그림으로 배우는 쿠버네티스 (Kubernetes)

조 훈 (Hoon Jo)
- CCIE DC, CKA&D, VCIX-NV6, RHCE, GCPx4

https://github.com/SysNet4Admin

https://app.vagrantup.com/SysNet4Admin

kubernetes

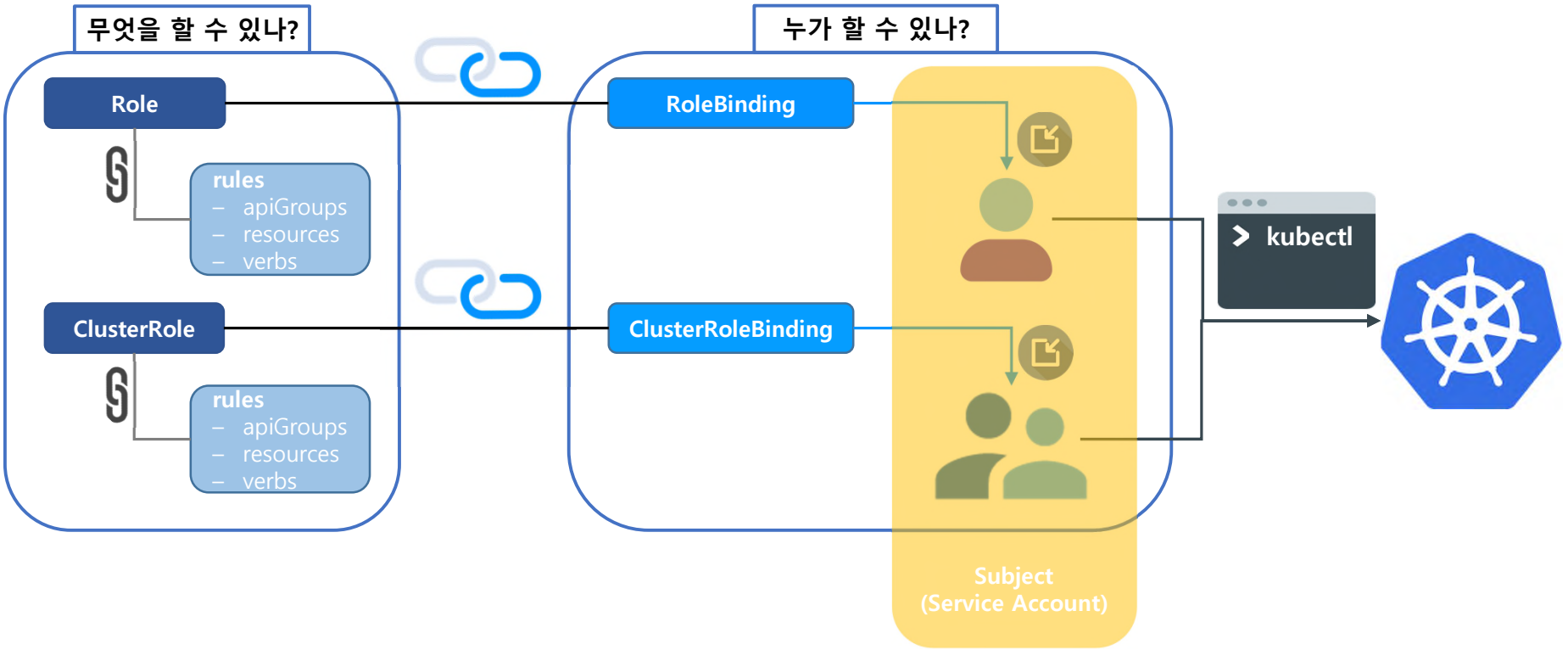
역할 기반 접근 제어(RBAC)

