7주차 쿠버네티스 RBAC

RBAC이란

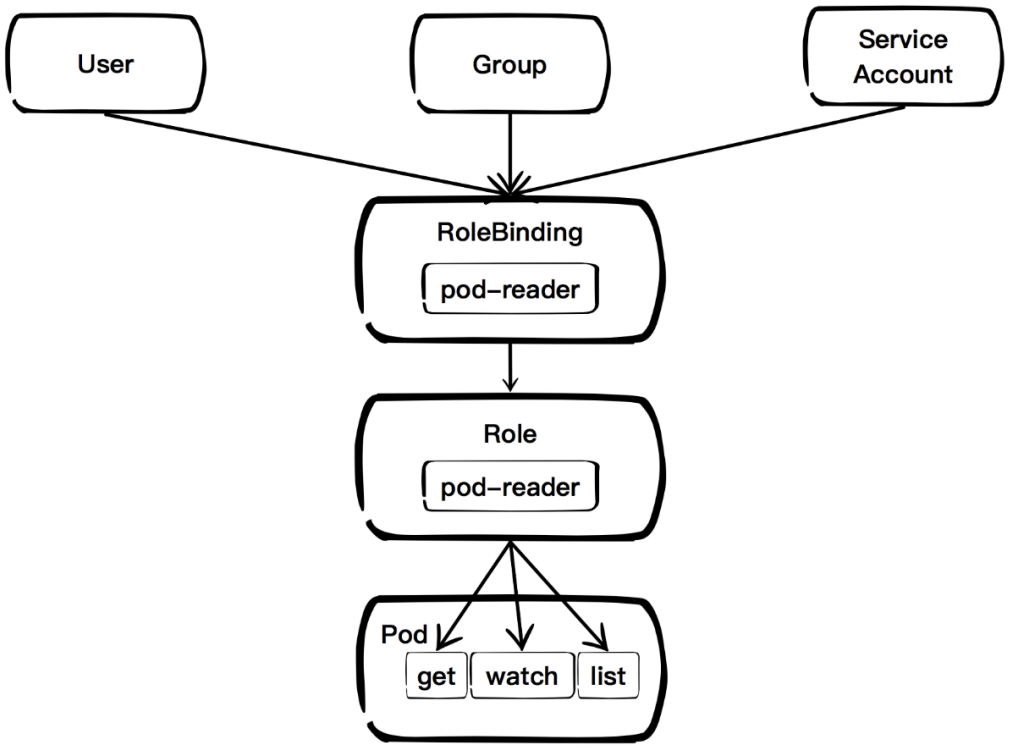
역할 기반 접근 제어(role-based access control, RBAC)는 컴퓨터 시스템 보안에서 권한이 있는 사용자들에게 시스템 접근을 통제하는 한 방법이다. [출처:위키백과]

쿠버네티스에서 RBAC은 일반 사용자, 서비스 계정에 권한을 부여한 롤에 따라 리소스에 대한 권한을 제어하는 기능을 제공한다. RBAC을 적절히 사용해 쿠버네티스 리소스의 보안을 확보할 수 있다.

다른 액세스 제어 방법과 비교하여 다음과 같은 장점이 있습니다.

- 클러스터에서 리소스 및 비리소스 권한을 모두 포함합니다.
- 전체 RBAC은 여러 API 객체에 의해 수행되며 다른 API 객체와 마찬가지로 kubectl 또는 API로 조작 할 수 있습니다.
- API Server를 다시 시작하지 않고도 런타임에 조정할 수 있습니다.

User & Role & RoleBinding



[출처 : wiki.shileizcc.com]

1. USER

사용자 개념.

- User
- Group
- ServiceAccount

일반 사용자

- 클러스터 외부에서 쿠버네티스를 조작하는 사용자

```
kubectl config view | grep current-context
```

서비스 계정

- 쿠버네티스 내부적으로 관리되며 파드가 쿠버네티스 API를 다룰 때 사용하는 사용자

yaml format

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: myuser
  namespace: default
```

2. Role

특정 api나 리소스에 대한 권한들을 명시해둔 규칙들의 집합

Verb

Verb	의미
create	새로운 리소스 생성
get	개별 리소스 조회
list	여러건의 리소스 조회
update	기존 리소스내용 전체 업데이트
patch	기존 리소스중 일부 내용 변경
delete	개별 리소스 삭제
deletecollection	여러 리소스 삭제

Role

- 속한 네임스페이스에 한 곳에만 적용

yaml format

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: read-role
rules:
- apiGroups: [""]
  resources: ["pods"]
  resourceNames: [“mypod"]
  verbs: ["get", "list"]
```

ClusterRole

- 클러스터 전체 네임스페이스에 적용

yaml format

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-clusterrole
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list"]
```

- Grouping 가능

yaml format

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: admin-aggregation
aggregationRule:
  clusterRoleSelectors:
  - matchLabels:
      kubernetes.io/bootstrapping: rbac-defaults
rules: []
```

