9th Week

아홉 번째 뵙겠습니다 ?!

▷ 잠시만 기다렸다가 30분 되면 시작하겠습니다~^^

▷ 이번 주에는 준비가 조금 부족합니다.

- 이해해 주실거죠!?

▷ Camera는 가급적 켜 주시면 대단히 감사하겠습니다 !!!

- 너무 부끄러우면 Snap Camera를 사용하시는 것 까지는~ ^^

▷ 오늘 수업 자료는 아래 링크에서 다운로드 받으실 수 있어요.

- https://github.com/whatwant-school/kubernetes

///

지난 수업 기억 나시나요?

https://kahoot.it/

///

Kubernetes

Authorization

Authorization Modes

- Kubernetes API-Server에 대한 authorization mode는 다음 4가지 방법이 있다.

  . Node : Kubelet에게 권한을 부여하기 위한 special-purpose authorization mode

  . ABAC : 속성 기반 접근 제어 (ABAC, Attribute-based access control), RBAC 이전에 주로 사용하던 방식

  . RBAC : 역할 기반 접근 제어(RBAC, Role-based access control), 최근에는 RBAC이 표준처럼 사용된다.

  . Webhook : HTTP Callback (이벤트 알림 용도)

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

///

Flip Learning

(RBAC - ServiceAccount/Role/ClusterRole)

/// 님

///

기본 정보 확인

- RBAC을 테스트 하기위해 기본 정보 확인을 해보자

```
❯ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://192.168.100.111:6443
  name: cluster.local
contexts:
- context:
    cluster: cluster.local
    user: kubernetes-admin
  name: kubernetes-admin@cluster.local
current-context: kubernetes-admin@cluster.local
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED

❯ APISERVER=https://192.168.100.111:6443
```

```
❯ kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-4b9h2   kubernetes.io/service-account-token   3      45d
```

decode 해서 사용해야 함

```
❯ kubectl get secrets default-token-4b9h2 -o jsonpath='{$.data.token}' ¦ base64 --decode
eyJhbGciOiJSUzI1NiIsImtpZCI6Img4LVYtQ20wRVhYOVNXak5tMENpUnRueHlGU05CLUJVczZhYXJwX3pRZmMifQ.eyJ ...

❯ TOKEN=eyJhbGciOiJSUzI1NiIsImtpZCI6Img4LVYtQ20wRVhYOVNXak5tMENpUnRueHlGU05CLUJVczZhYXJwX3p ...
```

마지막 %는 제외

```
❯ curl -D - --insecure --header "Authorization: Bearer $TOKEN" $APISERVER/api/v1
HTTP/2 200
cache-control: no-cache, private
content-type: application/json
x-kubernetes-pf-flowschema-uid: b390439f-a87f-4657-aa9c-593d3200192d
x-kubernetes-pf-prioritylevel-uid: 9777a9d3-3be9-4453-803c-2eeb5c194865
date: Fri, 18 Jun 2021 22:34:20 GMT

{
  "kind": "APIResourceList",
  "groupVersion": "v1",

...
```

///

Kubernetes

Authentication & Authorization

API-Server

- **Authentication plugin** : 클라이언트 인증 (신분증 확인)

  . 클라이언트의 사용자 이름, 사용자 ID, 속해 있는 그룹 정보 추출

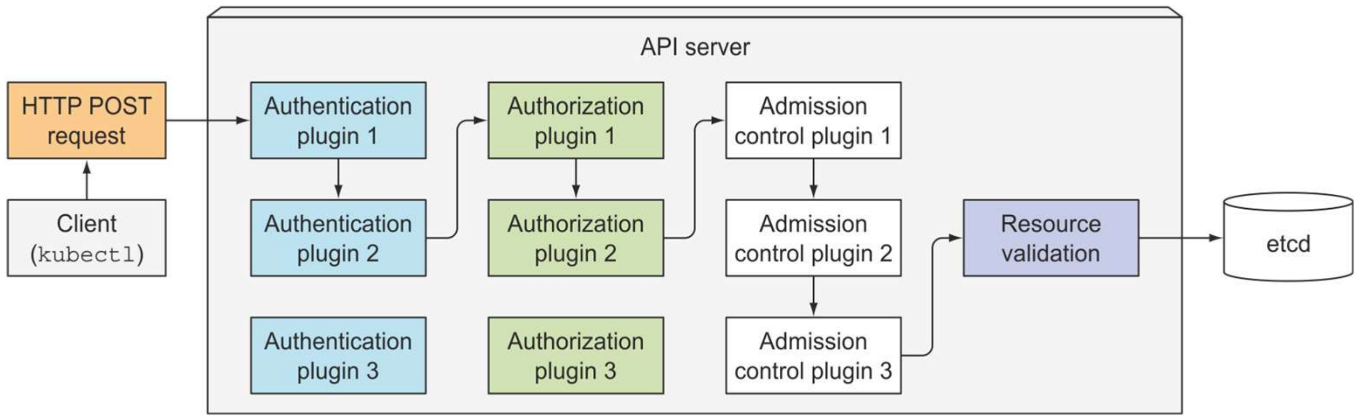
- **Authorization plugin** : 클라이언트 인가

  . 누가 어떤 권한을 갖고 어떤 행동을 할 수 있는지 확인

- **Admission control plugin** : 요청된 리소스 확인 및 수정 (강제 변환)

  . 리소스 생성/수정/삭제 요청인 경우에만 수행 (리소스 정의에서 누락된 필드 초기화/재정의 等)

  . ex) LimitRange, ResourceQuota, AlwaysPullImages, ServiceAccount, NamespaceLifecycle …



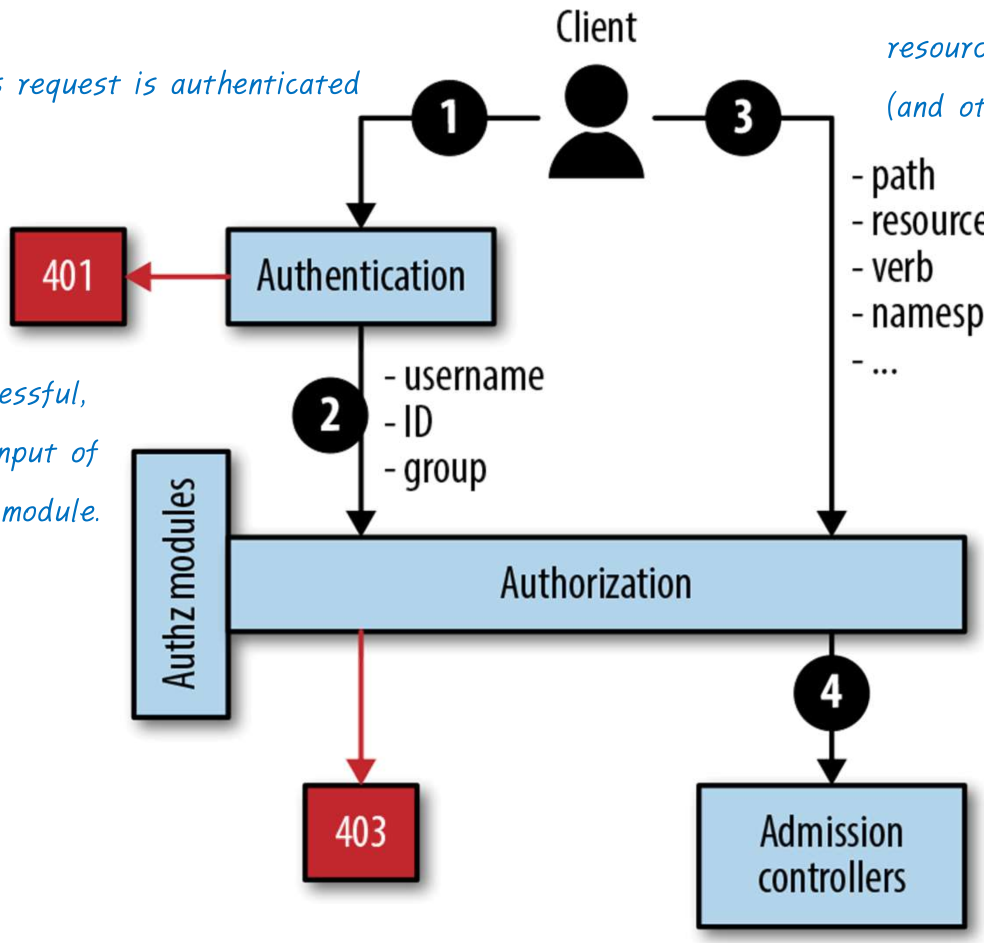
※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-11/93

Detail …

The client's request is authenticated

If the authentication was successful, the credentials are taken as one input of the authorization module.

