Red Hat Trusted Application Pipeline 1.0

管理符合企业合同的合规性

了解企业合同如何使您更好地验证和管理您提升的代码合规性。另外，自定义示例策略以适合您的企业标准。

Last Updated: 2024-07-02

Red Hat Trusted Application Pipeline 1.0 管理符合企业合同的合规性

了解企业合同如何使您更好地验证和管理您提升的代码合规性。另外，自定义示例策略以适合您的企业标准。

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

摘要

本文档提供了有关如何通过定义和执行策略来构建和测试容器镜像来管理和维护软件供应链安全的信息。

目录

前言

企业合同是一个策略驱动的工作流工具，用于通过定义和强制实施构建和测试容器镜像的策略来维护软件供应链安全性。安全 CI/CD 工作流应包含工件验证，以尽早检测问题。它是企业合同的职责，用于验证容器镜像是否已签名，并由已知和可信构建系统测试。

第 1 章 红帽受信任的应用程序 PIPELINE 的企业合同

软件供应链越复杂，使用可靠的检查和最佳实践来保证软件工件完整性和源代码的依赖性更为重要。镜像容器等工件。这是 Red Hat Enterprise Contract 进入 Red Hat Trusted Application Pipeline 构建并部署体验的地方。

企业合同是一个策略驱动的工作流工具，用于通过定义和强制实施构建和测试容器镜像的策略来维护软件供应链安全性。对于为软件工件(SLSA)验证创建 Supply-chain 级别的构建系统，例如，Tekton with Tekton Chains 和 GitHub Actions with SLSA GitHub Generator，检查签名并确认 attestations 的内容实际上与预期内容匹配，这是验证和维护软件供应链的完整性的一部分。安全 CI/CD 工作流应包含工件验证，以尽早检测问题。它是企业合同的职责，用于验证容器镜像是否已签名，并由已知和可信构建系统测试。

验证签名和测试的容器镜像的一般步骤如下：

1. 使用红帽受信任的应用程序管道创建或复制容器镜像。
2. 使用 Cosign 生成签名密钥。
3. 使用 Cosign 为容器镜像签名。
4. 使用 Cosign 准备镜像。
5. 使用企业合同 CLI 验证已签名且已测试的容器镜像。

但是，它意味着向软件工件（如容器镜像） *签名* 和测试是什么？*为什么选择它？怎么样？*

签名的软件工件（如容器镜像）会比未签名工件造成几个攻击向量的风险显著降低。当容器镜像被签名时，各种加密技术将镜像绑定到特定的实体或机构。结果是一个数字签名，用于验证镜像的真实性，以便您可以将其追溯到其创建者（实体或组织），同时还验证镜像在签名后没有被更改或篡改。有关软件供应链威胁的更多信息，请参阅 Supply 链威胁。

企业合同使用行业标准 Sigstore Cosign 作为库来验证您的容器镜像。有了 Trusted Artifact Signer，红帽支持的 Sigstore 框架版本，您可以使用 Sigstore 服务的自己的 on-prem 实例使用 Cosign CLI 签署并测试您的容器镜像。有关 Trusted Artifact Signer 的更多信息，请参阅 Red Hat Trusted Artifact Signer 。

与测试的软件 工件 一样，在没有验证的情况下无法发生这种情况。经过验证的 有关软件工件（如容器镜像）的可验证信息，用于描述该工件的生成位置、时间和方式。attestation 本身是一个经过身份验证的声明，采用元数据的形式，证明工件是完好且值得信任的。企业合同使用这些合同以加密方式验证构建是否未被篡改，并根据一组策略（如 SLSA 要求）检查构建。有关 SLSA 的更多信息，请参阅关于 SLSA。

当您将代码从 RHTAP 开发命名空间推送到 stage 命名空间时，或从 stage 命名空间推送到生产环境命名空间时，企业合同会自动运行其验证检查，以确保容器镜像已签名并被已知和可信构建系统测试。当您的镜像通过企业合同检查时，您可以将代码更改合并到一个环境到下一个环境。有关将应用程序部署到不同命名空间的更多信息，请参阅 受信任的应用程序管道软件模板。有关 RHTAP 保存部署清单的位置的更多信息，请参阅 RHTAP GitOps 存储库 及其 YAML 文件。

