Managing Kubernetes

2021-05-19
written by whatwant

Agenda

Chapter1. Kubernetes Overview

1주차: **Docker and Kubernetes**

Chapter2. Kubernetes Core

2주차: **Environment & POD**

3주차: **Replication and other controllers**

4주차: **Services**

5주차: **Volumes (Service 보충 수업 포함)**

6주차: **ConfigMaps and Secrets & Kubernetes REST API**

7주차: **Deployment & StatefulSet**

Chapter3. Kubernetes Managing

8주차: **Authentication and User Management & Authorization & Admission Control**

9주차: **Networking**

10주차: **Monitoring**

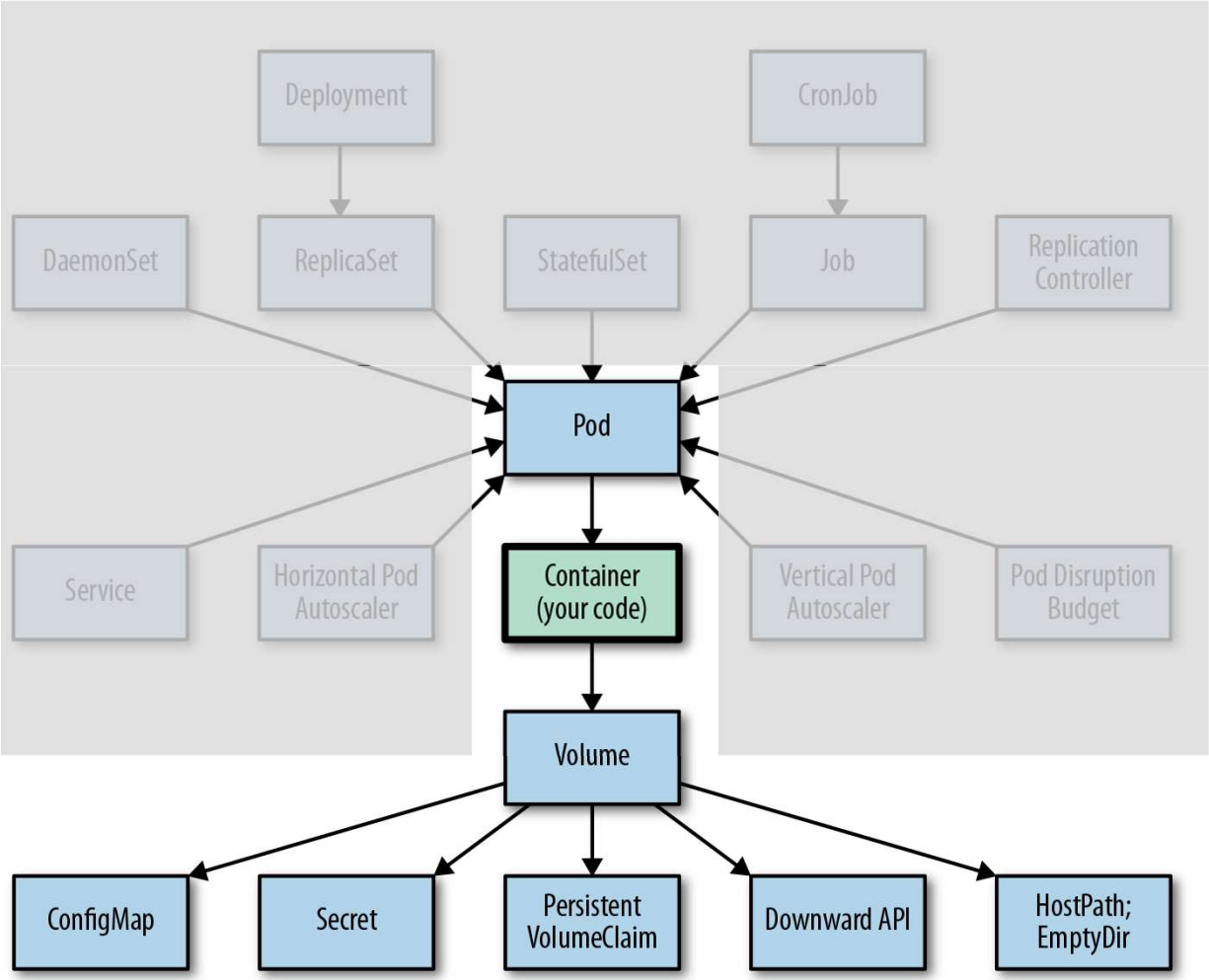
11주차: **Disaster Recovery**

※ 참고 : https://home.modulabs.co.kr/product/managing-kubernetes/

6 week
ConfigMaps and Secrets
& Kubernetes REST API

Today ...

Volume



[권장] 업무 환경 별도로 구축하기

- Master Node에서 작업하는 것은 추천하지 않음

  . 특히 Docker 환경 등을 맞추려면 번거로움 존재, 업무를 위한 추가 설치 작업하기에도 부담스러움

- 별도로 Ubuntu Desktop 설치하고 원격으로 작업하는 것을 추천

  . 18.04 이상 버전 추천

```
[ kubectl 설치 ]
> wget https://packages.cloud.google.com/apt/pool/kubectl_1.20.1-00_amd64_b927311062e6a4610d9ac3bc8560457ab23fbd697a3052c394a1d7cc9e46a17d.deb
> sudo dpkg --install kubectl_1.20.1-00_amd64_b927311062e6a4610d9ac3bc8560457ab23fbd697a3052c394a1d7cc9e46a17d.deb

[ master node 연결 ]
> mkdir ~/.kube
> scp <account>@<masternode ip>:/home/<account>/.kube/config ~/.kube/

[ docker 설치 ]
> wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/containerd.io_1.4.3-1_amd64.deb
> wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-ce-cli_20.10.1\~3-0\~ubuntu-focal_amd64.deb
> wget https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-ce_20.10.1\~3-0\~ubuntu-focal_amd64.deb

> sudo dpkg --install containerd.io_1.4.3-1_amd64.deb
> sudo dpkg --install docker-ce-cli_20.10.1\~3-0\~ubuntu-focal_amd64.deb
> sudo dpkg --install docker-ce_20.10.1\~3-0\~ubuntu-focal_amd64.deb
```

configMap

Environments

- 환경 변수 선언 및 Argument 사용하기

. Kubernetes YAML vs Dockerfile : K8s YAML 선언할 때 명시하는 방법 vs Docker Image 만들 때 명시하는 방법

→ Dockerfile에서 명시하게 되면 환경 변수 또는 arguments 값이 변경될 때 이미지를 변경해야 함 (비추!)

01-env_arg-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: print-greeting
spec:
  containers:
  - name: env-print-demo
    image: bash
    env:
    - name: GREETING
      value: "Warm greetings to"
    - name: HONORIFIC
      value: "The Most Honorable"
    - name: NAME
      value: "Kubernetes"
    command: ["echo"]
    args: ["$(GREETING) $(HONORIFIC) $(NAME)"]

  restartPolicy: OnFailure
```

K8s에서 환경 변수는 이렇게 사용하면 됨

```
❯ kubectl apply -f 01-env_arg-pod.yaml
pod/print-greeting created

❯ kubectl get pods -o wide
NAME             READY   STATUS      RESTARTS   AGE     IP              NODE      NOMINATED NODE   READINESS GATES
print-greeting   0/1     Completed   0          2m18s   10.233.103.64   worker2   <none>           <none>

❯ kubectl logs print-greeting
Warm greetings to The Most Honorable Kubernetes
```

※ 참고 : https://kubernetes.io/ko/docs/tasks/inject-data-application/define-environment-variable-container/

Arguments – 1/2

- Container의 arguments를 변경하는 사례를 위해 Docker Image 생성 및 업로드

. 10초에 한 번씩 실행

※ K8s Master Node에서 docker build 수행하면 네트워 이슈로 실패한다
→ 별도의 작업 PC를 이용하는 것을 추천!

02-fortuneloop.sh

```
#!/bin/bash
trap "exit" SIGINT
INTERVAL=$1
echo Configured to generate new fortune every $INTERVAL seconds
mkdir -p /var/htdocs
while :
do
  echo $(date) Writing fortune to /var/htdocs/index.html
  /usr/games/fortune > /var/htdocs/index.html
  sleep $INTERVAL
done
```

02-Dockerfile-fortune

```
FROM ubuntu:latest

RUN apt-get update
RUN apt-get -y install fortune

ADD 02-fortuneloop.sh /bin/fortuneloop.sh
RUN chmod +x /bin/fortuneloop.sh

ENTRYPOINT ["/bin/fortuneloop.sh"]
CMD [“10”]
```

Dockerfile에서 ENTRYPOINT와 CMD 구분하는 것을 추천!

