Red Hat Trusted Application Pipeline 1.0

红帽受信任的应用程序 Pipeline 入门

探索随时可用的软件模板，用于构建位于安全供应链功能的应用程序，如签名、测试、软件 Bill of Materials (SBOM)、SLSA 验证、CVE 扫描和发布策略保护rails。

Last Updated: 2024-07-02

Red Hat Trusted Application Pipeline 1.0 红帽受信任的应用程序 Pipeline 入门

探索随时可用的软件模板，用于构建位于安全供应链功能的应用程序，如签名、测试、软件 Bill of Materials (SBOM)、SLSA 验证、CVE 扫描和发布策略保护rails。

法律通告

摘要

本文档提供了有关使用现成的软件模板来构建带有安全供应链功能的应用程序，如签名、测试、软件 Bill of Materials (SBOM)、SLSA 验证、CVE 扫描和发布策略保护rails。

目录

第 1 章 了解 RHTAP 的基础

发现红帽受信任的应用程序管道(RHTAP)的强大基础，旨在在软件开发生命周期(SDLC)过程中改动网络安全实践。通过 RHTAP，您将开始一个转换传统安全措施的旅程，将尖端解决方案和 DevSecOps CI/CD 框架集成至部署。这一主动策略可加快开发人员加入、流程加速以及从头嵌入安全性。

1.1. 安全 CI/CD 框架

RHTAP 的核心是其安全 CI/CD 框架，旨在在软件开发中占据最高标准。通过与 Software Artifacts (SLSA)级别 3 的 Supply-chain 级别 保持一致，RHTAP 可确保每行代码都为安全提供贡献，从而大大提高早期漏洞检测和缓解措施。

1.2. 深入了解 RHTAP 的安全工具

确保软件在开发过程中的安全对于缓解潜在漏洞至关重要。RHTAP 利用了设计用于增强您的安全措施的强大工具套件。让我们探索 RHTAP 如何使用其组件 - RHDH、RHTAS 和 RHTPA - 以提供可靠的安全威胁。

Red Hat Developer Hub (RHDH)

- Red Hat Developer Hub 充当开发人员的自助服务门户。它简化了加入过程，并提供对保护软件开发所需的各种资源和工具的访问。此平台鼓励最佳实践，并有助于集成开发流程开始的安全措施。

Red Hat Trusted Artifact Signer (RHTAS)

- Red Hat Trusted Artifact Signer 侧重于通过签名和测试机制提高软件完整性。通过确保每个代码和每个工件都经过签名和测试时，RHTAS 提供可验证的信任链，以确认所使用的软件组件的真实性和安全性。

Red Hat Trusted Profile Analyzer (RHTPA)

- Red Hat Trusted Profile Analyzer，负责软件 Bills 的生成和管理(SBOM)。SBOM 对于保持透明度和合规性至关重要，因为它们提供了软件产品中包含的所有组件、库和依赖项的详细列表。RHTPA 自动创建 SBOMs，确保利益相关者对软件组成有准确且最新的信息。

1.3. 利用随时可用的软件模板

RHTAP 提供随时可用的软件模板，将安全性直接嵌入到开发工作流中，从而使开发人员能够集中精力进行创新，同时最大程度降低与安全性相关的距离。这些可随时可用的软件模板可完全自定义，确保它们能够无缝地满足您组织的唯一要求。

从右侧的集成功能中受益：

- **Red Hat Advanced Cluster Security (RHACS)：**针对漏洞部署准备。
- **Quay：**为您的容器镜像提供安全存储库。
- **Tekton pipelines ：**在自动化部署中启用精度。
- **GitOps：**保持一致性和自动配置管理。

1.4. 主要安全实践

RHTAP 融合了这些工具以有效地解决特定的安全顾虑：

- **漏洞扫描：** 通过每个拉取请求，RHTAP 会对您选择的 CVE 扫描程序进行全面的扫描，如 Advanced Cluster Security，以在最早的阶段识别和解决漏洞。
- **SBOM Generation：** RHTAP 的自动化生成 SBOMs 在维护软件透明度和合规性方面扮演着重要角色。通过提供软件组件的全面清单，组织可以更好地管理和保护其软件供应链。
- **容器镜像安全：** RHTAP 验证容器镜像是否符合 SLSA （软件工件的 Supply-chain 级别） 指南。这可以通过包含 41 规则的企业合同来实现，确保开发过程中使用的容器镜像满足严格的安全标准。

1.5. 路径转发

使用 DevSecOps 考虑集并使用 RHTAP 促进了安全高效的开发环境。这种持续的评估过程和提升使组织能够有效解决当前和未来的网络安全挑战。

后续步骤

- 保护应用程序开发的路径

其他资源

- 有关 Red Hat Developer Hub 的详情，请参考 开始使用 Red Hat Developer Hub 指南 。
- 有关 Red Hat Trusted Artifact Signer 的详情，请参考 RHTAS 部署指南。
- 有关红帽受信任的配置文件分析器的详情，请参考 快速入门指南。

第 2 章 保护应用程序开发的路径

红帽受信任的应用程序管道(RHTAP)显著提高容器化和部署应用程序的效率，使开发人员能够在几分钟内部署其工作。这种创新平台不仅有助于创建构建管道，以快速测试和集成应用程序更改，还用于识别针对供应链接攻击的安全措施。通过遵循软件工件(SLSA)安全框架的严格标准 Supply-chain 级别，RHTAP 可确保符合高级别的安全要求。

