任何配置更改。

1. 使用嵌入式数据或使用文件创建 secret :
 - a. 直接将配置详情嵌入到 Secret 资源 YAML 文件中，例如 :

custom-ssl-config-bundle.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: custom-ssl-config-bundle-secret
  namespace: quay-enterprise
data:
  config.yaml: |
    ALLOW_PULLS_WITHOUT_STRICT_LOGGING: false
    AUTHENTICATION_TYPE: Database
    ...
  extra_ca_cert_my-custom-ssl.crt: |
    -----BEGIN CERTIFICATE-----
    MIIDsDCCApigAwIBAgIUcQlzkHjF5i5TXLFy+sepFrZr/UswDQYJKoZIhvcNAQEL
    BQAwbzELMAkGA1UEBhMCSUUxDzANBgNVBAgMBkdbBTfDbWTEPMA0GA1UEBwwG
    R0FM
    ....
    -----END CERTIFICATE-----
```

接下来，从 YAML 文件创建 secret :

```
$ oc create -f custom-ssl-config-bundle.yaml
```

- b. 另外，您可以创建包含所需信息的文件，然后从这些文件创建 secret :

```
$ oc create secret generic custom-ssl-config-bundle-secret \
  --from-file=config.yaml \
  --from-file=extra_ca_cert_my-custom-ssl.crt=my-custom-ssl.crt
```

2. 创建或更新 QuayRegistry YAML 文件 **quayregistry.yaml**，引用所创建的 Secret，例如 :

quayregistry.yaml

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
```

```
namespace: quay-enterprise
spec:
  configBundleSecret: custom-ssl-config-bundle-secret
```

3. 使用 YAML 文件部署或更新 registry :

```
oc apply -f quayregistry.yaml
```

1.5. 最小配置

对于独立部署，以下功能需要配置选项：

- 服务器主机名
- HTTP 或 HTTPS
- 身份验证类型，如 Database 或 LDAP
- 用于加密数据的机密密钥
- 镜像存储
- 元数据的数据库
- Redis 用于构建日志和用户事件
- 标签过期选项

1.5.1. 最小配置文件示例

以下是一个最小配置文件示例，使用镜像的本地存储：

\$QUAY/config/config.yaml

```
AUTHENTICATION_TYPE: Database
BUILDLOGS_REDIS:
  host: quay-server.example.com
  password: strongpassword
  port: 6379
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
password: strongpassword
port: 6379
```



注意

SETUP_COMPLETE 字段表示配置已被验证。在开始 registry 前，您应该使用配置编辑器工具验证您的配置。

1.5.2. 本地存储

只有在部署 registry 进行验证时，才建议使用本地存储作为镜像。在这种情况下，在启动 registry 时在命令行中指定存储，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:v3.6.8
```

1.5.3. 云存储

存储配置在 [镜像存储一节中进行详细](#)。使用云存储时比较差别（例如，在 Google Cloud Platform 上）非常有用：

\$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
```

使用云存储启动 registry 时，命令行不需要配置：

```
$ 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:v3.6.8
```

第 2 章 配置字段

2.1. 所需的配置字段

以下部分涵盖必填字段：

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

2.2. 自动化选项

- [预配置 Quay 以实现自动化](#)
- [使用 API 创建第一个用户](#)

2.3. 可选配置字段

以下部分介绍了如何提供可选字段：

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

- [ACI](#)
- [JWT](#)
- [应用程序令牌](#)
- [其他](#)
- [旧选项](#)

2.4. 常规必填字段

表 2.1. 常规必填字段

字段	类型	描述
AUTHENTICATION_TYPE (必需)	字符串	用于凭证身份验证的验证引擎 值： One of Database,LDAP,JWT,Keystone, Keystone,OIDC Default: Database
PREFERRED_URL_SCHEME (必需)	字符串	访问 Red Hat Quay 值时使用的 URL 方案： , http 之一, https Default: http
SERVER_HOSTNAME (必需)	字符串	可以访问 Red Hat Quay 的 URL, 没有方案 示例： quay-server.example.com
DATABASE_SECRET_KEY (必需)	字符串	用于加密数据库中敏感字段的密 钥。这个值不能被设置后更改, 否 则所有查询字段 (例如: 存储库镜 像用户名和密码配置) 都无效。
SECRET_KEY (必需)	字符串	用于加密数据库中及运行时敏感字 段的密钥。设置时不应更改其值, 否则所有回复字段 (如加密密码凭 证) 都无效。
SETUP_COMPLETE (必需)	布尔值	这是软件之前版本超过一批的制造 商, 目前 必须使用 true 指定一个 值。

2.5. 数据库配置

您使用 `DB_URI` 字段和 `DB_CONNECTION_ARGS` 结构中的所需 `DB_URI` 字段和可选连接参数来配置与数据库的连接。`DB_CONNECTION_ARGS` 下定义的一些键值对是通用的，另一些则特定于数据库。特别是，SSL 配置取决于您要部署的数据库，下面提供了 PostgreSQL 和 MySQL 示例。

2.5.1. 数据库 URI

表 2.2. 数据库 URI

字段	类型	描述
<code>DB_URI</code> (必需)	字符串	用于访问数据库的 URI，包括任何凭证

例如：

```
postgresql://quayuser:quaypass@quay-server.example.com:5432/quay
```

2.5.2. 数据库连接参数

表 2.3. 数据库连接参数

字段	类型	描述
<code>DB_CONNECTION_ARGS</code>	对象	数据库的可选连接参数，如超时和 SSL
<code>.autorollback</code>	布尔值	是否使用线程连接 应该为 true
<code>.threadlocals</code>	布尔值	是否使用自动重新连接 应该 ALWAYS 为 true

2.5.2.1. PostgreSQL SSL 连接参数

下面提供了 PostgreSQL SSL 配置示例：

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

`sslmode` 选项确定是否存在与服务器协商安全 SSL TCP/IP 连接的安全 SSL TCP/IP 连接。有六个模式：

- **disable**：只尝试非 SSL 连接
- **Allow**：首先尝试非 SSL 连接；如果失败，请尝试 SSL 连接

- **prefer:** (默认) 首先尝试 SSL 连接；如果失败，请尝试非 SSL 连接
- **Requires :** 仅尝试 SSL 连接。如果存在 root CA 文件，验证证书的方式与是否指定了 `verify-ca` 的方式相同
- **verify-ca :** 仅尝试 SSL 连接，并验证服务器证书是否由可信证书颁发机构(CA)发出。
- **verify-full :** 仅尝试 SSL 连接，验证服务器证书是否由受信任的 CA 发布，并且请求的服务器主机名与证书中的服务器主机名匹配

有关 PostgreSQL 的有效参数的更多信息，请参阅 <https://www.postgresql.org/docs/current/libpq-connect.html>

2.5.2.2. MySQL SSL 连接参数

MySQL SSL 配置示例如下：

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

有关 MySQL 的有效连接参数的信息，请参考 <https://dev.mysql.com/doc/refman/8.0/en/connecting-using-uri-or-key-value-pairs.html>。

2.6. 镜像存储

2.6.1. 镜像存储功能

表 2.4. 存储配置特性

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

2.6.2. 镜像存储配置字段

您可以使用 `DISTRIBUTED_STORAGE_CONFIG` 字段指定所有存储引擎的列表，并使用 `DISTRIBUTED_STORAGE_PREFERENCE` 字段选择首选的存储引擎。

`DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS` 字段用于控制其默认复制的位置。

表 2.5. 存储配置字段

字段	类型	描述
DISTRIBUTED_STORAGE_CONFIG (必需)	对象	配置 Red Hat Quay 中要使用的存储引擎。每个键代表存储引擎的唯一标识符。该值由代表（密钥、值）的元组组成一个描述存储引擎参数的对象。 默认值：[]
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS (必需)	字符串数组	默认情况下，存储引擎（根据 ID 在 DISTRIBUTED_STORAGE_CONFIG 中）的列表，这些镜像应完全复制到所有其他存储引擎。
DISTRIBUTED_STORAGE_PREFERENCE (必需)	字符串数组	要使用的 DISTRIBUTED_STORAGE_CONFIG 的首选存储引擎（根据 ID）。首选引擎意味着，首先检查拉取和镜像推送到该镜像。 默认：false
MAXIMUM_LAYER_SIZE	字符串	镜像层的最大允许大小 Pattern:^[0-9]+(G M)\$ Example: 100G Default: 20G

2.6.3. 本地存储

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

2.6.4. OCS/NooBaa

```
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
```

2.6.5. Ceph / RadosGW Storage / Hitachi HCP

```
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  
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []  
DISTRIBUTED_STORAGE_PREFERENCE:  
  - default
```

2.6.6. AWS S3 存储

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

2.6.7. Google 云存储

```
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
```

2.6.8. Azure 存储

```
DISTRIBUTED_STORAGE_CONFIG:  
  azureStorage:  
    - AzureStorage
```

```

- azure_account_name: azure_account_name_here
  azure_account_key: azure_account_key_here
  azure_container: azure_container_here
  sas_token: some/path/
  storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
- azureStorage

```

2.6.9. Swift 存储

```

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

```

2.7. REDIS 配置字段

2.7.1. 构建日志

表 2.6. 构建日志配置

字段	类型	描述
BUILDLGOS_REDIS (必需)	对象	构建日志缓存的 redis 连接详情
.主机 (必需)	字符串	Redis 可以访问的主机名 示例： quay-server.example.com
.port (必需)	Number	Redis 可访问的端口 示例： 6379
.password	字符串	Redis 可访问的端口 示例： strongpassword

2.7.2. 用户事件

表 2.7. 用户事件配置

字段	类型	描述
USER_EVENTS_REDIS (必需)	对象	用户事件处理的 redis 连接详情
.主机 (必需)	字符串	Redis 可以访问的主机名 示例 : quay-server.example.com
.port (必需)	Number	Redis 可访问的端口 示例 : 6379
.password	字符串	Redis 可访问的端口 示例 : strongpassword

2.7.3. redis 配置示例

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

```
USER_EVENTS_REDIS:
  host: quay-server.example.com
  password: strongpassword
  port: 6379
```

2.8. 标签过期选项

表 2.8. 标签过期配置

字段	类型	描述
FEATURE_GARBAGE_COLLECTION	布尔值	存储库垃圾回收是否已启用 Default: True

字段	类型	描述
TAG_EXPIRATION_OPTIONS (必需)	字符串数组	用户可以在其命名空间中为标签选择过期的选项（如果已启用） Pattern: ^[0-9]+(w m d h s)\$
DEFAULT_TAG_EXPIRATION (必需)	字符串	默认可配置标签过期时间，用于时间机器。 Pattern: ^[0-9]+(w m d h s)\$ Default: 2w
FEATURE_CHANGE_TAG_EXPIRATION	布尔值	用户和机构是否可以在其命名空间中更改标签 Default: True

例如：

```

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

```

2.9. 预配置 QUAY 以实现自动化

Quay 具有多个支持自动化的配置选项。这些选项可以在部署前设置，以尽可能地缩短与用户界面交互的需要。

2.9.1. 允许 API 创建第一个用户

将配置选项 **FEATURE_USER_INITIALIZE** 设置为 **true**，以便您可以使用 API **/api/v1/user/initialize** 来创建第一个用户。此 API 端点不需要身份验证，这与需要由现有组织中 OAuth 应用生成的 OAuth 令牌的所有其他 registry API 调用不同。

部署 Quay 后，您可以使用 API 来创建用户，例如 **quayadmin**，只要还没有创建其他用户。如需更多信息，请参阅 [使用 API 创建第一个用户](#) 部分