```
❯ docker build -t fortune:v1.0 -f 02-Dockerfile-fortune .

❯ docker run -it --name fortune fortune:v1.0
Configured to generate new fortune every 10 seconds
Mon May 17 08:11:28 UTC 2021 Writing fortune to /var/htdocs/index.html
Mon May 17 08:11:39 UTC 2021 Writing fortune to /var/htdocs/index.html
Mon May 17 08:11:49 UTC 2021 Writing fortune to /var/htdocs/index.html
```

```
< DockerHub 內 fortune repository 생성 한 뒤 아래 내역 진행 >

❯ docker login

❯ docker tag fortune:v1.0 whatwant/fortune:v1.0

❯ docker push whatwant/fortune:v1.0
```

※ 참고 : https://github.com/luksa/kubernetes-in-action/tree/master/Chapter07

Arguments – 2/2

- Kubernetes YAML 정의할 때 Dockerfile에서 사용한 CMD 부분을 대체할 수 있다.

02-fortune-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: fortune
  labels:
    app: fortune
spec:
  containers:
  - name: html-generator
    image: whatwant/fortune:v1.0
    args: ["5"]
    volumeMounts:
    - name: web-fortune
      mountPath: /var/htdocs

  - name: web-server
    image: nginx:alpine
    volumeMounts:
    - name: web-fortune
      mountPath: /usr/share/nginx/html
      readOnly: true
    ports:
    - containerPort: 80

  volumes:
  - name: web-fortune
    emptyDir: {}
```

02-fortune-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: fortune-svc

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30444

  selector:
    app: fortune
```

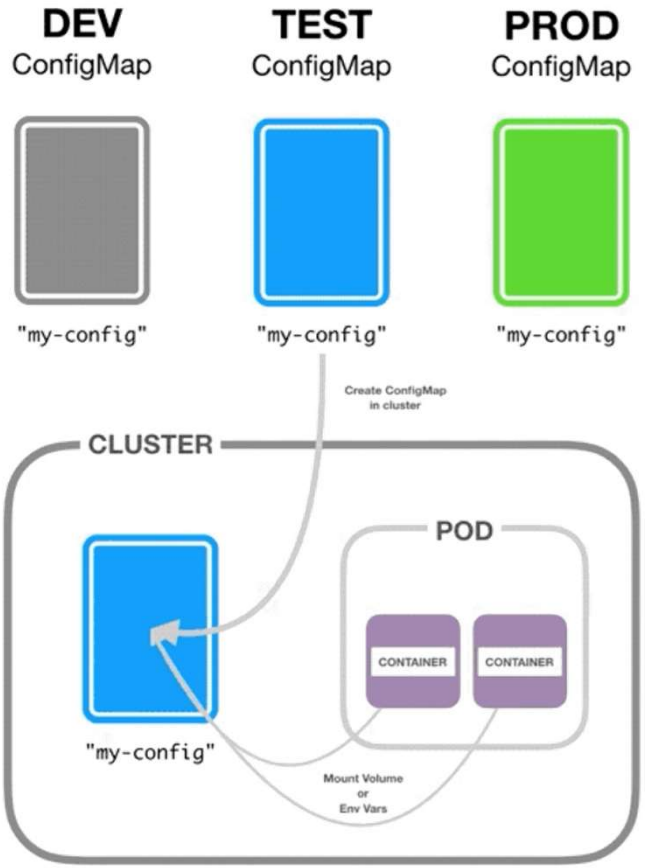
```
❯ kubectl apply -f 02-fortune-svc.yaml

❯ kubectl apply -f 02-fortune-pod.yaml

❯ kubectl logs fortune -c html-generator
Configured to generate new fortune every 5 seconds
Mon May 17 10:01:17 UTC 2021 Writing fortune to /var/htdocs/index.html
Mon May 17 10:01:22 UTC 2021 Writing fortune to /var/htdocs/index.html
Mon May 17 10:01:27 UTC 2021 Writing fortune to /var/htdocs/index.html
```

Dockerfile에서는 "10" 초였는데,
K8s에서 지정한 "5" 초로 된 것을 확인할 수 있다.

What is … configMap – 1/2

- 독립적인 리소스로 동일한 Pod라도 여러 개의 configMap을 선택하여 사용 가능

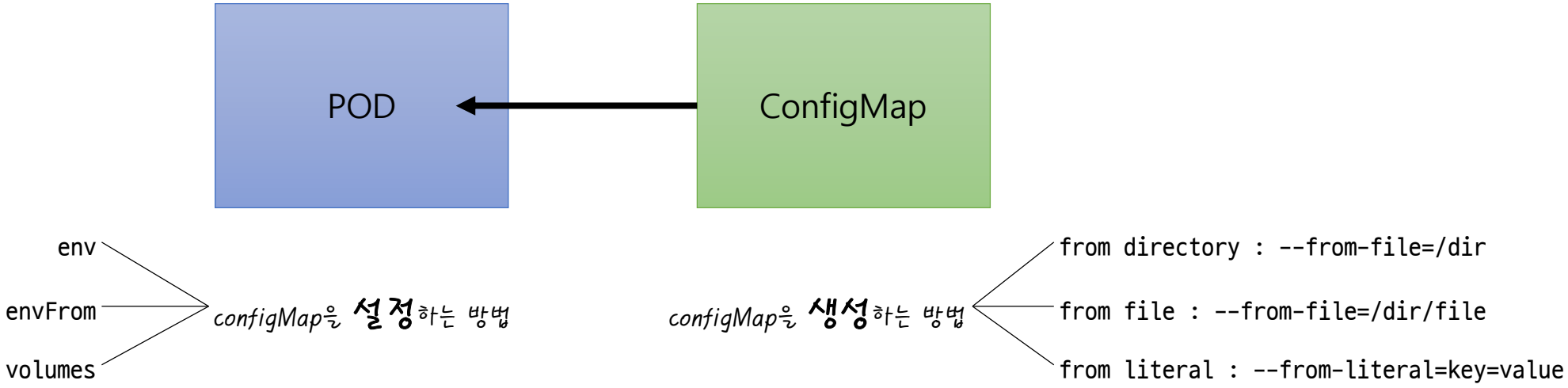
. configMap만 변경하여 Pod를 손쉽게 테스트



※ 참고 : https://timewizhan.tistory.com/entry/Kubernetes-ConfigMap

What is … configMap – 2/2

- configMap을 생성해 놓으면, Pod에서 가져다 사용하는 형태



How to … configMap : yaml – 1/2

03-container-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: container-configmap

data:
  map-hash-bucket-size: "128"
  ssl-protocols: SSLv2
```

03-volume-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: volume-configmap

data:
  index.html: |-
    <html>
    <h1>Hello from ConfigMap</h1>
    </html>
```

03-nginx-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80

    volumeMounts:
    - mountPath: /usr/share/nginx/html/index.html
      name: nginx-volume
      subPath: index.html

    envFrom:
      - configMapRef:
          name: container-configmap

  volumes:
  - name: nginx-volume
    configMap:
      name: volume-configmap
```

configMap 2개 선언해서

다양한 방법으로 사용하는 예제

※ 참고 : https://github.com/tncad/k8s-app-cpd/blob/a41111ae84c45fecb5c21ba96fec0ee113beff81/021-kustomize/full/deployment.yaml

How to … configMap : yaml – 2/2

```
❯ kubectl apply -f 03-container-configmap.yaml
❯ kubectl apply -f 03-volume-configmap.yaml
❯ kubectl apply -f 03-nginx-pod.yaml
❯ kubectl apply -f 03-nginx-svc.yaml

>kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          23m   10.233.103.67   worker2   <none>           <none>

❯ kubectl exec -it nginx -- printenv
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=nginx
TERM=xterm
map-hash-bucket-size=128
ssl-protocols=SSLv2
NGINX_SVC_PORT=tcp://10.233.36.129:80
NGINX_SVC_PORT_80_TCP_ADDR=10.233.36.129
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.233.0.1:443
NGINX_SVC_SERVICE_HOST=10.233.36.129
NGINX_SVC_PORT_80_TCP_PROTO=tcp
. . .
```

03-nginx-svc.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc

spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
    nodePort: 30501

  selector:
    app: nginx
```

How to ... configMap : cli

- 앞에서 YAML로 진행한 내역과 동일한 결과가 나온다.

