105Kubernetes 系列（九十八）使用 OIDC 进行 Kubernetes 身份验证：简化身份管理

介绍

随着容器化不断彻底改变软件开发和部署领域，Kubernetes 已成为强大的容器编排平台。凭借其高效管理和扩展应用程序的能力，Kubernetes 获得了广泛的欢迎。然而，确保安全访问 Kubernetes 集群和验证用户身份仍然是一个严峻的挑战。这就是 OpenID Connect (OIDC) 发挥作用的地方。在这篇博文中，我们将探讨使用 OIDC 进行 Kubernetes 身份验证，以及它如何简化 Kubernetes 集群内的身份管理。

了解 OIDC

OpenID Connect (OIDC) 是一种构建在 OAuth 2.0 框架之上的开放标准身份验证协议。它提供了一种安全、灵活的方式来对应用程序和系统中的用户进行身份验证和授权。 OIDC 在 OAuth 2.0 中引入了与身份相关的功能，使其成为在 Kubernetes 集群中对用户进行身份验证的理想选择。

OIDC 对于 Kubernetes 身份验证的好处

使用 OIDC 进行 Kubernetes 身份验证具有多种优势，包括：

1. 集中身份管理：OIDC 允许您利用现有身份提供商 (IdP) 基础设施进行用户身份验证。这可以实现用户身份的集中管理，减少管理开销并确保跨多个应用程序和平台的一致性。

2. 单点登录 (SSO) 体验：通过将 Kubernetes 与 OIDC 提供商集成，用户可以享受无缝的单点登录体验。经过身份验证后，用户可以访问多个 Kubernetes 集群和应用程序，而无需重新输入凭据，从而简化了用户体验。

3. 增强的安全性：OIDC 利用行业标准安全机制（例如 JSON Web 令牌 (JWT)）来传输身份信息。这可确保 OIDC 提供商和 Kubernetes 之间的安全通信，从而降低未经授权的访问或数据泄露的风险。

使用 OIDC 设置 Kubernetes 身份验证

要在 Kubernetes 中启用 OIDC 身份验证，您需要遵循以下常规步骤：

第 1 步：配置 OIDC 提供程序 ：首先，您需要设置一个 OIDC 提供程序，它可以是 Keycloak 等开源解决方案，也可以是 Okta 或 Azure Active Directory 等基于云的服务。使用必要的客户端凭据、范围和重定向 URI 配置提供程序。

步骤 2：配置 Kubernetes API 服务器 ：更新 Kubernetes API 服务器配置以启用 OIDC 身份验证。这涉及指定 OIDC 提供商的端点、客户端 ID 和客户端密码。此外，您可以定义映射到 Kubernetes RBAC 角色的组或声明，从而实现细粒度的访问控制。

步骤 3：配置 Kubernetes 集群角色 ：根据 OIDC 提供商的组或声明映射定义 Kubernetes 集群角色和角色绑定。这可确保通过 OIDC 进行身份验证的用户在集群内被授予适当的权限。

步骤 4：测试和验证： 配置到位后，通过使用 OIDC 凭据登录来测试 OIDC 身份验证。验证用户是否已成功通过身份验证并被授予预期的访问权限。

Kubernetes OIDC 身份验证的最佳实践

为了确保在 Kubernetes 中安全高效地实施 OIDC 身份验证，请考虑以下最佳实践：

1. 使用 HTTPS：始终通过启用 HTTPS 来保护 OIDC 提供程序、Kubernetes API 服务器和客户端应用程序之间的通信。

2. 启用 RBAC：实施基于角色的访问控制 (RBAC)，以根据用户组或声明实施细粒度的授权策略。 RBAC 允许您控制用户可以在 Kubernetes 集群内访问哪些资源和操作。

3. 令牌验证和轮换：验证从 OIDC 提供商收到的 JWT 令牌的完整性和真实性。实施 token 轮换机制，确保长期 token 的安全。

4. 监控和审计：设置监控和审计机制来跟踪和记录 Kubernetes 集群内的身份验证事件。这有助于检测和响应任何潜在的安全事件。

案例

让我们看一下使用特定规范通过 OIDC 配置 Kubernetes 身份验证的示例。

在此示例中，我们将使用 Keycloak 作为 OIDC 提供程序，并假设您已经设置了 Keycloak 实例。 Kubernetes集群正在运行1.21或更高版本。

第1步：配置OIDC提供商（Keycloak）

1. 在 Keycloak 中创建一个新 realm 或使用现有 realm。

2. 在为 Kubernetes 创建的 realm 中创建一个新客户端。将客户端协议设置为“openid-connect”，将访问类型设置为“机密”。

3. 使用 Kubernetes API 服务器的 OAuth2 回调端点的 URL（例如 https://kubernetes/api/v1/auth/callback）配置有效重定向 URI。

4. 记下创建的客户端的客户端 ID 和客户端密钥。

步骤 2：配置 Kubernetes API 服务器

1. 在主节点上打开 Kubernetes API 服务器配置文件（例如 /etc/kubernetes/manifests/kube-apiserver.yaml）。

