7 16 2일차

쿠버네티스 master plane node ip가 변하는 문제 발생

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQW
    server: https://192.168.128.2:6443
  name: kind-cwave-cluster
contexts:
- context:
    cluster: kind-cwave-cluster
    user: kind-cwave-cluster
  name: kind-cwave-cluster
current-context: kind-cwave-cluster
kind: Config
preferences: {}
users:
- name: kind-cwave-cluster
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhHZ
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFb3dJQkFBS0NBU
```

config 파일 ip 수정해서 해결했다.

껐다 키면서 ip가 재할당된 것이 원인

docker inspect kind로 ip 확인 후 config 수정

라벨 실습

라벨 부여

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl get po
NAME        READY   STATUS    RESTARTS      AGE
goapp-pod   1/1     Running   1 (34m ago)   15h
nginx-pod   1/1     Running   1 (34m ago)   16h
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl label node cwave-cluster-worker memsize=high
node/cwave-cluster-worker labeled
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl get po -o wide
NAME        READY   STATUS    RESTARTS      AGE   IP           NODE                   NOMINATED NODE   READINESS GATES
goapp-pod   1/1     Running   1 (35m ago)   15h   10.110.2.2   cwave-cluster-worker2  <none>           <none>
nginx-pod   1/1     Running   1 (35m ago)   16h   10.110.1.2   cwave-cluster-worker   <none>           <none>
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl get no -L memsize
NAME                          STATUS   ROLES           AGE   VERSION   MEMSIZE
cwave-cluster-control-plane   Ready    control-plane   18h   v1.29.4
cwave-cluster-worker          Ready    <none>          18h   v1.29.4   high
cwave-cluster-worker2         Ready    <none>          18h   v1.29.4
root@8a57a9152c43:/code/local/cwave-k8s/practice/label#
```

라벨 삭제

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl label no cwave-cluster-worker memsize-
node/cwave-cluster-worker unlabeled
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl get no -L memsize
NAME                          STATUS   ROLES           AGE   VERSION   MEMSIZE
cwave-cluster-control-plane   Ready    control-plane   18h   v1.29.4
cwave-cluster-worker          Ready    <none>          18h   v1.29.4
cwave-cluster-worker2         Ready    <none>          18h   v1.29.4
```

어노테이션 실습

어노테이션 부여

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl annotate po goapp-pod maker="dangtong", team="dev-team"
pod/goapp-pod annotated
```

어노테이션 삭제

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/label# kubectl annotate po goapp-pod maker-
pod/goapp-pod annotated
```

연습 4-1

- yaml 파일 코드

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  labels:
    name: myapp
    tier: FronEnd
    app: apache
spec:
  containers:
  - name: myapp
    image: bitnami/apache
    resources:
      limits:
        memory: "128Mi"
        cpu: "500m"
    ports:
      - containerPort: 8181
```

- pod 생성

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl apply -f test.yaml
pod/myapp created
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get po
NAME        READY   STATUS              RESTARTS      AGE
goapp-pod   1/1     Running             1 (48m ago)   15h
myapp       0/1     ContainerCreating   0             12s
nginx-pod   1/1     Running             1 (48m ago)   16h
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get po -o wide
NAME        READY   STATUS    RESTARTS      AGE   IP           NODE                   NOMINATED NODE   READINESS GATES
goapp-pod   1/1     Running   1 (48m ago)   15h   10.110.2.2   cwave-cluster-worker2   <none>           <none>
myapp       1/1     Running   0             24s   10.110.1.3   cwave-cluster-worker    <none>           <none>
nginx-pod   1/1     Running   1 (48m ago)   16h   10.110.1.2   cwave-cluster-worker    <none>           <none>
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get po -o wide --show-labels
NAME        READY   STATUS    RESTARTS      AGE   IP           NODE                   NOMINATED NODE   READINESS GATES   LABELS
goapp-pod   1/1     Running   1 (49m ago)   16h   10.110.2.2   cwave-cluster-worker2   <none>           <none>            app=application,env=
prod,name=goapp,tier=backend
myapp       1/1     Running   0             41s   10.110.1.3   cwave-cluster-worker    <none>           <none>            app=apache,name=myap
p,tier=FronEnd
nginx-pod   1/1     Running   1 (49m ago)   16h   10.110.1.2   cwave-cluster-worker    <none>           <none>            <none>
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1#
```

- app=apache인 pod만 조회