2.9.2. 启用常规 API 访问

将配置选项 **BROWSER_API_CALLS_XHR_ONLY** 设置为 **false**，以允许常规访问 Quay registry API。

2.9.3. 添加超级用户

虽然在部署后无法创建用户，但可以方便地确保第一个用户是一个具有完整权限的管理员。使用 **SUPER_USER** 配置对象，可以提前配置此设置。

2.9.4. 限制用户创建

配置了超级用户后，您可以限制创建新用户到超级用户组的能力。将 **FEATURE_USER_CREATION** 设置为 **false** 以限制用户创建。

2.9.5. 推荐的自动化配置

创建包含适当设置的 **config.yaml** 配置文件：

config.yaml

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

2.9.6. 使用初始配置部署 Operator

1. 使用配置文件创建 Secret

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

2. 创建 QuayRegistry YAML 文件 **quayregistry.yaml**，标识非受管组件以及引用所创建的 Secret，例如：

quayregistry.yaml

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

3. 部署 registry：

```
$ oc create -f quayregistry.yaml
```

4. 使用 API 创建第一个用户 **quayadmin**

2.9.7. 使用 API 创建第一个用户

当使用 API 创建第一个用户时，必须满足以下条件：

- 配置选项 `FEATURE_USER_INITIALIZE` 必须设置为 `true`
- 数据库中没有用户

如需有关预配置部署的更多信息，请参阅 [预配置 Quay 以实现自动化](#)

2.9.7.1. 调用 API

使用 `status.registryEndpoint` URL，调用 `/api/v1/user/initialize` API，传递用户名、密码和电子邮件地址。您还可以通过指定 `"access_token": true` 来请求 OAuth 令牌。

```
$ curl -X POST -k https://example-registry-quay-quay-
enterprise.apps.docs.quayteam.org/api/v1/user/initialize --header 'Content-Type: application/json' --
data '{"username": "quayadmin", "password": "quaypass123", "email": "quayadmin@example.com",
"access_token": true}'

{"access_token": "6B4QTRSTSD1HMIG915VPX7BMEZBVB9GPNY2FC2ED",
"email": "quayadmin@example.com", "encrypted_password": "1nZMLH57RIE5UGdL/yYpDOHLqiNCgi
mb6W9kfF8MjZ1xrfDpRyRs9NUnUuNuAitW", "username": "quayadmin"}
```

如果成功，该方法返回一个带有用户名、电子邮件和加密密码的对象。如果用户已存在于数据库中，则返回错误：

```
$ curl -X POST -k https://example-registry-quay-quay-
enterprise.apps.docs.quayteam.org/api/v1/user/initialize --header 'Content-Type: application/json' --
data '{"username": "quayuser2", "password": "quaypass123", "email": "quayuser2@example.com"}'

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

密码必须至少为 8 个字符，且不包含空格：

```
$ curl -X POST -k https://example-registry-quay-quay-
enterprise.apps.docs.quayteam.org/api/v1/user/initialize --header 'Content-Type: application/json' --
data '{"username": "quayadmin", "password": "pass123", "email": "quayadmin@example.com"}'

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

2.9.7.2. 使用 OAuth 令牌

现在，您可以在指定返回的 OAuth 代码调用 Quay API 的剩余部分。例如，获取当前用户的列表：

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

{
  "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。

2.9.7.2.1. 创建机构

要创建机构，使用 POST 调用 **api/v1/organization/** 端点：

```

$ 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"}'

```

```
"Created"
```

2.9.7.2.2. 获取机构详情

检索您创建的机构详情：

```

$ 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

```

```

{
  "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
}

```

2.10. 基本配置

表 2.9. 基本配置

字段	类型	描述
REGISTRY_TITLE	字符串	如果指定，registry Default: Quay Enterprise 的长格式标题
REGISTRY_TITLE_SHORT	字符串	如果指定，则 registry 的简短格式标题。 默认： Quay Enterprise
品牌	对象	Red Hat Quay UI 中的徽标和 URL 的自定义品牌。
.logo (必需)	字符串	主徽标镜像 URL 示例： /static/img/quay-horizontal-color.svg
.footer_img	字符串	UI footer 示例徽标： /static/img/RedHat.svg

字段	类型	描述
.footer_url	字符串	footer image 示例的链接： https://redhat.com
CONTACT_INFO	String 数组	如果指定，在联系页面中要显示的 联系信息。如果只指定了单一联系 人信息，联系页脚将直接链接。
[0]	字符串	添加要将电子邮件 Pattern: ^mailto:(.)+\$ Example: mailto:support@quay.io 的链 接
[1]	字符串	添加用于访问 IRC 聊天空间的链 接 Pattern: ^irc://(.)+\$ Example: irc://chat.freenode.net:6665/ quay
[2]	字符串	添加一个链接来调用电话号码+ Pattern: ^tel:(.)+\$ 示例： tel:+ reject930-3475
[3]	字符串	为定义的 URL Pattern: ^http(s)?://(.)+\$ Example: https://twitter.com/quayio

2.11. SSL 配置字段

表 2.10. SSL 配置

字段	类型	描述
PREFERRED_URL_SCHEME	字符串	http 之一, https Default: http
SERVER_HOSTNAME (必需)	字符串	可以访问 Red Hat Quay 的 URL, 没有方案 示例 : quay-server.example.com
SSL_CIPHERS	String 数组	如果指定, 则 nginx 定义的 SSL 密码列表用于启用和禁用 示例 : [CAMELLIA,!3DES]
SSL_PROTOCOLS	String 数组	如果指定, 则 nginx 配置为启用列 表中定义的 SSL 协议列表。从列 表中删除 SSL 协议在 Red Hat Quay 启动过程中禁用协议。 示例 : ['TLSv1','TLSv1.1','TLSv1.2']
SESSION_COOKIE_SECURE	布尔值	在 session cookies Default: False Recommendation 上是否应设置 secure 属性 : 对于所有使用 SSL 的安装, Set to True

2.11.1. 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

2.12. 在 RED HAT QUAY CONTAINER 中添加 TLS 证书

要在 Red Hat Quay 中添加自定义 TLS 证书，请在 Red Hat Quay 配置目录下创建名为 **extra_ca_certs/** 的新目录。将任何特定于站点的 TLS 证书复制到此新目录。

2.12.1. 在 Red Hat Quay 中添加 TLS 证书

1. 查看添加到容器中的证书

```
$ cat storage.crt
-----BEGIN CERTIFICATE-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
[...]
-----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:v3.6.8 "/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-----
MIIDTTCCAjWgAwIBAgIJAMVr9ngjJhzbMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
```

2.13. LDAP 配置字段

表 2.11. LDAP 配置

字段	类型	描述
AUTHENTICATION_TYPE (必需)	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与身份验证引擎 (LDAP 或 Keystone) 中的后备组同步 Default: true
FEATURE_NONSUPERUSER_TEAM_SYNCING_SETUP	布尔值	如果启用, 非 superusers 可以使用 LDAP Default: false 在团队上设置同步。
LDAP_ADMIN_DN	字符串	用于 LDAP 身份验证的管理员 DN。
LDAP_ADMIN_PASSWD	字符串	LDAP 身份验证的管理员密码。
LDAP_ALLOW_INSECURE_FALLBACK	布尔值	是否允许 SSL insecure fallback 进行 LDAP 身份验证。
LDAP_BASE_DN	String 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	用于 LDAP 身份验证的用户过滤器。
LDAP_USER_RDN	String 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, 如果需要的 Pattern: ^[0-9]+(w m d h s)\$ Example: 2h Default: 30m

2.13.1. LDAP 配置示例

\$QUAY/config/config.yamll

```
AUTHENTICATION_TYPE: LDAP
...
LDAP_ADMIN_DN: uid=testuser,ou=Users,o=orgid,dc=jumpexamplecloud,dc=com
LDAP_ADMIN_PASSWD: samplepassword
LDAP_ALLOW_INSECURE_FALLBACK: false
LDAP_BASE_DN:
  - o=orgid
  - dc=example
  - dc=com
LDAP_EMAIL_ATTR: mail
LDAP_UID_ATTR: uid
LDAP_URI: ldap://ldap.example.com:389
LDAP_USER_RDN:
  - ou=Users
```

2.14. 镜像配置字段

表 2.12. 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 Default: false
REPO_MIRROR_INTERVAL	Number	检查存储库镜像候选时的秒数 Default: 30
REPO_MIRROR_SERVER_HOSTNAME	字符串	替换 SERVER_HOSTNAME 作为镜像的目的地。 Default: None 示例: openshift-quay-service
REPO_MIRROR_TLS_VERIFY	布尔值	需要 HTTPS 并验证 Quay registry 在镜像过程中的证书。 默认 : false

2.15. 安全扫描程序配置字段

表 2.13. 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 Default: 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	此参数用于决定在重新索引之前具有之前失败或自上索引以来已更改状态的最短时间（以秒为单位）。数据通过 <code>manifestsecuritystatus</code> 表中的 last_indexed_datetime 来计算。这个参数用于避免在每次索引运行时都尝试重新索引每个失败清单。重新索引的默认时间为 300 秒。
SECURITY_SCANNER_V4_ENDPOINT	字符串	V4 安全扫描程序的端点 Pattern: <code>^http(s)?://(.)+\$</code> Example: http://192.168.99.101:6060
SECURITY_SCANNER_V4_PSK	字符串	Clair 所生成的预共享密钥(PSK)
SECURITY_SCANNER_INDEXING_INTERVAL	Number	安全扫描程序 Default: 30 中索引间隔的秒数 : 30
SECURITY_SCANNER_ENDPOINT	字符串	V2 安全扫描程序的端点 Pattern: <code>^http(s)?://(.)+\$</code> Example: http://192.168.99.100:6060
SECURITY_SCANNER_INDEXING_INTERVAL	字符串	这个参数用于决定安全扫描程序索引间隔间隔的秒数。当索引被触发时，Red Hat Quay 将查询其数据库以获取由 Clair 索引的清单。这包括还没有索引的、之前索引的清单和之前失败的清单。

以下是 re-indexing 的特殊情况：

当 Clair v4 索引清单时，结果应该是确定的。例如，同一清单应该生成相同的索引报告。这在扫描程序被改变前为 true，因为使用不同的扫描程序会生成与报告中返回的特定清单相关的信息。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改变。

在解析到 Quay 数据库时，Red Hat Quay 将这个索引状态保存到索引报告中。如果自清单之前已扫描以来此状态已更改，Quay 将会在定期索引过程中尝试重新索引该清单。

默认情况下，此参数设为 30 秒。如果用户希望更频繁地运行索引过程，用户可能会缩短。例如，如果他们不希望在推送新标签后等待 30 秒时间查看 UI 的安全扫描结果。如果用户希望对 Clair 的更多控制权以及 Quay 数据库执行的数据库操作的模式，用户也可以更改参数。

2.16. OCI 和 HELM 配置

现在，在 **FEATURE_GENERAL_OCI_SUPPORT** 属性中支持 Helm 的支持。如果需要显式启用该功能，例如之前禁用了该功能，或者已经从未默认启用的版本升级，则需要在 Quay 配置中添加两个属性来启用 OCI 工件的使用：

```
FEATURE_GENERAL_OCI_SUPPORT: true
FEATURE_HELM_OCI_SUPPORT: true
```

表 2.14. OCI 和 Helm 配置

字段	类型	描述
FEATURE_GENERAL_OCI_SUPPORT	布尔值	启用对 OCI 工件的支持 Default: True
FEATURE_HELM_OCI_SUPPORT	布尔值	启用对 Helm 工件的支持 Default: True



重要

从 Red Hat Quay 3.6 开始，**FEATURE_HELM_OCI_SUPPORT** 已被弃用，并将在以后的 Red Hat Quay 版本中删除。在 Red Hat Quay 3.6 中，默认支持 Helm 工件，并包括在 **FEATURE_GENERAL_OCI_SUPPORT** 属性中。用户不再需要更新其 config.yaml 文件来启用支持。