2.1. 安装概述

在利用 RHTAP 提供的大量好处之前，初始步骤涉及其在组织内安装。RHTAP 的安装围绕 7 个关键步骤构建：

1. 为 RHTAP 创建 GitHub 应用程序
2. 分叉模板目录
3. 创建 GitOps git 令牌
4. 创建 Docker 配置值
5. 创建 private-values.yaml 文件
6. 在集群中安装 RHTAP
7. 最终调整 GitHub 应用程序

2.2. 初始设置

在开始安装过程前，必须满足某些先决条件以确保平稳的设置和成功设置：

1. **Cluster Access**: 确保 ClusterAdmin 访问 OpenShift Container Platform (OCP)集群，可通过 CLI 和 Web 控制台访问。
2. **Red Hat Advanced Cluster Security (ACS)** ：获取 ACS 实例所需的值，包括：
   - ACS API 令牌：按照 此处 提供的说明创建 API 令牌。
   - ACS 中央端点 URL：通过引用 此处 提供的说明来配置端点。
3. 为**私有存储库配置 ACS** ：如果您在镜像 registry 中使用私有存储库，如 Quay.io，请相应地配置 ACS：
   - 对于 Quay.io，进入到 Integrations > Image Integrations 并选择 Quay.io 卡。
   - 添加您的 OAuth 令牌以访问您的特定 Quay.io 实例。
   - 通过测试按钮验证访问，以确保 ACS 能够在需要时扫描私有镜像。
4. **Quay.io 帐户** ：确保您有一个活跃的 Quay.io 帐户。
5. **Helm CLI 工具** ：根据 此处 提供的指南安装 Helm CLI 工具。
6. **GitHub 帐户** ：最后，确保您有一个 GitHub 帐户来方便某些安装过程。

完成这些先决条件后，您已准备好通过创建一个新的 GitHub 应用程序来启动安装过程，方法是为您的 RHTAP 实例创建专门用于 RHTAP 实例的新 GitHub 应用程序。

后续步骤

- 安装红帽受信任的应用程序 Pipeline

第 3 章 使用示例软件模板构建应用程序

RHTAP 将开发环境与 Red Hat Developer Hub (RHDH)中提供的可用软件模板转换。这些模板经过严格设计，旨在与红帽全面的工具套件(RHDH、RHTTAS、RHTPA)和技术进行无缝集成。这种集成促进了用于内部环境中安全、高效且面向开发人员的 SDLC 的固态框架。

除了这些基础元素外，RHTAP 的现成软件模板包括默认与关键技术集成，以进一步保护和优化您的开发体验：

- **ACS （高级集群安全性）：** 通过识别和减轻开发过程中早期的漏洞，确保您的应用程序从不便部署到部署，从而增强您的部署。
- **Quay ：** 作为容器镜像的安全隐患，提供可靠的存储库来持续扫描漏洞，使容器化应用程序保持安全。
- **Tekton Pipelines：** 使用精度自动化构建和部署流程，启用 CI/CD 框架，该框架可无缝集成到 SDLC 中，从而加快您的生产路径。
- **GitOps：** 通过在 Git 存储库中维护基础架构和应用程序配置来实现 GitOps 策略，确保所有环境中一致和自动化部署。

此外，RHTAP 支持跨各种流行编程语言（如 Java、Python、Node.js 和 Go）进行应用程序的开发和容器化，扩展您的应用程序开发功能。

安装 RHTAP 后，集群管理员可以使用特定模板和增强定制 Red Hat Developer Hub 门户。这种自定义过程对于使开发人员能够专注于代码至关重要，简化开发工作流并减少与管道、漏洞和策略相关的关注。

在继续自定义前，集群管理员需要熟悉可用的软件和管道模板。这种探索是获取 RHTAP 如何支持安全供应链的关键，为后续自定义打下基础。

3.1. 设置阶段

- 确保您已成功安装了 RHTAP。
- 使用 RHTAP 提供的链接，登录 Red Hat Developer Hub (RHDH)。RHDH 作为附带的开发人员平台运行，促进开发人员门户的创建。它为工程团队提供了一个统一平台，增强了开发流程，为高效制作高质量软件提供工具和资源。

3.2. 构建应用程序

在 RHDH 门户上，选择 **Create**，然后选择适当的模板。例如，Quarkus Java - Trusted Application Pipeline。

使用 RHTAP 提供的模板，在 RHDH 中为开发人员构建应用程序或微服务，基本上只是三个步骤过程：

- 提供应用程序信息
- 提供应用程序存储库信息
- 提供部署信息

提供应用程序信息

1. 在 **Name** 字段中，提供应用程序名称。您的名称可以包含小写字母(a-z)、数字(0-9)和短划线(-)，但它必须以小写字母数字字符开头和结尾。有效名称的示例为 **my-name** 或 **abc-123**，长度应范围为 1 到 63 个字符。

2. 从 **所有者** 下拉列表中，为此应用程序选择适当的 RHDH 组件所有者。默认值为 **user:guest**，如果系统中没有注册任何特定所有者，则会出现。如果您还没有注册所有者，请保留默认的 **user:guest** 选择。如果需要，您可以选择将 **guest** 替换为您的用户名，并个性化应用程序的所有权。

3. 选择 **Next**。系统显示 Application Repository Information 表单。

提供应用程序存储库信息

1. 从 **主机类型** 下拉列表中，选择适当的存储库主机类型。

2. 在 **Repository Owner** 字段中，输入拥有您要使用的 Git App 的机构名称。这可以是您所在机构的个人用户帐户、机构或项目。

3. 在 **Repository Name** 字段中，使用仅限于 A-Z、a-z、0-9、下划线(_)和短划线(-)的字符输入适当的存储库名称。系统使用此信息来命名它在主机存储库服务器上创建的存储库。

