



builds for Red Hat OpenShift 1.0

身份验证

了解运行时身份验证

了解运行时身份验证

法律通告

Copyright © 2023 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

摘要

本文档提供有关在运行时身份验证的信息。

目录

第 1 章 了解运行时身份验证	3
1.1. 构建 SECRET 注解	3
1.2. 对 GIT 存储库进行身份验证	3
1.3. 容器 REGISTRY 身份验证	5
1.4. 基于角色的访问控制	5

第 1 章 了解运行时身份验证

在构建镜像时，您可能需要在以下情况下定义身份验证：

- 向容器 registry 进行身份验证
- 从 Git 拉取源代码

身份验证通过定义存储所需敏感数据的机密进行。

1.1. 构建 SECRET 注解

您可以将注解 **build.shipwright.io/referenced.secret: "true"** 添加到构建 secret 中。基于此注解，构建控制器在事件（如为构建 secret 创建、更新或删除触发器）时会采取协调操作。以下示例显示了使用带有 secret 的注解：

```
apiVersion: v1
data:
  .dockerconfigjson: <pull_secret> 1
kind: Secret
metadata:
  annotations:
    build.shipwright.io/referenced.secret: "true" 2
  name: secret-docker
type: kubernetes.io/dockerconfigjson
```

1 base64 编码的 pull secret。

2 **build.shipwright.io/referenced.secret** 注解的值被设置为 **true**。

此注解过滤没有在构建实例中引用的 secret。例如，如果 secret 没有此注解，则即使为 secret 触发了事件，构建控制器也不会协调。通过协调触发事件，构建控制器可以在构建配置上重新触发验证，帮助您了解依赖项是否缺失。

1.2. 对 GIT 存储库进行身份验证

您可以为 Git 存储库定义以下类型的身份验证：

- 基本身份验证 (Basic authentication)
- Secure Shell (SSH)身份验证

您还可以在 **Build** CR 中使用两种类型的身份验证配置 Git secret。

1.2.1. 基本身份验证 (Basic authentication)

使用基本身份验证时，您必须配置 Git 存储库的用户名和密码。以下示例显示了对 Git 使用基本身份验证：

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-git-basic-auth
```

```

annotations:
  build.shipwright.io/referenced.secret: "true"
type: kubernetes.io/basic-auth ❶
stringData: ❷
  username: <cleartext_username>
  password: <cleartext_password>

```

- ❶ Kubernetes secret 的类型。
- ❷ 以明文形式存储您的用户名和密码的字段。

1.2.2. SSH 身份验证

通过 SSH 身份验证，您必须配置 Tekton 注解来指定 Git 存储库供应商的主机名。例如，用于 GitHub 的 **github.com** 或用于 GitLab 的 **gitlab.com**。

以下示例显示了对 Git 使用 SSH 身份验证：

```

apiVersion: v1
kind: Secret
metadata:
  name: secret-git-ssh-auth
  annotations:
    build.shipwright.io/referenced.secret: "true"
type: kubernetes.io/ssh-auth ❶
data:
  ssh-privatekey: | ❷
    # Insert ssh private key, base64 encoded

```

- ❶ Kubernetes secret 的类型。
- ❷ 用于在 Git 中进行身份验证的 SSH 密钥的 Base64 编码。您可以使用 **base64 ~/.ssh/id_rsa.pub** 命令来生成此密钥，其中 **~/.ssh/id_rsa.pub** 表示密钥的默认位置，它们通常用于向 Git 进行身份验证。

1.2.3. 使用 Git secret

在相关命名空间中创建 secret 后，您可以在 **Build** 自定义资源(CR)中引用它。您可以使用两种类型的身份验证配置 Git secret。

以下示例显示了使用带有 SSH 身份验证类型的 Git secret：

```

apiVersion: shipwright.io/v1beta1
kind: Build
metadata:
  name: buildah-golang-build
spec:
  source:
    git:
      url: git@gitlab.com:userjohn/newtaxi.git
      cloneSecret: secret-git-ssh-auth

```


以下示例显示了使用带有基本身份验证类型的 Git secret :

```
apiVersion: shipwright.io/v1beta1
kind: Build
metadata:
  name: buildah-golang-build
spec:
  source:
    git:
      url: https://gitlab.com/userjohn/newtaxi.git
      cloneSecret: secret-git-basic-auth
```

1.3. 容器 REGISTRY 身份验证

要将镜像推送到私有容器 registry, 您必须在对应命名空间中定义一个 secret, 然后在 **Build** 自定义资源 (CR) 中引用它。

流程

1. 运行以下命令以生成 secret :

```
$ oc --namespace <namespace> create secret docker-registry
<container_registry_secret_name> \
  --docker-server=<registry_host> \ 1
  --docker-username=<username> \ 2
  --docker-password=<password> \ 3
  --docker-email=<email_address>
```

- 1** & lt;registry_host> 值表示以下格式的 URL `https://<registry_server>/<registry_host>`。
- 2** & lt;username> 值是用户 ID。
- 3** & lt;password> 值可以是容器 registry 密码或访问令牌。

2. 运行以下命令来注解 secret :

```
$ oc --namespace <namespace> annotate secrets <container_registry_secret_name>
build.shipwright.io/referenced.secret='true'
```

3. 将 **spec.output.pushSecret** 字段的值设置为 **Build** CR 中的 secret 名称 :

```
apiVersion: shipwright.io/v1beta1
kind: Build
metadata:
  name: buildah-golang-build
  # ...
output:
  image: <path_to_image>
  pushSecret: <container_registry_secret_name>
```

1.4. 基于角色的访问控制

发行版本部署 YAML 文件包含两个使用 Builds 对象的集群范围角色。默认情况下会安装以下角色：

- **shpwright-build-aggregate-view** : 授予您读取 Builds 资源的访问权限，如 **BuildStrategy**、**ClusterBuildStrategy**、**Build** 和 **BuildRun**。此角色聚合到 Kubernetes 视图角色。
- **shipwright-build-aggregate-edit** : 授予您在命名空间级别上配置的 Builds 资源的访问权限。构建资源包括 **BuildStrategy**、**Build** 和 **BuildRun**。为所有 **ClusterBuildStrategy** 资源授予读访问权限。此角色聚合到 Kubernetes **edit** 和 **admin** 角色。

只有集群管理员对 **ClusterBuildStrategy** 资源具有写入权限。您可以通过创建具有这些权限的独立 Kubernetes **ClusterRole** 角色并将角色绑定到适当的用户来更改此设置。