其他资源

有关签名和测试容器镜像的更多信息，请参阅 签名容器镜像。

第 2 章 安装 ENTERPRISE CONTRACT 命令行

先决条件

- 在 Red Hat OpenShift Container Platform 版本 4.13 或更高版本上安装 Red Hat Trusted Artifact Signer。
- 安装了 **cosign** 和 **oc** 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，选择 **Command line tools**，进入 **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，然后设置执行位：

   Example

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

2. 将二进制文件移到 **$PATH** 环境中的位置：

   Example

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

验证

- 运行 **ec version** 命令。结果应该是您刚刚安装的企业合同 CLI 的版本。

第 3 章 创建策略

Enterprise Contract 策略是规则或一组规则和特定于企业合同的注解。企业合同可以执行多种策略检查，包括检查 红帽产品所需的所有策略规则。企业合同使用名为 Open Policy Agent 或 OPA 的一般目的策略引擎。opa 使用自己的语言（称为 Rego ）定义其策略规则。这意味着，在 Enterprise Contract 策略中的 OPA 中的策略规则也会在 Rego 中定义。

流程

1. 创建一个 Rego 文件来定义一个新的策略规则，如下例所示：

```
echo 'package zero_to_hero

import future.keywords.contains
import future.keywords.if
import future.keywords.in


# METADATA (1)
# title: Builder ID
# description: Verify the SLSA Provenance has the builder.id set to
#   the expected value.
# custom:
#   short_name: builder_id (2)
#   failure_msg: The builder ID %q is not the expected %q
#   solution: >-
#     Ensure the correct build system was used to build the container
#     image.
deny contains result if {
 some attestation in input.attestations (3)
 attestation.statement.predicateType == "https://slsa.dev/provenance/v0.2"

 expected := "https://localhost/dummy-id"
 got := attestation.statement.predicate.builder.id

 expected != got

 result := {
  "code": "zero_to_hero.builder_id",
  "msg": sprintf("The builder ID %q is not expected, %q", [got, expected])
 }
}
' > rules.rego
```

(1) **METADATA** 注释块 - 前面有 10 行的代码，它们都以 hashtags (#)--is 指定规则注解，以便企业合同可以包括这些注解详情其成功和违反报告。有关 rego 元数据和注解的更多信息，请参阅 元数据。有关企业合同策略规则必须包含的注解的更多信息，请参阅 规则注解。

(2) 此单一策略规则验证您的新策略规则中的 **builder.id** 是否与您的 Software Artifacts 的 Supply-chain Levels 中的 **builder.id** 匹配，或 SLSA。

(3) **input.attestations** 是一个 rego 对象，其中包含有关容器镜像、其签名及其 attestations 的所有信息。如需有关企业合同定义 **input.attestations** 内容的位置和方式的更多信息，请参阅 策略输入。

提示

您可以将 **input.attestations** 对象保存到 JSON 文件中，以便在指定新策略规则时从其中浏览。要将 **input.attestations** 保存为 JSON 文件，请运行类似以下示例的命令：

```
ec validate image --public-key cosign.pub --image "$REPOSITORY:latest" --policy
policy.yaml \
    --show-successes --info --output yaml
```

2. 创建策略配置以使用您的新策略规则，如下例所示：

```
echo "
---
sources:
  - policy:
      - $(pwd)/rules.rego
" > policy.yaml
```

3. 使用您的新策略验证您的容器镜像，并在成功和违反报告中显示其他信息，如下例所示：

```
ec validate image --public-key cosign.pub --image "$REPOSITORY:latest" --policy
policy.yaml \
    --show-successes --info --output yaml
```