2.17. 操作日志配置字段

2.17.1. 操作日志存储配置

表 2.15. 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 Default: True

字段	类型	描述
LOGS_MODEL	字符串	启用或禁用安全扫描程序 值：一个数据库 ,transition_reads_ both_writes_es,elasticsearch h Default: database
LOGS_MODEL_CONFIG	对象	操作日志的日志模型配置

- LOGS_MODEL_CONFIG [object]: Logs model config for action logs
 - elasticsearch_config [object] : Elasticsearch 集群配置
 - access_key [string] : Elasticsearch 用户（或 AWS ES 的 IAM 键）
 - 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] : 对 Elasticsearch 使用 ssl。默认为 True
 - 示例 : **True**
 - SECRET_KEY [string] : Elasticsearch 密码（或 AWS ES 的 IAM secret）
 - 示例 : **some_secret_string**
 - aws_region [string]: Amazon web service region
 - 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 configuration
 - aws_secret_key [string]: AWS secret key
 - 示例 : **some_secret_key**
 - stream_name [string]: Kinesis stream to send action logs to
 - 示例 : **logentry-kinesis-stream**

- `aws_access_key` [string]: AWS access key
 - 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]: 要保留在连接池中的最多连接数
 - 示例 : 10
- `aws_region` [string]: AWS 区域
 - 示例 : `us-east-1`
- `connect_timeout` [number]: 尝试连接超时前的秒数
 - 示例 : 5
- `producer` [string]: 日志记录到 Elasticsearch 时的 Logs producer
 - enum: `kafka`, `elasticsearch`, `kinesis_stream`
 - 示例 : `kafka`
- `kafka_config` [object]: Kafka 集群配置
 - 主题 [string]: Kafka 主题, 以发布日志条目
 - 示例 : `logentry`
 - `bootstrap_servers` [array]: Kafka 代理列表到引导客户端
 - `max_block_seconds` [number]: 在 `send ()` 期间要阻断的最大秒数, 因为缓冲区已满或元数据不可用
 - 示例 : 10

2.17.2. 操作日志轮转和归档配置

表 2.16. 操作日志轮转和归档配置

字段	类型	描述
<code>FEATURE_ACTION_LOG_ROTATION</code>	布尔值	启用日志轮转和存档会将所有存在时间超过 30 天的日志移动到 storage Default: false

字段	类型	描述
ACTION_LOG_ARCHIVE_LOCATION	字符串	如果启用了操作日志存档，则会启用存储引擎来放置存档数据 示例：s3_us_east
ACTION_LOG_ARCHIVE_PATH	字符串	如果启用了操作日志归档，存储中的路径会放置存档数据 示例：存档/操作日志
ACTION_LOG_ROTATION_THRESHOLD	字符串	轮转日志的时间间隔 示例：30d

2.18. 构建日志

表 2.17. 构建日志

字段	类型	描述
FEATURE_READER_BUILD_LOGS	布尔值	如果设置为 true，则构建日志可由对存储库具有读取访问权限的用户读取，而不是仅具有写入访问权限或管理员访问权限。 Default: False
LOG_ARCHIVE_LOCATION	字符串	在 DISTRIBUTED_STORAGE_CONFIG 中定义的存储位置，用于放置存档的构建日志 示例：s3_us_east
LOG_ARCHIVE_PATH	字符串	配置的存储引擎下的路径，以 JSON 形式放置存档构建日志 示例：存档/构建日志

2.19. DOCKERFILE 构建触发器字段

表 2.18. 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 None，则构建触发器自动禁用前可能会出现连续失败数量 Default: 100
SUCCESSIVE_TRIGGER_INTERNAL_ERROR_DISABLE_THRESHOLD	Number	如果没有 None，则构建触发器自动禁用前可能会出现成功内部错误的数量 Default: 5

2.19.1. GitHub 构建触发器

表 2.19. 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 Default: False
GITHUB_TRIGGER_CONFIG	对象	使用 GitHub（企业）进行构建触发器的配置
.GITHUB_ENDPOINT (必需)	字符串	GitHub(Enterprise) 示例 : https://github.com/
.API_ENDPOINT	字符串	要使用的 GitHub(Enterprise)API 的端点。必须为 github.com 示例 覆 盖 : https://api.github.com/
.CLIENT_ID (必需)	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。
.CLIENT_SECRET (必需)	字符串	此 Red Hat Quay 实例注册的客户端 secret。

2.19.2. Bitbucket 构建触发器

表 2.20. 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 Default: False
BITBUCKET_TRIGGER_CONFIG	对象	使用 BitBucket 配置构建触发器
.CONSUMER_KEY (必需)	字符串	此 Quay 实例注册的使用者密钥 (客户端 ID)
.CONSUMER_SECRET (必需)	字符串	此 Quay 实例注册的使用者机密 (客户端 secret)

2.19.3. GitLab 构建触发器

表 2.21. 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 Default: False
GITLAB_TRIGGER_CONFIG	对象	使用 Gitlab 进行构建触发器的配置
.GITLAB_ENDPOINT (必需)	字符串	运行 Gitlab(Enterprise)的端点
.CLIENT_ID (必需)	字符串	此 Quay 实例注册的客户端 ID
.CLIENT_SECRET (必需)	字符串	此 Quay 实例注册的客户端 secret

2.20. OAUTH 配置

表 2.22. OAuth 字段

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

2.20.1. GitHub OAuth

表 2.23. GitHub OAuth 字段

字段	类型	描述
FEATURE_GITHUB_LOGIN	布尔值	GitHub 登录是否被支持 **Default: False
GITHUB_LOGIN_CONFIG	对象	将 GitHub(Enterprise)用作外部登录提供程序的配置。
.ALLOWED_ORGANIZATIONS	String 数组	将 GitHub(Enterprise)机构的名称列入白名单以与 ORG_RESTRICT 选项搭配使用。
.API_ENDPOINT	字符串	要使用的 GitHub(Enterprise)API 的端点。必须为 github.com 示例覆盖： https://api.github.com/
.CLIENT_ID (必需)	字符串	此 Red Hat Quay 实例注册的客户端 ID; 无法与 GITHUB_TRIGGER_CONFIG 示例共享 示例： 0e8dbe15c4c7630b6780
.CLIENT_SECRET (必需)	字符串	此 Red Hat Quay 实例注册的客户端 secret 示例： e4a58ddd3d7408b7aec109e85564a0d153d3e846
.GITHUB_ENDPOINT (必需)	字符串	GitHub (企业) 示例 的端点： https://github.com/

字段	类型	描述
.ORG_RESTRICT	布尔值	如果为 true，则只有机构白名单中的用户才可以使用此供应商登录。

2.20.2. Google OAuth

表 2.24. Google OAuth 字段

字段	类型	描述
FEATURE_GOOGLE_LOGIN	布尔值	Google 登录是否被支持 **Default: False
GOOGLE_LOGIN_CONFIG	对象	使用 Google 进行外部身份验证的配置
.CLIENT_ID (必需)	字符串	此 Red Hat Quay 实例注册的客户端 ID 示例 : 0e8dbe15c4c7630b6780
.CLIENT_SECRET (必需)	字符串	此 Red Hat Quay 实例注册的客户端 secret 示例 : e4a58ddd3d7408b7aec109e855 64a0d153d3e846

2.21. 配置嵌套软件仓库

在 Red Hat Quay 3.6 中，在 `FEATURE_EXTENDED_REPOSITORY_NAMES` 属性下添加了对嵌套存储库路径名称的支持。用户必须手动将这个可选配置添加到 `config.yaml` 中才能启用支持。启用允许在仓库名称中使用 `/`。

```
FEATURE_EXTENDED_REPOSITORY_NAMES: true
```

表 2.25. OCI 和嵌套存储库配置

字段	类型	描述
FEATURE_EXTENDED_REPOSITORY_NAMES	布尔值	启用对嵌套软件仓库的支持 Default: False

2.22. 向 QUAY 添加其他 OCI 介质类型

Helm、cosign 和 ztsd 压缩方案工件默认内置在 Red Hat Quay 3.6 中。对于默认情况下不支持的任何其他 OCI 介质类型，您可以使用以下格式将它们添加到 Quay config.yaml 中的 `ALLOWED_OCI_ARTIFACT_TYPES` 配置中：

```
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，您可以通过在 config.yaml 中添加以下内容来添加 Singularity(SIF)支持：

```
...
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

当添加默认配置的 OCI 介质类型时，用户需要手动添加对 cosign 和 Helm 的支持。在默认情况下，支持 ztsd 压缩方案，因此用户不需要将 OCI 介质类型添加到其 config.yaml 中以启用支持。

2.23. 邮件配置

表 2.26. 邮件字段

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

字段	类型	描述
MAIL_DEFAULT_SENDER	字符串	如果指定，在 Red Hat Quay 发送电子邮件时，用作的电子邮件地址。 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。只有将 FEATURE_MAILING 设置为 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，是否使用 TLS 发送电子邮件 Default: True

2.24. 用户配置字段

表 2.27. 用户配置

字段	类型	描述
FEATURE_SUPER_USERS	布尔值	是否支持超级用户 Default: true
FEATURE_USER_CREATION	布尔值	用户可创建（非用户） Default: true

字段	类型	描述
FEATURE_USER_LAST_ACCESSED	布尔值	是否最后一次记录用户被访问的时间 Default: true
FEATURE_USER_LOG_ACCESS	布尔值	如果设置为 true，则用户可以访问其命名空间 Default: false 的审计日志
FEATURE_USER_METADATA	布尔值	是否收集并支持用户元数据 Default: false
FEATURE_USERNAME_CONFIRMATION	布尔值	如果设置为 true，用户可以确认其生成的用户名 Default: 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	创建用户是否必须被其他用户 Default: false 邀请
FRESH_LOGIN_TIMEOUT	字符串	新登录的时间要求用户重新输入其密码 示例 : 5m
USERFILES_LOCATION	字符串	用于放置用户上传的文件的存储引擎的 ID 示例 : s3_us_east
USERFILES_PATH	字符串	在存储下放置用户上传的文件 示例 : userfiles
USER_RECOVERY_TOKEN_LIFETIME	字符串	恢复用户帐户的令牌的时间长度为 Pattern: ^[0-9]+(w m d h s)\$ Default: 30m

字段	类型	描述
----	----	----

2.25. RECAPTCHA 配置

表 2.28. Recaptcha 字段

字段	类型	描述
FEATURE_RECAPTCHA	布尔值	用户登录和恢复 Default: False 是否需要 Recaptcha
RECAPTCHA_SECRET_KEY	字符串	如果启用了 recaptcha, 则 Recaptcha 服务的 secret 键
RECAPTCHA_SITE_KEY	字符串	如果启用了 recaptcha, 则 Recaptcha 服务的 site 键

2.26. ACI 配置

表 2.29. ACI 配置

字段	类型	描述
FEATURE_ACI_CONVERSION	布尔值	是否启用到 ACI Default: 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	用于为 ACI 签名的私钥名称
GPG2_PUBLIC_KEY_FILENAME	字符串	用于加密 ACI 的公钥的文件名

2.27. JWT 配置

表 2.30. JWT 配置

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

2.28. 应用程序令牌

表 2.31. 应用程序令牌配置

字段	类型	描述
FEATURE_APP_SPECIFIC_TOKENS	布尔值	如果启用，用户可以创建令牌供 Docker CLI Default: True
APP_SPECIFIC_TOKEN_EXPIRATION	字符串	外部应用令牌的过期。 默认 None Pattern: <code>^[0-9]+(w m d h s)\$</code>
EXPIRED_APP_SPECIFIC_TOKEN_GC	字符串	在垃圾回收 Default: 1d 前保留外部应用令牌的时间持续时间

