链滴

备战 CKA 每日一题——第 11 天 | k8s 访问控制 RBAC、Role、RoleBinding，并引出 kubectl 常用命令

作者：liabio

原文链接：https://ld246.com/article/1583918242316

来源网站：链滴

昨日考题

创建一个Role(只有cka namespace下pods的所有操作权限)和RoleBinding(使用serviceaccount认鉴权),使用对应serviceaccount作为认证信息对cka namespace下的pod进行操作以及对default namspace下的pods进行操作。

– Role和RoleBinding的名称的名称为cka-1202-role、cka-1202-rb

注意：请附所用命令、创建的Role、RoleBinding以及serviceaccount的完整yaml，可分多次评论。

昨日答案

创建Service Account：

```
[root@liabio cka]# kubectl create serviceaccount cka-1202-sa -n cka  -o yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: "2019-12-02T23:37:42Z"
  name: cka-1202-sa
  namespace: cka
  resourceVersion: "15159020"
  selfLink: /api/v1/namespaces/cka/serviceaccounts/cka-1202-sa
  uid: 6764e90c-cb28-4de1-9109-6e3d56941fcb
```

创建Role：

```
[root@liabio cka]# kubectl create role  cka-1202-role -n cka  --verb=* --resource=pods -oyam

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: "2019-12-02T23:40:26Z"
  name: cka-1202-role
  namespace: cka
  resourceVersion: "15159247"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/cka/roles/cka-1202-role
  uid: fc2c5593-2fd9-46d7-a809-99bcee32249e
rules:
- apiGroups:
  - ""
  resources:
  - pods
  verbs:
  - '*'
```

创建RoleBinding：

```
[root@liabio cka]# kubectl create rolebinding cka-1202-rb -n cka  --role=cka-1202-role --serv
ceaccount=cka:cka-1202-sa  -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  creationTimestamp: "2019-12-02T23:46:50Z"
```

```
  name: cka-1202-rb
  namespace: cka
  resourceVersion: "15159794"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/cka/rolebindings/cka-1202-rb
  uid: c00d104e-a531-4781-90f4-2821651492bf
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: cka-1202-role
subjects:
- kind: ServiceAccount
  name: cka-1202-sa
  namespace: cka
```

