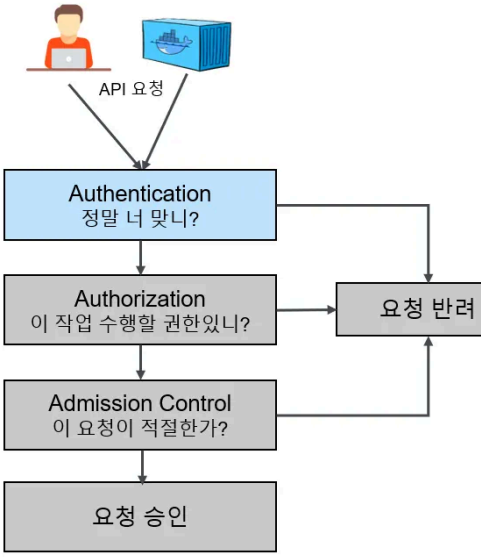
Role Based Access Control(RBAC)

1. API 인증

- Role-based access control(RBAC. 역할 기반 액세스 제어)
  - 사용자의 역할에 따라 리소스에 대한 접근 권한을 가짐

- API 인증

- API 서버에 접근하기 위해서는 인증작업 필요
  - 일반사용자(Human User) 또는 그룹(Group)
    클러스터 외부에서 쿠버네티스를 조작하는 사용자로, 다양한 방법으로 인증을 거친다.
  - 서비스계정(Service Account)
    쿠버네티스 내부적으로 관리되며 Pod가 쿠버네티스 API를 다룰 때 사용하는 계정이다.



- Service Account
  - 동작되는 모든 Pod는 ServiceAccount를 가지고 있음
    - ServiceAccount name: **default** , ServiceAccount Token :

```
kubectl run testpod --image nginx kubectl get pod testpod -o yaml apiVersion: v1 kind: Pod metadata: name: testpod spec: containers: - image: nginx .... serviceAccount: default serviceAccountName: default .. volumeMounts: - mountPath: /var/run/secrets/kubernetes.io/serviceaccount # ServiceAccount default가 사용하는 토큰 name: kube-api-access-497xn
```

  - Service Account 생성

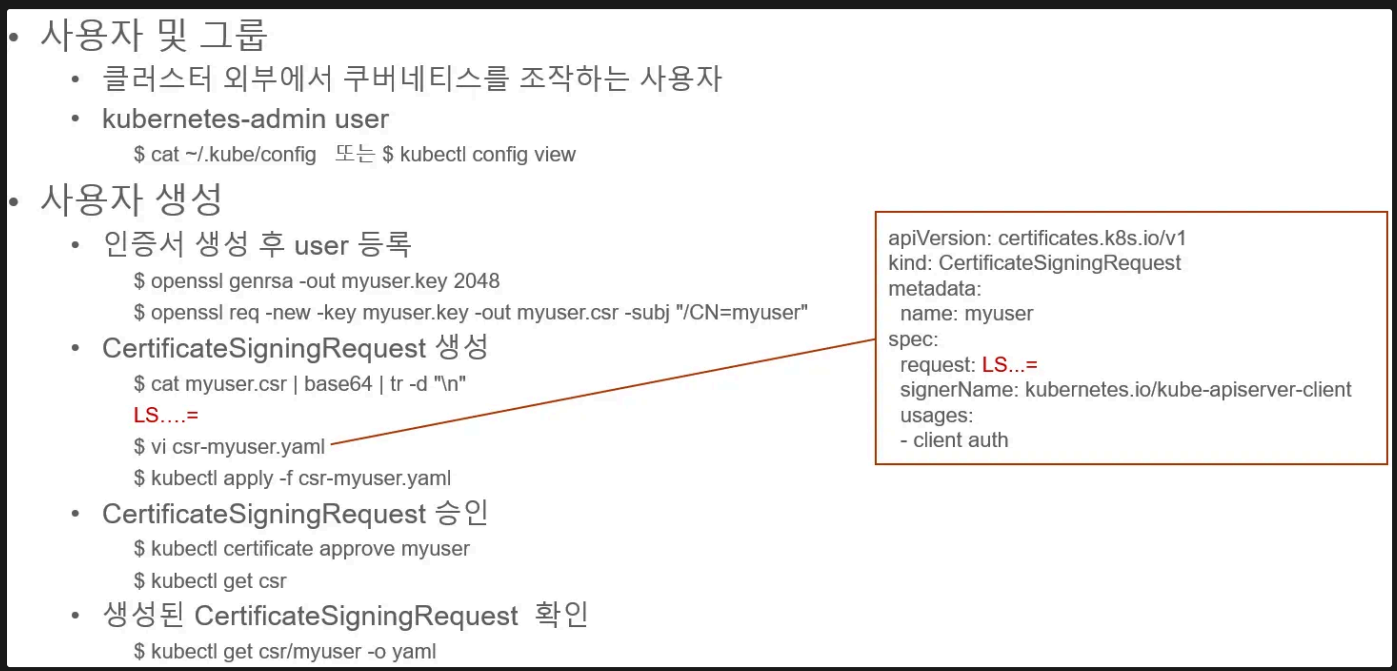
```
kubectl create serviceaccount monitor kubectl get serviceaccounts NAME SECRETS AGE default 0 68d monitor 0 13s
```