验证

- 检查 Successes 和 violations 报告，以确保您的新规则位于 **successes** 列表中。

其他资源

- 有关一组有用的企业合同策略规则，请参阅 ec-policies GitHub 存储库。
- 有关 OPA 和 Rego 的更多信息，请参阅 OPA 的策略 语言 内容。
- 有关 SLSA 认可的更多信息，请参阅 SLSA Provenance。

3.1. 配置策略

您可以使用内联 JSON 或 YAML 字符串配置 Enterprise Contract 策略。此策略有时称为配置或 合同，指定企业合同应在哪里查找用于应用您要强制执行的策略的规则和数据。您还可以包含或排除单个规则或特定规则软件包。

流程

1. 在命令行中将策略配置为 JSON 或 YAML 字符串，如下例所示：

```
ec validate image --policy '{
    "configuration": {
        "include": ["@minimal"]
    },
    "sources": [
        {
            "policy": ["oci::quay.io/enterprise-contract/ec-release-policy:latest"],
```

```
            "data": ["git::https://github.com/enterprise-contract/ec-policies//example/data"]
        }
    ]
}' ...
```

2. （可选）排除企业合同策略中规则的特定软件包，如下例所示：

```
exclude:
- attestation_task_bundle
- slsa_build_scripted_build
```

这个命令包括每个软件包中的每个规则，但指定软件包中的规则除外。

3. （可选）排除单个规则，如下例所示：

```
exclude:
- attestation_task_bundle.unacceptable_task_bundle
```

此命令包含 **attestation_task_bundle** 软件包中的每个规则，但 **unacceptable_task_bundle** 规则除外。

4. （可选）只包括特定软件包中的规则，如下例所示：

```
include:
- test
- java
```

此命令仅包含指定软件包中的规则。

5. （可选）只包括特定软件包中的一些规则。这意味着，您可以指定 **include** 和 **exclude** 来只选择您希望 Enterprise Contract 策略包含的规则，如下例所示：

```
include:
- "*" (1)
- attestation_task_bundle.unacceptable_task_bundle
exclude:
- attestation_task_bundle.*
```

(1) 星号 packagemanifests 作为通配符来匹配任何软件包。请注意，它与部分名称不匹配，这意味着您无法指定"s"与以"s"开头的每个软件包匹配。

这些命令指定 您只 包括来自 **attestation_task_bundle** 软件包中的 **unacceptable_task_bundle** 规则，并排除该软件包中的所有其他规则。

6. （可选）排除某些检查，以便 Enterprise Contract 即使这些检查失败或没有完成，也可以验证您的容器镜像，如下例所示：

```
exclude:
- test:get-clair-scan
- test:clamav-scan
```

此命令指定，如果任何确定的检查失败或未完成，企业合同仍然可以完成以验证您的容器镜像。

7. （可选）通过运行 config.policy. **include 命令或 config.policy.exclude** 命令以及字符串列表来修改软件包中规则的默认值。
   您的字符串列表应该包括以下之一：

   - "package name" - 从 Available rule collections 列表中的软件包中选择。
   - "rule name" - 输入软件包名称和规则代码的名称（以点(.)分隔）指定一个规则名称，如本例中所示： **attestation_type.unknown_att_type**。您可以在此处的"Attestation type"中找到规则代码。https://enterprisecontract.dev/docs/ec-policies/release_policy.html#attestation_type_package
   - "package name:term" - 一些策略规则处理项目列表。将 "term" 添加到 "package name" 字符串时，您可以从该列表中排除或包含特定项。这的工作方式与"软件包名称"类似，但它只适用于与该术语匹配的软件包中的策略规则。例如，如果您运行 test 软件包，您可以选择忽略给定的测试案例，但包含所有其他测试。
   - "rule name:term" - 这与 "package name:term" 类似，除了 包含软件包 中的项目外，您可以包含 ot 排除特定的软件包 策略规则。
   - "@collection name" - 将其添加到您的字符串中以指定预定义的规则集合。请确保使用 @ 符号为集合名称添加前缀。从 此处的 可用规则集合中选择。

