

K8S RESTful API 使用

西安电子科技大学 XenoWYC121

1. K8S 权限控制

与直接使用 kubectl 不同，使用 RESTful API 时，k8s api server 需要额外手段来验证用户的身份与权限。

1.1 常见 K8S 用户验证方式

1.1.1 TLS 客户端证书

使用自签发 TLS 证书。TLS 客户端证书认证使用公钥基础设施（PKI）来验证客户端的身份。客户端和服务端都持有证书，可以使用 certificates.k8s.io API 或者直接利用自己的 CA 证书进行签发。

优点：

- 使用密钥对，安全性高。

缺点：

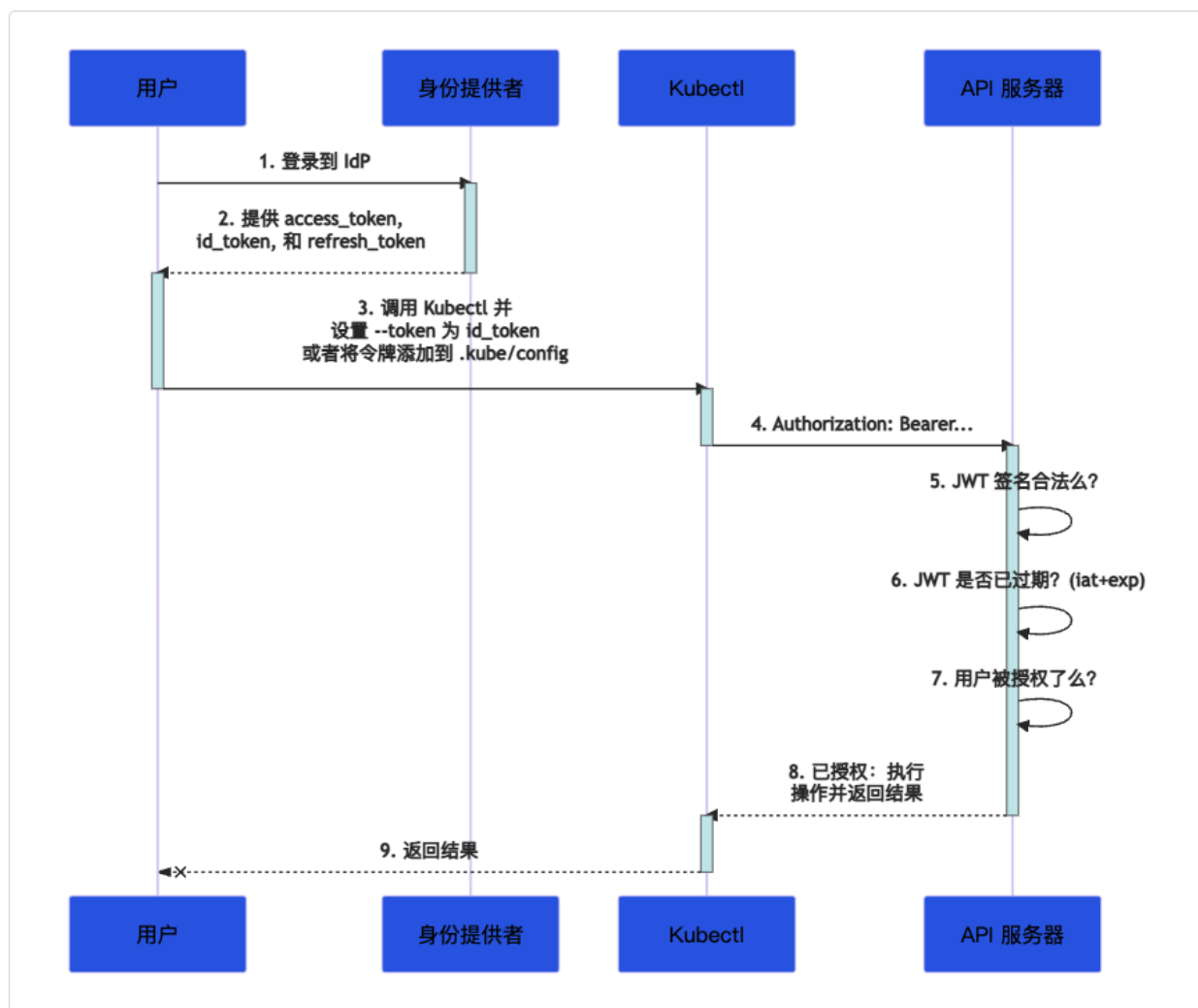
- 管理证书比较复杂，需要定期更新和分发。

1.1.2 OpenID Connect

OAuth（Open Authorization）是一个关于授权（authorization）的开放网络标准，允许用户授权第三方应用访问他们存储在其他服务提供者上的信息，而不需要将用户名和密码提供给第三方应用。

OAuth 在全世界得到了广泛的应用，目前的版本是 2.0。

OpenID Connect (OIDC) 是一种身份验证协议，基于 OAuth 2.0 系列规范。OAuth2 提供了 access_token 来解决授权第三方客户端访问受保护资源的问题，OpenID Connect 在这个基础上提供了 id_token 来解决第三方客户端标识用户身份的问题。