```
PS C:\kkk\local_code\cwave-k8s\practice\exercise\4-1> kubectl get po -L app
NAME        READY   STATUS    RESTARTS      AGE   APP
goapp-pod   1/1     Running   1 (57m ago)   16h   application
myapp       1/1     Running   0             66s   apache
nginx-pod   1/1     Running   1 (57m ago)   16h
PS C:\kkk\local_code\cwave-k8s\practice\exercise\4-1> kubectl get po -l app=apache
NAME    READY   STATUS    RESTARTS   AGE
myapp   1/1     Running   0          80s
PS C:\kkk\local_code\cwave-k8s\practice\exercise\4-1>
```

- 라벨 및 어노테이션 추가

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl label po myapp  env=dev
pod/myapp labeled
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl annotation po myapp  created_by=kevin
error: unknown command "annotation" for "kubectl"
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl annotate po myapp  created_by=kevin
pod/myapp annotated
```

- describe로 조회한다.

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl describe po myapp
Name:             myapp
Namespace:        default
Priority:         0
Service Account:  default
Node:             cwave-cluster-worker/192.168.128.3
Start Time:       Tue, 16 Jul 2024 09:42:02 +0900
Labels:           app=apache
                  env=dev
                  name=myapp
                  tier=FronEnd
Annotations:      created_by: kevin
Status:           Running
IP:               10.110.1.3
```

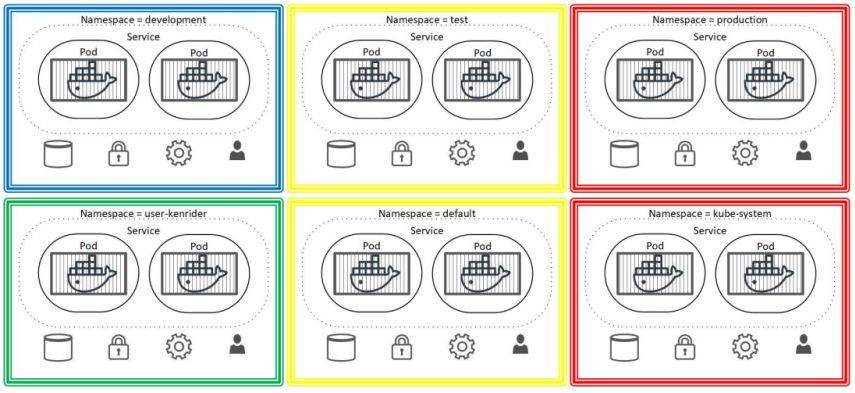
- pod 삭제

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl delete po myapp
pod "myapp" deleted
```

네임스페이스

라벨 셀렉터에서 다중조건 사용

- 쿠버네티스의 객체 이름의 범위를 제공하는것
- 동일한 리소스 이름을 여러 네임스페이스에서 여러 번 사용 가능(개발 / 검증/ 운영 에서 동일한 리소스 명 사용)
- 멀티-테넌트 환경으로 분리하는 효과