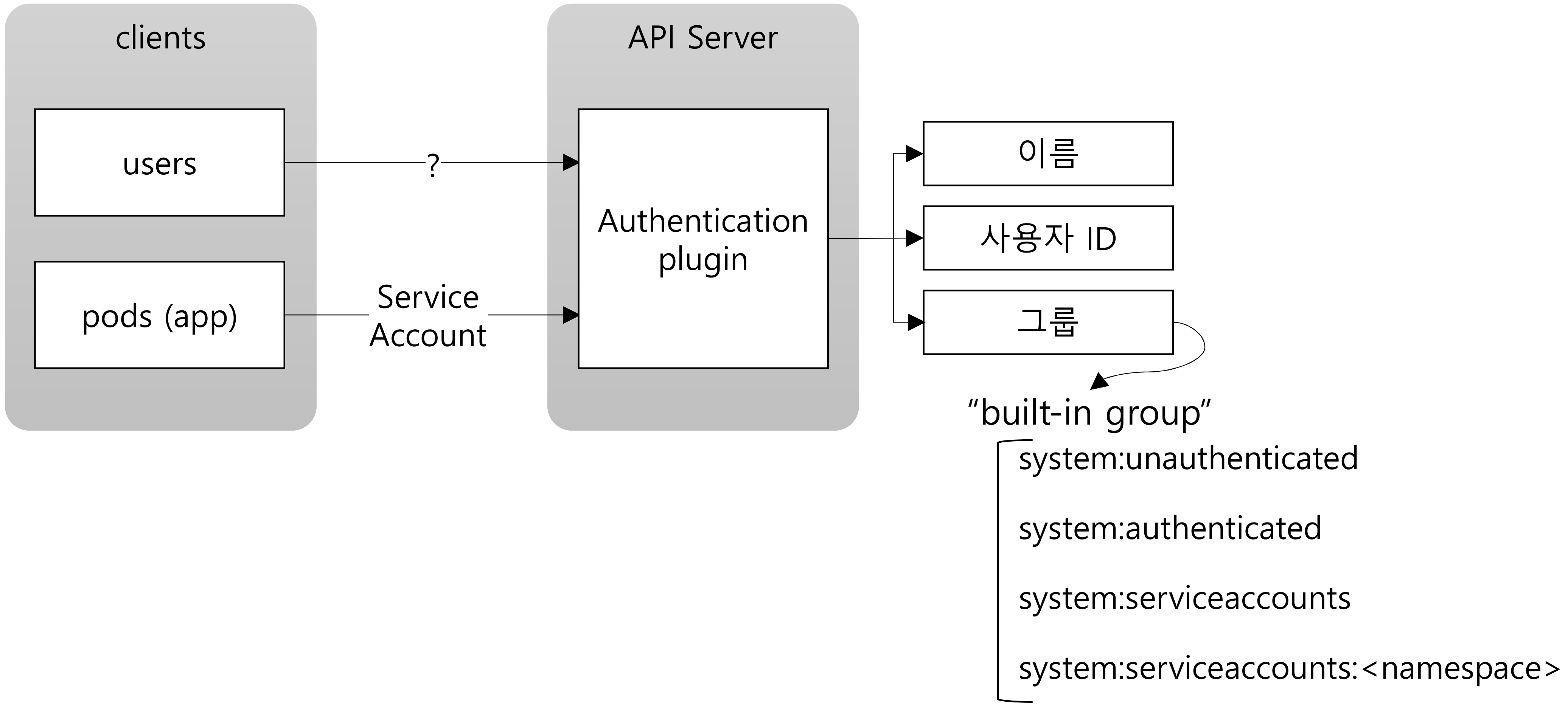
The second input to the authorization module is a vector containing the request path, resource, verb, and namespace (and other secondary attributes)



If the user or application is permitted to execute a certain action on a certain resource, the request is passed on further to the next component in the chain, the admission controller.

※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-12/114

Authentication – Users & Groups



///

Kubernetes

ServiceAccount

ServiceAccount is ...

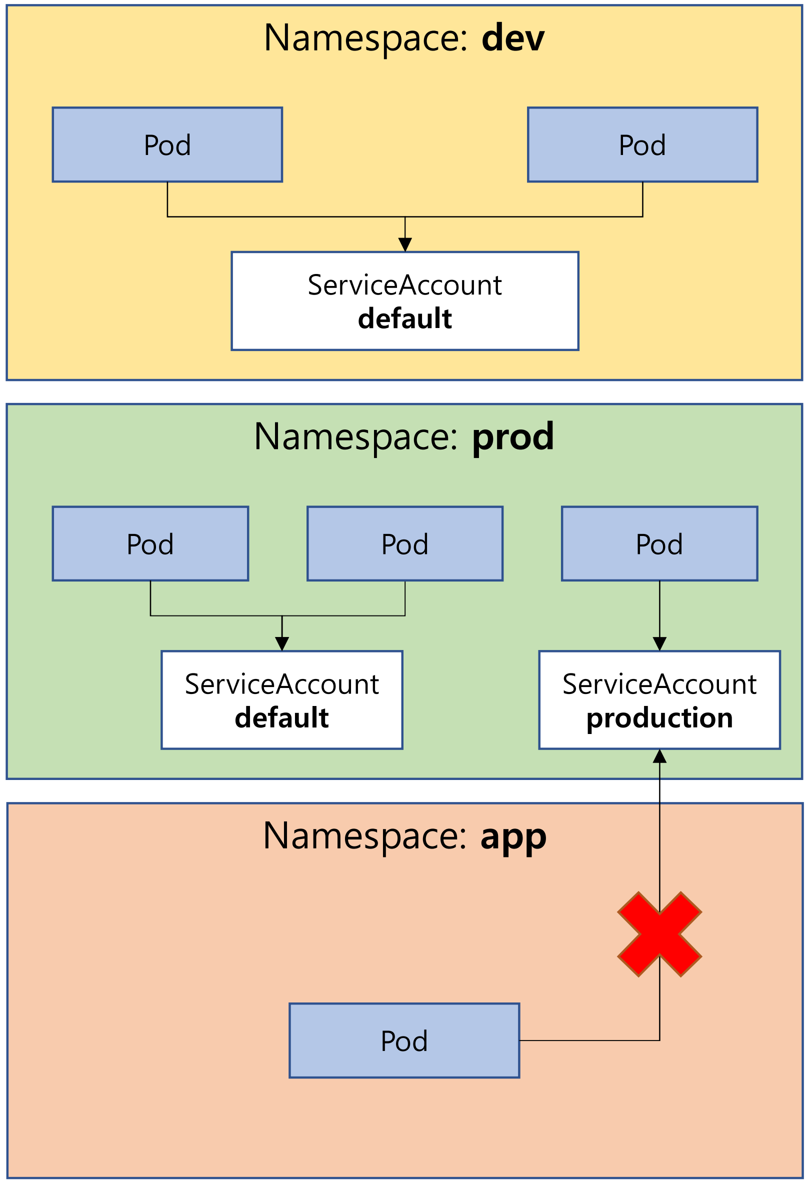
- Kubernetes에서는 `User Account`와 `Service Account` 개념을 구분
  . 하지만, `User Account`를 관리하거나 인증하는 방법을 제공하지는 않음
  . API-Server와 통신하기 위한 `**Service Account**` 기능만 제공

- 각 namespace에 대해 **default** service account 자동 생성

- 각 pod는 정확히 하나의 ServiceAccount만 연결
  . pod는 **같은 namespace**의 ServiceAccount만 사용 가능
  . pod manifest에서 account 지정 가능, 명시하지 않으면 default 사용

- default ServiceAccount는 **unauthenticated user**(인증되지 않은 사용자) 권한
  . 따라서 기본적으로 pod는 클러스터 상태를 볼 수 없음

※ 참고 : https://kubernetes.io/ko/docs/reference/access-authn-authz/service-accounts-admin/
※ 참고 : https://medium.com/@syper/kubernetes-%EB%B3%B4%EC%95%88-740b68758bb6



default

- namespace는 `default`라는 이름을 갖는 기본적인 serviceaccount를 갖고 있다.

```
remote ❯ kubectl get serviceaccounts -A

NAMESPACE         NAME                                 SECRETS   AGE
default           default                              1         38d
ingress-nginx     default                              1         35d
ingress-nginx     ingress-nginx                        1         35d
ingress-nginx     ingress-nginx-admission              1         35d
kube-node-lease   default                              1         38d
kube-public       default                              1         38d
kube-system       attachdetach-controller              1         38d
...
```

전체 namespaces의 모든 ServiceAccount 확인

```
remote ❯ kubectl describe serviceaccounts default

Name:                default
Namespace:           default
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   default-token-xf884
Tokens:              default-token-xf884
Events:              <none>
```

secret을 mount하고 있음을 볼 수 있다.

```
remote ❯ kubectl get secrets -o wide

NAME                  TYPE                                  DATA   AGE
default-token-xf884   kubernetes.io/service-account-token   3      38d
```

API Server 통신을 위한 3종 데이터

- ca.crt / namespace / token

ServiceAccount

namespace 정해서 ServiceAccount를 만들 수도 있고,

당연히 default namespace에도 ServiceAccount를 만들 수도 있다.

namespace.yaml

```yaml
apiVersion: v1
kind: Namespace

metadata:
  name: whatwant
```

serviceaccount.yaml

```yaml
apiVersion: v1
kind: ServiceAccount

metadata:
  name: whatwant
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f 07-week/serviceaccount/namespace.yaml

namespace/whatwant created

remote ❯ kubectl create -f \
07-week/serviceaccount/serviceaccount.yaml --namespace whatwant

serviceaccount/whatwant created

remote ❯ kubectl get serviceaccounts --namespace whatwant

NAME       SECRETS   AGE
default    1         3m
whatwant   1         50s
```

```
remote ❯ kubectl describe serviceaccounts whatwant

Error from server (NotFound): serviceaccounts "whatwant" not found


remote ❯ kubectl describe --namespace whatwant \
serviceaccounts whatwant

Name:                whatwant
Namespace:           whatwant
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   whatwant-token-scts8
Tokens:              whatwant-token-scts8
Events:              <none>


remote ❯ kubectl describe --namespace whatwant \
secrets whatwant-token-scts8

Name:         whatwant-token-scts8
Namespace:    whatwant
...

Data
====
ca.crt:     1099 bytes
namespace:  8 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IjJHUXBHX0NzeUl0a0xVSDh5Nk9CRVNleVBKcmxWdT
...
```