4. 在 **Repository Default Branch** 字段中，输入存储库的默认分支。默认情况下，此字段显示 **main**，您可以选择使其保持相同。

5. 在 **Repository Server** 字段中，输入没有 **HTTP** 协议且没有 **.git** 扩展名的 on-prem 主机 URL。例如，gitlab-gitlab.apps.cluster-ljg9z.sandbox219.opentlc.com

6. 选择 **Next**。系统显示 Deployment Information 表单。

提供部署信息

1. 在 **Image Registry** 字段中，输入没有 **HTTP** 协议的 on-prem 镜像 registry URL。例如： quay-tv2pb.apps.cluster-tv2pb.sandbox1194.opentlc.com。

2. 在 **Image Organization** 字段中，为在第 1 步中提供的镜像 registry 输入镜像机构。

3. 在 **Image Name** 字段中，按照以下准则输入合适的镜像名称：只使用小写字母、dights 和 separators。分隔符包含一个句点(.)、最多两个下划线(_)或一个或多个连字符(-)。例如，**my-app_1.2**。

注意

您必须确保名称不以分隔符启动或结尾。

4. 在 **Deployment Namespace** 字段中输入您要部署应用程序的命名空间或集群的前缀。系统将实际命名空间创建为 **rhtap-app-development**、**rhtap-app-stage** 和 **rhtap-app-prod**。

注意

rhtap-app 是默认的部署命名空间前缀。集群管理员可以选择自定义此前缀。有关如何自定义默认部署命名空间前缀的说明，请参阅自定义示例软件模板。

5. 选择 **Review** 来查看您添加的所有信息。

6. 选择 **Create**。RHTAP 启动一系列自动化任务，用于设置应用程序的基础架构和部署管道。这个过程涉及一些卸载后的关键操作：

   - **存储库创建和配置：** 在指定的托管服务(GitLab 或 GitHub)中自动创建新的存储库，专门用于您的应用程序。这包括包含部署配置的 GitOps 存储库的设置，以及应用程序代码所在的源存储库。

- **Argo CD 集成 ：**使用存储库时，会创建并配置 Argo CD 资源。Argo CD （声明性、Git 持续交付工具 springs）用于操作，在指定命名空间中编排应用程序的部署。
- 命名空间 **创建 ：**作为设置部署环境的一部分，各种命名空间会根据应用程序的要求自动生成。这包括用于开发、暂存和生产环境的独立命名空间，确保在阶段之间隔离和安全性。
- **Pipeline 定义：**最后，管道定义添加到设置中，为您提供 'Pipeline as code' 模型。这定义了构建、测试和部署应用程序的自动化工作流，并与最佳实践和安全措施保持一致。

3.3. 检查您的应用程序

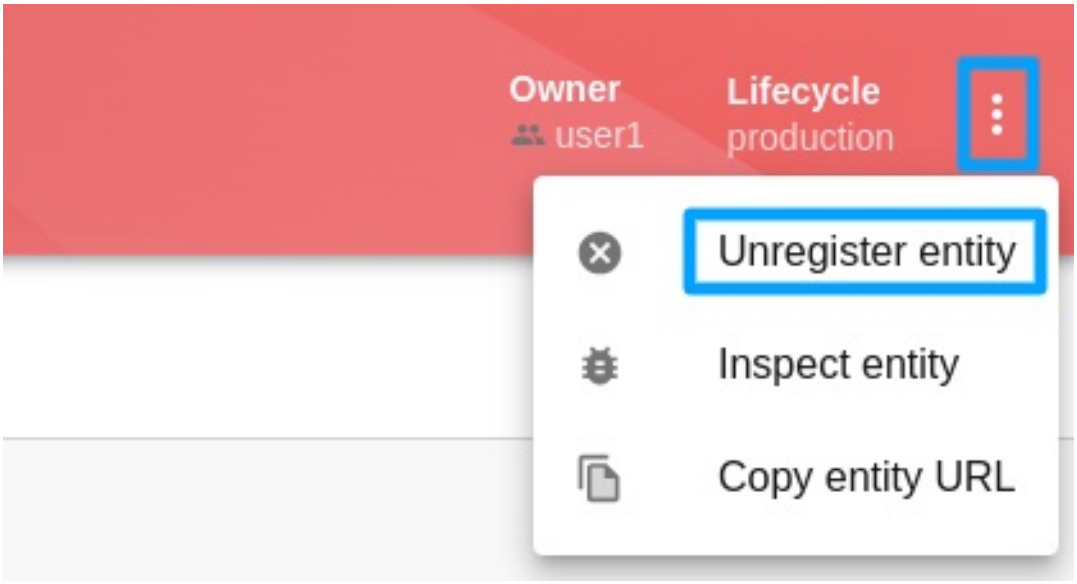
使用 RHTAP 成功创建了应用程序后，检查其组件、源代码、GitOps 配置和相关文档非常简单。以下是如何进行全面的分析：

- **访问应用程序：**
  - 要查找应用程序，请在目录 中选择 **Open 组件**。您还可以导航到系统列出新创建的应用程序的 **Catalog**。
- **检查源代码：**
  1. 转至 **Overview** 选项卡。
  2. 选择 **View Source** 以打开存储应用程序源代码的存储库。此步骤可让您了解应用程序的构造和逻辑。
- **查看 GitOps 存储库：**
  1. 在 **Overview** 选项卡中，使用 **Kind** 下拉菜单选择 **Resource** 并查找相关的 GitOps 存储库。
  2. 选择 **View Source** 来直接检查 GitOps 配置。另外，对于包括技术文档的更广泛概述，请从 **Catalog** 部分选择 **View TechDocs**，然后在 **Home** > **Repository** 部分下选择 GitOps 存储库。
- **查看文档：**
  1. 在 **Overview** 选项卡中，单击 **View Tech Docs**。
  2. 这会为您的应用程序打开技术文档，深入了解其功能、配置步骤以及如何有效地使用它。