3. RoleBinding

롤과 사용자를 묶어(binding)주는 역할

RoleBinding

- 속한 네임스페이스에 한 곳에만 적용

yaml format

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-rolebinding
  namespace: default
subjects:
- kind: ServiceAccount
  name: myuser
  apiGroup: ""
roleRef:
  kind: Role
  name: read-role
  apiGroup: rbac.authorization.k8s.io
```

ClusterRoleBinding

- 클러스터 전체 네임스페이스에 적용

yaml format

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-clusterrolebinding
subjects:
- kind: ServiceAccount
  name: myuser
  namespace: default
  apiGroup: ""
roleRef:
  kind: ClusterRole
  name: read-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

실습

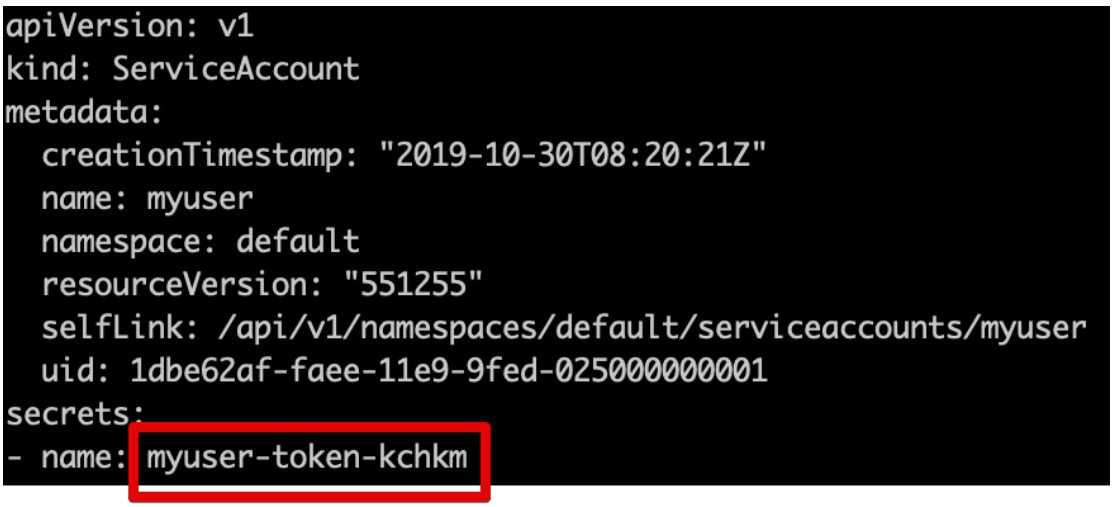
일반 사용자

1. 사용자 생성

```
> kubectl create serviceaccount myuser
```

생성 확인

```
> kubectl get serviceaccount myuser -o yaml
```



토큰 얻기

시크릿 조회

```
> kubectl get secret myuser-token-kchkm -o yaml
```

```
apiVersion: v1
data:
  ca.crt: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUN5RENDQWJDZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZBREFWTVJNd0VRWURWU
WFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTVZsClkyaTJiTmhpclF4RlF1MXRVSk05QkcvWW9WV213c1dOWm5rbVc
tyeQpIcDFPaHBFeXZhd0VDK1lYVENFK2lWb042S2pNME1EYnhaUHAra2NlaDczSEljanNHYTlDWHBlYys0dzI0RC9KCldxUjA0SnR2UnAvdE9HcitnajNIC
OHNNUngvd0dqcDZwYWg0YWMrSgprWjhZMXk5MW4xVzJZVEpWVSswQ0F3RUFBYU1qTUNFd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZ
tNSXRqbHN5UHYxdlVUT2hHWHYrWlU2VW84NEJFdjczV2NlbwpQaUZHY3g4cTYwTkNHY2RBVHcvN1BhWTRya0dhcGEwakhsNnhMeDA2SHRUamRxdmkxQXVzb
SFZIUXJjVkZZVWhtRnkxbnRqVzBma2FHNmtON3VMaVk3OGFwZk44VlJ5bDlqQ3huOCtRdQp5OU1IelE0eVcvSEdRUDJIbTYvWTFaVzVYMDh5WFBLeW10dzg
  namespace: ZGVmYXVsdA==
  token: ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklpSjkuZXlKcGMzTWlPaUpyZFdKbGNtNWxkR1Z6TDNObGNuWnBZMlZoWTJOdmRXNTBJaXdpYT
jJnv0SswVdOamIzVnVkQzl6WldOeVpYUXVibUZ0WlNJNkltMTVkWE5sY2kxMGIydGxiaTFyWTJocmJTSXNJbXQxWW1WeWJtVjBaWE11YVc4dmMyVnlkbWx
XNTBMM05sY25acFkyVXRZV05qYjNWdWRDNTFhV1FpT2lJeFpHSmxOakpoZi1wZVmxMVEV4WlRrdE9XWmxaQzB3TWpVd01EQXdNREF3TURFaUxDSnpkV0
3U5akdoeXZVbnpJd01OWFVkYVJOMTc2MWx0TEozdXdpQ0hLNWxReE45QW45N1E5RG1neWZXMUZwVVJ6VXJjc0lPZmkxRFFvdjI3ZnZpcmkzdVZsWjlJUEdz
ISm1CUnZtYlFTYWl3eGZMQXZ5TFFIYWdyY3VBSTJIb1RQd215YXo4VXcwWDhLbEwxeGRfd2dCQ0liNkprNHpGclRzNzZ6ZjhsdkJ2NUhESTN0c0FDcHJadr
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: myuser
    kubernetes.io/service-account.uid: 1dbe62af-faee-11e9-9fed-025000000001
  creationTimestamp: "2019-10-30T08:20:22Z"
  name: myuser-token-kchkm
  namespace: default
  resourceVersion: "551254"
  selfLink: /api/v1/namespaces/default/secrets/myuser-token-kchkm
  uid: 1dd3e17d-faee-11e9-9fed-025000000001
type: kubernetes.io/service-account-token
```

Base64 디코딩