```
❯ kubectl get configmaps -o wide
NAME                DATA   AGE
kube-root-ca.crt    1      13d

❯ kubectl get pods -o wide
No resources found in default namespace.
```

디렉토리를 지정할 수도 있고

```
❯ kubectl create configmap container-configmap --from-file=./container/
configmap/container-configmap created

❯ kubectl create configmap volume-configmap --from-file=./volume/index.html
configmap/volume-configmap created
```

파일로 지정할 수도 있다

```
❯ kubectl get configmaps -o wide
NAME                  DATA   AGE
container-configmap   2      31s
volume-configmap      1      18s
kube-root-ca.crt      1      13d

❯ kubectl apply -f 03-nginx-pod.yaml

❯ kubectl apply -f 03-nginx-svc.yaml
```

container/map-hash-bucket-size

128

container/ssl-protocols

SSLv2

```
❯ tree ./container
./container
├── map-hash-bucket-size
└── ssl-protocols
```

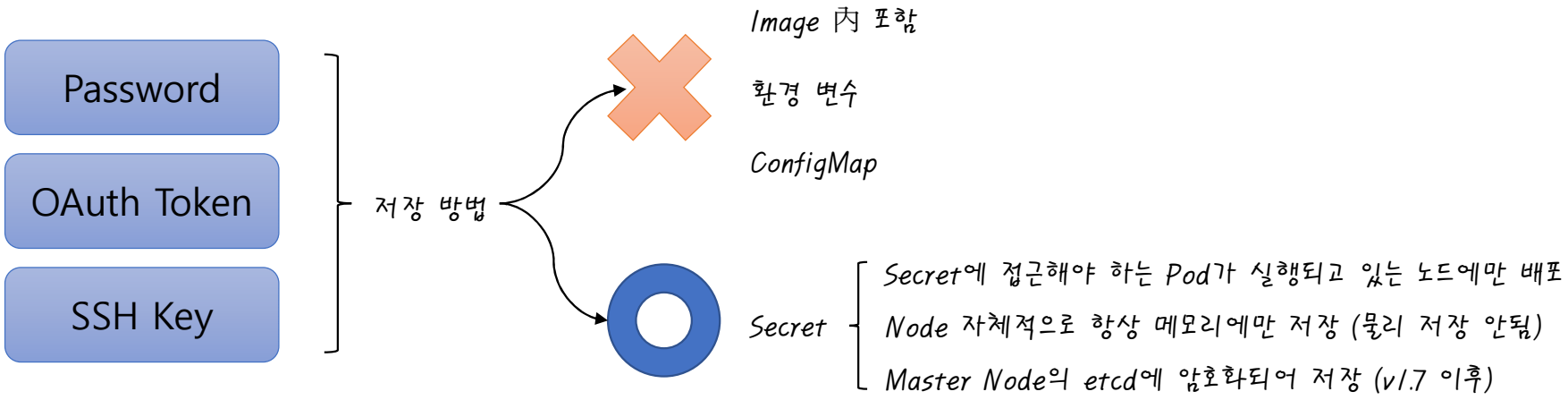
volume/index.html

```
<html>
<h1>Hello from ConfigMap</h1>
</html>
```

```
❯ tree ./volume
./volume
└── index.html
```

Secret

Why ... secret


Password
OAuth Token
SSH Key
저장 방법
Image 內 포함
환경 변수
ConfigMap
Secret
Secret에 접근해야 하는 Pod가 실행되고 있는 노드에만 배포
Node 자체적으로 항상 메모리에만 저장 (물리 저장 안됨)
Master Node의 etcd에 암호화되어 저장 (v1.7 이후)

Check … secret – 1/2

- default로 존재하고 있는 secret이 있다.

```
>kubectl get secrets -o wide
NAME                  TYPE                                  DATA   AGE
default-token-4b9h2   kubernetes.io/service-account-token   3      13d

> kubectl describe secrets default-token-4b9h2
Name:         default-token-4b9h2
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: 11609ba9-401a-4490-a430-c648bbcc2ef4

Type:  kubernetes.io/service-account-token

Data
====
namespace:  7 bytes
token:
eyJhbGciOiJSUzI1NiIsImtpZCI6Img4LVYtQ20wRVhYOVNXak5tMENpUnRueHlGU05CLUJVczZhYXJwX3pRZmMifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYW
. . . pwLW3pQN2ygQZm_jFtedwJqwkRWzTQTTc4P6b-Gb19VYHZToKQRM8Y6tov7HkIXyRRR3390rJdw9M6q3vLQH860YBEsklD3JdFSLXcRUsN8Ww
ca.crt:     1066 bytes
```

kubernetes API와 통신하기 위해
필요한 모든 것 !!!

namespace
token
ca.crt

Check … secret – 2/2

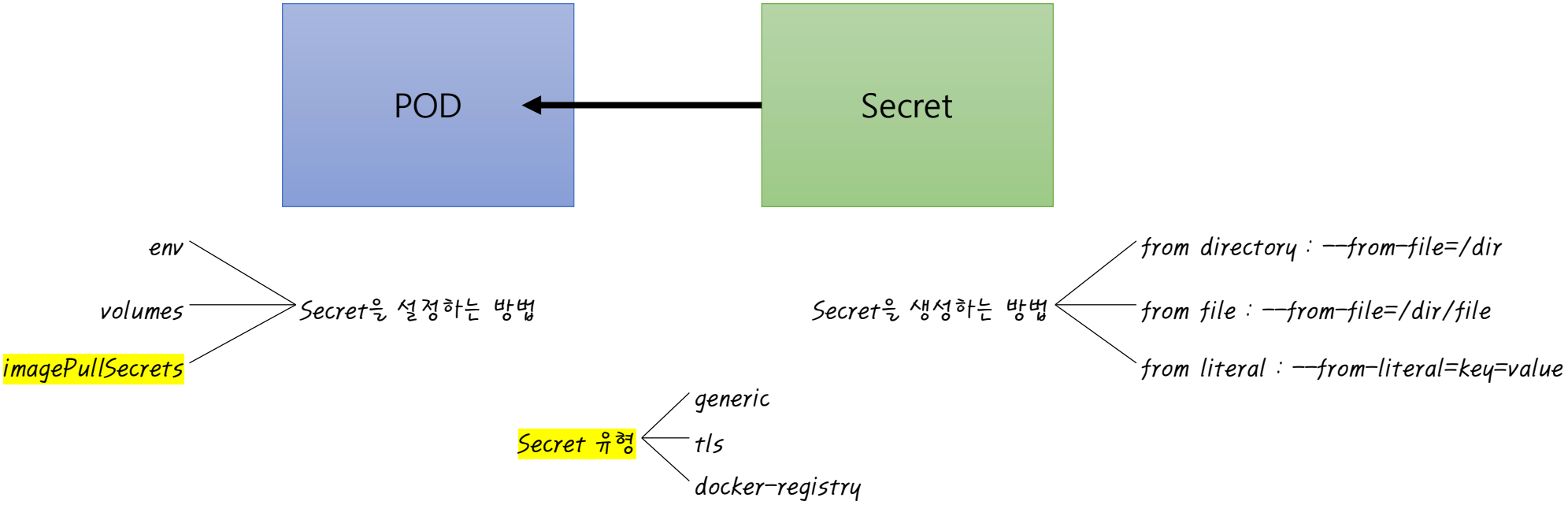
- Pod를 생성하면 기본적으로 포함된다.