验证：

获取到cka-1202-sa这个Service Account绑定的secret并base64 -d解码token字段：

```
[root@liabio ~]# kubectl get secret -n cka
NAME                  TYPE                                  DATA   AGE
cka-1202-sa-token-9rgp4   kubernetes.io/service-account-token   3      42m
default-token-r77xn       kubernetes.io/service-account-token   3      4d14h
[root@liabio ~]# kubectl get secret -n cka  cka-1202-sa-token-9rgp4 -ojson | jq .data.token
"ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklpSjkuZXlKcGMzTWlPaUpyZFdKbGNtNWx
R1Z6TDNObGNuWnBZMlZoWTJOdmRXNTBJaXdpYTNWaVpYSnVaWFJsY3k1cGJ5OXpaWEoy
VdObFlXTmpiM1Z1ZEM5dVlXMWxjM0JoWTJVaU9pSmphMkVpTENKcmRXSmxjbTVsZEdWekx
bHZMM05sY25acFkyVmhZMk52ZFc1MEwzTmxZM0psZEM1dVlXMWxJam9pWTJ0aExURXlNRE
0YzJFdGRHOXJaVzR0T1hKbmNEUWlMQ0pyZFdKbGNtNWxkR1Z6TG1sdkwzTmxjblpwWTJWa
kyTnZkVzUwTDNObGNuWnBZMlV0WVdOamIzVnVkQzV1WVcxbElqb2lZMnRoTFRFeU1ESXR
MkVpTENKcmRXSmxjbTVsZEdWekxtbHZMM05sY25acFkyVmhZMk52ZFc1MEwzTmxjblpwWT
VdFlXTmpiM1Z1ZEM1MWFXUWlPaUkyTnpZMFpUa3dZeTFqWWpJNExUUmtaVEV0T1RFd09
MDJaVE5rTlRZNU5ERm1ZMklpTENKemRXSWlPaUp6ZVhOMFpXMDZjMlZ5ZG1salpXRmpZMj
xYm5RNlkydGhPbU5yWVMweE1qQXlMWE5oSW4wLnFXanJUcTdEbVZTU01TM0h4YzR0bFd4
DdUNGtvUkNvVmkxMjVzZXNWRWJ2QUtEaTJ6MFhvNjJaNzAza2htQ1dsWTU1TkxPYWVKS2t
WXhYOWZMTEdYMnpPVWVFdzFvbUpmRkZpTm41NGxjOUhRTjlRXzVmTjRyYS1WNFZSaU5u
kFUeW43Yzc2aGk2Nks1aUh5WjB4bFRNcnBNQThXN1l2TmJnU1pIOXhnaFdSenpkSElKYWF1U
BTY0xtSk5MNmxGNGd5ZG9Xd0dDQy1QU0VjdGpKTkRtMF8zSTZoUkhEZkJzd3k2d0t4VGx4T3l
dE9yeUc0ckUzZzVqUWZOdV9BNTdTNVlocmEwWVM0emM0X0RvdXBmUC1zVjU3R0FQS1Jx
DZsRGdlOHo4cWFIaDRyb0k3RTNJbC1DRU9HS1JJeE52SWZVX3d0aHRrMG95aW5HR2wydw=
="
[root@liabio ~]# echo ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklpSjkuZXlKcGMzTWl
aUpyZFdKbGNtNWxkR1Z6TDNObGNuWnBZMlZoWTJOdmRXNTBJaXdpYTNWaVpYSnVaWFJ
Y3k1cGJ5OXpaWEoyYVdObFlXTmpiM1Z1ZEM5dVlXMWxjM0JoWTJVaU9pSmphMkVpTENKc
RXSmxjbTVsZEdWekxtbHZMM05sY25acFkyVmhZMk52ZFc1MEwzTmxZM0psZEM1dVlXMWxJ
m9pWTJ0aExURXlNREl0YzJFdGRHOXJaVzR0T1hKbmNEUWlMQ0pyZFdKbGNtNWxkR1Z6TG1
dkwzTmxjblpwWTJWaFkyTnZkVzUwTDNObGNuWnBZMlV0WVdOamIzVnVkQzV1WVcxbElqb2
ZMnRoTFRFeU1ESXRjMkVpTENKcmRXSmxjbTVsZEdWekxtbHZMM05sY25acFkyVmhZMk52ZF
1MEwzTmxjblpwWTJVdFlXTmpiM1Z1ZEM1MWFXUWlPaUkyTnpZMFpUa3dZeTFqWWpJNExU
mtaVEV0T1RFd09TMDJaVE5rTlRZNU5ERm1ZMklpTENKemRXSWlPaUp6ZVhOMFpXMDZjMlZ
ZG1salpXRmpZMjkxYm5RNlkydGhPbU5yWVMweE1qQXlMWE5oSW4wLnFXanJUcTdEbVZTU0
TM0h4YzR0bFd4ODdUNGtvUkNvVmkxMjVzZXNWRWJ2QUtEaTJ6MFhvNjJaNzAza2htQ1dsW
U1TkxPYWVKS2taWXhYOWZMTEdYMnpPVWVFdzFvbUpmRkZpTm41NGxjOUhRTjlRXzVmTjR
YS1WNFZSaU5uQkFUeW43Yzc2aGk2Nks1aUh5WjB4bFRNcnBNQThXN1l2TmJnU1pIOXhnaFd
enpkSElKYWF1UXBTY0xtSk5MNmxGNGd5ZG9Xd0dDQy1QU0VjdGpKTkRtMF8zSTZoUkhEZkJ
```

```
d3k2d0t4VGx4T3lIdE9yeUc0ckUzZzVqUWZOdV9BNTdTNVlocmEwWVM0emM0X0RvdXBmUC
zVjU3R0FQS1JxODZsRGdlOHo4cWFIaDRyb0k3RTNJbC1DRU9HS1JJeE52SWZVX3d0aHRrMG9
aW5HR2wydw== | base64 -d
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia
ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJja2EiLCJrdWJlcm5ldGVzLmlv
3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoiY2thLTEyMDItc2EtdG9rZW4tOXJncDQiLCJrdWJ
cm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiY2thLTEyMDIt
2EiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI2
zY0ZTkwYy1jYjI4LTRkZTEtOTEwOS02ZTNkNTY5NDFmY2IiLCJzdWIiOiJzeXN0ZW06c2VydmljZ
FjY291bnQ6Y2thOmNrYS0xMjAyLXNhIn0.qWjrTq7DmVSSMS3Hxc4tlWx87T4koRCoVi125sesV
bvAKDi2z0Xo62Z703khmCWlY55NLOaeJKkZYxX9fLLGX2zOUeEw1omJfFFiNn54lc9HQN9Q_5f
4ra-V4VRiNnBATyn7c76hi66K5iHyZ0xlTMrpMA8W7YvNbgSZH9xghWRzzdHIJaauQpScLmJNL6
F4gydoWwGCC-PSEctjJNDm0_3I6hRHDfBswy6wKxTlxOyHtOryG4rE3g5jQfNu_A57S5Yhra0YS4
c4_DoupfP-sV57GAPKRq86lDge8z8qaHh4roI7E3Il-CEOGKRIxNvIfU_wthtk0oyinGGI2w[root@li
bio ~]#
```