2. 将以下参数添加到 kube-apiserver 容器规范中：

```
- --oidc-issuer-url=https://keycloak/auth/realms/<realm>
- --oidc-client-id=<client-id>
- --oidc-username-claim=preferred_username
- --oidc-groups-claim=groups
- --oidc-ca-file=/etc/kubernetes/pki/keycloak-ca.crt
- --oidc-username-prefix=oidc:
```

将 `<realm>` 替换为您的 Keycloak realm 的名称，将 `<client-id>` 替换为您在 Keycloak 中创建的 Kubernetes 客户端的客户端 ID。确保提供 OIDC CA 文件的正确路径，其中包含 Keycloak 实例的证书。

3. 保存配置文件并让 Kubernetes 自动应用更改。

步骤3：配置Kubernetes集群角色

1. 在 Kubernetes 中创建映射到 Keycloak 组的 ClusterRole。例如：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: keycloak-admins
rules:
  - apiGroups: [""]
    resources: ["pods", "services"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

2. 创建 ClusterRoleBinding 以将 ClusterRole 分配给特定的 Keycloak 组。例如：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: keycloak-admins-binding
subjects:
  - kind: Group
    name: keycloak-admins
roleRef:
  kind: ClusterRole
  name: keycloak-admins
  apiGroup: rbac.authorization.k8s.io
```

将 `keycloak-admins` 替换为要映射到 ClusterRole 的 Keycloak 组的名称。

第 4 步：测试和验证

1. 重新启动 Kubernetes API 服务器以应用更改。

2. 使用 OIDC 身份验证流程访问 Kubernetes API 服务器。例如，您可以使用 kubectl 命令行工具：