3.4. 取消注册应用程序

这个过程从目录和资源视图中删除应用程序的源和 GitOps 存储库，基本上是隐藏它。应用程序本身在集群中保持运行。未注册的应用程序可以随时重新注册。

1. 导航到 **Catalog**，再选择您要取消注册的组件。
2. 选择与组件关联的垂直三dot 菜单，然后选择 **Unregister entity**。此时会出现确认对话框。



3. 选择 **Unregister Location**。这会从您的目录视图中删除应用程序的 Git 存储库。
4. 从 **Kind** 下拉列表中选择 **Catalog**，选择 **Resource**，然后取消注册对应的 GitOps 资源。
5. 运行以下命令从集群中删除应用程序：

```
oc delete application your-app-name-app-of-apps -n rhtap (1)
```

(1) **rhtap** 是 default 命名空间。另外， **your-app-name** 是应用程序的名称。

后续步骤

- 更新代码并查看安全分析

第 4 章 更新代码并查看安全分析

使用 RHTAP 成功构建组件后，下一步是进行代码更改并转至安全见解。

4.1. 启动代码更新

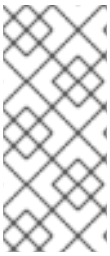
使用 RHTAP 时，这个过程非常简单。

1. 选择 **Catalog**，然后选择您要查看安全见解的适当组件。
2. 在 **Overview** 选项卡中，选择 **View Source** 选项卡来查看 GitLab 或 GitHub 中的项目。在这里，您还可以选择 **View Tech Docs** 来查看项目的文档。文档的来源是您的 **存储库中的** 文档目录。如果您更新这些文件，并且管道成功运行，您的技术文档也会更新。

4.2. 更改代码

通过访问存储库，您可以参与使用 git 存储库的熟悉过程。以下是如何进行：

- **克隆** 您的存储库，以便在本地或开发环境中使用存储库。
- **修改** 应用程序，如更新技术文档或 **index.html**，或添加新的功能或 bug 修复。
- **提交您的更改**。
- **将** 您的更改推送到存储库。



注意

- 您还可以使用 **GitLab 或 GitHub** UI 在 web 界面中直接更新您的代码。
- **仅用于 GitLab 用户：** 您必须在 GitLab 中设置 webhook 和 secret，以便在代码更新后自动触发管道运行。有关在 GitLab 中设置 webhook 和 secret 的详情，请参考为自动管道配置 GitLab Webhook。

4.3. 查看管道运行

在对代码进行更改后查看管道运行：

1. 返回到 RHDH 平台，了解您的更改的进展方式。
2. 导航到 **Catalog**，再选择您刚才修改的特定组件。
3. 打开 **CI** 选项卡，以访问管道运行。

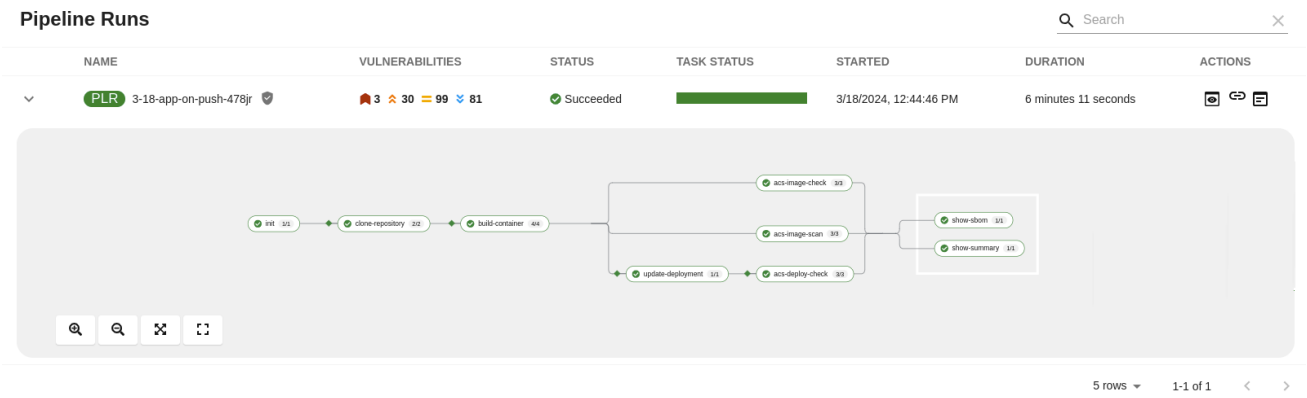
除了管道运行外，RHDH 通过其他标签页提供宝贵见解：

- cd : 深入了解使用 **CD** 选项卡由 ArgoCD 和 GitOps 管理的部署。
- 拓扑 ：使用 **Topology** 选项卡在 development 命名空间中视觉化应用程序部署。

4.4. 查看安全见解

当您更新代码并推送更改时，系统会自动触发 **on-push** 管道。默认情况下，RHTAP 使用标准构建管道快速、容器化部署，并通过满足 软件工件(SLSA)级别 3 规格来增强供应链安全性。