把解码后的信息添加到将添加到~/.kube/config中，注意到下面加了name为coderaction的context name为coderaction的user

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDLQo=
    server: https://10.0.0.0:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: coderaction
  name: coderaction
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: coderaction
  user:
    token: eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2Nv
W50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJja2EiLCJrdWJlcm5l
GVzLmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoiY2thLTEyMDItc2EtdG9rZW4tOXJncD
iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiY2th
TEyMDItc2EiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51
WQiOiI2NzY0ZTkwYy1jYjI4LTRkZTEtOTEwOS02ZTNkNTY5NDFmY2IiLCJzdWIiOiJzeXN0ZW06c
VydmljZWFjY291bnQ6Y2thOmNrYS0xMjAyLXNhIn0.qWjrTq7DmVSSMS3Hxc4tlWx87T4koRC
Vi125sesVEbvAKDi2z0Xo62Z703khmCWlY55NLOaeJKkZYxX9fLLGX2zOUeEw1omJfFFiNn54lc9
QN9Q_5fN4ra-V4VRiNnBATyn7c76hi66K5iHyZ0xlTMrpMA8W7YvNbgSZH9xghWRzzdHIJaau
pScLmJNL6lF4gydoWwGCC-PSEctjJNDm0_3I6hRHDfBswy6wKxTlxOyHtOryG4rE3g5jQfNu_A57
5Yhra0YS4zc4_DoupfP-sV57GAPKRq86lDge8z8qaHh4roI7E3Il-CEOGKRIxNvIfU_wthtk0oyinGG
2w
- name: kubernetes-admin
  user:
```

```
    client-certificate-data: LS0tLS1CRUdJTiB1M1Y2NDTnpPUT0KLS0tLS1FTkQgQ0VSVElGSUNBEUtLS0tLQo=
    client-key-data: LS0tLS1CBS0NBUUVBdjNpTkx5eUEwaVdmOU1hUjA3cVFTOEtFWS0tLS0tC==
```

通过切换到coderaction这个use-context可以发现，get默认分区下的Pod时提示system:serviceaccount:cka:cka-1202-sa没有权限，但可以正常获取cka namespace下的Pods

```
[root@liabio cka]# kubectl config use-context kubernetes-admin@kubernetes
Switched to context "kubernetes-admin@kubernetes".
[root@liabio cka]# kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
cka-1128-01-7b8b8cb79-mll6d   1/1     Running   118        32h
[root@liabio cka]#
[root@liabio cka]#
[root@liabio cka]# kubectl get node
NAME     STATUS   ROLES    AGE    VERSION
liabio   Ready    master   141d   v1.15.2
[root@liabio cka]# kubectl config use-context coderaction
Switched to context "coderaction".
[root@liabio cka]# kubectl get pod
Error from server (Forbidden): pods is forbidden: User "system:serviceaccount:cka:cka-1202-sa
 cannot list resource "pods" in API group "" in the namespace "default"
[root@liabio cka]# kubectl get pod -n cka
No resources found.
```

昨日解析

k8s对于访问 API 来说提供了两个步骤的安全措施：认证和授权。认证解决用户是谁的问题，授权解用户能做什么的问题。通过合理的权限管理，能够保证系统的安全可靠。

k8s集群的所有操作基本上都是通过kube-apiserver这个组件进行的，它提供HTTP RESTful形式的AP供集群内外客户端调用。需要注意的是：认证授权过程只存在HTTPS形式的API中。也就是说，如果户端使用HTTP连接到kube-apiserver，那么是不会进行认证授权的。所以说，可以这么设置，在集内部组件间通信使用HTTP，集群外部就使用HTTPS，这样既增加了安全性，也不至于太复杂。

本题主要是考察授权：基于角色的访问控制（RBAC）的考题。

RBAC官方文档：

https://kubernetes.io/docs/reference/access-authn-authz/rbac/

创建RoleBinding 、Role、Service Account官网命令指导：

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#-em-rolebindin-em-

使用 kubeconfig 文件组织集群访问：

https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/

context相关操作官方命令指南：

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#config

基于角色的访问控制（RBAC）是一种基于企业内各个用户的角色来调节对计算机或网络资源的访问方法。

RBAC使用rbac.authorization.k8s.io API组 驱动授权决策，使管理员可以通过Kubernetes API动态置策略。

从1.8开始，RBAC模式是稳定的，并由rbac.authorization.k8s.io/v1 API提供支持。

要启用RBAC，请通过启动apiserver --authorization-mode=RBAC

RBAC API声明了四个顶级类型：

Role和ClusterRole

在RBAC API中，Role包含代表一组权限的规则。权限纯粹是累加的（没有“拒绝”规则）。可以在namespace中用Role或在集群范围内用ClusterRole。

Role只能用于授予对单个名称空间内资源的访问权限。

ClusterRole由于它们是集群范围的，因此它们还可以用于授予以下权限：

- 集群范围内的资源（如节点）
- 非资源端点（例如“ /healthz”）
- 所有namespace中的命名空间资源（例如pod）

RoleBinding和ClusterRoleBinding

RoleBinding向一个或一组用户授予在Role中定义的权限。它包含subjects（User，Group或Service ccount），以及对所授予角色的引用。可以在namespace中使用RoleBinding或在集群范围内使用ClusterRoleBinding。

RoleBinding可以引用同一namespace下的Role。

roleRef是实际创建绑定的方式。该kind可以是Role或ClusterRole，并且name将引用具体名字的Rol或ClusterRole

ClusterRoleBinding可以在集群级别和所有namespace中授予权限。

创建Role命令：

kubectl create role NAME --verb=verb --resource=resource.group/subresource [--resource-name=resourcename] [--dry-run]

--verb指定，对资源的操作动作集合，包括get、delete、update、create、patch、watch、list，有操作动作为*

--resource指定可操作资源类型集合；

--resource-name指定可操作资源名称集合；

如：

[root@liabio ~]# kubectl create role pod-reader-cka -n cka --verb=get --verb=list --resourc