2.29. 其他字段

表 2.32. 其他字段

字段	类型	描述
ALLOW_PULLS_WITHOUT_STRICT_LOGGING	字符串	<p>如果为 true，则即使无法写入拉取审计日志条目，拉取仍会成功。如果数据库处于只读状态，且需要拉取以便在此期间继续时，这很有用。</p> <p>Default: False</p>
AVATAR_KIND	字符串	<p>要显示的 avatars 类型，可以是生成的内联（本地）或 Gravatar(gravatar)</p> <p>Values: local, gravatar</p>
BROWSER_API_CALLS_XHR_ONLY	布尔值	<p>如果启用，则只允许从浏览器</p> <p>Default: True 中标记为 XHR 的 API 调用</p>
DEFAULT_NAMESPACE_MAXIMUM_BUILD_COUNT	Number	<p>在一个命名空间中可以排队的默认构建数量。</p> <p>默认： None</p>
ENABLE_HEALTH_DEBUG_SECRET	字符串	<p>如果指定，可提供给健康端点的 secret，以便在不以超级用户身份验证时查看完整的调试信息</p>
EXTERNAL_TLS_TERMINATION	布尔值	<p>如果支持 TLS，则设置为 true，但在 Quay 之前在层终止</p>
FRESH_LOGIN_TIMEOUT	字符串	<p>新登录的时间要求用户重新输入其密码</p> <p>示例： 5m</p>
HEALTH_CHECKER	字符串	<p>配置的健康检查</p> <p>示例： <pre>('RDSAwareHealthCheck', {'access_key': 'foo', 'secret_key': 'bar'})</pre></p>
PROMETHEUS_NAMESPACE	字符串	<p>应用到所有公开的 Prometheus metrics</p> <p>Default: quay</p>

字段	类型	描述
PUBLIC_NAMESPACES	String 数组	如果在公共命名空间中定义了命名空间，它将出现在 所有用户 的存储库列表页面上，无论用户是否为命名空间的成员。通常，这由在配置一组 "well-known" 命名空间的企业客户使用。
REGISTRY_STATE	字符串	registry 的状态 值：normal 或 read-only
SEARCH_MAX_RESULT_PAGE_COUNT	Number	在被限制 Default: 10 之前，用户可以搜索的最大页面数。
SEARCH_RESULTS_PER_PAGE	Number	根据搜索页面返回的结果数 Default: 10
V1_PUSH_WHITELIST	String 数组	如果 FEATURE_RESTRICTED_V1_PUSH 设置为 true，则支持 V1 推送的命名空间名称的数组。
V2_PAGINATION_SIZE	Number	V2 registry API 中每个页面返回的结果数
WEBHOOK_HOSTNAME_BLACKLIST	String 数组	在除 localhost 外验证时，从 Webhook 中禁止的主机名集合
CREATE_PRIVATE_REPO_ON_PUSH	布尔值	通过推送创建的新软件仓库是否设置为专用可见性 Default: True
CREATE_NAMESPACE_ON_PUSH	布尔值	新推送到不存在的机构是否会创建 Default: False
NON_RATE_LIMITED_NAMESPACES	String 数组	如果使用 FEATURE_RATE_LIMITS 启用速率限制，您可以为需要无限访问的特定命名空间覆盖它。

2.30. 旧配置字段

有些字段已弃用或过时的：

表 2.33. 旧字段

字段	类型	描述
FEATURE_BLACKLISTED_EMAILS	布尔值	如果设为 true, 则在电子邮件域列入黑名单时无法创建新的用户帐户
BLACKLISTED_EMAIL_DOMAINS	String 数组	FEATURE_BLACKLISTED_EMAILS 被设置为 true 示例: "example.com", "example.com" , "example.org"
BLACKLIST_V2_SPEC	字符串	Red Hat Quay 将响应 V2 的 Docker CLI 版本为 不受支持的 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

第 3 章 环境变量

Red Hat Quay 支持有限的用于动态配置的环境变量。

3.1. GEO-REPLICATION

所有区域应使用完全相同的配置，但存储后端除外，可以使用 `QUAY_DISTRIBUTED_STORAGE_PREFERENCE` 环境变量进行显式配置。

表 3.1. 异地复制配置

变量	类型	描述
<code>QUAY_DISTRIBUTED_STORAGE_PREFERENCE</code>	字符串	首选存储引擎（根据 <code>DISTRIBUTED_STORAGE_CONFIG</code> 使用）。

3.2. 数据库连接池

Red Hat Quay 由很多不同的进程组成，它们都在同一个容器内运行。很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。这些每个进程连接池配置为保持最多 20 个连接。在重度负载下，可以为 Red Hat Quay 容器中的每个进程填写连接池。在某些部署和加载下，这可能需要分析，以确保 Red Hat Quay 不超过数据库配置的最大连接数。

现在，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。

可以通过设置环境变量 `DB_CONNECTION_POOLING={true|false}` 来切换数据库连接池。

表 3.2. 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，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
...
```

3.3. HTTP 连接计数

可以使用环境变量同时指定 HTTP 连接的数量。这些可指定为整个组件，也可以指定为特定的组件。每个进程的默认值为 50 个并行连接。

表 3.3. HTTP 连接计数配置

变量	类型	描述
WORKER_CONNECTION_COUNT	Number	同时 HTTP 连接 Default: 50
WORKER_CONNECTION_COUNT_REGISTRY	Number	同步 registry 的 HTTP 连接 Default: WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_WEB	Number	Web UI 同时 HTTP 连接 Default: WORKER_CONNECTION_COUNT
WORKER_CONNECTION_COUNT_SECSCAN	Number	Clair Default: WORKER_CONNECTION_COUNT 同步 HTTP 连接

3.4. WORKER 计数变量

表 3.4. worker 计数变量

变量	类型	描述
WORKER_COUNT	Number	进程数的通用覆盖
WORKER_COUNT_REGISTRY	Number	指定在 Quay 容器中处理 Registry 请求的进程数量 值：在 8 到 64 之间的 Integer
WORKER_COUNT_WEB	Number	指定容器 值（在 2 到 32 之间）处理 UI/Web 请求的进程数
WORKER_COUNT_SECSCAN	Number	指定处理安全扫描扫描（如 Clair）集成的进程数量 值：2 到 4 之间的 Integer

第 4 章 使用配置工具在 OPENSIFT 中重新配置 QUAY

4.1. 访问配置编辑器

在 QuayRegistry 屏幕的 Details 部分中，可以使用 config 编辑器的端点，以及指向含有登录配置编辑器凭证的 secret 的链接：

The screenshot shows the 'Quay Registry overview' page for a registry named 'example'. The page is organized into two columns of key-value pairs:

- Name:** example
- Current Version:** 3.5.2
- Namespace:** openshift-operators (with a lock icon)
- Config Editor Credentials Secret:** example-quay-config-editor-credentials-9ffgftc7 (with a lock icon)
- Registry Endpoint:** example-quay-openshift-operators.apps.docs.quayteam.org
- Config Editor Endpoint:** example-quay-config-editor-openshift-operators.apps.docs.quayteam.org
- Labels:** No labels (with an 'Edit' link)
- Annotations:** 1 annotation (with a link)
- Created at:** Jun 24, 5:33 pm
- Owner:** No owner

4.1.1. 检索配置编辑器凭证

1. 点击配置编辑器 secret 的链接：

Project: openshift-operators ▾

Secrets > Secret details

S example-quay-config-editor-credentials-9ffgfgtfc7 Add Secret to workload Actions ▾

Managed by  example

[Details](#) [YAML](#)

Secret details

Name
example-quay-config-editor-credentials-9ffgfgtfc7

Namespace
 openshift-operators

Type
Opaque

Labels Edit 

quay-operator/quayregistry=example

Annotations
4 annotations 

Created at
 Jun 25, 11:40 am

Owner
 example

Data [Reveal values](#)

password

..... 

username

..... 

- 在 Secret Details 屏幕的 Data 部分，点 **Reveal values** 来查看登录到配置编辑器的凭证：

Data [Hide values](#)

password

Zr1lN6tCtZeVww4q 

username

quayconfig 

4.1.2. 登录到配置编辑器

浏览到配置编辑器端点，然后输入用户名（通常是 `quayconfig`）以及访问配置工具的对应密码：

Red Hat Quay Setup

Custom SSL Certificates

This section lists any custom or self-signed SSL certificates that are installed in the Quay container on startup after being read from the `extra_ca_certs` directory in the configuration volume.

Custom certificates are typically used in place of publicly signed certificates for corporate-internal services.

Please **make sure** that all custom names used for downstream services (such as Clair) are listed in the certificates below.

Upload certificates:

Select custom certificate to add to configuration. Must be in PEM format and end extension '.crt'

CERTIFICATE FILENAME	STATUS	NAMES HANDLED
extra_ca_certs/service-ca.crt	✔ Certificate is valid	openshift-service-serving-signer@1624454606

Basic Configuration

Registry Title:

Name of registry to be displayed in the Contact Page.

Registry Title Short:

Enterprise Logo URL:

Enter the full URL to your company's logo.

Contact Information:

Information to show in the Contact Page. If none specified, CoreOS contact information is displayed.

Server Configuration

Server Hostname:

The HTTP host (and optionally the port number if a non-standard HTTP/HTTPS port) of the location where the registry will be accessible on the network.

TLS:

Validate Configuration Changes TLS also enables HTTP Strict Transport Security.

This prevents downgrade attacks and cookie theft, but browsers will reject all future insecure connections on this hostname.

4.1.3. 更改配置

在本例中，通过配置编辑器工具添加超级用户：

1. 为时间机器功能添加过期周期，如 4w：

Time Machine

Time machine keeps older copies of tags within a repository for the configured period of time, after which they are garbage collected. This allows users to revert tags to older images in case they accidentally pushed a broken image. It is highly recommended to have time machine enabled, but it does take a bit more space in storage.

Allowed expiration periods:

The expiration periods allowed for configuration. The default tag expiration "must" be in this list.

Default expiration period:

The default tag expiration period for all namespaces (users and organizations). Must be expressed in a duration string form: `30m`, `1h`, `1d`, `2w`.

Allow users to select expiration: Enable Expiration Configuration

If enabled, users will be able to select the tag expiration duration for the namespace(s) they administrate, from the configured list of options.

2. 选择 **Validate Configuration Changes** 以确保更改有效3. 按 **Reconfigure Quay** 按钮应用更改：

Validating configuration

✓ CONFIGURATION VALIDATED

✓ Configuration Validated

Continue Editing

Download

Reconfigure Quay

4. 配置工具通知您已向 Quay 提交了更改：

Validating configuration

✓ CONFIGURATION VALIDATED

✓ CONFIG SENT TO OPERATOR

✓ Configuration Validated

Continue Editing

Download

Reconfigure Quay



注意

使用配置工具 UI 配置 Red Hat Quay 可能会导致 registry 短时间不可用，同时应用更新的配置。

4.2. 在 UI 中监控重新配置

4.2.1. QuayRegistry 资源

重新配置 Operator 后，您可以跟踪特定 QuayRegistry 实例的 YAML 选项卡中的重新部署进度，本例中为 `example-registry`：

Project: quay-enterprise ▾

Installed Operators > quay-operator.v3.6.0 > QuayRegistry details

 example-registryDetails YAML Resources Events

