- 深入分析集群安全机制
  - 1.简介
    - 1.Api Server 的认证授权、准入控制机制及保护敏感信息的Secret机制等
    - 2.集群安全性必须考虑的目标
      - 1.保证容器与其所在宿主机的隔离
      - 2.限制容器给基础设施或其他容器带来的干扰
      - 3.最小权限原则——合理限制所有组件的权限,确保组件只执行它被授权的行为,通过限制单个组件的能力来限制它的权限范围
      - 4.明确组件间边界的划分
      - 5.划分普通用户和管理员角色
      - 6.在必要时允许将管理员权限赋给普通用户
      - 7.允许拥有Secret数据的应用在集群中运行
  - 2.API Server认证管理
    - 1.kubernetes集群提供了3种级别的客户端身份认证方式
      - 1.最严格的HTTPS证书认证:基于CA根证书签名的双向数字证书认证方式
      - 2.HTTP Token认证:通过一个Token来识别合法用户
      - 3.HTTP Base认证:通过用户名+密码的方式认证
    - 2.HTTPS证书认证的原理
    - 3.CA认证大概包含下面几个步骤
      - 1.HTTPS通信双方的服务端向CA机构申请证书,CA机构是可信的第三方机构,它可以是一个公认的权威企业,也可以是企业自身
      - 2.HTTPS通信双方的客户端向CA机构申请证书,CA机构下发根证书、客户端证书及私钥给申请者
      - 3.客户端向服务端发起请求,服务端下发服务端证书给客户端.客户端接收到证书后,通过私钥解密证书,并利用服务器端证书中的公钥认证证书信息比较证书里的消息
      - 4.客户端发送客户端证书给服务器,服务端在接收证书后,通过私钥解密证书,获取客户端证书公钥,并用该公钥认证证书信息,确认客户端是否合法
      - 5.客户端通过随机密钥加密信息,并发送加密后的信息给服务端.在服务端和客户端协商好加密方案后,客户端会产生一个随机的密钥,客户端通过协商好的加密方案加密该随机密钥,并发送该随机密钥到服务器端.服务器端接收这个密钥后,双方通信的所有内容都通过该随机密钥加密
    - 4.HTTP Token的认证原理 1.
  - 3.API Server授权管理
    - 1.API Server的几种授权策略
      - 1.AlwaysDeny:表示拒绝所有请求,一般用于测试
      - 2.AlwaysAllow:允许接收所有请求,如果集群不需要授权流程,则可以采用该策略
      - 3.ABAC(Attribute-Based Access Control):基于属性的访问控制,表示使用用户配置的授权规则对用户请求进行匹配和控制
      - 4.Webhook:通过调用外部REST服务对用户进行授权
      - 5.RBAC:基于角色的访问控制
      - 6.Node:是一种专用的模式,用于对kubelet发出的请求进行访问控制
    - 2.ABAC授权模式详解
      - 1.指定授权策略文件的路径和名称(--authorization- policy-file=SOME_FILENAME)
      - 2.apiVersion:当前版本为abac.authorization.kubernetes.io/v1beta1
      - 3.kind:被设置为Policy
      - 4.spec:指详细的策略设置,包括主题属性、资源属性、非资源属性这三个字段
        - 1.主题属性
          - 1.user(用户名):字符串类型,该字段类型的用户名来源于Token文件(--token-auth-file参数设置的文件)或基本认证文件中 用户名称段的值
          - 2.group(用户组):在被设置为“system:authenticated”时表示匹配所有已认证的请求,在被设置为“system:unauthenticated”时表示匹配所有未认证的请求
        - 2.资源属性
          - 1.apiGroup(API组):字符串类型,表明匹配哪些API Group
          - 2.namespace(命名空间):字符串类型,表明该策略允许访问某个Namespace的资源
          - 3.resource(资源):字符串类型,API资源对象
        - 3.非资源属性
          - 1.nonResourcePath(非资源对象类路径):非资源对象类的URL路径
          - 2.readonly(只读标识):布尔类型,当它的值为true时,表明仅允许GET请求通过
      - 5.ABAC授权算法
        - 1.在API Server收到请求之后,首先识别出请求携带的策略对象的属性,然后根据在策略文件中定义的策略对这些属性进行逐条匹配,以判定是否允许授权
        - 2.如果有至少一条匹配成功,那么这个请求就通过了授权(不过还是可能在后续其他授权校验中失败)
        - 3.常见匹配策略
          - 1.要允许所有认证用户做某事,可以写一个策略,将group属性设置为system:authenticated
          - 2.要允许所有未认证用户做某事,可以把策略的group属性设置为system:unauthenticated
          - 3.要允许一个用户做任何事,将策略的apiGroup、namespace、resource和nonResourcePath属性设置为“*”即可
      - 6.使用kubectl时的授权机制
        - 1.kubectl使用API Server的/api和/apis端点来获取版本信息
        - 2.当使用ABAC授权模式时,特殊资源必须显示地通过nonResourcePath属性进行设置
          - 1.API版本协商过程中的/api、/api/*、/apis和/apis/*
          - 2.使用kubectl version命令从服务器获取版本时的/version
          - 3.create/update操作过程中的/swaggerapi/*
      - 7.常见的ABAC授权示例
        - 1.允许用户alice对所有资源做任何操作:{"apiVersion":"abac.authorization.kubernetes.io/v1beta1","kind":"Policy","spec":{"user":"alice","namespace":"*","resource":"*","apiGroup":"*"}}
        - 2.kubelet可以读取任意Pod:{"apiVersion":"abac.authorization.kubernetes.io/v1beta1","kind":"Policy","spec":{"user":"`","namespace":"*","resource":"pods","readonly":"true"}}
        - 3.kubelet可以读写Event对象:{"apiVersion":"abac.authorization.kubernetes.io/v1beta1","kind":"Policy","spec":{"user":"kubelet","namespace":"*","resource":"events"}}
