클라우드 시스템 요구사항 분석

- 컨테이너 오케스트레이션 쿠버네티스 -

조 이름: STEAM

조원: 김민지

설예림

진승우

현룡관

1. 2개의 작업 클러스터 생성, 백업, 업그레이드

1번 문제

[ETCD 백업]

https://127.0.0.1:2379에서 실행 중인 etcd의 snapshot을 생성하고 snapshot을 /data/etcd-snapshot.db에 저장합니다.

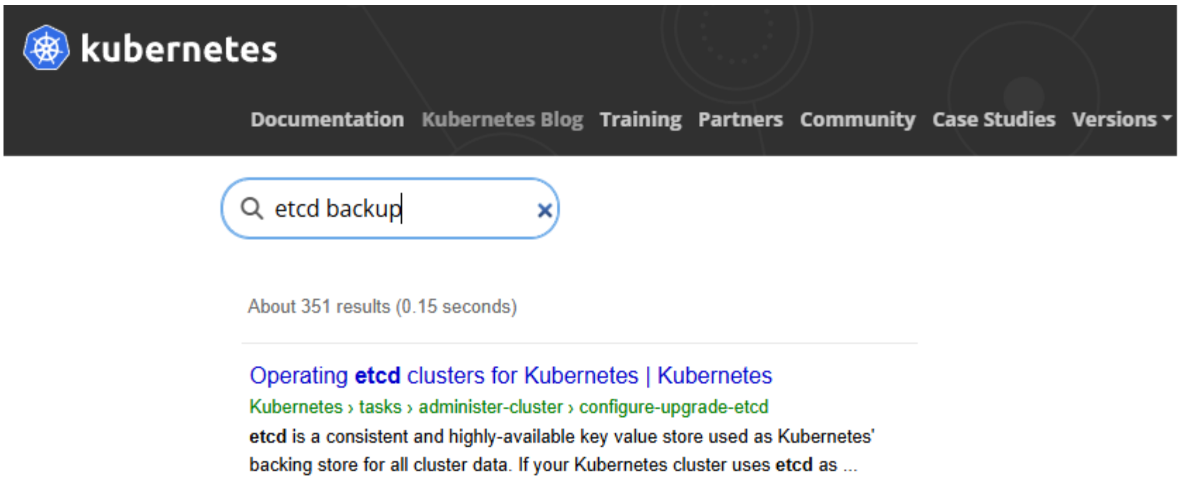
그런 다음 다시 스냅샷을 복원합니다.

etcdctl을 사용하여 서버에 연결하기 위해 다음 TLS 인증서/키가 제공됩니다.

CA certificate: /etc/kubernetes/pki/etcd/ca.crt

Client certificate: /etc/kubernetes/pki/etcd/server.crt

Client key: /etc/kubernetes/pki/etcd/server.key

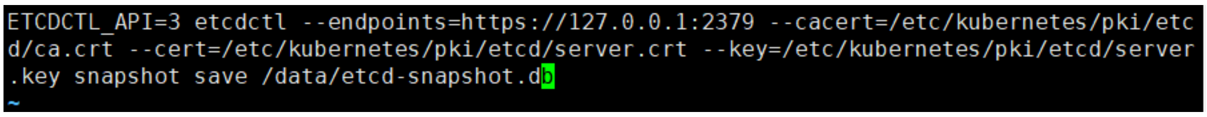


About 351 results (0.15 seconds)

Operating **etcd** clusters for Kubernetes | Kubernetes
Kubernetes › tasks › administer-cluster › configure-upgrade-etcd
etcd is a consistent and highly-available key value store used as Kubernetes' backing store for all cluster data. If your Kubernetes cluster uses **etcd** as ...

검색 : etcd backup

```
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 \
  --cacert=<trusted-ca-file> --cert=<cert-file> --key=<key-file> \
  snapshot save <backup-file-location>
```


```
ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key snapshot save /data/etcd-snapshot.db
~
```

snapshot 명령어 수정

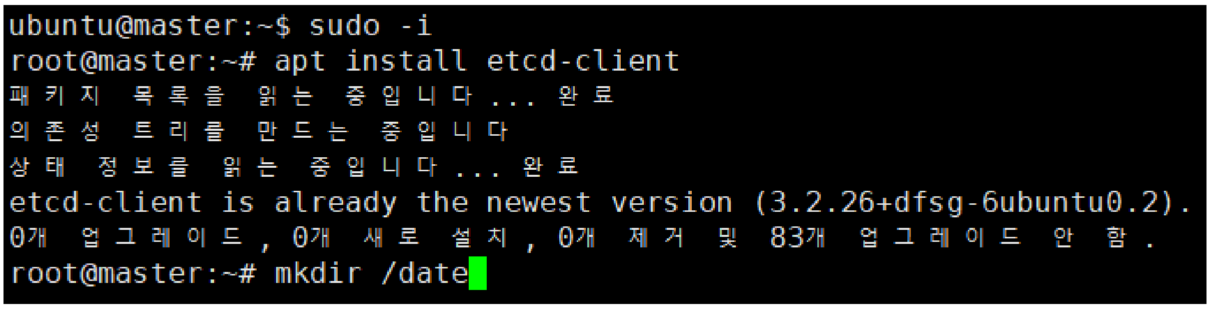
CA certificate: /etc/kubernetes/pki/etcd/ca.crt

Client certificate: /etc/kubernetes/pki/etcd/server.crt

Client key: /etc/kubernetes/pki/etcd/server.key

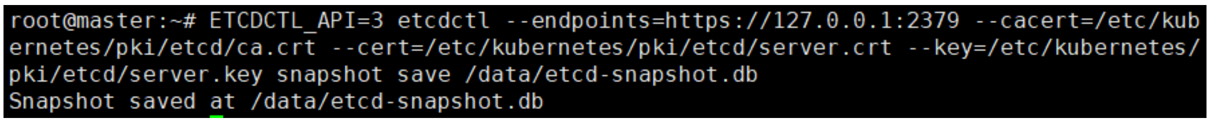
실행 중인 etcd의 snapshot을 생성하고 snapshot을 /data/etcd-snapshot.db에 저장

스냅샷을 복원

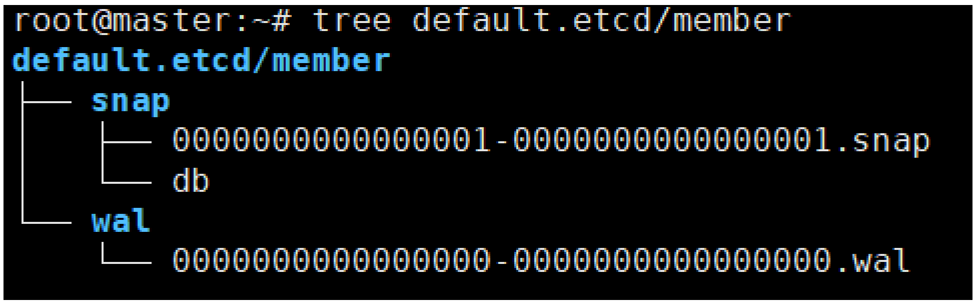

```
ubuntu@master:~$ sudo -i
root@master:~# apt install etcd-client
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
etcd-client is already the newest version (3.2.26+dfsg-6ubuntu0.2).
0개 업그레이드, 0개 새로 설치, 0개 제거 및 83개 업그레이드 안 함.
root@master:~# mkdir /date
```

apt install etcd-client

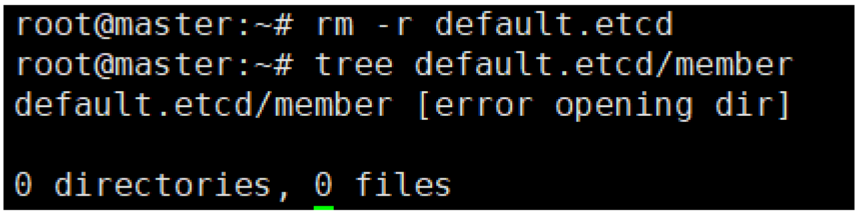
etcd client 설치


```
root@master:~# ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key snapshot save /data/etcd-snapshot.db
Snapshot saved at /data/etcd-snapshot.db
```

etcdctl을 사용하여 서버에 연결하기 위해 다음 TLS 인증서/키가 제공

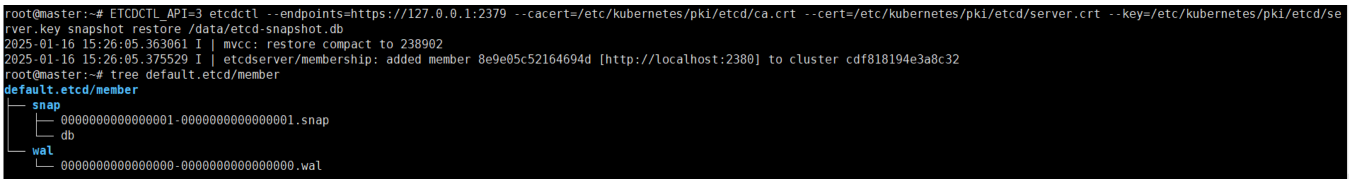

```
root@master:~# tree default.etcd/member
default.etcd/member
├── snap
│   ├── 0000000000000001-0000000000000001.snap
│   └── db
└── wal
    └── 0000000000000000-0000000000000000.wal
```

Etcd 스냅샷 저장 확인



```
root@master:~# rm -r default.etcd
root@master:~# tree default.etcd/member
default.etcd/member [error opening dir]

0 directories, 0 files
```

스냅샷 삭제



```
root@master:~# ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/se
rver.key snapshot restore /data/etcd-snapshot.db
2025-01-16 15:26:05.363061 I | mvcc: restore compact to 238902
2025-01-16 15:26:05.375529 I | etcdserver/membership: added member 8e9e05c52164694d [http://localhost:2380] to cluster cdf818194e3a8c32
root@master:~# tree default.etcd/member
default.etcd/member
├── snap
│   ├── 0000000000000001-0000000000000001.snap
│   └── db
└── wal
    └── 0000000000000000-0000000000000000.wal
```

스냅샷 복원 및 확인

사용한 명령어

ETCDCTL_API=3 etcdctl –endpoints=https://127.0.0.1:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubenetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key snapshot restore /data/etcd-snapshot.db

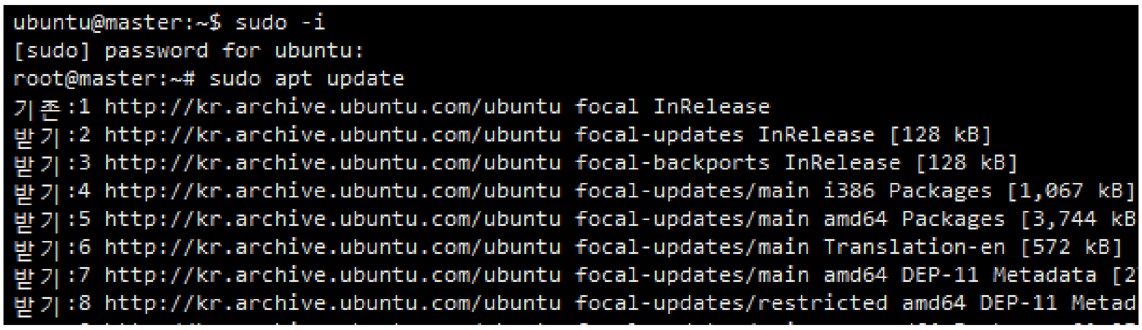
2번 문제

[Cluster Upgrade]

마스터 노드의 모든 Kubernetes control plane및 node 구성 요소를 버전 1.29.6-1.1 버전으로 업그레이드합니다.

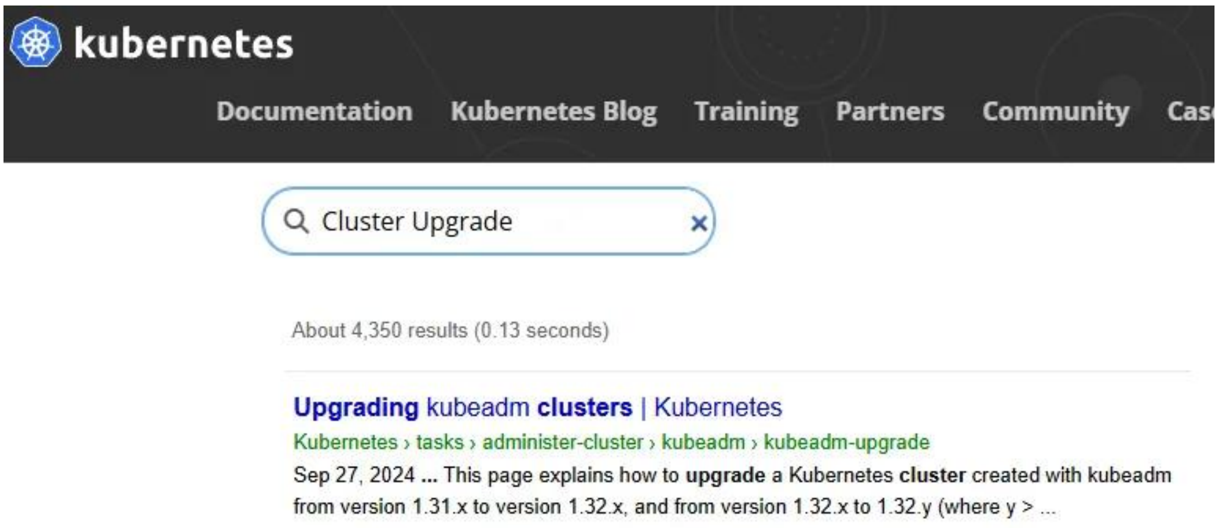
master 노드를 업그레이드하기 전에 drain 하고 업그레이드 후에 uncordon해야 합니다.

"주의사항" 반드시 Master Node에서 root권한을 가지고 작업을 실행해야 한다.


```
ubuntu@master:~$ sudo -i
[sudo] password for ubuntu:
root@master:~# sudo apt update
기존:1 http://kr.archive.ubuntu.com/ubuntu focal InRelease
받기:2 http://kr.archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
받기:3 http://kr.archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
받기:4 http://kr.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [1,067 kB]
받기:5 http://kr.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [3,744 kB
받기:6 http://kr.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [572 kB]
받기:7 http://kr.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [2
받기:8 http://kr.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 DEP-11 Metad
```
sudo -i

sudo apt update

구성 요소를 업데이트



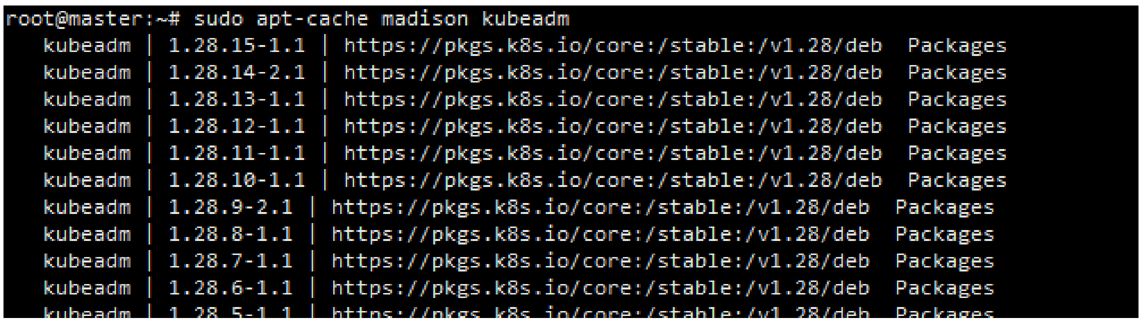
About 4,350 results (0.13 seconds)

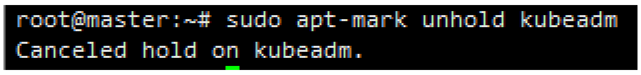
Upgrading kubeadm **clusters** | Kubernetes

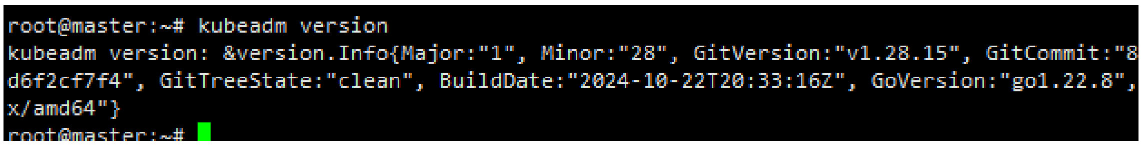
Kubernetes › tasks › administer-cluster › kubeadm › kubeadm-upgrade

Sep 27, 2024 ... This page explains how to **upgrade** a Kubernetes **cluster** created with kubeadm from version 1.31.x to version 1.32.x, and from version 1.32.x to 1.32.y (where y > ...