테넌트

IT적인 구분이 아닌 사람의 환경 조직을 나누는

쿠버네티스는 외부에서 들어오는 모든 트래픽을 차단하라를 네임스페이스로 수행 가능

어떤 pod로 오는건 막고 어떤 pod에서 오는건 열어둬라

네임스페이스 목적

1. 리소스 관리
2. 트래픽 네트워크 차단 관리 ← 네트워크 담당자가 필요없이 네트워크 지식이 없이 개발자가 할 수 있게 해줌 ← 데브옵스가 한다. 네트워크 실무자들은 이런 걸 모를 수 있다.
3. 개발 검증 운용

   aws에 배포를 할 때 개발계 서비스계 운영계로 나누어서 따로 배포를 한 다음 테스트 진행

   창업을 했을 때 최소 2개는 한다. 본인 local 노트북 배포 aws 운영계 배포

kubectl namespace

- 네임스페이스 목록

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get ns
NAME                 STATUS   AGE
default              Active   19h
kube-node-lease      Active   19h
kube-public          Active   19h
kube-system          Active   19h
local-path-storage   Active   19h
```

- 다른 네임스페이스에서의 pod를 보자

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubens kube-system
Context "kind-cwave-cluster" modified.
Active namespace is "kube-system".
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get po
NAME                                                 READY   STATUS    RESTARTS      AGE
coredns-76f75df574-2b6ps                             1/1     Running   1 (75m ago)   19h
coredns-76f75df574-f95kg                             1/1     Running   1 (75m ago)   19h
etcd-cwave-cluster-control-plane                     1/1     Running   0             75m
kindnet-45rmr                                        1/1     Running   1 (75m ago)   19h
kindnet-brc7c                                        1/1     Running   1 (75m ago)   19h
kindnet-bvnmr                                        1/1     Running   1 (75m ago)   19h
kube-apiserver-cwave-cluster-control-plane           1/1     Running   0             75m
kube-controller-manager-cwave-cluster-control-plane  1/1     Running   1 (75m ago)   19h
kube-proxy-5cl2j                                     1/1     Running   1 (75m ago)   19h
kube-proxy-ms9mj                                     1/1     Running   1 (75m ago)   19h
kube-proxy-n4z7n                                     1/1     Running   1 (75m ago)   19h
kube-scheduler-cwave-cluster-control-plane           1/1     Running   1 (75m ago)   19h
```

- default에서의 pod를 보자

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubens default
Context "kind-cwave-cluster" modified.
Active namespace is "default".
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get po
NAME        READY   STATUS    RESTARTS      AGE
goapp-pod   1/1     Running   1 (76m ago)   16h
nginx-pod   1/1     Running   1 (76m ago)   16h
```

네임스페이스 실습

- 명령어로 namespace 제작

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl create ns my-ns
namespace/my-ns created
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get ns
NAME                 STATUS   AGE
default              Active   19h
kube-node-lease      Active   19h
kube-public          Active   19h
kube-system          Active   19h
local-path-storage   Active   19h
my-ns                Active   14s
```

- 내가 현재 만든 ns에 아무것도 없는 것 확인

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubens my-ns
Context "kind-cwave-cluster" modified.
Active namespace is "my-ns".
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get po
No resources found in my-ns namespace.
```

- ns에 만들어졌는지 확인

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl apply -f test.yaml
pod/myapp created
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/4-1# kubectl get po
NAME    READY   STATUS    RESTARTS   AGE
myapp   1/1     Running   0          2s
```

- yaml로 제작

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: first-namespace
```

- ns 확인

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/ns# kubectl apply -f ns.yaml
namespace/first-namespace created
root@8a57a9152c43:/code/local/cwave-k8s/practice/ns# kubectl get ns
NAME                 STATUS   AGE
default              Active   19h
first-namespace      Active   4s
kube-node-lease      Active   19h
kube-public          Active   19h
kube-system          Active   19h
local-path-storage   Active   19h
my-ns                Active   3m32s
```