        - 4.用户bob只能读取projectCaribou中的Pod:{"apiVersion":"abac.authorization.kubernetes.io/v1beta1","kind":"Policy","spec":{"user":"bob","namespace":"projectCaribou","resource":"pods","readonly":true}}
        - 5.任何用户都可以对非资源类路径进行只读请求:{"apiVersion":"abac.authorization.kubernetes.io/v1beta1","kind":"Policy","spec":{"group":"system.authenticated","readonly":true,"nonResourcePath":"*"}}、{"apiVersion":"abac.authorization.kubernetes.io/v1beta1","kind":"Policy","spec":{"group":"system.unauthenticated","readonly":true,"nonResourcePath":"*"}}
      - 8.对Service Account进行授权
        - 1.Service Account会自动生成一个ABAC用户名(username),用户名按照以下命名规则生成:(system:serviceaccount:<namespace>:<serviceaccountname>
        - 2.创建新的命名空间时,会产生一个如下名称的Service Account:(system:serviceaccount:<namespace>:default)
        - 3.如果希望kube-system命名空间中的Service Account“default”具有全部权限,就要在策略文件中加入如下内容:{"apiVersion":"abac.authorization.kubernetes.io/v1beta1","kind":"Policy","spec":{"user":"system:serviceaccount:kube-system:default","namespace":"*","resource":"*","apiGroup":"*"}}
    - 3.Webhook授权模式详解
      - 1.Webhook定义了一个HTTP回调接口,实现Webhook的应用会在指定事件发生时,向一个URL地址发送(POST)通知信息
      - 2.启用Webhook授权模式后,Kubernetes会调用外部REST服务对用户进行授权
    - 4.RBAC授权模式详解
      - 1.优势
        - 1.对集群中的资源和非资源权限均有完整的覆盖
        - 2.整个RBAC完全由几个API对象完成,同其他API对象一样,可以用kubectl或API进行操作
        - 3.可以在运行时进行调整,无需重启API Server
      - 1.角色(Role)
        - 1.一个角色就是一组权限的集合,这里的权限都是许可形式,不存在拒绝规则
        - 2.在一个命名空间中,可以用角色来定义一个角色,如果是集群级别的,就需要使用ClusterRole
        - 3.角色只能对命名空间进行授权
        - 4.rules中的参数说明
          - 1.apiGroup:支持的API组列表
          - 2.resources:支持的资源对象列表,例如pods、deployments、jobs等
          - 3.verbs:对资源对象的操作方法列表,例如get、watch、list、delete、replace、patch等
      - 2.集群角色(ClusterRole)
        - 1.集群角色除了具有和角色一致的命名空间内资源的管理能力,因其集群级别的范围,还可以用于以下特殊元素的授权
          - 1.集群范围的资源,例如Node
          - 2.非资源型的路径,例如“/healthz”
          - 3.包含全部命名空间的资源,例如pods
      - 3.角色绑定(RoleBinding)和集群角色绑定(ClusterRoleBinding)
        - 1.角色绑定或集群角色绑定用来把一个角色绑定到一个目标上,绑定目标可以是User(用户)、Group(组)或者Service Account.
        - 2.使用RoleBinding为某个命名空间授权,使用ClusterRoleBinding为集群范围授权
      - 4.对资源的引用方式
        - 1.多数资源可以用其名称的字符串来表达,也就是Endpoint中的URL相对路径
        - 2.用斜线“/”来分割资源和下级资源

- 2.RBAC的API资源对象说明
  - 5.常见的角色示例
    - 1.允许读取核心API组中的Pod资源:
      rules:
      - apiGroups:[""]
        resources:["pods"]
        verbs:["get","list","watch"]
    - 2.允许读写extensions和apps两个API组中的deployment资源:
      rules:
      - apiGroups:["extensions","apps"]
        resources:["deployments"]
        verbs:["get","list","watch","create","update","patch","delete"]
    - 3.允许读取pods及读写jobs"
      rules:
      - apiGroups:[""]
        resources:["pods"]
        verbs:["get","list","watch"]
      rules:
      - apiGroups:["batch","extensions"]
        resources:["jods"]
        verbs:["get","list","watch","create","update","patch","delete"]
    - 4.允许读取一个名为my-config的ConfigMap(必须绑定到一个RoleBinding来限制到一个Namespace下的ConfigMap):
      rules:
      - apiGroups:[""]
        resources:["configmaps"]
        resourceNames:["my-config"]
        verbs:["get"]
    - 5.读取核心组的Node资源(Node属于集群级的资源,所以必须存在于ClusterRole中,并使用ClusterRoleBinding进行绑定):
      rules:
      - apiGroups:[""]
        resources:["nodes"]
        verbs:["get","list","watch"]
    - 6.允许对非资源端点“/healthz”及其所有子路径进行GET和POST操作(必须使用ClusterRole和ClusterRoleBinding):
      rules:
      - nonResourceURLs:["/healthz","/healthz/*"]
        verbs:["get","post"]