其他资源

有关发行策略详情的完整列表，请参阅 发布策略。

第 4 章 签名容器镜像

先决条件

- 访问 OpenShift Web 控制台。
- 在 OpenShift 版本 4.13 或更高版本上运行的 Red Hat Trusted Artifact Signer 安装。
- 安装了 **ec**、**cosign** 和 **oc** 二进制文件的工作站。

流程

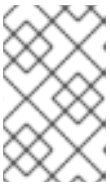
1. 登录到您的 OpenShift 集群：

   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

   若要查找您的命令行登录令牌和 URL，请登录 OpenShift Web 控制台。单击您的用户名，然后单击 **Copy login 命令**。如果出现提示，请再次输入您的用户名和密码，然后单击 **Display Token**。

2. 登录到 Trusted Artifact Signer。确保配置您的 Trusted Artifact Signer shell 环境以签名和验证容器镜像；例如：

   ```
   cd sigstore-ocp
   source tas-env-variables.sh
   ```

   您还可以选择手动设置环境变量；例如：

   ```
   export OPENSHIFT_APPS_SUBDOMAIN=apps.$(oc get dns cluster -o jsonpath='{
   .spec.baseDomain }')
   export OIDC_AUTHENTICATION_REALM=sigstore
   export FULCIO_URL=https://fulcio.$OPENSHIFT_APPS_SUBDOMAIN
   export OIDC_ISSUER_URL=https://keycloak-keycloak-
   system.$OPENSHIFT_APPS_SUBDOMAIN/auth/realms/$OIDC_AUTHENTICATION_REALM

   export REKOR_URL=https://rekor.$OPENSHIFT_APPS_SUBDOMAIN
   export TUF_URL=https://tuf.$OPENSHIFT_APPS_SUBDOMAIN
   ```

   Example

   ```
   $ source ./tas-env-vars.sh
   ```

3. 运行以下命令，从 OpenShift 集群注销：**oc logout**。

4. 找到您要签名和 attest 的容器镜像；例如：**IMAGE=quay.io/lucarval/rhtas-test@sha256:6b95efc134c2af3d45472c0a2f88e6085433df058cc210abb2bb061ac4d74359**。

5. 指明您要使用 Trusted Artifact Signer 而不是公共 Sigstore 部署签名和测试您的容器镜像的 RHTAP。输入这个命令：co **sign initialize --mirror=$TUF_URL --root=$TUF_URL/root.json**.

6. 使用以下命令为容器镜像签名：

```
cosign sign -y --fulcio-url=$FULCIO_URL --rekor-url=$REKOR_URL \
    --oidc-issuer=$OIDC_ISSUER_URL $IMAGE
```

7. 出现提示时，登录到在安装 Trusted Artifact Signer 时安装的 RHTAP 实例的 Keycloak 实例。因此，Keycloak 可以为您进行身份验证。

后续步骤

您的镜像现已签名。现在，您可以：

1. 在测试时创建 SLSA 验证，并将它与容器镜像相关联。

2. 使用 Red Hat Enterprise Contract 来验证容器镜像。

其他资源

- 有关 Red Hat Trusted Artifact Signer 的更多信息，特别是它如何与 RHTAP 和企业合同一起使用，请参阅 Red Hat Trusted Artifact Signer Deployment 。
- 有关 Keycloak 的更多信息，请参阅 Keycloak.org。

4.1. 生成签名密钥以签名和测试容器镜像

您必须有一个签名密钥，然后才能签署并测试容器镜像。

先决条件

- 安装了 **cosign** 二进制文件的工作站。

流程

1. 在 CLI 中，运行这个命令：cosign **generate-key-pair**。

2. 出现提示时，为密钥对输入新密码。确保您的密码是 memorable and strong。

验证

- 现在，您的工作目录中应当有两个新文件：一个 **cosign.pub** 文件和 **cosign.key** 文件。
  - **cosign.pub** 文件包含您的公钥。您可以与需要验证容器镜像的任何协作者共享此密钥。
  - **cosign.key** 文件是用于签名内容的私钥。只有负责签名和测试镜像的人员应有权访问 **cosign.key** 文件。