현재 적용된 인가(Authorization) 설정

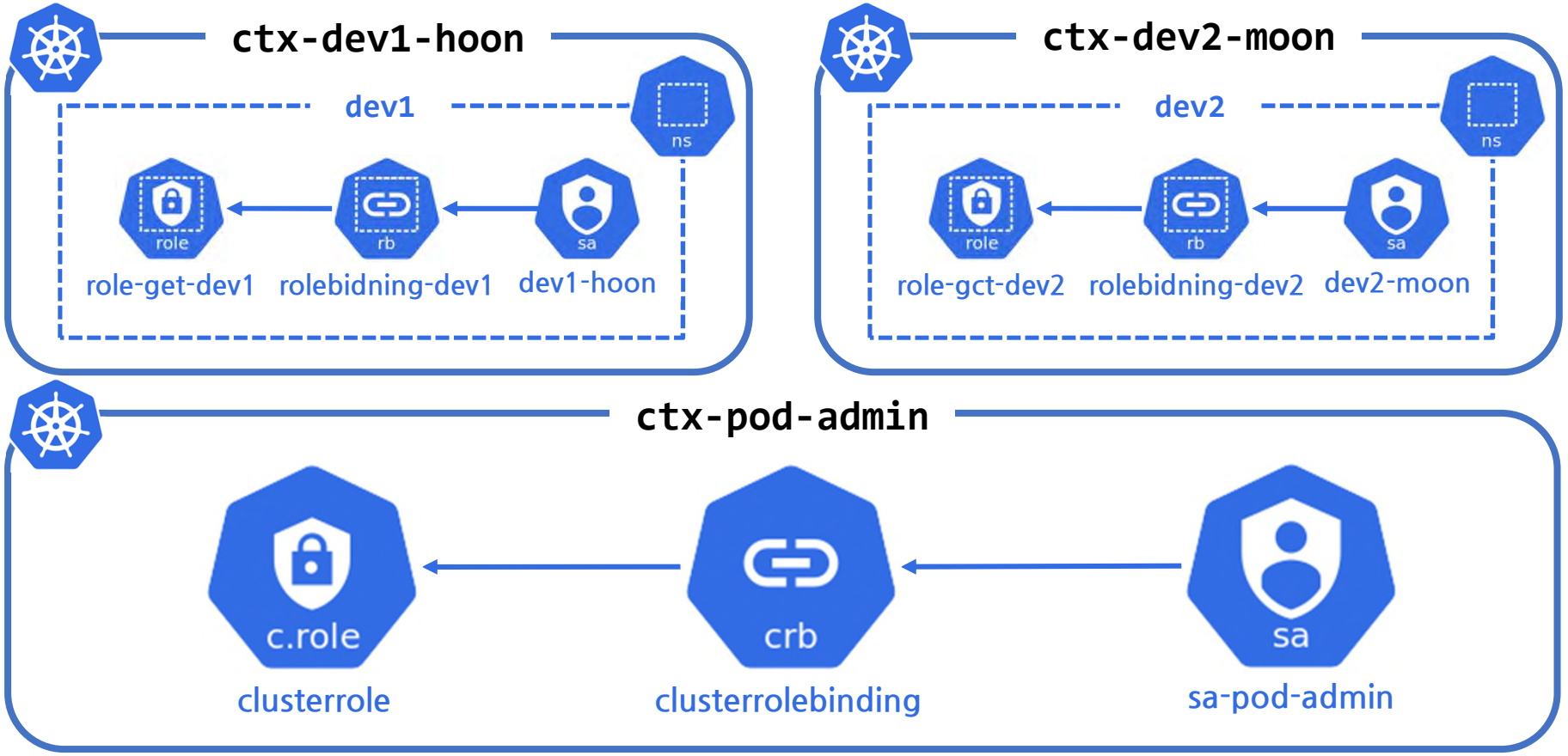
```
[root@m-k8s ~]# kubectl describe pod kube-apiserver-m-k8s -n kube-system ¦ grep -i authorization -F3
      kube-apiserver
      --advertise-address=192.168.1.10
      --allow-privileged=true
      --authorization-mode=Node,RBAC
      --client-ca-file=/etc/kubernetes/pki/ca.crt
      --enable-admission-plugins=NodeRestriction
      --enable-bootstrap-token-auth=true
```

인가 모드	현재 적용	설명
Node	O	스케줄된 노드의 kubelet에서 인가를 결정함
ABAC	X	속성 기반 접근 제어 (Attribute-based access control)
RBAC	O	역할 기반 접근 제어(Role-based access control) 정해진 롤 또는 사용자가 지정한 롤을 이용해서 인가를 제어함
Webhook	X	HTTP Post를 기반으로 페이로드 요청을 보고 인가를 제어함

