[접근/권한] APIs and Access

Label과 Annotation

- 레이블 : 오브젝트를 식별하는데 도움이 되는 문자열 키/쌍 (쿼리 가능)
- 어노테이션 : 단순 주석. 모든 API오브젝트는 주석 포함 가능(쿼리 불가)
  - 쿠버네티스의 실험적인 기능
  - 제작사별 특이한 기능. 메타데이터로 가능하므로 그래픽 아이콘도 가능.

```
[centos@osk-master-01 .kube]$ kubectl describe pod -n kube-system calico-kube-con
Name:                 calico-kube-controllers-58497c65d5-wnfnv
Namespace:            kube-system
Priority:             2000000000
Priority Class Name:  system-cluster-critical
Node:                 osk-master-01.kr-central-1.c.internal/172.30.5.193
Start Time:           Sat, 14 Aug 2021 11:43:38 +0000
Labels:               k8s-app=calico-kube-controllers
                      pod-template-hash=58497c65d5
Annotations:          cni.projectcalico.org/containerID: d315adb5bb72f128f74dac3663e481356be6ae4ca58efc9820a2f47a14de0583
                      cni.projectcalico.org/podIP: 192.168.205.193/32
                      cni.projectcalico.org/podIPs: 192.168.205.193/32
Status:               Running
```

API 접근

- API 서버 : 중앙 접근 포인트. Stateless 이며(etcd 활용), 복제 가능
- 주요 기능

  1) API 관리 : 서버에서 API를 노출하고 관리하는 프로세스

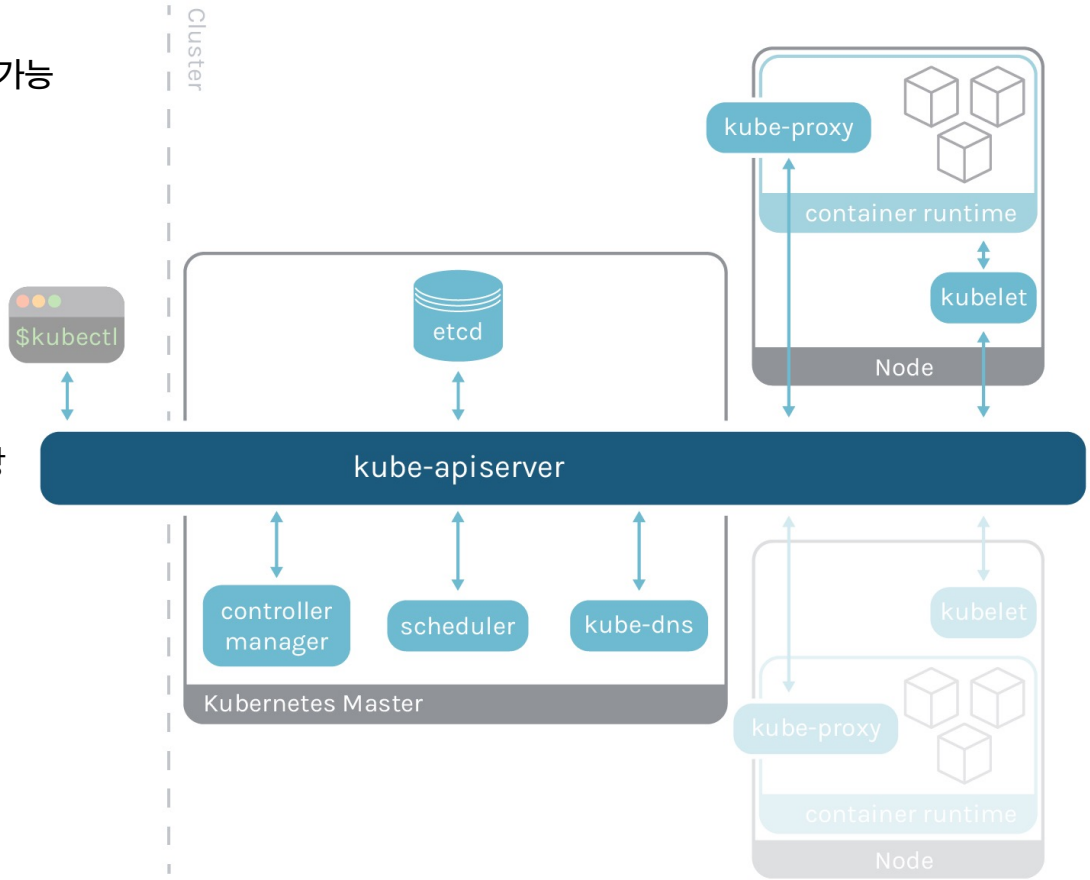
  2) 요청 처리 : 클라이언트의 개별 API 요청을 처리하는 기능

  - 대부분 HTTP 형태로 요청, 콘텐츠는 JSON 기반이 많음
  - 요청의 유형 예 : GET / LIST / POST / DELETE

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

  3) 내부 제어 루프 : API 작동에 필요한 백그라운드 작업을 담당

  (대부분은 컨트롤러 매니저에서 수행)