```

1  apiVersion: quay.redhat.com/v1
2  kind: QuayRegistry
3  metadata:
4    selfLink: >=
5    /apis/quay.redhat.com/v1/namespaces/quay-enterprise/quayregistries/example-registry
6    resourceVersion: '78140'
7    name: example-registry
8    uid: 0a77c77c-b560-4d52-9d8a-ba8481ab4d04
9    creationTimestamp: '2021-09-24T10:13:02Z'
10   generation: 7
11  > managedFields: --
45   namespace: quay-enterprise
46   finalizers:
47     - quay-operator/finalizer
48   spec:
49  > components: --
68   configBundleSecret: example-registry-quay-config-bundle-zb9c7
69   status:
70     conditions:
71     - lastTransitionTime: '2021-09-24T10:14:40Z'
72       lastUpdateTime: '2021-09-24T10:14:40Z'
73       message: all registry component healthchecks passing
74       reason: HealthChecksPassing
75       status: 'True'
76       type: Available
77     - lastTransitionTime: '2021-09-24T11:23:02Z'
78       lastUpdateTime: '2021-09-24T11:23:02Z'
79       message: all objects created/updated successfully
80       reason: ComponentsCreationSuccess
81       status: 'False'
82       type: RolloutBlocked
83   configEditorCredentialsSecret: example-registry-quay-config-editor-credentials-gbtbkh94kh
84   configEditorEndpoint: >=
85     https://example-registry-quay-config-editor-quay-enterprise.apps.docs.quayteam.org
86   currentVersion: 3.6.0
87   lastUpdated: '2021-09-24 11:23:02.084685976 +0000 UTC'

```

 This object has been updated.

Click reload to see the new version.

Save

Reload

Cancel

每次状态发生变化时，系统都会提示您重新载入数据以查看更新的版本。最后，Operator 将协调更改，且不会报告不健康的组件。

Project: quay-enterprise ▾

Installed Operators > quay-operator.v3.6.0 > QuayRegistry details

 example-registryDetails YAML Resources Events

```

1  apiVersion: quay.redhat.com/v1
2  kind: QuayRegistry
3  metadata:
4  ▾ selfLink: >-
5    /apis/quay.redhat.com/v1/namespaces/quay-enterprise/quayregistries/example-registry
6    resourceVersion: '79051'
7    name: example-registry
8    uid: 0a77c77c-b560-4d52-9d8a-ba8481ab4d04
9    creationTimestamp: '2021-09-24T10:13:02Z'
10   generation: 7
11  > managedFields:--
43   namespace: quay-enterprise
44  ▾ finalizers:
45    - quay-operator/finalizer
46  ▾ spec:
47  > components:--
66   configBundleSecret: example-registry-quay-config-bundle-zb9c7
67  ▾ status:
68  ▾ conditions:
69  ▾ - lastTransitionTime: '2021-09-24T10:14:40Z'
70    lastUpdateTime: '2021-09-24T10:14:40Z'
71    message: all registry component healthchecks passing
72    reason: HealthChecksPassing
73    status: 'True'
74    type: Available
75  ▾ - lastTransitionTime: '2021-09-24T11:23:02Z'
76    lastUpdateTime: '2021-09-24T11:23:02Z'
77    message: all objects created/updated successfully
78    reason: ComponentsCreationSuccess
79    status: 'False'
80    type: RolloutBlocked
81   configEditorCredentialsSecret: example-registry-quay-config-editor-credentials-gbtbkh94kh
82  ▾ configEditorEndpoint: >-
83    https://example-registry-quay-config-editor-quay-enterprise.apps.docs.quayteam.org
84   currentVersion: 3.6.0
85   lastUpdated: '2021-09-24 11:23:02.084685976 +0000 UTC'
86   registryEndpoint: 'https://example-registry-quay-quay-enterprise.apps.docs.quayteam.org'
87   unhealthyComponents: {}
88

```

Save

Reload

Cancel

4.2.2. 事件

QuayRegistry 的 Events 选项卡显示与重新部署相关的一些事件：

Streaming events...		Showing 491 events
example-registry-quay-app	Generated from horizontal-pod-autoscaler failed to get cpu utilization: did not receive metrics for any ready pods	quay-enterprise Sep 24, 12:16 pm 29 times in the last an hour
example-registry-quay-app-c7698bfc-lsx2	Generated from kubelet on docs-k95iz-worker-d-tzg54.c.quay-devel.internal Readiness probe failed: Get "http://10.128.2.40:8080/health/instance": dial tcp 10.128.2.40:8080: connect: connection refused	quay-enterprise Sep 24, 12:16 pm
example-registry-quay-app	Generated from deployment-controller Scaled down replica set example-registry-quay-app-c7698bfc to 0	quay-enterprise a few seconds ago
example-registry-quay-app-c7698bfc-lsx2	Generated from kubelet on docs-k95iz-worker-d-tzg54.c.quay-devel.internal Stopping container quay-app	quay-enterprise a few seconds ago
example-registry-quay-app-c7698bfc	Generated from replicaset-controller Deleted pod: example-registry-quay-app-c7698bfc-lsx2	quay-enterprise a few seconds ago

在 OpenShift 控制台 Home → Events 下，针对受重新配置影响的所有资源的流传输事件：

Streaming events...		Showing 491 events
example-registry-quay-app	Generated from horizontal-pod-autoscaler failed to get cpu utilization: did not receive metrics for any ready pods	quay-enterprise Sep 24, 12:16 pm 29 times in the last an hour
example-registry-quay-app-c7698bfc-lsx2	Generated from kubelet on docs-k95iz-worker-d-tzg54.c.quay-devel.internal Readiness probe failed: Get "http://10.128.2.40:8080/health/instance": dial tcp 10.128.2.40:8080: connect: connection refused	quay-enterprise Sep 24, 12:16 pm
example-registry-quay-app	Generated from deployment-controller Scaled down replica set example-registry-quay-app-c7698bfc to 0	quay-enterprise a few seconds ago
example-registry-quay-app-c7698bfc-lsx2	Generated from kubelet on docs-k95iz-worker-d-tzg54.c.quay-devel.internal Stopping container quay-app	quay-enterprise a few seconds ago
example-registry-quay-app-c7698bfc	Generated from replicaset-controller Deleted pod: example-registry-quay-app-c7698bfc-lsx2	quay-enterprise a few seconds ago

4.3. 在重新配置后访问更新的信息

4.3.1. 在 UI 中访问更新的配置工具凭证

由于已经为配置工具创建了新 pod，因此系统将创建一个新 secret，并在下一次尝试登录时需要使用更新的密码：

Data

Hide values

password

DTmdv2-3oSIWQdIp

username

quayconfig

4.3.2. 在 UI 中访问更新的 config.yaml

使用 config 捆绑包访问更新的 config.yaml 文件。

1. 在 QuayRegistry 详情屏幕上点 Config Bundle Secret
2. 在 Secret Details 屏幕的 Data 部分，点 Reveal values 查看 config.yaml 文件
3. 检查是否应用了更改。在这种情况下，4w 应该位于 TAG_EXPIRATION_OPTIONS 列表中：

```
...
SERVER_HOSTNAME: example-quay-openshift-operators.apps.docs.quayteam.org
SETUP_COMPLETE: true
```

SUPER_USERS:

- quayadmin

TAG_EXPIRATION_OPTIONS:

- 2w

- 4w

...

第 5 章 QUAY OPERATOR 组件

Quay 是一个强大的容器 registry 平台，因此有多个依赖项。它们包括数据库、对象存储、Redis 等。Quay Operator 管理对 Quay 的意见部署及其对 Kubernetes 的依赖项。这些依赖项被视为 *组件*，并通过 QuayRegistry API 配置。

在 QuayRegistry 自定义资源中，`spec.components` 字段配置组件。每个组件包含两个字段：`kind` - 组件的名称，`managed` - 布尔值（无论组件生命周期是由 Operator 处理）。默认情况下（显示此字段），所有组件都将在协调后自动填充，以获得可见性：

```
spec:
  components:
    - managed: true
      kind: clair
    - managed: true
      kind: postgres
    - managed: true
      kind: objectstorage
    - managed: true
      kind: redis
    - managed: true
      kind: horizontalpodautoscaler
    - managed: true
      kind: route
    - managed: true
      kind: mirror
    - managed: true
      kind: monitoring
    - managed: true
      kind: tls
```

5.1. 使用受管组件

除非 QuayRegistry 自定义资源指定其他，否则 Operator 将对以下受管组件使用默认值：

- Postgres：要存储 registry 元数据，请使用 [Software Collections](#) 中的 Postgres 10 版本
- Redis：处理 Quay 构建器协调和一些内部日志记录
- objectstorage：用于存储镜像层 blob，使用 Noobaa/RHOCS 提供的 `ObjectBucketClaim` Kubernetes API
- Clair：提供镜像漏洞策略扫描
- HorizontalPodAutoscaler：根据 memory/cpu 消耗调整 Quay pod 的数量
- mirror：配置存储库镜像 worker（支持可选存储库镜像）
- Route：从外部 OpenShift 提供指向 Quay registry 的外部入口点
- monitoring：功能包括 Grafana 仪表盘、对单个指标的访问以及通知的提示以经常重启 Quay Pod
- TLS：配置 Red Hat Quay 或 OpenShift 是否处理 TLS

Operator 将处理 Red Hat Quay 使用受管组件所需的配置和安装工作。如果 Quay Operator 对环境意见进行了建议，您可以为 Operator 提供非受管资源(overrides)的 Operator，如以下部分所述。

5.2. 将非受管组件用于依赖项

如果您有现有的组件，如 Postgres、Redis 或对象存储，则您要首先在 Quay 配置捆绑包(config.yaml)中配置它们，然后在 QuayRegistry 中引用捆绑包（作为 Kubernetes Secret），同时表示哪些组件是非受管状态。



注意

Quay 配置编辑器也可用于创建或修改现有配置捆绑包，并简化更新 Kubernetes Secret 的过程，特别是用于多项更改。当通过配置编辑器更改 Quay 配置并发送到 Operator 时，将更新 Quay 部署来反映新配置。

5.2.1. 使用现有的 Postgres 数据库

1. 使用所需的数据库字段创建配置文件 config.yaml：

config.yaml:

```
DB_URI: postgresql://test-quay-database:postgres@test-quay-database:5432/test-quay-database
```

2. 使用配置文件创建 Secret：

```
$ kubectl create secret generic --from-file config.yaml=./config.yaml config-bundle-secret
```

3. 创建 QuayRegistry YAML 文件 quayregistry.yaml，将 postgres 组件标记为非受管状态，并引用所创建的 Secret：

quayregistry.yaml

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

4. 按照以下部分所述部署 registry。

5.2.2. NooBaa 非受管存储

1. 在 Storage → Object Bucket Claims 的控制台中创建一个 NooBaa Object Bucket Claim。
2. 检索 Object Bucket Claim Data 的详情，包括 Access Key、Bucket Name、Endpoint(hostname)和 Secret Key。

3. 使用 Object Bucket Claim 信息创建 config.yaml 配置文件：

```
DISTRIBUTED_STORAGE_CONFIG:
  default:
    - RHOCSSStorage
    - access_key: WmrXtSGk8B3nABCDEFGH
      bucket_name: my-noobaa-bucket-claim-8b844191-dc6c-444e-9ea4-87ece0abcdef
      hostname: s3.openshift-storage.svc.cluster.local
      is_secure: true
      port: "443"
      secret_key: X9P5SDGJtmSuHFCMSLMbdNCMfUABCDEFGH+C5QD
      storage_path: /datastorage/registry
DISTRIBUTED_STORAGE_DEFAULT_LOCATIONS: []
DISTRIBUTED_STORAGE_PREFERENCE:
  - default
```

5.2.3. 禁用 Horizontal Pod Autoscaler

HorizontalPodAutoscalers 已添加到 Clair、Quay 和 Mirror pod 中，以便在负载高峰期间自动扩展。

因为 HPA 默认配置为受管，Quay 的 pod 数量将 Clair 和存储库镜像设置为 2。这有助于在通过 Operator 更新/配置 Quay 或重新调度事件期间出现停机的问题。

如果要禁用自动扩展或创建自己的 HorizontalPodAutoscaler，只需在 QuayRegistry 实例中将组件指定为 Unmanaged：

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: example-registry
  namespace: quay-enterprise
spec:
  components:
    - kind: horizontalpodautoscaler
      managed: false
```

