k8s 프로젝트

클라우드 시스템
요구사항 분석

Team Renew

권택, 박소정, 안웅렬, 윤승원, 황준서

1. ETCD 스냅샷 생성 및 복원

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

(1) 스냅샷 생성 및 저장

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

(2) 스냅샷 저장 확인

```
root@master:~# ls -l /data/
합 계  4008
drwxr-xr-x 2 root root    4096  1월  15 16:53 cka
-rw-r--r-- 1 root root 4096032  1월  16 13:54 etcd-snapshot.db
```

(3) 스냅샷 복원

```
root@master:~# ETCDCTL_API=3 etcdctl --endpoints=https://127.0.0.1:2379 --cacert=/etc/kube
rnetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pk
i/etcd/server.key snapshot restore /data/etcd-snapshot.db
2025-01-16 14:04:00.892579 I | mvcc: restore compact to 204892
2025-01-16 14:04:00.920159 I | etcdserver/membership: added member 8e9e05c52164694d [http:
//localhost:2380] to cluster cdf818194e3a8c32
```

2. 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.

(1) 업그레이드 버전 확인

```
guru@k8s-master:~$ sudo apt-cache madison kubeadm
   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
   kubeadm | 1.28.4-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.3-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.2-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.1-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
   kubeadm | 1.28.0-1.1 | https://pkgs.k8s.io/core:/stable:/v1.28/deb  Packages
```

(2) 업그레이드 버전 설치

```
guru@k8s-master:~$ sudo apt-get install -y kubeadm='1.28.8-1.1'
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
패키지 kubeadm는 이미 최신 버전입니다 (1.28.8-1.1).
0개 업그레이드, 0개 새로 설치, 0개 제거 및 78개 업그레이드 안 함.
guru@k8s-master:~$ sudo apt-mark hold kubeadm
kubeadm 패키지 고정으로 설정.
```

(3) 업그레이드 플랜 확인

```
guru@k8s-master:~$ sudo kubeadm upgrade plan
```

(4) 업그레이드 실행

```
guru@k8s-master:~$ sudo kubeadm upgrade apply v1.28.8-1.1
[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-syste
m get cm kubeadm-config -o yaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade/version] You have chosen to change the cluster version to "v1.28.8-1.1"
[upgrade/versions] Cluster version: v1.28.8
[upgrade/versions] kubeadm version: v1.28.8
[upgrade/version] FATAL: the --version argument is invalid due to these errors:
```

(5) 노드 드레인 적용

```
guru@k8s-master:~$ kubectl drain k8s-master --ignore-daemonsets
node/k8s-master cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-vcgrw, kube-syste
m/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
```

(6) Kubelet 및 Kubectl 업그레이드

```
root@k8s-master:~# sudo apt-get update && sudo apt-get install -y kubelet='1.28.8-1.1' kubectl='1.28.8-1.1'
기존:1 http://kr.archive.ubuntu.com/ubuntu focal InRelease
기존:2 http://kr.archive.ubuntu.com/ubuntu focal-updates InRelease
기존:3 http://kr.archive.ubuntu.com/ubuntu focal-backports InRelease
기존:4 https://download.docker.com/linux/ubuntu focal InRelease
기존:5 http://security.ubuntu.com/ubuntu focal-security InRelease
받기:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  InRelease [1,192 B]
오류:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  InRelease
  다음 서명이 올바르지 않습니다: EXPKEYSIG 234654DA9A296436 isv:kubernetes OBS Project <isv:kubernetes@build.opens
use.org>
내려받기 1,192 바이트, 소요시간 25초 (47 바이트/초)
패키지 목록을 읽는 중입니다... 완료
W: An error occurred during the signature verification. The repository is not updated and the previous index files
 will be used. GPG error: https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb
InRelease: 다음 서명이 올바르지 않습니다: EXPKEYSIG 234654DA9A296436 isv:kubernetes OBS Project <isv:kubernetes@bu
ild.opensuse.org>
W: https://pkgs.k8s.io/core:/stable:/v1.28/deb/InRelease 파일을 받는데 실패했습니다  다음 서명이 올바르지 않습니다
: EXPKEYSIG 234654DA9A296436 isv:kubernetes OBS Project <isv:kubernetes@build.opensuse.org>
W: Some index files failed to download. They have been ignored, or old ones used instead.
패키지 목록을 읽는 중입니다... 완료
의존성 트리를 만드는 중입니다
상태 정보를 읽는 중입니다... 완료
패키지 kubectl는 이미 최신 버전입니다 (1.28.8-1.1).
패키지 kubelet는 이미 최신 버전입니다 (1.28.8-1.1).
0개 업그레이드, 0개 새로 설치, 0개 제거 및 78개 업그레이드 안 함.
```

(7) 노드 차단 해제

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

(8) 노드 버전 확인

```
root@k8s-master:~# kubectl get nodes
NAME          STATUS                     ROLES           AGE    VERSION
k8s-master    Ready,SchedulingDisabled   control-plane   290d   v1.28.8
k8s-worker1   Ready                      <none>          290d   v1.28.8
k8s-worker2   Ready                      <none>          290d   v1.28.8
```

3. Service Account, Role, RoleBinding

애플리케이션 운영중 특정 namespace의 Pod들을 모니터할 수 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.

(1) Namespace 생성 및 확인

```
ubuntu@master:~$ kubectl create namespace api-access
namespace/api-access created
ubuntu@master:~$ kubectl get ns
NAME              STATUS   AGE
api-access        Active   4s
default           Active   7d23h
kube-node-lease   Active   7d23h
kube-public       Active   7d23h
kube-system       Active   7d23h
```

(2) Service Account 생성 및 확인

```
ubuntu@master:~$ kubectl create serviceaccount pod-viewer -n=api-access
serviceaccount/pod-viewer created
ubuntu@master:~$ kubectl get sa -n=api-access
NAME         SECRETS   AGE
default      0         80s
pod-viewer   0         12s
```

(3) Role 생성 및 확인

```
guru@k8s-master:~$ kubectl create role podreader-role -n api-access --resource=pod
--verb=watch,list,get
role.rbac.authorization.k8s.io/podreader-role created
guru@k8s-master:~$ kubectl get role -n api-access
NAME             CREATED AT
podreader-role   2025-01-20T06:50:05Z
```

(4) RoleBinding 생성 및 확인

```
guru@k8s-master:~$ kubectl create rolebinding podreader-rolebinding --serviceaccount
=api-accesss:pod-viewer --role=podreader-role -n api-access
rolebinding.rbac.authorization.k8s.io/podreader-rolebinding created
guru@k8s-master:~$ kubectl get rolebinding -n api-access
NAME                    ROLE                   AGE
podreader-rolebinding   Role/podreader-role   13s
```

```
ubuntu@master:~$ kubectl get rolebinding -n api-access
NAME                    ROLE                   AGE
podreader-rolebinding   Role/podreader-role   95s
```

4. Service Account, ClusterRole, ClusterRoleBinding

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

(1) ClusterRole 생성 및 확인

```
guru@k8s-master:~$ kubectl create clusterrole deployment-clusterrole --resource
=Deployment,StatefulSet,DaemonSet --verb=create
clusterrole.rbac.authorization.k8s.io/deployment-clusterrole created
guru@k8s-master:~$ kubectl get clusterrole deployment-clusterrole
NAME                     CREATED AT
deployment-clusterrole   2025-01-20T06:53:30Z
```

(2) Service Account 생성 및 확인

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

(3) ClusterRoleBinding 생성

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

(4) ClusterRoleBinding 확인

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

5. 노드 비우기

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

(1) 노드 확인

```
guru@k8s-master:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE    VERSION
k8s-master    Ready    control-plane   290d   v1.28.8
k8s-worker1   Ready    <none>          290d   v1.28.8
k8s-worker2   Ready    <none>          290d   v1.28.8
```

(2) 노드 스케줄링 불가능 설정

```
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-syste
m/weave-net-sbnc5
evicting pod kube-system/coredns-5dd5756b68-zm5c6
evicting pod default/kubernetes-simple-app-6554bd5d45-l8qhb
evicting pod devops/eshop-order-5df8688cb7-226wl
pod/kubernetes-simple-app-6554bd5d45-l8qhb evicted
pod/eshop-order-5df8688cb7-226wl evicted
pod/coredns-5dd5756b68-zm5c6 evicted
node/k8s-worker2 drained
```

(3) 노드 상태 확인

```
guru@k8s-master:~$ kubectl get nodes
NAME          STATUS                     ROLES           AGE    VERSION
k8s-master    Ready                      control-plane   290d   v1.28.8
k8s-worker1   Ready                      <none>          290d   v1.28.8
k8s-worker2   Ready,SchedulingDisabled   <none>          290d   v1.28.8
```

(4) 파드 Rescheduling

```
guru@k8s-master:~$ kubectl get po -o wide
NAME                                READY   STATUS             RESTARTS
AGE     IP          NODE          NOMINATED NODE   READINESS GATES
nginxsp-k8s-worker1                 1/1     Running            5 (59m ago)
4h14m   10.46.0.1   k8s-worker1   <none>           <none>
guru@k8s-master:~$ kubectl uncordon k8s-worker2
node/k8s-worker2 uncordoned
```

6. NodeSelector를 이용한 Pod 스케줄링

다음의 조건으로 pod를 생성하세요.

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

(1) eshop-store.yaml 파일 생성

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

(2) NodeSelector 추가

```
guru@k8s-master:~$ vi eshop-store.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: eshop-store
  name: eshop-store
spec:
  containers:
  - image: nginx
    name: eshop-store
  nodeSelector:
    disktype: ssd
```

(3) Pod 적용

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

(4) Pod 확인

```
guru@k8s-master:~$ kubectl get po eshop-store -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP          NODE          NOMINATED NODE   READINESS GATES
eshop-store   1/1     Running   0          61s   10.32.0.4   k8s-worker1   <none>           <none>
```

7. 환경 변수, 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“

(1) Namespace 생성

```
ubuntu@master:~$ kubectl create ns cka-exam
namespace/cka-exam created
ubuntu@master:~$ kubectl get ns
NAME              STATUS   AGE
api-access        Active   28m
cka-exam          Active   5s
```

(2) pod-01.yaml 파일 생성

```
ubuntu@console:~$ kubectl run pod-01.yaml -n cka-exam --image=busynox --env=CERT=
"CKA-cert" --dry-run=client -o yaml > pod-01.yaml
```

(3) 환경 변수, Command, Args 추가

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

(4) Pod 적용

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

(5) Pod 확인

```
ubuntu@master:~$ kubectl describe po -n cka-exam
Name:             pod-01
Namespace:        cka-exam
    Command:
      /bin/sh
    Args:
      -c
      while true; do echo $(CERT); sleep 10;done
    State:          Running
      Started:      Thu, 16 Jan 2025 15:31:56 +0900
    Ready:          True
    Restart Count:  0
    Environment:
      CERT:  CKA-cert
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from
```

8. 로그 확인 및 저장

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

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

(1) Pod 확인

```
guru@k8s-master:~$ kubectl get po
NAME                          READY   STATUS    RESTARTS   AGE
nginx-static-pod-k8s-worker1  1/1     Running   0          14m
```

(2) 로그 모니터링 및 저장

```
root@k8s-master:~# kubectl logs nginx-static-pod-k8s-worker1 > /opt/REPORT/2023/pod-
log
root@k8s-master:~# cat /opt/REPORT/2023/pod-log
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform c
onfiguration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.s
h
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/def
ault.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/d
efault.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 07:27:15 [notice] 1#1: using the "epoll" event method
2025/01/16 07:27:15 [notice] 1#1: nginx/1.27.3
2025/01/16 07:27:15 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/16 07:27:15 [notice] 1#1: OS: Linux 5.15.0-130-generic
2025/01/16 07:27:15 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/16 07:27:15 [notice] 1#1: start worker processes
2025/01/16 07:27:15 [notice] 1#1: start worker process 29
2025/01/16 07:27:15 [notice] 1#1: start worker process 30
```

9. Static Pod 생성

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

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

(1) YAML 파일 생성 (/etc/kubernetes/manifests 경로)

```
guru@k8s-worker1:~$ ls /etc/kubernetes/manifests/
nginx-static-pod.yaml
```

(2) 포트 설정 추가

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

(3) Static Pod 적용 및 작동 여부 확인

```
ubuntu@k8s-master:~$ kubectl get po -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP
  NODE          NOMINATED NODE   READINESS GATES
nginx-static-pod-k8s-worker1   1/1     Running   0          98s   10.44.0.1
  k8s-worker1   <none>           <none>
```

10. Multi-Container Pod 생성

4개의 컨테이너를 동작시키는 eshop-frontend Pod를 생성하시오. pod image: nginx, redis, memcached, consul

(1) eshop-frontend.yaml 파일 생성

```
ubuntu@k8s-master:~$ kubectl run eshop-frontend --image=nginx --dry-run=clie
nt -o yaml > eshop-frontend.yaml
ubuntu@k8s-master:~$ ls -l eshop*
-rw-rw-r-- 1 ubuntu ubuntu 259 Jan 20 16:04 eshop-frontend.yaml
```

(2) Multi-Container 설정 추가

```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
```

(3) Pod 적용

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

(4) 작동 여부 확인

- consul 버전 문제로 3/4개 동작 중

```
ubuntu@master:~$ kubectl get pod
NAME                        READY   STATUS             RESTARTS   AGE
eshop-frontend              3/4     ImagePullBackOff   0          54s
```

11. Deployment 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

(1) eshop-payment.yaml 파일 생성

```
ubuntu@k8s-master:~$ kubectl create deploy eshop-payment --image=nginx:1.16
--replicas=3 --dry-run=client -o yaml > eshop-payment.yaml
ubuntu@k8s-master:~$ ls -l eshop*
-rw-rw-r-- 1 ubuntu ubuntu 421 Jan 20 16:12 eshop-payment.yaml
```

(2) Label 추가

```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
```

(3) Deployment 적용 및 기록

```
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
```

(4) Rolling Update

```
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
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
```

(5) RollBack

```
ubuntu@master:~$ kubectl rollout undo deployment eshop-payment
deployment.apps/eshop-payment rolled back
ubuntu@master:~$ kubectl get po
NAME                              READY   STATUS             RESTARTS       AGE
eshop-frontend                    3/4     ImagePullBackOff   0              13m
eshop-payment-bfd69c669-2bscl     1/1     Running            0              29s
eshop-payment-bfd69c669-d4xgb     1/1     Running            0              28s
eshop-payment-bfd69c669-sf7bg     1/1     Running            0              26s
nginx-static-pod-worker1          1/1     Running            1 (134m ago)   12m
ubuntu@master:~$ kubectl describe po eshop-payment-bfd69c669-2bscl | grep -i nginx
  nginx:
    Image:          nginx:1.16
```

12. ClusterIP 서비스 생성

'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

(1) Namespace 생성 및 확인

```
ubuntu@master:~$ kubectl create ns devops
namespace/devops created
ubuntu@master:~$ kubectl get ns devops
NAME     STATUS   AGE
devops   Active   10s
```

(2) eshop-order.yaml 파일 생성

```
ubuntu@k8s-master:~$ kubectl create deploy eshop-order -n devops --image=ngi
nx --replicas=2 --dry-run=client -o yaml > eshop-order.yaml
ubuntu@k8s-master:~$ ls -l eshop*
-rw-rw-r-- 1 ubuntu ubuntu 428 Jan 20 16:14 eshop-order.yaml
```

(3) Label, Namespace 설정 추가

```yaml
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
```

(4) Deployment 파일 적용

```
ubuntu@master:~$ kubectl apply -f eshop-order.yaml
```

(5) Deployment 작동 여부 확인

```
ubuntu@master:~$ kubectl get deploy -n devops
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
eshop-order   2/2     2            2           102s
```

(6) Pod 작동 여부 확인

```
ubuntu@master:~$ kubectl get po -n devops | grep -i eshop-order
eshop-order-5df8688cb7-chcmt   1/1     Running   0          3m12s
eshop-order-5df8688cb7-jchdm   1/1     Running   0          3m11s
```

(7) Service 생성 및 확인

```
ubuntu@master:~$ kubectl expose deploy eshop-order -n devops --name=eshop-order-svc --port=8
0 --target-port=80
service/eshop-order-svc exposed
ubuntu@master:~$ kubectl get svc eshop-order-svc -n devops
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
eshop-order-svc   ClusterIP   10.110.33.236   <none>        80/TCP    25s
```

13. NodePort 서비스 생성

‘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.

(1) front-end.yaml 파일 생성

```
ubuntu@master:~$ kubectl create deploy front-end --image=nginx --replicas=2 --dry-run=client
 -o yaml > front-end.yaml
ubuntu@master:~$ ls
eshop-frontend.yaml  eshop-payment.yaml  pod-01.yaml  다운로드  바탕화면  사진  템플릿
eshop-order.yaml     front-end.yaml      공개         문서      비디오    음악
```

(2) Label 추가

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

(3) Deployment 적용 및 작동 여부 확인

```
deployment.apps/front-end created
ubuntu@master:~$ kubectl get deploy front-end
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
front-end   2/2     2            2           11s
```

(4) Service 수정

```
ubuntu@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
ubuntu@master:~$ vi front-end-nodesvc.yaml
```

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    run: nginx
  name: front-end-nodesvc
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
    nodePort: 30200
  selector:
    run: nginx
  type: NodePort
```

(5) Service 적용 및 작동 여부 확인

```
ubuntu@master:~$ kubectl apply -f front-end-nodesvc.yaml
service/front-end-nodesvc created
ubuntu@master:~$ kubectl get svc front-end-nodesvc
NAME               TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
front-end-nodesvc   NodePort   10.111.94.237   <none>        80:30200/TCP   15s
```

14. 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.

(1) Namespace 생성 및 확인

```
ubuntu@master:~$ kubectl create ns customera
namespace/customera created
ubuntu@master:~$ kubectl create ns customerb
namespace/customerb created
ubuntu@master:~$ kubectl get ns customera customerb
NAME        STATUS   AGE
customera   Active   14s
customerb   Active   12s
```

(2) Namespace 라벨링

```
ubuntu@master:~$ kubectl label ns customera partition=customera
namespace/customera labeled
ubuntu@master:~$ kubectl label ns customerb partition=customerb
namespace/customerb labeled
ubuntu@master:~$
ubuntu@master:~$ kubectl get ns -l partition
NAME        STATUS   AGE
customera   Active   87s
customerb   Active   85s
```

(3) Pod 생성 및 확인

```
ubuntu@master:~$ kubectl run poc --image=nginx --port=80 --labels=app=poc
pod/poc created
ubuntu@master:~$ kubectl get po poc
NAME   READY   STATUS    RESTARTS   AGE
poc    1/1     Running   0          35s
```

(4) Network Policy 생성

```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
```

(5) Network Policy 적용 및 확인

```
ubuntu@master:~$ kubectl apply -f netpol.yaml
networkpolicy.networking.k8s.io/allow-web-from-customera created
ubuntu@master:~$ kubectl get netpol
NAME                        POD-SELECTOR   AGE
allow-web-from-customera    app=poc        8s
```

```
guru@k8s-master:~$ kubectl get netpol
NAME                        POD-SELECTOR   AGE
allow-web-from-customera    app=poc        2d20h
```

15. Ingress

Create a new nginx Ingress resource as follows:

- Name: ping
- Namespace: ing-internal
- Exposing service hi on path /hi using service port 5678

(1) Namespace 생성

```
guru@k8s-master:~$ kubectl create ns ing-internal
namespace/ing-internal created
guru@k8s-master:~$ kubectl get ns ing-internal
NAME           STATUS   AGE
ing-internal   Active   14s
```

(2) Yaml 파일 수정

```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
```

(3) Ingress 적용 및 확인

```
guru@k8s-master:~$ kubectl apply -f ingress.yaml
ingress.networking.k8s.io/ping created
guru@k8s-master:~$ kubectl get ingress -n ing-internal
NAME   CLASS           HOSTS   ADDRESS   PORTS   AGE
ping   nginx-example   *                 80      24s
```

16. 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.

(1) Pod 생성 및 확인

```
guru@k8s-master:~$ kubectl run resolver --image=nginx --port=80
pod/resolver created
guru@k8s-master:~$ kubectl get pod resolver
NAME       READY   STATUS    RESTARTS   AGE
resolver   1/1     Running   0          2m1s
```

(2) Service 생성 및 확인

```
guru@k8s-master:~$ kubectl expose pod resolver --name=resolver-service --port=80
service/resolver-service exposed
guru@k8s-master:~$ kubectl get svc resolver-service
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
resolver-service   ClusterIP   10.107.210.76   <none>        80/TCP    36s
```

(3) DNS 조회 확인

```
guru@k8s-master:~$ kubectl run test-nslookup --image=busybox:1.28 -it --restart=Never --rm -
- nslookup 10.107.210.76
If you don't see a command prompt, try pressing enter.
warning: couldn't attach to pod/test-nslookup, falling back to streaming logs: unable to upg
rade connection: container test-nslookup not found in pod test-nslookup_default
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      10.107.210.76
Address 1: 10.107.210.76 resolver-service.default.svc.cluster.local
pod "test-nslookup" deleted
```

(4) DNS 조회 결과 저장

```
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --restart=Never --rm -
- nslookup 10.107.210.76 > /var/CKA2023/nginx.svc
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 erro
r occurred: error attaching to container: container is in CONTAINER_EXITED state
root@k8s-master:~# ls /var/CKA2023
nginx.svc
```

```
root@k8s-master:~# kubectl run test-nslookup --image=busybox:1.28 -it --restart=Never --rm -
- nslookup 10-107-210-76.default.pod.cluster.local > /var/CKA2023/nginx.pod
root@k8s-master:~# ls /var/CKA2023
nginx.pod  nginx.svc
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:      10-107-210-76.default.pod.cluster.local
Address 1: 10.107.210.76 resolver-service.default.svc.cluster.local
pod "test-nslookup" deleted
```

17. 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

(1) Pod 생성

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

(2) Volume Mounts, Volumes 설정 추가

```yaml
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
      readOnly: true
      name: weblog
  volumes:
  - name: weblog
    emptyDir: {}
```

(3) Pod 적용

```
guru@k8s-master:~$ kubectl apply -f weblog.yaml
pod/weblog created
guru@k8s-master:~$ kubectl get pod weblog
NAME     READY   STATUS    RESTARTS   AGE
weblog   2/2     Running   0          35s
```

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.
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.

(1) Pod 생성

```
ubuntu@k8s-master:~$ kubectl run fleuntd --image=fluentd --port=80 --dry-run
=client -o yaml > fluentd.yaml
ubuntu@k8s-master:~$ vi fluentd.yaml
```

(2) Pod 수정

```yaml
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
```

(3) Pod 적용

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

(4) Pod 확인

```
root@k8s-master:~# kubectl describe po fluentd
```

```
    Mounts:
      /var/lib/docker/containers from containersdir (rw)
      /var/log from logdir (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-v5pqx (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:
```

19. Persistent Volume

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

volume type은 hostPath이고 위치는 /tmp/app-config입니다.

(1) Pod 생성 및 Volume 설정

```
root@k8s-master:~# vi pv001.yaml
```

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

(2) Pod 적용 및 확인

```
root@k8s-master:~# kubectl apply -f pv001.yaml
persistentvolume/pv001 created
```

```
ubuntu@k8s-master:~$ kubectl get pv pv001
NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
pv001   1Gi        RWX            Retain           Available
                        54s
```