///

Break

///

Kubernetes

Role

Role is …

- **Role** : 특정 namespace에 대한 권한을 설정

- **ClusterRole** : cluster 전체에 대한 권한을 설정

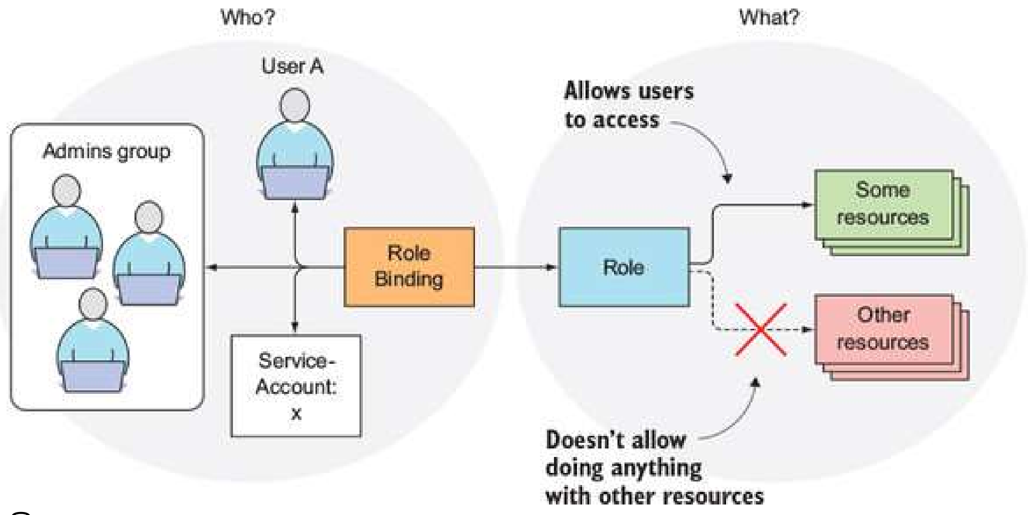
- 각 pod는 정확히 **하나**의 ServiceAccount만 연결

  . pod는 **같은 namespace**의 ServiceAccount만 사용 가능

  . pod manifest에서 account 지정 가능, **명시하지 않으면 default** 사용

- default ServiceAccount는 unauthenticated user(인증되지 않은 사용자) 권한

  . 따라서 **기본적으로 pod는 클러스터 상태를 볼 수 없음**



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

Rules is ...

① **apiGroups** : 사용할 api들을 명시

. core API는 ""로 표현

. ex) deployments를 사용하기 위해선 apps api가 필요 → "apps" 추가

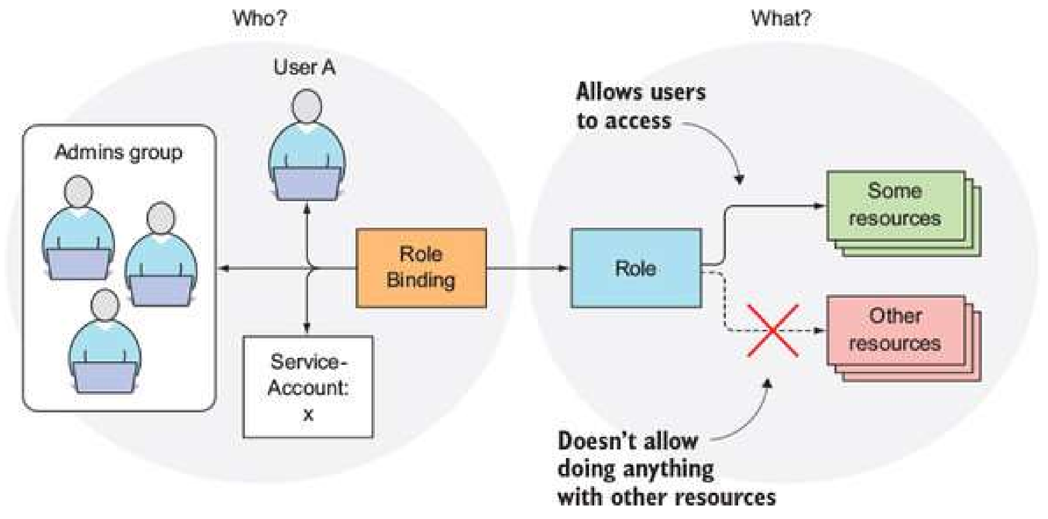
② **resources** : pods, deployments 같은 resources를 명시

. 전체를 지칭할 때에는 "*"를 사용

③ **verbs** : get, edit, list와 같은 verbs 명시

. 전체를 지칭할 때에는 "*"를 사용

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



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

api-resources

- api 정보를 확인해보자.

```
remote ❯ kubectl api-resources -o wide

NAME                              SHORTNAMES APIVERSION                         NAMESPACED KIND                            VERBS
bindings                                     v1                                 true       Binding                         [create]
componentstatuses                 cs         v1                                 false      ComponentStatus                 [get list]
configmaps                        cm         v1                                 true       ConfigMap                       [create delete deletecollection get list patch update watch]
endpoints                         ep         v1                                 true       Endpoints                       [create delete deletecollection get list patch update watch]
events                            ev         v1                                 true       Event                           [create delete deletecollection get list patch update watch]
limitranges                       limits     v1                                 true       LimitRange                      [create delete deletecollection get list patch update watch]
namespaces                        ns         v1                                 false      Namespace                       [create delete get list patch update watch]
nodes                             no         v1                                 false      Node                            [create delete deletecollection get list patch update watch]
persistentvolumeclaims            pvc        v1                                 true       PersistentVolumeClaim           [create delete deletecollection get list patch update watch]
persistentvolumes                 pv         v1                                 false      PersistentVolume                [create delete deletecollection get list patch update watch]
pods                              po         v1                                 true       Pod                             [create delete deletecollection get list patch update watch]
podtemplates                                 v1                                 true       PodTemplate                     [create delete deletecollection get list patch update watch]
replicationcontrollers            rc         v1                                 true       ReplicationController           [create delete deletecollection get list patch update watch]
resourcequotas                    quota      v1                                 true       ResourceQuota                   [create delete deletecollection get list patch update watch]
secrets                                      v1                                 true       Secret                          [create delete deletecollection get list patch update watch]
serviceaccounts                   sa         v1                                 true       ServiceAccount                  [create delete deletecollection get list patch update watch]
services                          svc        v1                                 true       Service                         [create delete get list patch update watch]
mutatingwebhookconfigurations                admissionregistration.k8s.io/v1    false      MutatingWebhookConfiguration    [create delete deletecollection get list patch update watch]
validatingwebhookconfigurations              admissionregistration.k8s.io/v1    false      ValidatingWebhookConfiguration  [create delete deletecollection get list patch update watch]
customresourcedefinitions         crd,crds   apiextensions.k8s.io/v1            false      CustomResourceDefinition        [create delete deletecollection get list patch update watch]
apiservices                                  apiregistration.k8s.io/v1          false      APIService                      [create delete deletecollection get list patch update watch]
controllerrevisions                          apps/v1                            true       ControllerRevision              [create delete deletecollection get list patch update watch]
daemonsets                        ds         apps/v1                            true       DaemonSet                       [create delete deletecollection get list patch update watch]
deployments                       deploy     apps/v1                            true       Deployment                      [create delete deletecollection get list patch update watch]
replicasets                       rs         apps/v1                            true       ReplicaSet                      [create delete deletecollection get list patch update watch]
statefulsets                      sts        apps/v1                            true       StatefulSet                     [create delete deletecollection get list patch update watch]
tokenreviews                                 authentication.k8s.io/v1           false      TokenReview                     [create]
localsubjectaccessreviews                    authorization.k8s.io/v1            true       LocalSubjectAccessReview        [create]
selfsubjectaccessreviews                     authorization.k8s.io/v1            false      SelfSubjectAccessReview         [create]
selfsubjectrulesreviews                      authorization.k8s.io/v1            false      SelfSubjectRulesReview          [create]
subjectaccessreviews                         authorization.k8s.io/v1            false      SubjectAccessReview             [create]
...
```