```
❯ kubectl run web --image=nginx:latest
pod/web created

❯ kubectl describe pods web
Name:         web
Namespace:    default
Priority:     0
Node:         worker2/192.168.100.113
Start Time:   Mon, 17 May 2021 23:38:24 +0900
Labels:       run=web
Annotations:  cni.projectcalico.org/podIP: 10.233.103.71/32
              cni.projectcalico.org/podIPs: 10.233.103.71/32
Status:       Running
IP:           10.233.103.71
IPs:
  IP:  10.233.103.71
Containers:
  web:
    Container ID:
docker://6f68639da0426655dcde7ee6153ce25b2a3bf9576a193b3664f533656053819d
    Image:          nginx:latest
```

```
Image ID:       docker-pullable://nginx@sha256:df13abe. . .7902331dcea50f4f1f2
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 17 May 2021 23:38:28 +0900
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-4b9h2 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-4b9h2:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-4b9h2
    Optional:    false
QoS Class:       BestEffort
. . .
```

What is ... Secret

- configMap과 상당히 유사하다.



How to … Secret : docker registry – 1/3

- Secret 유형 中 Docker Registry 관련된 사항을 알아보자

. 이는 DockerHub에서 private으로 등록된 이미지를 다운로드 받을 수 있도록 해준다

04-index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Docker Nginx</title>
</head>
<body>
  <h2>Hello from Nginx container</h2>
</body>
</html>
```

04-Dockerfile

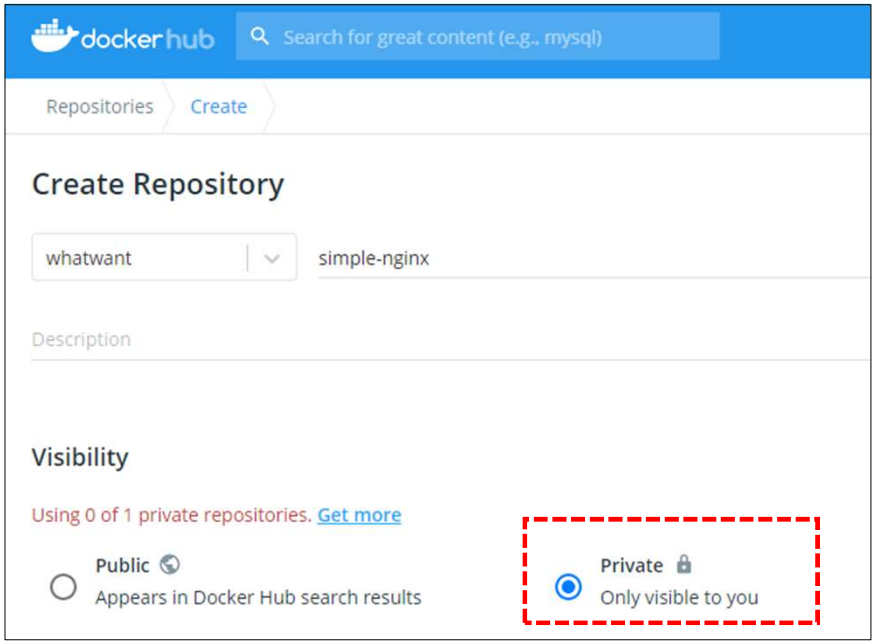
```
FROM nginx:latest

COPY ./04-index.html /usr/share/nginx/html/index.html
```

```
❯ docker login

❯ docker build -t whatwant/simple-nginx:v0.1 -f 04-Dockerfile .
```

local에 build 한 뒤, tagging 하고 push를 했었는데,

이렇게 바로 DockerHub 用 tagging으로 build를 바로 해버릴 수도 있다.

```
❯ docker push whatwant/simple-nginx:v0.1
```



How to … Secret : docker registry – 2/3

- private repository에 있는 이미지는 그냥 접근할 수 없다.

04-private-fail-pod.yaml

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

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

```
❯ kubectl apply -f 04-private-fail-pod.yaml
pod/simple-nginx created

❯ kubectl get pods -o wide
NAME           READY   STATUS         RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
simple-nginx   0/1     ErrImagePull   0          80s   10.233.103.72   worker2   <none>           <none>

❯ kubectl describe pods simple-nginx
Name:         simple-nginx
Namespace:    default
. . .
Events:
  Type     Reason     Age                             From               Message
  ----     ------     ----                            ----               -------
  Normal   Scheduled  <invalid>                       default-scheduler  Successfully assigned default/simple-nginx to worker2
  Normal   BackOff    <invalid> (x4 over <invalid>)   kubelet            Back-off pulling image "whatwant/simple-nginx:v0.1"
  Warning  Failed     <invalid> (x4 over <invalid>)   kubelet            Error: ImagePullBackOff
  Normal   Pulling    <invalid> (x4 over <invalid>)   kubelet            Pulling image "whatwant/simple-nginx:v0.1"
  Warning  Failed     <invalid> (x4 over <invalid>)   kubelet            Failed to pull image "whatwant/simple-nginx:v0.1": rpc error: code = Unknown desc =
Error response from daemon: pull access denied for whatwant/simple-nginx, repository does not exist or may require 'docker login': denied: requested access to
the resource is denied
  Warning  Failed     <invalid> (x4 over <invalid>)   kubelet            Error: ErrImagePull
```

secret 정보 없이

private repo를 접근하기에 에러 발생!

How to … Secret : docker registry – 3/3

04-private-success-pod.yaml

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

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

  imagePullSecrets:
    - name: simple-credential
```

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

```
❯ kubectl create secret docker-registry simple-credential --docker-username=whatwant --docker-password='xxx' --docker-email='whatwant@gmail.com'
secret/simple-credential created

❯ kubectl apply -f 04-private-success-pod.yaml
pod/simple-nginx created

❯ kubectl get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE   IP              NODE      NOMINATED NODE   READINESS GATES
simple-nginx   1/1     Running   0          21s   10.233.103.76   worker2   <none>           <none>
```

Type of Secret

빌트인 타입	사용처
Opaque	임의의 사용자 정의 데이터
kubernetes.io/service-account-token	서비스 어카운트 토큰
kubernetes.io/dockercfg	직렬화 된(serialized) ~/.dockercfg 파일
kubernetes.io/dockerconfigjson	직렬화 된 ~/.docker/config.json 파일
kubernetes.io/basic-auth	기본 인증을 위한 자격 증명(credential)
kubernetes.io/ssh-auth	SSH를 위한 자격 증명
kubernetes.io/tls	TLS 클라이언트나 서버를 위한 데이터
bootstrap.kubernetes.io/token	부트스트랩 토큰 데이터

※ 참고 : https://kubernetes.io/ko/docs/concepts/configuration/secret/

How to … Secret : git clone – 1/2

- private repository를 clone 받는 pod를 구성해보자

[아래 sample 진행을 위한 precondition]
- github.com 內 private repository가 존재 한다 (아니라면, 각자의 조건에 맞는 서버 정보를 알고 있어야 한다)
- 나의 계정에 있는 SSH Key로 접근 가능한 repository다. (아니라면, 접근 가능한 SSH Key를 따로 갖고 있다)

```
❯ ssh-keyscan github.com > ./05-known_hosts
# github.com:22 SSH-2.0-babeld-74336b10
# github.com:22 SSH-2.0-babeld-74336b10
# github.com:22 SSH-2.0-babeld-74336b10

❯ kubectl create secret generic git-creds --from-file=ssh=$HOME/.ssh/id_rsa --from-file=known_hosts=./05-known_hosts
secret/git-creds created
```

ssh, known_hosts 2개의 key를 갖고 있는 secret을 생성한 것이다.