4.2. 使用 ENTERPRISE CONTRACT 和 TRUSTED ARTIFACT SIGNER 验证容器镜像签名

安装 Red Hat Trusted Artifact Signer service RHTAS 时，您可以使用 **ec** 二进制文件来验证使用 RHTAS 服务无密钥签名框架的容器镜像的验证和签名。有关安装 RHTAS 的更多信息，请参阅使用 Operator Lifecycle Manager 安装红帽受信任的工件 Signer。

先决条件

- 在 OpenShift 版本 4.13 或更高版本上运行的 RHTAS 安装。
- 访问 OpenShift Web 控制台。
- 安装了 **cosign** 和 **oc** 二进制文件的工作站。

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，单击右上角的？图标，然后选择 **Command Line Tools**。
   b. 在 **ec download** 部分中，点您的平台的链接。
   c. 打开一个终端，解压缩 **.gz** 文件，并在 **ec** 二进制文件上设置执行位：

      Linux 和 macOS 示例

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

   d. 将 **ec** 二进制文件移到 **$PATH** 环境中的目录中：

      Example

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

提示

运行 **ec validate image --help** 命令来查看所有 image 验证命令选项。

2. 为容器镜像签名和验证配置 shell 环境。
   a. 打开一个终端，从 **sigstore-ocp** 目录运行 **tas-env-variables.sh** 脚本：

      Example

      ```
      cd sigstore-ocp
      source tas-env-variables.sh
      ```

   b. （可选）手动设置环境变量：

      Example

      ```
      export OPENSHIFT_APPS_SUBDOMAIN=apps.$(oc get dns cluster -o jsonpath='{
      .spec.baseDomain }')
      export OIDC_AUTHENTICATION_REALM=sigstore
      ```

```
export FULCIO_URL=https://fulcio.$OPENSHIFT_APPS_SUBDOMAIN
export OIDC_ISSUER_URL=https://keycloak-keycloak-system.$OPENSHIFT_APPS_SUBDOMAIN/auth/realms/$OIDC_AUTHENTICATION_REALM
export REKOR_URL=https://rekor.$OPENSHIFT_APPS_SUBDOMAIN
export TUF_URL=https://tuf.$OPENSHIFT_APPS_SUBDOMAIN
```

Example

$ source ./tas-env-vars.sh

3. *初始化更新框架(TUF)系统：*

Example

cosign initialize --mirror=$TUF_URL --root=$TUF_URL/root.json

4. *为容器镜像签名：*

语法

Cosign sign -y --fulcio-url=$FULCIO_URL --rekor-url=$REKOR_URL --oidc-issuer=$OIDC_ISSUER_URL IMAGE_NAME

Example

cosign sign -y --fulcio-url=$FULCIO_URL --rekor-url=$REKOR_URL --oidc-issuer=$OIDC_ISSUER_URL example-hello-world@sha256:2788a47fd0ef1ece30898c1e608050ea71036d3329b9772dbb3d1f693f745c

在打开的 Web 浏览器中，使用电子邮件地址为容器镜像签名。

5. *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": []
}
```

6. *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 example.io/hello-world:latest
```