주요 api-resources

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND	VERBS
nodes	no	v1	FALSE	Node	[create delete deletecollection get list patch update watch]
namespaces	ns	v1	FALSE	Namespace	[create delete get list patch update watch]
pods	po	v1	TRUE	Pod	[create delete deletecollection get list patch update watch]
configmaps	cm	v1	TRUE	ConfigMap	[create delete deletecollection get list patch update watch]
secrets		v1	TRUE	Secret	[create delete deletecollection get list patch update watch]
services	svc	v1	TRUE	Service	[create delete get list patch update watch]
serviceaccounts	sa	v1	TRUE	ServiceAccount	[create delete deletecollection get list patch update watch]
persistentvolumes	pv	v1	FALSE	PersistentVolume	[create delete deletecollection get list patch update watch]
persistentvolumeclaims	pvc	v1	TRUE	PersistentVolumeClaim	[create delete deletecollection get list patch update watch]
replicasets	rs	apps/v1	TRUE	ReplicaSet	[create delete deletecollection get list patch update watch]
deployments	deploy	apps/v1	TRUE	Deployment	[create delete deletecollection get list patch update watch]
statefulsets	sts	apps/v1	TRUE	StatefulSet	[create delete deletecollection get list patch update watch]
daemonsets	ds	apps/v1	TRUE	DaemonSet	[create delete deletecollection get list patch update watch]
jobs		batch/v1	TRUE	Job	[create delete get list patch update watch]
cronjobs	cj	batch/v1beta1	TRUE	CronJob	[create delete deletecollection get list patch update watch]
ingresses	ing	extensions/v1beta1	TRUE	Ingress	[create delete deletecollection get list patch update watch]
ingresses	ing	networking.k8s.io/v1	TRUE	Ingress	[create delete deletecollection get list patch update watch]
ingressclasses		networking.k8s.io/v1	FALSE	IngressClass	[create delete deletecollection get list patch update watch]
roles		rbac.authorization.k8s.io/v1	TRUE	Role	[create delete deletecollection get list patch update watch]
rolebindings		rbac.authorization.k8s.io/v1	TRUE	RoleBinding	[create delete deletecollection get list patch update watch]
clusterroles		rbac.authorization.k8s.io/v1	FALSE	ClusterRole	[create delete deletecollection get list patch update watch]
clusterrolebindings		rbac.authorization.k8s.io/v1	FALSE	ClusterRoleBinding	[create delete deletecollection get list patch update watch]
storageclasses	sc	storage.k8s.io/v1	FALSE	StorageClass	[create delete deletecollection get list patch update watch]
volumeattachments		storage.k8s.io/v1	FALSE	VolumeAttachment	[create delete deletecollection get list patch update watch]

Role YAML

role-1.yaml

```
apiVersion: rbac.authorization.k8c.io/v1
kind: Role
metadata:
  namespace: whatwant
  name: whatwant-role

rules:
- apiGroups: ["", "extensions", "apps"]
  resources: ["deployments", "replicasets", "pods"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

본인이 원하는 방식으로 표현하면 된다.

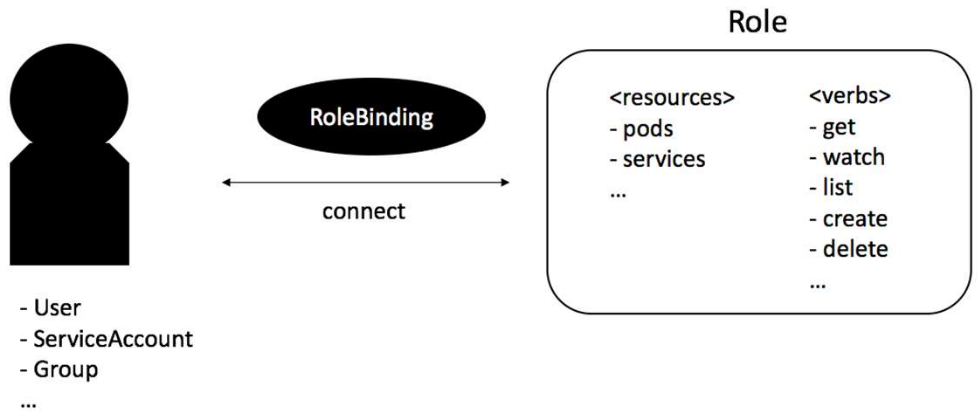
role-2.yaml

```
apiVersion: rbac.authorization.k8c.io/v1
kind: Role
metadata:
  namespace: whatwant
  name: whatwant-role

rules:
- apiGroups:
  - ""
  - extensions
  - apps

  resources:
  - deployments
  - replicasets
  - pods

  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

RoleBinding is …

- Role과 User/Group/ServiceAccount를 묶어(binding)주는 역할

- RoleBinding은 특정 namespace 하나에 적용



① **subjects** : 어떤 유형의 사용자 계정과 연결하는지 설정

. apiGroup: “” → core API 그룹으로 설정

② **roleRef** : 사용자에게 어떤 Role을 할당할지 설정

. roleRef.kind → Role or ClusterRole 명시

※ 참고 : https://blog.dudaji.com/kubernetes/2019/05/01/k8s-authorization-of-sa-with-rbac.html

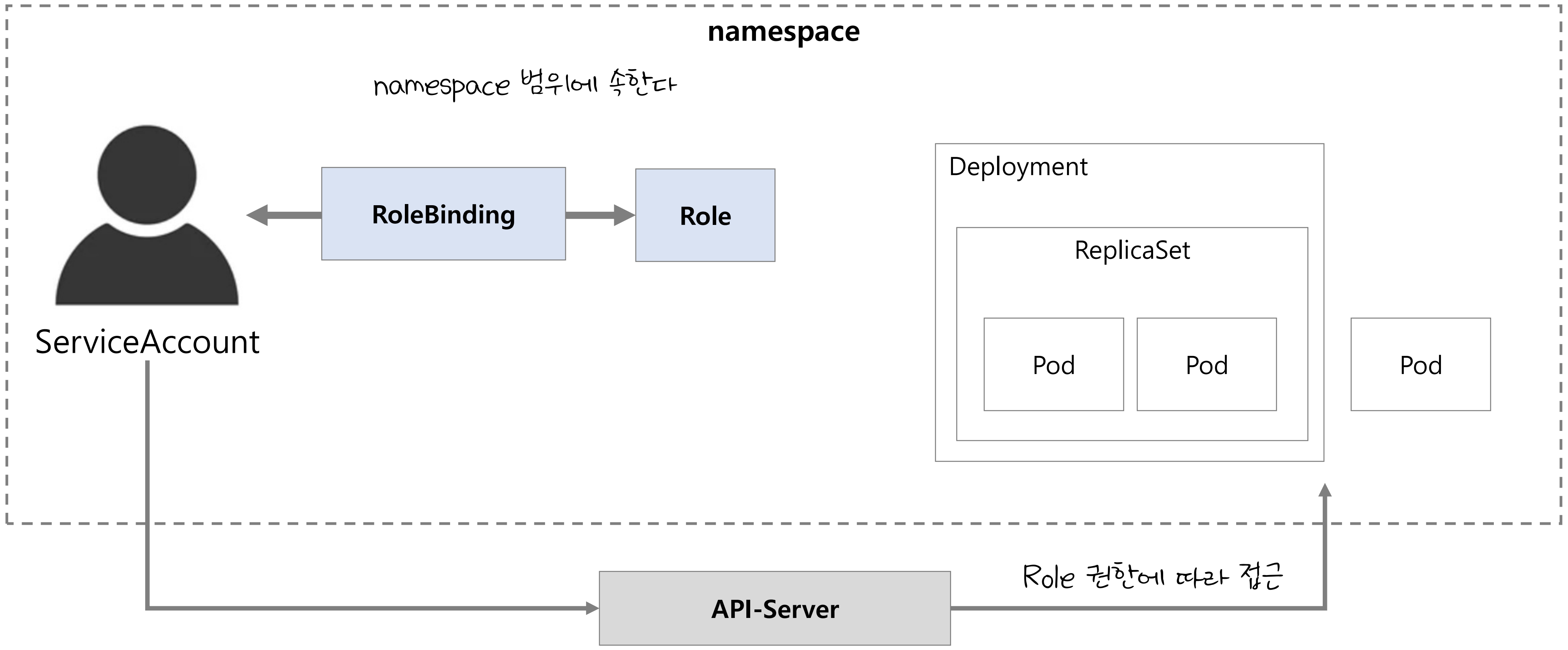
RoleBinding YAML

rolebinding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: whatwant-rb
  namespace: whatwant

subjects:
- kind: ServiceAccount
  name: whatwant
  apiGroup: ""

roleRef:
  kind: Role
  name: whatwant-role
  apiGroup: rbac.authorization.k8s.io
```

///

Hands-On


namespace
namespace 범위에 속한다
ServiceAccount
RoleBinding
Role
Deployment
ReplicaSet
Pod
Pod
Pod
API-Server
Role 권한에 따라 접근

Create

Namespace / Role / ServiceAccount / RoleBinding 리소스를 생성하자

role-basic.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: whatwant-role

rules:
- apiGroups: ["", "extensions", "apps"]
  resources: ["*"]
  verbs: ["*"]
```

rolebinding-basic.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: whatwant-rb

subjects:
- kind: ServiceAccount
  name: whatwant
  apiGroup: ""

roleRef:
  kind: Role
  name: whatwant-role
  apiGroup: rbac.authorization.k8s.io
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f 07-week/serviceaccount/namespace.yaml

namespace/whatwant created


remote ❯ kubectl create --namespace whatwant \
-f 07-week/role/role-basic.yaml

role.rbac.authorization.k8s.io/whatwant-role created


remote ❯ kubectl create --namespace whatwant \
-f 07-week/serviceaccount/serviceaccount.yaml

serviceaccount/whatwant created


remote ❯ kubectl create --namespace whatwant -f 07-
week/role/rolebinding-basic.yaml

rolebinding.rbac.authorization.k8s.io/whatwant-rb created
```

describe

Role / RoleBinding 내역을 한 번 살펴보자. 현재, RoleBinding까지 모두 완료된 상태이다.