```
❯ kubectl get secrets -o wide
NAME                  TYPE                                  DATA   AGE
default-token-4b9h2   kubernetes.io/service-account-token   3      13d
git-creds             Opaque                                2      14s
```

※ 참고 : https://github.com/kubernetes/git-sync

How to … Secret : git clone – 2/2

05-git-sync-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: git-sync
spec:
  volumes:
  - name: git-secret
    secret:
      secretName: git-creds
      defaultMode: 0400
  containers:
  - name: git-sync
    image: k8s.gcr.io/git-sync:v3.1.5

    args:
     - "--ssh"
     - "--repo=git@github.com:whatwant-school/private-repo"
     - "--dest=private-repo"
     - "--branch=main"
     - "--depth=1"

    securityContext:
      runAsUser: 65533 # git-sync user

    volumeMounts:
    - name: git-secret
      mountPath: /etc/git-secret
      readOnly: true

  securityContext:
    fsGroup: 65533 # to make SSH key readable
```

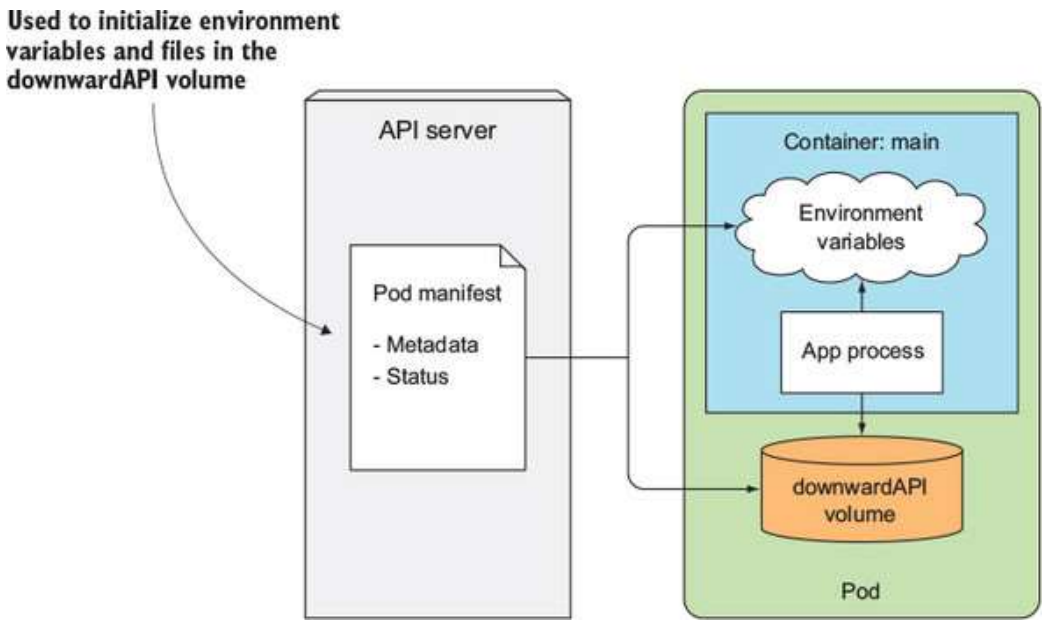
```
❯ kubectl apply -f 05-git-sync-pod.yaml

$ ls -al
total 16
drwxrwxrwt 1 root     root  4096 May 18 02:10 .
drwxr-xr-x 1 root     root  4096 May 18 02:10 ..
drwx------ 2 git-sync 65533 4096 May 18 02:10 .ssh
drwxr-xr-x 4 git-sync 65533 4096 May 18 02:10 git
$ cd git
$ ls -al
total 16
drwxr-xr-x 4 git-sync 65533 4096 May 18 02:10 .
drwxrwxrwt 1 root     root  4096 May 18 02:10 ..
drwxr-xr-x 9 git-sync 65533 4096 May 18 02:10 .git
lrwxrwxrwx 1 git-sync 65533   44 May 18 02:10 private-repo -> rev-56af7aa4197fee5da30ccdd562cc2ea5e7
drwxr-xr-x 2 git-sync 65533 4096 May 18 02:10 rev-56af7aa4197fee5da30ccdd562cc2ea5e7
$ pwd
/tmp/git
$ ls -al private-repo/
total 16
drwxr-xr-x 2 git-sync 65533 4096 May 18 02:10 .
drwxr-xr-x 4 git-sync 65533 4096 May 18 02:10 ..
-rw-r--r-- 1 git-sync 65533   71 May 18 02:10 .git
-rw-r--r-- 1 git-sync 65533   60 May 18 02:10 README.md
$ exit
```

Downward API

What is … Downward API – 1/2

- Pod의 IP, 호스트 노드 이름, Pod 자체의 이름과 같이 실행 시점까지 알려지지 않은 데이터를 얻기 위한 방법

  . Downward API는 애플리케이션이 호출해서 데이터를 가져오는 REST Endpoint와는 다르다.



※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-8/11

What is … Downward API – 2/2

Information available via	item	comment
fieldRef:	metadata.name	the pod's name
	metadata.namespace	the pod's namespace
	metadata.uid	the pod's UID
	metadata.labels['<KEY>']	the value of the pod's label <KEY> (for example, metadata.labels['mylabel'])
	metadata.annotations['<KEY>']	the value of the pod's annotation <KEY> (for example, metadata.annotations['myannotation'])
resourceFieldRef:	A Container's CPU limit	
	A Container's CPU request	
	A Container's memory limit	
	A Container's memory request	
	A Container's hugepages limit	providing that the DownwardAPIHugePages feature gate is enabled
	A Container's hugepages request	providing that the DownwardAPIHugePages feature gate is enabled
	A Container's ephemeral-storage limit	
	A Container's ephemeral-storage request	
downwardAPI volume fieldRef:	metadata.labels	all of the pod's labels, formatted as label-key="escaped-label-value" with one label per line
	metadata.annotations	all of the pod's annotations, formatted as annotation-key="escaped-annotation-value" with one annotation per line
environment variables:	status.podIP	the pod's IP address
	spec.serviceAccountName	the pod's service account name, available since v1.4.0-alpha.3
	spec.nodeName	the node's name, available since v1.4.0-alpha.3
	status.hostIP	the node's IP, available since v1.7.0-alpha.1

※ 참고 : https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/#capabilities-of-the-downward-api

How to … Downward API : Pod – 1/2

https://k8s.io/examples/pods/inject/dapi-volume.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: kubernetes-downwardapi-volume-example
  labels:
    zone: us-est-coast
    cluster: test-cluster1
    rack: rack-22
  annotations:
    build: two
    builder: john-doe

spec:
  containers:
    - name: client-container
      image: k8s.gcr.io/busybox
      command: ["sh", "-c"]

      args:
      - while true; do
          if [[ -e /etc/podinfo/labels ]]; then
            echo -en '₩n₩n'; cat /etc/podinfo/labels; fi;
          if [[ -e /etc/podinfo/annotations ]]; then
            echo -en '₩n₩n'; cat /etc/podinfo/annotations; fi;
          sleep 5;
        done;
```

```
volumeMounts:
        - name: podinfo
          mountPath: /etc/podinfo

 volumes:
   - name: podinfo
     downwardAPI:
       items:
         - path: "labels"
           fieldRef:
             fieldPath: metadata.labels
         - path: "annotations"
           fieldRef:
             fieldPath: metadata.annotations
```

※ 참고 : https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/

How to … Downward API : Pod – 2/2

```
❯ kubectl apply -f https://k8s.io/examples/pods/inject/dapi-volume.yaml
pod/kubernetes-downwardapi-volume-example created

❯ kubectl logs kubernetes-downwardapi-volume-example
cluster="test-cluster1"
rack="rack-22"
zone="us-est-coast"

build="two"
builder="john-doe"
kubectl.kubernetes.io/ . . .
. . .
```

Pod의 labels, annotations 정보 출력

```
❯ kubectl exec -it kubernetes-downwardapi-volume-example \
-- cat /etc/podinfo/labels
cluster="test-cluster1"
rack="rack-22"
zone="us-est-coast"%
```

Downward API에 잘 저장되어 있는지 확인

```
❯ kubectl exec -it kubernetes-downwardapi-volume-example – cat\
/etc/podinfo/annotations
build="two"
builder="john-doe"
. . . %

❯ kubectl exec -it kubernetes-downwardapi-volume-example -- ls -laR /etc/podinfo
/etc/podinfo:
total 4
drwxrwxrwt    3 root     root           120 May 18 04:22 .
drwxr-xr-x    1 root     root          4096 May 18 04:22 ..
drwxr-xr-x    2 root     root            80 May 18 04:22 ..2021_05_18_04_22_01.776905510
lrwxrwxrwx    1 root     root            31 May 18 04:22 ..data -> ..2021_05_18_04_22_01.776905510
lrwxrwxrwx    1 root     root            18 May 18 04:21 annotations -> ..data/annotations
lrwxrwxrwx    1 root     root            13 May 18 04:21 labels -> ..data/labels

