ThoughtWorks®

Kubernetes RBAC

刘振伟&曹英勇

Role & ClusterRole

一个角色，包含了一组权限规则，权限以纯粹的累加形式存在，没有“否定”权限

Namespace中的角色可以由”Role”对象来定义

整个集群范围内的角色可以由”ClusterRole”对象来定义

Role

一个Role对象职能用于授权给某一个namespace中资源的访问权限

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group, 还可以指定为 ["extensions", "apps"]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

ClusterRole

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  # "namespace" omitted since ClusterRoles are not namespaced
  name: secret-reader
rules:
- apiGroups: [""]
  resources: [“secrets”,”pods"]
  verbs: ["get", "watch", "list"]
```

RoleBinding与ClusterRoleBinding

将一个角色中定义的各种权限授予一个或者一组用户。

包含了一组相关主体（即subject)—包括用户(User)、用户组(Group)、或者服务账户(Service Account)以及对被授予角色的引用。

在命名空间中可以通过RoleBinding对象授予权限，而集群范围的权限授予则通过ClusterRoleBinding对象完成

RoleBinding

可以引用在同一命名空间内定义的Role对象。 下面示例中定义的RoleBinding对象在”default”命名空间中将”pod-reader”角色授予用户”jane”。 这一授权将允许用户”jane”从”default”命名空间中读取pod

```
 kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

RoleBinding

RoleBinding对象也可以引用一个ClusterRole对象用于在RoleBinding所在的命名空间内授予用户对所引用的ClusterRole中定义的命名空间资源的访问权限。
这一点允许管理员在整个集群范围内首先定义一组通用的角色，然后再在不同的命名空间中复用这些角色

```
  kind: RoleBinding
  apiVersion: rbac.authorization.k8s.io/v1
  metadata:
    name: read-secrets
    namespace: development # This only grants permissions within
the "development" namespace.
  subjects:
  - kind: User
    name: dave # Name is case sensitive
    apiGroup: rbac.authorization.k8s.io
  roleRef:
    kind: ClusterRole
    name: secret-reader
    apiGroup: rbac.authorization.k8s.io
```

对资源的引用-－子资源

大多数资源由代表其名字的字符串表示，例如”pods”，就像它们出现在相关API endpoint的URL中一样。然而，有一些Kubernetes API还 包含了”子资源”，比如pod的logs。在Kubernetes中，pod logs endpoint的URL格式为

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

如果需要角色绑定主体读取pods以及pod log，您需要定义以下角色

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-and-pod-logs-reader
rules:
- apiGroups: [""]
  resources: ["pods", "pods/log"]
  verbs: ["get", "list"]
```

对资源的引用－resourceNames

通过resourceNames列表，角色可以针对不同种类的请求根据资源名引用资源实例

例如，如果需要限定一个角色绑定主体只能”get”或者”update”一个configmap时，您可以定义以下角色

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: configmap-updater
rules:
- apiGroups: [""]
  resources: ["configmaps"]
  resourceNames: ["my-configmap"]
  verbs: ["update", "get"]
```

ClusterRoles集合

从Kubernetes 1.9开始，可以使用aggregationRule关键字创建多个ClusterRole集合后的ClusterRole
该ClusterRole的rules是其匹配到的所有ClusterRole的rules的并集。如

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: monitoring
aggregationRule:
  clusterRoleSelectors:
  - matchLabels:
      rbac.example.com/aggregate-to-monitoring: "true"
rules: [] # Rules are automatically filled in by the controller manager.
```

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: monitoring-endpoints
  labels:
    rbac.example.com/aggregate-to-monitoring: "true"
# These rules will be added to the "monitoring" role.
rules:
- apiGroups: [""]
  Resources: ["services", "endpoints", "pods"]
  verbs: ["get", "list", "watch"]
```

DEMO

demo

Step 1

```
kubectl create serviceaccount duizhang

kubectl describe secret duizhang-token-xxxxxxx
```

demo

Step 2

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group, 还可以指定为 ["extensions", "apps"]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

demo

Step 3

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: read-pods
 namespace: default
subjects:
 - kind: ServiceAccount
   name: duizhang
   namespace: default
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

demo

Step 4

```
export KUBECONFIG=/root/nichousha.config

kubectl config set-cluster nichousha\
  --certificate-authority=/etc/kubernetes/ssl/ca.pem \
  --embed-certs=true \
  --server=https://192.168.56.50:6443
```

demo

Step 5

```
kubectl config set-credentials duizhang  —token=xxxxx
```

demo

Step 6

```
kubectl config set-context test \
  --cluster=nichousha \
  --user=duizhang

kubectl config use-context test
```

THANK YOU

ThoughtWorks®