```
echo 'ZXlKaGJHY2lPaUpTVXpJ..' | base64 -D
```

토큰 얻기 2

```
kubectl -n default describe secret $(kubectl -n default get secret | grep myuser | awk '{print $1}')
```

```
Name:         myuser-token-kchkm
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: myuser
              kubernetes.io/service-account.uid: 1dbe62af-faee-11e9-9fed-025000000001

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1025 bytes
namespace:  7 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2Ui
2hrbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJteXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3
jY291bnQ6ZGVmYXVsdDpteXVzZXIifQ.GkEFJHos9htPZu5GtmSKo5lmPFHrqMceCu9jGhyvUnzIwMNXUdaRN1761ltLJ3uwiCHK5lQxN9An97Q9DmgyfW1FpURzUrcsIOfi1DQov27fviri3
mi4CTHJmBRvmbQSaiwxfLAvyLQHagrcuAI2HoTPwmyaz8Uw0X8KlL1xd_wgBCIb6Jk4zFrTs76zf8lvBv5HDI3tsACprZvs8tKUBPabDLZxz6cVDAeW8rgRL1wmbdQveYmceCnEA
```

2. Role 생성

ClusterRole

read-clusterrole.yaml

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-clusterrole
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list"]
```

```
kubectl apply -f read-clusterrole.yaml
```

3. RoleBinding 생성

read-clusterrolebinding.yaml

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: read-clusterrolebinding
subjects:
- kind: ServiceAccount
  name: myuser
  namespace: default
  apiGroup: ""
roleRef:
  kind: ClusterRole
  name: read-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

```
kubectl apply -f read-clusterrolebinding.yaml
```

4. 사용자 전환

사용자 인증 토큰 설정

```
kubectl config set-credentials myuser --token=eyJhbGciOiJSUzI1NiIsImt...
```

컨텍스트 생성

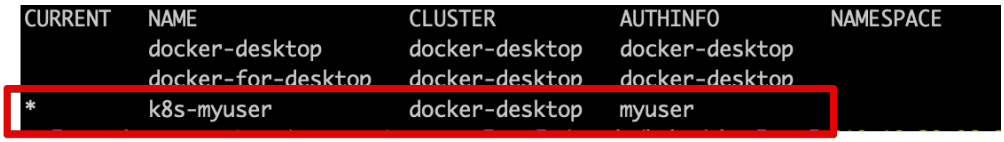
```
kubectl config set-context k8s-myuser --cluster=docker-desktop --user=myuser
```

컨텍스트 전환

```
kubectl config use-context k8s-myuser
```

확인

```
kubectl config get-contexts
```



5. 권한 테스트

myuser 가 가지고 있는 권한을 사용할 수 있다.

```
kubectl get pod --all-namespaces
```

myuser 가 가지고 있지 않은 권한에서는 아래 에러를 뱉어야 하는데

```
kubectl get deployment --all-namespaces
```

에러를 뱉지 않는다.

도커로 설치한 쿠버네티스에는 모든 serviceaccount에게 관리자 권한을 부여하는 docker-for-desktop-binding이라는 클러스터롤바인딩이 기본값으로 설정되어 있다.

```
kubectl describe clusterrolebinding docker-for-desktop-binding
```

그래서 cluster-admin 을 수정하여 테스트해본다.

admin 사용자로 전환

```
kubectl config use-context docker-desktop
```

cluster-admin 수정

```
kubectl edit clusterrole cluster-admin
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: "2019-10-16T07:58:57Z"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: cluster-admin
  resourceVersion: "555762"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterroles/cluster-admin
  uid: ce809d5a-efea-11e9-8a8c-025000000001
rules:
- apiGroups:
  - '*'
  resources:
  - pod
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

myuser 사용자 전환

```
kubectl config use-context myuser
```

아래 다시 수행시 권한이 없기 때문에 에러를 뱉는다.

```
kubectl get deployment --all-namespaces
```

```
Error from server (Forbidden): deployments.extensions is forbidden: User "system:serviceaccount:default:myuser" cannot list resource "deployments" in API group "extensions" at the cluster scope
```