### RBAC

1. RBAC基本概念

RBAC（Role-Based Access Control，基于角色的访问控制）是一种基于企业内个人用户的角色来管理对计算机或网络资源的访问方法，其在Kubernetes 1.5版本中引入，在1.6时升级为Beta版本，并成为Kubeadm安装方式下的默认选项。启用RBAC需要在启动APIServer时指定--authorization-mode=RBAC。

RBAC使用rbac.authorization.k8s.io API组来推动授权决策，允许管理员通过Kubernetes API动态配置策略。

RBAC API声明了4种顶级资源对象，即Role、ClusterRole、RoleBinding、ClusterRoleBinding，管理员可以像使用其他API资源一样使用kubectl API调用这些资源对象。例如：kubectl create -f (resource).yml。

2. Role和ClusterRole

Role和ClusterRole的关键区别是，Role是作用于命名空间内的角色，ClusterRole作用于整个集群的角色。

在RBAC API中，Role包含表示一组权限的规则。权限纯粹是附加允许的，没有拒绝规则。

Role只能授权对单个命名空间内的资源的访问权限，比如授权对default命名空间的读取权限：

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

namespace: default

name: pod-reader

rules:

- apiGroups: [""] # "" indicates the core API group

resources: ["pods"]

verbs: ["get", "watch", "list"]

ClusterRole也可将上述权限授予作用于整个集群的Role，主要区别是，ClusterRole是集群范围的，因此它们还可以授予对以下内容的访问权限：

* 集群范围的资源（如Node）。
* 非资源端点（如/healthz）。
* 跨所有命名空间的命名空间资源（如Pod）。

比如，授予对任何特定命名空间或所有命名空间中的secret的读权限（取决于它的绑定方式）：

kind: ClusterRole

apiVersion: rbac.authorization.k8s.io/v1

metadata:

# "namespace" omitted since ClusterRoles are not namespaced

name: secret-reader

rules:

- apiGroups: [""]

resources: ["secrets"]

verbs: ["get", "watch", "list"]

3. RoleBinding和ClusterRoleBinding

RoleBinding将Role中定义的权限授予User、Group或Service Account。RoleBinding和ClusterRoleBinding最大的区别与Role和ClusterRole的区别类似，即RoleBinding作用于命名空间，ClusterRoleBinding作用于集群。

RoleBinding可以引用同一命名空间的Role进行授权，比如将上述创建的pod-reader的Role授予default命名空间的用户jane，这将允许jane读取default命名空间中的Pod：

# This role binding allows "jane" to read pods in the "default" namespace.

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 #this must be Role or ClusterRole

name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to

apiGroup: rbac.authorization.k8s.io

说明：

* roleRef：绑定的类别，可以是Role或ClusterRole。

RoleBinding也可以引用ClusterRole来授予对命名空间资源的某些权限。管理员可以为整个集群定义一组公用的ClusterRole，然后在多个命名空间中重复使用。

比如，创建一个RoleBinding引用ClusterRole，授予dave用户读取development命名空间的Secret：

# This role binding allows "dave" to read secrets in the "development" namespace.

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

ClusterRoleBinding可用于在集群级别和所有命名空间中授予权限，比如允许组manager中的所有用户都能读取任何命名空间的Secret：

# This cluster role binding allows anyone in the "manager" group to read secrets in any namespace.

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: read-secrets-global

subjects:

- kind: Group

name: manager # Name is case sensitive

apiGroup: rbac.authorization.k8s.io

roleRef:

kind: ClusterRole

name: secret-reader

apiGroup: rbac.authorization.k8s.io

4. 对集群资源的权限控制

在Kubernetes中，大多数资源都由其名称的字符串表示，例如pods。但是一些Kubernetes API涉及的子资源（下级资源），例如Pod的日志，对应的Endpoint的URL是：

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

在这种情况下，pods是命名空间资源，log是Pod的下级资源，如果对其进行访问控制，要使用斜杠来分隔资源和子资源，比如定义一个Role允许读取Pod和Pod日志：

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"]

针对具体资源（使用resourceNames指定单个具体资源）的某些请求，也可以通过使用get、delete、update、patch等进行授权，比如，只能对一个叫my-configmap的configmap进行get和update操作：

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

namespace: default

name: configmap-updater

rules:

- apiGroups: [""]

resources: ["configmaps"]

resourceNames: ["my-configmap"]

verbs: ["update", "get"]

注 意

如果使用了resourceNames，则verbs不能是list、watch、create、deletecollection等。

5. 聚合ClusterRole

从Kubernetes 1.9版本开始，Kubernetes可以通过一组ClusterRole创建聚合ClusterRoles，聚合ClusterRoles的权限由控制器管理，并通过匹配ClusterRole的标签自动填充相对应的权限。

比如，匹配rbac.example.com/aggregate-to-monitoring: "true"标签来创建聚合ClusterRole：

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.