```
kubectl config set-credentials oidc-user --auth-provider=oidc \
  --auth-provider-arg=idp-issuer-url=https://keycloak/auth/realms/<realm> \
  --auth-provider-arg=client-id=<client-id> \
  --auth-provider-arg=client-secret=<client-secret> \
  --auth-provider-arg=refresh-token=<refresh-token> \
  --auth-provider-arg=id-token=<id-token>
```

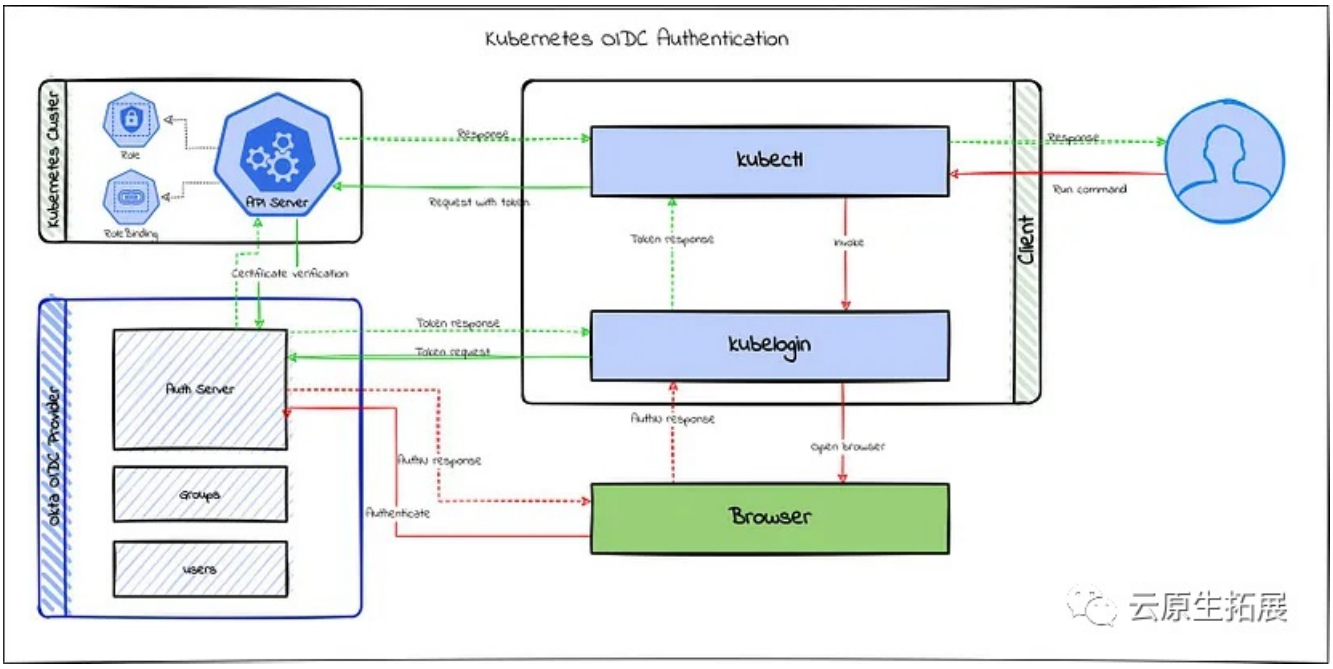
将 `<realm>` 、 `<client-id>` 、 `<client-secret>` 、 `<refresh-token>` 和 `<id-token>` 替换为从 Keycloak 获取的适当值。确保为您的设置使用正确的 URL 和身份验证参数。

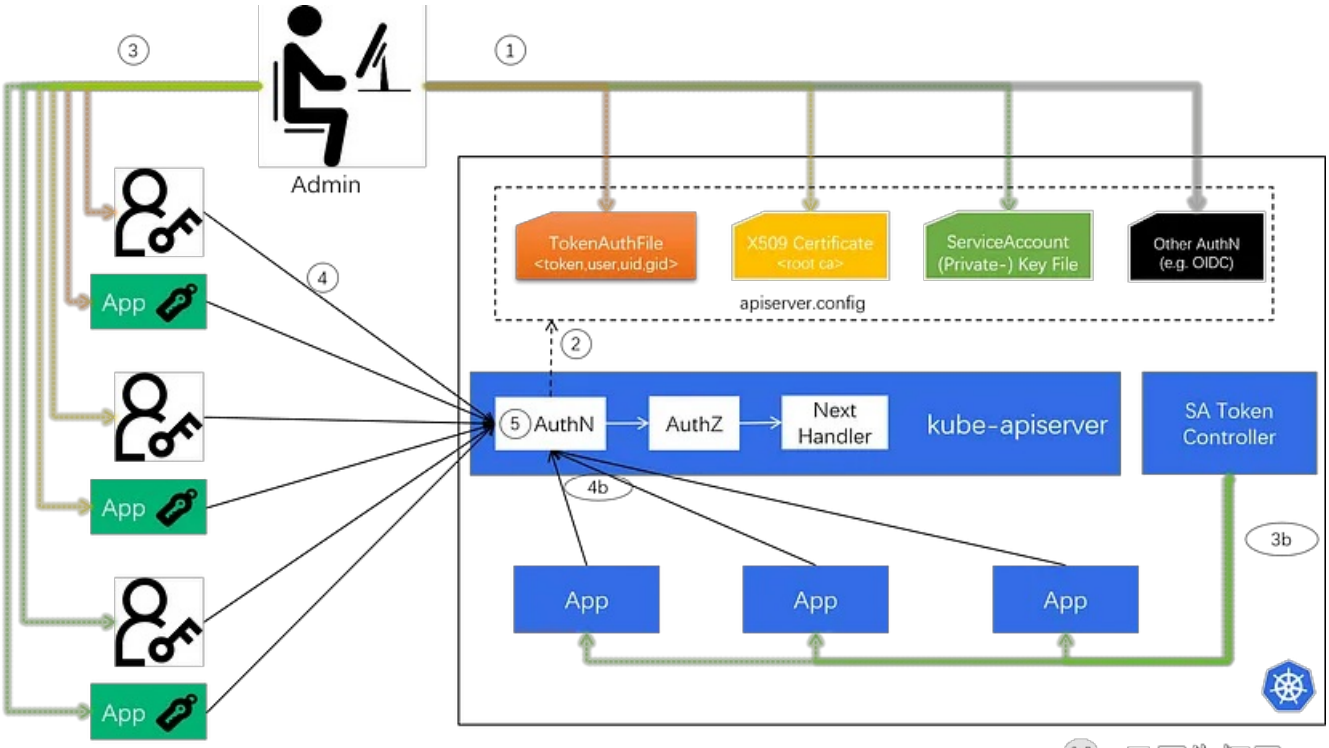
3. 验证您是否可以根据分配的角色和组映射访问 Kubernetes 资源。

就是这样！您已使用 Keycloak 作为 OIDC 提供程序成功配置了 OIDC 的 Kubernetes 身份验证。此设置允许通过 Keycloak 进行身份验证的用户根据分配的角色和组映射访问 Kubernetes 资源。请记住根据您的特定 OIDC 提供商和 Kubernetes 设置来调整配置。

总结

使用 OIDC 的 Kubernetes 身份验证为管理 Kubernetes 集群中的用户身份提供了强大且可扩展的解决方案。通过利用 OIDC 提供商，组织可以简化身份验证流程、提供无缝的单点登录体验并增强安全性。 OIDC 的灵活性和可扩展性使其成为将 Kubernetes 与现有身份管理基础设施集成的宝贵工具。与最佳实践相结合，OIDC 身份验证可以帮助组织在 Kubernetes 环境中实现安全高效的身份管理，使他们能够专注于交付可靠且可扩展的应用程序。




3
Admin
1
TokenAuthFile
<token,user,uid,gid>
X509 Certificate
<root ca>
ServiceAccount
(Private-) Key File
Other AuthN
(e.g. OIDC)
apiserver.config
4
2
App
App
App
App
App
5 AuthN
AuthZ
Next
Handler
kube-apiserver
SA Token
Controller
4b
3b
App
App
App
Out-of-cluster users/apps
Kubernetes cluster (control plane components and in-cluster apps)