그림출처 : https://sysdig.com/blog/monitor-kubernetes-api-server/

V1 Group API / API 리소스

- API 그룹은 쿠버네티스 API를 더 쉽게 확장하게 설계됨

  API 그룹은 REST 경로와 오브젝트의 apiVersion 필드에 명시
- 쿠버네티스에는 두 종류의 API 그룹 존재

	REST 경로	apiVersion
V1 group(Core/Legacy)	/api/v1	apiVersion: v1
이름 있는 그룹(후속)	/apis/$GROUP_NAME/$VERSION	apiVersion: $GROUP_NAME/$VERSION

- API 리소스 탐방

  https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22
- (참고) API 버젼 : alpha=불안정하고 상용환경에 부적합,

  beta=안정적이나 최종 개선 예정,

  GA(General Availability)=안정적

인증서

- 쿠버네티스에서는 다양한 인증방법이 있으나, 대부분 PKI(공개 키 인프라) 기반으로 상호 인증
- PKI? 그 전에 대칭키/비대칭키 이론 학습

  - 대칭 키 : 암호화와 복호화에 같은 암호 키(비밀 키만 있음)를 쓰는 알고리즘

    (군대 암구호. 서로 동일하게 알고 있는 키)

  - 비대칭 키 : 공개 키와 비밀 키가 존재하며, 공개 키는 누구나 알 수 있지만

    그에 대응하는 비밀 키는 키의 소유자만이 가지고 있음

    (공개 키 : 촛농, 비밀 키 : 물감)

    → 역으로 생각하면 물감이 있으면 비밀 키 소유자가 맞겠구나

- PKI : 공개된 키를 개인이나 집단을 대표하는 믿을 수 있는 주체와 엮는 것이며
  이는 인증 기관(Certificate authority, 이하 CA)의 등록/인증 발행을 통해 성립

  (성근이가 지은이에게 편지를 보내는데.. 빨간 물감으로, 성근이는 지은이 좋아해)

  (지은이를 사모하던 나쁜남자가 편지 슬쩍하더니, 노란 물감으로, 성근이는 지은이 싫어)

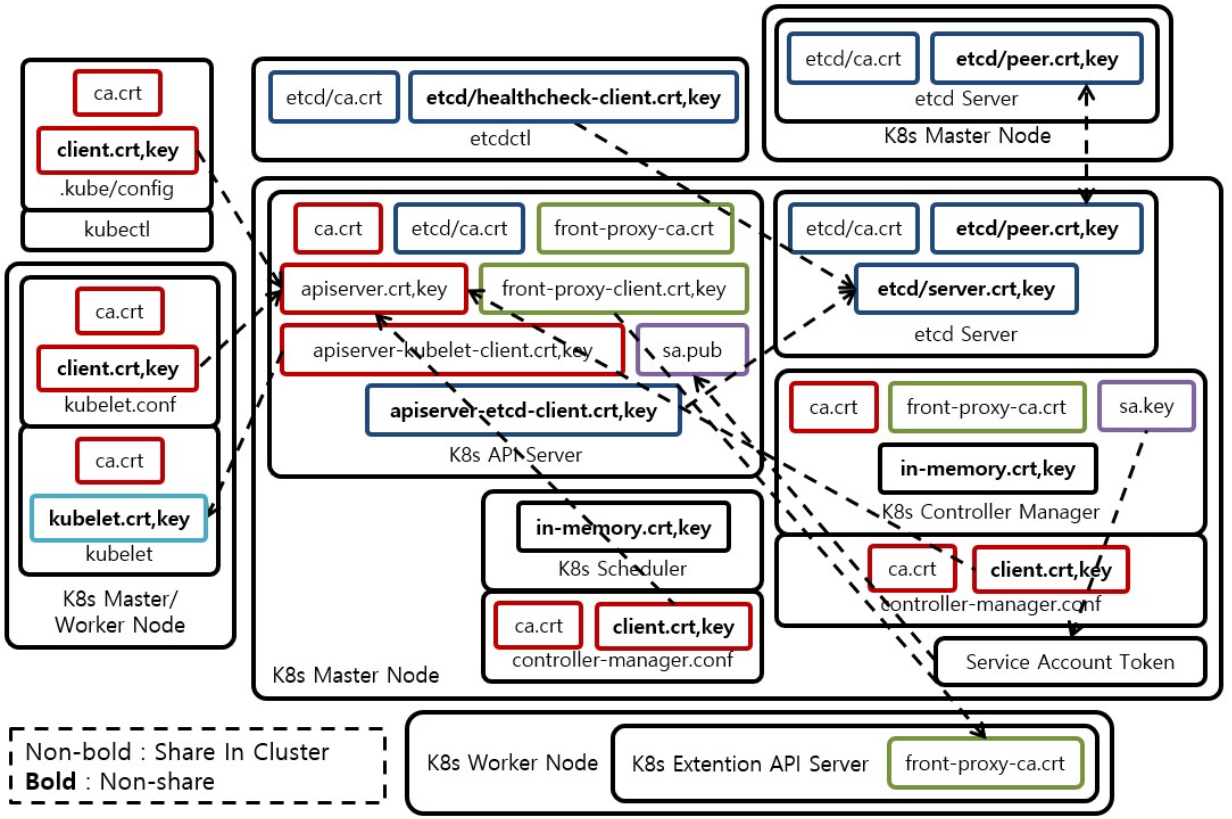
  (→ 선생님 등판, '성근이는 빨간 물감이야' 선생님 인증서 + 빨간 물감으로, 나 너 좋아해)