5.3. 在 KUBERNETES 上部署时添加证书

在 Kubernetes 上部署时，Red Hat Quay 将作为卷挂载到 secret 中来存储配置资产。不幸的是，这当前会破坏超级用户面板的上传证书功能。

要解决这个问题，可在部署 Red Hat Quay 后将 base64 编码证书添加到 secret 中。以下是如何：

1. 首先使用 base64 编码证书内容：

```
$ cat ca.crt
-----BEGIN CERTIFICATE-----
MIIDljCCAn6gAwIBAgIBATANBgkqhkiG9w0BAQsFADA5MRcwFQYDVQQKDA5MQUlu
TEICQ09SRS5TTzEeMBwGA1UEAwwVQ2VydGhmaWNhdGUgQXV0aG9yaXR5MB4XDTE2
MDExMjA2NTkxMFOxDTM2MDExMjA2NTkxMFOxOTExMjA2NTkxMFOxOTExMjA2NTkxMFOx
UkUuU08xHjAcBgNVBAMMFUNlcnRpZmlyYXRIIEF1dGhvcml0eTCCASlwdQYJKoZI
[...]
-----END CERTIFICATE-----
```

```
$ cat ca.crt | base64 -w 0
[...]
c1psWGpqeGIPQmNEWkJPMjJ5d0pDemVnR2QNCnRsbW9JdEF4YnFSdVd3PT0KLS0tLS1F
TkQgQ0VSVEIGSUNBVEUtLS0tLQo=
```

2. 使用 `kubectl` 工具编辑 `quay-enterprise-config-secret`。

```
$ kubectl --namespace quay-enterprise edit secret/quay-enterprise-config-secret
```

3. 为证书添加一个条目，并在条目下粘贴完整的 base64 编码字符串：

```
custom-cert.crt:
c1psWGpqeGIPQmNEWkJPMjJ5d0pDemVnR2QNCnRsbW9JdEF4YnFSdVd3PT0KLS0tLS1F
TkQgQ0VSVEIGSUNBVEUtLS0tLQo=
```

4. 最后，回收所有 Red Hat Quay Pod。使用 `kubectl delete` 删除所有 Red Hat Quay Pod。Red Hat Quay Deployment 将自动使用新证书数据调度替代 pod。

5.4. 使用 OPERATOR 配置 OCI 和 HELM

Quay 配置的自定义可以在包含配置捆绑包的 `secret` 中提供。在适当的命名空间中执行以下命令，它将创建名为 `quay-config-bundle` 的新 `secret`，其包含启用 OCI 支持的必要属性。

`quay-config-bundle.yaml`

```
apiVersion: v1
stringData:
  config.yaml: |
    FEATURE_GENERAL_OCI_SUPPORT: true
    FEATURE_HELM_OCI_SUPPORT: true
kind: Secret
metadata:
  name: quay-config-bundle
  namespace: quay-enterprise
type: Opaque
```

重要

从 Red Hat Quay 3.6 开始，`FEATURE_HELM_OCI_SUPPORT` 已被弃用，并将在以后的 Red Hat Quay 版本中删除。在 Red Hat Quay 3.6 中，默认支持 Helm 工件，并包括在 `FEATURE_GENERAL_OCI_SUPPORT` 属性中。用户不再需要更新其 `config.yaml` 文件来启用支持。

在适当的命名空间中创建 `secret`，在本例中为 `quay-enterprise`：

```
$ oc create -n quay-enterprise -f quay-config-bundle.yaml
```

指定 `spec.configBundleSecret` 字段的 `secret`：

`quay-registry.yaml`

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

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

使用指定配置创建 registry :

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

5.5. 卷大小覆盖

从 Red Hat Quay v3.6.2 开始，您可以指定为受管组件置备的存储资源所需的大小。Clair 和 Quay PostgreSQL 数据库的默认大小为 **50Gi**。现在，您可以预先选择足够大的容量，可能是因为性能的原因，或者在您的存储后端没有调整大小功能的情况下选择。

在以下示例中，Clair 和 Quay PostgreSQL 数据库的卷大小已设置为 **70Gi** :

```
apiVersion: quay.redhat.com/v1
kind: QuayRegistry
metadata:
  name: quay-example
  namespace: quay-enterprise
spec:
  configBundleSecret: config-bundle-secret
  components:
    - kind: objectstorage
      managed: false
    - kind: route
      managed: true
    - kind: tls
      managed: false
    - kind: clair
      managed: true
      overrides:
        volumeSize: 70Gi
    - kind: postgres
      managed: true
      overrides:
        volumeSize: 70Gi
```

第 6 章 使用配置 API

该配置工具有 4 个端点，可用于构建、验证、捆绑和部署配置。config-tool API 记录在 <https://github.com/quay/config-tool/blob/master/pkg/lib/editor/API.md> 中。在本小节中，您将了解如何使用 API 检索当前配置以及如何验证您所做的任何更改。

6.1. 检索默认配置

如果您首次运行配置工具，且没有现有配置，您可以检索默认配置。以 config 模式启动容器：

```
$ sudo podman run --rm -it --name quay_config \
  -p 8080:8080 \
  registry.redhat.io/quay/quay-rhel8:v3.6.8 config secret
```

使用配置 API 的 config 端点获取默认值：

```
$ curl -X GET -u quayconfig:secret http://quay-server:8080/api/v1/config | jq
```

返回的值是 JSON 格式的默认配置：

```
{
  "config.yaml": {
    "AUTHENTICATION_TYPE": "Database",
    "AVATAR_KIND": "local",
    "DB_CONNECTION_ARGS": {
      "autorollback": true,
      "threadlocals": true
    },
    "DEFAULT_TAG_EXPIRATION": "2w",
    "EXTERNAL_TLS_TERMINATION": false,
    "FEATURE_ACTION_LOG_ROTATION": false,
    "FEATURE_ANONYMOUS_ACCESS": true,
    "FEATURE_APP_SPECIFIC_TOKENS": true,
    ....
  }
}
```

6.2. 检索当前配置

如果您已经配置并部署了 Quay registry，请停止容器并以配置模式重启它，以作为卷载入现有配置：

```
$ sudo podman run --rm -it --name quay_config \
  -p 8080:8080 \
  -v $QUAY/config:/conf/stack:Z \
  registry.redhat.io/quay/quay-rhel8:v3.6.8 config secret
```

使用 API 的 config 端点获取当前的配置：

```
$ curl -X GET -u quayconfig:secret http://quay-server:8080/api/v1/config | jq
```

返回的值是 JSON 格式当前的配置，包括数据库和 Redis 配置数据：

```
{
  "config.yaml": {
    ....
    "BROWSER_API_CALLS_XHR_ONLY": false,
    "BUILDLOGS_REDIS": {
      "host": "quay-server",
      "password": "strongpassword",
      "port": 6379
    },
    "DATABASE_SECRET_KEY": "4b1c5663-88c6-47ac-b4a8-bb594660f08b",
    "DB_CONNECTION_ARGS": {
      "autorollback": true,
      "threadlocals": true
    },
    "DB_URI": "postgresql://quayuser:quaypass@quay-server:5432/quay",
    "DEFAULT_TAG_EXPIRATION": "2w",
    ....
  }
}
```

6.3. 使用 API 验证配置

您可以通过将其发送到 `config/validate` 端点来验证配置：

```
curl -u quayconfig:secret --header 'Content-Type: application/json' --request POST --data '
{
  "config.yaml": {
    ....
    "BROWSER_API_CALLS_XHR_ONLY": false,
    "BUILDLOGS_REDIS": {
      "host": "quay-server",
      "password": "strongpassword",
      "port": 6379
    },
    "DATABASE_SECRET_KEY": "4b1c5663-88c6-47ac-b4a8-bb594660f08b",
    "DB_CONNECTION_ARGS": {
      "autorollback": true,
      "threadlocals": true
    },
    "DB_URI": "postgresql://quayuser:quaypass@quay-server:5432/quay",
    "DEFAULT_TAG_EXPIRATION": "2w",
    ....
  }
} http://quay-server:8080/api/v1/config/validate | jq
```

返回的值是一个包含配置中发现的错误的数组。如果配置有效，则返回空数组 []。

6.4. 确定必填字段

您可以通过在 `config/validate` 端点中发布空配置结构来确定必填字段：

```
curl -u quayconfig:secret --header 'Content-Type: application/json' --request POST --data '
{
  "config.yaml": {
  }
} http://quay-server:8080/api/v1/config/validate | jq
```

返回的值是一个数组，表示需要哪些字段：

```
[
  {
    "FieldGroup": "Database",
    "Tags": [
      "DB_URI"
    ],
    "Message": "DB_URI is required."
  },
  {
    "FieldGroup": "DistributedStorage",
    "Tags": [
      "DISTRIBUTED_STORAGE_CONFIG"
    ],
    "Message": "DISTRIBUTED_STORAGE_CONFIG must contain at least one storage
location."
  },
  {
    "FieldGroup": "HostSettings",
    "Tags": [
      "SERVER_HOSTNAME"
    ],
    "Message": "SERVER_HOSTNAME is required"
  },
  {
    "FieldGroup": "HostSettings",
    "Tags": [
      "SERVER_HOSTNAME"
    ],
    "Message": "SERVER_HOSTNAME must be of type Hostname"
  },
  {
    "FieldGroup": "Redis",
    "Tags": [
      "BUILDLOGS_REDIS"
    ],
    "Message": "BUILDLOGS_REDIS is required"
  }
]
```

第 7 章 使用配置工具

7.1. 自定义 SSL 证书 UI

配置工具可用于加载自定义证书，以便于访问外部数据库等资源。选择要上传的自定义证书，确保它们采用 PEM 格式，扩展为 .crt。

Custom SSL Certificates

This section lists any custom or self-signed SSL certificates that are installed in the Quay container on startup after being read from the `extra_ca_certs` directory in the configuration volume.

Custom certificates are typically used in place of publicly signed certificates for corporate-internal services.

Please **make sure** that all custom names used for downstream services (such as Clair) are listed in the certificates below.

Upload certificates:

Select custom certificate to add to configuration. Must be in PEM format and end extension '.crt'

CERTIFICATE FILENAME	STATUS	NAMES HANDLED
extra_ca_certs/service-ca.crt	✔ Certificate is valid	openshift-service-serving-signer@1632474198

配置工具还显示任何上传的证书列表。上传自定义 SSL 证书后，它将显示在列表中：

Custom SSL Certificates

This section lists any custom or self-signed SSL certificates that are installed in the Quay container on startup after being read from the `extra_ca_certs` directory in the configuration volume.

Custom certificates are typically used in place of publicly signed certificates for corporate-internal services.

Please **make sure** that all custom names used for downstream services (such as Clair) are listed in the certificates below.

Upload certificates:

Select custom certificate to add to configuration. Must be in PEM format and end extension '.crt'

CERTIFICATE FILENAME	STATUS	NAMES HANDLED	
extra_ca_certs/service-ca.crt	✔ Certificate is valid	openshift-service-serving-signer@1632474198	⚙
extra_ca_certs/my-custom-ssl-cert.crt	✔ Certificate is valid	quay-server.example.com	⚙

7.2. 基本配置

Basic Configuration

Registry Title:

Name of registry to be displayed in the Contact Page.

Registry Title Short:

Enterprise Logo URL:

Enter the full URL to your company's logo.

Contact Information:

Information to show in the Contact Page. If none specified, CoreOS contact information is displayed.

7.2.1. 联系信息

Basic Configuration

Registry Title:

Name of registry to be displayed in the Contact Page.

Registry Title Short:

Enterprise Logo URL:

Enter the full URL to your company's logo.

Contact Information: URL

- E-mail
- IRC
- Telephone
- URL

Contact Page. If none specified, CoreOS contact information is displayed.