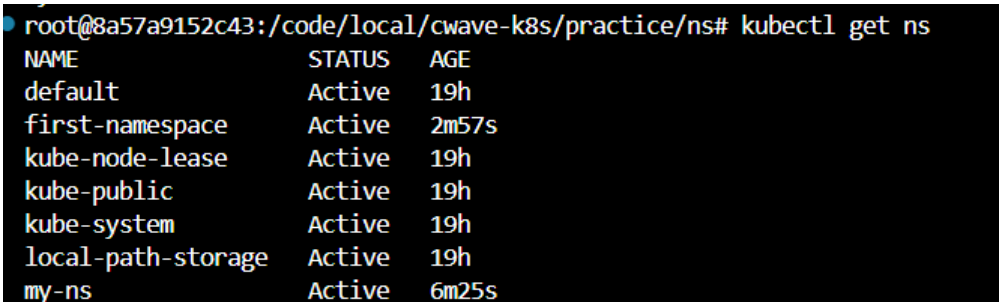
연습문제 5-1

[연습문제 5-1]

1. 쿠버네티스 클러스터에 몇개의 네임스페이가 존재 하나요?
2. my-dev 라는 네임스페이를 생성하고 nginx Pod를 배포 하세요
3. 현재 네임스페이스(Current Namespace)를 Kube-system 으로 변경 하세요
4. 모든 네임스페이스의 모든 리소스를 한번에 조회 하세요

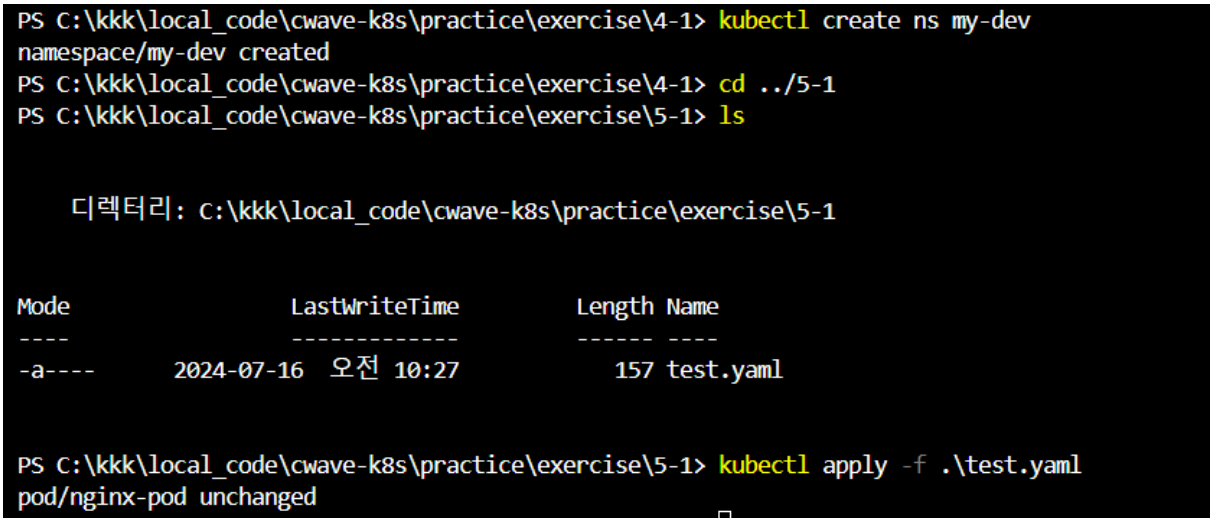
6. kubectl 기본 사용법

1. 네임스페이스 확인



```
root@8a57a9152c43:/code/local/cwave-k8s/practice/ns# kubectl get ns
NAME                 STATUS   AGE
default              Active   19h
first-namespace      Active   2m57s
kube-node-lease      Active   19h
kube-public          Active   19h
kube-system          Active   19h
local-path-storage   Active   19h
my-ns                Active   6m25s
```

