Kubernetes 安全框架（上）-RBAC

作者：Leif160519
原文链接：https://ld246.com/article/1595755054196
来源网站：链滴

- 访问K8S集群的资源需要过三关：认证、鉴权、准入控制
- 普通用户若要安全访问集群API

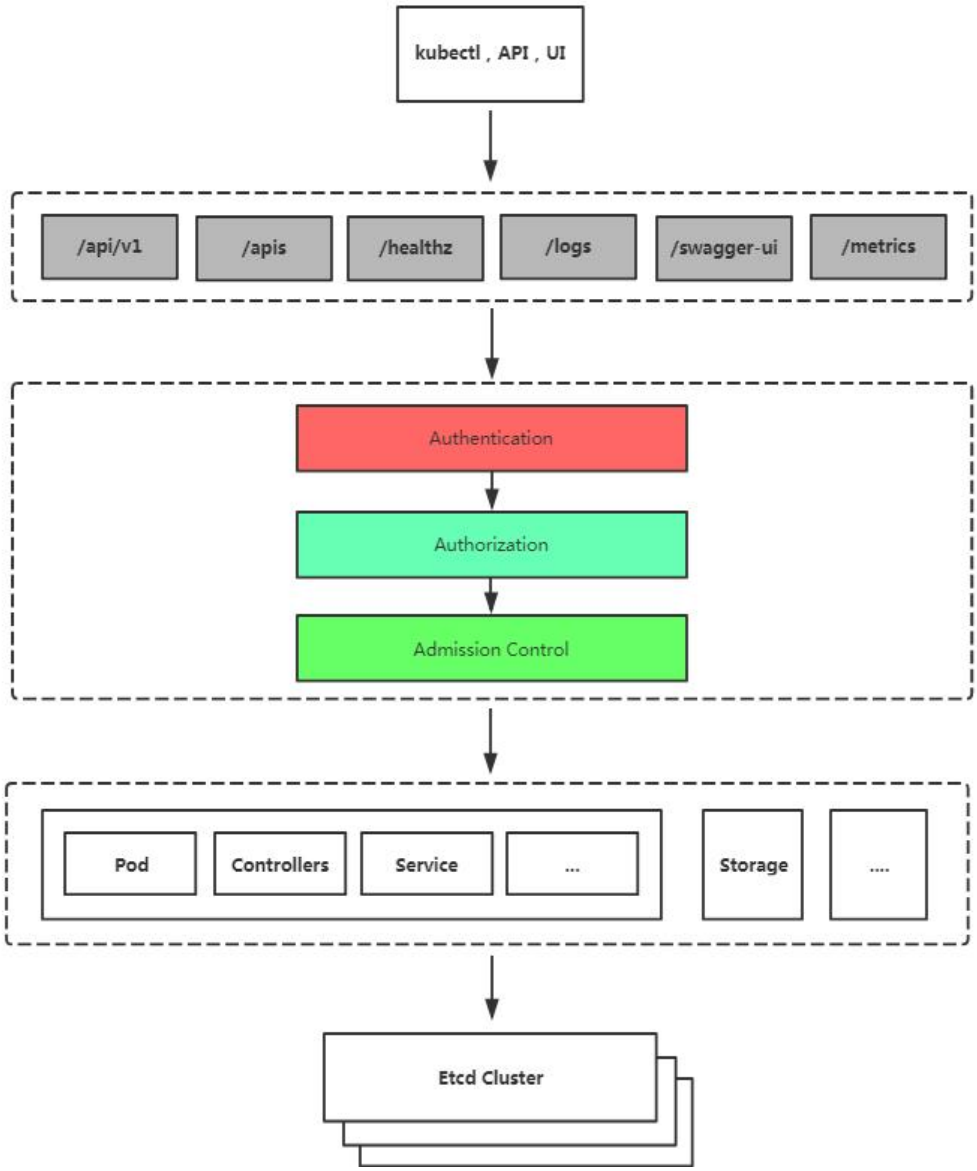
Server，往往需要证书、Token或者用户名+密码；Pod访问，需要 ServiceAccount

- K8S安全控制框架主要由下面3个阶段进行控制，每一个阶段都支持插件方式，通过API Server配置启用插件。

1.Authentication（鉴权）

2.Authorization（授权）

3.Admission Control（准入控制）



1.鉴权（Authentication）

三种客户端身份认证：

- HTTPS 证书认证：基于CA证书签名的数字证书认证
- HTTP Token认证：通过一个Token来识别用户
- HTTP Base认证：用户名+密码的方式认证