  .crt 파일 .key 파일

그림출처 : 국방부, 엄마쌤 블로그 https://m.post.naver.com/my.naver?memberNo=45323207

인증서

- 쿠버네티스 컨트롤 플레인에서 자체 CA 역할 수행

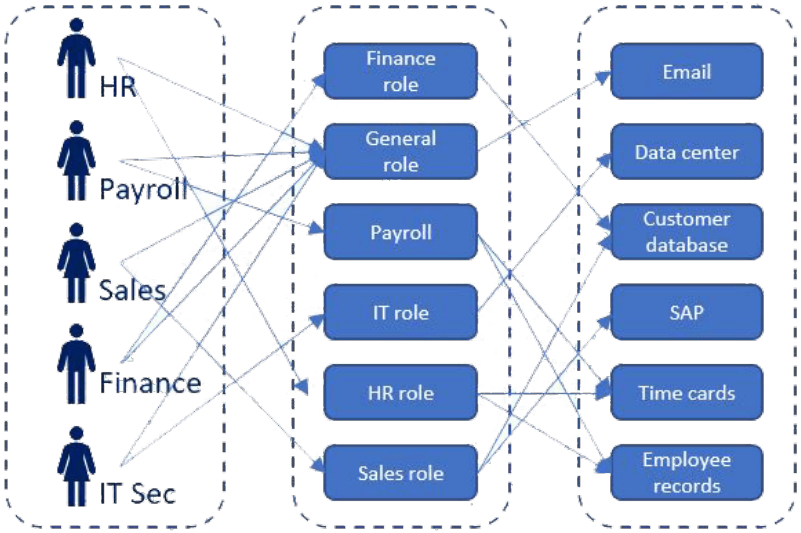


그림출처 : https://ssup2.github.io/theory_analysis/Kubernetes_PKI/

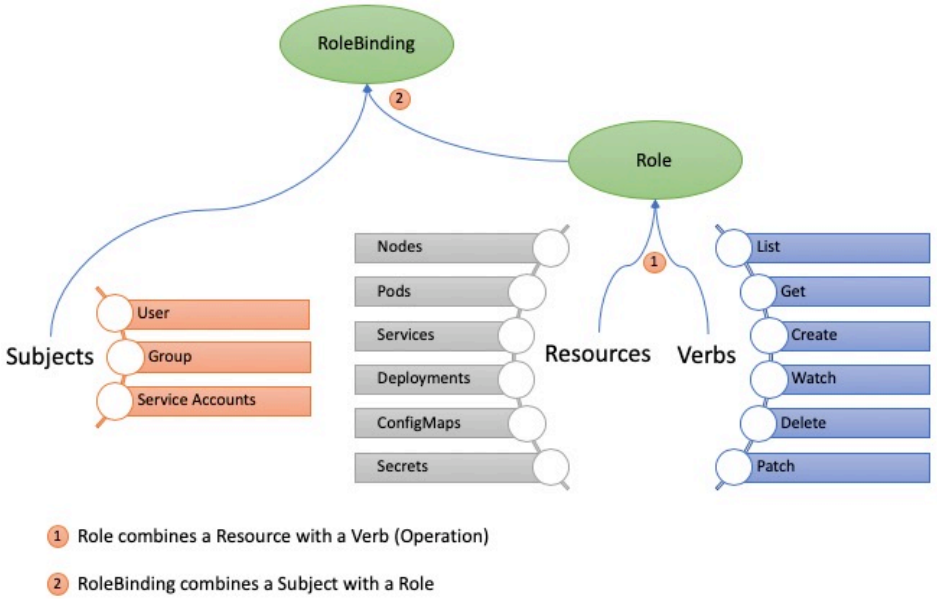
RBAC(Role Based Access Control)