然后创建与标签选择器匹配的ClusterRole向聚合ClusterRole添加规则：

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"]

6. Role示例

以下示例允许读取核心API组中的资源Pods（只写了规则rules部分）：

rules:

- apiGroups: [""]

resources: ["pods"]

verbs: ["get", "list", "watch"]

允许在extensions和apps API组中读写deployments：

rules:

- apiGroups: ["extensions", "apps"]

resources: ["deployments"]

verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

允许对Pods的读和Job的读写：

rules:

- apiGroups: [""]

resources: ["pods"]

verbs: ["get", "list", "watch"]

- apiGroups: ["batch", "extensions"]

resources: ["jobs"]

verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

允许读取一个名为my-config的ConfigMap（必须绑定到一个RoleBinding来限制到一个命名空间下的ConfigMap）：

rules:

- apiGroups: [""]

resources: ["configmaps"]

resourceNames: ["my-config"]

verbs: ["get"]

允许读取核心组Node资源（Node属于集群级别的资源，必须放在ClusterRole中，并使用ClusterRoleBinding进行绑定）：

rules:

- apiGroups: [""]

resources: ["nodes"]

verbs: ["get", "list", "watch"]

允许对非资源端点/healthz和所有其子资源路径的Get和Post请求（必须放在ClusterRole并与ClusterRoleBinding进行绑定）：

rules:

- nonResourceURLs: ["/healthz", "/healthz/\*"] # '\*' in a nonResourceURL is a suffix glob match

verbs: ["get", "post"]

7. RoleBinding示例

以下示例绑定为名为“alice@example.com”的用户（只显示subjects部分）：

subjects:

- kind: User

name: "alice@example.com"

apiGroup: rbac.authorization.k8s.io

绑定为名为“frontend-admins”的组：

subjects:

- kind: Group

name: "frontend-admins"

apiGroup: rbac.authorization.k8s.io

绑定为kube-system命名空间中的默认Service Account：

subjects:

- kind: ServiceAccount

name: default

namespace: kube-system

绑定为qa命名空间中的所有Service Account：

subjects:

- kind: Group

name: system:serviceaccounts:qa

apiGroup: rbac.authorization.k8s.io

绑定所有Service Account：

subjects:

- kind: Group

name: system:serviceaccounts

apiGroup: rbac.authorization.k8s.io

绑定所有经过身份验证的用户（v1.5+）：

subjects:

- kind: Group

name: system:authenticated

apiGroup: rbac.authorization.k8s.io

绑定所有未经过身份验证的用户（v1.5+）：

subjects:

- kind: Group

name: system:unauthenticated

apiGroup: rbac.authorization.k8s.io

对于所有用户：

subjects:

- kind: Group

name: system:authenticated

apiGroup: rbac.authorization.k8s.io

- kind: Group

name: system:unauthenticated

apiGroup: rbac.authorization.k8s.io

8. 命令行的使用

权限的创建可以使用命令行直接创建，较上述方式更加简单、快捷，下面我们逐一介绍常用命令的使用。

（1）kubectl create role

创建一个Role，命名为pod-reader，允许用户在Pod上执行get、watch和list：

kubectl create role pod-reader --verb=get --verb=list --verb=watch --resource=pods

创建一个指定了resourceNames的Role，命名为pod-reader：

kubectl create role pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-name=anotherpod

创建一个命名为foo，并指定APIGroups的Role：

kubectl create role foo --verb=get,list,watch --resource=replicasets.apps

针对子资源创建一个名为foo的Role：

kubectl create role foo --verb=get,list,watch --resource=pods,pods/status

针对特定/具体资源创建一个名为my-component-lease-holder的Role：

kubectl create role my-component-lease-holder --verb=get,list,watch,update --resource=lease --resource-name=my-component

（2）kubectl create clusterrole

创建一个名为pod-reader的ClusterRole，允许用户在Pod上执行get、watch和list：

kubectl create clusterrole pod-reader --verb=get,list,watch --resource=pods

创建一个名为pod-reader的ClusterRole，并指定resourceName：

kubectl create clusterrole pod-reader --verb=get --resource=pods --resource-name=readablepod --resource-name=anotherpod

使用指定的apiGroup创建一个名为foo的ClusterRole：

kubectl create clusterrole foo --verb=get,list,watch --resource=replicasets.apps

使用子资源创建一个名为foo的ClusterRole：

kubectl create clusterrole foo --verb=get,list,watch --resource=pods,pods/status

使用non-ResourceURL创建一个名为foo的ClusterRole：

kubectl create clusterrole "foo" --verb=get --non-resource-url=/logs/\*

使用指定标签创建名为monitoring的聚合ClusterRole：

kubectl create clusterrole monitoring --aggregation-rule="rbac.example.com/aggregate-to-monitoring=true"

（3）kubectl create rolebinding

创建一个名为bob-admin-binding的RoleBinding，将名为admin的ClusterRole绑定到名为acme的命名空间中一个名为bob的user：

kubectl create rolebinding bob-admin-binding --clusterrole=admin --user=bob --namespace=acme

创建一个名为myapp-view-binding的RoleBinding，将名为view的ClusterRole，绑定到acme命名空间中名为myapp的ServiceAccount：

kubectl create rolebinding myapp-view-binding --clusterrole=view --serviceaccount=acme:myapp --namespace=acme

（4）kubectl create clusterrolebinding

创建一个名为root-cluster-admin-binding 的clusterrolebinding，将名为cluster-admin的ClusterRole绑定到名为root的user：

kubectl create clusterrolebinding root-cluster-admin-binding --clusterrole=cluster-admin --user=root

创建一个名为myapp-view-binding的clusterrolebinding，将名为view的ClusterRole绑定到acme命名空间中名为myapp的ServiceAccount：

kubectl create clusterrolebinding myapp-view-binding --clusterrole=view --serviceaccount=acme:myapp