```
=pods --resource-name=readablepod --resource-name=anotherpod -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: "2019-12-03T03:50:34Z"
  name: pod-reader-cka
  namespace: cka
  resourceVersion: "15179947"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/cka/roles/pod-reader-cka
  uid: 16742721-4890-43de-9725-d6c721c6e4cf
rules:
- apiGroups:
  - ""
  resourceNames:
  - readablepod
  - anotherpod
  resources:
  - pods
  verbs:
  - get
  - list
```

创建RoleBinding

```
kubectl create rolebinding NAME --clusterrole=NAME|--role=NAME [--user=username] [--gr
up=groupname] [--serviceaccount=namespace:serviceaccountname] [--dry-run]
```

--role指定RoleBinding的roleRef中的Role名称；

--clusterrole指定RoleBinding的roleRef中的ClusterRole名称；

--serviceaccount指定RoleBinding的subjects集合；

--user指定RoleBinding的subjects下User的名称；

如：

```
[root@liabio ~]# kubectl create rolebinding admin-cka -n cka --clusterrole=admin --user=use
1 --user=user2 --group=group1 -oyaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  creationTimestamp: "2019-12-03T03:47:55Z"
  name: admin-cka
  namespace: cka
  resourceVersion: "15179732"
  selfLink: /apis/rbac.authorization.k8s.io/v1/namespaces/cka/rolebindings/admin-cka
  uid: 4d4eacfb-3ba0-4fa1-96c3-c624fbafb12c
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: user1
```

```
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: user2
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: group1
```

创建ServiceAccount

kubectl create serviceaccount NAME [--dry-run]

如：

```
[root@liabio cka]# kubectl create serviceaccount cka-1202-sa -n cka  -o yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: "2019-12-02T23:37:42Z"
  name: cka-1202-sa
  namespace: cka
  resourceVersion: "15159020"
  selfLink: /api/v1/namespaces/cka/serviceaccounts/cka-1202-sa
  uid: 6764e90c-cb28-4de1-9109-6e3d56941fcb
```

今日考题

创建两个deployment名字分别为cka-1203-01、cka-1203-02；

cka-1203-01的Pod加label：cka：cka-1203-01；

cka-1203-02的Pod加label：cka：cka-1203-02；

请用利用kubectl命令label选择器查出这两个deployment，并按照创建时间排序。

例如：

```
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
cka-1203-01   1/1     1            1           8m40s
cka-1203-02   1/1     1            1           8m38
```

作者简介

作者：小碗汤，一位热爱、认真写作的小伙，目前维护原创公众号：『我的小碗汤』，专注于写linux golang、docker、kubernetes等知识等提升硬实力的文章，期待你的关注。转载说明：务必注明来（注明：来源于公众号：我的小碗汤， 作者：小碗汤）