  - Pod를 실행 시 ServiceAccount 할당

```
cat << END > pod-sa.yaml apiVersion: v1 kind: Pod metadata: name: testpod namespace: default spec: containers: - image: nginx name: testpod serviceAccount: monitor serviceAccountName: monitor END
kubectl apply -f pod-sa.yaml kubectl get pod testpod -o yaml | grep -i Service
```

  ▶ 실습 : 참고

- User

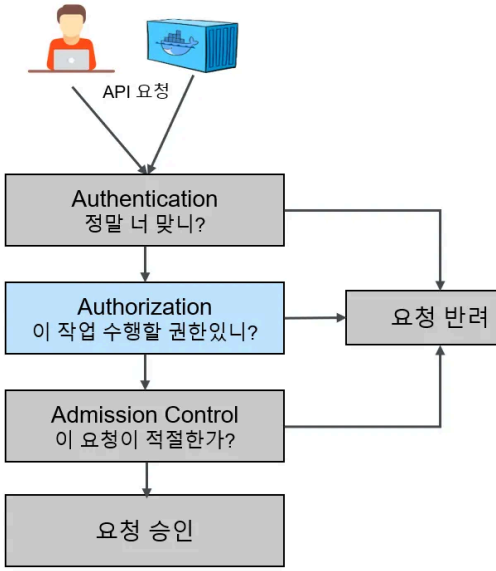


2. API 권한

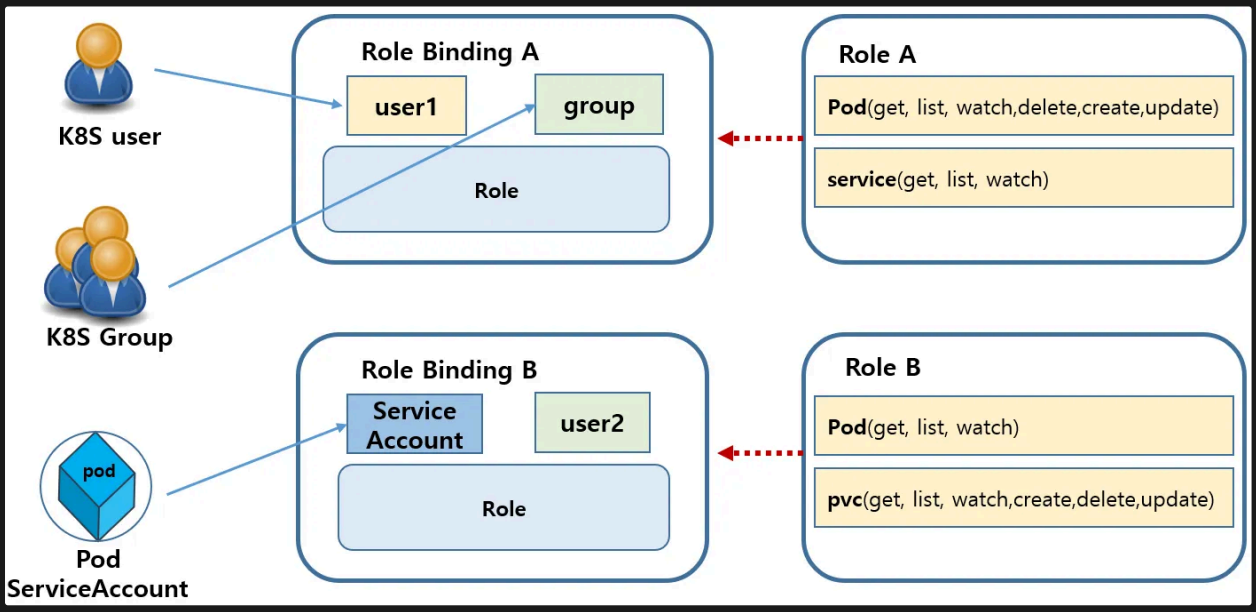
API 권한

- 권한

- 특정 유저나 ServiceAccount가 접근하려는 API에 접근 권한을 설정.
- 권한 있는 User만 접근하도록 허용
- 권한제어
- Role
  - 어떤 API를 이용할 수 있는지의 정의
  - 쿠버네티스의 사용권한을 정의
  - 지정된 네임스페이스에서만 유효
- RoleBinding
  - 사용자/그룹 또는 Service Account와 role을 연결



- Role & RoleBinding



- Role
  - 어떤 API를 사용할 수 있는지 권한정의. 지정된 네임스페이스에서만 유효
- Role 예제 : default 네임스페이스의 Pod에 대한 get, watch, list 할 수 있는 권한