2. my -dev 네임 스페이스 생성 && nginx pod 배포



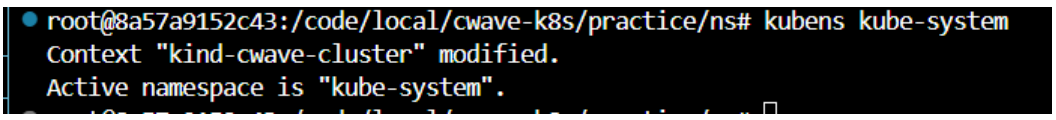
```
PS C:\kkk\local_code\cwave-k8s\practice\exercise\4-1> kubectl create ns my-dev
namespace/my-dev created
PS C:\kkk\local_code\cwave-k8s\practice\exercise\4-1> cd ../5-1
PS C:\kkk\local_code\cwave-k8s\practice\exercise\5-1> ls


    디렉터리: C:\kkk\local_code\cwave-k8s\practice\exercise\5-1


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        2024-07-16  오전 10:27            157 test.yaml


PS C:\kkk\local_code\cwave-k8s\practice\exercise\5-1> kubectl apply -f .\test.yaml
pod/nginx-pod unchanged
```

3. 네임 스페이스 변경



```
root@8a57a9152c43:/code/local/cwave-k8s/practice/ns# kubens kube-system
Context "kind-cwave-cluster" modified.
Active namespace is "kube-system".
```

kubectl 기본 명령어

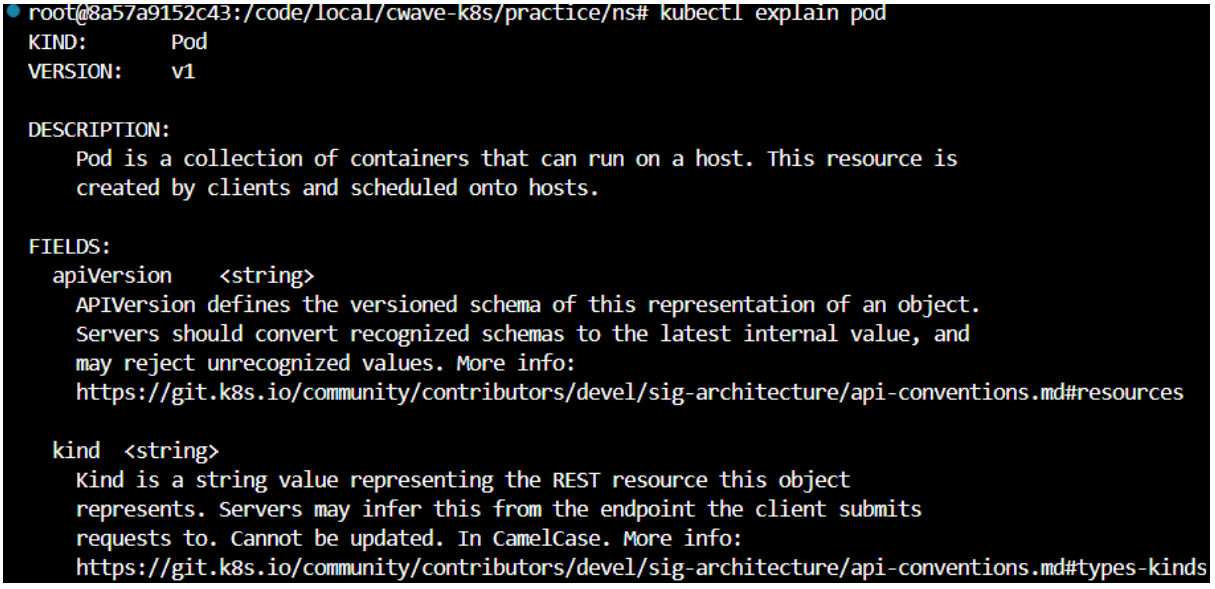
> 💡 명령어가 기억이 나지 않으면 칠 수 있는 데까지 치고 -h로 옵션을 확인해라

지금까지 배운 resource는 2개

- pod
- namespace

explain

명령어를 설명해준다

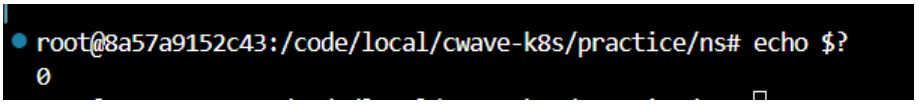


Liveness Probe

HTTP는 커넥션 할 때 쓰고

TCP는 커넥션을 해서 사용이다. 중요하다

💡 echo $?를 하면 이전 명령어의 값의 결과를 확인 가능하다.