역할 기반 접근 제어의 전체 구조



컨텍스트(쿠버네티스 클러스터)로 구분된 구조



ns-sa-dev-both.yaml

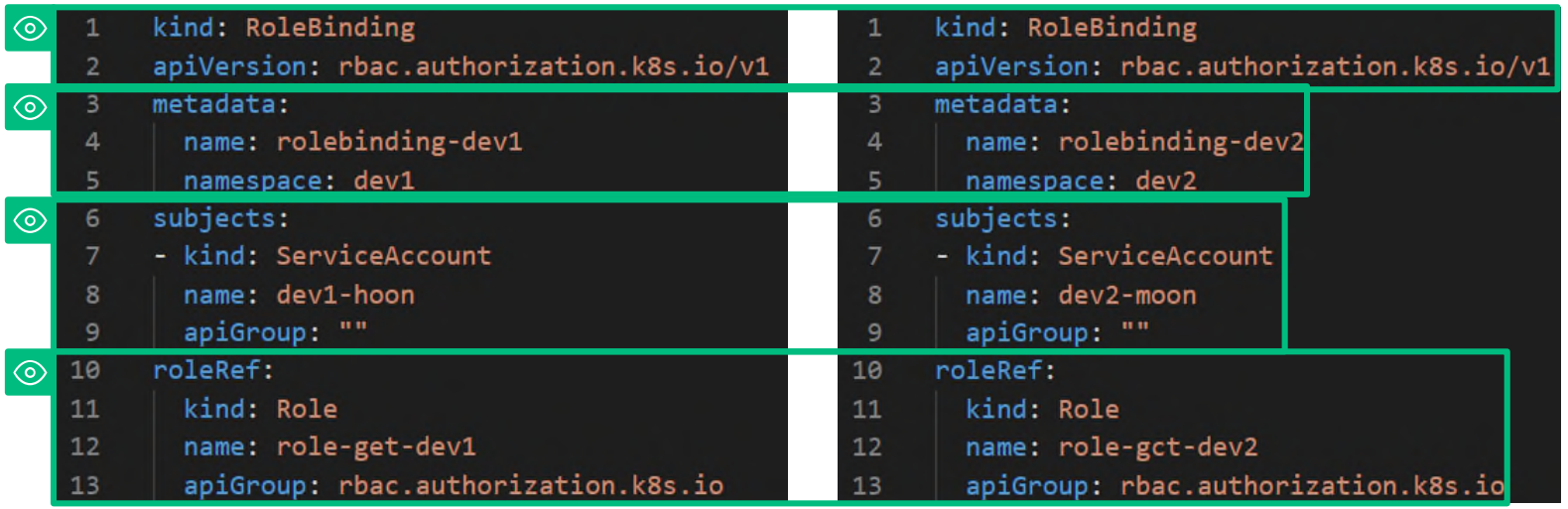
```yaml
# dev1 namespace and account
apiVersion: v1
kind: Namespace
metadata:
  name: dev1
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dev1-hoon
  namespace: dev1
---
# dev2 namespace and account
apiVersion: v1
kind: Namespace
metadata:
  name: dev2
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dev2-moon
  namespace: dev2
```

role-get-dev1.yaml, role-gct-dev2.yaml

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: dev1
  name: role-get-dev1
rules:
- apiGroups: ["*"]
  resources: ["pods", "deployments"]
  verbs: ["get", "list"]
```

```yaml
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: dev2
  name: role-gct-dev2
rules:
- apiGroups: ["*"]
  resources: ["pods", "deployments"]
  verbs: ["get", "list","create"]
```

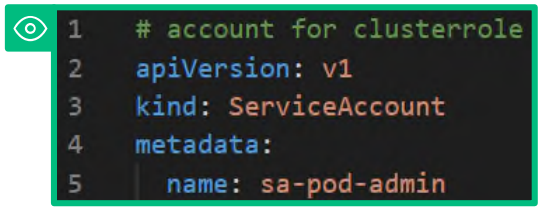
rolebidning-dev1.yaml, rolebidning-dev2.yaml