```
remote ❯ kubectl get --namespace whatwant roles whatwant-role

NAME            CREATED AT
whatwant-role   2022-03-02T16:27:51Z


remote ❯ kubectl describe --namespace whatwant roles whatwant-role

Name:         whatwant-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources      Non-Resource URLs  Resource Names  Verbs
  ---------      -----------------  --------------  -----
  *              []                 []              [*]
  *.apps         []                 []              [*]
  *.extensions   []                 []              [*]
```

```
remote ❯ kubectl get --namespace whatwant \
rolebindings whatwant-rb -o wide

NAME          ROLE                 AGE    USERS   GROUPS   SERVICEACCOUNTS
whatwant-rb   Role/whatwant-role   3m39s                   /whatwant


remote ❯ kubectl describe --namespace whatwant rolebindings whatwant-rb

Name:         whatwant-rb
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  whatwant-role
Subjects:
  Kind            Name      Namespace
  ----            ----      ---------
  ServiceAccount  whatwant
```

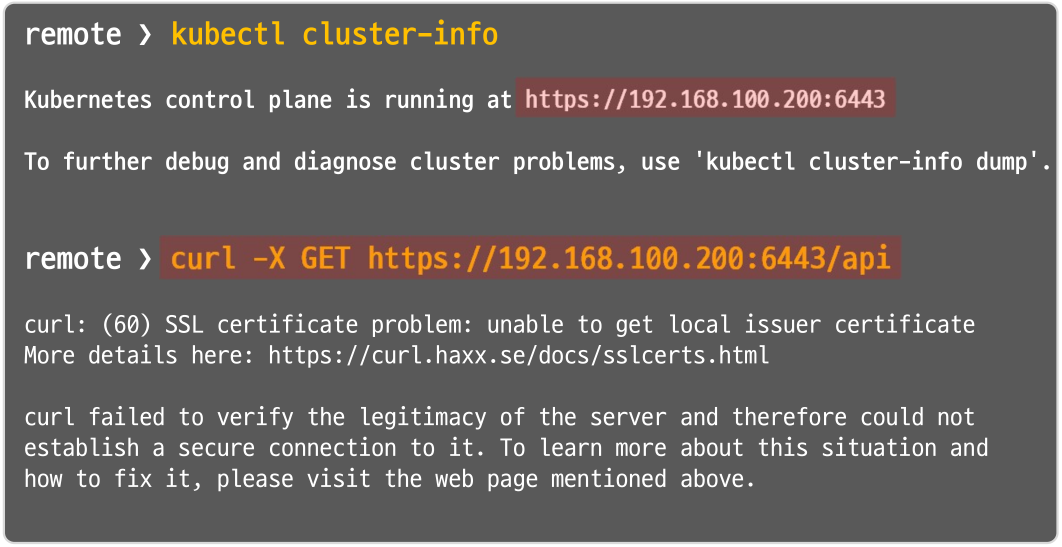
///

Connect to API-Server

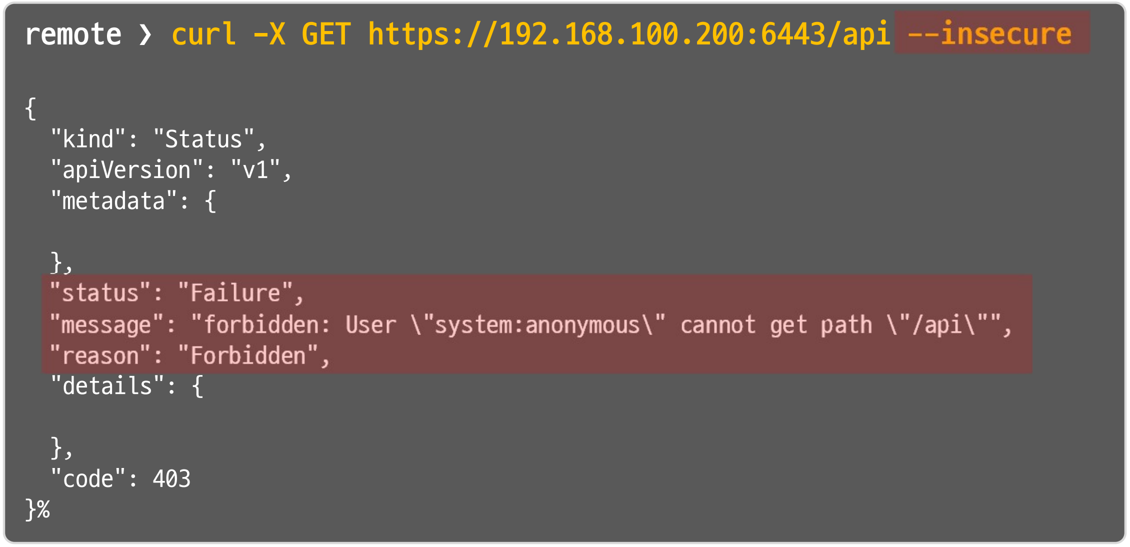
앞에서 생성한 RoleBinding까지 마친 ServiceAccount가 어떻게 사용되는지 살펴보자.

더 앞의 그림에서 본 것처럼 API-Server와 통신을 요청하게 되면 충분한 권한이 있는지 여부에 따라 반응을 하게 된다.

그러면 먼저 API-Server의 주소를 먼저 확인한 뒤 접근 해보자.



```
remote ❯ kubectl cluster-info

Kubernetes control plane is running at https://192.168.100.200:6443

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.


remote ❯ curl -X GET https://192.168.100.200:6443/api

curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```



```
remote ❯ curl -X GET https://192.168.100.200:6443/api --insecure

{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/api\"",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
}%
```

https를 사용하기 위한 SSL 인증서 이슈가 있다.

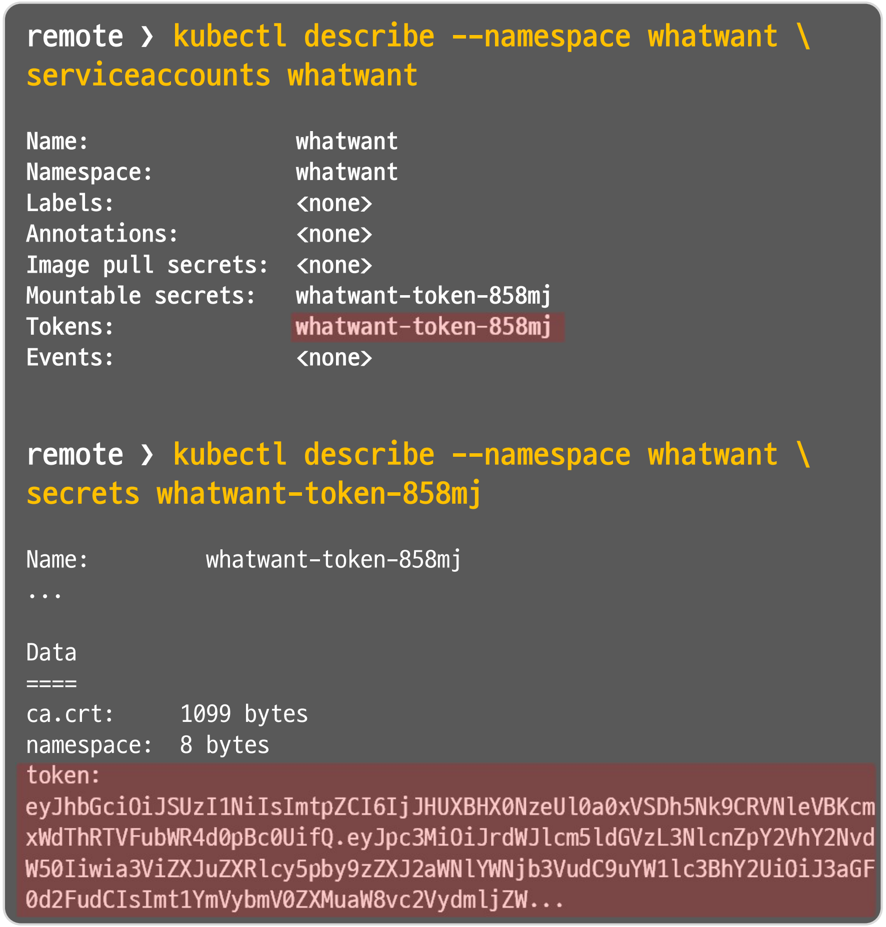
여기에서 사용하고 있는 것이 사설 인증서 이기에 발생하는 것인데, 일단 지금은 `--insecure` 옵션을 통해 무시해버리도록 하자.

그런데, 그렇게 해도 여전히 계정을 별도로 지정해주지 않았기에 Forbidden 상황이다.

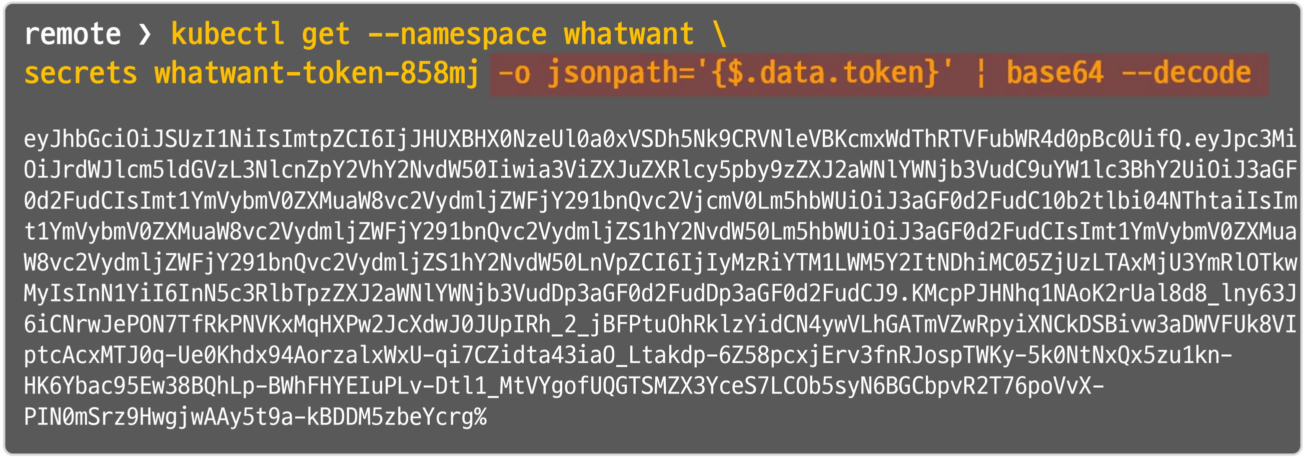
Get Token

API-Server에 ServiceAccount를 사용하기 위해서는 token 형태로 header에 포함해서 전달해야 한다.

