챕터1 CKA 복습과 포드 컨테이너 디자인

Created 2021년 11월 6일 오후 2:48

Tags 비어 있음

▼ 목차

쿠버네티스 클러스터 재구성하기

kubeadm 설치

https://kubernetes.io/ko/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

```
cat <<EOF > kube_install.sh sudo apt-get update sudo apt-get install -y apt-
transport-https ca-certificates curl sudo curl -fsSLo
/usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg echo "deb [signed-
by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list EOF bash kube_install.sh
```

도커 cgroup 이름 변경하기

https://stackoverflow.com/questions/43794169/docker-change-cgroup-driver-to-systemd

```
cat <<EOF > /etc/docker/daemon.json { "exec-opts":
["native.cgroupdriver=systemd"] } EOF service docker restart
```

Weavenet 설치

https://www.weave.works/docs/net/latest/kubernetes/kube-addon/

워드프레스 설치

예시: WordPress와 MySQL을 퍼시스턴트 볼륨에 배포하기

https://kubernetes.io/ko/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/

워드프레스 배포 스크립트

```
mkdir wordpress cd wordpress cat <<EOF >./kustomization.yaml secretGenerator:
- name: mysql-pass literals: - password=test1234 resources: - mysql-deploymen
t.yaml - wordpress-deployment.yaml EOF cat <<EOF >./mysql-deployment.yaml apiV
ersion: v1 kind: Service metadata: name: wordpress-mysql labels: app: wordpres
s spec: ports: - port: 3306 selector: app: wordpress tier: mysql clusterIP: No
ne --- apiVersion: apps/v1 kind: Deployment metadata: name: wordpress-mysql la
bels: app: wordpress spec: selector: matchLabels: app: wordpress tier: mysql s
trategy: type: Recreate template: metadata: labels: app: wordpress tier: mysql
spec: containers: - image: mysql:5.6 name: mysql env: - name: MYSQL_ROOT_PASSW
ORD valueFrom: secretKeyRef: name: mysql-pass key: password ports: - container
Port: 3306 name: mysql EOF cat <<EOF >./wordpress-deployment.yaml apiVersion:
v1 kind: Service metadata: name: wordpress labels: app: wordpress spec: ports:
- port: 80 selector: app: wordpress tier: frontend type: LoadBalancer --- apiV
ersion: apps/v1 kind: Deployment metadata: name: wordpress labels: app: wordpr
ess spec: selector: matchLabels: app: wordpress tier: frontend strategy: type:
Recreate template: metadata: labels: app: wordpress tier: frontend spec: conta
iners: - image: wordpress:4.8-apache name: wordpress env: - name: WORDPRESS_DB
_HOST value: wordpress-mysql - name: WORDPRESS_DB_PASSWORD valueFrom: secretKe
yRef: name: mysql-pass key: password ports: - containerPort: 80 name: wordpres
s EOF kubectl apply -k ./
```

쿠버네티스 치트시트

https://kubernetes.io/docs/reference/kubectl/cheatsheet/

배시 자동 완성

```
source <(kubectl completion bash) # setup autocomplete in bash into the
current shell, bash-completion package should be installed first. echo "source
<(kubectl completion bash)" >> ~/.bashrc # add autocomplete permanently to
your bash shell. alias k=kubectl complete -F __start_kubectl k
```

멀티컨테이너

연습문제 - 하나의 파드에서 nginx와 redis 이미지를 모두 실행하는 yaml을 만들고 실행하라.

```
cat <<EOF | kubectl apply -f - apiVersion: v1 kind: Pod metadata: name: two-
containers spec: containers: - name: nginx-container image: nginx:1.21.3 -
name: redis-container image: redis:6.2.6 EOF
```

두 개 이상의 컨테이너가 올라와 있는 환경에서는 로그를 확인하거나 exec 명령을 통해 특정 명령을 실행할 컨테이너를 선택해야 한다.

```
# kubectl logs two-containers nginx-container # kubectl exec -it two-
containers -c nginx-container -- bash
```

라이브네스 프로브 실습

라이브네스 프로브 테스트 코드 실행

```
kubectl apply -f https://k8s.io/examples/pods/probe/exec-liveness.yaml
```

디스크라이브 명령으로 Event를 확인

```
# kubectl describe pod liveness-exec Events: Type Reason Age From Message ----
------- ---- ---- ------- Normal Scheduled 45s default-scheduler Successfully
assigned default/liveness-exec to gasbugs-02 Normal Pulling 44s kubelet
Pulling image "k8s.gcr.io/busybox" Normal Pulled 42s kubelet Successfully
pulled image "k8s.gcr.io/busybox" in 2.407193872s Normal Created 41s kubelet
Created container liveness Normal Started 41s kubelet Started container
liveness Warning Unhealthy 0s (x3 over 10s) kubelet Liveness probe failed:
cat: can't open '/tmp/healthy': No such file or directory Normal Killing 0s
kubelet Container liveness failed liveness probe, will be restarted
```

HTTP 요청을 활용한 라이브네스 프로브

```
kubectl apply -f https://k8s.io/examples/pods/probe/http-liveness.yaml
```

```
# kubectl describe pod liveness-http Events: Type Reason Age From Message ----
------- ---- ---- -------- Normal Scheduled 22s default-scheduler Successfully
assigned default/liveness-http to gasbugs-03 Normal Pulling 21s kubelet
Pulling image "k8s.gcr.io/liveness" Normal Pulled 18s kubelet Successfully
pulled image "k8s.gcr.io/liveness" in 2.462709392s Normal Created 18s kubelet
Created container liveness Normal Started 17s kubelet Started container
liveness Warning Unhealthy 1s (x3 over 7s) kubelet Liveness probe failed: HTTP
probe failed with statuscode: 500 Normal Killing 1s kubelet Container liveness
failed liveness probe, will be restarted
```

사이드카 컨테이너

사이드카 컨테이너 예제

```
# nginx-sidecar.yaml apiVersion: v1 kind: Pod metadata: name: nginx-sidecar
spec: containers: - name: nginx image: nginx ports: - containerPort: 80
volumeMounts: - name: varlognginx mountPath: /var/log/nginx - name: sidecar-
access image: busybox args: [/bin/sh, -c, 'tail -n+1 -f
/var/log/nginx/access.log'] volumeMounts: - name: varlognginx mountPath:
/var/log/nginx - name: sidecar-error image: busybox args: [/bin/sh, -c, 'tail
-n+1 -f /var/log/nginx/error.log'] volumeMounts: - name: varlognginx
mountPath: /var/log/nginx volumes: - name: varlognginx emptyDir: {}
```

애플리케이션 배포

```
vim nginx-sidecar.yaml kubectl apply -f nginx-sidecar.yaml
```

액세스 로그를 남기도록 유도

```
kubectl exec nginx-sidecar -c nginx -- curl 127.0.0.1 -s
```

액세스 로그는 sidecar-access 에서 보도록 한다.

```
# kubectl logs nginx-sidecar sidecar-access 127.0.0.1 - -
[06/Nov/2021:07:56:12 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/7.64.0" "-"
127.0.0.1 - - [06/Nov/2021:07:56:13 +0000] "GET / HTTP/1.1" 200 615 "-"
"curl/7.64.0" "-" 127.0.0.1 - - [06/Nov/2021:07:56:15 +0000] "GET / HTTP/1.1"
200 615 "-" "curl/7.64.0" "-"
```

어댑터 컨테이너

어댑터 컨테이너 예제

k8s-adaptor-container-pattern
bbachi

어댑터 컨테이너 배포

```
kubectl apply -f https://raw.githubusercontent.com/bbachi/k8s-adaptor-container-
pattern/master/pod.yml
```

어댑터 컨테이너 요청 및 실행

```
# kubectl port-forward adapter-container-demo 8080:3080 # curl localhost:8080/
logs [{"time":"Sat Nov 6 08:20:13 UTC 2021","message":"This is log"},{"time":
"Sat Nov 6 08:20:18 UTC 2021","message":"This is log"},{"time":"Sat Nov 6 08:2
0:23 UTC 2021","message":"This is log"},{"time":"Sat Nov 6 08:20:28 UTC 2021",
"message":"This is log"},{"time":"Sat Nov 6 08:20:33 UTC 2021","message":"This
is log"},{"time":"Sat Nov 6 08:20:38 UTC 2021","message":"This is log"},{"tim
e":"Sat Nov 6 08:20:43 UTC 2021","message":"This is log"},{"time":"Sat Nov 6 0
8:20:48 UTC 2021","message":"This is log"},{"time":"Sat Nov 6 08:20:53 UTC 202
1","message":"This is log"},{"time":"Sat Nov 6 08:20:58 UTC 2021","message":"T
his is log"},{"time":"Sat Nov 6 08:21:03 UTC 2021","message":"This is log"},{
"time":"Sat Nov 6 08:21:08 UTC 2021","message":"This is log"},{"time":"Sat Nov
6 08:21:13 UTC 2021","message":"This is log"},{"time":"Sat Nov 6 08:21:18 UTC
2021","message":"This is log"},{"time":"Sat Nov 6 08:21:23 UTC 2021","message"
:"This is log"},{"time":"Sat Nov 6 08:21:28 UTC 2021","message":"This is log"}
,{"time":"Sat Nov 6 08:21:33 UTC 2021","message":"This is log"},{"time":"Sat N
ov 6 08:21:38 UTC 2021","message":"This is log"},{"time":"Sat Nov 6 08:21:43 U
TC 2021","message":"This is log"},{"time":"Sat Nov 6 08:21:48 UTC 2021","messa
ge":"This is log"},{"time":"Sat Nov 6 08:21:53 UTC 2021","message":"This is lo
g"}]
```

앰배서더 컨테이너

github 프로젝트 예제

k8s-ambassador-container-pattern.git
bbachi

다음 명령을 사용해 클러스터에 배포

```
kubectl apply -f https://raw.githubusercontent.com/bbachi/k8s-ambassador-conta
iner-pattern/master/pod.yml
```

앰배서더 컨테이너로 요청

```
$ kubectl exec -it ambassador-container-demo -c ambassador-container -- curl l
ocalhost:9000 <현재는 403으로 정상적으로 통신 불가>
```

로그에서 통신 정보 확인

```
kubectl logs ambassador-container-demo main-container
```

초기화 컨테이너

참조 문서

https://kubernetes.io/ko/docs/concepts/workloads/pods/init-containers/

vim myapp.yaml 작성

```
apiVersion: v1 kind: Pod metadata: name: myapp-pod labels: app: myapp spec: co
ntainers: - name: myapp-container image: busybox:1.28 command: ['sh', '-c', 'e
cho The app is running! && sleep 3600'] initContainers: - name: init-myservice
image: busybox:1.28 command: ['sh', '-c', "until nslookup myservice.$(cat /va
r/run/secrets/kubernetes.io/serviceaccount/namespace).svc.cluster.local; do ec
ho waiting for myservice; sleep 2; done"] - name: init-mydb image: busybox:1.2
8 command: ['sh', '-c', "until nslookup mydb.$(cat /var/run/secrets/kubernete
s.io/serviceaccount/namespace).svc.cluster.local; do echo waiting for mydb; sl
eep 2; done"]
```

vim services.yaml 작성

```
--- apiVersion: v1 kind: Service metadata: name: myservice spec: ports: - prot
ocol: TCP port: 80 targetPort: 9376 --- apiVersion: v1 kind: Service metadata:
name: mydb spec: ports: - protocol: TCP port: 80 targetPort: 9377
```

Job과 CronJob

job 예제 배포

```
cat <<EOF | kubectl apply -f - apiVersion: batch/v1 kind: Job metadata: name:
pi spec: template: spec: containers: - name: pi image: perl command: ["perl",
"-Mbignum=bpi", "-wle", "print bpi(2000)"] restartPolicy: Never backoffLimit:
4 EOF
```

job 병렬 실행 예제

```
cat <<EOF | kubectl apply -f - apiVersion: batch/v1 kind: Job metadata: name:
pi-parallelism spec: completions: 5 # 목표 완료 파드 개수 parallelism: 2 # 동시 실행
가능 파드 개수 template: spec: containers: - name: pi image: perl command:
["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"] restartPolicy: Never
backoffLimit: 4 EOF
```

CronJob 예제 실행

```
cat <<EOF | kubectl apply -f - # cronjob-1.yaml apiVersion: batch/v1 kind: Cro
nJob metadata: name: hello-1 spec: concurrencyPolicy: Allow schedule: "*/1 * *
* *" jobTemplate: spec: template: spec: containers: - name: hello image: busyb
ox args: - /bin/sh - -c - date; echo Hello from the Kubernetes cluster restart
Policy: OnFailure EOF
```

리플레이스 정책을 적용한 CronJob 예제

```
cat <<EOF | kubectl apply -f - # cronjob-1.yaml apiVersion: batch/v1 kind:
CronJob metadata: name: hello-2 spec: concurrencyPolicy: Replace schedule:
"*/1 * * * *" jobTemplate: spec: template: spec: containers: - name: hello
image: busybox args: - /bin/sh - -c - date; echo Hello from the Kubernetes
cluster; sleep 100; restartPolicy: OnFailure EOF
```

시스템 리소스 요구사항과 제한 설정

컨테이너 별 리소스 제한 설정

https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

```
cat <<EOF | kubectl apply -f - apiVersion: apps/v1 kind: Deployment metadata:
name: nginx spec: replicas: 3 selector: matchLabels: app: nginx template:
metadata: labels: app: nginx spec: containers: - name: nginx image: nginx
ports: - containerPort: 80 resources: requests: memory: "200Mi" cpu: "1m"
limits: memory: "400Mi" cpu: "2m" EOF
```

노드에 할당되어있는 파드가 얼마나 많은 리소스를 예약했는지 확인할 수 있다.

```
# kubectl describe nodes gasbugs-02 <중략> Non-terminated Pods: (3 in total)
Namespace Name CPU Requests CPU Limits Memory Requests Memory Limits Age -----
---- ---- ------------ ---------- --------------- -------------- --- default
nginx-7fdbdd879d-jssw9 1m (0%) 2m (0%) 200Mi (5%) 400Mi (10%) 54s kube-system
kube-proxy-dcqq4 0 (0%) 0 (0%) 0 (0%) 0 (0%) 10h kube-system weave-net-qvmgw
100m (5%) 0 (0%) 200Mi (5%) 0 (0%) 10h Allocated resources: (Total limits may
be over 100 percent, i.e., overcommitted.) Resource Requests Limits -------- -
------- ------ cpu 101m (5%) 2m (0%) memory 400Mi (10%) 400Mi (10%) ephemeral-
storage 0 (0%) 0 (0%) hugepages-1Gi 0 (0%) 0 (0%) hugepages-2Mi 0 (0%) 0 (0%)
Events: <none>
```

limit range 설정하기

https://kubernetes.io/ko/docs/concepts/policy/limit-range/

```
# vim cpu-mem-min-max-default-lr.yaml apiVersion: v1 kind: LimitRange
metadata: name: cpu-mem-min-max-default-lr spec: limits: - max: cpu: "800m"
memory: "1Gi" min: cpu: "100m" memory: "99Mi" default: # default Limit cpu:
700m memory: 900Mi defaultRequest: cpu: 110m memory: 111Mi type: Container
```

파드를 실행하는 간단한 명령어

```
kubectl run nginx-lr --image=nginx
```

nginx-lr에 lr 정책에 따라 리소스 제한이 자동으로 default 값이 입력됐는지 확인한다.

```
kubectl get pod nginx-lr -o yaml # 출력 spec: containers: - image: nginx
imagePullPolicy: Always name: nginx-lr resources: limits: cpu: 700m memory:
900Mi requests: cpu: 110m memory: 111Mi
```