上述第一种：

apiserver http接口：https通信

1. 安全通信
2. 基于证书来认证（从证书里拿用户名、用户组）

2.授权（Authorization）

RBAC（Role-Based Access

原文链接：Kubernetes 安全框架（上）-RBAC

Control，基于角色的访问控制）：负责完成授权（Authorization）工作。

根据API请求属性，决定允许还是拒绝。

- user：用户名
- group：用户分组
- extra：用户额外信息
- API
- 请求路径：例如/api，/healthz
- API请求方法：get，list，create，update，patch，watch，delete
- HTTP请求方法：get，post，put，delete
- 资源
- 子资源
- 命名空间
- API组

3.准入控制（Admission Control）

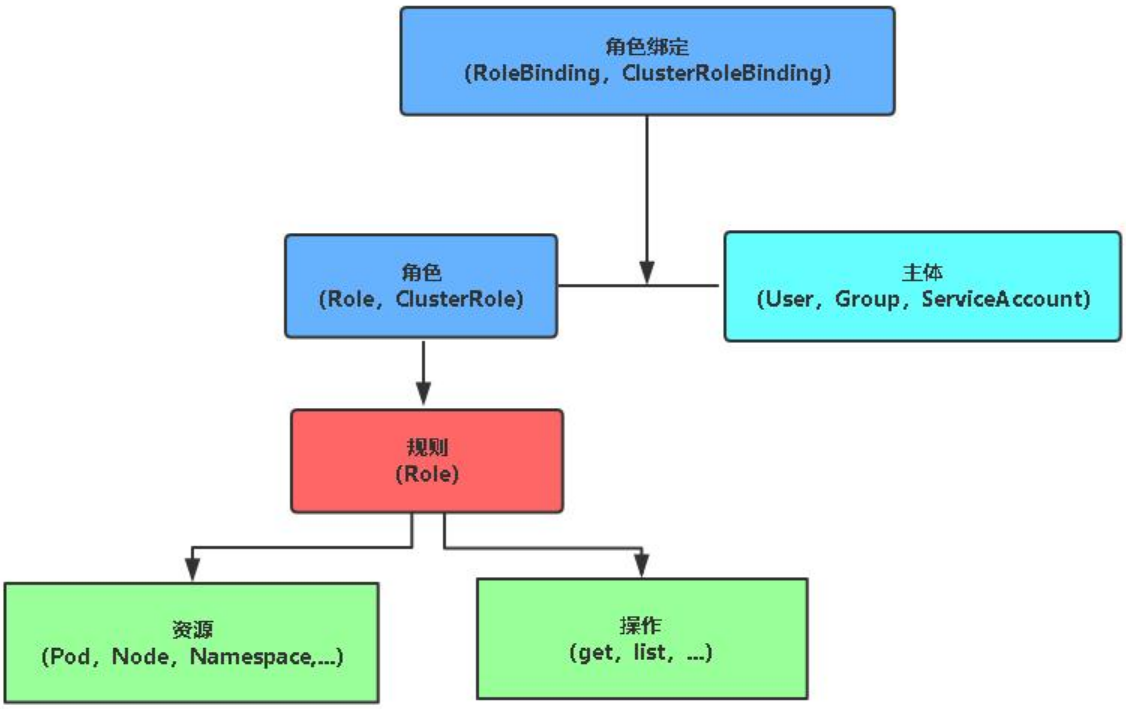
AdminssionControl实际上是一个准入控制器插件列表，发送到APIServer的请求都需要经过这个列中的每个准入控制器插件的检查，检查不通过，则拒绝请求。

准入控制：

- 用户可以定制化插件
- 官方实现一些高级插件

基于角色的权限访问控制：RBAC

RBAC（Role-Based Access
Control，基于角色的访问控制），允许通过Kubernetes API动态配置策略。



分类

角色

- Role：授权特定命名空间的访问权限
- ClusterRole：授权所有命名空间的访问权限

角色绑定

•RoleBinding：将角色绑定到主体（即subject）

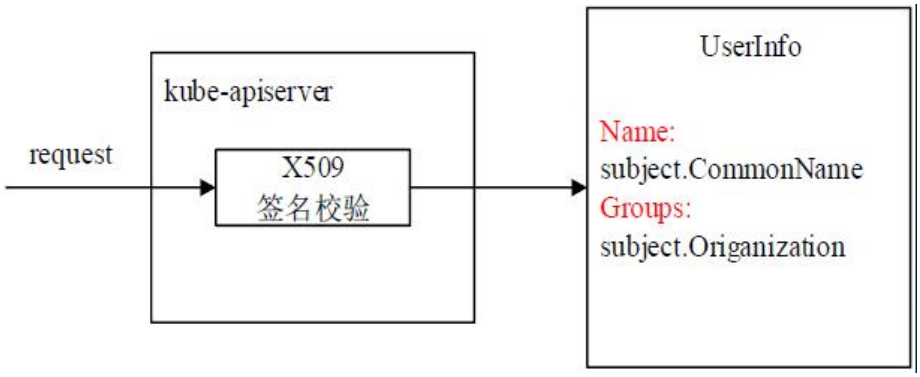
•ClusterRoleBinding：将集群角色绑定到主体

主体（subject）

- User：用户
- Group：用户组
- ServiceAccount：服务账号（给应用去赋予权限，参考Ingress配置，当k8s应用程序要访问apiserver时，需要配置ServiceAccount，让应用程序携带token去访问apiserver）