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

verb	의미
create	새로운 리소스 생성
get	개별 리소스 조회
list	여러 건의 리소스 조회
update	기존 리소스 내용 전체 업데이트
patch	기존 리소스 중 일부 내용 변경
delete	개별 리소스 삭제
deletecollection	여러 리소스 삭제

  - pod는 코어 API이기 때문에 apiGroups를 따로 지정하지 않는다. 만약 리소스가 job이라면 apiGroups에 "batch"를 지
  - resources에는 pods, deployments, services 등 사용할 API resource들을 명시한다.
  - verbs에는 단어 그대로 나열된 API 리소스에 허용할 기능을 나열한다.

- RoleBinding
  - 사용자/그룹 또는 Service Account와 role을 연결
- RoleBinding 예제 : default 네임스페이스에서 유저 jane 에게 pod-reader의 role을 할당

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

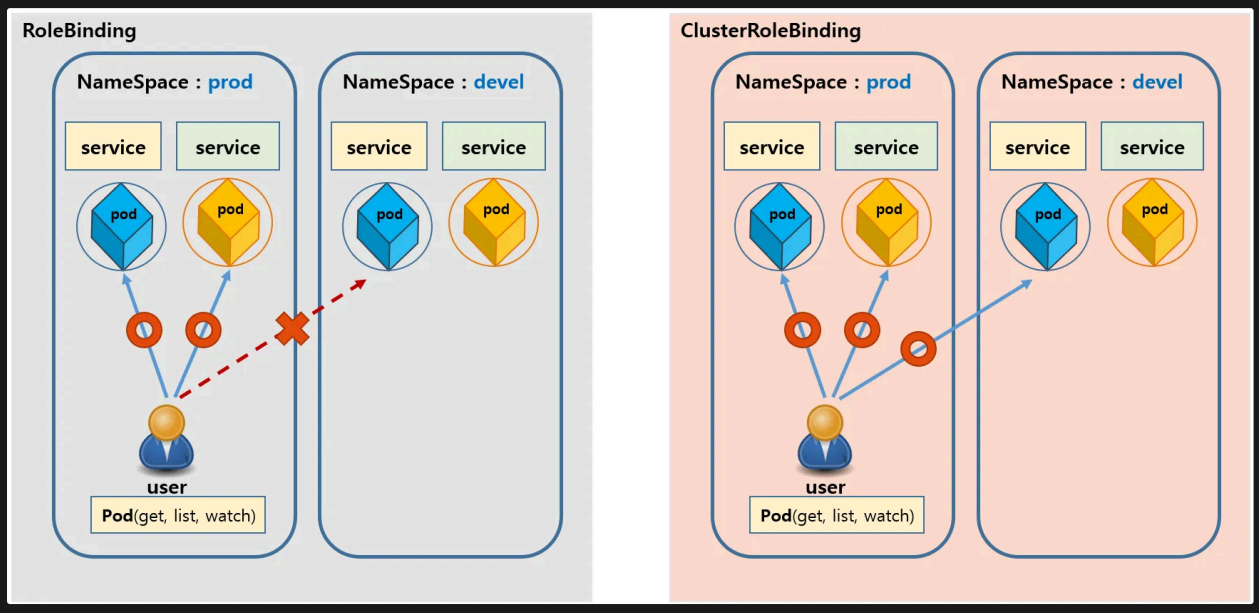
→ 유저 jane 에게 앞서 정의한 pod-reader의 role을 할당

serviceAccount 적용 예:

```
subjects:
- kind: ServiceAccount
  name: podman
  apiGroup: rbac.authorization.k8s.io
```