- 인증 모듈은 4개가 존재하며, 이중 RBAC이 가장 대표적이고 효과적인 방식
  https://kubernetes.io/docs/reference/access-authn-authz/rbac/

[RBAC]



[RBAC in Kubernetes]



그림출처 : http://logicalshift.blogspot.com/2019/05/kubernetes-rbac.html

Role, Cluster Role

- 작업 수행에 대한 권한. '어떤 resource에 어떤 verb 권한을?'
- 차이 : kind 에 들어가는 종류명, namespace 존재 여부 (범위만 다름)
- Cluster Role 사용 예시 : 클러스터 관리자, 쿠버네티스 컨트롤러

[Role]

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
-  apiGroups: [""] # "" 는 core API group를 의미
   resources: ["pods"]
   verbs: ["get", "watch", "list"]
-  apiGroups: ["extensions", "apps"]
   resources: ["deployments"]
   verbs: ["get", "list", "watch", "create", "update",
   "patch", "delete"]
```

[Cluster Role]

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata: # 클러스터 롤이므로 "네임스페이스" 는 생략됨

 name: secret-reader
rules:
-  apiGroups: [""]
   resources: ["secrets"]
   verbs: ["get", "watch", "list"]
-  apiGroups: ["extensions", "apps"]
   resources: ["deployments"]
   verbs: ["get", "list", "watch", "create", "update",
   "patch", "delete"]
```

RoleBinding, ClusterRoleBinding

- Role을 사용자/그룹/Service Account에 연결. '어떤 resource에 어떤 verb 권한을,' + '누구에게 줄 것인가?'
- 차이 : kind 에 들어가는 종류명, namespace 존재 여부 (범위만 다름)
- Cluster Role 사용 예시 : 클러스터 관리자, 쿠버네티스 컨트롤러

[Role]

```yaml
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
```

[Cluster Role]

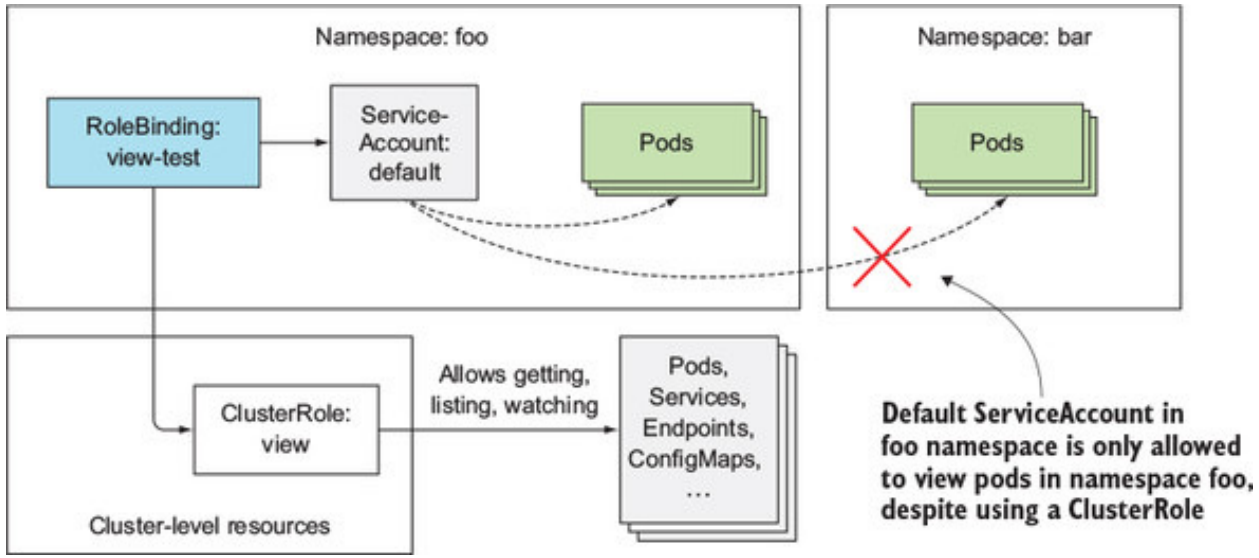
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: read-secrets-global

 subjects:
 - kind: Group
   name: manager
   apiGroup: rbac.authorization.k8s.io
roleRef:
   kind: ClusterRole
   name: secret-reader
   apiGroup: rbac.authorization.k8s.io
```

Service Account

- 파드 내부에서 실행하는 프로세스가 쿠버네티스 API를 호출하고 싶다면? Service Account 활용

  # kubectl create serviceaccount test



그림출처 : Kubernetes in Action