/etc/podinfo/..2021_05_18_04_22_01.776905510:
total 8
drwxr-xr-x    2 root     root            80 May 18 04:22 .
drwxrwxrwt    3 root     root           120 May 18 04:22 ..
-rw-r--r--    1 root     root          1208 May 18 04:22 annotations
-rw-r--r--    1 root     root            58 May 18 04:22 labels
```

※ 참고 : https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/

How to … Downward API : Container – 1/2

https://k8s.io/examples/pods/inject/dapi-volume-resources.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kubernetes-downwardapi-volume-example-2

spec:
  containers:
    - name: client-container
      image: k8s.gcr.io/busybox:1.24
      command: ["sh", "-c"]

      args:
      - while true; do
          echo -en '₩n';
          if [[ -e /etc/podinfo/cpu_limit ]]; then
            echo -en '₩n'; cat /etc/podinfo/cpu_limit; fi;
          if [[ -e /etc/podinfo/cpu_request ]]; then
            echo -en '₩n'; cat /etc/podinfo/cpu_request; fi;
          if [[ -e /etc/podinfo/mem_limit ]]; then
            echo -en '₩n'; cat /etc/podinfo/mem_limit; fi;
          if [[ -e /etc/podinfo/mem_request ]]; then
            echo -en '₩n'; cat /etc/podinfo/mem_request; fi;
          sleep 5;
        done;
```

```yaml
resources:
  requests:
    memory: "32Mi"
    cpu: "125m"
  limits:
    memory: "64Mi"
    cpu: "250m"

volumeMounts:
  - name: podinfo
    mountPath: /etc/podinfo
```

```yaml
volumes:
  - name: podinfo
    downwardAPI:
      items:
        - path: "cpu_limit"
          resourceFieldRef:
            containerName: client-container
            resource: limits.cpu
            divisor: 1m
        - path: "cpu_request"
          resourceFieldRef:
            containerName: client-container
            resource: requests.cpu
            divisor: 1m
        - path: "mem_limit"
          resourceFieldRef:
            containerName: client-container
            resource: limits.memory
            divisor: 1Mi
        - path: "mem_request"
          resourceFieldRef:
            containerName: client-container
            resource: requests.memory
            divisor: 1Mi
```

※ 참고 : https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/

How to … Downward API : Container – 2/2

```
❯ kubectl apply -f https://k8s.io/examples/pods/inject/dapi-volume-resources.yaml
pod/kubernetes-downwardapi-volume-example-2 created

❯ kubectl exec -it kubernetes-downwardapi-volume-example-2 – cat \ /etc/podinfo/cpu_limit
250%
```

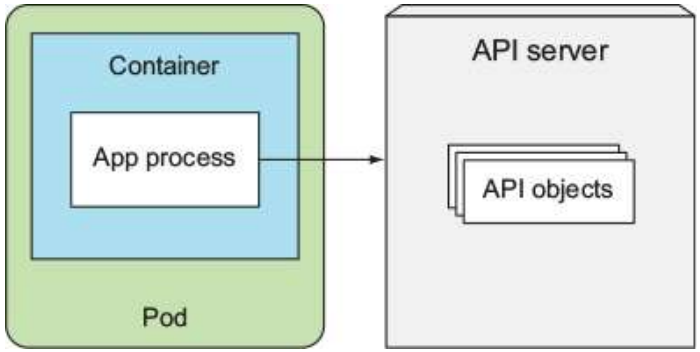
```
❯ kubectl exec -it kubernetes-downwardapi-volume-example-2 -- ls -laR /etc/podinfo
/etc/podinfo:
total 4
drwxrwxrwt    3 root     root           160 May 18 04:38 .
drwxr-xr-x    1 root     root          4096 May 18 04:38 ..
drwxr-xr-x    2 root     root           120 May 18 04:38 ..2021_05_18_04_38_21.294331485
lrwxrwxrwx    1 root     root            31 May 18 04:38 ..data -> ..2021_05_18_04_38_21.294331485
lrwxrwxrwx    1 root     root            16 May 18 04:38 cpu_limit -> ..data/cpu_limit
lrwxrwxrwx    1 root     root            18 May 18 04:38 cpu_request -> ..data/cpu_request
lrwxrwxrwx    1 root     root            16 May 18 04:38 mem_limit -> ..data/mem_limit
lrwxrwxrwx    1 root     root            18 May 18 04:38 mem_request -> ..data/mem_request

/etc/podinfo/..2021_05_18_04_38_21.294331485:
total 16
drwxr-xr-x    2 root     root           120 May 18 04:38 .
drwxrwxrwt    3 root     root           160 May 18 04:38 ..
-rw-r--r--    1 root     root             3 May 18 04:38 cpu_limit
-rw-r--r--    1 root     root             3 May 18 04:38 cpu_request
-rw-r--r--    1 root     root             2 May 18 04:38 mem_limit
-rw-r--r--    1 root     root             2 May 18 04:38 mem_request
```

※ 참고 : https://kubernetes.io/docs/tasks/inject-data-application/downward-api-volume-expose-pod-information/

REST API

What is ... REST API

- Service와 Pod에 관한 정보 → Service 관련 환경 변수, DNS
- 다른 Resource의 정보가 필요하거나 가능한 한 최신 정보에 접근해야 하는 경우 → API 서버와 직접 통신




```
❯ kubectl cluster-info
Kubernetes control plane is running at https://192.168.100.111:6443

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

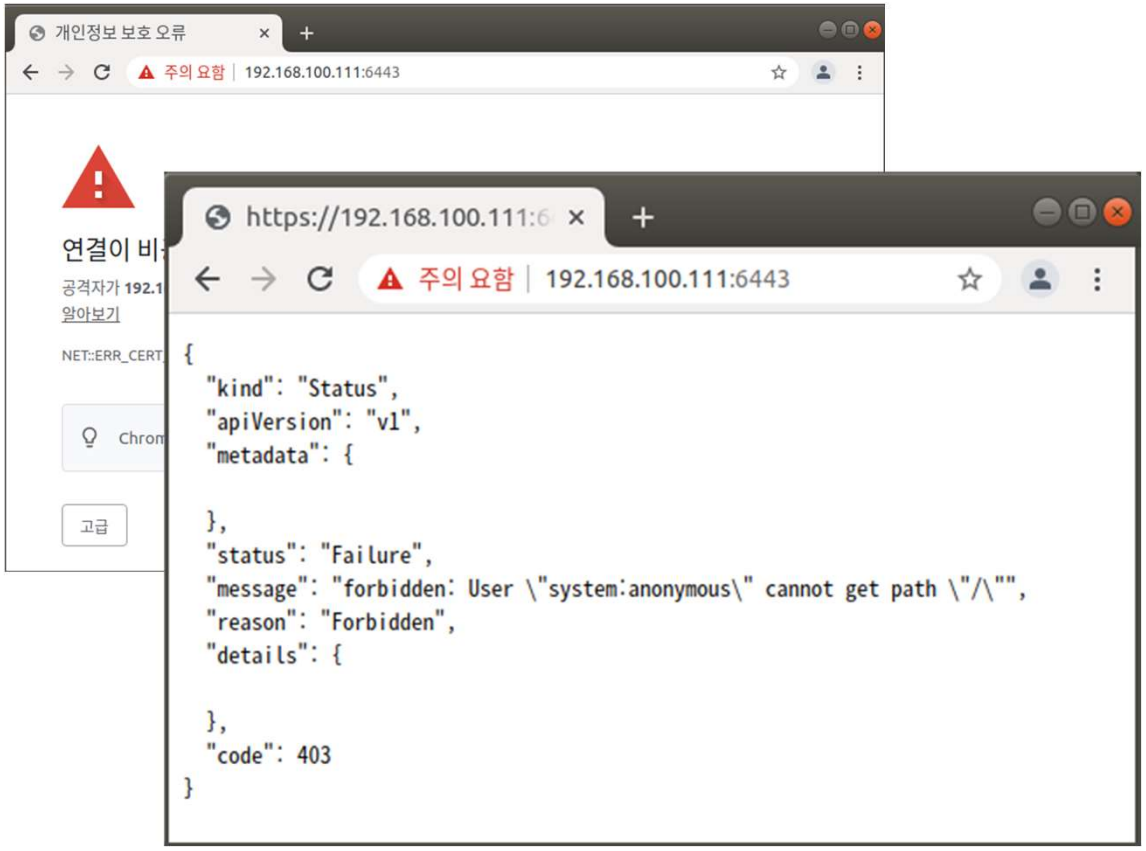
※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-8/62

Where is … REST API – 1/2

- REST API 접근을 위해서는 인증이 필요

  . 그냥 접근하면 당연히 거부