검색어: Cluster Upgrade

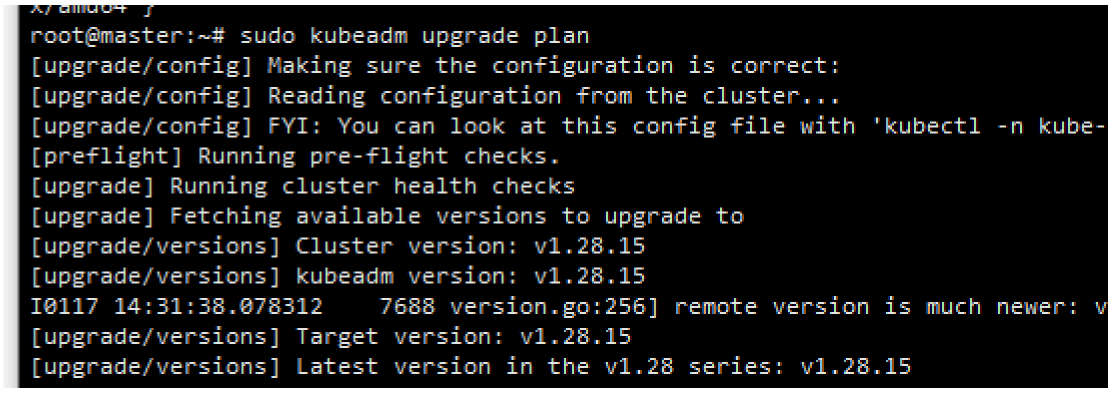

```
root@master:~# sudo apt-cache madison kubeadm
   kubeadm | 1.28.15-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.14-2.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.13-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.12-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.11-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.10-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.9-2.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.8-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.7-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.6-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.5-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
```
sudo apt-cache madison kubeadm


```
root@master:~# sudo apt-mark unhold kubeadm
Canceled hold on kubeadm.
```
sudo apt-mark unhold kubeadm



```
root@master:~# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"28", GitVersion:"v1.28.15", GitCommit:"8
d6f2cf7f4", GitTreeState:"clean", BuildDate:"2024-10-22T20:33:16Z", GoVersion:"go1.22.8",
x/amd64"}
root@master:~#
```

kubeadm의 버전을 보여줌



```
root@master:~# sudo kubeadm upgrade plan
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.28.15
[upgrade/versions] kubeadm version: v1.28.15
I0117 14:31:38.078312    7688 version.go:256] remote version is much newer: v
[upgrade/versions] Target version: v1.28.15
[upgrade/versions] Latest version in the v1.28 series: v1.28.15
```

sudo kubectl upgrade plan

업그레이드 후(Upgrade worker nodes 전까지만 해도 괜찮음)

Sudo kubeadm upgrade node, sudo kubeadm upgrade apply

2. 역할 기반 제어

3번 문제

[Service Account, Role, RoleBinding 생성]

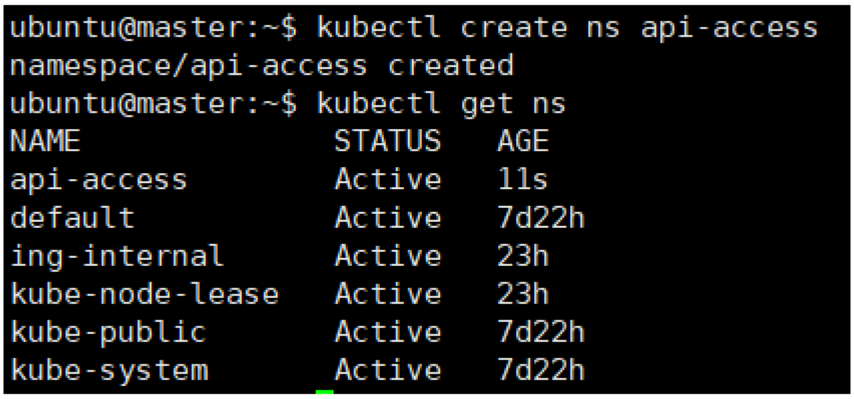
애플리케이션 운영중 특정 namespace의 Pod들을 모니터할수 있는 서비스가 요청되었습니다.

api-access 네임스페이스의 모든 pod를 view할 수 있도록 다음의 작업을 진행하시오.

1. api-access라는 새로운 namespace에 pod-viewer라는 이름의 Service Account를 만듭니다.

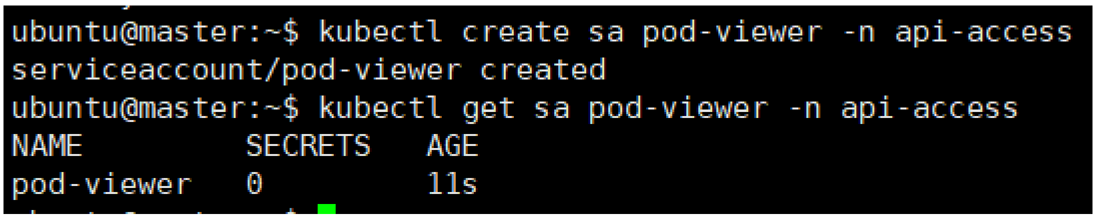
2. podreader-role이라는 이름의 Role과 podreader-rolebinding이라는 이름의 RoleBinding을 만듭니다.

3. 앞서 생성한 ServiceAccount를 API resource Pod에 대하여 watch, list, get을 허용하도록 매핑하시오.



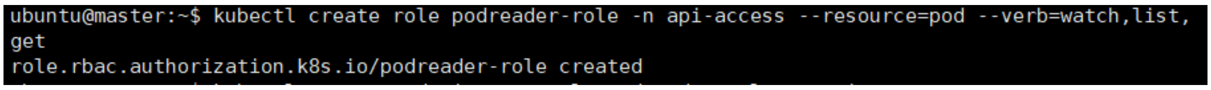
```
ubuntu@master:~$ kubectl create ns api-access
namespace/api-access created
ubuntu@master:~$ kubectl get ns
NAME              STATUS   AGE
api-access        Active   11s
default           Active   7d22h
ing-internal      Active   23h
kube-node-lease   Active   23h
kube-public       Active   7d22h
kube-system       Active   7d22h
```

namespace api-access 생성 및 확인



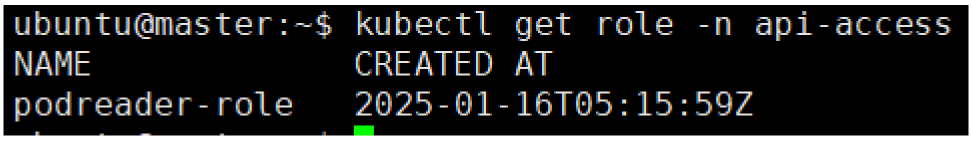
```
ubuntu@master:~$ kubectl create sa pod-viewer -n api-access
serviceaccount/pod-viewer created
ubuntu@master:~$ kubectl get sa pod-viewer -n api-access
NAME         SECRETS   AGE
pod-viewer   0         11s
```

api-access라는 새로운 namespace에 pod-viewer라는 Service Account 만듬

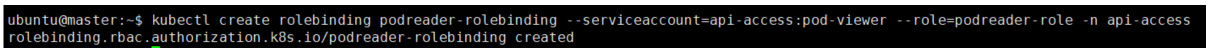


```
ubuntu@master:~$ kubectl create role podreader-role -n api-access --resource=pod --verb=watch,list,
get
role.rbac.authorization.k8s.io/podreader-role created
```

podreader-role 이름의 Role 생성

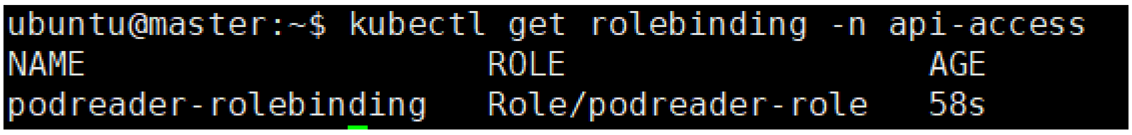


```
ubuntu@master:~$ kubectl get role -n api-access
NAME             CREATED AT
podreader-role   2025-01-16T05:15:59Z
```



```
ubuntu@master:~$ kubectl create rolebinding podreader-rolebinding --serviceaccount=api-access:pod-viewer --role=podreader-role -n api-access
rolebinding.rbac.authorization.k8s.io/podreader-rolebinding created
```

podreader-rolebinding이라는 이름의 RoleBinding 생성



```
ubuntu@master:~$ kubectl get rolebinding -n api-access
NAME                    ROLE                  AGE
podreader-rolebinding   Role/podreader-role   58s
```

Kubectl get rolebinding -n api-access

Rolebinding 명령어로 api-access를 확인

4번 문제

[Service Account, ClusterRole, ClusterRoleBinding 생성]

애플리케이션 배포를 위해 새로운 ClusterRole을 생성하고 특정 namespace의

ServiceAccount를 바인드하시오.

다음의 resource type에서만 Create가 허용된 ClusterRole deployment-clusterrole을

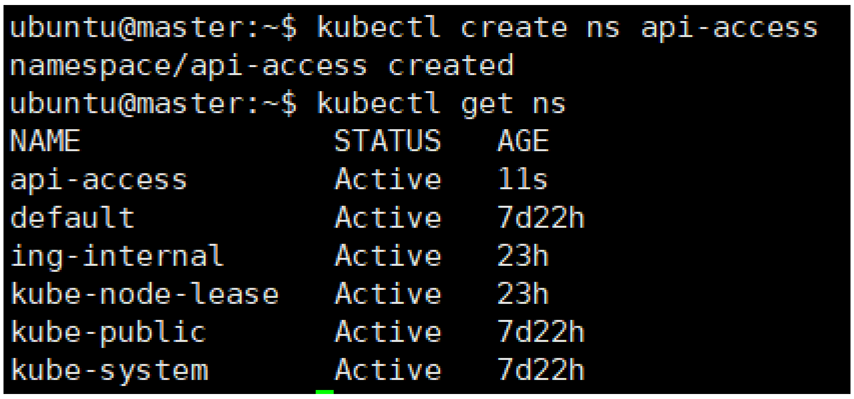
생성합니다.

Resource Type: Deployment StatefulSet DaemonSet

미리 생성된 namespace api-access 에 cicd-token이라는 새로운 ServiceAccount를 만듭니다.

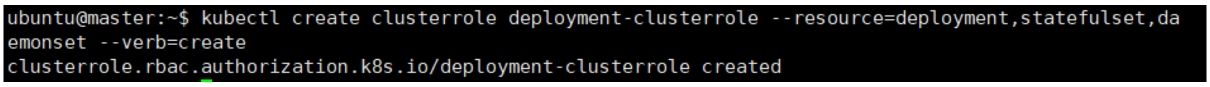
ClusterRole deployment-clusterrole을 namespace api-access 로 제한된 새 ServiceAccount

cicd-token에 바인딩하세요.



```
ubuntu@master:~$ kubectl create ns api-access
namespace/api-access created
ubuntu@master:~$ kubectl get ns
NAME                STATUS   AGE
api-access          Active   11s
default             Active   7d22h
ing-internal        Active   23h
kube-node-lease     Active   23h
kube-public         Active   7d22h
kube-system         Active   7d22h
```

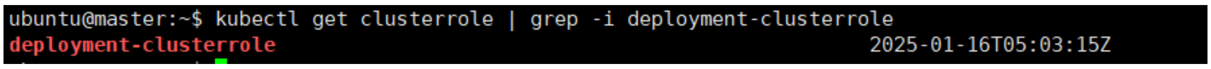
namespace 생성



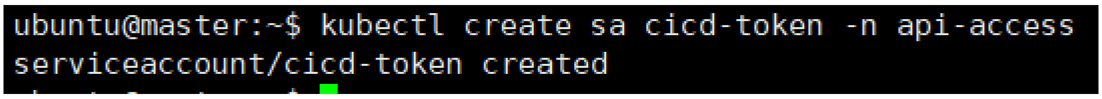
```
ubuntu@master:~$ kubectl create clusterrole deployment-clusterrole --resource=deployment,statefulset,da
emonset --verb=create
clusterrole.rbac.authorization.k8s.io/deployment-clusterrole created
```

resource type에서만 Create가 허용된 ClusterRole deployment-clusterrole 생성

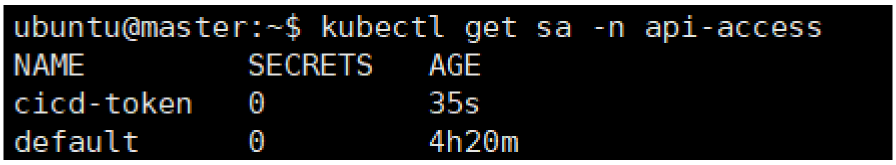
resource type : Deployment StatefulSet DaemonSet

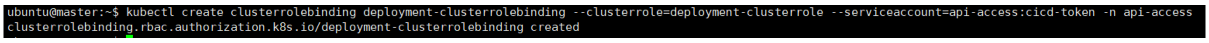


```
ubuntu@master:~$ kubectl get clusterrole | grep -i deployment-clusterrole
deployment-clusterrole                                                 2025-01-16T05:03:15Z
```

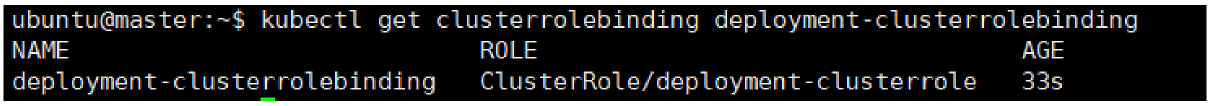

```
ubuntu@master:~$ kubectl create sa cicd-token -n api-access
serviceaccount/cicd-token created
```

namespace api-access에 cicd-token이라는 새로운 ServiceAccount 생성


```
ubuntu@master:~$ kubectl get sa -n api-access
NAME         SECRETS   AGE
cicd-token   0         35s
default      0         4h20m
```


```
ubuntu@master:~$ kubectl create clusterrolebinding deployment-clusterrolebinding --clusterrole=deployment-clusterrole --serviceaccount=api-access:cicd-token -n api-access
clusterrolebinding.rbac.authorization.k8s.io/deployment-clusterrolebinding created
```

ClusterRole deployment-clusterrole을 namespace api-access로 제한된 새 ServiceAccount cicd-token에 바인딩


```
ubuntu@master:~$ kubectl get clusterrolebinding deployment-clusterrolebinding
NAME                            ROLE                                 AGE
deployment-clusterrolebinding   ClusterRole/deployment-clusterrole   33s
```

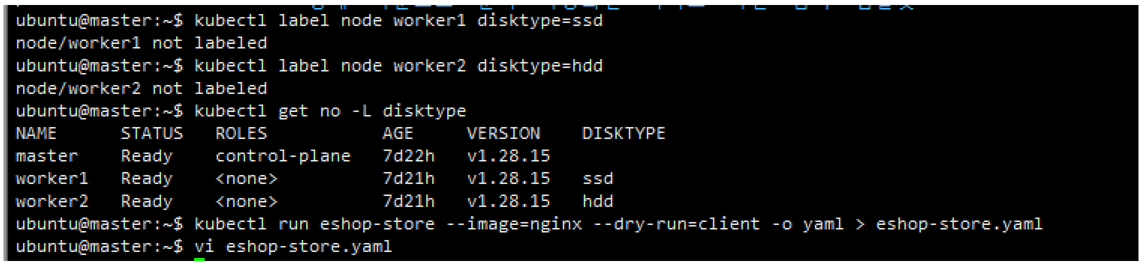
Kubectl get clusterrolebinding deployment-clusterrolebinding 명령어로 rolebinding 확인

3. 노드 관리

5번 문제

[노드 비우기]

k8s-worker2 노드를 스케줄링 불가능하게 설정하고, 해당 노드에서 실행 중인 모든 Pod을 다른 node로 reschedule 하세요.


```
ubuntu@master:~$ kubectl label node worker1 disktype=ssd
node/worker1 not labeled
ubuntu@master:~$ kubectl label node worker2 disktype=hdd
node/worker2 not labeled
ubuntu@master:~$ kubectl get no -L disktype
NAME      STATUS   ROLES           AGE    VERSION    DISKTYPE
master    Ready    control-plane   7d22h  v1.28.15
worker1   Ready    <none>          7d21h  v1.28.15   ssd
worker2   Ready    <none>          7d21h  v1.28.15   hdd
ubuntu@master:~$ kubectl run eshop-store --image=nginx --dry-run=client -o yaml > eshop-store.yaml
ubuntu@master:~$ vi eshop-store.yaml
```

1. 검색어: nodeselector (첫번째 -> 맨마지막 Learn how to use nodeSelector.)

2. 노드에 라벨추가

kubectl label node worker1 disktype=ssd

kubectl label node worker2 disktype=hdd

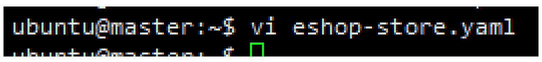
kubectl get no -L disktype

6번 문제

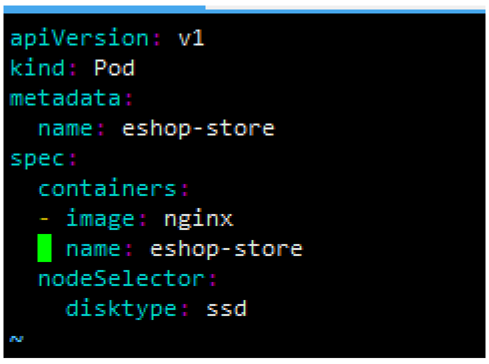
[Pod Scheduling]

3. 파드 생성 야물

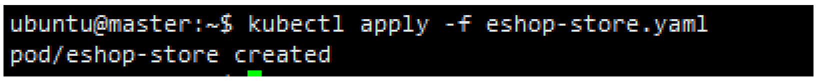
kubectl run eshop-store --image=nginx --dry-run=client -o yaml > eshop-store.yaml



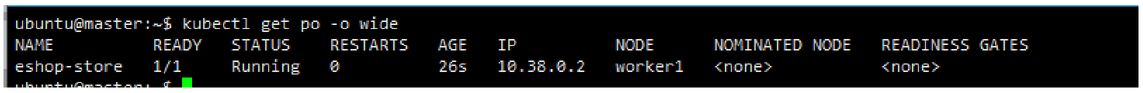
```
ubuntu@master:~$ vi eshop-store.yaml
```



```
apiVersion: v1
kind: Pod
metadata:
  name: eshop-store
spec:
  containers:
  - image: nginx
    name: eshop-store
  nodeSelector:
    disktype: ssd
~
```



```
ubuntu@master:~$ kubectl apply -f eshop-store.yaml
pod/eshop-store created
```



```
ubuntu@master:~$ kubectl get po -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP          NODE      NOMINATED NODE   READINESS GATES
eshop-store   1/1     Running   0          26s   10.38.0.2   worker1   <none>           <none>
```

4. 야물 파일에 nodeSelector 추가

5. kubectl apply -f eshop-store.yaml

확인 kubectl get po -o wide

4. 파드 생성

7번 문제

[환경변수, command, args 적용]

'cka-exam'이라는 namespace를만들고, 'cka-exam' namespace에 아래와 같은 Pod를

생성하시오.

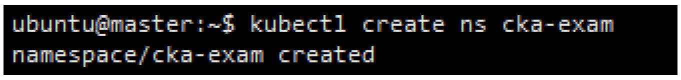
pod Name: pod-01

image: busybox

환경변수 : CERT = "CKA-cert"

command: /bin/sh

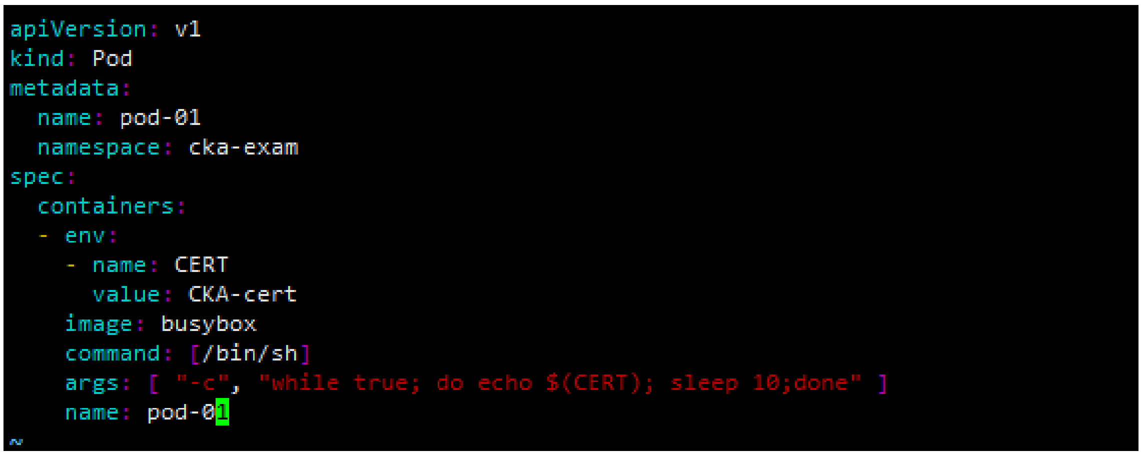
args: "-c", "while true; do echo $(CERT); sleep 10;done



```
ubuntu@master:~$ kubectl create ns cka-exam
namespace/cka-exam created
```

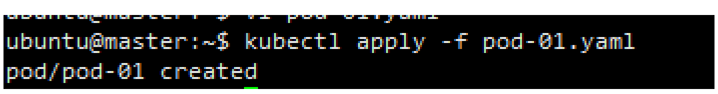
1. 검색어: command

2. kubectl create ns cka-exam



```
apiVersion: v1
kind: Pod
metadata:
  name: pod-01
  namespace: cka-exam
spec:
  containers:
  - env:
    - name: CERT
      value: CKA-cert
    image: busybox
    command: [/bin/sh]
    args: [ "-c", "while true; do echo $(CERT); sleep 10;done" ]
    name: pod-01
~
```

vi 파일 수정



```
ubuntu@master:~$ kubectl apply -f pod-01.yaml
pod/pod-01 created
```

3. kubectl run pod-01 -n cka-exam --image=busybox --env=CERT="CKA-cert" --dry-run=client -o

yaml > pod-01.yaml

추가: command, arg

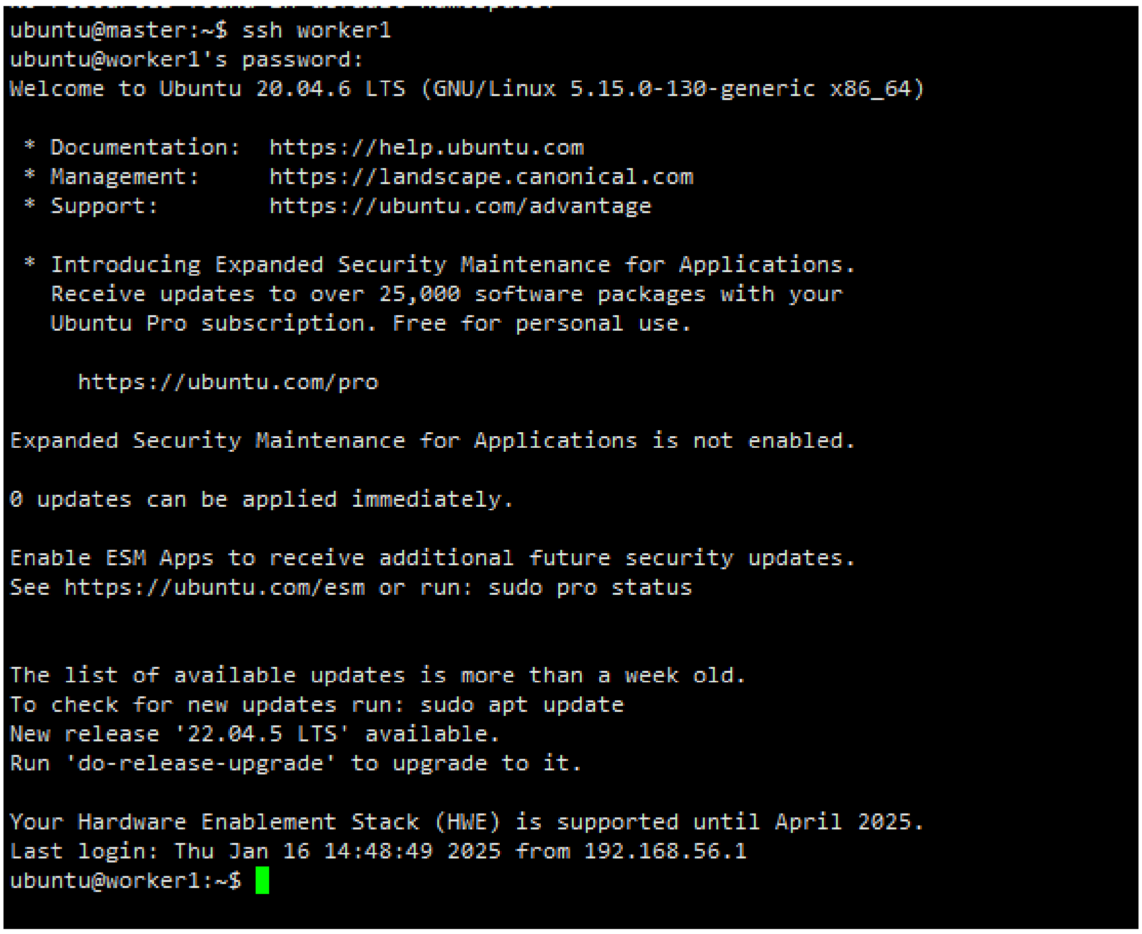
8번 문제

[로그 확인]

Pod "nginx-static-pod-k8s-worker1"의 log를 모니터링하고, 메세지를 포함하는 로그라인을

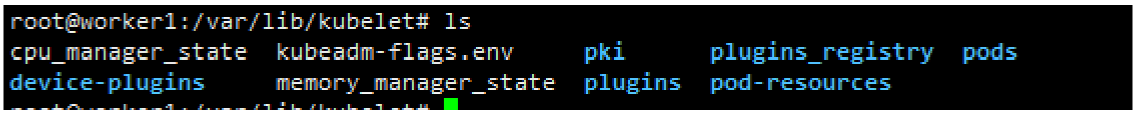
추출하세요.

추출된 결과는 /opt/REPORT/2023/pod-log에 기록하세요.


```
ubuntu@master:~$ ssh worker1
ubuntu@worker1's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-130-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Thu Jan 16 14:48:49 2025 from 192.168.56.1
ubuntu@worker1:~$
```

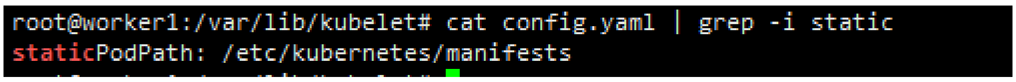
1. ssh worker1 접속

2. worker1에서 sudo -i


```
root@worker1:/var/lib/kubelet# ls
cpu_manager_state  kubeadm-flags.env    pki      plugins_registry  pods
device-plugins     memory_manager_state plugins  pod-resources
```

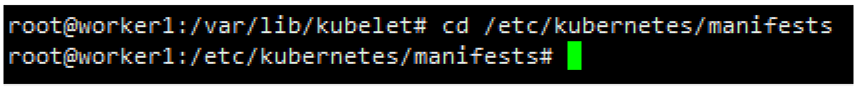
3. cd /var/lib/kubelet

4. ls


```
root@worker1:/var/lib/kubelet# cat config.yaml | grep -i static
staticPodPath: /etc/kubernetes/manifests
```

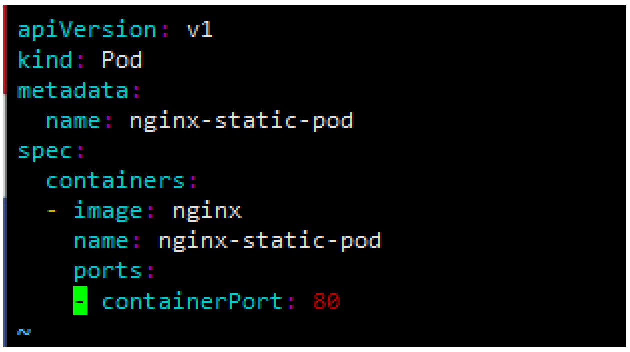
5. cat config.yaml | grep -i static (staticPodPath 경로 확인)

(/etc/kubernetes/manifests/)

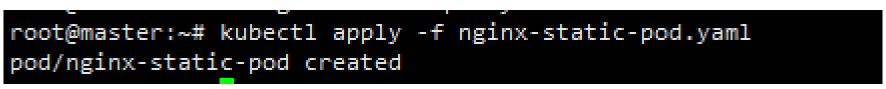


```
root@worker1:/var/lib/kubelet# cd /etc/kubernetes/manifests
root@worker1:/etc/kubernetes/manifests#
```

6. cd /etc/kubernetes/manifests



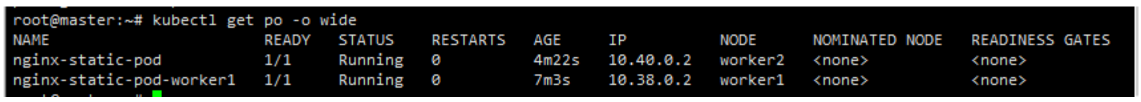
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-static-pod
spec:
  containers:
  - image: nginx
    name: nginx-static-pod
    ports:
    - containerPort: 80
~
```



```
root@master:~# kubectl apply -f nginx-static-pod.yaml
pod/nginx-static-pod created
```

7. 또다른 창-master

kubectl run nginx-static-pod --image=nginx --port=80 --dry-run=client -o yaml > nginx-static-pod.yaml(수정) -> 내용 복사



```
root@master:~# kubectl get po -o wide
NAME                       READY   STATUS    RESTARTS   AGE     IP           NODE      NOMINATED NODE   READINESS GATES
nginx-static-pod           1/1     Running   0          4m22s   10.40.0.2    worker2   <none>           <none>
nginx-static-pod-worker1   1/1     Running   0          7m3s    10.38.0.2    worker1   <none>           <none>
```

8. 다른 창: worker

   vi nginx-static-pod.yaml

확인 master : kubectl get po -o wide

9번 문제

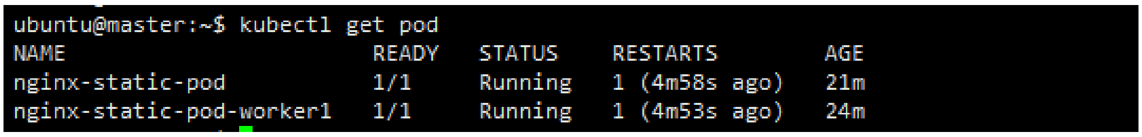
[Static Pod 생성]

worker1 노드에 nginx-static-pod.yaml 라는 이름의 Static Pod를 생성하세요.

pod name: nginx-static-pod

image: nginx

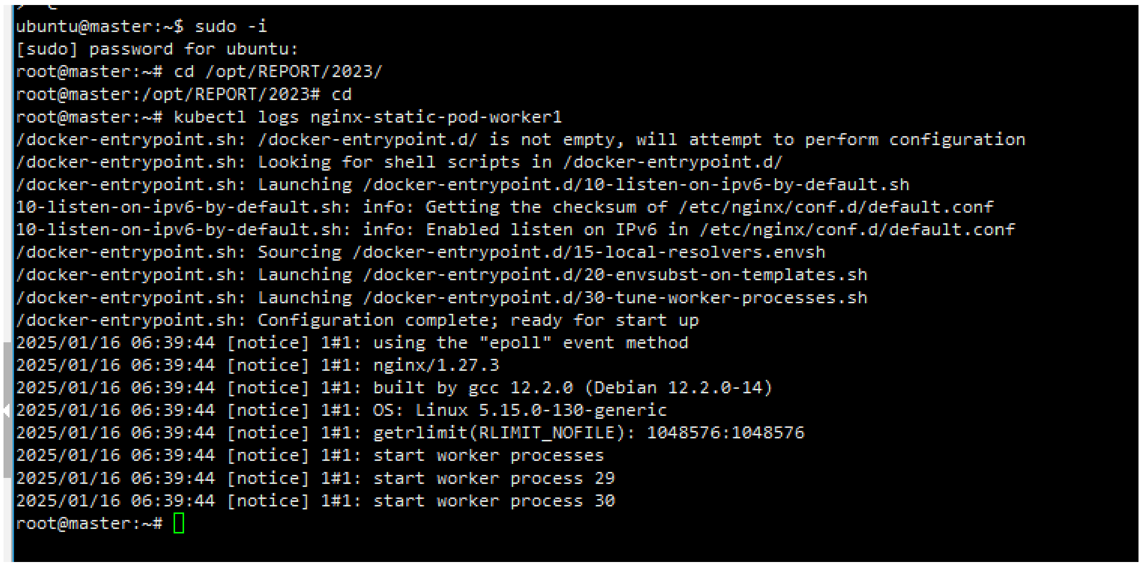
port : 80



```
ubuntu@master:~$ kubectl get pod
NAME                       READY   STATUS    RESTARTS       AGE
nginx-static-pod           1/1     Running   1 (4m58s ago)  21m
nginx-static-pod-worker1   1/1     Running   1 (4m53s ago)  24m
```

1. kubectl get pod

   sudo -i

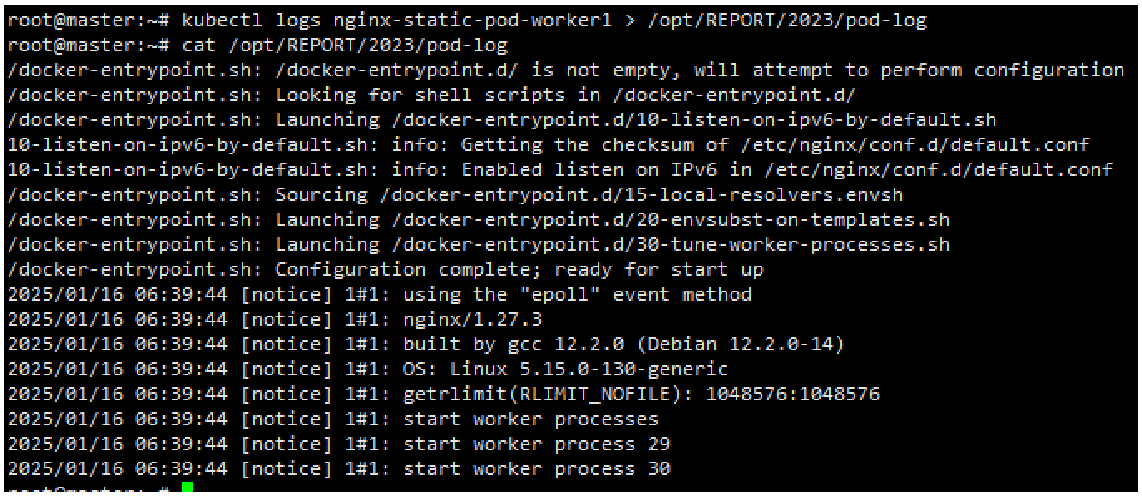


```
ubuntu@master:~$ sudo -i
[sudo] password for ubuntu:
root@master:~# cd /opt/REPORT/2023/
root@master:/opt/REPORT/2023# cd
root@master:~# kubectl logs nginx-static-pod-worker1
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/01/16 06:39:44 [notice] 1#1: using the "epoll" event method
2025/01/16 06:39:44 [notice] 1#1: nginx/1.27.3
2025/01/16 06:39:44 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/16 06:39:44 [notice] 1#1: OS: Linux 5.15.0-130-generic
2025/01/16 06:39:44 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/16 06:39:44 [notice] 1#1: start worker processes
2025/01/16 06:39:44 [notice] 1#1: start worker process 29
2025/01/16 06:39:44 [notice] 1#1: start worker process 30
root@master:~#
```

2. mkdir -p /opt/REPORT/2023/ (시험때는 x)

3. kubectl logs nginx-static-pod-worker1

   kubectl logs 파드이름



```
root@master:~# kubectl logs nginx-static-pod-worker1 > /opt/REPORT/2023/pod-log
root@master:~# cat /opt/REPORT/2023/pod-log
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/01/16 06:39:44 [notice] 1#1: using the "epoll" event method
2025/01/16 06:39:44 [notice] 1#1: nginx/1.27.3
2025/01/16 06:39:44 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/16 06:39:44 [notice] 1#1: OS: Linux 5.15.0-130-generic
2025/01/16 06:39:44 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/16 06:39:44 [notice] 1#1: start worker processes
2025/01/16 06:39:44 [notice] 1#1: start worker process 29
2025/01/16 06:39:44 [notice] 1#1: start worker process 30
```

4. kubectl logs nginx-static-pod-worker1 > /opt/REPORT/2023/pod-log

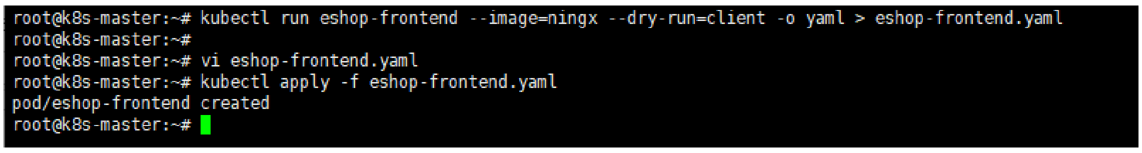
5. cat /opt/REPORT/2023/pod-log

10번 문제

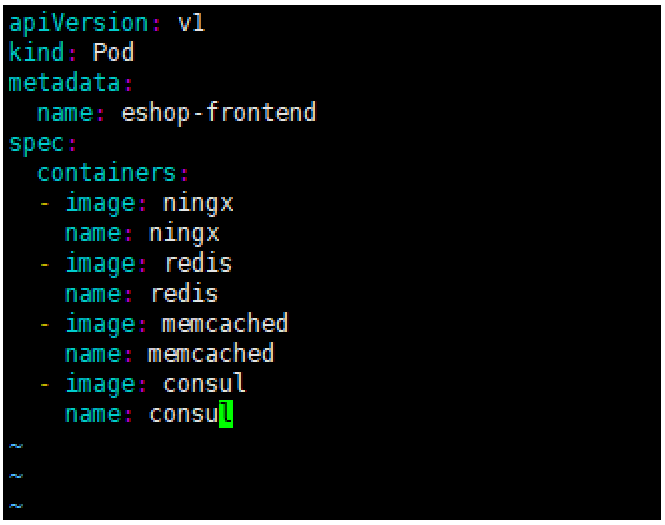
[Multi Container Pod 생성]

4개의 컨테이너를 동작시키는 eshop-frontend Pod를 생성하시오.

  pod image: nginx, redis, memcached, consul



```
root@k8s-master:~# kubectl run eshop-frontend --image=ningx --dry-run=client -o yaml > eshop-frontend.yaml
root@k8s-master:~#
root@k8s-master:~# vi eshop-frontend.yaml
root@k8s-master:~# kubectl apply -f eshop-frontend.yaml
pod/eshop-frontend created
root@k8s-master:~#
```



```
apiVersion: v1
kind: Pod
metadata:
  name: eshop-frontend
spec:
  containers:
  - image: ningx
    name: ningx
  - image: redis
    name: redis
  - image: memcached
    name: memcached
  - image: consul
    name: consul
~
~
~
```

1. kubectl run eshop-frontend --image=nginx --dry-run=client -o yaml > eshop-frontend.yaml
2. vi eshop-frontend.yaml (2yy,p 복사,붙여넣기)
3. kubectl apply -f eshop-frontend.yaml

5. 쿠버네티스 컨트롤러와 네트워킹

11번 문제

[Rolling Update & Roll Back]

Deployment를 이용해 nginx 파드를 3개 배포한 다음 컨테이너 이미지 버전을 rolling

update하고 update record를 기록합니다.

마지막으로 컨테이너 이미지를 previous version으로 roll back 합니다.

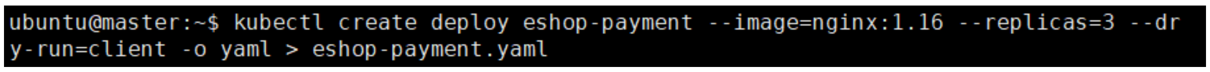
name: eshop-payment

Image : nginx

Image version: 1.16

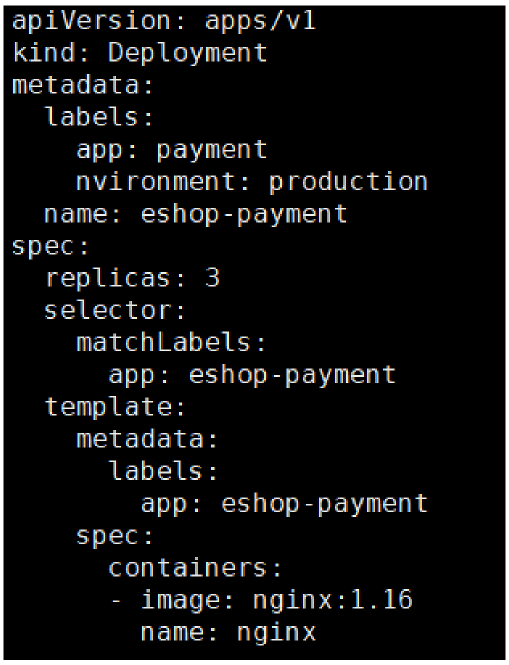
update image version: 1.17

label: app=payment, environment=production



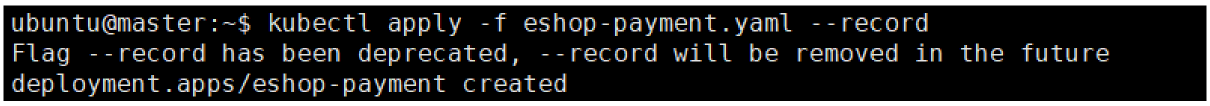
```
ubuntu@master:~$ kubectl create deploy eshop-payment --image=nginx:1.16 --replicas=3 --dr
y-run=client -o yaml > eshop-payment.yaml
```

Rolling Update & Roll Back 야물 생성



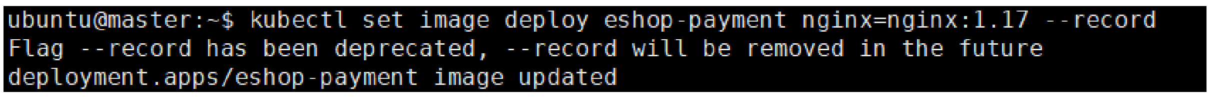
```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: payment
    nvironment: production
  name: eshop-payment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eshop-payment
  template:
    metadata:
      labels:
        app: eshop-payment
    spec:
      containers:
      - image: nginx:1.16
        name: nginx
```

vi eshop-payment.yaml 야물 수정



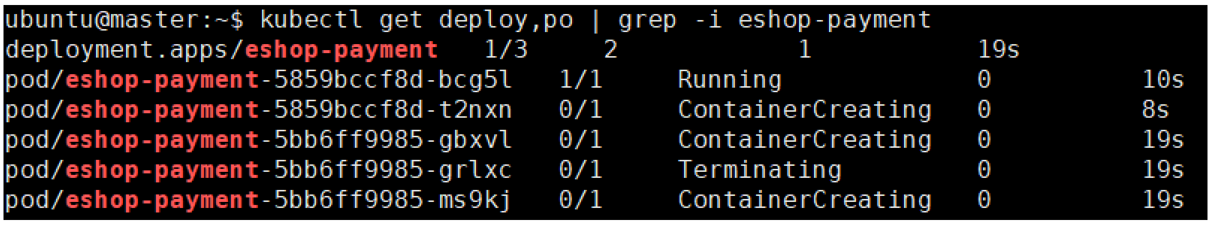
```
ubuntu@master:~$ kubectl apply -f eshop-payment.yaml --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/eshop-payment created
```

적용 및 레코드(기록)



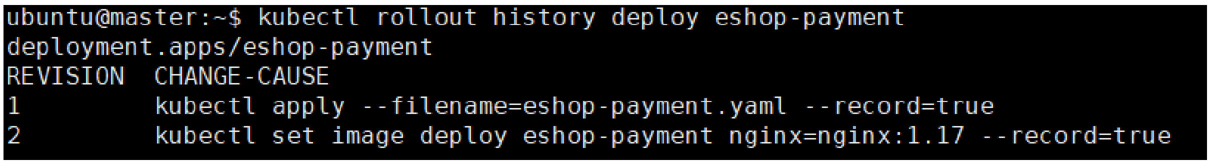
```
ubuntu@master:~$ kubectl set image deploy eshop-payment nginx=nginx:1.17 --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/eshop-payment image updated
```

컨테이너 이미지 버전 rolling update 및 레코드(기록)



```
ubuntu@master:~$ kubectl get deploy,po | grep -i eshop-payment
deployment.apps/eshop-payment   1/3     2            1           19s
pod/eshop-payment-5859bccf8d-bcg5l   1/1     Running             0          10s
pod/eshop-payment-5859bccf8d-t2nxn   0/1     ContainerCreating   0          8s
pod/eshop-payment-5bb6ff9985-gbxvl   0/1     ContainerCreating   0          19s
pod/eshop-payment-5bb6ff9985-grlxc   0/1     Terminating         0          19s
pod/eshop-payment-5bb6ff9985-ms9kj   0/1     ContainerCreating   0          19s
```

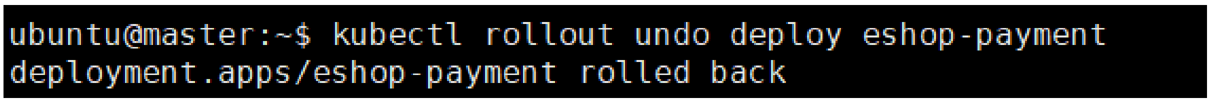
eshop-payment pod 확인



```
ubuntu@master:~$ kubectl rollout history deploy eshop-payment
deployment.apps/eshop-payment
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=eshop-payment.yaml --record=true
2         kubectl set image deploy eshop-payment nginx=nginx:1.17 --record=true
```

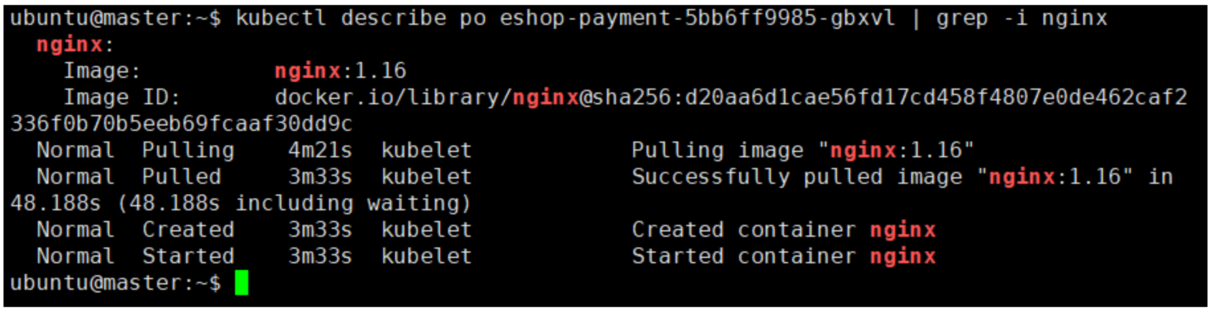
kubectl rollout history deploy eshop-payment

기록 확인



```
ubuntu@master:~$ kubectl rollout undo deploy eshop-payment
deployment.apps/eshop-payment rolled back
```

이전 버전 Roll back



```
ubuntu@master:~$ kubectl describe po eshop-payment-5bb6ff9985-gbxvl | grep -i nginx
  nginx:
    Image:          nginx:1.16
    Image ID:       docker.io/library/nginx@sha256:d20aa6d1cae56fd17cd458f4807e0de462caf2
336f0b70b5eeb69fcaaf30dd9c
  Normal  Pulling    4m21s  kubelet            Pulling image "nginx:1.16"
  Normal  Pulled     3m33s  kubelet            Successfully pulled image "nginx:1.16" in
48.188s (48.188s including waiting)
  Normal  Created    3m33s  kubelet            Created container nginx
  Normal  Started    3m33s  kubelet            Started container nginx
ubuntu@master:~$ 
```

kubectl describe po eshop-payment- ** | grep -I nginx

12번 문제

[ClutserIP]

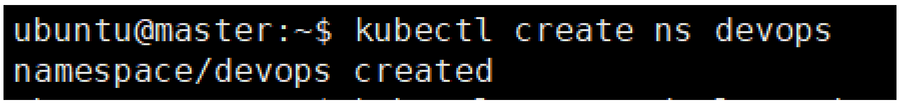
'devops' namespace에서 deployment eshop-order를 다음 조건으로 생성하시오.

- image: nginx, replicas: 2, label: name=order

'eshop-order' deployment의 Service를 만드세요.

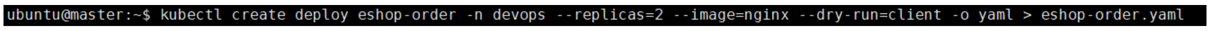
Service Name: eshop-order-svc

Type: ClusterIP, Port: 80



```
ubuntu@master:~$ kubectl create ns devops
namespace/devops created
```

devops namespace 생성

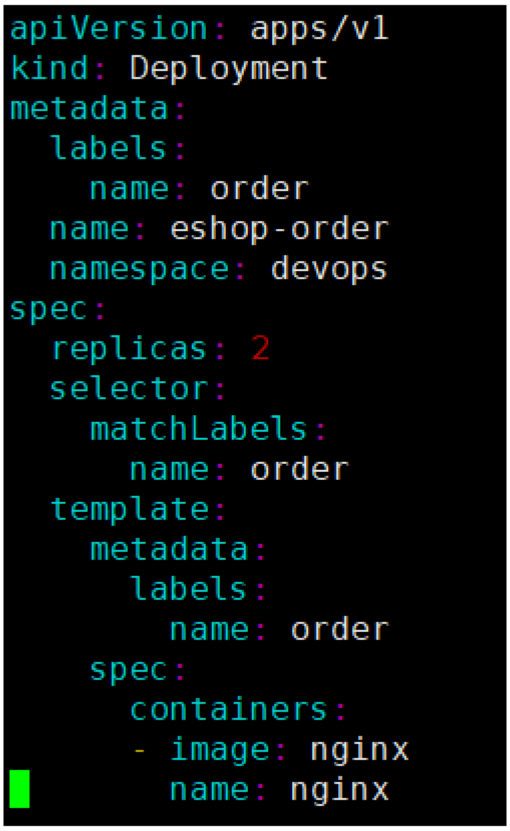


```
ubuntu@master:~$ kubectl create deploy eshop-order -n devops --replicas=2 --image=nginx --dry-run=client -o yaml > eshop-order.yaml
```

kubectl create deploy eshop-order -n devops --replicas=2 --image=nginx --dry-run=client -o yaml

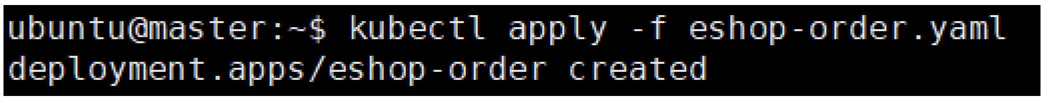
> eshop-order.yaml

야물 파일 생성



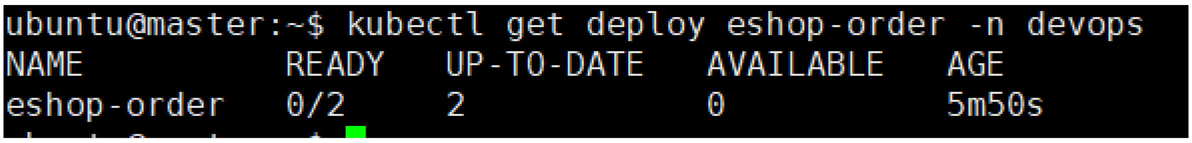
```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: order
  name: eshop-order
  namespace: devops
spec:
  replicas: 2
  selector:
    matchLabels:
      name: order
  template:
    metadata:
      labels:
        name: order
    spec:
      containers:
      - image: nginx
        name: nginx
```

vi eshop-order.yaml 야물 파일 수정

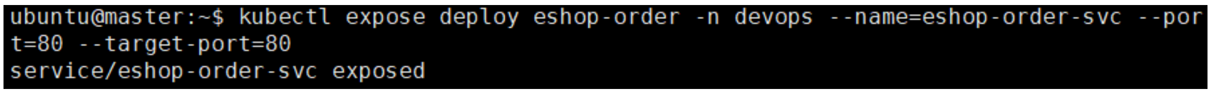


```
ubuntu@master:~$ kubectl apply -f eshop-order.yaml
deployment.apps/eshop-order created
```

야물 파일 적용

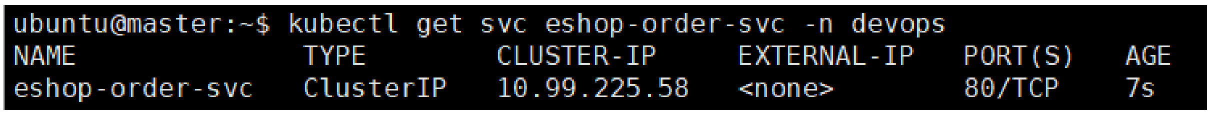


```
ubuntu@master:~$ kubectl get deploy eshop-order -n devops
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
eshop-order   0/2     2            0           5m50s
```



```
ubuntu@master:~$ kubectl expose deploy eshop-order -n devops --name=eshop-order-svc --por
t=80 --target-port=80
service/eshop-order-svc exposed
```

--target-port=80



```
ubuntu@master:~$ kubectl get svc eshop-order-svc -n devops
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
eshop-order-svc   ClusterIP   10.99.225.58    <none>        80/TCP    7s
```

Cluster IP 확인 가능 명령어

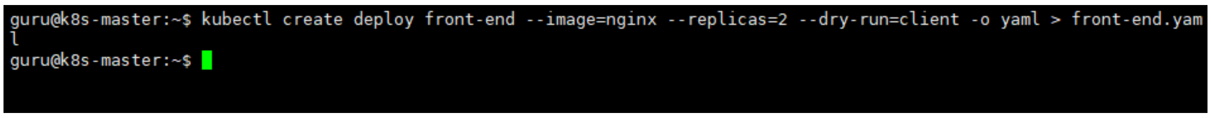
13번 문제

[NodePort]

‘front-end’ deployment를 다음 조건으로 생성하시오.

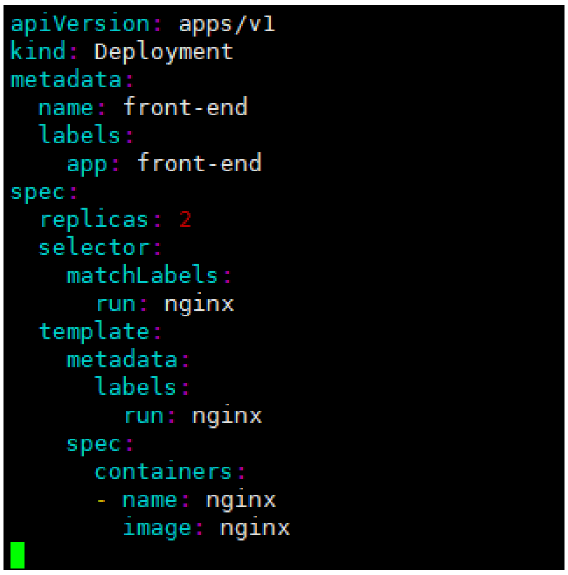
image: nginx, replicas: 2, label: run=nginx

'front-end' deployment의 nginx 컨테이너를 expose하는 'front-end-nodesvc'라는 새 service를 만듭니다. Front-end로 동작중인 Pod에는 node의 **30200** 포트로 접속되어야 합니다.

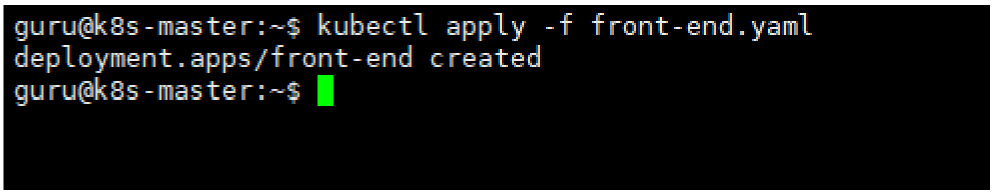


```
guru@k8s-master:~$ kubectl create deploy front-end --image=nginx --replicas=2 --dry-run=client -o yaml > front-end.yam
l
guru@k8s-master:~$ 
```

야물 파일 생성, 수정



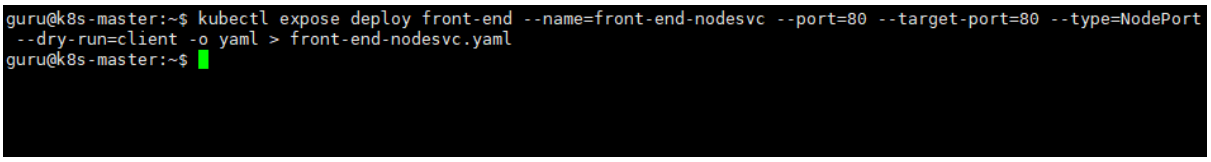
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: front-end
  labels:
    app: front-end
spec:
  replicas: 2
  selector:
    matchLabels:
      run: nginx
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```



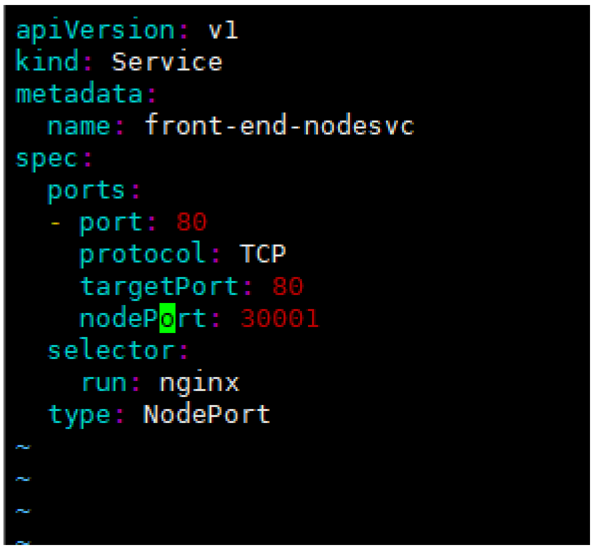
```
guru@k8s-master:~$ kubectl apply -f front-end.yaml
deployment.apps/front-end created
guru@k8s-master:~$ 
```

야물 파일 적용

서비스 생성, 내용 수정

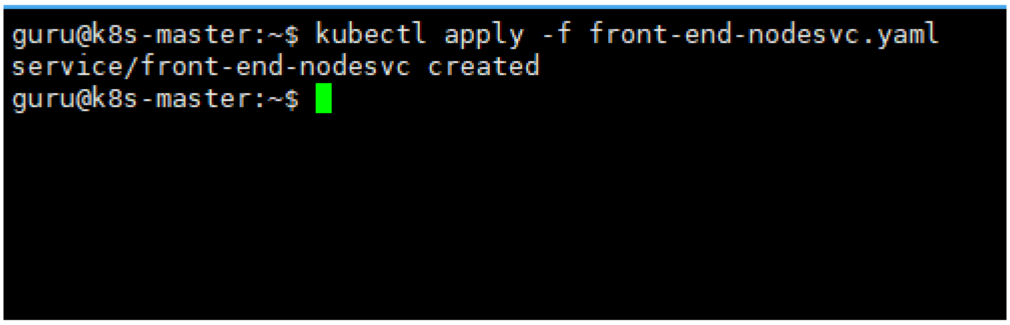


```
guru@k8s-master:~$ kubectl expose deploy front-end --name=front-end-nodesvc --port=80 --target-port=80 --type=NodePort
 --dry-run=client -o yaml > front-end-nodesvc.yaml
guru@k8s-master:~$ 
```

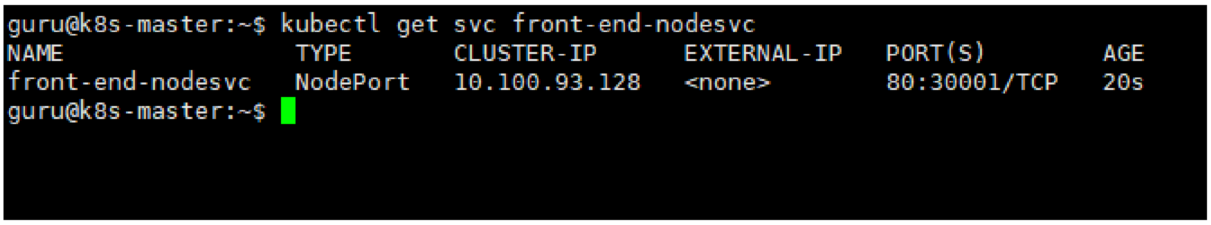


```
apiVersion: v1
kind: Service
metadata:
  name: front-end-nodesvc
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
    nodePort: 30001
  selector:
    run: nginx
  type: NodePort
~
~
~
```

서비스 적용



```
guru@k8s-master:~$ kubectl apply -f front-end-nodesvc.yaml
service/front-end-nodesvc created
guru@k8s-master:~$ 
```


```
guru@k8s-master:~$ kubectl get svc front-end-nodesvc
NAME                TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
front-end-nodesvc   NodePort   10.100.93.128   <none>        80:30001/TCP   20s
guru@k8s-master:~$
```

30001 포트로 접속 가능 확인

6. NetworkPolicy & Ingress

14번 문제

[Network Policy]

customera, customerb를 생성한 후, 각각 PARTITION=customera, PARTITION=customerb를 라벨링하시오.

default namespace에 다음과 같은 pod를 생성하세요.

name: poc

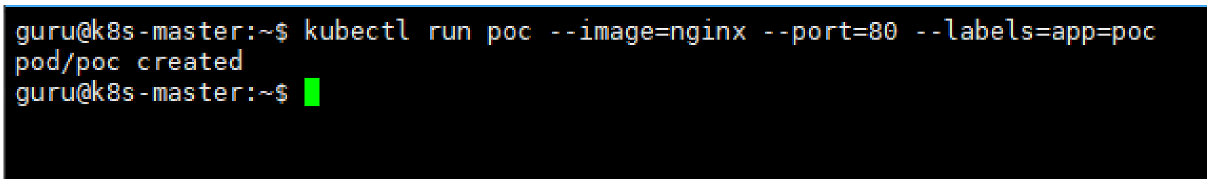
image: nginx

port: 80

label: app=poc

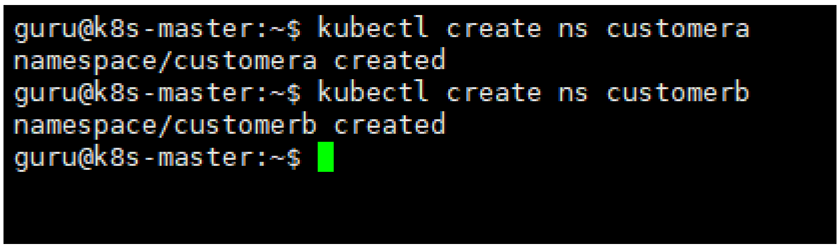
"partition=customera"를 사용하는 namespace에서만 poc의 80포트로 연결할 수 있도록 default namespace에 'allow-web-from-customera'라는 network Policy를 설정하세요.

보안 정책상 다른 namespace의 접근은 제한합니다.


```
guru@k8s-master:~$ kubectl run poc --image=nginx --port=80 --labels=app=poc
pod/poc created
guru@k8s-master:~$
```

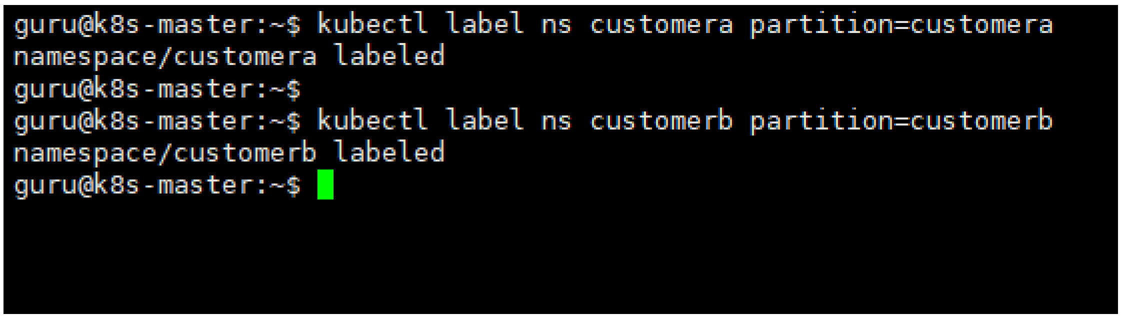
kubectl run poc --image=nginx --port=80 --labels=poc

파드 생성



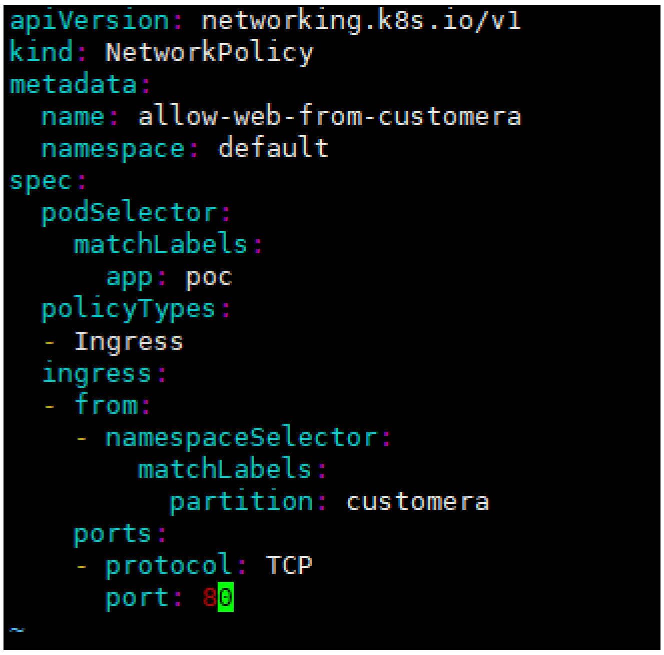
```
guru@k8s-master:~$ kubectl create ns customera
namespace/customera created
guru@k8s-master:~$ kubectl create ns customerb
namespace/customerb created
guru@k8s-master:~$
```

namespace customera, customerb 생성



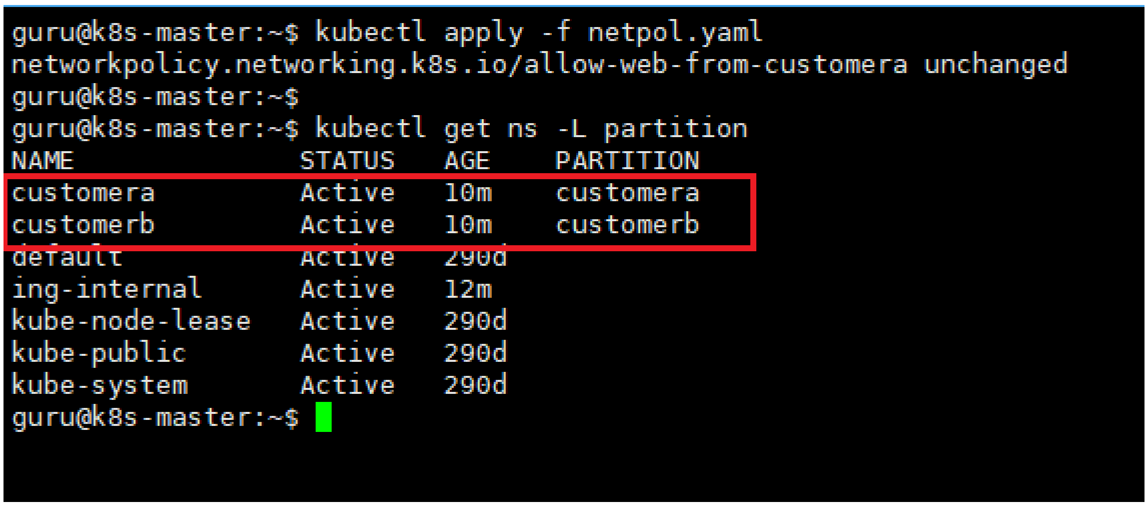
```
guru@k8s-master:~$ kubectl label ns customera partition=customera
namespace/customera labeled
guru@k8s-master:~$
guru@k8s-master:~$ kubectl label ns customerb partition=customerb
namespace/customerb labeled
guru@k8s-master:~$
```

namespace 라벨링 부여



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-web-from-customera
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: poc
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          partition: customera
    ports:
    - protocol: TCP
      port: 80
~
```

vi netpol.yaml 수정



```
guru@k8s-master:~$ kubectl apply -f netpol.yaml
networkpolicy.networking.k8s.io/allow-web-from-customera unchanged
guru@k8s-master:~$
guru@k8s-master:~$ kubectl get ns -L partition
NAME              STATUS   AGE    PARTITION
customera         Active   10m    customera
customerb         Active   10m    customerb
default           Active   290d
ing-internal      Active   12m
kube-node-lease   Active   290d
kube-public       Active   290d
kube-system       Active   290d
guru@k8s-master:~$
```

적용 및 확인

15번 문제

[Ingress]

Create a new nginx Ingress resource as follows:

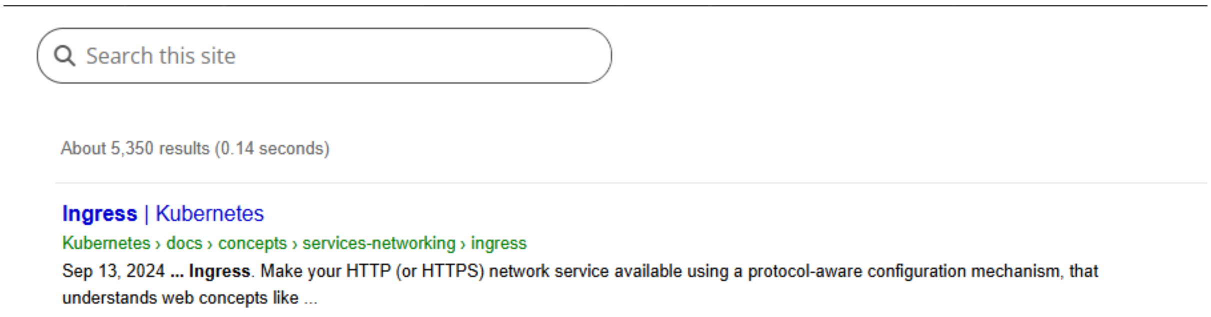
- Name: ping

- Namespace: ing-internal

- Exposing service hi on path /hi using service port 5678

쿠버네티스 사이트에서

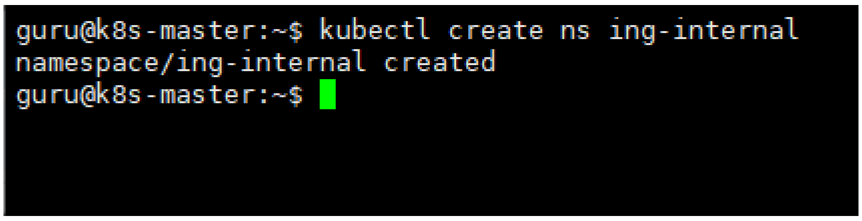
ingress 검색



Search this site

About 5,350 results (0.14 seconds)

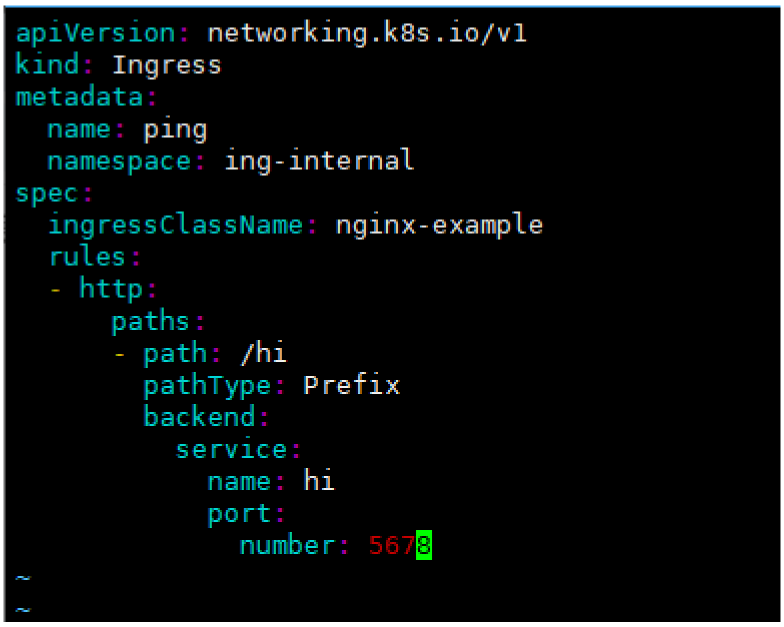
Ingress | Kubernetes
Kubernetes › docs › concepts › services-networking › ingress
Sep 13, 2024 ... **Ingress**. Make your HTTP (or HTTPS) network service available using a protocol-aware configuration mechanism, that understands web concepts like ...



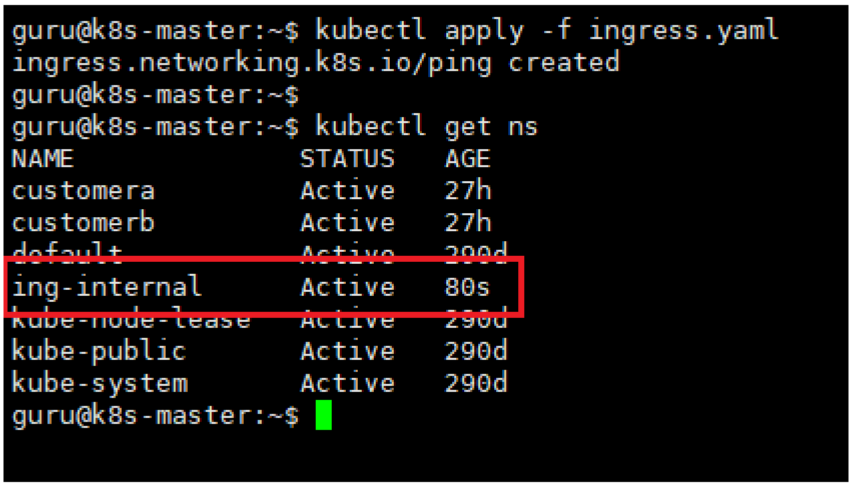
```
guru@k8s-master:~$ kubectl create ns ing-internal
namespace/ing-internal created
guru@k8s-master:~$ 
```

namespace 생성

vi ingress.yaml 수정



```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ping
  namespace: ing-internal
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /hi
        pathType: Prefix
        backend:
          service:
            name: hi
            port:
              number: 5678
~
~
```



```
guru@k8s-master:~$ kubectl apply -f ingress.yaml
ingress.networking.k8s.io/ping created
guru@k8s-master:~$
guru@k8s-master:~$ kubectl get ns
NAME                  STATUS   AGE
customera             Active   27h
customerb             Active   27h
default               Active   290d
ing-internal          Active   80s
kube-node-lease       Active   290d
kube-public           Active   290d
kube-system           Active   290d
guru@k8s-master:~$ 
```

적용 및 확인

확인 명령어 kubectl apply -f ingress.yaml

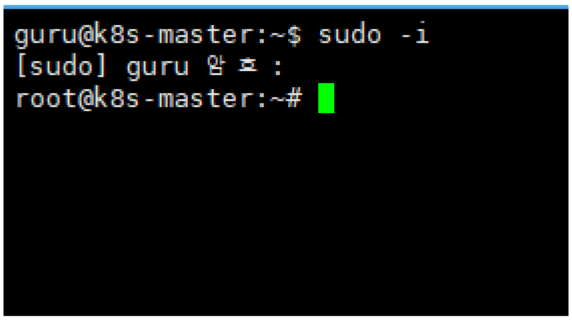
16번 문제

[Service and DNS Lookup]

image nginx를 사용하는 resolver pod를 생성하고 resolver-service라는 service를 구성합니다.

클러스터 내에서 service와 pod 이름을 조회할 수 있는지 테스트합니다.- dns 조회에 사용하는 pod 이미지는 busybox:1.28이고, service와 pod 이름 조회는 nlsookup을 사용합니다.

- service 조회 결과는 /var/CKA2023/nginx.svc에 pod name 조회 결과는 /var/CKA2023/nginx.pod 파일에 기록합니다.

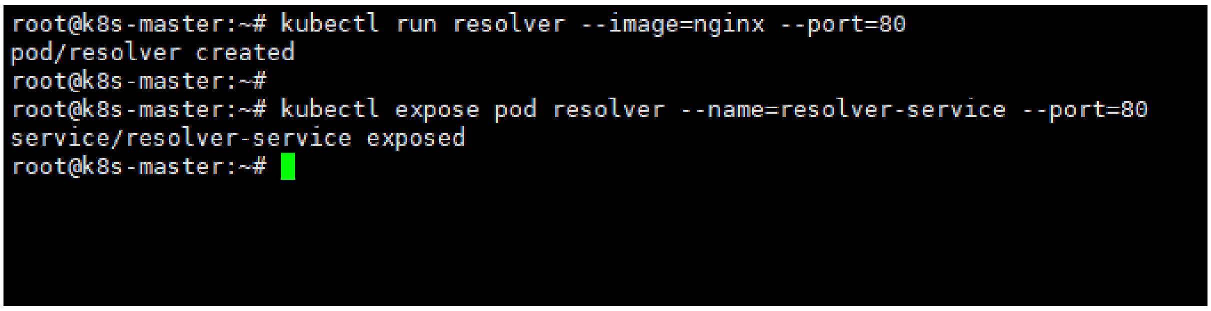


```
guru@k8s-master:~$ sudo -i
[sudo] guru 암호 :
root@k8s-master:~#
```

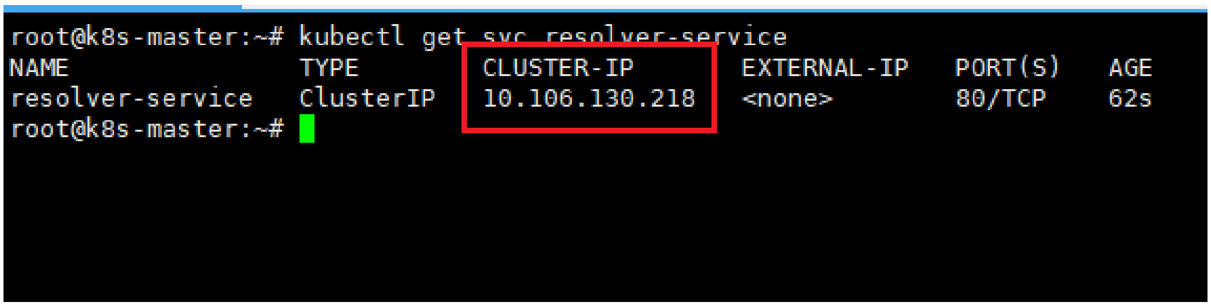
sudo -i

pod 생성, 서비스 구성, 주소 확인

kubectl run resolver --image=nginx --port=80

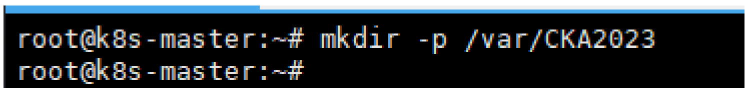


```
root@k8s-master:~# kubectl run resolver --image=nginx --port=80
pod/resolver created
root@k8s-master:~#
root@k8s-master:~# kubectl expose pod resolver --name=resolver-service --port=80
service/resolver-service exposed
root@k8s-master:~#
```



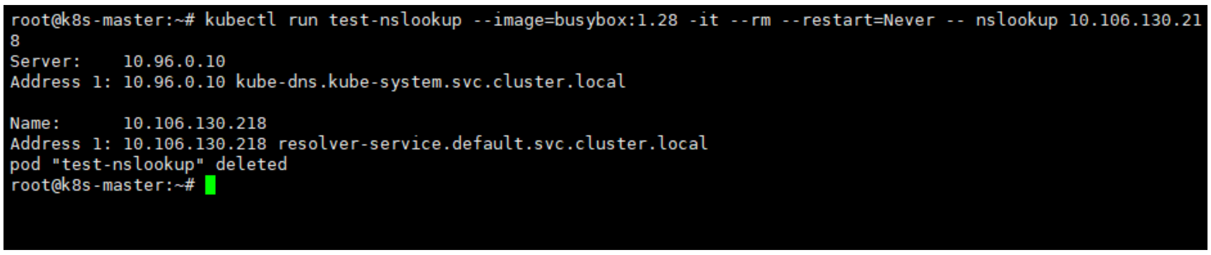
```
root@k8s-master:~# kubectl get svc resolver-service
NAME               TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
resolver-service   ClusterIP   10.106.130.218   <none>        80/TCP    62s
root@k8s-master:~#
```

디렉터리 생성



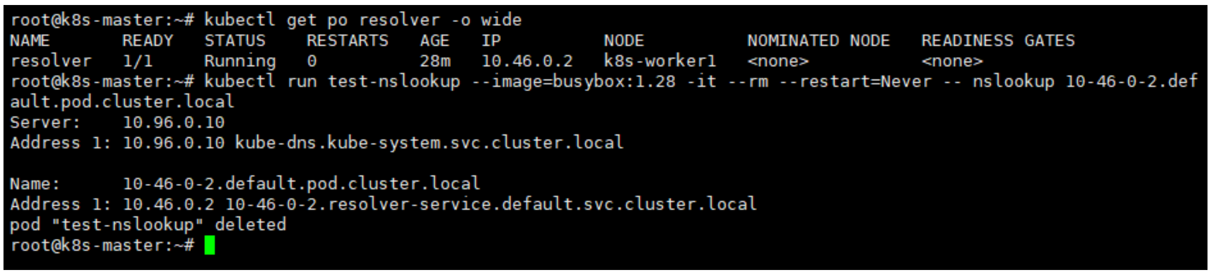
```
root@k8s-master:~# mkdir -p /var/CKA2023
root@k8s-master:~#
```

DNS 조회를 위한 임시 파드

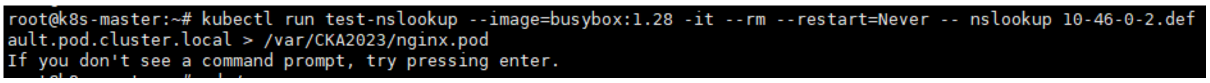


```
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10.106.130.21
8
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.106.130.218
Address 1: 10.106.130.218 resolver-service.default.svc.cluster.local
pod "test-nslookup" deleted
root@k8s-master:~#
```

주소 확인 및 FQDN을 사용하여 특정 파드의 DNS 해석이 올바르게 동작하는지 확인, DNS이름 조회



```
root@k8s-master:~# kubectl get po resolver -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP           NODE          NOMINATED NODE   READINESS GATES
resolver   1/1     Running   0          28m   10.46.0.2    k8s-worker1   <none>           <none>
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10-46-0-2.def
ault.pod.cluster.local
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10-46-0-2.default.pod.cluster.local
Address 1: 10.46.0.2 10-46-0-2.resolver-service.default.svc.cluster.local
pod "test-nslookup" deleted
root@k8s-master:~#
```



```
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10-46-0-2.def
ault.pod.cluster.local > /var/CKA2023/nginx.pod
If you don't see a command prompt, try pressing enter.
```

DNS 해석 결과 확인 후 결과 저장

kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup **-**-**-**.default.pod.cluster.local > /var/CKA2023/nginx.pod

7. Resource 관리

17번 문제

[emptyDir Volume]

다음 조건에 맞춰서 nginx 웹서버 pod가 생성한 로그파일을 받아서 STDOUT으로 출력하는

busybox 컨테이너를 운영하시오.

Pod Name: **weblog**

Web container:

- Image: **nginx:1.17**

- Volume mount : **/var/log/nginx**

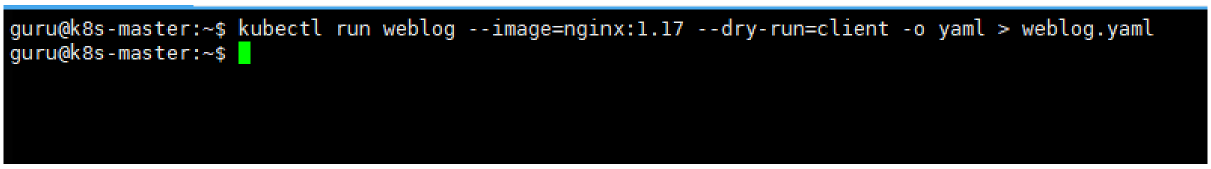
- Readwrite

 Log container:- Image: busybox- args: /bin/sh, -c, "tail -n+1 -f /data/access.log"- Volume mount : /data- readonly

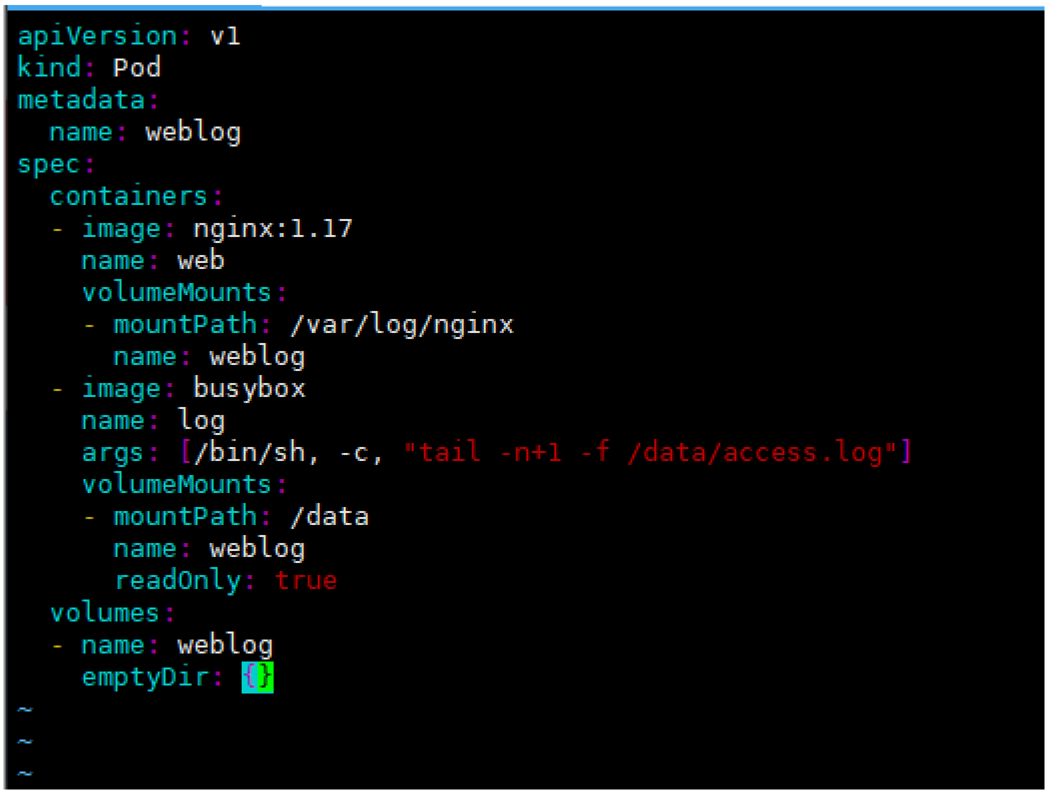
 emptyDir 볼륨을 통한 데이터 공유

조건에 맞는 야물파일 생성

kubectl run weblog --image=nginx:1.17 --dry-run=client -o yaml > weblog.yaml

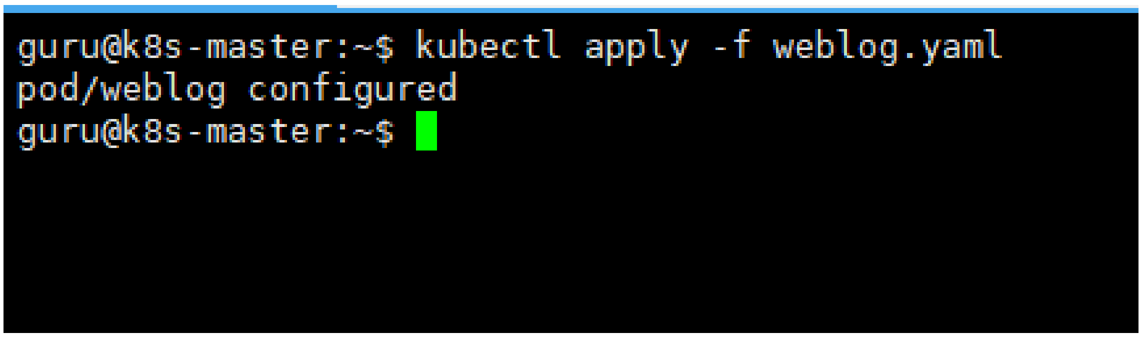


```
guru@k8s-master:~$ kubectl run weblog --image=nginx:1.17 --dry-run=client -o yaml > weblog.yaml
guru@k8s-master:~$ 
```

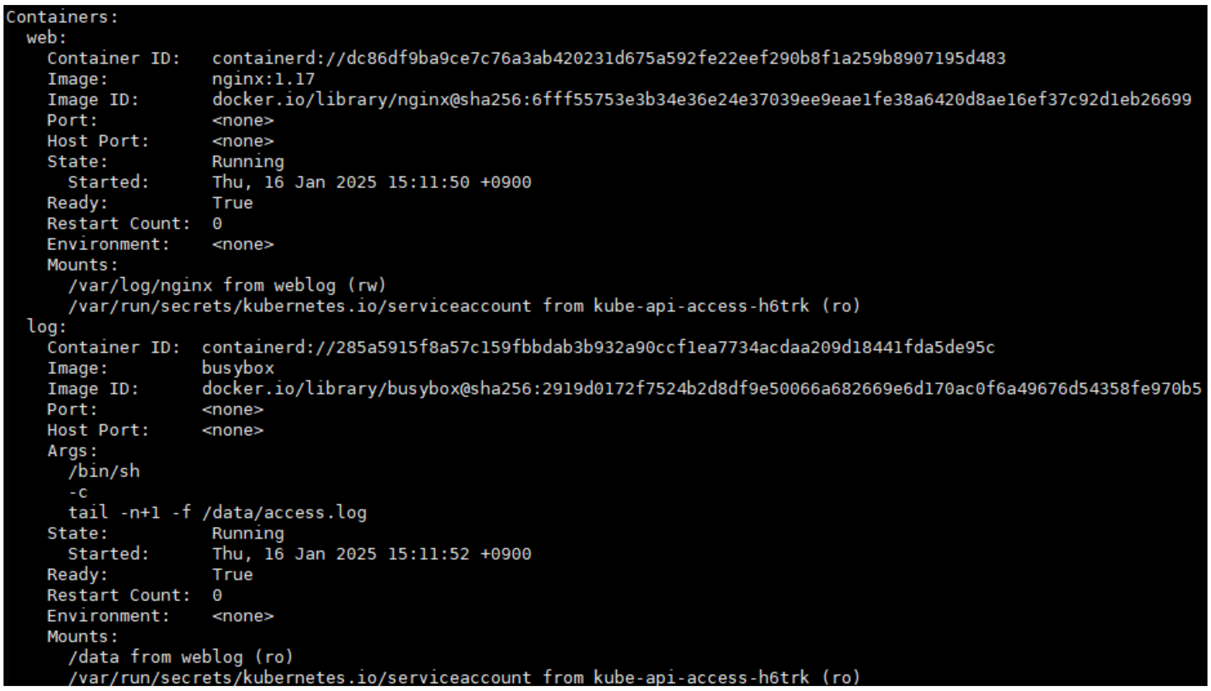


```
apiVersion: v1
kind: Pod
metadata:
  name: weblog
spec:
  containers:
  - image: nginx:1.17
    name: web
    volumeMounts:
    - mountPath: /var/log/nginx
      name: weblog
  - image: busybox
    name: log
    args: [/bin/sh, -c, "tail -n+1 -f /data/access.log"]
    volumeMounts:
    - mountPath: /data
      name: weblog
      readOnly: true
  volumes:
  - name: weblog
    emptyDir: {}
~
~
~
```

vi 수정 후 적용



```
guru@k8s-master:~$ kubectl apply -f weblog.yaml
pod/weblog configured
guru@k8s-master:~$ 
```



```
Containers:
  web:
    Container ID:   containerd://dc86df9ba9ce7c76a3ab420231d675a592fe22eef290b8f1a259b8907195d483
    Image:          nginx:1.17
    Image ID:       docker.io/library/nginx@sha256:6fff55753e3b34e36e24e37039ee9eae1fe38a6420d8ae16ef37c92d1eb26699
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 16 Jan 2025 15:11:50 +0900
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/log/nginx from weblog (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-h6trk (ro)
  log:
    Container ID:  containerd://285a5915f8a57c159fbbdab3b932a90ccf1ea7734acdaa209d18441fda5de95c
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:2919d0172f7524b2d8df9e50066a682669e6d170ac0f6a49676d54358fe970b5
    Port:          <none>
    Host Port:     <none>
    Args:
      /bin/sh
      -c
      tail -n+1 -f /data/access.log
    State:          Running
      Started:      Thu, 16 Jan 2025 15:11:52 +0900
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /data from weblog (ro)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-h6trk (ro)
```

확인 명령어

kubectl describe pod weblog -o wide

18번 문제

[HostPath Volume]

1. /data/cka/fluentd.yaml 파일을 만들어 새로은 Pod 생성하세요

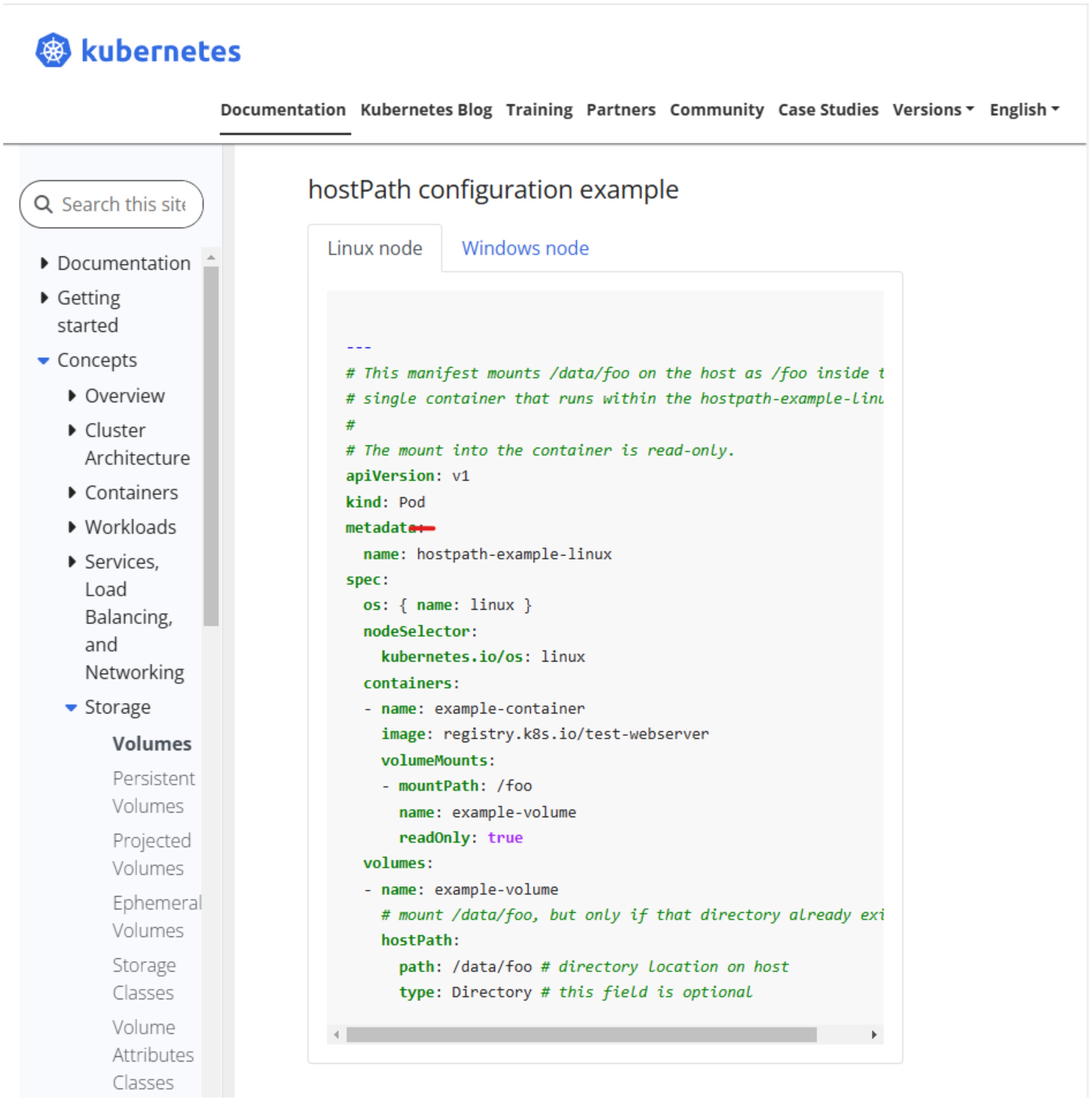
신규생성 Pod Name: fluentd, image: fluentd, namespace: default)

2. 위 조건을 참고하여 다음 조건에 맞게 볼륨마운트를 설정하시오.

1. Worker node의 도커 컨테이너 디렉토리 : /var/lib/docker/containers 동일 디렉토리로

pod에 마운트 하시오.

2. Worker node의 /var/log 디렉토리를 fluentd Pod에 동일이름의 디렉토리 마운트하시오.



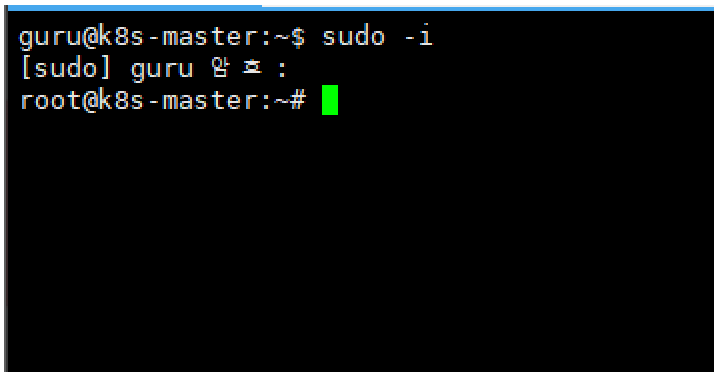
hostPath configuration example

```
---
# This manifest mounts /data/foo on the host as /foo inside t
# single container that runs within the hostpath-example-linu
#
# The mount into the container is read-only.
apiVersion: v1
kind: Pod
metadata:
  name: hostpath-example-linux
spec:
  os: { name: linux }
  nodeSelector:
    kubernetes.io/os: linux
  containers:
  - name: example-container
    image: registry.k8s.io/test-webserver
    volumeMounts:
    - mountPath: /foo
      name: example-volume
      readOnly: true
  volumes:
  - name: example-volume
    # mount /data/foo, but only if that directory already exi
    hostPath:
      path: /data/foo # directory location on host
      type: Directory # this field is optional
```

쿠버네티스 블로그에서 hostpath 검색

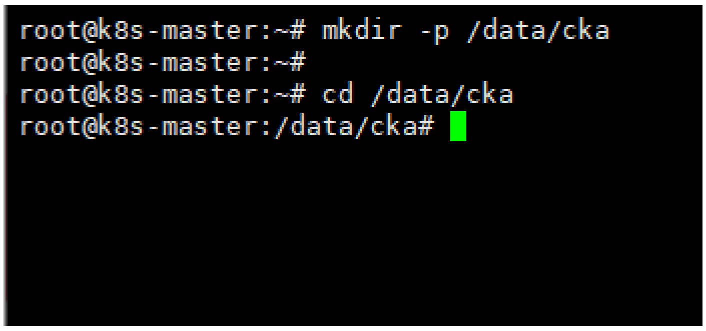
관리자 계정으로 들어가기

sudo -i



```
guru@k8s-master:~$ sudo -i
[sudo] guru 암호:
root@k8s-master:~#
```

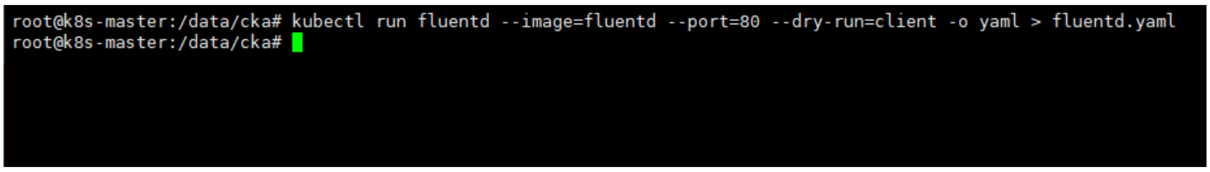
/data/cka 디렉터리 생성 및 진입



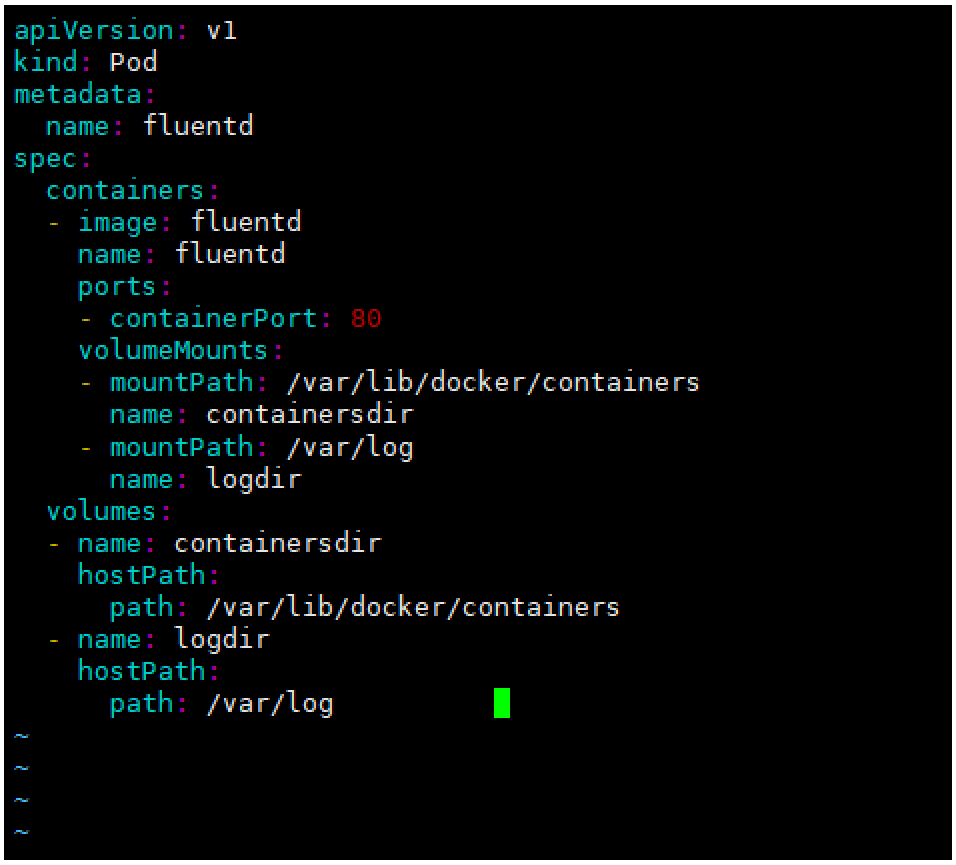
```
root@k8s-master:~# mkdir -p /data/cka
root@k8s-master:~#
root@k8s-master:~# cd /data/cka
root@k8s-master:/data/cka#
```

생성 조건에 따른 야물파일 생성

kubectl run fluentd --image=fluentd --port=80 --dry-run=client -o yaml > fluentd.yaml

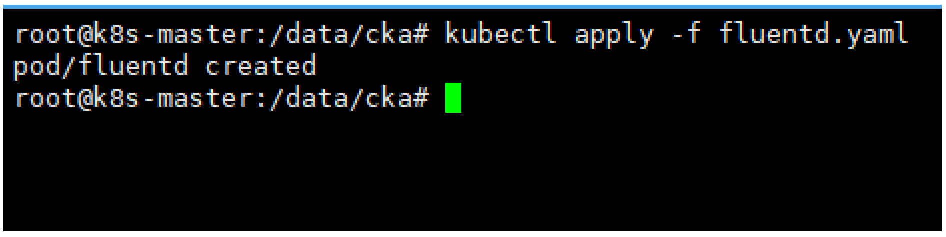


```
root@k8s-master:/data/cka# kubectl run fluentd --image=fluentd --port=80 --dry-run=client -o yaml > fluentd.yaml
root@k8s-master:/data/cka#
```



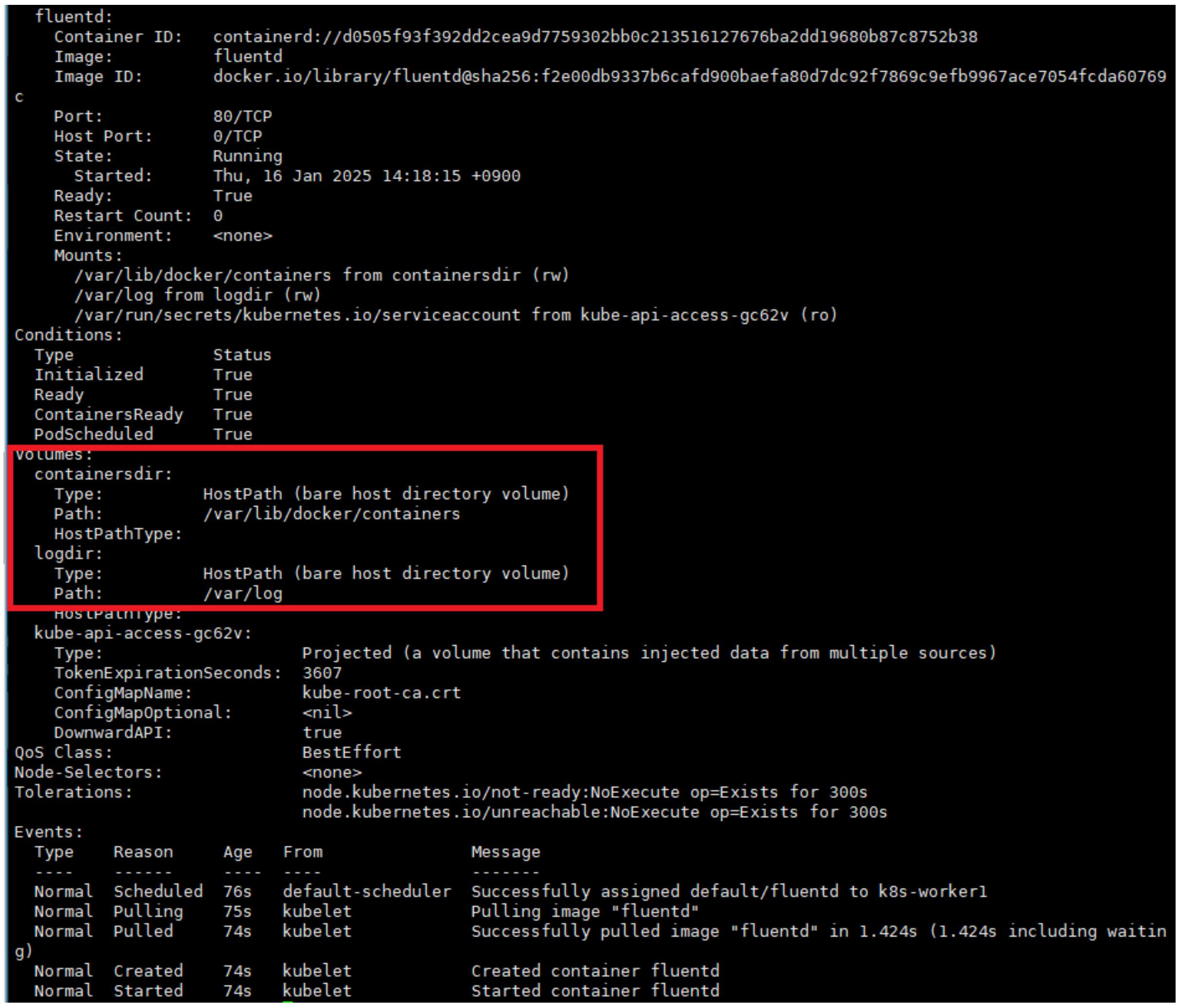
```
apiVersion: v1
kind: Pod
metadata:
  name: fluentd
spec:
  containers:
  - image: fluentd
    name: fluentd
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /var/lib/docker/containers
      name: containersdir
    - mountPath: /var/log
      name: logdir
  volumes:
  - name: containersdir
    hostPath:
      path: /var/lib/docker/containers
  - name: logdir
    hostPath:
      path: /var/log
~
~
~
~
```

vi fluentd.yaml 진입 후 수정



```
root@k8s-master:/data/cka# kubectl apply -f fluentd.yaml
pod/fluentd created
root@k8s-master:/data/cka# 
```

야물 파일 적용



```
  fluentd:
    Container ID:   containerd://d0505f93f392dd2cea9d7759302bb0c213516127676ba2dd19680b87c8752b38
    Image:          fluentd
    Image ID:       docker.io/library/fluentd@sha256:f2e00db9337b6cafd900baefa80d7dc92f7869c9efb9967ace7054fcda60769
c
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 16 Jan 2025 14:18:15 +0900
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/lib/docker/containers from containersdir (rw)
      /var/log from logdir (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-gc62v (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  containersdir:
    Type:          HostPath (bare host directory volume)
    Path:          /var/lib/docker/containers
    HostPathType:
  logdir:
    Type:          HostPath (bare host directory volume)
    Path:          /var/log
    HostPathType:
  kube-api-access-gc62v:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  76s   default-scheduler  Successfully assigned default/fluentd to k8s-worker1
  Normal  Pulling    75s   kubelet            Pulling image "fluentd"
  Normal  Pulled     74s   kubelet            Successfully pulled image "fluentd" in 1.424s (1.424s including waitin
g)
  Normal  Created    74s   kubelet            Created container fluentd
  Normal  Started    74s   kubelet            Started container fluentd
```

확인 명령어

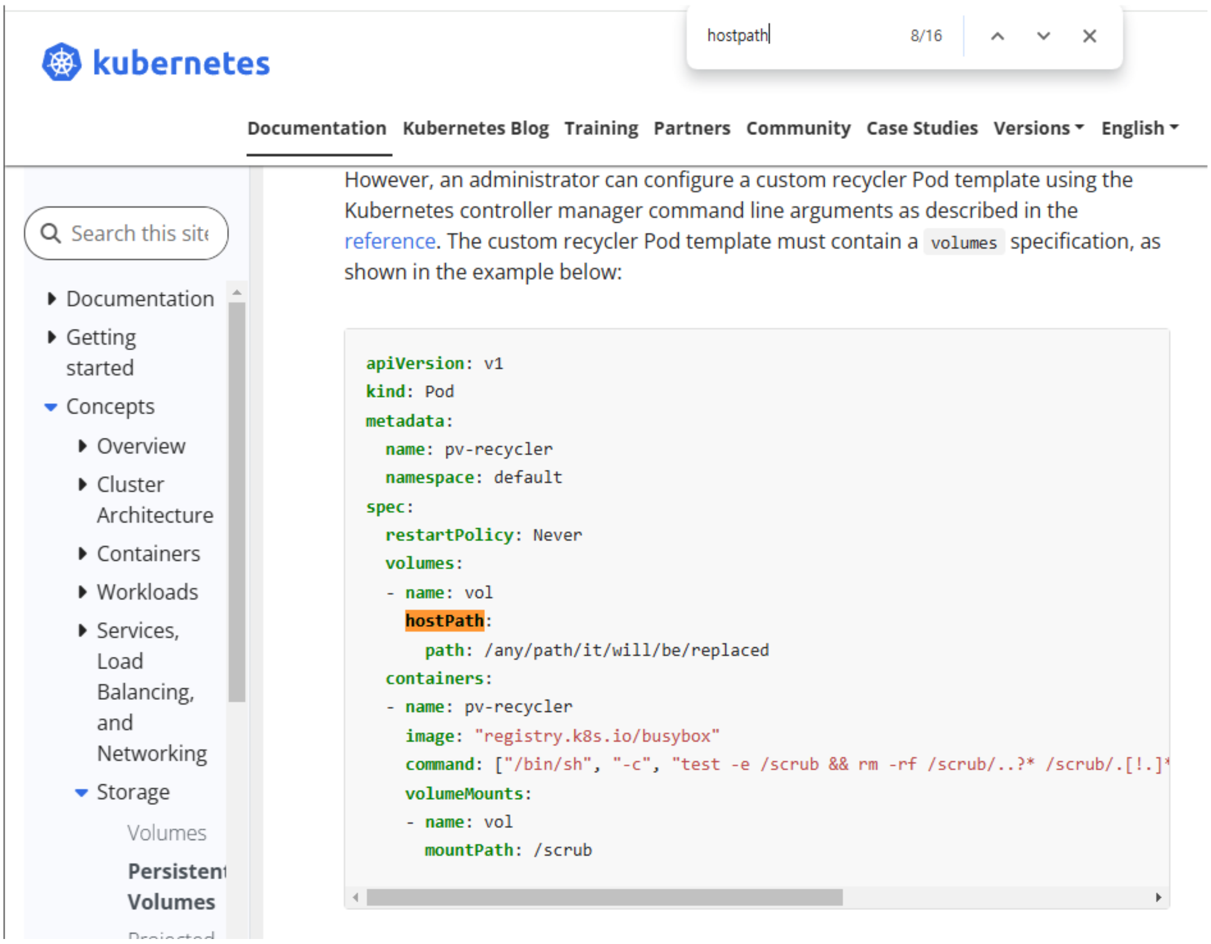
kubectl get pod fluentd

19번 문제

[Persistent Volume]

pv001라는 이름으로 size 1Gi, access mode ReadWriteMany를 사용하여 persistent volume을 생성합니다.

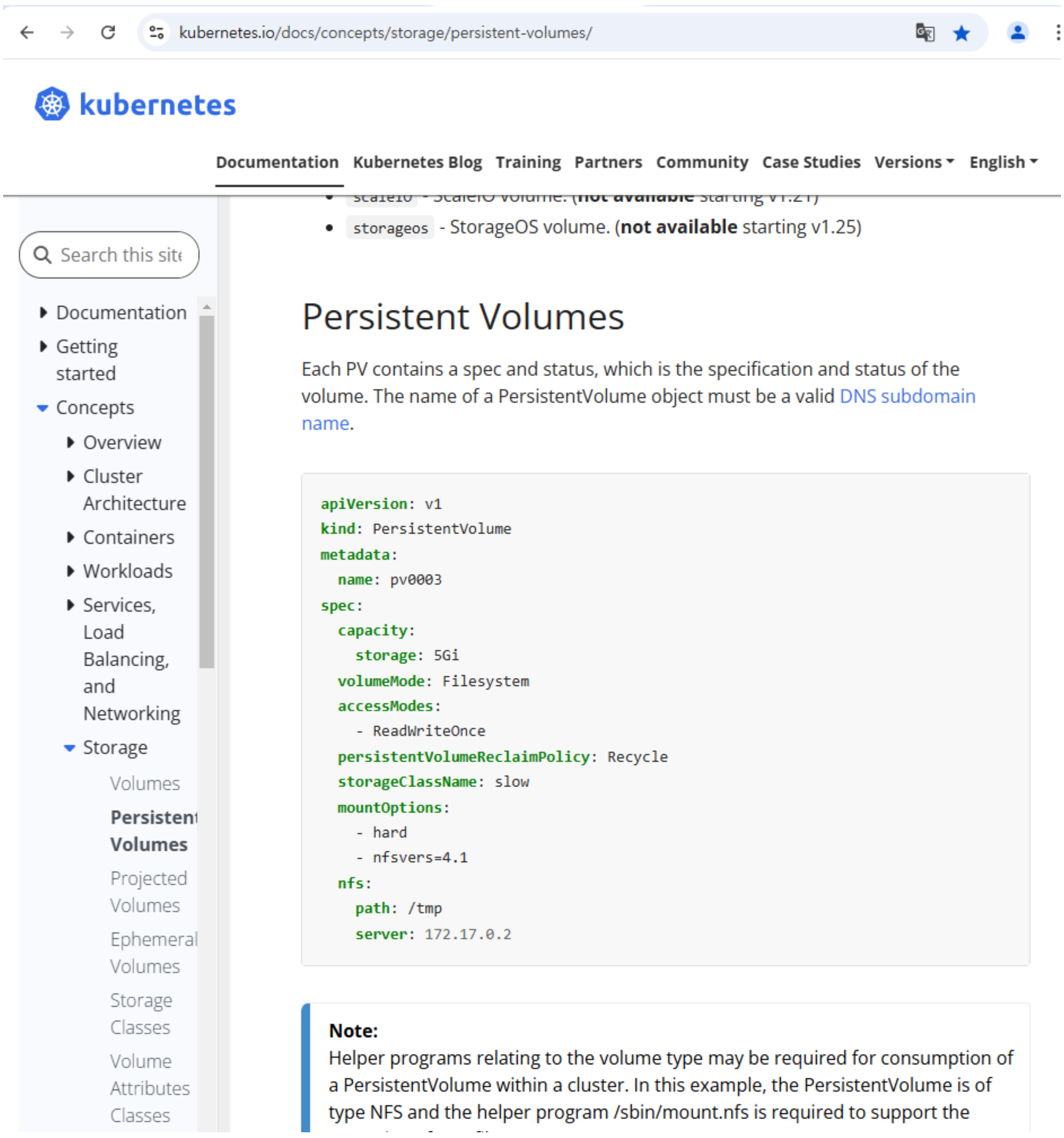
volume type은 hostPath이고 위치는 /tmp/app-config입니다.



However, an administrator can configure a custom recycler Pod template using the Kubernetes controller manager command line arguments as described in the reference. The custom recycler Pod template must contain a `volumes` specification, as shown in the example below:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pv-recycler
  namespace: default
spec:
  restartPolicy: Never
  volumes:
  - name: vol
    hostPath:
      path: /any/path/it/will/be/replaced
  containers:
  - name: pv-recycler
    image: "registry.k8s.io/busybox"
    command: ["/bin/sh", "-c", "test -e /scrub && rm -rf /scrub/..?* /scrub/.[!.]*
    volumeMounts:
    - name: vol
      mountPath: /scrub
```

hostpath 검색


kubernetes.io/docs/concepts/storage/persistent-volumes/

- `scaleio` - ScaleIO volume. (**not available** starting v1.21)
- `storageos` - StorageOS volume. (**not available** starting v1.25)

Persistent Volumes

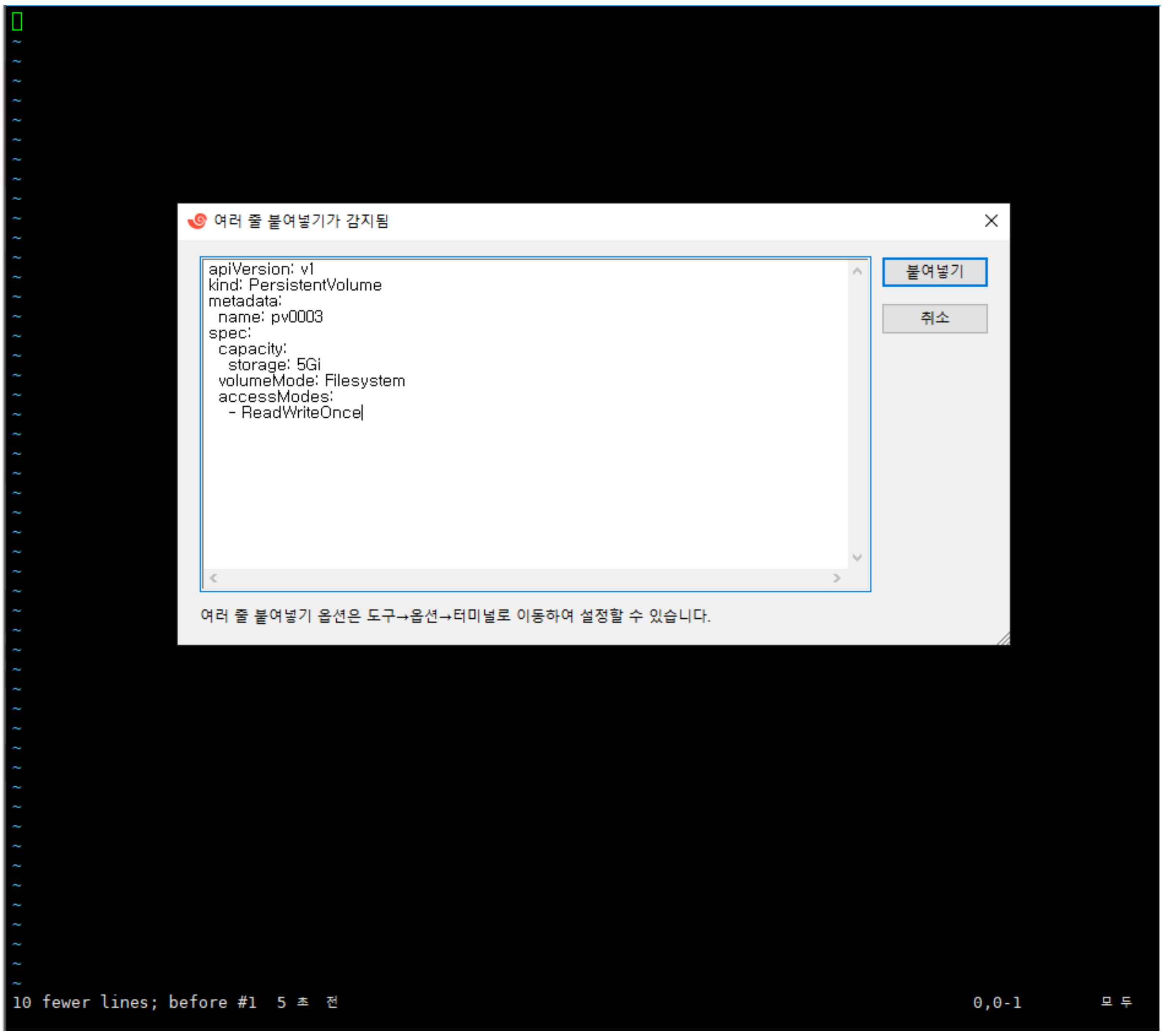
Each PV contains a spec and status, which is the specification and status of the volume. The name of a PersistentVolume object must be a valid DNS subdomain name.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

Note:
Helper programs relating to the volume type may be required for consumption of a PersistentVolume within a cluster. In this example, the PersistentVolume is of type NFS and the helper program /sbin/mount.nfs is required to support the

Persistent Volume 검색

복붙 및 내용 수정

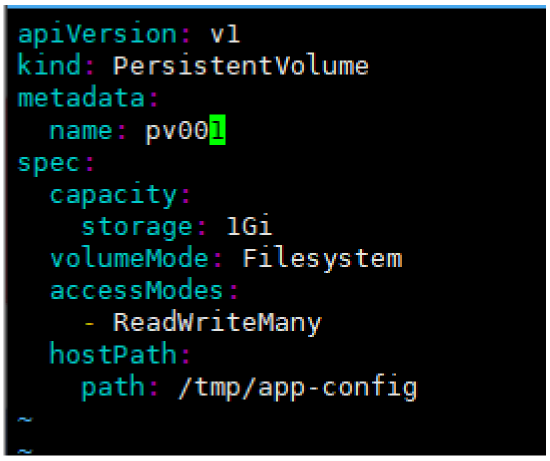


```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0003
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
```

여러 줄 붙여넣기 옵션은 도구→옵션→터미널로 이동하여 설정할 수 있습니다.

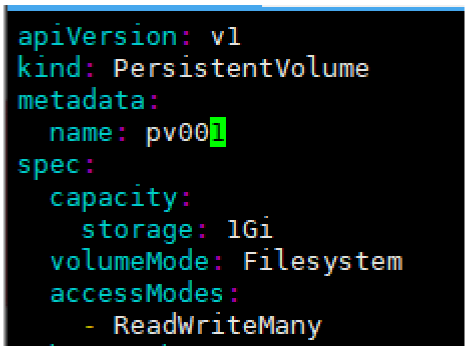
```
10 fewer lines; before #1  5 초 전                                0,0-1         모두
```

vi yaml 파일 만들기



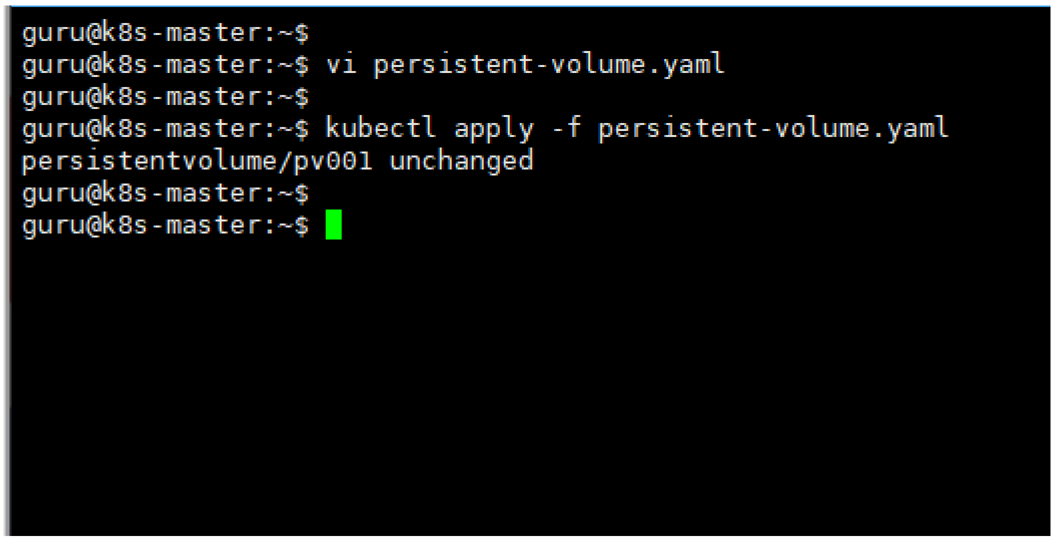
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv001
spec:
  capacity:
    storage: 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /tmp/app-config
~
~
```

복붙 및 내용 수정



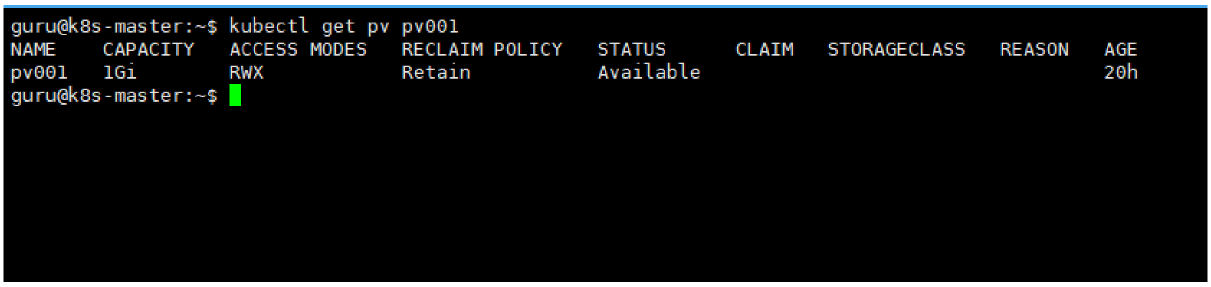
```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv001
spec:
  capacity:
    storage: 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
```

문제 요구사항으로 내용 수정



```
guru@k8s-master:~$
guru@k8s-master:~$ vi persistent-volume.yaml
guru@k8s-master:~$
guru@k8s-master:~$ kubectl apply -f persistent-volume.yaml
persistentvolume/pv001 unchanged
guru@k8s-master:~$
guru@k8s-master:~$
```

yaml 파일 적용



```
guru@k8s-master:~$ kubectl get pv pv001
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM    STORAGECLASS   REASON   AGE
pv001    1Gi        RWX            Retain           Available                                    20h
guru@k8s-master:~$
```

확인 명령어

kubectl get pv pv