ServiceAccount의 Secret 정보를 통해 token 값을 확인해보자.



```
remote ❯ kubectl describe --namespace whatwant \
serviceaccounts whatwant

Name:                whatwant
Namespace:           whatwant
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   whatwant-token-858mj
Tokens:              whatwant-token-858mj
Events:              <none>


remote ❯ kubectl describe --namespace whatwant \
secrets whatwant-token-858mj

Name:         whatwant-token-858mj
...

Data
====
ca.crt:     1099 bytes
namespace:  8 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6IjJHUXBHX0NzeUl0a0xVSDh5Nk9CRVNleVBKcm
xWdThRTVFubWR4d0pBc0UifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2Nvd
W50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJ3aGF
0d2FudCIsImt1YmVybmV0ZXMuaW8vc2VydmljZW...
```



```
remote ❯ kubectl get --namespace whatwant \
secrets whatwant-token-858mj -o jsonpath='{$.data.token}' ¦ base64 --decode

eyJhbGciOiJSUzI1NiIsImtpZCI6IjJHUXBHX0NzeUl0a0xVSDh5Nk9CRVNleVBKcmxWdThRTVFubWR4d0pBc0UifQ.eyJpc3Mi
OiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJ3aGF
0d2FudCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJ3aGF0d2FudC10b2tlbi04NThtaiIsIm
t1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJ3aGF0d2FudCIsImt1YmVybmV0ZXMua
W8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6IjIyMzRiYTM1LWM5Y2ItNDhiMC05ZjUzLTAxMjU3YmRlOTkw
MyIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDp3aGF0d2FudDp3aGF0d2FudCJ9.KMcpPJHNhq1NAoK2rUal8d8_lny63J
6iCNrwJePON7TfRkPNVKxMqHXPw2JcXdwJ0JUpIRh_2_jBFPtuOhRklzYidCN4ywVLhGATmVZwRpyiXNCkDSBivw3aDWVFUk8VI
ptcAcxMTJ0q-Ue0Khdx94AorzalxWxU-qi7CZidta43iaO_Ltakdp-6Z58pcxjErv3fnRJospTWKy-5k0NtNxQx5zu1kn-
HK6Ybac95Ew38BQhLp-BWhFHYEIuPLv-Dtl1_MtVYgofUQGTSMZX3YceS7LCOb5syN6BGCbpvR2T76poVvX-
PIN0mSrz9HwgjwAAy5t9a-kBDDM5zbeYcrg%
```

-o jsonpath='{$.data.token}' 형식으로 결과값을 출력하면,

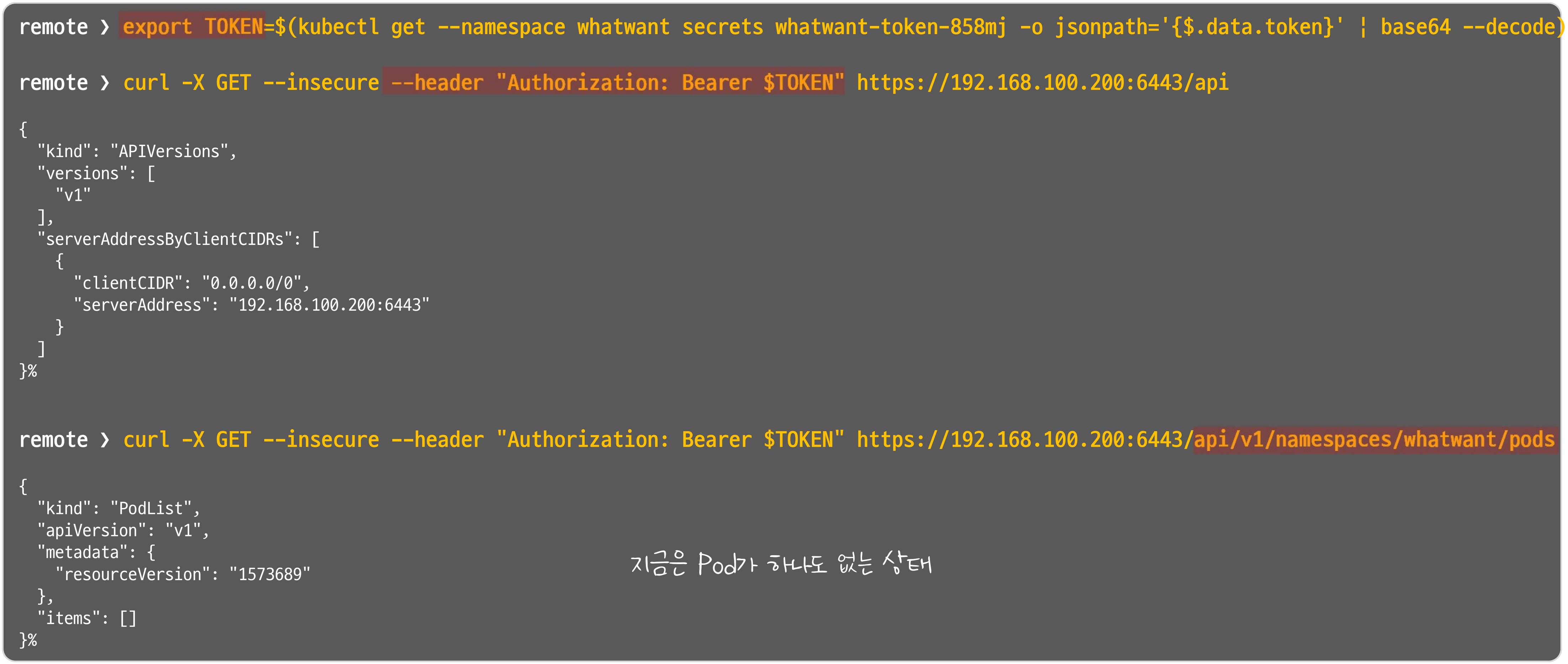
token 값이 base64 encoding 된 형태로 나오기 때문에

pipe 방식으로 decoding 하도록 해서 값이 나오도록 했다.

Re-Connect to API-Server

Token 값을 그대로 사용하기에는 불편하니까 환경변수로 등록 해놓고 사용해보자.

중요한 것은 header 항목으로 `Authorization: Bearer` 값에 token 값을 넣어주는 것이다.



```
remote ❯ export TOKEN=$(kubectl get --namespace whatwant secrets whatwant-token-858mj -o jsonpath='{$.data.token}' | base64 --decode)

remote ❯ curl -X GET --insecure --header "Authorization: Bearer $TOKEN" https://192.168.100.200:6443/api

{
  "kind": "APIVersions",
  "versions": [
    "v1"
  ],
  "serverAddressByClientCIDRs": [
    {
      "clientCIDR": "0.0.0.0/0",
      "serverAddress": "192.168.100.200:6443"
    }
  ]
}%


remote ❯ curl -X GET --insecure --header "Authorization: Bearer $TOKEN" https://192.168.100.200:6443/api/v1/namespaces/whatwant/pods

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "1573689"
  },
  "items": []
}%
```