图 4.1. 成功管道运行



此可视化表示概述了每个管道任务。**绿色** 状态表示成功完成，简化您的工作流，而无需进行大量配置。

初始构建管道任务包括：

- **init** ：初始化管道，配置重建标志和身份验证，并为 PipelineRun 生成镜像存储库 secret。
- **clone-repository**: 将指定的存储库克隆到工作区，为使用 git-clone 任务操作做好准备。
- **build-container** ：此任务使用 Buildah 将源代码转换为容器镜像，然后推送到指定的 registry。此任务的主要操作和结果包括：
  - **容器镜像创建和部署：** Buildah 将源代码编译到容器镜像中。成功创建后，镜像将推送到指定的镜像 registry。
  - **软件 Bill of Materials (SBOM) Generation:**作为确保透明度和合规性的一部分，生成了 SBOM，详细说明容器镜像中包含的组件、库和依赖项。然后，此 SBOM 被嵌入到最终的容器镜像中，以便轻松访问和验证。
  - **SBOM Publishing ：** 除了将 SBOM 合并到容器镜像外，此任务使用 Cosign 将 SBOM 推送为独立镜像。这有助于通过安全性和合规团队更轻松地管理和验证 SBOM。
  - **工件创建：** 任务生成关键的安全工件，包括镜像签名(**.sig**)和 attestation (**.att**)。这些工件对于验证容器镜像及其内容的完整性和真实性至关重要，为部署管道中的信任和安全验证提供可靠的机制。
- **update-deployment** ：使用新构建的镜像更新部署环境。在这个版本中，在 **Overlay** > **Development** 目录的 GitOps 存储库中执行。
- **ACS** 任务：对代码和部署配置进行安全评估，确保遵守已建立的安全策略和最佳实践。
- **show-sbom** ：创建应用程序中使用的所有软件组件和库的完整列表，提高透明度和支持漏洞管理。
- **Summary** ：提供 PipelineRun 的摘要，包括 PipelineRun 信息，并删除 PipelineRun 使用的镜像存储库 secret。

注意

您可以点击管道运行中的任何任务来访问任务成功完成时生成的日志和其他详情，这通过 **绿色** 检查表示。

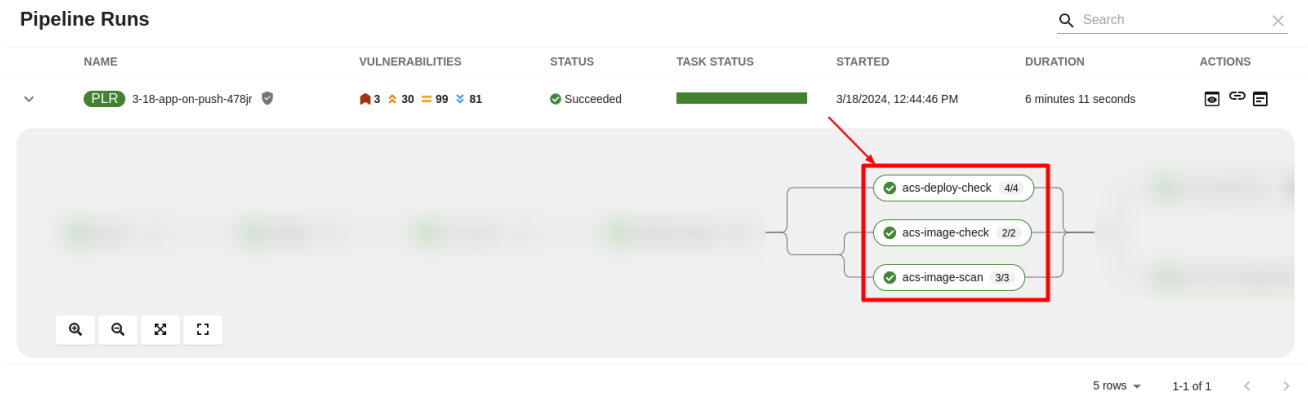
4.4.1. Red Hat Advanced Cluster Security 任务

RHTAP 在管道中利用 Red Hat Advanced Cluster Security (RHACS)及其安全检查。如果安装并配置 RHACS，管道会运行 RHACS 任务（例如 **roxctl 镜像扫描**），并在完成后显示绿色检查。但是，如果没有安装或升级 RHACS，管道会跳过 RHACS 任务。

注意

- 只有在您已经安装并配置 RHACS 作为 RHTAP 安装过程的一部分时，管道中的 RHACS 任务才会成功。有关安装 RHACS 的详细信息，请参阅安装 Red Hat Advanced Cluster Security for Kubernetes。
- 如果您没有在 RHTAP 安装过程中安装和配置 RHACS，请参阅 配置 ACS。

图 4.2. 管道运行中的 RHACS 任务



管道使用 **roxctl** 包含三个 RHACS 任务来执行全面的安全检查：

- **roxctl image scan** - 返回 JSON 格式镜像中发现的组件和漏洞。
- **roxctl image check** - 检查镜像中安全策略的构建时违反情况。例如，'No log4j allowed' 或者没有 curl，wim 或生产镜像中的软件包管理器。
- **roxctl deployment check** - 检查 YAML 部署文件中的安全策略的 build-time 和 deploy-time 违反情况。