0이 정상이고 다른 모든 숫자가 오류인 것 리눅스에서 그렇다

서비스하지 않는 프로그램 = 배치 프로그램

EXEC로 한다.

모든 준비가 끝나야 SOCKET을 연다 LISTEN을 하는 것은 모든 준비가 다 되었다는 것..

공통설정

- InitailDelaySeconds

Delay seconds는 넉넉하게 최소 3배는 잡아야 한다. 최악의 상황까지 가정

- Timeout seconds

제일 안좋은게 응답을 늦게 주는 것

- successThreshold

  성공했다고 판단하기 위해 필요한 성공 횟수

- failure Threshold

  실패했다고 판단하기 위해 필요한 실패 횟수

http 추가 설정

대부분의 인증 정보가 헤더에 들어가기에 헤더 적어야한다.

실습

- yaml code

```
apiVersion: v1
kind: Pod
metadata:
```

```
  name: liveness-pod
  labels:
    name: liveness
spec:
  containers:
  - name: liveness-container
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
     httpGet:
       path: /healthz
       port: 8080
       httpHeaders:
       - name: Custom-Header
         value: Awesome
     initialDelaySeconds: 3
     periodSeconds: 3
    resources:
      limits:
        memory: "128Mi"
        cpu: "500m"
    ports:
      - containerPort: 8080
```

- 3번 하고 안되니 쿠버네티스에서 포기

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/livenessProbe# kubectl get po -w
NAME           READY   STATUS    RESTARTS       AGE
goapp-pod      1/1     Running   1 (122m ago)   17h
liveness-pod   1/1     Running   0              19s
nginx-pod      1/1     Running   1 (122m ago)   17h
liveness-pod   1/1     Running   1 (2s ago)     23s
liveness-pod   1/1     Running   2 (2s ago)     41s
liveness-pod   1/1     Running   3 (2s ago)     59s
liveness-pod   0/1     CrashLoopBackOff   3 (0s ago)     75s
^Croot@8a57a9152c43:/code/local/cwave-k8s/practice/livenessProbe# kubectl describe po liveness-pod
Name:             liveness-pod
```

- describe로 상태 확인

```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  88s                  default-scheduler  Successfully assigned default/liveness-pod to cwave-cluster-worker2
  Normal   Pulled     86s                  kubelet            Successfully pulled image "k8s.gcr.io/liveness" in 1.958s (1.958s including waitin
g)
  Normal   Pulled     66s                  kubelet            Successfully pulled image "k8s.gcr.io/liveness" in 1.432s (1.432s including waitin
g)
  Normal   Created    48s (x3 over 86s)    kubelet            Created container liveness-container
  Normal   Started    48s (x3 over 86s)    kubelet            Started container liveness-container
  Normal   Pulled     48s                  kubelet            Successfully pulled image "k8s.gcr.io/liveness" in 1.003s (1.003s including waitin
g)
  Normal   Pulling    31s (x4 over 88s)    kubelet            Pulling image "k8s.gcr.io/liveness"
  Warning  Unhealthy  31s (x9 over 73s)    kubelet            Liveness probe failed: HTTP probe failed with statuscode: 500
  Normal   Killing    31s (x3 over 67s)    kubelet            Container liveness-container failed liveness probe, will be restarted
root@8a57a9152c43:/code/local/cwave-k8s/practice/livenessProbe#
```

86초 동안 3번 발생했다.

Replication Controller

liveness probe는 객체는 아니다. Replication Controller는 객체이다.
pod를 복제하는 데 사용된다.
소프트웨어는 관심사의 분리 역할을 나눠 설계하는게 트렌드이다.

복제 컨트롤러라는 객체는 2라 설정하면 pod를 2개를 유지하도록 노력한다.

복제 컨트롤러가 있으면 전체 노드가 죽지 않는 이상 노드가 계속 살아난다.

복제 컨트롤러는 무언가를 감시한다.
라벨, 갯수, 파드 템플릿 복제컨트롤러를 구성한다 세 개의 요소가

파드 템플릿에는 야말파일이 들어간다고 보면 된다.

실습