지금은 Pod가 하나도 없는 상태

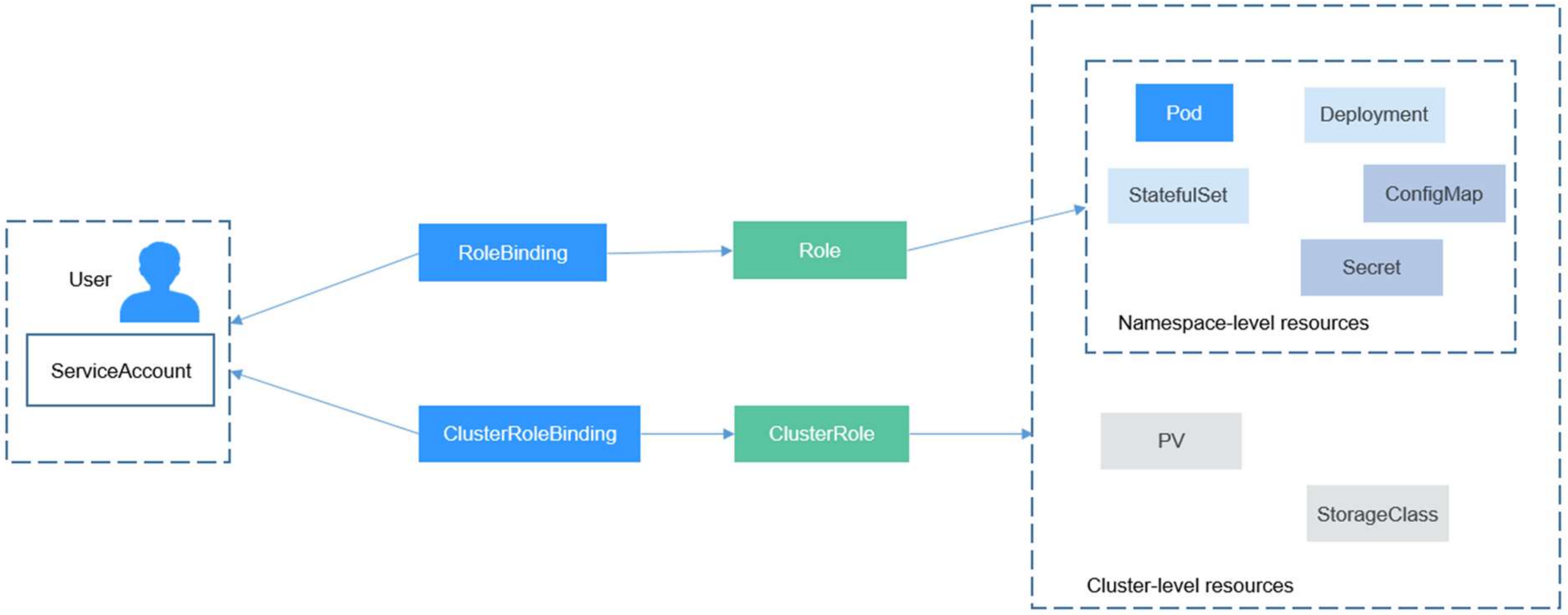
///

Kubernetes

ClusterRole

ClusterRole is …

- ClusterRole은 Role과 비슷하지만 cluster기반 권한 부여이므로, node, endpoint, 모든 namespace에 대한 권한 셋팅 가능



※ 참고 : https://support.huaweicloud.com/intl/en-us/usermanual-cce/cce_01_0189.html

///

get clusterroles

현재 모든 clusterroles 목록을 확인해보자

```
remote ❯ kubectl get clusterroles

NAME                                                        CREATED AT
admin                                                       2022-01-22T08:15:26Z
calico-kube-controllers                                     2022-01-22T08:16:48Z
calico-node                                                 2022-01-22T08:16:33Z
cluster-admin                                               2022-01-22T08:15:26Z
edit                                                        2022-01-22T08:15:26Z
...
system:kube-controller-manager                              2022-01-22T08:15:26Z
system:kube-dns                                             2022-01-22T08:15:26Z
system:kube-scheduler                                       2022-01-22T08:15:26Z
...


remote ❯ kubectl get clusterrolebindings

NAME                            ROLE                                         AGE
calico-kube-controllers         ClusterRole/calico-kube-controllers          41d
calico-node                     ClusterRole/calico-node                      41d
cluster-admin                   ClusterRole/cluster-admin                    41d
...
system:kube-controller-manager  ClusterRole/system:kube-controller-manager   41d
system:kube-dns                 ClusterRole/system:kube-dns                  41d
system:kube-scheduler           ClusterRole/system:kube-scheduler            41d
system:metrics-server           ClusterRole/system:metrics-server            41d
system:monitoring               ClusterRole/system:monitoring                41d
system:node                     ClusterRole/system:node                      41d
system:node-proxier             ClusterRole/system:node-proxier              41d
system:node-webhook             ClusterRole/system:node-webhook              41d
system:public-info-viewer       ClusterRole/system:public-info-viewer        41d
...
```

```
remote ❯ kubectl describe clusterroles system:kube-dns

Name:         system:kube-dns
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  endpoints  []                 []              [list watch]
  services   []                 []              [list watch]


remote ❯ kubectl describe clusterrolebinding system:kube-dns

Name:         system:kube-dns
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  system:kube-dns
Subjects:
  Kind            Name      Namespace
  ----            ----      ---------
  ServiceAccount  kube-dns  kube-system

remote ❯ kubectl describe clusterroles cluster-admin

Name:         cluster-admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  *.*        []                 []              [*]
             [*]                []              [*]
```

ClusterRoleBinding : cluster-admin

clusterrolebinding.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding

metadata:
  name: cluster-admin-clusterrolebinding

subjects:
- kind: ServiceAccount
  name: whatwant
  namespace: whatwant

roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
```

ServiceAccount whatwant에

cluster-admin ClusterRole을 Binding해보자.

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f 07-week/role/clusterrolebinding.yaml

clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-clusterrolebinding created


remote ❯ kubectl describe clusterrolebindings cluster-admin-clusterrolebinding

Name:         cluster-admin-clusterrolebinding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  cluster-admin
Subjects:
  Kind            Name      Namespace
  ----            ----      ---------
  ServiceAccount  whatwant  whatwant

remote ❯ kubectl describe clusterrole cluster-admin

Name:         cluster-admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  *.*        []                 []              [*]
             [*]                []              [*]
```

ClusterRoleBinding : cluster-admin

```
remote ❯ export TOKEN=$(kubectl get --namespace whatwant secrets whatwant-token-858mj -o jsonpath='{$.data.token}' | base64 --decode)

remote ❯ curl -X GET --insecure --header "Authorization: Bearer $TOKEN" https://192.168.100.200:6443/api/v1/namespaces/default/services

{
  "kind": "ServiceList",
  "apiVersion": "v1",
...
```

whatwant namespace에서 생성한 whatwant ServiceAccount인데,

cluster-admin ClusterRole을 Binding 시켜주니 default namespace에 대한 권한도 갖고 있다.

```
remote ❯ kubectl delete clusterrolebindings cluster-admin-clusterrolebinding

clusterrolebinding.rbac.authorization.k8s.io "cluster-admin-clusterrolebinding" deleted


remote ❯ curl -X GET --insecure --header "Authorization: Bearer $TOKEN" https://192.168.100.200:6443/api/v1/namespaces/default/services

{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "services is forbidden: User \"system:serviceaccount:whatwant:whatwant\" cannot list resource \"services\" in API group \"\" in the namespace \"default\"",
  "reason": "Forbidden",
  "details": {
    "kind": "services"
  },
  "code": 403
}%
```

cluster-admin ClusterRoleBinding을 삭제하니

default namespace에 대한 권한도 사라졌음을 볼 수 있다.

///

Example #1

ImagePullSecrets

Volume 내용 중

Secret 부분에서

ImagePullSecrets를 기억하시나요?

docker registry : Create secret

- Registry Server 인증 정보를 secret으로 등록하자

pod-private-success.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-private
  labels:
    app: nginx

spec:
  containers:
  - name: nginx
    image: whatwant/simple-nginx:v0.1

  imagePullSecrets:
    - name: my-docker-hub
```

```
kubectl create secret docker-registry <secret-name> \
                  [ --docker-server=<your-registry-server> \ ]
                    --docker-username=<your-name> \
                    --docker-password=<your-password> \
                    --docker-email=<your-email>
```

```
remote ❯ kubectl create secret docker-registry my-docker-hub \
                               --docker-username=whatwant \
                               --docker-password='xxx' \
                               --docker-email='whatwant@gmail.com'
secret/my-docker-hub created


remote ❯ kubectl get secrets -o wide

NAME                  TYPE                                  DATA   AGE
default-token-xf884   kubernetes.io/service-account-token   3      24d
my-docker-hub         kubernetes.io/dockerconfigjson        1      19s
```

```
remote ❯ kubectl create -f ./05-week/secret/pod-private-success.yaml

pod/pod-private created


remote ❯ kubectl get pods

NAME          READY   STATUS    RESTARTS   AGE
pod-private   1/1     Running   0          3m4s
```

Pod 정의할 때

Secret name을 지정하는 방식이었는데,

ServiceAccount를 이용해봅시다.

ServiceAccount

serviceaccount.yaml

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dockerhub-account

imagePullSecrets:
- name: my-dockerhub
```

pod-private-serviceaccount.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-private
  labels:
    app: nginx

spec:
  containers:
  - name: nginx
    image: whatwant/simple-nginx:v0.1
    imagePullPolicy: Always

  serviceAccountName: dockerhub-account
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create secret docker-registry my-dockerhub --docker-username=whatwant \
                            --docker-password='xxx' --docker-email='whatwant@gmail.com'

secret/my-dockerhub created


remote ❯ kubectl create -f 07-week/imagepullsecrets/serviceaccount.yaml

serviceaccount/dockerhub-account created


remote ❯ kubectl create -f 07-week/imagepullsecrets/pod-private-serviceaccount.yaml