这些任务可确保遵循开发阶段的安全策略和配置。

视觉化 RHACS 报告

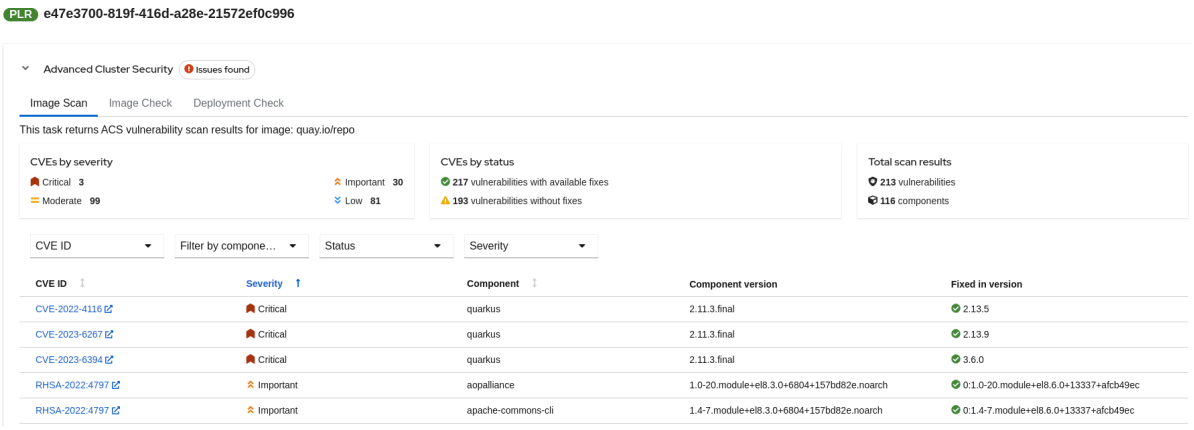
在 Red Hat Developer Hub 中，在 **CI** 选项卡下，Pipeline Runs 部分提供了通过结构化的弹出窗口访问和解释详细任务报告的功能。弹出由以下部分组成：

- **Red Hat Advanced Cluster Security （在 RHACS 任务可用性上显示条件）：**本节显示所有 RHACS 任务，如镜像扫描、镜像检查和部署检查，并提供安全问题的初始概述。
- **其他：** 本节显示 **PipelineRun** 的结果，如 IMAGE_URL 和 **IMAGE_DIGEST**。本节只有在弹出窗口中有多个部分（例如，企业合同或 {RHRHACSLongName}）时才会显示。

查看 RHACS 报告：

1. 选择 **Catalog** 并打开适当的组件，您要查看 RHACS 报告
2. 选择 **CI** tab > **Actions** 列 > **View output** 图标，并查看软件组件中详细的 RHACS 报告。

图 4.3. 详细的 RHACS 报告



注意

如果您有适当的权限，您可以通过导航到 RHACS 控制台来管理漏洞、策略和查看特定镜像的详细漏洞报告。如需更多信息，请参阅 查看仪表板。

解释 RHACS 报告

Red Hat Advanced Cluster Security (RHACS)任务生成的详细报告有助于提供安全见解，对于维护可靠的安全状况至关重要。

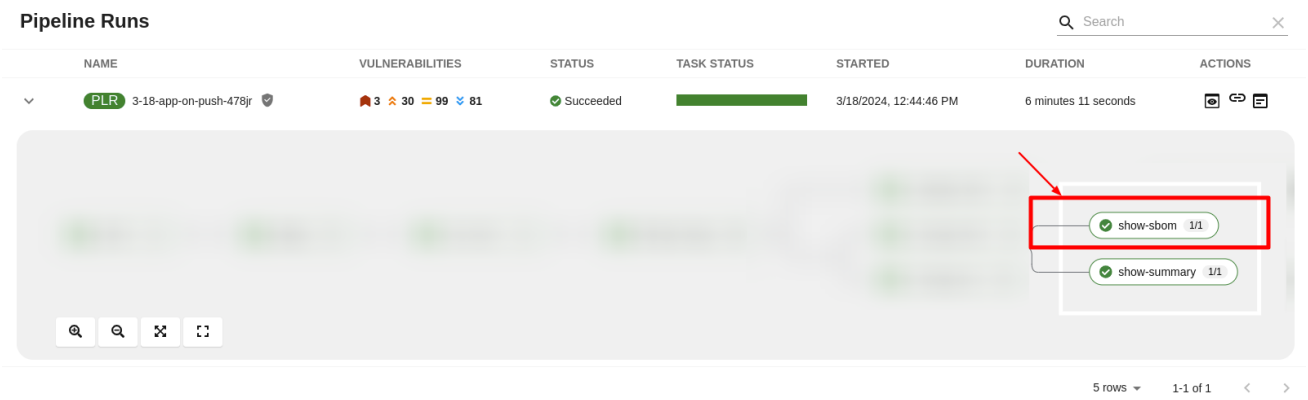
以下是如何解释 **roxctl 镜像扫描** (Image Scan)报告的示例。您可以应用类似的方法来解释 **roxctl 镜像检查** (Image Check)和 **roxctl 部署检查** (Deployment Check)的报告。

- **漏洞明细：** RHACS 按严重性对检测到的漏洞进行分类(Critical、重要、中等、低)、状态（可修复、不可修复），并提供扫描结果概述。此分类包括分析的漏洞和组件的总数，以及识别的特定常见漏洞和暴露(CVE)。
- **提供的详情：** 对于每个识别的漏洞，报告包括：
  - **CVE ID ：** 漏洞的唯一标识符。
  - **严重性 ：** 漏洞构成的威胁级别。
  - **组件 ：** 受这个漏洞影响的软件组件。
  - **组件版本：** 受影响组件的版本。
  - **补救 Suggestions:** 建议解决此漏洞，包括修复漏洞的版本（如果适用）。