```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-dev1
  namespace: dev1
subjects:
- kind: ServiceAccount
  name: dev1-hoon
  apiGroup: ""
roleRef:
  kind: Role
  name: role-get-dev1
  apiGroup: rbac.authorization.k8s.io
```

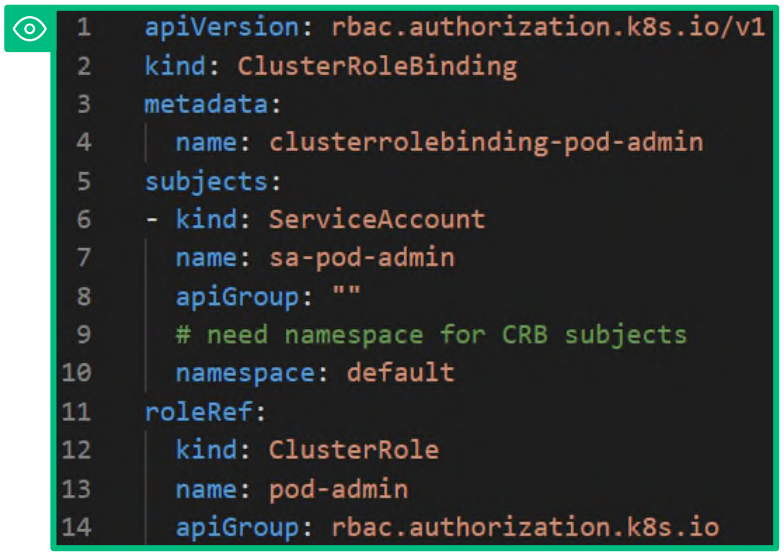
```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-dev2
  namespace: dev2
subjects:
- kind: ServiceAccount
  name: dev2-moon
  apiGroup: ""
roleRef:
  kind: Role
  name: role-gct-dev2
  apiGroup: rbac.authorization.k8s.io
```

sa-pod-admin, clusterrole, clusterrolebinding.yaml



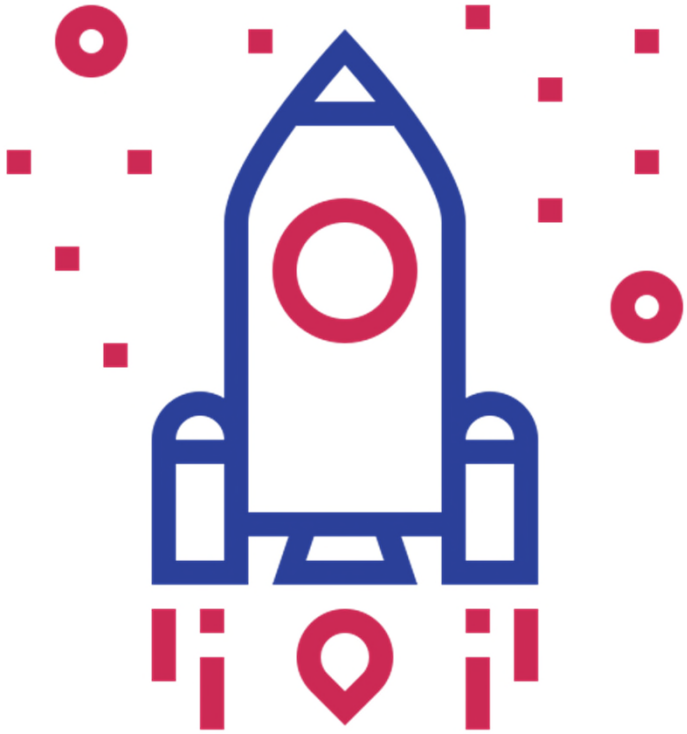
```yaml