pod/pod-private created


remote ❯ kubectl describe pods pod-private

Name:         pod-private
...
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  13s   default-scheduler  Successfully assigned default/pod-private to worker2
  Normal  Pulling    12s   kubelet            Pulling image "whatwant/simple-nginx:v0.1"
  Normal  Pulled     10s   kubelet            Successfully pulled image "whatwant/simple-nginx:v0.1" in 1.95494831s
  Normal  Created    10s   kubelet            Created container nginx
  Normal  Started    10s   kubelet            Started container nginx
```

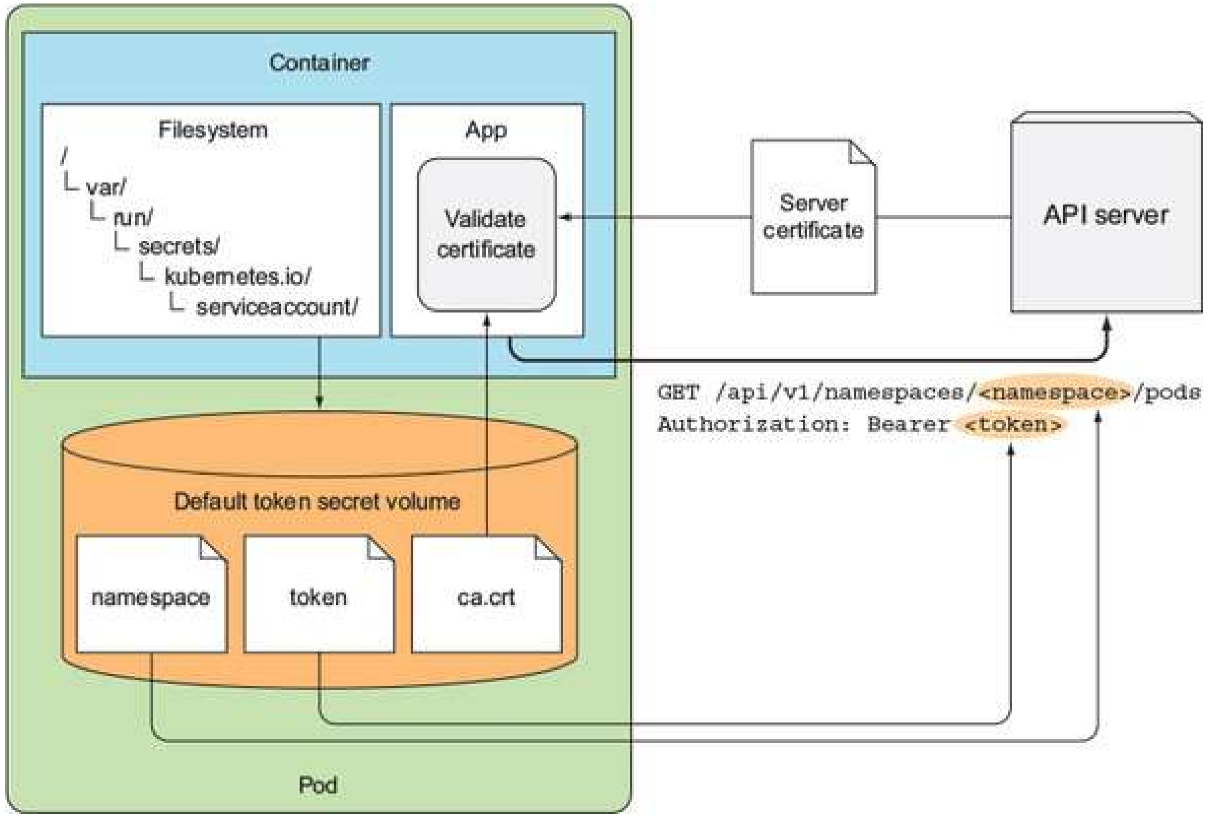
///

Kubernetes

Pod ~ API-Server

Default token secret volume

- namespace / token / ca.crt

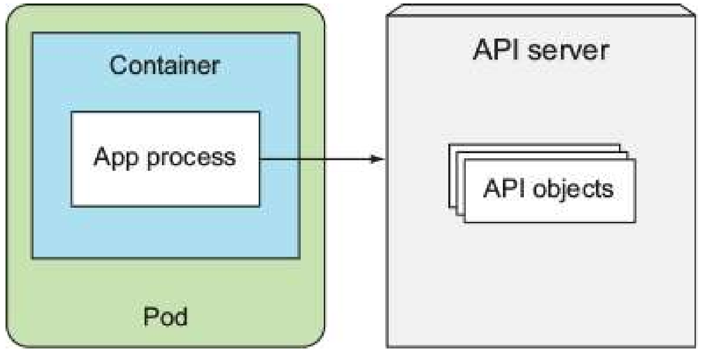


```
GET /api/v1/namespaces/<namespace>/pods
Authorization: Bearer <token>
```

※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-8/104

Create Pod

- Pod(실제로는 Container)에서 API-Server에 접근하는 것을 직접 해보기 위해 실습용 Pod를 하나 생성해보자



pod-ubuntu.yaml

```
apiVersion: v1
kind: Pod

metadata:
  name: ubuntu

spec:
  containers:
  - image: ubuntu:20.04
    name: ubuntu
    command: ["/bin/sleep", "3650d"]
```

```
remote ❯ git clone https://github.com/whatwant-school/advanced-kubernetes.git
remote ❯ cd advanced-kubernetes

remote ❯ kubectl create -f 07-week/api-server/pod-ubuntu.yaml

pod/ubuntu created

remote ❯ kubectl get services

NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.233.0.1    <none>        443/TCP   41d
```

※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-8/62

ca.crt

```
remote ❯ kubectl exec -it ubuntu -- /bin/bash


root@ubuntu:/# apt update


root@ubuntu:/# apt install curl
```

curl 설치

```
root@ubuntu:/# env ¦ grep KUBERNETES_SERVICE

KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_HOST=10.233.0.1
```

API-Server 정보가 환경변수로 등록되어 있다. (Service)

```
root@ubuntu:/# curl https://kubernetes

curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.haxx.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
how to fix it, please visit the web page mentioned above.
```

SSL 인증서 이슈로 이렇게 나온다.

```
root@ubuntu:/# ls /var/run/secrets/kubernetes.io/serviceaccount

ca.crt  namespace  token
```

```
root@ubuntu:/# curl --cacert \
/var/run/secrets/kubernetes.io/serviceaccount/ca.crt https://kubernetes

{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
}
```

SSL 인증서를 이렇게 명시적으로 알려주면 해결된다.

환경변수 `CURL_CA_BUNDLE`에 경로를 등록해 놓으면 자동으로 참조한다.

```
root@ubuntu:/# export \
CURL_CA_BUNDLE=/var/run/secrets/kubernetes.io/serviceaccount/ca.crt

root@ubuntu:/# curl https://kubernetes

{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/\"",
  "reason": "Forbidden",
  "details": {
  },
  "code": 403
}
```

인증서 문제는 해결이 되었지만, Authentication/Authorization 문제는 남아있다.

token

```
root@ubuntu:/# export TOKEN=\
$(cat /var/run/secrets/kubernetes.io/serviceaccount/token)
```

TOKEN 값을 header에 넣어서 API-Server와 통신

```
root@ubuntu:/# curl -H "Authorization: Bearer $TOKEN" \
https://kubernetes/api/v1/namespaces/default/pods

{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "pods is forbidden: User
\"system:serviceaccount:default:default\" cannot list resource \"pods\" in
API group \"\" in the namespace \"default\"",
  "reason": "Forbidden",
  "details": {
    "kind": "pods"
  },
  "code": 403
```

하지만, 권한이 부족

```
remote ❯ kubectl create clusterrolebinding \
permissive-binding --clusterrole=cluster-admin \
--group=system:serviceaccounts

clusterrolebinding.rbac.authorization.k8s.io/permissive-binding created
```

default 계정에 cluster-admin 권한 binding (학습을 위한 임시방편)

```
root@ubuntu:/# curl -H "Authorization: Bearer $TOKEN" \
https://kubernetes/api/v1/namespaces/default/pods

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "1615795"
  },
  "items": [
    {
      "metadata": {
        "name": "ubuntu",
        "namespace": "default",
        "uid": "fad80820-ae8f-4cb0-9ea2-d8bbce66ea6f",
        "resourceVersion": "1611830",
        "creationTimestamp": "2022-03-04T22:03:42Z",
        "annotations": {
          "cni.projectcalico.org/containerID":
"0334ab86fb943ea9155a259ee13b4e72a0b1faa8b6de21e42231b8267dcbb2f2",
          "cni.projectcalico.org/podIP": "10.233.103.84/32",
          "cni.projectcalico.org/podIPs": "10.233.103.84/32"
        },
        "managedFields": [
          {
            "manager": "calico",
            "operation": "Update",
            "apiVersion": "v1",
            "time": "2022-03-04T22:03:42Z",
            "fieldsType": "FieldsV1",
            "fieldsV1":
{"f:metadata":{"f:annotations":{".":{},"f:cni.projectcalico.org/containerID":{},"f:cni.pr
ojectcalico.org/podIP":{},"f:cni.projectcalico.org/podIPs":{}}}},
...
```

이제 제대로 결과를 돌려준다.

namespace

```
root@ubuntu:/# export NS=$(cat /var/run/secrets/kubernetes.io/serviceaccount/namespace)

root@ubuntu:/# echo $NS

default
```

Pod가 있는 namespace 정보를 확인할 수 있다.

```
root@ubuntu:/# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/namespaces/$NS/pods

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "1616235"
  },
  "items": [
    {
      "metadata": {
        "name": "ubuntu",
        "namespace": "default",
        "uid": "fad80820-ae8f-4cb0-9ea2-d8bbce66ea6f",
        "resourceVersion": "1611830",
        "creationTimestamp": "2022-03-04T22:03:42Z",
        "annotations": {
          "cni.projectcalico.org/containerID": "0334ab86fb943ea9155a259ee13b4e72a0b1faa8b6de21e42231b8267dcbb2f2",
          "cni.projectcalico.org/podIP": "10.233.103.84/32",
          "cni.projectcalico.org/podIPs": "10.233.103.84/32"
        },
        "managedFields": [
          {
            "manager": "calico",
            "operation": "Update",
...
```

///