대우능력개발원
DAEWOO DEVELOPMENT OF ABILITY

CKA 프로젝트

BSFAN | 김태경
박종승
윤재영
김효은

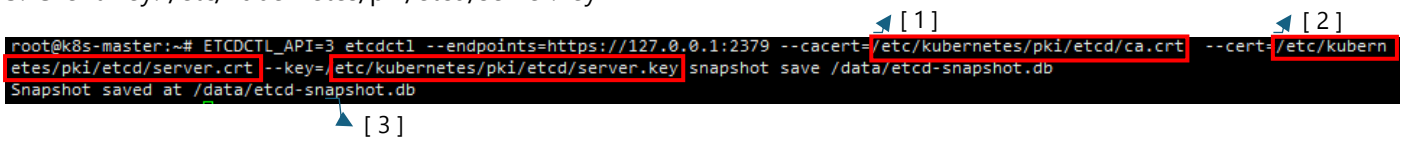
2개의 작업 클러스터 생성, 백업, 업그레이드

1. ETCD 백업

1. https://127.0.0.1:2379에서 실행 중인 etcd의 snapshot을 생성하고 snapshot을 /data/etcd-snapshot.db에 저장합니다.
2. 다음 다시 스냅샷을 복원합니다.
3. 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

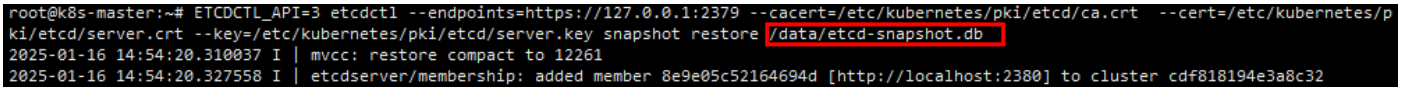
etcd 백업

1. CA certificate: /etc/kubernetes/pki/etcd/ca.crt
2. Client certificate: /etc/kubernetes/pki/etcd/server.crt
3. Client key: /etc/kubernetes/pki/etcd/server.key

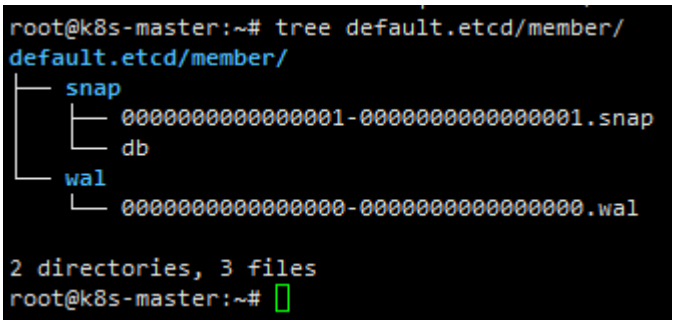


etcd 복원

/data/etcd- snapshot.db



확인



2개의 작업 클러스터 생성, 백업, 업그레이드

2. Cluster Upgrade

1. 마스터 노드의 모든 Kubernetes control plane및 node 구성 요소를 버전 1.29.6-1.1 버전으로 업그레이드합니다.
2. master 노드를 업그레이드하기 전에 drain 하고 업그레이드 후에 uncordon해야 합니다.

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

master drain

kubectl drain k8s-master --ignore-daemonsets

```
root@k8s-master:~# kubectl drain k8s-master --ignore-daemonsets
node/k8s-master already cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-vcgrw, kube-system/weave-net-lchss
evicting pod kube-system/coredns-5dd5756b68-z4gmf
evicting pod kube-system/coredns-5dd5756b68-6fgkp
pod/coredns-5dd5756b68-z4gmf evicted
pod/coredns-5dd5756b68-6fgkp evicted
node/k8s-master drained
```

master uncordon

kubectl uncordon k8s-master

```
root@k8s-master:~# kubectl uncordon k8s-master
node/k8s-master uncordoned
```

역할 기반 제어

3. 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.

- api-access라는 새로운 namespace에 pod-viewer라는 이름의 Service Account를 만듭니다.
- podreader-role이라는 이름의 Role과 podreader-rolebinding이라는 이름의 RoleBinding을 만듭니다.
- 앞서 생성한 ServiceAccount를 API resource Pod에 대하여 watch, list, get을 허용하도록 매핑하시오.

네임스페이스 생성

```
guru@k8s-master:~$ kubectl create ns api-access
namespace/api-access created
```

네임스페이스 확인

```
guru@k8s-master:~$ kubectl get ns api-access
NAME         STATUS   AGE
api-access   Active   20m
```

Service Accont 생성

sa pod-viewer -n api-access

```
guru@k8s-master:~$ kubectl create sa pod-viewer -n api-access
serviceaccount/pod-viewer created
```

Service Accont 확인

kubectl get sa -n api-access

```
guru@k8s-master:~$ kubectl get sa -n api-access
NAME         SECRETS   AGE
default      0         4m59s
pod-viewer   0         18s
```

Role 생성 - name: podreader-role / resource pod에 대하여 watch, list, get 허용

kubectl create role podreader-role --resource=pod --verb=watch,list,get -n api-access

```
guru@k8s-master:~$ kubectl create role podreader-role --resource=pod --verb=watch,list,get, -n api-access
Warning: '' is not a standard resource verb
role.rbac.authorization.k8s.io/podreader-role created
```

role 확인

kubectl describe role -n api-access

```
guru@k8s-master:~$ kubectl describe role -n api-access
Name:         podreader-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources  Non-Resource URLs  Resource Names  Verbs
  ---------  -----------------  --------------  -----
  pods       []                 []              [watch list get ]
```

RoleBinding 생성 - name: podreader-rolebinding

kubectl create rolebinding podreader-rolebinding --role=podreader-role --serviceaccount=api-access:pod-viewer -n api-access

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

Rolebindind 확인

kubectl describe rolebinding -n api-access

```
guru@k8s-master:~$ kubectl describe rolebinding -n api-access
Name:         podreader-rolebinding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  podreader-role
Subjects:
  Kind            Name        Namespace
  ----            ----        ---------
  ServiceAccount  pod-viewer  api-access
```

역할 기반 제어

4. Service Account, ClusterRole, ClusterRoleBinding 생성

애플리케이션 배포를 위해 새로운 ClusterRole을 생성하고 특정 namespace의 ServiceAccount를 바인드하시오.

1. 다음의 resource type에서만 Create가 허용된 ClusterRole deployment-clusterrole을 생성합니다.
   - Resource Type: Deployment StatefulSet DaemonSet
2. 미리 생성된 namespace api-access 에 cicd-token이라는 새로운 ServiceAccount를 만듭니다.
3. ClusterRole deployment-clusterrole을 namespace api-access 로 제한된 새 ServiceAccount cicd-token 에 바인딩하세요.

2. 네임스페이스 api-access에 cicd-token 새로운 sa 생성

```
guru@k8s-master:~$ kubectl create sa cicd-token -n api-access
serviceaccount/cicd-token created
```

1. resource type에서만 Create가 허용된 ClusterRole deployment-clusterrole을 생성

```
guru@k8s-master:~$ kubectl create clusterrole deployment-clusterrole --resource=deployment,statefulset,daemonset --v
erb=create
clusterrole.rbac.authorization.k8s.io/deployment-clusterrole created
```

3. ClusterRole deployment-clusterrole을 namespace api-access 로 제한된 새 ServiceAccount cicd-token에 바인딩

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

확인

Service Account

kubectl describe sa -n api-access cicd-token

```
guru@k8s-master:~$ kubectl describe sa -n api-access cicd-token
Name:                cicd-token
Namespace:           api-access
Labels:              <none>
Annotations:         <none>
Image pull secrets:  <none>
Mountable secrets:   <none>
Tokens:              <none>
Events:              <none>
guru@k8s-master:~$
```

확인

ClusterRole

kubectl describe -n api-access clusterrole deployment-clusterrole

```
guru@k8s-master:~$ kubectl describe -n api-access clusterrole deployment-clusterrole
Name:         deployment-clusterrole
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources          Non-Resource URLs  Resource Names  Verbs
  ---------          -----------------  --------------  -----
  daemonsets.apps    []                 []              [create]
  deployments.apps   []                 []              [create]
  statefulsets.apps  []                 []              [create]
```

확인

ClusterRole Binding

kubectl describe -n api-access clusterrolebinding deployment-clusterrolebinding

```
guru@k8s-master:~$ kubectl describe -n api-access clusterrolebinding deployment-clusterrolebinding
Name:         deployment-clusterrolebinding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  deployment-clusterrole
Subjects:
  Kind            Name        Namespace
  ----            ----        ---------
  ServiceAccount  cicd-token  api-access
```

노드관리

5. 노드 비우기

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

node 확인

kubectl get no

```
guru@k8s-master:~$ kubectl get no
NAME          STATUS   ROLES           AGE    VERSION
k8s-master    Ready    control-plane   293d   v1.28.8
k8s-worker1   Ready    <none>          293d   v1.28.8
k8s-worker2   Ready    <none>          293d   v1.28.8
```

테스트 deploy 파일 생성

kubectl create deploy deploy-test --image=nginx:1.14 --replicas=2

```
guru@k8s-master:~$ kubectl create deploy deploy-test --image=nginx:1.14 --replicas=2
deployment.apps/deploy-test created
```

kubectl get po -o wide

```
guru@k8s-master:~$ kubectl get po -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP          NODE          NOMINATED NODE   READINE
SS GATES
deploy-test-5cd79d8497-fmj72   1/1     Running   0          46s   10.32.0.3   k8s-worker2   <none>           <none>
deploy-test-5cd79d8497-gc7r2   1/1     Running   0          46s   10.32.0.2   k8s-worker2   <none>           <none>
```

k8s-worker2 노드 스케줄링 Disabled 설정 및 확인

kubectl cordon k8s-worker2

kubectl get no

```
guru@k8s-master:~$ kubectl cordon k8s-worker2
node/k8s-worker2 cordoned
guru@k8s-master:~$ kubectl get no
NAME          STATUS                     ROLES           AGE    VERSION
k8s-master    Ready                      control-plane   293d   v1.28.8
k8s-worker1   Ready                      <none>          293d   v1.28.8
k8s-worker2   Ready,SchedulingDisabled   <none>          293d   v1.28.8
```

node drain

kubectl drain k8s-worker2 --ignore-daemonsets

```
guru@k8s-master:~$ kubectl drain k8s-worker2 --ignore-daemonsets
node/k8s-worker2 already cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-7rl68, kube-system/weave-net-sbnc5
evicting pod default/deploy-test-5cd79d8497-gc7r2
evicting pod default/deploy-test-5cd79d8497-fmj72
pod/deploy-test-5cd79d8497-gc7r2 evicted
pod/deploy-test-5cd79d8497-fmj72 evicted
node/k8s-worker2 drained
```

확인

kubectl get po -o wide1

kubectl get no

```
guru@k8s-master:~$ kubectl get po -o wide
NAME                           READY   STATUS    RESTARTS   AGE     IP          NODE          NOMINATED NODE   READI
NESS GATES
deploy-test-5cd79d8497-7vtt5   1/1     Running   0          2m10s   10.40.0.2   k8s-worker1   <none>           <none
>
deploy-test-5cd79d8497-kxl46   1/1     Running   0          2m10s   10.40.0.1   k8s-worker1   <none>           <none
>
guru@k8s-master:~$ kubectl get no
NAME          STATUS                     ROLES           AGE    VERSION
k8s-master    Ready                      control-plane   293d   v1.28.8
k8s-worker1   Ready                      <none>          293d   v1.28.8
k8s-worker2   Ready,SchedulingDisabled   <none>          293d   v1.28.8
```

노드관리

6. Pod Scheduling

다음의 조건으로 pod를 생성하세요.

- Name: eshop-store
- Image: nginx
- Nodeselector: disktype=ssd

worker1,2 노드에 라벨링 추가

kubectl label node worker1 disktype=ssd

kubectl label node worker2 disktype=hdd

```
guru@k8s-master:~$ kubectl label node k8s-worker1 disktype=ssd
node/k8s-worker1 labeled
guru@k8s-master:~$ kubectl label node k8s-worker2 disktype=hdd
node/k8s-worker2 labeled
```

라벨링 추가 여부 확인

```
guru@k8s-master:~$ kubectl get no -L disktype
NAME          STATUS   ROLES           AGE    VERSION   DISKTYPE
k8s-master    Ready    control-plane   293d   v1.28.8
k8s-worker1   Ready    <none>          293d   v1.28.8   ssd
k8s-worker2   Ready    <none>          293d   v1.28.8   hdd
```

pod 생성을 위한 yaml파일 작성

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

vi eshop-store.yaml 수정

수정 전

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: eshop-store
  name: eshop-store
spec:
  containers:
  - image: nginx
    name: eshop-store
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

수정 후

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

yaml 적용

kubectl apply -f eshop-store.yaml

```
guru@k8s-master:~$ kubectl apply -f eshop-store.yaml
pod/eshop-store created
```

pod 확인

kubectl get po eshop-store -o wide

```
guru@k8s-master:~$ kubectl get po eshop-store -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP          NODE          NOMINATED NODE   READINESS G
ATES
eshop-store   1/1     Running   0          70s   10.40.0.1   k8s-worker1   <none>           <none>
```

파드 생성

7. 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"
- ommand: /bin/sh
- args: "-c", "while true; do echo $(CERT); sleep 10;done"

네임스페이스 생성

kubectl create ns cka-exam

```
guru@k8s-master:~$ kubectl create ns cka-exam
namespace/cka-exam created
```

네임스페이스 확인

```
guru@k8s-master:~$ kubectl get ns
NAME              STATUS   AGE
cka-exam          Active   105s
default           Active   293d
kube-node-lease   Active   293d
kube-public       Active   293d
kube-system       Active   293d
```

pod 생성 및 환경변수 조건 추가하여 yaml파일 생성

```
guru@k8s-master:~$ kubectl run pod-01 --image=busybox -n cka-exam --env=CERT=CKA-cert --dry-run=client -o yaml > pod
-01.yaml
guru@k8s-master:~$
```

vi pod-01.yaml 확인 및 수정

- pod Name: pod-01
- image: busybox
- 환경변수 : CERT = "CKA-cert"
- ommand: /bin/sh
- args: "-c", "while true; do echo $(CERT); sleep 10;done"

수정 전

```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: pod-01
  name: pod-01
  namespace: cka-exam
spec:
  containers:
  - env:
    - name: CERT
      value: CKA-cert
    image: busybox
    name: pod-01
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

수정 후

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

yaml 적용

kubectl apply -f pod-01.yaml

```
guru@k8s-master:~$ kubectl apply -f pod-01.yaml
pod/pod-01 created
```

pod 확인

kubectl get pod-01 -n cka-exam -o wide

```
guru@k8s-master:~$ kubectl get po pod-01 -n cka-exam -o wide
NAME     READY   STATUS    RESTARTS   AGE    IP          NODE          NOMINATED NODE   READINESS GATES
pod-01   1/1     Running   0          113s   10.32.0.2   k8s-worker2   <none>           <none>
```

파드 생성

8. Static Pod 생성

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

- pod name : nginx-static-pod
- image : nginx
- port : 80

worker 1 접속

```
guru@k8s-master:~$ ssh k8s-worker1
The authenticity of host 'k8s-worker1 (10.100.0.106)' can't be established.
ECDSA key fingerprint is SHA256:p6Kg7OSM9ozOHdkSA5fz9qlEScr9wis1+l7XjJpNtCI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'k8s-worker1,10.100.0.106' (ECDSA) to the list of known hosts.
guru@k8s-worker1's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-79-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.

219 updates can be applied immediately.
174 of these updates are standard security updates.
추가 업데이트를 확인하려면 apt list --upgradable 을 실행하세요.

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
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Mon Apr  1 11:03:01 2024 from 10.100.0.2
guru@k8s-worker1:~$
```

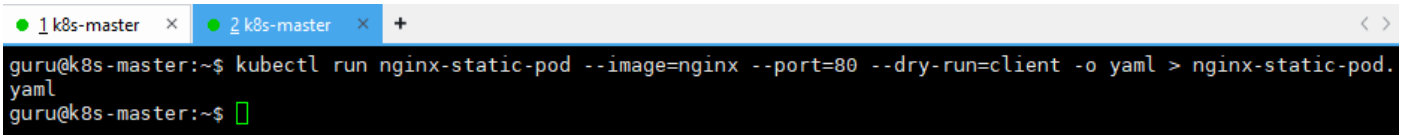
worker1의 static-pod 위치 확인

1. cd /var/lib/kubelet 이동
2. cat config.yaml | grep -i static 으로 위치 확인
3. cd /etc/kubernetes/manifests 이동

```
guru@k8s-worker1:~$ cd /var/lib/kubelet
guru@k8s-worker1:/var/lib/kubelet$ cat config.yaml | grep -i static
staticPodPath: /etc/kubernetes/manifests
guru@k8s-worker1:/var/lib/kubelet$ cd /etc/kubernetes/manifests
guru@k8s-worker1:/etc/kubernetes/manifests$
```

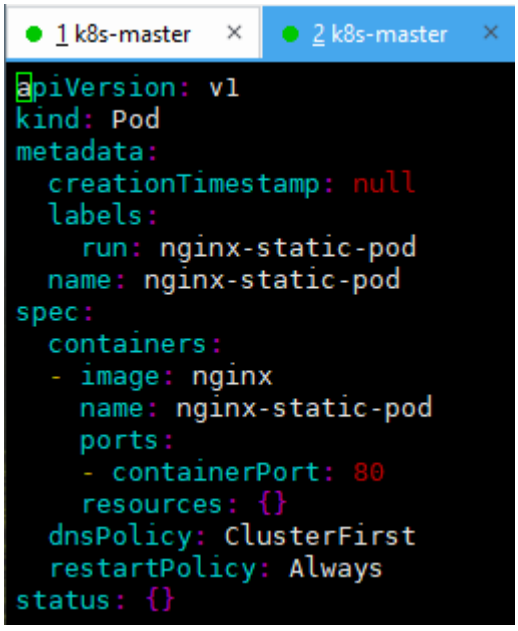
(추가 세션) master 창에서 nginx-static-pod.yaml 파일 생성

kubectl run nginx-static-pod --image=nginx --port=80 --dry-run=client -o yaml > nginx-static-pod.yaml

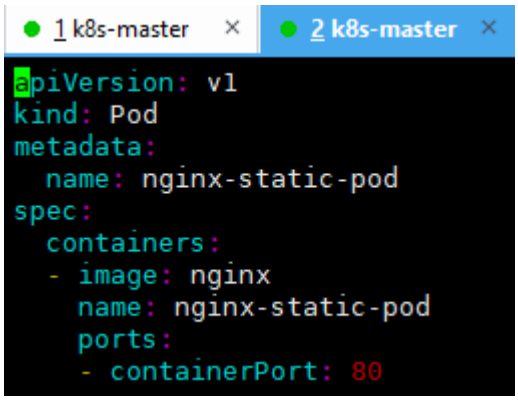


vi nginx-static-pod.yaml 수정 - worker1 노드에 수정된 코드 내용을 붙여넣을 예정

수정 전



수정 후

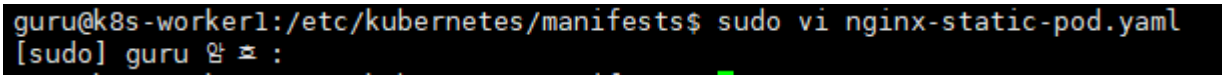


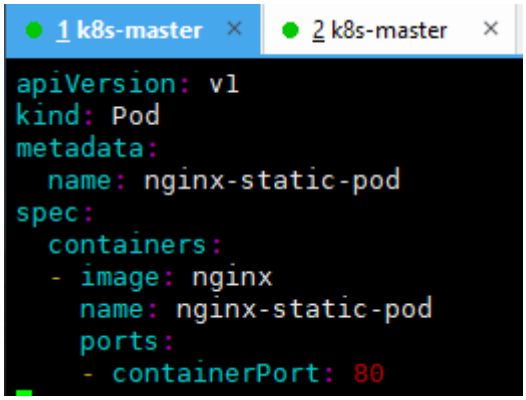
- pod name : nginx-static-pod
- image : nginx
- port : 80

worker1 에서 yaml 파일 생성

[** worker1 사용자 계정으로는 입력한 내용이 저장되지 않아 root 계정으로 진입 **]

sudo vi nginx-static-pod.yaml (수정한 코드 붙여넣기)





- (추가세션) master 창에서 수정된 코드 복사 후 worker1에서 실행한 vi nginx-static-pod. yaml 안에 코드 붙여넣기
- 이 후 마스터 창 vi nginx-static-pod. yaml 나가기 :q!

master 노드에서 'nginx-static-pod-k8s-worker1' pod 확인

kubectl get po nginx-static-pod-k8s-worker1 -o wide

```
guru@k8s-master:~$ kubectl get po nginx-static-pod-k8s-worker1 -o wide
NAME                          READY   STATUS    RESTARTS       AGE   IP          NODE          NOMINATED NODE   REA
DINESS GATES
nginx-static-pod-k8s-worker1   1/1     Running   2 (9m9s ago)   25m   10.40.0.2   k8s-worker1   <none>           <no
ne>
```

파드 생성

9. 로그 확인

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

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

nginx-static-pod-worker1 작동 여부 확인

kubectl get po nginx-static-pod-worker1

```
guru@k8s-master:~$ kubectl get po nginx-static-pod-k8s-worker1
NAME                          READY   STATUS    RESTARTS       AGE
nginx-static-pod-k8s-worker1   1/1     Running   3 (2m57s ago)   37m
```

root 계정 전환 (sudo -i)

/opt/REPORT/2023 디렉터리 생성

```
guru@k8s-master:~$ sudo -i
[sudo] guru 암호 :
root@k8s-master:~# mkdir -p /opt/REPORT/2023/
root@k8s-master:~#
```

worker1 log 모니터링

```
root@k8s-master:~# mkdir -p /opt/REPORT/2023/
root@k8s-master:~# kubectl logs nginx-static-pod-k8s-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
2025/01/19 15:44:13 [notice] 1#1: using the "epoll" event method
2025/01/19 15:44:13 [notice] 1#1: nginx/1.27.3
2025/01/19 15:44:13 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/19 15:44:13 [notice] 1#1: OS: Linux 5.15.0-79-generic
2025/01/19 15:44:13 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/19 15:44:13 [notice] 1#1: start worker processes
2025/01/19 15:44:13 [notice] 1#1: start worker process 29
2025/01/19 15:44:13 [notice] 1#1: start worker process 30
root@k8s-master:~#
```

/opt/REPORT/2023/pod-log에 기록 및 확인

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

cat /opt/REPORT/2023/pod-log

```
root@k8s-master:~# kubectl logs nginx-static-pod-k8s-worker1 > /opt/REPORT/2023/pod-log
root@k8s-master:~#
root@k8s-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
2025/01/19 15:44:13 [notice] 1#1: using the "epoll" event method
2025/01/19 15:44:13 [notice] 1#1: nginx/1.27.3
2025/01/19 15:44:13 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/19 15:44:13 [notice] 1#1: OS: Linux 5.15.0-79-generic
2025/01/19 15:44:13 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/19 15:44:13 [notice] 1#1: start worker processes
2025/01/19 15:44:13 [notice] 1#1: start worker process 29
2025/01/19 15:44:13 [notice] 1#1: start worker process 30
```

파드 생성

10. 로그 확인

4개의 컨테이너를 동작시키는 eshop-frontend Pod를 생성하시오.

- pod image : nginx, redis, memcached, consul

eshop-frontend Pod 생성 및 적용 / 확인

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

vi eshop-frontend.yaml

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

pod image :

- nginx
- redis
- memcached
- consul

kubectl apply -f eshop-frontend.yaml

kubectl get po

```
guru@k8s-master:~$ kubectl run eshop-frontend --image=nginx --dry-run=client -o yaml > eshop-frontend.yaml
guru@k8s-master:~$ vi eshop-frontend.yaml
guru@k8s-master:~$ kubectl apply -f eshop-frontend.yaml
pod/eshop-frontend created
guru@k8s-master:~$ kubectl get po
NAME                          READY   STATUS              RESTARTS      AGE
eshop-frontend                0/4     ContainerCreating   0             11s
nginx-static-pod-k8s-worker1  1/1     Running             3 (10m ago)   3h40m
guru@k8s-master:~$ kubectl get po
NAME                          READY   STATUS             RESTARTS      AGE
eshop-frontend                3/4     ImagePullBackOff   0             108s
nginx-static-pod-k8s-worker1  1/1     Running            3 (12m ago)   3h42m
```

쿠버네티스 컨트롤러와 네트워킹

11. Rolling Update & Roll Back

1. Deployment를 이용해 nginx 파드를 3개 배포한 다음 컨테이너 이미지 버전을 rolling update하고 update record를 기록합니다.
2. 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

1. Deployment를 이용해 nginx 파드를 3개 배포한 다음 컨테이너 이미지 버전을 rolling update하고 update record를 기록

kubectl create deployment eshop-payment --image=nginx:1.16 --replicas=3 --dry-run=client -o yaml > eshop-payment.yaml

```
guru@k8s-master:~$ kubectl create deployment eshop-payment --image=nginx:1.16 --replicas=3 --dry-run=client -o yaml
> eshop-payment.yaml
```

vi eshop eshop-payment.yaml

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

- name: eshop-payment
- Image : nginx
- Image version: 1.16
- update image version: 1.17
- label: app=payment, environment=production

kubectl apply -f eshop-payment.yaml –record

```
guru@k8s-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
```

kubectl set image deploy eshop-payment nginx=nginx:1.17 –record

```
guru@k8s-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
```

버전 확인

kubectl get deploy,po | grep -i eshop-payment

kubectl describe po eshop-payment-5c4cc77dcd-58fb2 | grep -i nginx

kubectl rollout history deploy eshop-payment

```
guru@k8s-master:~$ kubectl get deploy,po | grep -i eshop-payment
deployment.apps/eshop-payment   3/3     3            3           77s
pod/eshop-payment-5c4cc77dcd-58fb2   1/1     Running   0           20s
pod/eshop-payment-5c4cc77dcd-gj46h   1/1     Running   0           28s
pod/eshop-payment-5c4cc77dcd-rd8kp   1/1     Running   0           17s
guru@k8s-master:~$ kubectl describe po eshop-payment-5c4cc77dcd-58fb2 | grep -i nginx
  nginx:
    Image:          nginx:1.17
    Image ID:       docker.io/library/nginx@sha256:6fff55753e3b34e36e24e37039ee9eae1fe38a6420d8ae16ef37c92d1eb26699
  Normal  Pulled     70s   kubelet            Container image "nginx:1.17" already present on machine
  Normal  Created    70s   kubelet            Created container nginx
  Normal  Started    70s   kubelet            Started container nginx
guru@k8s-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
```

2. 컨테이너 이미지를 previous version으로 roll back

kubectl rollout undo deploy eshop-payment

```
guru@k8s-master:~$ kubectl rollout undo deploy eshop-payment
deployment.apps/eshop-payment rolled back
```

확인

kubectl get po

kubectl describe po eshop-payment-7ddf6d4467-ddxsk | grep -i nginx

```
guru@k8s-master:~$ kubectl get po
NAME                             READY   STATUS    RESTARTS      AGE
eshop-payment-7ddf6d4467-ddxsk   1/1     Running   0             36s
eshop-payment-7ddf6d4467-pfwnq   1/1     Running   0             35s
eshop-payment-7ddf6d4467-v5w5c   1/1     Running   0             33s
nginx-static-pod-k8s-worker1     1/1     Running   3 (70m ago)   11m
guru@k8s-master:~$ kubectl describe po eshop-payment-7ddf6d4467-ddxsk | grep -i nginx
  nginx:
    Image:          nginx:1.16
    Image ID:       docker.io/library/nginx@sha256:d20aa6d1cae56fd17cd458f4807e0de462caf2336f0b70b5eeb69fcaaf30dd9c
  Normal  Pulled     74s   kubelet            Container image "nginx:1.16" already present on machine
  Normal  Created    74s   kubelet            Created container nginx
  Normal  Started    74s   kubelet            Started container nginx
```

쿠버네티스 컨트롤러와 네트워킹

12. ClusterIP

1. 'devops' namespace에서 deployment eshop-order를 다음 조건으로 생성하시오.
    - image: nginx, replicas: 2, label: name=order
2. 'eshop-order' deployment의 Service를 만드세요.
    - Service Name: eshop-order-svc
    - Type: ClusterIP
    - Port: 80

kubectl get ns

kubectl create ns devops

kubectl get ns

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

vi eshop-order.yaml

kubectl apply -f eshop-order.yaml

kubectl expose deploy eshop-order -n devops --name=eshop-order-svc --port=80 --target-port=80

kubectl get deploy eshop-order -n devops

kubectl get svc eshop-order-svc -n devops

```
guru@k8s-master:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   290d
kube-node-lease   Active   290d
kube-public       Active   290d
kube-system       Active   290d
guru@k8s-master:~$ kubectl create ns devops
namespace/devops created
guru@k8s-master:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   290d
devops            Active   7s
kube-node-lease   Active   290d
kube-public       Active   290d
kube-system       Active   290d
guru@k8s-master:~$ kubectl create deploy eshop-order -n devops --image=nginx --replicas=2 --dry-run=client -
o yaml > eshop-order.yaml
guru@k8s-master:~$ kubectl apply -f eshop-order.yaml
deployment.apps/eshop-order created
guru@k8s-master:~$ vi eshop-order.yaml
guru@k8s-master:~$ kubectl apply -f eshop-order.yaml
deployment.apps/order created
guru@k8s-master:~$ kubectl espose deploy eshop-order -n devops --name=eshop-order-svc --port=80 --target-por
t=80
error: unknown command "espose" for "kubectl"

Did you mean this?
        expose
guru@k8s-master:~$ kubectl expose deploy eshop-order -n devops --name=eshop-order-svc --port=80 --target-por
t=80
service/eshop-order-svc exposed
guru@k8s-master:~$ kubectl get deploy,svc eshop-order -n devops
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
eshop-order   2/2     2            2           5m38s
Error from server (NotFound): services "eshop-order" not found
guru@k8s-master:~$ kubectl get deploy eshop-order -n devops
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
eshop-order   2/2     2            2           5m52s
guru@k8s-master:~$ kubectl get svc eshop-order-svc -n devops
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
eshop-order-svc   ClusterIP   10.98.103.237   <none>        80/TCP    59s
```

쿠버네티스 컨트롤러와 네트워킹

13. NodePort

1. 'front-end' deployment를 다음 조건으로 생성하시오.
   - image: nginx, replicas: 2, label: run=nginx

2. 'front-end' deployment의 nginx 컨테이너를 expose하는 'front-end-nodesvc'라는 새 service를 만듭니다.

~~3. Front-end로 동작중인 Pod에는 node의 **30200** 포트로 접속되어야 합니다.~~

kubectl create deploy front-end --image=nginx --replicas=2 --dry-run=client -o yaml > front-end,yaml

vi front-end,yaml

kubectl expose deploy front-end --name= front-end-nodesvc --type=NodePort --port=80 --target-port=80 --dry-run=client -o yaml > front-end-nodesvc.yaml

vi front-end-nodesvc.yaml

kubectl apply -f vi front-end-nodesvc.yaml

kubectl get deploy,svc

vi front-end-nodesvc.yaml

kubectl apply -f front-end-nodesvc.yaml

kubectl get deploy,svc

NetworkPolicy & Ingress

14. Network Policy (정책)

1. customera, customerb를 생성한 후, 각각 PARTITION=customera, PARTITION=customerb를 라벨링하시오.
2. default namespace에 다음과 같은 pod를 생성하세요.

- name: pocimage: nginxport: 80
- label: app=poc
- "partition=customera"를 사용하는 namespace에서만 poc의

3. 80포트로 연결할 수 있도록 default namespace에 'allow-web-from-customera'라는 network Policy를 설정하세요.보안 정책상 다른 namespace의 접근은 제한합니다.

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

✔ customera, customerb 네임스페이스 생성

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

✔ 네임스페이스 확인

```
guru@k8s-master:~$ kubectl get ns
NAME              STATUS   AGE
api-access        Active   3d20h
customera         Active   2m31s
customerb         Active   2m29s
default           Active   291d
devops            Active   2d20h
ing-internal      Active   19h
kube-node-lease   Active   291d
kube-public       Active   291d
kube-system       Active   291d
guru@k8s-master:~$
```

✔ customera, customerb 파티션 및 라벨링

```
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
```

✔ 네임스페이스 Partition 및 라벨 확인

```
guru@k8s-master:~$ kubectl get ns
NAME              STATUS   AGE
api-access        Active   3d20h
customera         Active   20s
customerb         Active   19s
default           Active   291d
devops            Active   2d20h
ing-internal      Active   19h
kube-node-lease   Active   291d
kube-public       Active   291d
kube-system       Active   291d
guru@k8s-master:~$ kubectl get ns -L partition
NAME              STATUS   AGE     PARTITION
api-access        Active   3d20h
customera         Active   25s     customera
customerb         Active   24s     customerb
default           Active   291d
devops            Active   2d20h
ing-internal      Active   19h
kube-node-lease   Active   291d
kube-public       Active   291d
kube-system       Active   291d
guru@k8s-master:~$
```

Partiton 생성과 라벨 customera, customer가 생성된 것을 확인할 수 있다.

📝 default namespace에 다음과 같은 pod를 생성하세요

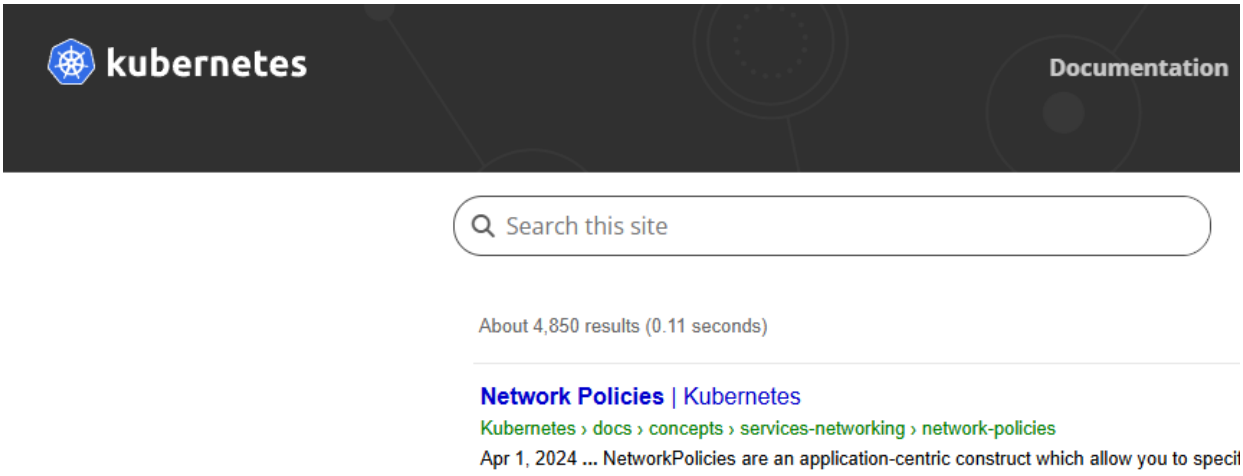
✔ poc 생성

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

✔ pod 확인

```
guru@k8s-master:~$ kubectl get po poc
NAME   READY   STATUS    RESTARTS   AGE
poc    0/1     Pending   0          5s
```

✔ Kubernetes.io/docs 내 Network Policy 검색



✔ Network Policy Recource 검색

service/networking/networkpolicy.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

80포트로 연결할 수 있도록 default namespace에 'allow-web-from-customera'라는 network Policy를 설정하세요. 보안 정책 상 다른 namespace의 접근은 제한합니다.

✔ vi netpol.yaml 생성 및 수정

수정 전

1 k8s-master | 2 k8s-master

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

수정 후

1 k8s-master | 2 k8s-master

```yaml
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
```

✔ yaml 적용

```
guru@k8s-master:~$ kubectl apply -f netpol.yaml
networkpolicy.networking.k8s.io/allow-web-from-customera unchanged
```

✔ pod 확인

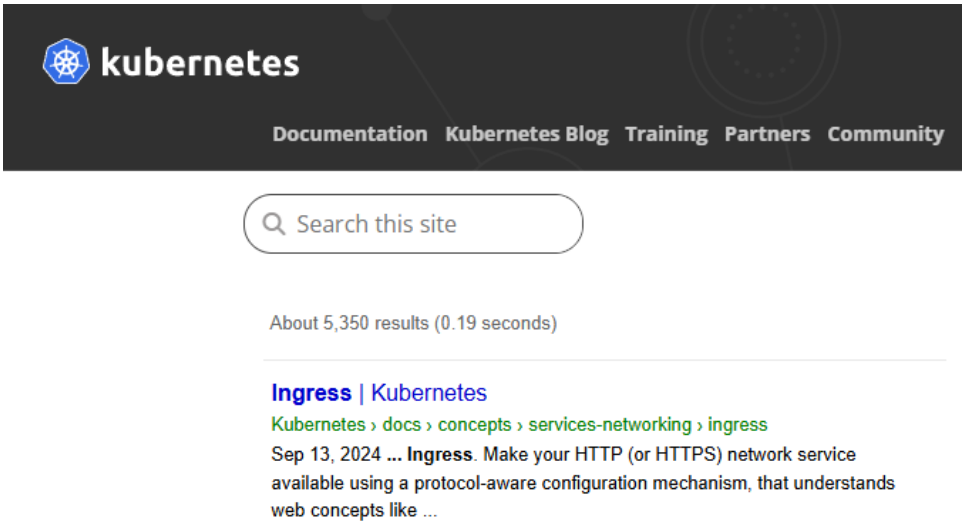
```
guru@k8s-master:~$ kubectl get netpol
NAME                      POD-SELECTOR   AGE
allow-web-from-customera  app=poc        47h
```

15. Ingress

1. Create a new nginx Ingress resource as follows
   - Name : ping
   - Namespace : ing-internal
   - Exposing service hi on path /hi using service port 5678

Create a new nginx Ingress resource as follows

✔ Kubenates 내 ingress 검색



The Ingress resource

A minimal Ingress resource example:

service/networking/minimal-ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

✔ 네임스페이스 생성

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

✔ 네임스페이스 확인

```
guru@k8s-master:~$ kubectl get ns ing-internal
NAME           STATUS   AGE
ing-internal   Active   29s
```

✔ vi ingress.yaml

```yaml
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
```

- **Name** : ping
- **Namespace** : ing-internal
- **Exposing service hi on path /hi using service port 5678**
  - Service name : hi
  - Path : /hi
  - port : 5679

apply -f 적용 내용포함 요망

✔ Ingress 확인

```
guru@k8s-master:~$ kubectl get Ingress -n ing-internal
NAME   CLASS           HOSTS   ADDRESS   PORTS   AGE
ping   nginx-example   *                 80      5m18s
```

16. Service and DNS Lookup

1. image nginx를 사용하는 resolver pod를 생성하고 resolver-service라는 service를 구성합니다.
2. 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.

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

✔ Resolver pod를 생성

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

✔ pod 확인

```
guru@k8s-master:~$ kubectl get po
NAME       READY   STATUS    RESTARTS   AGE
resolver   1/1     Running   0          60s
```

✔ 서비스 생성 kubectl expose pod resolver --name=resolver-service --port=80

```
guru@k8s-master:~$ kubectl expose pod resolver --name=resolver-service --port=80
service/resolver-service exposed
```

✔ 서비스 확인 kubectl get svc resolver-service

```
guru@k8s-master:~$ kubectl get svc resolver-service
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
resolver-service   ClusterIP   10.100.54.140   <none>        80/TCP    84s
```

클러스터 내에서 service와 pod 이름을 조회할 수 있는지 테스트합니다.

✔ Run test-nslookup

```
guru@k8s-master:~$ sudo -i
root@k8s-master:~# mkdir -p /var/CKA2023
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10.100.54.140
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.100.54.140
Address 1: 10.100.54.140 resolver-service.default.svc.cluster.local
pod "test-nslookup" deleted
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10.100.54.140 > /var/CKA2023/nginx.svc
```

root 전환 및 mkdir -p

```
guru@k8s-master:~$ sudo -i
root@k8s-master:~# mkdir -p /var/CKA2023
```

run test-nslookup 10.100.54.140

```
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10.100.54.140
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.100.54.140
Address 1: 10.100.54.140 resolver-service.default.svc.cluster.local
pod "test-nslookup" deleted
```

run test-nslookup 10.100.54.140 –image=busybox:1.28 -it --rm --restart=Never – resolver 10.100.54.140 > nginx.svc

```
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10.100.54.140 > nginx.svc
root@k8s-master:~#
```

cat /var/cka2023/nginx.svc

```
root@k8s-master:~# cat /var/CKA2023/nginx.svc
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.100.54.140
Address 1: 10.100.54.140 resolver-service.default.svc.cluster.local
pod "test-nslookup" deleted
```

get po resolver -o wide

```
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          3m54s   10.46.0.2   k8s-worker1   <none>           <none>
```

kube run test-nslookup –image=busybox:1.28 -it --rm --restart=Never – nslookp 10-46-0-2 default.pod.cluster.local

kube run test-nslookup –image=busybox:1.28 -it --rm --restart=Never – nslookp 10-46-0-2 default.pod.cluster.local > /var/CKA2023/nginx.pod

```
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10-46-0-2.default.pod.cluster.local
If you don't see a command prompt, try pressing enter.
warning: couldn't attach to pod/test-nslookup, falling back to streaming logs: Internal error occurred: error attaching to container: failed to load task: no runnin
c5bcb71730542f00b8e5c4f595a45f8d43f8c not found: not found
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:~# kubectl run test-nslookup --image=busybox:1.28 -it --rm --restart=Never -- nslookup 10-46-0-2.default.pod.cluster.local > /var/CKA2023/nginx.pod
root@k8s-master:~#
```

catl /var/CKA2023/nginx.pod

```
root@k8s-master:~# cat /var/CKA2023/nginx.pod
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
```

Resoure 관리

17. emptyDir Volume

1. 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

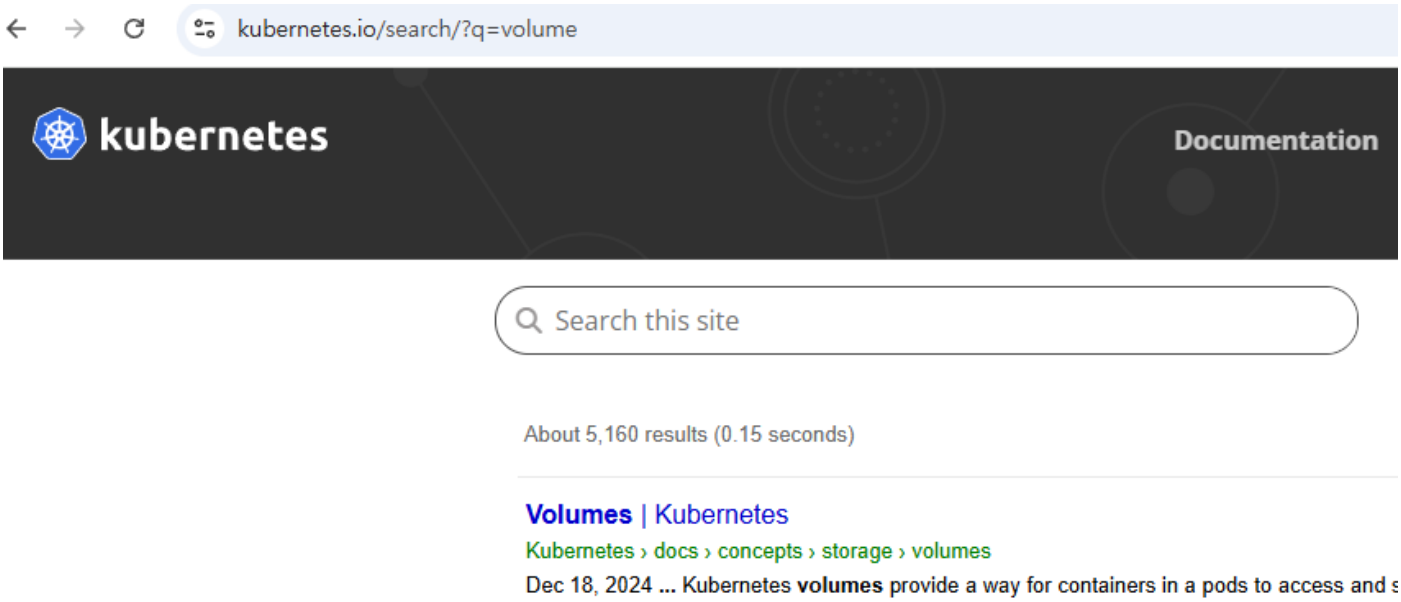
   Log container:

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

~~2. emptyDir 볼륨을 통한 데이터 공유~~

📝 **다음 조건에 맞춰서 nginx 웹서버 pod가 생성한 로그파일을 받아서 STDOUT으로 출력하는 busybox 컨테이너를 운영하시오.**

쿠버네티스 검색(emptyDir)



volume 내 emptyDir 검색

emptyDir configuration example

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: registry.k8s.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir:
      sizeLimit: 500Mi
```

weblog.yaml 확인 및 생성

```
guru@k8s-master:~$ kubectl run weblog --image=nginx:1.17 --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: weblog
  name: weblog
spec:
  containers:
  - image: nginx:1.17
    name: weblog
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
guru@k8s-master:~$ kubectl run weblog --image=nginx:1.17 --dry-run=client -o yaml > weblog.yaml
```

vi weblog.yaml 수정

```
1 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
    arg: [/bin/sh, -c, "tail -n+1 -f /data/access.log"]
    volumeMounts:
    - mountPath: /data
      name: weblog
      readOnly: true
  volumes:
  - name: weblog
    emptyDir: {}
```

yaml 적용

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

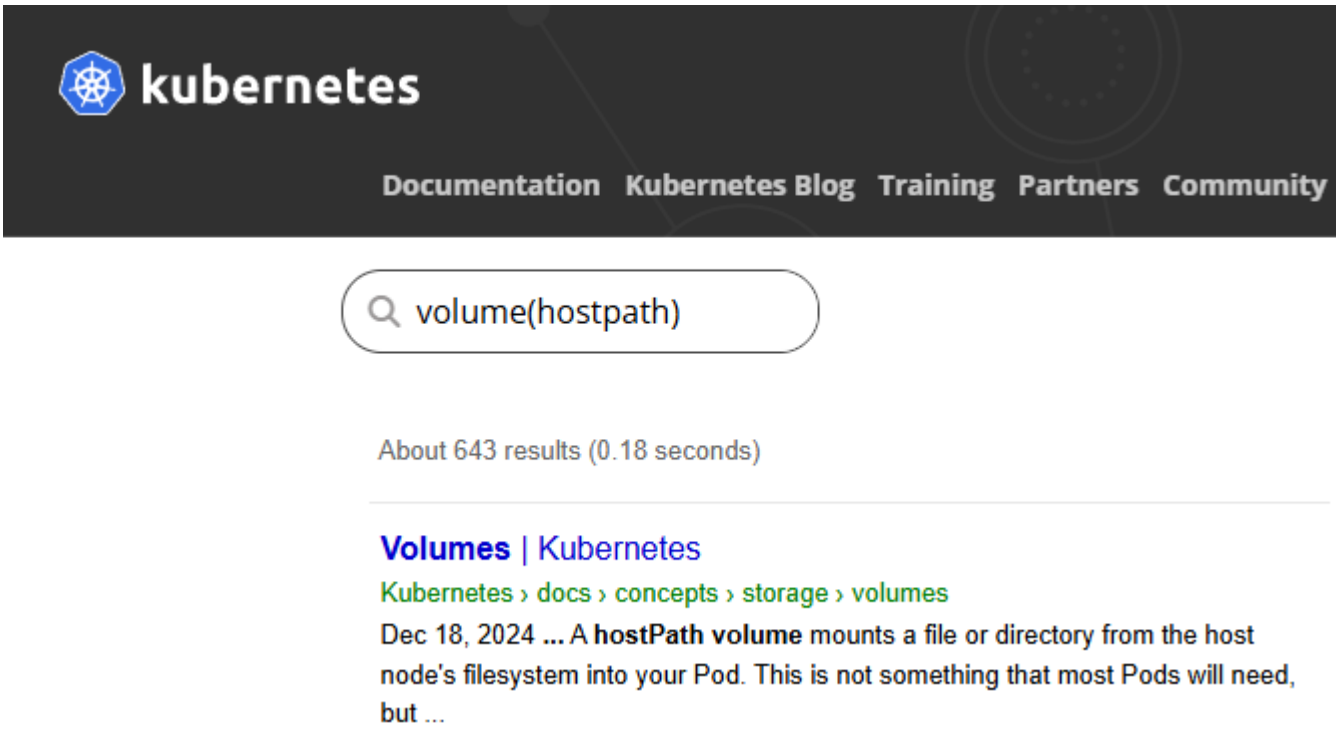
pod 확인

```
guru@k8s-master:~$ kubectl get po weblog
NAME     READY   STATUS    RESTARTS   AGE
weblog   0/2     Pending   0          2m45s
```

18. 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.
3. Worker node의 도커 컨테이너 디렉토리 : /var/lib/docker/containers 동일 디렉토리로 pod에 마운트 하시오.
4. Worker node의 /var/log 디렉토리를 fluentd Pod에 동일이름의 디렉토리 마운트 하시오.

kubenetis/docs 내 volume(hostpath) 검색



volume 내 hostpath 검색

hostPath configuration example

Linux node | Windows node

```yaml
---
# This manifest mounts /data/foo on the host as /foo inside the
# single container that runs within the hostpath-example-linux Pod.
#
# 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 exists
    hostPath:
      path: /data/foo # directory location on host
      type: Directory # this field is optional
```

/data/cka/fluentd.yaml 이동 및 vi 생성

```
guru@k8s-master:~$ cd /data/cka
guru@k8s-master:/data/cka$ vi fluentd.yaml
```

vi fluentd.yaml 코드 붙여넣기 및 수정

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

yaml 설정

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

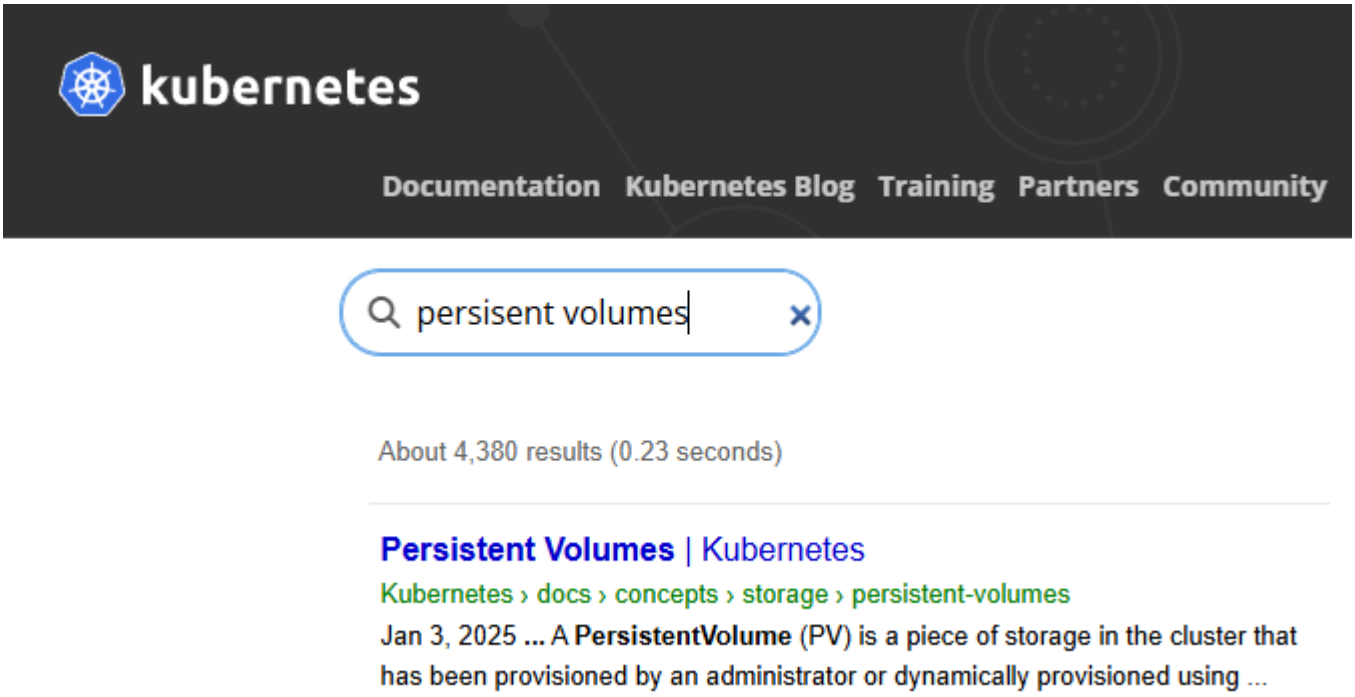
pod 확인

```
guru@k8s-master:/data/cka$ kubectl get po fluentd
NAME      READY   STATUS    RESTARTS   AGE
fluentd   0/1     Pending   0          9s
```

19. Persistent Volume

1. pv001라는 이름으로 size 1Gi, access mode ReadWriteMany를 사용하여 persistent volume을 생성합니다.
2. volume type은 hostPath이고 위치는 /tmp/app-config입니다.

쿠버네티스 내 persistent volume 또는 pv 검색



persistent volume 내 kind: 검색

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.

```yaml
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
```

vi pv.yaml 생성 후 코드 붙여넣기 및 수정

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

yaml 적용

```
guru@k8s-master:~$ kubectl apply -f pv.yaml
persistentvolume/pv001 created
```

pv 확인

```
guru@k8s-master:~$ kubectl get pv pv001
NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv001   1Gi        RWX            Retain           Available                                   31s
```

describe pv 확인

```
guru@k8s-master:~$ kubectl describe pv pv001
Name:            pv001
Labels:          <none>
Annotations:     <none>
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:
Status:          Available
Claim:
Reclaim Policy:  Retain
Access Modes:    RWX
VolumeMode:      Filesystem
Capacity:        1Gi
Node Affinity:   <none>
Message:
Source:
    Type:          HostPath (bare host directory volume)
    Path:          /tmp/app-config
    HostPathType:
Events:            <none>
guru@k8s-master:~$
```