# account for clusterrole
apiVersion: v1
kind: ServiceAccount
metadata:
  name: sa-pod-admin
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: pod-admin
rules:
- apiGroups: ["*"]
  resources: ["pods","deployments","deployments/scale"]
  verbs: ["*"]
```

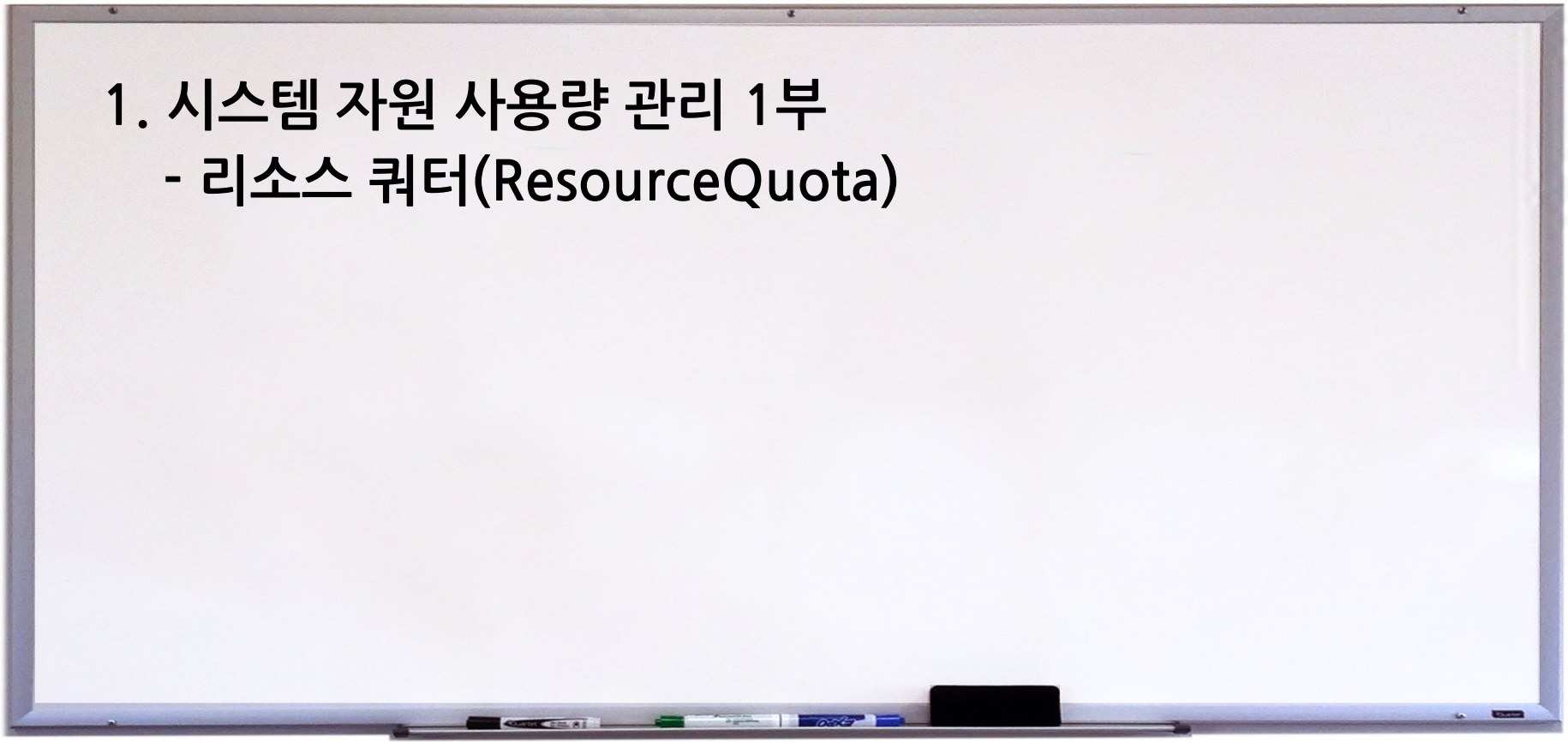


```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: clusterrolebinding-pod-admin
subjects:
- kind: ServiceAccount
  name: sa-pod-admin
  apiGroup: ""
  # need namespace for CRB subjects
  namespace: default
roleRef:
  kind: ClusterRole
  name: pod-admin
  apiGroup: rbac.authorization.k8s.io
```

역할 기반 접근 제어 동작 확인



다음 강좌에는...



kubernetes