4.4.2. 了解 SBOM

show-sbom 任务通过列出组件使用的所有软件库，帮助识别漏洞和安全影响，从而为供应链透明性贡献。

图 4.4. 管道运行中的 show-sbom 任务



查看 SBOM

流程

1. 选择 **Catalog** 并打开适当的组件，供您查看 SBOM。
2. 选择 **CI** 选项卡，然后选择链接图标。系统显示 SBOM 任务日志，您可以使用 Web 浏览器立即搜索 SBOM 的术语，以指明软件供应链中的漏洞。例如，尝试搜索 **log4j**。

图 4.5. SBOM 详情

在 CLI 中下载 SBOM

先决条件

- 已安装 Cosign CLI 工具。
- **build-container** 和 **show-sbom** 任务成功运行。

流程

1. 展开适当的成功管道运行，然后选择 **show-summary** 任务。
2. 查找并复制 SBOM 镜像 URL，并在终端中运行以下命令：

   cosign 命令示例

   ```
   $ cosign download sbom <the-sbom-url-you-copied>
   ```

   a. （可选）要以可搜索格式查看完整的 SBOM，请运行以下命令来重定向输出：

      cosign 命令示例

      ```
      $ cosign download sbom <the-sbom-url-you-copied> > sbom.txt
      ```

阅读 SBOM
在 SBOM 中，如以下示例摘录所示，您可以看到项目使用的每个库的四个特征：

- 其作者或发布者
- 其名称
- 其版本
- 其许可证

这些信息可帮助您验证单个库是否安全提供、更新和合规。

SBOM 示例

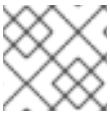
```
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.4",
  "serialNumber": "urn:uuid:89146fc4-342f-496b-9cc9-07a6a1554220",
  "version": 1,
  "metadata": {
    ...
  },
  "components": [
    {
      "bom-ref": "pkg:pypi/flask@2.1.0?package-id=d6ad7ed5aac04a8",
      "type": "library",
      "author": "Armin Ronacher <armin.ronacher@active-4.com>",
      "name": "Flask",
      "version": "2.1.0",
      "licenses": [
        {
          "license": {
            "id": "BSD-3-Clause"
          }
        }
      ],
      "cpe": "cpe:2.3:a:armin-ronacher:python-Flask:2.1.0:*:*:*:*:*:*:*",
      "purl": "pkg:pypi/Flask@2.1.0",
      "properties": [
        {
```

```
"name": "syft:package:foundBy",
"value": "python-package-cataloger"
...
```

第 5 章 部署应用程序并查看安全分析

组织利用结构化的应用程序部署方法，通常涉及开发、预生产和生产阶段。此过程通常是自动化和受定义的规则和触发器的管制。

本指南概述了在 OpenShift GitOps 中通过 ArgoCD 部署应用程序，从而在所有阶段都启用持续部署。ArgoCD 有助于基于 GitOps 的部署策略，将 Git 存储库作为基础架构配置的单一数据源进行读取。此存储库更新会在环境中触发部署。



注意

本指南演示了一个部署方法示例；机构可能会采用适合其工作流的任何方法。

5.1. 将构建提升到预生产环境或生产环境

将构建从一个环境提升到另一个环境（如从 development 到 stage 或 production）涉及通过拉取请求 (PR)更新 GitOps 存储库。

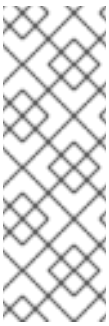
1. 在 RHDH 平台上，选择 **Catalog**。
2. 从 **Kind** 下拉列表中，选择 **Resource**，然后选择适当的 GitOps 存储库。
3. 在 Overview 选项卡中，选择 **View Source**。
4. （可选）选择 **Catalog**，然后在 **Overview** 选项卡中选择 **View TechDocs**
   a. 在 **Home** > **Repository** 部分中，选择 GitOps 存储库。
5. 克隆 GitOps 存储库，再前往 **component/<app-name>** 目录。

注意

确保本地克隆为最新版本。

6. 签出新分支。
7. 在存储库中，找到 **component/<app-name>/overlays** 目录，在其中找到 开发、**stage** 和 **prod** 子目录，各自对应于一个环境。
8. 手动将应用程序从开发环境移到阶段或生产环境。

移动应用程序	执行此操作

移动应用程序	执行此操作
从开发到暂存环境	1. 展开 **development** 目录，再选择 **deployment-patch.yaml**。 2. 复制容器镜像 URL。例如：quay.io/<username>/<app-name>/imageurl。 3. 进入 **stage** 目录，选择 **deployment-patch.yaml**，并将现有容器镜像 URL 替换为复制的目录。  注意 如果要提升其他配置更改（例如，副本）除了开发到阶段环境的容器镜像外，请复制 **development** 目录中的 **deployment-patch.yaml** 文件中的更改，并将它们粘贴到 **stage** 目录中的 **deployment-patch.yaml** 文件中。
从阶段到生产环境	1. 展开 **stage** 目录，再选择 **deployment-patch.yaml**。 2. 复制容器镜像 URL。例如：quay.io/<username>/<app-name>/imageurl。 3. 前往 **prod** 目录，选择 **deployment-patch.yaml**，并将现有的容器镜像 URL 替换为复制的镜像 URL。  注意 如果要提升其他配置更改（例如，副本）除了从阶段到生产环境的容器镜像外，请复制 **stage** 目录中的 **deployment-patch.yaml** 文件中的更改，并将它们粘贴到 **prod** 目录中的 **deployment-patch.yaml** 文件中。