→ 앞서 정의한 Role을 명세
apiGroup에는 rbac api를 사용하기 때문에 rbac.authorization.k8s.io로 설정

- ClusterRole & ClusterRoleBinding



- ClusterRole
  - 어떤 API를 사용할 수 있는지 권한 정의. 클러스터 전체(전체 네임스페이스)에서 유효
  - ClusterRole 예제 : 전체 네임스페이스의 Secret에 대한 get, watch, list 할 수 있는 권한

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

- ClusterRoleBinding
  - 사용자/그룹 또는 Service Account와 role을 연결
  - ClusterRoleBinding 예제 : manager 그룹의 모든 사용자가 모든 네임스페이스의 secret을 읽을 수 있도록 구성

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

subjects → manager 그룹

roleRef → ClusterRole

3. API 권한 설정 실습

▸ 3-1 User 생성 후 role, roleBinding 구성

3-2. ServiceAccount 생성 후 role, roleBinding/clusterrole, clusterRoleBinding 구성

1. ServiceAccount 생성
   - msa 네임스페이스에서 운영되는 ServiceAccount monitor를 생성

```
# serviceAccount 생성(monitor) or user계정 생성(myuser) kubectl create namespace msa kubectl create serviceaccount monitor --namespace msa # servieAccount 확인 # servieAccount 생성시 monitor(API에 TOKEN등록) 생성됨. kubectl get sa -n msa
```

2-1. Role/RoleBinding

- role
  - name : app-reader
  - 권한 : pods,services,deployments에 대해서 get, list, watch
  - List
  - namespace: msa

```
# Role: app-reader kubectl create role app-reader --verb=get --verb=list --verb=watch --resource=pods,deployments,services -n msa kubectl get role -n msa kubectl describe role app-reader -n msa Name: app-reader Labels: <none> Annotations: <none> PolicyRule: Resources Non-Resource URLs Resource Names Verbs --------- ----------------- -------------- ----- pods [] [] [get list watch] services [] [] [get list watch] deployments.apps [] [] [get list watch]
```

- RoleBinding
  - name: app-rolebind-monitor
  - namespace: msa
  - serviceaccount : monitor
  - role: app-reader

```
# RoleBinding : app-rolebind-monitor kubectl create rolebinding app-rolebind-monitor --role=app-reader --serviceaccount=msa:monitor -n msa kubectl get rolebindings -n msa NAME ROLE AGE app-rolebind-monitor Role/app-reader 70s kubectl describe rolebindings -n msa app-rolebind-monitor Name: app-rolebind-monitor Labels: <none> Annotations: <none> Role: Kind: Role Name: app-reader Subjects: Kind Name Namespace ---- ---- --------- ServiceAccount monitor msa
```

2-2. ClusterRole/ClusterRoleBinding 구성

- ClusterRole
  - name : app-reader-cr
  - 권한 : pods,services,deployments에 대해서 get, list, watch
  - namespace: msa

```
# ClusterRole : app-reader-cr # clusetrole은 namespace 붙여서 적용하지않음. 왜? 전체클러스터 대상이므로-n msa kubectl create clusterrole app-reader-cr --verb=get,list,watch --resource=pods,services,deployments
```

- ClusterRoleBinding
  - name: app-clusterrolebind-monitor
  - namespace: msa
  - serviceaccount : monitor
  - clusterRole: app-reader-cr

```
# ClusterRoleBinding # clusetrole은 namespace 붙여서 적용하지않음. 왜? 전체클러스터 대상이므로 -n msa는 생략함.
kubectl create clusterrolebinding app-clusterrolebind-monitor --clusterrole=app-reader-cr --serviceaccount=msa:monitor
```

3. 특정 ServiceAccount를 사용하는 Pod 운영