```
❯ curl https://192.168.100.111:6443
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
```



Where is … REST API – 2/2

- `kubectl proxy` 기능을 이용해서 인증 우회 가능

  . 학습 용도로만…

```
❯ kubectl proxy
Starting to serve on 127.0.0.1:8001
```

종료되지 않고 지속 실행

```
❯ curl http://localhost:8001
{
  "paths": [
    "/.well-known/openid-configuration",
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
    "/apis/admissionregistration.k8s.io/v1",
    "/apis/admissionregistration.k8s.io/v1beta1",
    "/apis/apiextensions.k8s.io",
    "/apis/apiextensions.k8s.io/v1",
    "/apis/apiextensions.k8s.io/v1beta1",
. . .
```

터미널 하나 더 띄워서 실행

localhost:8001

localhost:8001


```
{
  "paths": [
    "/.well-known/openid-configuration",
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
    "/apis/admissionregistration.k8s.io/v1",
    "/apis/admissionregistration.k8s.io/v1beta1",
    "/apis/apiextensions.k8s.io",
    "/apis/apiextensions.k8s.io/v1",
    "/apis/apiextensions.k8s.io/v1beta1",
    "/apis/apiregistration.k8s.io",
    "/apis/apiregistration.k8s.io/v1",
    "/apis/apiregistration.k8s.io/v1beta1",
    "/apis/apps",
    "/apis/apps/v1",
    "/apis/authentication.k8s.io",
    "/apis/authentication.k8s.io/v1",
    "/apis/authentication.k8s.io/v1beta1",
    "/apis/authorization.k8s.io",
    "/apis/authorization.k8s.io/v1",
    "/apis/authorization.k8s.io/v1beta1",
    "/apis/autoscaling",
    "/apis/autoscaling/v1",
    "/apis/autoscaling/v2beta1",
    "/apis/autoscaling/v2beta2",
    "/apis/batch",
    "/apis/batch/v1",
    "/apis/batch/v1beta1",
    "/apis/certificates.k8s.io",
    "/apis/certificates.k8s.io/v1",
    "/apis/certificates.k8s.io/v1beta1",
```

How to … REST API – 1/2

localhost:8001/api/v1

localhost:8001/api/v1


```
{
  "kind": "APIResourceList",
  "groupVersion": "v1",
  "resources": [
    {
      "name": "bindings",
      "singularName": "",
      "namespaced": true,
      "kind": "Binding",
      "verbs": [
        "create"
      ]
    },
    {
      "name": "componentstatuses",
      "singularName": "",
      "namespaced": false,
      "kind": "ComponentStatus",
      "verbs": [
        "get",
        "list"
      ],
      "shortNames": [
        "cs"
      ]
    },
    {
      "name": "configmaps",
      "singularName": "",
      "namespaced": true,
      "kind": "ConfigMap",
      "verbs": [
        "create",
        "delete",
```


localhost:8001/apis/batch

localhost:8001/apis/batch


```
{
  "kind": "APIGroup",
  "apiVersion": "v1",
  "name": "batch",
  "versions": [
    {
      "groupVersion": "batch/v1",
      "version": "v1"
    },
    {
      "groupVersion": "batch/v1beta1",
      "version": "v1beta1"
    }
  ],
  "preferredVersion": {
    "groupVersion": "batch/v1",
    "version": "v1"
  }
}
```


localhost:8001/apis/batch/v1


```
{
  "kind": "APIResourceList",
  "apiVersion": "v1",
  "groupVersion": "batch/v1",
  "resources": [
    {
      "name": "jobs",
      "singularName": "",
      "namespaced": true,
      "kind": "Job",
      "verbs": [
        "create",
        "delete",
        "deletecollection",
        "get",
        "list",
        "patch",
        "update",
        "watch"
      ],
      "categories": [
        "all"
      ],
      "storageVersionHash": "mudhfqk/qZY="
    },
    {
      "name": "jobs/status",
      "singularName": "",
      "namespaced": true,
      "kind": "Job",
      "verbs": [
        "get",
        "patch",
        "update"
```

How to … REST API – 2/2

- http://localhost:8001/api/v1/namespaces/default/pods

localhost:8001/api/v1/na

localhost:8001/api/v1/namespaces/default/pods


```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "650407"
  },
  "items": [
    {
      "metadata": {
        "name": "kubernetes-downwardapi-volume-example",
        "namespace": "default",
        "uid": "b1627ccb-8c48-4550-adac-de408ac0b743",
        "resourceVersion": "643727",
        "creationTimestamp": "2021-05-18T04:21:59Z",
        "labels": {
          "cluster": "test-cluster1",
          "rack": "rack-22",
          "zone": "us-est-coast"
        },
        "annotations": {
          "build": "two",
          "builder": "john-doe",
          "cni.projectcalico.org/podIP": "10.233.103.81/32",
          "cni.projectcalico.org/podIPs": "10.233.103.81/32",
          "kubectl.kubernetes.io/last-applied-configuration": "
{\"apiVersion\":\"v1\",\"kind\":\"Pod\",\"metadata\":{\"annotations\":
{\"build\":\"two\",\"builder\":\"john-doe\"},\"labels\":{\"cluster\":\"test-
cluster1\",\"rack\":\"rack-22\",\"zone\":\"us-est-coast\"},\"name\":\"kubernetes-
```

Pod에서 REST API 사용하기 – 1/6

- Pod에서 API Server와 통신하기 위해서는 다음의 3단계가 필요

Pod에서 REST API 사용하기 – 2/6

API Server 찾기

- API Server 찾기 in Host

  . `kubectl get services`로 확인

- API Server 찾기 in Pod

  . 환경 변수

  . DNS

```
❯ kubectl apply -f 06-curl-pod.yaml
pod/curl created

❯ kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.233.0.1   <none>        443/TCP   13d

❯ kubectl exec -it curl -- /bin/bash

root@curl:/# env | grep KUBERNETES_SERVICE
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_HOST=10.233.0.1
root@curl:/# curl https://kubernetes
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: http://curl.haxx.se/docs/sslcerts.html

curl performs SSL certificate verification by default, using a "bundle"
 of Certificate Authority (CA) public keys (CA certs). If the default
 bundle file isn't adequate, you can specify an alternate file
 using the --cacert option.
. . .
```

인증이 안되었기에 통신이 되지는 않는다

06-curl-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: curl

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

Pod에서 curl 실행해보기 위해

curl 지원하는 이미지로 임의의 Pod 생성

Pod에서 REST API 사용하기 – 3/6

API Server 확인

- serviceaccount에 있는 secret 데이터 中 `ca.crt` 사용

```
root@curl:/# ls /var/run/secrets/kubernetes.io/serviceaccount/
ca.crt        namespace  token
```

3종 secret 정보가 있다.

```
root@curl:/# 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
}
```

앞의 메시지와는 차이가 있다.

일단, 서명 관계는 확인이 된 것이다.

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

매번 '--cacert' 옵션을 지정하기가 번거로우니, 환경 변수로 선언 !!!