9. 提交并推送您的更新。
10. 创建拉取请求(PR)。此操作将启动一个提升管道运行，用于根据 Red Hat Enterprise Contract (Enterprise Contract)策略验证更新的容器镜像。管道运行以可视化方式表示所有任务，**绿色** 状态表示成功完成。
    a. 查看 RHDH 中的 **CI** 选项卡中的提升管道。

11. 检查并合并 PR。合并 PR 触发器 ArgoCD，然后自动应用必要的更改，将构建提升到下一个环境。

    a. 查看 RHDH 中的 **CD** 选项卡中的最新部署更新。它显示应用的当前状态、部署详情、管道运行的作者、提交消息（例如，Promote stage 到 prod），以及容器镜像是否适用于生产环境。

验证

- 要评估应用程序成功提升，请进入 **Topology** 选项卡。您可以在指定命名空间中查看应用程序的发布。

5.2. 查看安全见解

promotion 管道运行提供了管道中所有任务的可视化表示。**绿色** 状态表示成功完成，简化您的工作流，而无需进行大量配置。

提升管道任务包括：

- **clone-repository**: 将指定的存储库克隆到工作区，为使用 git-clone 任务操作做好准备。
- **gather-deploy-images** ：根据 PR 中的更改来扫描容器镜像。
- **verify-enterprise-contract** ：验证已更改的容器镜像。此任务可确保镜像源自公司标准或已批准的构建系统。它利用企业合同(EC)策略与 Sigstore 的 cosign 工具一起评估镜像签名和测试的完整性。

注意

您可以点击管道运行中的任何任务来访问日志以及与该任务相关的附加详情。

5.2.1. 企业合同任务

企业合同是一系列工具，旨在维护软件供应链安全性。它允许定义和强制实施与容器镜像构建和测试相关的策略。

企业合同可确保红帽受信任的应用程序管道生成的容器镜像在将其发布到生产之前满足明确定义的要求。如果镜像无法满足这些条件，EC 会生成报告概述需要解决的特定问题。

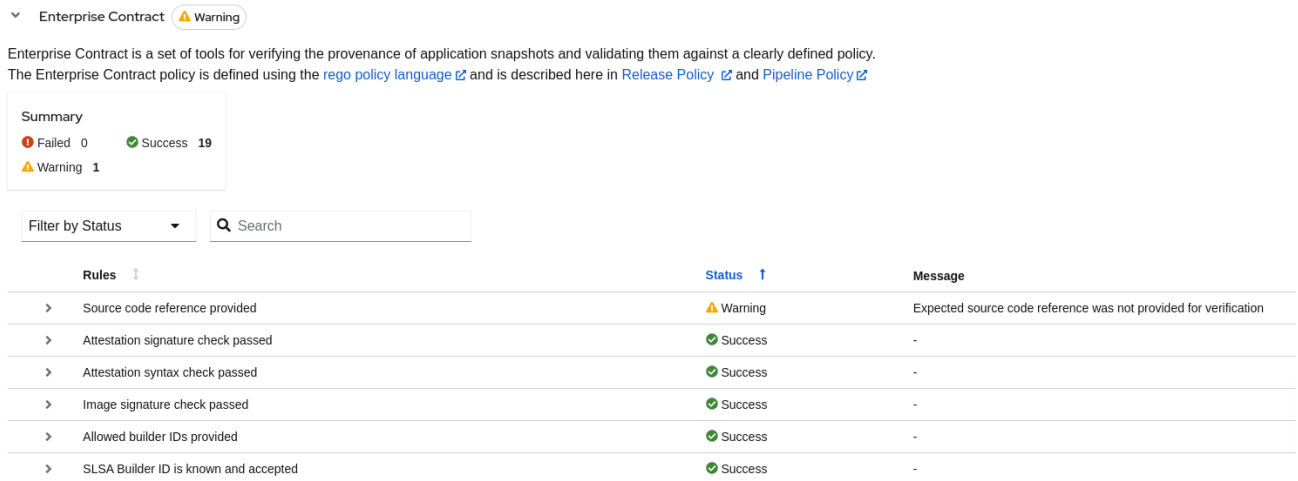
Red Hat Trusted Application Pipeline 构建过程使用 Tekton 链 来创建签名 到 构建管道的测试。然后，EC 利用此签名的测试来加密验证构建的完整性，并根据一组策略对其进行评估。这些策略可确保构建过程遵循规定的最佳实践和任何特定于组织的安全准则。

解释合规性报告
由企业合同(EC)扫描生成的详细报告是提供安全见解，对于维护可靠的安全状况至关重要。以下是如何解释这些报告：

- 策略合规 **概述：** EC 扫描评估应用程序是否符合软件工件(SLSA)安全框架的 Supply Chain Level。报告会列出执行的检查、每个检查的状态（成功、警告、失败），并为观察到的任何警告或失败提供消息。
- **提供的详情：** 策略报告详情：
  - **成功检查** ：满足策略规则的计数和细节。
  - **警告和失败** ：任何导致警告或失败的策略规则，以及解释原因的消息。

- **规则** 合规性：应用遵循各个策略规则的方式，如源代码引用和测试检查。

图 5.1. EC 报告



使用合规性报告中的见解

insights gleaned EC 扫描报告对于优先考虑安全和合规性工作至关重要：

- **查看策略合规性：** 通过密切审查 SLSA 和其他相关标准，确保应用程序的完整性。根据 EC 扫描建议解决任何合规性差距。
- **简化报告审核：** 员工在报告中提供过滤器，专注于重要的区域，促进了更高效的审查流程，并能够快速识别关键问题和合规性差距。

其他资源

- 有关 EC 策略和配置的更多信息，请参阅管理符合企业合同的合规性。

更新于 2024-07-02