Server Configuration

7.3. 服务器配置

Server Configuration

Server Hostname:

The HTTP host (and optionally the port number if a non-standard HTTP/HTTPS port) of the location where the registry will be accessible on the network.

TLS: None (Not For Production)

! Running without TLS should not be used for production workloads!

7.3.1. 服务器配置选择

Server Configuration

Server Hostname:

The HTTP host (and optionally the port number if a non-standard HTTP/HTTPS port) of the location where the registry will be accessible on the network.

TLS:

- Red Hat Quay handles TLS
- My own load balancer handles TLS (Not Recommended)
- None (Not For Production)

i Enabling TLS also enables [HTTP Strict Transport Security](#). This prevents downgrade attacks and cookie theft, but browsers will reject all future insecure connections on this hostname.

Certificate: Select a replacement file: No file chosen
The certificate must be in PEM format.

Private key: Select a replacement file: No file chosen

7.3.2. TLS 配置

Server Configuration

Server Hostname:

The HTTP host (and optionally the port number if a non-standard HTTP/HTTPS port) of the location where the registry will be accessible on the network.

TLS:

i Enabling TLS also enables [HTTP Strict Transport Security](#). This prevents downgrade attacks and cookie theft, but browsers will reject all future insecure connections on this hostname.

Certificate: Select a replacement file: No file chosen
The certificate must be in PEM format.

Private key: Select a replacement file: No file chosen

7.4. 数据库配置

您可以在 PostgreSQL 和 MySQL 之间进行选择：

Database

Quay uses a database as its primary metadata storage.

Database Type: MySQL Postgres

Database Server:

The server (and optionally, custom port) where the database lives

Username:

This user must have **full access** to the database

Password:

Database Name:

SSL Certificate: No file chosen

Optional SSL certificate (in PEM format) to use to connect to the database



注意

在 Red Hat Quay 3.6 中，MySQL 和 MariaDB 数据库已弃用。在以后的 Red Hat Quay 版本中会删除对这些数据库的支持。如果启动新的 Red Hat Quay 安装，强烈建议您使用 PostgreSQL。

7.4.1. PostgreSQL 配置

输入连接到数据库的详情：

Database

Quay uses a database as its primary metadata storage.

Database Type:

Database Server:

The server (and optionally, custom port) where the database lives

Username:

This user must have **full access** to the database

Password:

Database Name:

SSL Certificate: No file chosen

Optional SSL certicate (in PEM format) to use to connect to the database

这将生成 `postgresql://quayuser:quaypass@quay-server.example.com:5432/quay` 形式的 DB_URI 字段。

如果您需要对连接参数进行精细的控制，请参阅配置指南中的“数据基准连接参数”部分。

7.5. 数据一致性

Data Consistency Settings

Relax constraints on consistency guarantees for specific operations to enable higher performance and availability.

Allow repository pulls even if audit logging fails.

If enabled, failures to write to the audit log will fallback from the database to the standard logger for registry pulls.

7.6. 时间机器配置

Time Machine

Time machine keeps older copies of tags within a repository for the configured period of time, after which they are garbage collected. This allows users to revert tags to older images in case they accidentally pushed a broken image. It is highly recommended to have time machine enabled, but it does take a bit more space in storage.

- Allowed expiration periods:
- 0s [Remove](#)
 - 1d [Remove](#)
 - 1w [Remove](#)
 - 2w [Remove](#)
 - 4w [Remove](#)

The expiration periods allowed for configuration. The default tag expiration *must* be in this list.

Default expiration period:

The default tag expiration period for all namespaces (users and organizations). Must be expressed in a duration string form: `30m`, `1h`, `1d`, `2w`.

Allow users to select expiration: **Enable Expiration Configuration**

If enabled, users will be able to select the tag expiration duration for the namespace(s) they administrate, from the configured list of options.

7.7. REDIS 配置

Redis

A [redis](#) key-value store is required for real-time events and build logs.

Redis Hostname:

Redis port:

Access to this port and hostname must be allowed from all hosts running the enterprise registry

Redis password:

7.8. 存储库镜像配置

Repository Mirroring

If enabled, scheduled mirroring of repositories from registries will be available.

Enable Repository Mirroring

i A repository mirror service must be running to use this feature. Documentation on setting up and running this service can be found at [Running Repository Mirroring Service](#).

Require HTTPS and verify certificates of Quay registry during mirror.

7.9. REGISTRY 存储配置

- 代理存储
- 存储异地复制
- 存储引擎

7.9.1. 启用存储复制

1. 向下滚动到标题 **Registry Storage** 的部分。
2. 点 **Enable Storage Replication**。
3. 添加每个要复制数据的存储引擎。必须列出要使用的所有存储引擎。
4. 如果需要将所有镜像复制到所有存储引擎，请在各个存储引擎配置中单击 **Replicate** 到存储引擎。这将确保所有镜像都复制到该存储引擎。要改为启用按命名空间复制，请联系支持。
5. 完成后，单击 **Save Configuration Changes**。配置更改将在 Red Hat Quay 下次重启时生效。
6. 为 Georeplications 添加存储并启用"Replicate to存储引擎"后，您需要在所有存储间同步现有镜像数据。要做到这一点，您需要 `oc exec`（或 `docker/kubectl exec`）到容器中，并运行：

```
# scl enable python27 bash
# python -m util.backfillreplication
```

这是在添加新存储后同步内容的一个时间操作。

7.9.2. 存储引擎

7.9.2.1. 本地存储

Registry Storage

Registry images can be stored either locally or in a remote storage system.

! Do not use "Locally mounted directory" Storage Engine for any production configurations. Mounted NFS volumes are not supported. Local storage is meant for test-only installations.

Proxy storage via Quay

If enabled, all requests to storage engine(s) will be proxied through Quay . Should only be enabled if storage cannot be directly accessed by external nodes talking to the registry.

Enable Storage Replication

If enabled, replicates storage to other regions. See [documentation](#) for more information.

Location ID: default

Storage Engine:

Storage Directory:

7.9.2.2. Amazon S3 存储

Location ID:	default
Storage Engine:	<input type="text" value="Amazon S3"/>
S3 Bucket:	<input type="text" value="my-cool-bucket"/>
Storage Directory:	<input type="text" value="/datastorage/registry"/>
AWS Access Key (optional if using IAM):	<input type="text" value="accesskeyhere"/>
AWS Secret Key (optional if using IAM):	<input type="text" value="secretkeyhere"/>
S3 Host:	<input type="text" value="s3.amazonaws.com"/>
S3 Port:	<input type="text" value="443"/>

7.9.2.3. Azure blob 存储

Location ID:	default
Storage Engine:	<input type="text" value="Azure Blob Storage"/>
Azure Storage Container:	<input type="text" value="container"/>
Storage Directory:	<input type="text" value="/datastorage/registry"/>
Azure Account Name:	<input type="text" value="accountnamehere"/>
Azure Account Key:	<input type="text" value="accountkeyhere"/>
Azure SAS Token:	<input type="text" value="sastokenhere"/>

7.9.2.4. Google 云存储

Location ID:	default
Storage Engine:	<input type="text" value="Google Cloud Storage"/>
Cloud Access Key:	<input type="text" value="accesskeyhere"/>
Cloud Secret Key:	<input type="text" value="secretkeyhere"/>
GCS Bucket:	<input type="text" value="my-cool-bucket"/>
Storage Directory:	<input type="text" value="/datastorage/registry"/>

7.9.2.5. Ceph 对象网关(RADOS)存储

Location ID:	default
Storage Engine:	<input type="text" value="Ceph Object Gateway (RADOS)"/>
Rados Server Hostname:	<input type="text" value="my.rados.hostname"/>
Custom Port (optional):	<input type="text" value="443"/>
Is Secure:	<input type="checkbox"/> Require SSL
Access Key:	<input type="text" value="accesskeyhere"/> See Documentation for more information
Secret Key:	<input type="text" value="secretkeyhere"/>
Bucket Name:	<input type="text" value="my-cool-bucket"/>
Storage Directory:	<input type="text" value="/datastorage/registry"/>

7.9.2.6. OpenStack(Swift)存储配置

Location ID:	default
Storage Engine:	<input type="text" value="OpenStack Storage (Swift)"/>
Swift Auth Version:	<input type="text"/>
Swift Auth URL:	<input type="text" value="http://swiftdomain/auth/v1.0"/>
Swift Container Name:	<input type="text" value="mycontainer"/> The swift container for all objects. Must already exist inside Swift.
Storage Path:	<input type="text" value="/datastorage/registry"/>
Username:	<input type="text" value="accesskeyhere"/> Note: For Swift V1, this is "username:password" (-U on the CLI).
Key/Password:	<input type="text" value="secretkeyhere"/> Note: For Swift V1, this is the API token (-K on the CLI).
CA Cert Filename:	<input type="text" value="conf/stack/swift.cert"/>
Temp URL Key (optional):	<input type="text"/> If enabled, will allow for faster pulls directly from Swift. See Documentation for more information
OS Options:	No entries defined <hr/> Add Key-Value: <input type="text"/> <input type="text" value="Value"/> <input type="button" value="Add Entry"/>

7.9.2.7. CloudFront + Amazon S3 存储配置

Location ID:	default
Storage Engine:	CloudFront + Amazon S3
S3 Bucket:	my-cool-bucket
Storage Directory:	/datastorage/registry
AWS Access Key (optional if using IAM):	accesskeyhere
AWS Secret Key (optional if using IAM):	secretkeyhere
S3 Host:	s3.amazonaws.com
S3 Port:	443
CloudFront Distribution Domain Name:	somesubdomain.cloudfront.net
CloudFront Key ID:	APKATHISISAKEYID
CloudFront Private Key:	Please select a file to upload as default-cloudfront-signing-key.pem : <input type="button" value="Choose file"/> No file chosen

7.10. 操作日志配置

7.10.1. 操作日志存储配置

7.10.1.1. 数据库操作日志存储

Action Log Storage Configuration

Action logs can be stored in the database or Elasticsearch. In the latter case, the actions logs can (optionally) be sent to a data stream first.

Logs storage:

7.10.1.2. Elasticsearch 操作日志存储

📄 Action Log Storage Configuration

Action logs can be stored in the database or Elasticsearch. In the latter case, the actions logs can (optionally) be sent to a data stream first.

Logs storage:

Elasticsearch config

Elasticsearch hostname:

Elasticsearch port:

Access to this port and hostname must be allowed from all hosts running the enterprise registry

Elasticsearch access key:

Elasticsearch secret key:

AWS region:

Index prefix:

Logs Producer:

7.10.2. 操作日志轮转和归档

📄 Action Log Rotation and Archiving

All actions performed in Quay are automatically logged. These logs are stored in a database table, which can become quite large. Enabling log rotation and archiving will move all logs older than 30 days into storage.

Enable Action Log Rotation

Storage location:

The storage location in which to place archived action logs. Logs will only be archived to this single location.

Storage path:

The path under the configured storage engine in which to place the archived logs in JSON form.

Log Rotation Threshold:

The number of days after which to archive action logs to storage. Must be expressed in a duration string form: `30m`, `1h`, `1d`, `2w`.

Note: The rotation threshold should be considered a balancing act between user history and size of database.

Larger time windows will result in more logs, but give users more history to view. Anything less than `2w` is not recommended.

📄 Action Log Rotation and Archiving

All actions performed in Quay are automatically logged. These logs are stored in a database table, which can become quite large. Enabling log rotation and archiving will move all logs older than 30 days into storage.

Enable Action Log Rotation

Storage location:

The storage location in which to place archived action logs. Logs will only be archived to this single location.

7.11. 安全扫描程序配置

🔍 Security Scanner

If enabled, all images pushed to Quay will be scanned via the external security scanning service, with vulnerability information available in the UI and API, as well as async notification support.