7. *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 example.io/hello-world:latest
   Supply Chain Security Related artifacts for an image: example.io/hello-
world@sha256:7de5fa822a9d1e507c36565ee0cf50c08faa64505461c844a3ce3944d23efa35
└──    Attestations for an image tag: example.io/hello-world:sha256-
7de5fa822a9d1e507c36565ee0cf50c08faa64505461c844a3ce3944d23efa35.att
   └──
sha256:40d94d96a6d3ab3d94b429881e1b470ae9a3cac55a3ec874051bdecd9da06c2e
└──    Signatures for an image tag: example.io/hello-world:sha256-
7de5fa822a9d1e507c36565ee0cf50c08faa64505461c844a3ce3944d23efa35.sig
   └──
sha256:f32171250715d4538aec33adc40fac2343f5092631d4fc2457e2116a489387b7
```

8. *使用 Enterprise Contract 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 example.io/hello-world:latest \
--certificate-identity 'jdoe@example.com' \
--certificate-oidc-issuer '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，其中包含任何安全违反情况的详细信息。添加 **--info** 标志时，报告包括更多详情和可能的解决方案。

其他资源

- 有关 RHTAS 的更多信息，请参阅 Red Hat Trusted Artifact Signer 的产品文档 。
- 有关 TUF 的更多信息，请参阅更新框架。
- 有关使用 Cosign 签名和验证容器镜像的更多信息，请参阅 ADD LATER。
- 有关 SLSA 认可 predicate 规格的更多信息，请参阅 SLSA Provenance。

第 5 章 ATTESTING AND VERIFY A CONTAINER IMAGE

在企业合同可以验证您的已签名的容器镜像之前，您必须首先创建 SLSA 认可，并将其与您的容器镜像相关联。经过验证是有关软件工件的可验证信息，包括在哪里、时间和提供链中给定软件"链接"的方式。有关 Software Artifacts (SLSA)验证的 Supply-chain 级别的更多信息，请参阅 SLSA Provenance。

先决条件

- 签名的容器镜像。
- 访问 OpenShift Web 控制台。
- 在 OpenShift 版本 4.13 或更高版本上运行的 Red Hat Trusted Artifact Signer 安装。
- 安装了 **cosign** 和 **oc** 二进制文件的工作站。

流程

1. 创建 SLSA provenance **predicate.json** 文件，例如：

```
echo '{
 "builder": {
   "id": "https://localhost/dummy-id"
 },
 "buildType": "https://localhost/dummy-type",
 "invocation": {},
 "buildConfig": {},
 "metadata": {
   "buildStartedOn": "2023-09-25T16:26:44Z",
   "buildFinishedOn": "2023-09-25T16:28:59Z",
   "completeness": {
     "parameters": false,
     "environment": false,
     "materials": false
   },
   "reproducible": false
 },
 "materials": []
}
' > predicate.json
```

2. 签名并测试您刚才创建的 **predicate.json** 文件，例如：

```
cosign attest -y --fulcio-url=$FULCIO_URL \
  --rekor-url=$REKOR_URL \
  --oidc-issuer=$OIDC_ISSUER_URL \
  --predicate predicate.json \
  --type slsaprovenance $IMAGE
```

Keycloak 将打开，以便在签署容器镜像时基于登录自动进行身份验证。

3. 使用企业合同验证签名和测试，例如：

```
ec validate image --image $IMAGE \
  --certificate-identity-regexp '.*' \
```

```
--certificate-oidc-issuer-regexp '.*' \
--output yaml --show-successes
```

重要

运行 **ec validate image** 命令时尽量具体，以便每个签名与预期 **的身份** 匹配。

验证

- 当企业合同验证容器镜像后，会打开所有企业合同验证和签名的详细报告。

其他资源

- 有关 Red Hat Trusted Artifact Signer 的更多信息，特别是它如何与 RHTAP 和企业合同一起使用，请参阅 Red Hat Trusted Artifact Signer Deployment 。
- 有关 SLSA 的更多信息，请参阅 slsa.dev。
- 有关 Keycloak 的更多信息，请参阅 keycloak.org。

5.1. 验证 JSON 和 YAML 定义

当您考虑保护软件供应链的所有选项时，您可以选择将策略应用到任务或管道定义。例如，您可能希望使用企业合同来检查给定管道中执行哪些任务。您可能想要使一些管道任务强制执行，只允许执行某些任务，或者在任务和管道运行前强制执行的任何其他约定。

先决条件

- 您已安装了 Enterprise Contract CLI。有关安装说明，请参阅 安装企业合同 CLI 指南。您可以在 **ec-cli** GitHub 仓库 中找到所需的文件。

流程

要验证企业合同 JSON 或 YAML 定义是否有效，请执行以下步骤：

- 在 Enterprise Contract CLI 中，输入 **ec validate input** 命令。