优点:

- 使用 JWT 令牌, 安全性高。
- 易于集成第三方身份验证提供者。

缺点:

- 配置复杂
- 需要额外的身份提供者

1.1.3 Webhook Token

通过 HTTP 回调请求到外部服务进行身份验证。

主要流程如下:

1. 在 Kubernetes apiserver 中配置外部身份认证服务器地址。
2. 客户端发送请求时携带有 token

3. Kubernetes apiserver 接收到请求后，将 Token 转发给配置的外部服务器。
4. 外部服务器验证 Token 的有效性，并返回认证结果。
5. 根据返回的结果，apiserver 决定是否授权访问资源。

优点：

- 灵活性高，可以自定义身份验证逻辑
- 易于集成现有系统

缺点：

- 配置和维护复杂，需要编写和管理 Webhook 服务

1.1.4 Service Account Token

Service Account 是 Kubernetes 中的一种内置身份，常用于在 Pod 中运行的应用程序进行身份验证和授权。

优点：

- 内置于 Kubernetes，易于使用和管理。
- 可以与 RBAC（角色访问控制）结合，精细化权限管理。

缺点：

- 安全性不如上述几种方法，因此不适用于外部用户认证。

1.2 K8S 权限控制

1.2.1 基于角色的访问控制 RBAC

基于角色的访问控制是一种访问控制机制，通过将权限分配给用户的角色，而不是直接分配给用户来管理访问权限。每个角色代表一组特定的权限，用户被分配到这些角色中，从而继承相应的权限。

1.2.1.1 主要概念

1. 用户：需要访问系统的实体
2. 角色：一组权限的集合，表示特定的职位或职责
3. 权限：对资源的操作，如增删改查

4. 资源：需要保护的對象，如文件、数据库记录、应用程序功能等等

1.2.1.2 工作方式

1. 定义角色：根据业务需求定义各种角色，并为每个角色分配相应的权限。
2. 分配角色：将用户分配到适当的角色中。用户继承该角色的所有权限。
3. 访问控制：当用户尝试访问资源时，系统根据用户所属的角色来决定是否允许访问。

1.2.1.3 用户与角色的关系

在 RBAC 中，用户与权限解耦，不直接关联。角色是权限的集合，代表了特定的职责。角色会作为关联权限与用户的中间层，赋予用户一个角色，用户将继承角色的权限。通过这种方式，可以简化权限管理，提高系统的安全性和灵活性。

1.2.1.4 RBAC 的优点

1. 简化权限管理：通过角色分配权限，而不是直接给每个用户分配权限，简化了权限管理工作。
2. 灵活：将用户与权限解耦，可以方便地根据业务需求和用户职责的变化，快速调整权限。修改角色的权限后，属于该角色的所有用户权限都会改变。
3. 便于审计：RBAC 提供了明确的权限分配，可以方便地审计和跟踪谁对哪些资源有访问权限，有助于合规性检查。

1.2.2 service account token 使用

Service Account Token 方式简单易用，并且可以轻松和 K8S RBAC 相结合，便于演示。**本文中第二部分 POD 管理相关内容的鉴权都由 Service Account Token 方式实现。**以下将说明如何创建 service account、创建角色、绑定角色以及生成 token。

1.2.2.1 创建 service account

创建 service account 有两种方式，可以使用 kubectl 命令，也可以声明 yaml 文件。这里创建的 service account 不需要额外配置，可以直接使用命令：

```
kubectl create sa user1
```

1.2.2.2 创建角色

同样拥有两种创建方式，此处使用命令。通过以下命令创建 pod-admin 角色，并授予对于 pod 资源的 get,list,watch,create,update,patch,delete 权限：

```
kubectl create role pod-admin --verb=get,list,watch,create,update,patch,delete --resource=pods
```

1.2.2.3 角色绑定

通过以下命令将 pod-admin 角色与 user1 service account 绑定，实现用户授权。

```
kubectl create rolebinding user-rolebinding --role=pod-admin --serviceaccount=default:user1
```

1.2.2.4 token(secret)生成

注意：如果 k8s 版本小于 1.24，则不需要此步骤。1.24.0 更新之后创建 Service Account 不会自动生成 Secret 需要对其手动创建。

使用命令生成 secret 时需要添加的字段多，不如 yaml 文件方便，所以此处使用 yaml 声明 secret：

```
apiVersion: v1
kind: Secret
metadata:
  name: user-secret
  namespace: default
  annotations:
    kubernetes.io/service-account.name: "user1"
type: kubernetes.io/service-account-token
```

使用以下命令即可查看生成的 secret：

```
kubectl get secret user-secret -o jsonpath='{.data}'
```

2. POD 管理

以下内容为了便于阅读使用 python 实现了对应的接口。

2.1 POD 创建

RESTful API:

POST /api/v1/namespaces/{namespace}/pods

参数:

- namespace (路径参数): string, 必需
- body: Pod, 必需
- dryRun (查询参数): string
- fieldManager (查询参数): string
- fieldValidation (查询参数): string
- pretty (查询参数): string

响应:

- 200 (Pod): OK
- 201 (Pod): Created
- 202 (Pod): Accepted
- 401: Unauthorized

解释:

使用 POST 方法提交的 POD 声明(json 或 yaml 格式), 用于创建一个新的 POD 资源, 请求体为 POD 声明。

2.2 POD 删除

RESTful API:

DELETE /api/v1/namespaces/{namespace}/pods/{name}

参数:

- name (路径参数): string, 必需
- namespace (路径参数): string, 必需
- body: DeleteOptions
- dryRun (查询参数): string

- gracePeriodSeconds (查询参数): integer
- pretty (查询参数): string
- propagationPolicy (查询参数): string

响应:

- 200 (Pod): OK
- 202 (Pod): Accepted
- 401: Unauthorized

解释:

关键参数为路径参数: name, 用于指定要删除的 POD 名称。

2.3 POD 查看

RESTful API:

GET /api/v1/namespaces/{namespace}/pods/{name}

参数:

- name (路径参数): string, 必需
- namespace (路径参数): string, 必需
- pretty (查询参数): string

响应:

- 200 (Pod): OK
- 401: Unauthorized

解释:

获取指定 POD 的详细信息

2.4 列出 POD

RESTful API:

GET /api/v1/namespaces/{namespace}/pods

参数

- namespace (路径参数): string, 必需
- allowWatchBookmarks (查询参数): boolean
- continue (查询参数): string
- fieldSelector (查询参数): string
- labelSelector (查询参数): string
- limit (查询参数): integer
- pretty (查询参数): string
- resourceVersion (查询参数): string
- resourceVersionMatch (查询参数): string
- sendInitialEvents (查询参数): boolean
- timeoutSeconds (查询参数): integer
- watch (查询参数): boolean

响应:

- 200 (PodList): OK
- 401: Unauthorized

解释:

列出 namespace 下符合要求的所有 pod

2.5 更新 POD

RESTful API:

PUT /api/v1/namespaces/{namespace}/pods/{name}

参数:

- name (路径参数): string, 必需
- namespace (路径参数): string, 必需
- body: Pod, 必需
- dryRun (查询参数): string
- fieldManager (查询参数): string

- fieldValidation (查询参数): string
- pretty (查询参数): string

响应:

- 200 (Pod): OK
- 201 (Pod): Created
- 401: Unauthorized

解释:

使用 PUT 方法, 上传完整 POD 声明(json 或 yaml 格式), 用于修改一个已有 POD 资源, 请求体为 POD 声明。

注意:

1. 只有以下字段可以更新:

- spec.containers[*].image
- spec.initContainers[*].image
- spec.activeDeadlineSeconds
- spec.tolerations
- spec.terminationGracePeriodSeconds

2. 更新 POD 需要完整的 POD 声明!

- 使用 GET 方法获取 POD 的完整 yaml 或 json 声明, 再在 GET 得到的 POD 声明上进行字段修改。
- 不可使用创建 POD 时的 yaml!!!

2.6 通用查询参数 (2.1-2.5 都会使用的)

2.6.1 pretty

如果设置为 'true', 那么输出是规范的打印。(字段会被缩进和换行, 便于阅读)

2.6.2 fieldValidation

fieldValidation 指示服务器如何处理请求 (POST/PUT/PATCH) 中包含未知或重复字段的对象。有效值为:

- **Ignore**: 这将忽略从对象中默默删除的所有未知字段，并将忽略除解码器遇到的最后一个重复字段之外的所有字段。这是在 v1.23 之前的默认行为。
- **Warn**: 这将针对从对象中删除的各个未知字段以及所遇到的各个重复字段，分别通过标准警告响应头发出警告。如果没有其他错误，请求仍然会成功，并且只会保留所有重复字段中的最后一个。这是 v1.23+ 版本中的默认设置。
- **Strict**: 如果从对象中删除任何未知字段，或者存在任何重复字段，将使请求失败并返回 `BadRequest` 错误。

2.6.3 dryRun

如果此字段存在，那么其值只能是 `All`。DryRun 是一个检验请求，带有 DryRun 参数的请求只会检查其合法性，而不会对资源产生影响。使用这种方式，可以仅仅通过调用来确定一个请求的参数是否合理，账号是否有足够的权限调用这个接口。

2.6.4 gracePeriodSeconds

删除对象前的持续时间（秒数）。值必须为非负整数。取值为 0 表示立即删除。如果该值为 `nil`，将使用指定类型的默认宽限期。如果没有指定，默认为每个对象的设置值。0 表示立即删除。

2.6.5 propagationPolicy

该字段决定了删除对象是所使用的垃圾回收策略。

2.6.6 timeoutSeconds

`list/watch` 调用的超时秒数。

2.6.7 resourceVersion

`resourceVersion` 对请求所针对的资源版本设置约束。

2.6.8 limit

`limit` 是一个列表调用返回的最大响应数。

2.6.9 labelSelector

通过标签限制返回对象列表