案例

为aliang用户授权default命名空间Pod读取权限



图片解释：当给apiserver发起一个请求时，apiserver会查看校验x509格式的证书，从中提取用户名组，然后匹配当前创建的RBAC规则，看能不能满足现在发起的请求，如果不满足的话直接拒绝请求

- 1.用K8S CA签发客户端证书：　cert.sh

```
cat > ca-config.json <<EOF
{
  "signing": {
    "default": {
      "expiry": "87600h"
    },
    "profiles": {
      "kubernetes": {
        "usages": [
            "signing",
            "key encipherment",
            "server auth",
            "client auth"
        ],
        "expiry": "87600h"
      }
    }
  }
}
EOF

cat > aliang-csr.json <<EOF
{
  "CN": "aliang",
  "hosts": [],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "k8s",
      "OU": "System"
    }
  ]
```

```
}
EOF

cfssl gencert -ca=/etc/kubernetes/pki/ca.crt -ca-key=/etc/kubernetes/pki/ca.key -config=ca-onfig.json -profile=kubernetes aliang-csr.json | cfssljson -bare aliang
```

参数解释:

ca-config.json ： ca配置文件

aliang-csr.json：证书请求文件

CN: 用户名

O:用户组

若是二进制部署的k8s集群，请注意证书路径！！！

```
[root@k8s-master rbac]# ll
总用量 24
-rw-r--r-- 1 root root  997 7月  26 16:16 aliang.csr
-rw-r--r-- 1 root root  219 7月  26 16:16 aliang-csr.json
-rw------- 1 root root 1675 7月  26 16:16 aliang-key.pem
-rw-r--r-- 1 root root 1233 7月  26 16:16 aliang.pem
-rw-r--r-- 1 root root  292 7月  26 16:16 ca-config.json
-rwxr-xr-x 1 root root  741 5月  24 23:30 cert.sh
```

- 2.生成kubeconfig授权文件　kubeconfig.sh

```
kubectl config set-cluster kubernetes \
   --certificate-authority=/etc/kubernetes/pki/ca.crt \
   --embed-certs=true \
   --server=https://192.168.0.7:6443 \
   --kubeconfig=aliang.kubeconfig

# 设置客户端认证
kubectl config set-credentials aliang \
   --client-key=aliang-key.pem \
   --client-certificate=aliang.pem \
   --embed-certs=true \
   --kubeconfig=aliang.kubeconfig

# 设置默认上下文
kubectl config set-context kubernetes \
   --cluster=kubernetes \
   --user=aliang \
   --kubeconfig=aliang.kubeconfig

# 设置当前使用配置
kubectl config use-context kubernetes \
   --kubeconfig=aliang.kubeconfig
```

192.168.0.7指的是master节点的IP，不能随意指定

```
[root@k8s-master rbac]# ll
总用量 36
-rw-r--r-- 1 root root  997 7月  26 16:16 aliang.csr
-rw-r--r-- 1 root root  219 7月  26 16:16 aliang-csr.json
-rw------- 1 root root 1675 7月  26 16:16 aliang-key.pem
-rw------- 1 root root 5593 7月  26 16:29 aliang.kubeconfig
-rw-r--r-- 1 root root 1233 7月  26 16:16 aliang.pem
-rw-r--r-- 1 root root  292 7月  26 16:16 ca-config.json
-rwxr-xr-x 1 root root  741 5月  24 23:30 cert.sh
-rwxr-xr-x 1 root root  644 7月  26 16:27 kubeconfig.sh
```

aliang.kubeconfig内容格式与~/.kube/config一致

此时，使用kubectl --kubeconfig=aliang.kubeconfig get pod还是没有权限访问pod的，因为还需创建RBAC权限策略

- 3.创建RBAC权限策略

```
#定义role角色,权限组:
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]

 ---

#创建角色绑定,声明主体:
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: aliang
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

参数解释:

resources:可以访问的资源(后面需使用复数形式)

verbs：操作资源的方法

若想新增其他权限，请参看官方文档:使用 RBAC 鉴权，即时生效

apply过后既可以使用kubectl --kubeconfig=aliang.kubeconfig get pod访问pod了，但其余的资比如svc，指定命名空间或者是删除pod也无权访问。