```
root@curl:/# curl https://kubernetes
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

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

  },
  "code": 403
}
```

'--cacert' 옵션 없이 서명 확인 작업이 이루어진다.

Pod에서 REST API 사용하기 – 4/6

API Server 인증

- serviceaccount에 있는 secret 데이터 中 `token` 사용

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

token 활용을 위해 환경 변수로 설정

```
root@curl:/# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1
{
  "kind": "APIResourceList",
  "groupVersion": "v1",
  "resources": [
    {
      "name": "bindings",
      "singularName": "",
      "namespaced": true,
      "kind": "Binding",
      "verbs": [
        "create"
      ]
    },
. . .
```

인증 정보를 포함해서 접근을 하니까 Pass !!!

Pod에서 REST API 사용하기 – 5/6

API Server
인증

- 권한이 없을 수도 있다 !!!

```
root@curl:/# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/namespaces
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

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

namespaces 정보를 보려고 하니, 거부 당한다

- 공부를 위해 serviceaccount에 cluster-admin 권한을 줘버리자 !!!

```
> kubectl create clusterrolebinding permissive-binding --clusterrole=cluster-admin --group=system:serviceaccounts
clusterrolebinding.rbac.authorization.k8s.io/permissive-binding created
```

역할 기반 액세스 제어(RBAC) 비활성화

Pod에서 REST API 사용하기 – 6/6

API Server 인증

- Pod가 속한 namespace에 있는 pods 목록을 볼 수 있다

```
root@curl:/# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/namespaces
{
  "kind": "NamespaceList",
  "apiVersion": "v1",
  "metadata": {
. . .
```

이제 권한이 있어서 잘 보인다 !!!

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

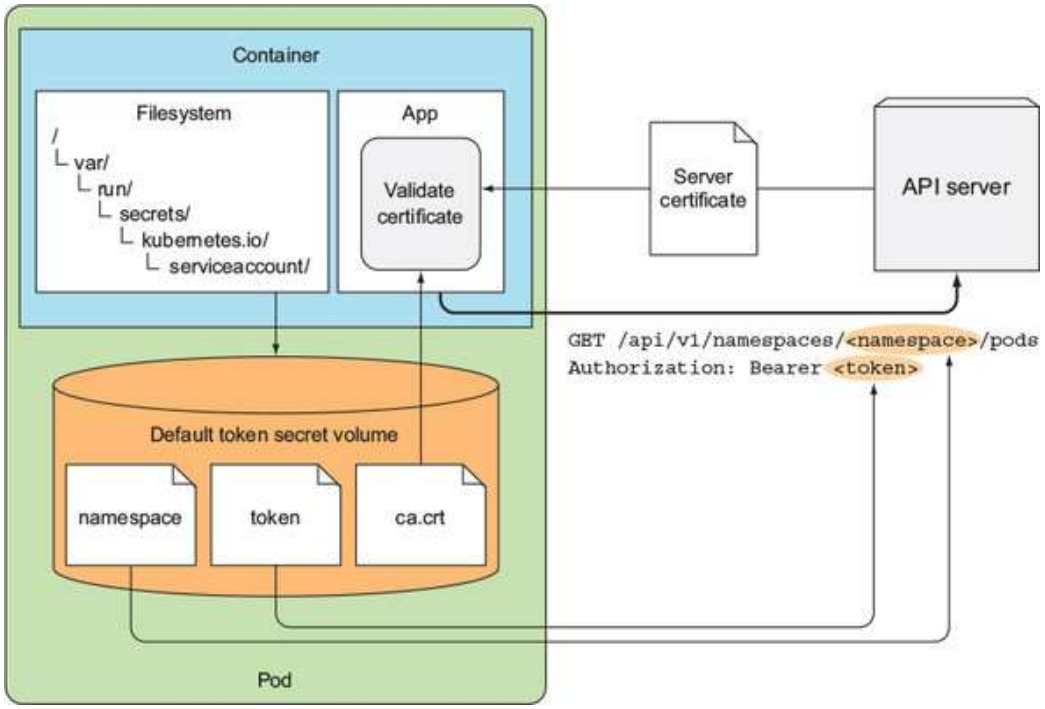
secret에 있는 namespace 정보를 환경 변수로 선언하자 !!!

```
root@curl:/# curl -H "Authorization: Bearer $TOKEN" https://kubernetes/api/v1/namespaces/$NS/pods
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "660748"
  },
  "items": [
    {
      "metadata": {
        "name": "curl",
        "namespace": "default",
        "uid": "efd9f6c3-2aba-45aa-992b-f77172bf5048",
. . .
```

namespace에 속한 pods 목록을 잘 보여준다

Summary ... REST API

- Pod와 API Server 間 통신의 세 가지 측면



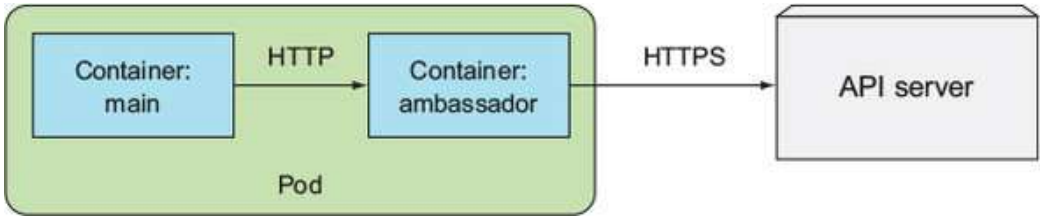
※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-8/104

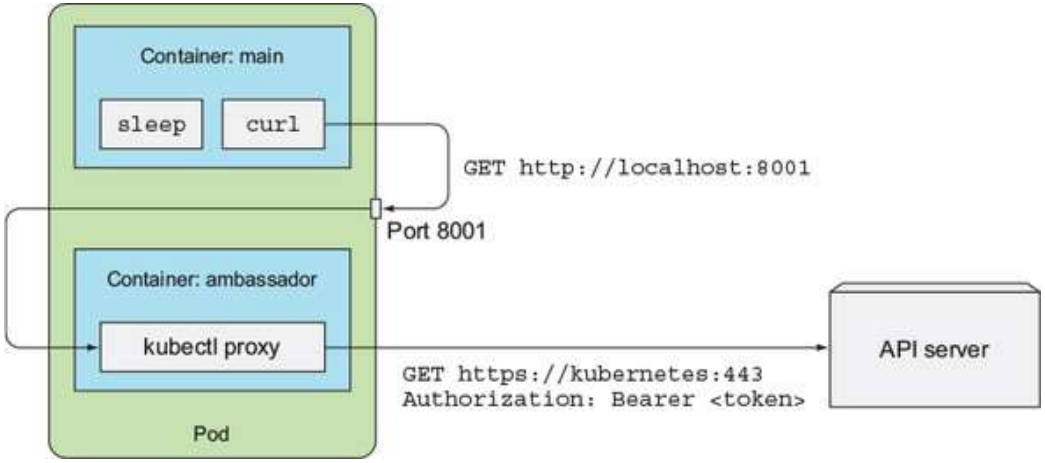
Ambassador Pattern – 1/2

- API Server와 직접 통신하는 Container를 별도로 구성

. App은 HTTP로 ambassador에 연결하고 ambassador proxy가 API Server에 대한 HTTPS 연결을 처리하도록 구성

→ 보안도 투명하게 관리되는 장점





※ 참고 : https://livebook.manning.com/book/kubernetes-in-action/chapter-8/183

Ambassador Pattern – 2/2

- ambassador 역할을 하는 container를 추가해서 그것을 통해 API Server를 이용

07-ambassator-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: ambassador

spec:
  containers:
  - name: curl
    image: curlimages/curl
    command: ["/bin/sleep", "3650d"]

  - name: ambassador
    image: luksa/kubectl-proxy:latest
```

kubectl proxy 기능을 하는 container를 추가

```
❯ kubectl apply -f 07-ambassator-pod.yaml
pod/ambassador created
```

container까지 지정해서 실행 !!!

```
❯ kubectl exec -it ambassador -c curl -- /bin/bash

root@ambassador:/# curl http://localhost:8001
{
  "paths": [
    "/.well-known/openid-configuration",
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
    "/apis/admissionregistration.k8s.io/v1",
    "/apis/admissionregistration.k8s.io/v1beta1",
    "/apis/apiextensions.k8s.io",
    "/apis/apiextensions.k8s.io/v1",
    "/apis/apiextensions.k8s.io/v1beta1",
    "/apis/apiregistration.k8s.io",
. . .
```

https://kahoot.it/