Enable Security Scanning

i A scanner compliant with the Quay Security Scanning API must be running to use this feature. Documentation on running Clair can be found at [Running Clair Security Scanner](#).

Security Scanner Endpoint:

The HTTP URL at which the security scanner is running.

Security Scanner PSK:

Clair Pre-Shared Key. Make sure to include this value in your Clair config.

7.12. 应用程序 REGISTRY 配置

📦 Application Registry

If enabled, an additional registry API will be available for managing applications (Kubernetes manifests, Helm charts) via the [App Registry specification](#). A great place to get started is to install the [Helm Registry Plugin](#).

Enable App Registry

7.13. 电子邮件配置

✉ E-mail

Valid e-mail server configuration is required for notification e-mails and the ability of users to reset their passwords.

Enable E-mails

SMTP Server:

SMTP Server Port:

TLS: Require TLS

Mail Sender:

E-mail address from which all e-mails are sent. If not specified, `support@quay.io` will be used.

Authentication: Requires Authentication

Username:

Password:

7.14. 内部验证配置

Internal Authentication

Authentication for the registry can be handled by either the registry itself, LDAP, Keystone, or external JWT endpoint.

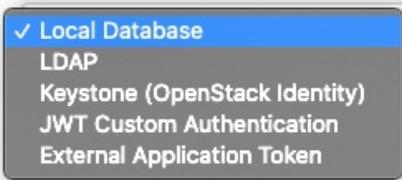
Additional **external** authentication providers (such as GitHub) can be used in addition for **login into the UI**.

Authentication:

Internal Authentication

Authentication for the registry can be handled by either the registry itself, LDAP, Keystone, or external JWT endpoint.

Additional **external** authentication providers (such as GitHub) can be used in addition for **login into the UI**.

Authentication: 

7.14.1. LDAP

Authentication:

Team synchronization: Enable Team Synchronization Support
If enabled, organization administrators who are also superusers can set teams to have their membership synchronized with a backing group in LDAP.

LDAP URI:
The full LDAP URI, including the ldap:// or ldaps:// prefix.

Base DN:
A Distinguished Name path which forms the base path for looking up all LDAP records.
 Example: dc=my,dc=domain,dc=com

User Relative DN:
A Distinguished Name path which forms the base path for looking up all user LDAP records, relative to the Base DN defined above.
 Example: ou=employees

Secondary User Relative DNs: No RDNs defined

A list of Distinguished Name path(s) which forms the secondary base path(s) for looking up all user LDAP records, relative to the Base DN defined above. These path(s) will be tried if the user is not found via the primary relative DN.
 Example: [ou=employees]

Additional User Filter Expression:  **NOTE:** This query is added **unescaped** to user lookups, so be **VERY** careful with the query you specify.

If specified, the additional filter used for all user lookup queries. Note that all Distinguished Names used in the filter must be full paths; the **base_dn** is not added automatically here. Must be wrapped in parens.
 Example: (someOtherField=someOtherValue)
 Example: (memberOf=some.full.path.to.a.group)
 Example: ((someFirstField=someValue)(someOtherField=someOtherValue))
 Example: (&(someFirstField=someValue)(someOtherField=someOtherValue))

Administrator DN:
The Distinguished Name for the Administrator account. This account must be able to login and view the records for all user accounts.
 Example: uid=admin,ou=employees,dc=my,dc=domain,dc=com

Administrator DN Password:  **Note:** This will be stored in **plaintext** inside the config.yaml, so setting up a dedicated account or using a **password hash** is **highly** recommended.

The password for the Administrator DN.

UID Attribute:
The name of the property field in your LDAP user records that stores your users' username. Typically "uid".

Mail Attribute:
The name of the property field in your LDAP user records that stores your users' e-mail address(es). Typically "mail".

7.14.2. Keystone (OpenStack 身份)

Internal Authentication

Authentication for the registry can be handled by either the registry itself, LDAP, Keystone, or external JWT endpoint.

Additional **external** authentication providers (such as GitHub) can be used in addition for **login into the UI**.

It is highly recommended to require encrypted client passwords. External passwords used in the Docker client will be stored in **plaintext!** [Enable this requirement now.](#)

Authentication:

Team synchronization: **Enable Team Synchronization Support**
 If enabled, organization administrators who are also superusers can set teams to have their membership synchronized with a backing group in Keystone.

Keystone API Version:

Keystone Authentication URL:
 The URL (starting with http or https) of the Keystone Server endpoint for auth.

Keystone Administrator Username:
 The username for the Keystone admin.

Keystone Administrator Password:
 The password for the Keystone admin.

Keystone Administrator Tenant:
 The tenant (project/group) that contains the administrator user.

7.14.3. JWT 自定义身份验证

Authentication:

JSON Web Token authentication allows your organization to provide an HTTP endpoint that verifies user credentials on behalf of Quay . Documentation on the API required can be found here: <https://github.com/coreos/jwt-auth-example>.

Authentication Issuer:
 The id of the issuer signing the JWT token. Must be unique to your organization.

Public Key: Please select a file to upload as `jwt-authn.cert`: No file chosen
 A certificate containing the public key portion of the key pair used to sign the JSON Web Tokens. This file must be in PEM format.

User Verification Endpoint:
 The URL (starting with http or https) on the JWT authentication server for verifying username and password credentials. Credentials will be sent in the `Authorization` header as Basic Auth, and this endpoint should return `200 OK` on success (or a `4**` otherwise).

User Query Endpoint:
 The URL (starting with http or https) on the JWT authentication server for looking up users based on a prefix query. This is optional. The prefix query will be sent as a query parameter with name `query`.

User Lookup Endpoint:
 The URL (starting with http or https) on the JWT authentication server for looking up a user by username or email address. The username or email address will be sent as a query parameter with name `username`.

7.14.4. 外部应用程序令牌

Internal Authentication

Authentication for the registry can be handled by either the registry itself, LDAP, Keystone, or external JWT endpoint. Additional **external** authentication providers (such as GitHub) can be used in addition for **login into the UI**.

Authentication:

7.15. 外部验证(OAUTH)配置

7.15.1. GitHub（企业）身份验证

External Authorization (OAuth)

GitHub (Enterprise) Authentication

If enabled, users can use GitHub or GitHub Enterprise to authenticate to the registry.

Note: A registered GitHub (Enterprise) OAuth application is required. View instructions on how to [Create an OAuth Application in GitHub](#)

Enable GitHub Authentication

GitHub:
 GitHub Enterprise

GitHub Endpoint:

The GitHub Enterprise endpoint. Must start with http:// or https://.

OAuth Client ID:

OAuth Client Secret:

Organization Filtering: **Restrict By Organization Membership**

If enabled, only members of specified GitHub Enterprise organizations will be allowed to login via GitHub Enterprise.

7.15.2. Google 身份验证

External Authorization (OAuth)

GitHub (Enterprise) Authentication

If enabled, users can use GitHub or GitHub Enterprise to authenticate to the registry.

Note: A registered GitHub (Enterprise) OAuth application is required. View instructions on how to [Create an OAuth Application in GitHub](#)

Enable GitHub Authentication

Google Authentication

If enabled, users can use Google to authenticate to the registry.

Note: A registered Google OAuth application is required. Visit the [Google Developer Console](#) to register an application.

Enable Google Authentication

OAuth Client ID:

OAuth Client Secret:

7.16. 访问设置配置

Access Settings

Various settings around access and authentication to the registry.

Basic Credentials **Login to User Interface via credentials**

Login:

If enabled, users will be able to login to the user interface via their username and password credentials.

If disabled, users will only be able to login to the user interface via one of the configured External Authentication providers.

External Application tokens **Allow external application tokens**

If enabled, users will be able to generate external application tokens for use on the Docker and rkt CLI. Note that these tokens will not be required unless "App Token" is chosen as the Internal Authentication method above.

External application token expiration

The expiration time for user generated external application tokens. If none, tokens will never expire.

Anonymous Access: **Enable Anonymous Access**

If enabled, public repositories and search can be accessed by anyone that can reach the registry, even if they are not authenticated. Disable to only allow authenticated users to view and pull "public" resources.

User Creation: **Enable Non-Superuser User Creation**

If enabled, user accounts can be created by anyone (unless restricted below to invited users). Users can always be created in the users panel in this superuser tool, even if this feature is disabled. If disabled, users can **ONLY** be created in the superuser tool or via team sync.

Invite-only User Creation: **Enable Invite-only User Creation**

If enabled, user accounts can only be created when a user has been invited, by e-mail address, to join a team. Users can always be created in the users panel in this superuser tool, even if this feature is enabled.

Encrypted Client Password: **Require Encrypted Client Passwords**

If enabled, users will not be able to login from the Docker command line with a non-encrypted password and must generate an encrypted password to use. This feature is highly recommended for setups with external authentication, as Docker currently stores passwords in plaintext on user's machines.

Prefix username autocompletion: **Allow prefix username autocompletion**

If disabled, autocompletion for users will only match on exact usernames.

Team Invitations: **Require Team Invitations**

If enabled, when adding a new user to a team, they will receive an invitation to join the team, with the option to decline. Otherwise, users will be immediately part of a team when added by a team administrator.

Super Users:

- quayadmin [Remove](#)

Users included in this list will be given elevated access to Quay.

7.17. DOCKERFILE 构建支持

☰ Dockerfile Build Support

If enabled, users can submit Dockerfile's to be built and pushed by Quay .

Enable Dockerfile Build

Note: Build workers are required for this feature. See [Adding Build Workers](#) for instructions on how to setup build workers.

🔗 GitHub (Enterprise) Build Triggers

If enabled, users can setup GitHub or GitHub Enterprise triggers to invoke Registry builds.

Note: A registered GitHub (Enterprise) OAuth application (**separate from GitHub Authentication**) is required. View instructions on how to [Create an OAuth Application in GitHub](#)

Enable GitHub Triggers

🔗 BitBucket Build Triggers

If enabled, users can setup BitBucket triggers to invoke Registry builds.

Note: A registered BitBucket OAuth application is required.

Enable BitBucket Triggers

🔗 GitLab Build Triggers

If enabled, users can setup GitLab triggers to invoke Registry builds.

Note: A registered GitLab OAuth application is required. Visit the [GitLab applications admin panel](#) to create a new application.

The callback URL to use is: `https://quay-server.example.com/oauth2/gitlab/callback/trigger`

Enable GitLab Triggers

7.17.1. GitHub（企业）构建触发器

🔗 GitHub (Enterprise) Build Triggers

If enabled, users can setup GitHub or GitHub Enterprise triggers to invoke Registry builds.

Note: A registered GitHub (Enterprise) OAuth application (**separate from GitHub Authentication**) is required. View instructions on how to [Create an OAuth Application in GitHub](#)

Enable GitHub Triggers

GitHub:

GitHub Endpoint:

The GitHub Enterprise endpoint. Must start with http:// or https://.

OAuth Client ID:

OAuth Client Secret:

7.17.2. Bitbucket 构建触发器

BitBucket Build Triggers

If enabled, users can setup BitBucket triggers to invoke Registry builds.

Note: A registered BitBucket OAuth application is required.

Enable BitBucket Triggers

OAuth Consumer Key:

OAuth Consumer Secret:

7.17.3. GitLab 构建触发器

GitLab Build Triggers

If enabled, users can setup GitLab triggers to invoke Registry builds.

Note: A registered GitLab OAuth application is required. Visit the [GitLab applications admin panel](#) to create a new application.

The callback URL to use is: `https://quay-server.example.com/oauth2/gitlab/callback/trigger`

Enable GitLab Triggers

GitLab:

GitLab CE/EE



GitLab Endpoint:

https://my.gitlabserver

The GitLab Enterprise endpoint. Must start with http:// or https://.

Application Id:

Secret: