Red Hat Trusted Artifact Signer 1

部署指南

在 Red Hat OpenShift 上安装和配置 Trusted Artifact Signer 服务

Last Updated: 2024-06-28

Red Hat Trusted Artifact Signer 1 部署指南

在 Red Hat OpenShift 上安装和配置 Trusted Artifact Signer 服务

法律通告

摘要

此部署指南为您提供在 Red Hat OpenShift 上安装 Trusted Artifact Signer 服务的信息，并验证安装是否成功。 红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 信息

目录

前言

欢迎使用 Red Hat Trusted Artifact Signer Deployment Guide!

这些流程可帮助您在 OpenShift 上部署完整的 Trusted Artifact Signer (RHTAS)软件堆栈，并验证部署。您可以在此处查看官方 RHTAS 发行注记。https://access.redhat.com/documentation/zh-cn/red_hat_trusted_artifact_signer/1.0/html/release_notes/

第 1 章 使用 OPERATOR LIFECYCLE MANAGER 安装 TRUSTED ARTIFACT SIGNER

您可以安装 Red Hat Trusted Artifact Signer (RHTAS) operator，并使用 OpenShift 的 Operator Lifecycle Manager (OLM)部署 RHTAS 服务。此部署为您提供选择 OpenID Connect (OIDC)供应商的基本签名框架。您必须至少配置以下 OIDC 供应商之一： Red Hat Single Sign-on (SSO)、Google、Amazon Secure Token Service (STS)或 GitHub。如果您不想使用默认值，您还可以选择自定义数据库解决方案。

先决条件

- Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15。
- 使用 **cluster-admin** 角色访问 OpenShift Web 控制台。
- 安装了 **oc** 二进制文件的工作站。

流程

1. 使用具有 **cluster-admin** 角色的用户登录 OpenShift Web 控制台。
2. 从 **Administrator** 视角中，展开 **Operators** 导航菜单，然后点 **OperatorHub**。
3. 在搜索字段中，键入 **trusted**，然后单击 **Red Hat Trusted Artifact Signer**标题。
4. 单击 **Install** 按钮，以显示 Operator 详情。
5. 接受默认值，点 ***Install*** *Operator* 页面中的 Install，并等待安装完成。

注意

Trusted Artifact Signer operator 安装到 **openshift-operators** 命名空间中，所有依赖项都会被自动安装。

6. 安装完成后，会自动为您创建一个新项目。新项目名称是 **trusted-artifact-signer**。
7. 可选。您可以为 Trusted Artifact Signer 服务使用替代数据库，而不是默认的数据库供应商。如果要在 OpenShift 上使用 Amazon 的相关数据库服务(RDS) 或 OpenShift 上的自我管理数据库，请在继续执行此安装前首先按照其中一个流程操作。配置其中一个其他数据库提供程序后，您可以继续这个过程的下一步。
8. 以部署 Trusted Artifact Signer 服务。
    a. 从导航菜单中展开 **Operators**，点 **Installed Operators**。
    b. 从项目下拉列表中，选择 **trusted-artifact-signer**。
    c. 点 **Red Hat Trusted Artifact Signer**。
    d. 单击 **Securesign** 选项卡，然后单击 **Create Securesign** 按钮。
    e. 在 *Create Securesign* 页面上，选择 **YAML 视图**。

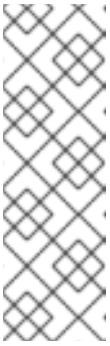
f. 您可以在此部署期间将 Google OAuth、Amazon STS、红帽 SSO 或 GitHub OAuth 配置为初始 OIDC 供应商。在 **spec.fulcio.config.OIDCIssuers** 部分下，使用 OIDC 供应商 URL 编辑以下三行，并相应地设置 **ClientID**。

Example

```
...
OIDCIssuers:
  - Issuer: 'OIDC_ISSUER_URL':
    ClientID: CLIENT_ID
    IssuerURL: 'OIDC_ISSUER_URL'
    Type: email
...
```

重要

您可以在同一配置中定义多个不同的 OIDC 供应商。



注意

如果红帽的 SSO 已作为 OIDC 供应商实施，请运行以下命令查找签发者 URL：

```
$ echo https://$(oc get route keycloak -n keycloak-system | tail -n 1 | awk '{print $2}')/auth/realms/trusted-artifact-signer
```

将 **ClientID** 设置为 **trusted-artifact-signer**。

g. 可选。如果使用除默认数据库以外的其他数据库，则在 **spec.trillian** 部分下，将 **create** 设置为 **false**，并指定数据库 secret 对象的名称。

Example

```
...
trillian:
  database:
    create: false
    databaseSecretRef:
      name: trillian-mysql
...
```

h. 点 **Create** 按钮。

9. 点 **All instances** 选项卡观察部署状态，直到 **CTlog**、**Fulcio**、**Rekor**、**Trillian** 和 **TUF** 实例就绪。

注意

Securesign 实例不提供状态。

10. 您可以在 OpenShift 控制台中使用 Prometheus 检查新的 Trusted Artifact Signer 服务的健康状态。在导航菜单中，展开 **Observe**，然后点 **Dashboards**。

11. 通过签署容器镜像 或 Git 提交 来验证安装。https://access.redhat.com/documentation/zh-cn/red_hat_trusted_artifact_signer/1/html/deployment_guide/verify_the_trusted_artifact_signer_s and-verifying-containers-by-using-cosign-from-the-command-line-interface_deploy

其他资源

- 有关 RHTAS 组件和版本号的更多信息，请参阅 RHTAS 部署指南 中的附录。

第 2 章 验证 TRUSTED ARTIFACT SIGNER 服务安装

2.1. 使用命令行界面使用 COSIGN 签名和验证容器

通过 **cosign** 工具，您可以使用红帽的Trusted Artifact Signer (RHTAS)服务签名和验证开放容器项目(OCI)容器镜像以及其他构建工件。

先决条件

- 在 Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15 上安装 RHTAS。
- 访问 OpenShift Web 控制台。
- 安装了 **podman** 二进制文件的工作站。

流程

1. 将 OpenShift 集群的 **cosign** 二进制文件下载到您的工作站。
   a. 登录到 OpenShift Web 控制台。在主页中，单击 **?** 图标，单击 **Command line tools**，前往 **cosign** download 部分，然后单击您的平台的链接。
   b. 在工作站上打开一个终端，解压缩二进制 **.gz** 文件，并设置执行位：

      Example

      ```
      $ gunzip cosign-amd64.gz
      $ chmod +x cosign-amd64
      ```

   c. 将二进制文件移到 **$PATH** 环境中的位置：

      Example

      ```
      $ sudo mv cosign-amd64 /usr/local/bin/cosign
      ```

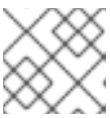
2. 切换到 RHTAS 项目：

   语法

   ```
   oc project PROJECT_NAME
   ```

   Example

   ```
   $ oc project trusted-artifact-signer
   ```

   注意

   使用 RHTAS 安装的项目名称。

3. 配置 shell 环境，以执行容器镜像签名和验证。

   Example

```
$ export TUF_URL=$(oc get tuf -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
$ export OIDC_ISSUER_URL=https://$(oc get route keycloak -n keycloak-system | tail -n 1 | awk '{print $2}')/auth/realms/trusted-artifact-signer
$ export COSIGN_FULCIO_URL=$(oc get fulcio -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
$ export COSIGN_REKOR_URL=$(oc get rekor -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
$ export COSIGN_MIRROR=$TUF_URL
$ export COSIGN_ROOT=$TUF_URL/root.json
$ export COSIGN_OIDC_CLIENT_ID="trusted-artifact-signer"
$ export COSIGN_OIDC_ISSUER=$OIDC_ISSUER_URL
$ export COSIGN_CERTIFICATE_OIDC_ISSUER=$OIDC_ISSUER_URL
$ export COSIGN_YES="true"
$ export SIGSTORE_FULCIO_URL=$COSIGN_FULCIO_URL
$ export SIGSTORE_OIDC_ISSUER=$COSIGN_OIDC_ISSUER
$ export SIGSTORE_REKOR_URL=$COSIGN_REKOR_URL
$ export REKOR_REKOR_SERVER=$COSIGN_REKOR_URL
```

4. 初始化更新框架(TUF)系统：

Example

```
$ cosign initialize
```

5. 为测试容器镜像签名。

a. 创建一个空容器镜像：

Example

```
$ echo "FROM scratch" > ./tmp.Dockerfile
$ podman build . -f ./tmp.Dockerfile -t ttl.sh/rhtas/test-image:1h
```

b. 将空容器镜像推送到 **ttl.sh** 临时 registry：

Example

```
$ podman push ttl.sh/rhtas/test-image:1h
```

c. 为容器镜像签名：

语法

```
cosign sign -y IMAGE_NAME:TAG
```

Example

```
$ cosign sign -y ttl.sh/rhtas/test-image:1h
```

此时会打开一个 Web 浏览器，允许您使用电子邮件地址为容器镜像签名。

d. 删除临时 Docker 文件：

Example

```
$ rm ./tmp.Dockerfile
```

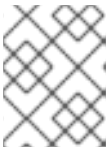
6. 使用证书身份和签发者验证签名的容器镜像：

语法

```
cosign verify --certificate-identity=SIGNING_EMAIL_ADDR IMAGE_NAME:TAG
```

Example

```
$ cosign verify --certificate-identity=jdoe@redhat.com ttl.sh/rhtas/test-image:1h
```



注意

您还可以使用 **cosign** 命令 **--certificate-identity-regexp** 和 **--certificate-oidc-issuer-regexp** 的以下选项对证书身份和签发者使用正则表达式。

其他资源

- 在 OpenShift 上安装 Red Hat Trusted Artifact Signer 。
- 自定义红帽受信任的应用程序管道.
- 如需了解有关 签名和验证 Git 提交的详细信息，请参阅 RHTAS 部署指南中的使用 Gitsign 的 签名和验证提交。
- 更新框架 主页.

2.2. 使用命令行界面使用 GITSIGN 签名和验证提交

gitsign 工具可让您使用红帽的 Trusted Artifact Signer (RHTAS)服务签名和验证 Git 存储库提交。

先决条件

- 在 Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15 上安装 RHTAS。
- 访问 OpenShift Web 控制台。
- 从 OpenShift 集群下载 **cosign** 二进制文件。

流程

1. 将 **gitsign** 二进制文件从 OpenShift 集群下载到您的工作站。
   a. 登录到 OpenShift Web 控制台。在主页中，单击 **?** 图标，单击 **Command line tools**，前往 **gitsign** 下载部分，然后单击您的平台的链接。
   b. 在工作站上打开一个终端，解压缩 .gz 文件，并设置执行位：

Example

```
$ gunzip gitsign-amd64.gz
$ chmod +x gitsign-amd64
```

c. 将二进制文件移到 **$PATH** 环境中的位置：

Example

```
$ sudo mv gitsign-amd64 /usr/local/bin/gitsign
```

2. 切换到 RHTAS 项目：

语法

```
oc project PROJECT_NAME
```

示例

```
$ oc project trusted-artifact-signer
```

注意

使用 RHTAS 安装的项目名称。

3. 配置 shell 环境以进行提交签名和验证：

示例

```
$ export TUF_URL=$(oc get tuf -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
$ export OIDC_ISSUER_URL=https://$(oc get route keycloak -n keycloak-system | tail -n 1 |
awk '{print $2}')/auth/realms/trusted-artifact-signer
$ export COSIGN_FULCIO_URL=$(oc get fulcio -o jsonpath='{.items[0].status.url}' -n
trusted-artifact-signer)
$ export COSIGN_REKOR_URL=$(oc get rekor -o jsonpath='{.items[0].status.url}' -n trusted-
artifact-signer)
$ export COSIGN_MIRROR=$TUF_URL
$ export COSIGN_ROOT=$TUF_URL/root.json
$ export COSIGN_OIDC_CLIENT_ID="trusted-artifact-signer"
$ export COSIGN_OIDC_ISSUER=$OIDC_ISSUER_URL
$ export COSIGN_CERTIFICATE_OIDC_ISSUER=$OIDC_ISSUER_URL
$ export COSIGN_YES="true"
$ export SIGSTORE_FULCIO_URL=$COSIGN_FULCIO_URL
$ export SIGSTORE_OIDC_ISSUER=$COSIGN_OIDC_ISSUER
$ export SIGSTORE_REKOR_URL=$COSIGN_REKOR_URL
$ export REKOR_REKOR_SERVER=$COSIGN_REKOR_URL
```

4. 使用 RHTAS 服务配置本地存储库配置以签署您的提交：

示例

```
$ git config --local commit.gpgsign true
$ git config --local tag.gpgsign true
$ git config --local gpg.x509.program gitsign
```

```
$ git config --local gpg.format x509
$ git config --local gitsign.fulcio $SIGSTORE_FULCIO_URL
$ git config --local gitsign.rekor $SIGSTORE_REKOR_URL
$ git config --local gitsign.issuer $SIGSTORE_OIDC_ISSUER
$ git config --local gitsign.clientID trusted-artifact-signer
```

5. 将提交提交到本地存储库：

   示例

   ```
   $ git commit --allow-empty -S -m “Test of a signed commit”
   ```

   此时会打开一个 Web 浏览器，允许您使用电子邮件地址为提交签名。

6. 初始化更新框架(TUF)系统：

   示例

   ```
   $ cosign initialize
   ```

7. 验证提交：

   语法

   ```
   gitsign verify --certificate-identity=SIGNING_EMAIL --certificate-oidc-
   issuer=$SIGSTORE_OIDC_ISSUER HEAD
   ```

   示例

   ```
   $ gitsign verify --certificate-identity=jdoe@redhat.com --certificate-oidc-
   issuer=$SIGSTORE_OIDC_ISSUER HEAD
   ```

其他资源

- 在 OpenShift 上安装 Red Hat Trusted Artifact Signer 。
- 自定义红帽受信任的应用程序管道.
- 有关 签名和验证容器镜像的详细信息，请参阅 RHTAS 部署指南中的使用命令行界面 部分签名和验证容器。
- 更新框架 主页.

2.3. 使用企业合同验证容器镜像上的签名

企业合同(EC)是维护软件供应链安全性的工具，您可以使用它来定义和执行容器镜像的策略。您可以使用 **ec** 二进制文件来验证使用红帽的 Trusted Artifact Signer (RHTAS)签名框架的容器镜像的待测试和签名。

先决条件

- 在 Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15 上安装 RHTAS。
- 安装了 **oc**、**cosign** 和 **podman** 二进制文件的工作站。

- 访问 OpenShift Web 控制台。

流程

1. 从 OpenShift 集群下载 **ec** 二进制文件。

   a. 登录 OpenShift Web 控制台。在主页中，单击 **?** 图标，单击 **Command line tools**，前往 **ec** download 部分，然后单击您的平台的链接。

   b. 在工作站上打开一个终端，解压缩二进制 .gz 文件，并设置执行位：

   Example

   ```
   $ gunzip ec-amd64.gz
   $ chmod +x ec-amd64
   ```

   c. 将二进制文件移到 **$PATH** 环境中的位置：

   Example

   ```
   $ sudo mv ec-amd64 /usr/local/bin/ec
   ```

2. 切换到 RHTAS 项目：

   语法

   ```
   oc project PROJECT_NAME
   ```

   Example

   ```
   $ oc project trusted-artifact-signer
   ```

   注意

   使用 RHTAS 安装的项目名称。

3. 配置 shell 环境，以执行容器镜像签名和验证。

   Example

   ```
   $ export TUF_URL=$(oc get tuf -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
   $ export OIDC_ISSUER_URL=https://$(oc get route keycloak -n keycloak-system | tail -n 1 |
   awk '{print $2}')/auth/realms/trusted-artifact-signer
   $ export COSIGN_FULCIO_URL=$(oc get fulcio -o jsonpath='{.items[0].status.url}' -n
   trusted-artifact-signer)
   $ export COSIGN_REKOR_URL=$(oc get rekor -o jsonpath='{.items[0].status.url}' -n trusted-
   artifact-signer)
   $ export COSIGN_MIRROR=$TUF_URL
   $ export COSIGN_ROOT=$TUF_URL/root.json
   $ export COSIGN_OIDC_CLIENT_ID="trusted-artifact-signer"
   $ export COSIGN_OIDC_ISSUER=$OIDC_ISSUER_URL
   $ export COSIGN_CERTIFICATE_OIDC_ISSUER=$OIDC_ISSUER_URL
   $ export COSIGN_YES="true"
   ```

```
$ export SIGSTORE_FULCIO_URL=$COSIGN_FULCIO_URL
$ export SIGSTORE_OIDC_ISSUER=$COSIGN_OIDC_ISSUER
$ export SIGSTORE_REKOR_URL=$COSIGN_REKOR_URL
$ export REKOR_REKOR_SERVER=$COSIGN_REKOR_URL
```

4. 初始化更新框架(TUF)系统：

Example

```
$ cosign initialize
```

5. 为测试容器镜像签名。

a. 创建一个空容器镜像：

Example

```
$ echo "FROM scratch" > ./tmp.Dockerfile
$ podman build . -f ./tmp.Dockerfile -t ttl.sh/rhtas/test-image:1h
```

b. 将空容器镜像推送到 **ttl.sh** 临时 registry：

Example

```
$ podman push ttl.sh/rhtas/test-image:1h
```

c. 为容器镜像签名：

语法

```
cosign sign -y IMAGE_NAME:TAG
```

Example

```
$ cosign sign -y ttl.sh/rhtas/test-image:1h
```

此时会打开一个 Web 浏览器，允许您使用电子邮件地址为容器镜像签名。

d. 删除临时 Docker 文件：

Example

```
$ rm ./tmp.Dockerfile
```

6. 创建 **predicate.json** 文件：

Example

```
{
  "builder": {
    "id": "https://localhost/dummy-id"
  },
```

```
  "buildType": "https://example.com/tekton-pipeline",
  "invocation": {},
  "buildConfig": {},
  "metadata": {
    "completeness": {
      "parameters": false,
      "environment": false,
      "materials": false
    },
    "reproducible": false
  },
  "materials": []
}
```

有关目的和模式布局的更多信息，请参阅 SLSA 认可 predicate 规格。

7. 将 **predicate.json** 文件与容器镜像关联：

语法

```
cosign attest -y --predicate ./predicate.json --type slsaprovenance IMAGE_NAME:TAG
```

Example

```
$ cosign attest -y --predicate ./predicate.json --type slsaprovenance ttl.sh/rhtas/test-image:1h
```

8. 验证容器镜像是否至少有一个测试和签名：

语法

```
cosign tree IMAGE_NAME:TAG
```

Example

```
$ cosign tree ttl.sh/rhtas/test-image:1h

  Supply Chain Security Related artifacts for an image: ttl.sh/rhtas/test-
image@sha256:7de5fa822a9d1e507c36565ee0cf50c08faa64505461c844a3ce3944d23efa35

└──  Attestations for an image tag: ttl.sh/rhtas/test-image:sha256-
7de5fa822a9d1e507c36565ee0cf50c08faa64505461c844a3ce3944d23efa35.att
   └──
sha256:40d94d96a6d3ab3d94b429881e1b470ae9a3cac55a3ec874051bdecd9da06c2e
└──  Signatures for an image tag: ttl.sh/rhtas/test-image:sha256-
7de5fa822a9d1e507c36565ee0cf50c08faa64505461c844a3ce3944d23efa35.sig
   └──
sha256:f32171250715d4538aec33adc40fac2343f5092631d4fc2457e2116a489387b7
```

9. 使用企业联系人验证容器镜像：

语法

```
ec validate image --image IMAGE_NAME:TAG --certificate-identity-regexp
'SIGNER_EMAIL_ADDR' --certificate-oidc-issuer-regexp 'keycloak-keycloak-system' --output
yaml --show-successes
```

Example

```
$ ec validate image --image ttl.sh/rhtas/test-image:1h --certificate-identity-regexp
'jdoe@example.com' --certificate-oidc-issuer-regexp 'keycloak-keycloak-system' --output
yaml --show-successes

success: true
successes:
  - metadata:
      code: builtin.attestation.signature_check
    msg: Pass
  - metadata:
      code: builtin.attestation.syntax_check
    msg: Pass
  - metadata:
      code: builtin.image.signature_check
    msg: Pass
ec-version: v0.1.2427-499ef12
effective-time: "2024-01-21T19:57:51.338191Z"
key: ""
policy: {}
success: true
```

企业合同生成 pass-fail 报告，其中包含有关任何安全违反情况的详细信息。添加 **--info** 标志时，报告包括所有找到违反情况的详情和可能的解决方案。

其他资源

- 在 OpenShift 上安装 Red Hat Trusted Artifact Signer 。
- 使用企业合同管理合规性.
- 如需更多信息，请参阅 Enterprise Contract 网站。

第 3 章 配置额外的 OPENID CONNECT 供应商

3.1. 将 GOOGLE 配置为 TRUSTED ARTIFACT SIGNER 的 OPENID CONNECT 供应商

您可以将 Google OAuth 2.0 用作 Red Hat Trusted Artifact Signer (RHTAS)服务的 OpenID Connect (OIDC)供应商。您可以决定在RHAS 部署期间或稍后配置 Google OAuth。

重要

您可以在同一配置中定义多个不同的 OIDC 供应商。

先决条件

- Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15。
- 使用 **cluster-admin** 角色访问 OpenShift Web 控制台。
- 安装了 **oc**、**podman** 二进制文件的工作站。
- 在 Google Cloud Console 中，使用以下设置创建一个 OAuth 客户端 ID：
  - 将应用程序类型设置为 "Web Application"。
  - 授权重定向 URI 必须包含：http://localhost/auth/callback。

流程

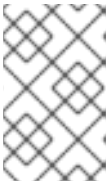
1. 在工作站上打开一个终端，并登录到 OpenShift：

   语法

   ```
   oc login --token=TOKEN --server=SERVER_URL_AND_PORT
   ```

   示例

   ```
   $ oc login --token=sha256~ZvFDBvoIYAbVECixS4-WmkN4RfnNd8Neh3y1WuiFPXC --
   server=https://example.com:6443
   ```

   注意

   您可以从 OpenShift Web 控制台在命令行中查找您的登录令牌和 URL。登录 OpenShift Web 控制台。点您的用户名，然后点 **Copy login 命令**。如果被要求，请再次提供您的用户名和密码，然后单击 **Display Token** 查看命令。

2. 更新 RHTAS 配置。

   a. 打开以编辑 **Securesign** 资源：

   语法

   ```
   oc edit Securesign NAME -n NAMESPACE
   ```

Example

```
$ oc edit Securesign securesign-sample -n trusted-artifact-signer
```

注意

您必须使用为 RHTAS 安装创建的项目名称作为命名空间。

b. 在 **OIDCIssuers** 部分下，使用 Google 客户端标识符、签发者的 URL 添加新子，并将 **Type** 值设置为 **email** ：

语法

```
...
OIDCIssuers:
  - Issuer: "https://accounts.google.com"
    IssuerURL: "https://accounts.google.com"
    ClientID: "CLIENT_ID"
    Type: email
...
```

将 Google 客户端标识符添加到 **ClientID** 字段。

c. 保存更改，退出编辑器。几秒钟后，操作员会自动重新配置 RHTAS 软件堆栈。

3. 更改 OIDC 签发者和客户端 id 环境变量以使用 Google：

示例

```
$ export OIDC_ISSUER_URL=https://accounts.google.com
$ export COSIGN_OIDC_CLIENT_ID="314919563931-
35zke44ouf2oiztjg7v8o8c2ge9usnd1.apps.googleexample.com"
```

4. 将您的 secret 从 Google Console 复制并粘贴到纯文本文件中：

语法

```
echo SECRET > my-google-client-secret
```

5. 如果您已运行 RHTAS 服务，您可以通过签署测试容器镜像来验证更新的配置。

a. 创建一个空容器镜像：

Example

```
$ echo "FROM scratch" > ./tmp.Dockerfile
$ podman build . -f ./tmp.Dockerfile -t ttl.sh/rhtas/test-image:1h
```

b. 将空容器镜像推送到 **ttl.sh** 临时 registry：

Example

```
$ podman push ttl.sh/rhtas/test-image:1h
```

c. 删除临时 Docker 文件：

Example

```
$ rm ./tmp.Dockerfile
```

d. 为容器镜像签名：

语法

```
cosign sign -y --oidc-client-secret-file=SECRET_FILE IMAGE_NAME:TAG
```

Example

```
$ cosign sign -y --oidc-client-secret-file=my-google-client-secret ttl.sh/rhtas/test-image:1h
```

此时会打开一个 Web 浏览器，允许您使用电子邮件地址为容器镜像签名。

其他资源

- 创建 Google OAuth 凭据.

3.2. 将 RED HAT SSO 配置为 TRUSTED ARTIFACT SIGNER 的 OPENID CONNECT 供应商

您可以使用 Red Hat Single Sign-On (SSO)作为红帽的 Trusted Artifact Signer (RHTAS)服务的 OpenID Connect 供应商。这可让您为应用程序和安全服务的 Keycloak 身份验证环境。

先决条件

- Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15。
- 使用 **cluster-admin** 角色访问 OpenShift Web 控制台。
- 有 1 GB 的容器存储可用于 Keycloak PostgreSQL 数据库。
- 安装了 **oc** 二进制文件的工作站。

流程

1. 使用具有 **cluster-admin** 角色的用户登录 OpenShift Web 控制台。
2. 创建一个新项目来部署 Keycloak 服务。
   a. 从 **Administrator** 视角中，从导航菜单中展开 **Home**，再单击 **Projects**。
   b. 点 **Create Project** 按钮。
   c. **新项目名称为 keycloak-system**，然后单击 **Create** 按钮。
3. 从导航菜单中展开 **Operators**，然后点 **OperatorHub**。

4. 在搜索字段中，键入 **sso**，然后单击 **Red Hat Single Sign-on** 标题。
5. 单击 **Install** 按钮，以显示 Operator 详情。
6. 如果尚未设置，请从 **Installed Namespace** 下拉菜单中选择 **keycloak-system**。
7. **在** *Install Operator* 页面中点 Install，并等待安装完成。
8. 安装完成后，点 **View Operator**。
9. 在 workstation 终端中登录到 OpenShift 集群：

   语法

   ```
   oc login --token=TOKEN --server=SERVER_URL_AND_PORT
   ```

   Example

   ```
   $ oc login --token=sha256~ZvFDBvoIYAbVECixS4-WmkN4RfnNd8Neh3y1WuiFPXC --
   server=https://example.com:6443
   ```

   注意

   您可以从 OpenShift Web 控制台查找要在命令行中使用的登录令牌和 URL。登录 OpenShift Web 控制台。点您的用户名，然后点 **Copy login 命令**。如果被要求，请再次提供您的用户名和密码，然后单击 **Display Token** 查看命令。

10. 切换到 Keycloak 项目：

    Example

    ```
    $ oc project keycloak-system
    ```

11. 创建 Keycloak 实例：

    Example

    ```
    $ cat <<EOF | oc apply -f -
    apiVersion: keycloak.org/v1alpha1
    kind: Keycloak
    metadata:
      labels:
        app: sso
      name: keycloak
    spec:
      externalAccess:
        enabled: true
      instances: 1
      keycloakDeploymentSpec:
        imagePullPolicy: Always
      postgresDeploymentSpec:
        imagePullPolicy: Always
    EOF
    ```

12. 创建 Keycloak 域：

Example

```
$ cat <<EOF | oc apply -f -
apiVersion: keycloak.org/v1alpha1
kind: KeycloakRealm
metadata:
  labels:
    app: sso
  name: trusted-artifact-signer
spec:
  instanceSelector:
    matchLabels:
      app: sso
  realm:
    displayName: Red-Hat-Trusted-Artifact-Signer
    enabled: true
    id: trusted-artifact-signer
    realm: trusted-artifact-signer
    sslRequired: none
EOF
```

13. 创建 Keycloak 客户端：

Example

```
$ cat <<EOF | oc apply -f -
apiVersion: keycloak.org/v1alpha1
kind: KeycloakClient
metadata:
  labels:
    app: sso
  name: trusted-artifact-signer
spec:
  client:
    attributes:
      request.object.signature.alg: RS256
      user.info.response.signature.alg: RS256
    clientAuthenticatorType: client-secret
    clientId: trusted-artifact-signer
    defaultClientScopes:
    - profile
    - email
    description: Client for Red Hat Trusted Artifact Signer authentication
    directAccessGrantsEnabled: true
    implicitFlowEnabled: false
    name: trusted-artifact-signer
    protocol: openid-connect
    protocolMappers:
    - config:
        claim.name: email
        id.token.claim: "true"
        jsonType.label: String
        user.attribute: email
```

```
        userinfo.token.claim: "true"
      name: email
      protocol: openid-connect
      protocolMapper: oidc-usermodel-property-mapper
    - config:
        claim.name: email-verified
        id.token.claim: "true"
        user.attribute: emailVerified
        userinfo.token.claim: "true"
      name: email-verified
      protocol: openid-connect
      protocolMapper: oidc-usermodel-property-mapper
    - config:
        claim.name: aud
        claim.value: trusted-artifact-signer
        id.token.claim: "true"
        access.token.claim: "true"
        userinfo.token.claim: "true"
      name: audience
      protocol: openid-connect
      protocolMapper: oidc-hardcoded-claim-mapper
    publicClient: true
    standardFlowEnabled: true
    redirectUris:
    - "*"
  realmSelector:
    matchLabels:
      app: sso
EOF
```

14. 创建 Keycloak 用户：

Example

```
$ cat <<EOF | oc apply -f -
apiVersion: keycloak.org/v1alpha1
kind: KeycloakUser
metadata:
  labels:
    app: sso
  name: jdoe
spec:
  realmSelector:
    matchLabels:
      app: sso
  user:
    email: jdoe@redhat.com
    enabled: true
    emailVerified: true
    credentials:
      - type: "password"
        value: "secure"
    firstName: Jane
    lastName: Doe
    username: jdoe
EOF
```

- 设置用户名、用户的电子邮件地址以及密码或引用 secret 对象。

15. 返回到 OpenShift Web 控制台，点 **All instances** 选项卡监视并等待 Keycloak 系统成功初始化。

其他资源

- 在 OpenShift 上安装 Red Hat Trusted Artifact Signer 。

3.3. 将 AMAZON STS 配置为 TRUSTED ARTIFACT SIGNER 的 OPENID CONNECT 供应商

您可以将 Amazon 的安全令牌服务(STS)用作红帽受信任的工件签名程序(RHTAS)服务的 OpenID Connect (OIDC)供应商。您可以决定在RHAS 部署期间或稍后配置 Amazon STS。

重要

您可以在同一配置中定义多个不同的 OIDC 供应商。

先决条件

- 在 Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15 上安装 RHTAS。
- 使用 **cluster-admin** 角色访问 OpenShift Web 控制台。
- 安装了 **oc**、**podman** 和 **aws** 二进制文件的工作站。
- 为 OpenShift 环境启用受管 Amazon Web Service (AWS)资源。
- 创建了一个具有完整权限的 Amazon Identity and Access Management (IAM)用户 。这允许访问运行 IAM 操作。
  - 为此用户创建访问密钥。

流程

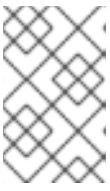
1. 在工作站上打开一个终端，并登录到 OpenShift：

语法

```
oc login --token=TOKEN --server=SERVER_URL_AND_PORT
```

Example

```
$ oc login --token=sha256~ZvFDBvoIYAbVECixS4-WmkN4RfnNd8Neh3y1WuiFPXC --
server=https://example.com:6443
```



注意

您可以从 OpenShift Web 控制台在命令行中查找您的登录令牌和 URL。登录 OpenShift Web 控制台。点您的用户名，然后点 **Copy login 命令**。如果被要求，请再次提供您的用户名和密码，然后单击 **Display Token** 查看命令。

2. 查找 AWS OIDC 供应商 URL：

Example

```
$ oc get authentication cluster -o jsonpath='{.spec.serviceAccountIssuer}'
```

3. 更新 RHTAS 配置。

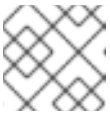
a. 打开以编辑 **Securesign** 资源：

语法

```
oc edit Securesign NAME -n NAMESPACE
```

Example

```
$ oc edit Securesign securesign-sample -n trusted-artifact-signer
```



注意

您必须使用为 RHTAS 安装创建的项目名称作为命名空间。

b. 在 **OIDCIssuers** 部分下，使用 AWS STS 客户端标识符、签发者的 URL 添加新子，并将 **Type** 值设置为 **kubernetes** ：

Example

```
...
OIDCIssuers:
  - Issuer: "https://example.s3.us-east-1.aws.com/47bd6cg0vs5nn01mue83fbof94dj4m9c"
    IssuerURL: "https://example.s3.us-east-
1.aws.com/47bd6cg0vs5nn01mue83fbof94dj4m9c"
    ClientID: "trusted-artifact-signer"
    Type: kubernetes
...
```

c. 保存更改，退出编辑器。几秒钟后，操作员会自动重新配置 RHTAS 软件堆栈。

4. 输入您的访问密钥、secret 密钥、默认区域和输出格式来配置 AWS 命令行工具：

Example

```
$ aws configure
```

5. 设置以下环境变量：

Example

```
$ export account_id=$(aws sts get-caller-identity --query "Account" --output text)
$ export oidc_provider="$(oc get authentication cluster -o
jsonpath='{.spec.serviceAccountIssuer}' | cut -d '/' -f3-)"
```

```
$ export role_name=rhtas-sts
$ export namespace=rhtas-sts
$ export service_account=cosign-sts
```

6. 创建与新创建的 IAM 角色关联的信任策略：

Example

```
$ cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::${account_id}:oidc-provider/${oidc_provider}"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "${oidc_provider}:aud": "trusted-artifact-signer"
        }
      }
    }
  ]
}
EOF
```

7. 使用信任策略为 RHTAS 服务创建一个新的 IAM 角色：

Example

```
$ aws iam create-role --role-name rhtas-sts --assume-role-policy-document file://trust-relationship.json --description "Red Hat Trusted Artifact Signer STS Role"
```

8. 在启用了 STS 的 OpenShift 集群中，创建一个新项目命名空间：

语法

```
oc new-project NAMESPACE
```

Example

```
$ oc new-project rhtas-sts
```

9. 为假设 IAM 角色创建服务帐户，并在 OpenShift 项目命名空间中运行工作负载。

   a. 创建服务帐户清单：

   Example

   ```
   $ cat >service_account.yaml <<EOF
   apiVersion: v1
   ```

```
kind: ServiceAccount
metadata:
  name: $service_account
  namespace: $namespace
  annotations:
    eks.amazonaws.com/role-arn: "arn:aws:iam::${account_id}:role/${role_name}"
    # optional: Defaults to "sts.amazonaws.com" if not set
    eks.amazonaws.com/audience: "trusted-artifact-signer"
    # optional: When "true", adds AWS_STS_REGIONAL_ENDPOINTS env var to
containers
    eks.amazonaws.com/sts-regional-endpoints: "true"
    # optional: Defaults to 86400 for expirationSeconds if not set
    eks.amazonaws.com/token-expiration: "86400"
EOF
```

b. 将服务帐户清单应用到 OpenShift：

Example

```
$ oc apply -f service_account.yaml
```

10. 创建新的部署工作负载，以便在镜像 registry 中签名容器镜像。

a. 创建部署清单：

Example

```
$ cat >deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cosign-sts
  namespace: ${namespace}
spec:
  selector:
    matchLabels:
      app: cosign-sts
  template:
    metadata:
      labels:
        app: cosign-sts
    spec:
      securityContext:
        runAsNonRoot: true
      serviceAccountName: cosign-sts
      containers:
      - args:
        - -c
        - env; cosign initialize --mirror=\$COSIGN_MIRROR --root=\$COSIGN_ROOT;
while true; do sleep 86400; done
        command:
        - /bin/sh
        name: cosign
        image: registry.redhat.io/rhtas-tech-preview/cosign-
rhel9@sha256:f4c2cec3fc1e24bbe094b511f6fe2fe3c6fa972da0edacaf6ac5672f06253a3e
```

```
        pullPolicy: IfNotPresent
        env:
        - name: AWS_ROLE_SESSION_NAME
          value: signer-identity-session
        - name: AWS_REGION
          value: us-east-1
        - name: OPENSHIFT_APPS_SUBDOMAIN
          value: $(oc get cm -n openshift-config-managed  console-public -o go-template="{{
.data.consoleURL }}" | sed 's@https://@@; s/^[^.]*\.//')
        - name: OIDC_AUTHENTICATION_REALM
          value: "trusted-artifact-signer"
        - name: COSIGN_FULCIO_URL
          value: $(oc get fulcio -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
        - name: COSIGN_OIDC_ISSUER
          value: $(oc get authentication cluster -o jsonpath='{.spec.serviceAccountIssuer}')
        - name: COSIGN_CERTIFICATE_OIDC_ISSUER
          value: $(oc get authentication cluster -o jsonpath='{.spec.serviceAccountIssuer}')
        - name: COSIGN_REKOR_URL
          value: $(oc get rekor -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
        - name: COSIGN_MIRROR
          value: $(oc get tuf -o jsonpath='{.items[0].status.url}' -n trusted-artifact-signer)
        - name: COSIGN_ROOT
          value: "$(oc get tuf -o jsonpath='{.items[0].status.url}' -n trusted-artifact-
signer)/root.json"
        - name: COSIGN_YES
          value: "true"
        securityContext:
          allowPrivilegeEscalation: false
          capabilities:
            drop:
            - ALL
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext:
        runAsNonRoot: true
      serviceAccount: ${service_account}
      serviceAccountName: ${service_account}
      terminationGracePeriodSeconds: 30
EOF
```

b. 将部署清单应用到 OpenShift：

Example

```
$ oc apply -f deployment.yaml
```

11. 创建测试容器镜像以签名。

a. 创建一个空容器镜像：

Example

```
$ echo "FROM scratch" > ./tmp.Dockerfile
$ podman build . -f ./tmp.Dockerfile -t ttl.sh/rhtas/test-image:1h
```

b. 将空容器镜像推送到 **ttl.sh** 临时 registry：

Example

```
$ podman push ttl.sh/rhtas/test-image:1h
```

c. 删除临时 Docker 文件：

Example

```
$ rm ./tmp.Dockerfile
```

12. 通过签名和验证测试容器镜像来验证配置。

a. 在正在运行的 pod 中打开远程 shell 会话：

语法

```
oc rsh -n NAMESPACE deployment/cosign-sts env IMAGE=IMAGE_NAME:TAG /bin/sh
```

Example

```
$ oc rsh -n rhtas-sts deployment/cosign-sts env IMAGE=ttl.sh/rhtas/test-image:1h /bin/sh
```

b. 为容器镜像签名：

Example

```
$ cosign sign -y --identity-token=$(cat $AWS_WEB_IDENTITY_TOKEN_FILE)
ttl.sh/rhtas/test-image:1h
```

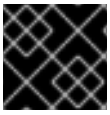
c. 验证签名的容器镜像：

Example

```
$ cosign verify --certificate-identity=https://kubernetes.io/namespaces/$(cat
/var/run/secrets/kubernetes.io/serviceaccount/namespace)/serviceaccounts/cosign-sts --
certificate-oidc-issuer=$COSIGN_CERTIFICATE_OIDC_ISSUER ttl.sh/rhtas/test-
image:1h
```

3.4. 将 GITHUB 配置为 TRUSTED ARTIFACT SIGNER 的 OPENID CONNECT 供应商

当使用红帽单点登录(SSO)服务作为 Red Hat Trusted Artifact Signer (RHTAS)服务的 OpenID Connect (OIDC)服务时，您可以使用 GitHub OAuth 2.0。此流程指导您将 GitHub OAuth 与 OpenShift 上的现有 Red Hat SSO 部署集成。



重要

您可以在同一配置中定义多个不同的 OIDC 供应商。

先决条件

- 在 Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15 上安装 RHTAS。
- 正在运行的 Red Hat SSO 实例。
- 安装了 **oc** 二进制文件的工作站。
- 创建 GitHub OAuth 应用，并在注册应用后记录客户端标识符和机密值。

重要

在注册新的 GitHub OAuth 应用程序时，您必须指定 **Homepage URL** 和 **Authorization 回调 URL**。为这两个字段输入占位符值，例如 **https://localhost:8080**。稍后，您将使用这些字段的预期值修改 GitHub OAuth 应用程序。

流程

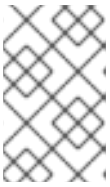
1. 在工作站上打开一个终端，并登录到 OpenShift：

   语法

   ```
   oc login --token=TOKEN --server=SERVER_URL_AND_PORT
   ```

   Example

   ```
   $ oc login --token=sha256~ZvFDBvoIYAbVECixS4-WmkN4RfnNd8Neh3y1WuiFPXC --
   server=https://example.com:6443
   ```

   注意

   您可以从 OpenShift Web 控制台在命令行中查找您的登录令牌和 URL。登录 OpenShift Web 控制台。点您的用户名，然后点 **Copy login 命令**。如果被要求，请再次提供您的用户名和密码，然后单击 **Display Token** 查看命令。

2. 登录到 Red Hat SSO 控制台。

   a. 从命令行查找 Red Hat SSO 控制台 URL：

      Example

      ```
      $ oc get routes -n keycloak-system keycloak -o jsonpath='https://{.spec.host}'
      ```

   b. 将 Red Hat SSO 控制台 URL 复制并粘贴到您的 Web 浏览器中。

   c. 单击 **Administration Console**。

   d. 从命令行检索 **admin** 密码：

Example

```
$ oc get secret/credential-keycloak -n keycloak-system -o jsonpath='{
.data.ADMIN_PASSWORD }' | base64 -d
```

复制此命令的输出。

e. 在 Web 浏览器中，以 **admin** 用户身份登录，并将密码粘贴到对应的字段中。点 **Sign In** 按钮。

3. 从导航菜单的下拉菜单中选择您的域。
4. 添加 GitHub 身份提供程序。
   a. 在导航菜单中点 **Identity Providers**。
   b. 在 **Add provider...** 下拉菜单中选择 **GitHub**。
   c. 将 GitHub OAuth 客户端标识符添加到 **客户端 ID** 字段。
   d. 将 GitHub OAuth 客户端 secret 添加到 **Client Secret** 字段中。
   e. **打开 Trust Email** 选项。
   f. 点 **Save** 按钮。
5. 将身份提供程序映射程序添加到新创建的身份提供程序。
   a. 点 **Mapper** 选项卡。
   b. 点 **Create** 按钮。
   c. 为新映射程序指定 **Name**。
   d. 将 **Mapper Type** 更改为 **Hardcoded Attribute**。
   e. 将 **User Attribute** 字段设置为 **emailVerified**。
   f. 将 **User Attribute Value** 字段设置为 **true**。
   g. 点 **Save** 按钮。
6. 在 *GitHub Identity Provider Settings* 页面中，复制 **Redirect URI** 值，并将它粘贴到 GitHub OAuth 应用 **授权回调 URL** 字段。另外，将此值粘贴到 **Homepage URL** 字段中，但删除 URL 字符串的 **broker/github/endpoint** 部分。
7. 单击 **Update Application**。现在，您可以使用 GitHub 作为 OIDC 供应商来签署 提交和 容器。
8. 在签名工件时，会打开网页浏览器并提示您登录到 Red Hat SSO 帐户。点 **GitHub** 按钮使用您的凭证登录。
9. 单击 **Authorize** 按钮，以启用 GitHub 用户详情，以供 Red Hat SSO 访问。

其他资源

- 在 OpenShift 上安装 Red Hat Trusted Artifact Signer 。

- 使用 cosign 进行签名和验证。
- 使用 gitsign 进行签名和验证。

第 4 章 为 TRUSTED ARTIFACT SIGNER 配置替代数据库

您可以将 Trillian 的 Red Hat Trusted Artifact Signer (RHTAS)默认数据库替换为外部管理的 MariaDB 数据库实例。数据库实例可以是云托管的数据库提供程序，如 Amazon 的 Relational Database Service (RDS)，或者在 OpenShift 中自己的数据库部署。

4.1. 先决条件

- Red Hat OpenShift Container Platform 版本 4.13、4.14 或 4.15。

4.2. 为受信任的工件签名程序配置 AMAZON RDS

使用这个流程，您可以将红帽的 Trusted Artifact Signer (RHTAS)默认数据库替换为 Amazon 的 Relational Database Service (RDS)管理的 MariaDB 实例。

重要

红帽建议在生产环境中使用高可用性 MariaDB 数据库。

先决条件

- 可访问 Amazon RDS 控制台的 Amazon Web Service (AWS)帐户。
- 使用 **cluster-admin** 角色访问 OpenShift Web 控制台。
- 安装了 **oc**、**curl** 和 **mysql** 二进制文件的工作站。
- 具有特权的命令行访问权限，以创建数据库并填充 MariaDB 实例。

流程

1. 打开 Amazon RDS 控制台，并创建一个新的 MariaDB 实例。
   a. 等待 MariaDB 实例部署好，并且可用。
2. 从您的工作站，通过提供区域端点、端口和用户凭证来登录到新数据库：

   语法

   ```
   mysql -h REGIONAL_ENDPOINT -P 3306 -u USER_NAME -p
   ```

   Example

   ```
   $ mysql -h exampledb.1234.us-east-1.rds.amazonaws.com -P 3306 -u admin -p
   ```

3. 创建名为 trillian 的新数据库：

   Example

   ```
   create database trillian;
   ```

4. 切换到新创建的数据库：

Example

```
use trillian;
```

5. 创建名为 **trillian** 的新数据库用户，并为新创建的用户设置 *PASSWORD* ：

语法

```
CREATE USER trillian@'%' IDENTIFIED BY 'PASSWORD';
GRANT ALL PRIVILEGES ON trillian.* TO 'trillian'@'%';
FLUSH PRIVILEGES;
```

6. 断开与数据库的连接：

Example

```
EXIT
```

7. 下载数据库配置文件：

Example

```
$ curl -o dbconfig.sql
https://raw.githubusercontent.com/securesign/trillian/main/storage/mysql/schema/storage.sql
```

8. 将数据库配置应用到新数据库：

语法

```
mysql -h FQDN_or_SERVICE_ADDR -P 3306 -u USER_NAME -p PASSWORD -D
DB_NAME < PATH_TO_CONFIG_FILE
```

Example

```
$ mysql -h rhtasdb.example.com -P 3306 -u trillian -p mypassword123 -D trillian <
dbconfig.sql
```

9. 在工作站上打开一个终端，并登录到 OpenShift：

语法

```
oc login --token=TOKEN --server=SERVER_URL_AND_PORT
```

Example

```
$ oc login --token=sha256~ZvFDBvoIYAbVECixS4-WmkN4RfnNd8Neh3y1WuiFPXC --
server=https://example.com:6443
```

注意

您可以从 OpenShift Web 控制台在命令行中查找您的登录令牌和 URL。登录 OpenShift Web 控制台。点您的用户名，然后点 **Copy login 命令**。如果被要求，请再次提供您的用户名和密码，然后单击 **Display Token** 查看命令。

10. 创建一个新的 Secret，其中包含之前创建的 MariaDB 实例中 Trillian 数据库的凭证：

语法

```
oc create secret generic OBJECT_NAME \
--from-literal=mysql-database=trillian \
--from-literal=mysql-host=FQDN_or_SERVICE_ADDR \
--from-literal=mysql-password=PASSWORD \
--from-literal=mysql-port=3306 \
--from-literal=mysql-root-password=PASSWORD \
--from-literal=mysql-user=USER_NAME
```

Example

```
$ oc create secret generic trillian-mysql \
--from-literal=mysql-database=trillian \
--from-literal=mysql-host=mariadb.trusted-artifact-signer.svc.cluster.local \
--from-literal=mysql-password=mypassword123 \
--from-literal=mysql-port=3306 \
--from-literal=mysql-root-password=myrootpassword123 \
--from-literal=mysql-user=trillian
```

您可以将 OpenShift 内部服务名称用于 MariaDB 实例。

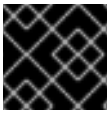
11. 现在，您可以部署 Trusted Artifact Signer 服务来使用此数据库。如果您遵循 Trusted Artifact Signer 安装过程，您可以 继续下一步。

其他资源

- 在 OpenShift 中创建新机密对象。

4.3. 在 OPENSHIFT 中为受信任的工件签名程序配置数据库

使用这个流程，您可以将红帽的 Trusted Artifact Signer (RHTAS)默认数据库替换为 Amazon 的 Relational Database Service (RDS)管理的 MariaDB 实例。



重要

红帽建议在生产环境中使用高可用性 MariaDB 数据库。

先决条件

- 创建 OpenShift 项目的权限，并从 OpenShift 示例目录部署数据库实例。
- 使用 **cluster-admin** 角色访问 OpenShift Web 控制台。
- 安装了 **oc**、**curl** 和 **mysql** 二进制文件的工作站。

- 具有特权的命令行访问权限，以创建数据库并填充 MariaDB 实例。

流程

1. 登录到您要部署 RHTAS 服务的 OpenShift Web 控制台：
2. 进入 **Developer** 视角。
3. 如果项目已存在，选择 **trusted-artifact-signer** 项目，否则为数据库创建一个新项目：
   a. 若要创建新项目，请单击下拉菜单，再 **单击创建项目按钮**。
   b. 将新项目命名为 **trusted-artifact-signer**，然后点 **Create** 按钮。
4. 在 *Developer Catalog* 卡上，单击 **Database**。
5. 选择 **MariaDB**，然后单击 **Instantiate Template** 按钮。

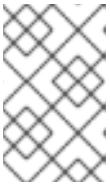
重要

不要选择 **MariaDB (Ephemeral)**。

6. 在 *Instantiate Template* 页面中，配置以下字段：
   a. 在 **MariaDB Database Name** 字段中，输入 **trillian**。
   b. 在 **Volume Capacity** 字段中，输入 **5Gi**。
   c. 点 **Create** 按钮。
7. 开始远程 shell 会话：
   a. 在 *Topology* 页面中，选择 MariaDB pod 会打开一个侧面板，点 **Resources** 选项卡。
   b. 在 *Pods* 部分下，点 MariaDB pod 名称。
   c. 点 **Terminal** 选项卡启动到 MariaDB pod 的远程 shell 会话。
8. 在远程 shell 会话中，验证您可以连接到 Trillian 数据库：

Example

```
$ mysql -u $MYSQL_USER -p$MYSQL_PASSWORD -D$MYSQL_DATABASE
```



注意

凭据通过服务名称(**mariadb**)存储在机密对象中，包含数据库的名称和用户名，以及数据库 root 密码。记录这些凭证，因为稍后在创建数据库 secret 对象时将使用这些凭证。

9. 断开与数据库的连接：

Example

```
EXIT
```

10. 下载数据库配置文件：

Example

```
$ curl -o dbconfig.sql
https://raw.githubusercontent.com/securesign/trillian/main/storage/mysql/schema/storage.sql
```

11. 将数据库配置应用到新数据库：

语法

```
mysql -h FQDN_or_SERVICE_ADDR -P 3306 -u USER_NAME -p PASSWORD -D
DB_NAME < PATH_TO_CONFIG_FILE
```

Example

```
$ mysql -h rhtasdb.example.com -P 3306 -u trillian -p mypassword123 -D trillian <
dbconfig.sql
```

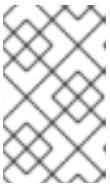
12. 在工作站上打开一个终端，并登录到 OpenShift：

语法

```
oc login --token=TOKEN --server=SERVER_URL_AND_PORT
```

Example

```
$ oc login --token=sha256~ZvFDBvoIYAbVECixS4-WmkN4RfnNd8Neh3y1WuiFPXC --
server=https://example.com:6443
```



注意

您可以从 OpenShift Web 控制台在命令行中查找您的登录令牌和 URL。登录 OpenShift Web 控制台。点您的用户名，然后点 **Copy login 命令**。如果被要求，请再次提供您的用户名和密码，然后单击 **Display Token** 查看命令。

13. 创建一个新的 Secret，其中包含之前创建的 MariaDB 实例中 Trillian 数据库的凭证：

语法

```
oc create secret generic OBJECT_NAME \
--from-literal=mysql-database=trillian \
--from-literal=mysql-host=FQDN_or_SERVICE_ADDR \
--from-literal=mysql-password=PASSWORD \
--from-literal=mysql-port=3306 \
--from-literal=mysql-root-password=PASSWORD \
--from-literal=mysql-user=USER_NAME
```

Example

```
$ oc create secret generic trillian-mysql \
```

```
--from-literal=mysql-database=trillian \
--from-literal=mysql-host=mariadb.trusted-artifact-signer.svc.cluster.local \
--from-literal=mysql-password=mypassword123 \
--from-literal=mysql-port=3306 \
--from-literal=mysql-root-password=myrootpassword123 \
--from-literal=mysql-user=trillian
```

您可以将 OpenShift 内部服务名称用于 MariaDB 实例。

14. 现在，您可以部署 Trusted Artifact Signer 服务来使用此数据库。如果您遵循 Trusted Artifact Signer 安装过程，您可以 继续下一步。

其他资源

- 在 OpenShift 中创建新机密对象。

附录 A. 可信 ARTIFACT SIGNER 组件和版本号

下表列出了红帽的 Trusted Artifact Signer (RHTAS)软件组件及其对应的版本用于 1.0.2 版本。

表 A.1. 二进制文件

组件	Version
Cosign	2.2.3
gitsign	0.8.1
rekor	1.3.6
EC	0.2.1

表 A.2. Trillian

组件	Version
日志服务器	1.6.0
Log Signer	1.6.0
数据库	1.6.0
Redis	1.6.0

表 A.3. Rekor

组件	Version
服务器	1.3.6
回填 Redis	1.3.6
Rekor Search UI	1.3.6

表 A.4. Fulcio

组件	Version
服务器	1.4.5

表 A.5. 证书转换

组件	Version
certificate Transparency Go	1.1.8

表 A.6. 构建

组件	Version
更新框架(TUF)服务器	0.6.16
createctconfig	0.6.16
ctlog-managectroots	0.6.16
trillian-createtree	0.6.16
trillian-createdb	0.6.16
fulcio-createcerts	0.6.16

其他资源

- 有关 Trillian 的更多信息，请参阅 GitHub 上的项目页面。
- 有关 Sigstore 的 Rekor 的更多信息，请参阅 GitHub 上的项目页面。
- 有关 Sigstore 的 Fulcio 的更多信息，请参阅 GitHub 上的项目页面。
- 有关 Sigstore 的 Timestamp Authority 的更多信息，请参阅 GitHub 上的项目页面。
- 有关 TUF 的更多信息，请查看由 Linux Foundation 托管的 主页。