```yaml
io.k8s.api.core.v1.ReplicationController (v1@replicationcontroller.json)
apiVersion: v1
kind: ReplicationController
metadata:
  name: goapp-rc
spec:
  replicas: 5
  selector:
    app: goapp-pod
  template:
    metadata:
      name: goapp
      labels:
        app: goapp-pod
    spec:
      containers:
        - name: goapp
          image: <Image>
          ports:
            - containerPort: <Port>

```

라벨 이름하고 셀렉터 이름이 같아야 한다

```
hostname: goapp-rc-sw7kc
root@8a57a9152c43:/code# kubectl get all
NAME                 READY   STATUS    RESTARTS   AGE
pod/goapp-rc-78vff   1/1     Running   0          2m14s
pod/goapp-rc-jsdnz   1/1     Running   0          2m14s
pod/goapp-rc-nws6b   1/1     Running   0          2m14s
pod/goapp-rc-pz8nt   1/1     Running   0          2m14s
pod/goapp-rc-sw7kc   1/1     Running   0          2m14s

NAME                              DESIRED   CURRENT   READY   AGE
replicationcontroller/goapp-rc    5         5         5       2m14s

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.120.0.1    <none>        443/TCP   10m
root@8a57a9152c43:/code#
```

- portforwading

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/Replication_Controller# kubectl port-forward goapp-rc-sw7kc 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

- curl 확인

```
root@8a57a9152c43:/code# curl http://localhost:8080
hostname: goapp-rc-sw7kc
root@8a57a9152c43:/code#
```

삭제

```
root@8a57a9152c43:/code# kubectl get po -w
NAME            READY   STATUS    RESTARTS   AGE
goapp-rc-78vff  1/1     Running   0          3m44s
goapp-rc-jsdnz  1/1     Running   0          3m44s
goapp-rc-nws6b  1/1     Running   0          3m44s
goapp-rc-pz8nt  1/1     Running   0          3m44s
goapp-rc-sw7kc  1/1     Running   0          3m44s
kubectl delete po goapp-rc-nws6b
goapp-rc-nws6b  1/1     Terminating   0          4m22s
goapp-rc-vj9jn  0/1     Pending       0          0s
goapp-rc-vj9jn  0/1     Pending       0          0s
goapp-rc-vj9jn  0/1     ContainerCreating   0          0s
goapp-rc-nws6b  0/1     Terminating         0          4m22s
goapp-rc-nws6b  0/1     Terminating         0          4m22s
goapp-rc-nws6b  0/1     Terminating         0          4m22s
goapp-rc-nws6b  0/1     Terminating         0          4m22s
goapp-rc-vj9jn  1/1     Running             0          2s
```

감시할 때 하나가 삭제되면 5개를 유지하려 했으니 하나를 바로 만든다.

다만 여기서 중요한 것이 다 삭제가 된 이후에 만들어진다.

라벨 변경

```
root@8a57a9152c43:/code# kubectl label po goapp-rc-pz8nt app=goapp --overwrite
pod/goapp-rc-pz8nt labeled
root@8a57a9152c43:/code# kubectl get po -L app
NAME            READY   STATUS    RESTARTS   AGE     APP
goapp-rc-78vff  1/1     Running   0          8m2s    goapp-pod
goapp-rc-jsdnz  1/1     Running   0          8m2s    goapp-pod
goapp-rc-kdncm  1/1     Running   0          7s      goapp-pod
goapp-rc-pz8nt  1/1     Running   0          8m2s    goapp
goapp-rc-sw7kc  1/1     Running   0          8m2s    goapp-pod
goapp-rc-vj9jn  1/1     Running   0          3m40s   goapp-pod
```

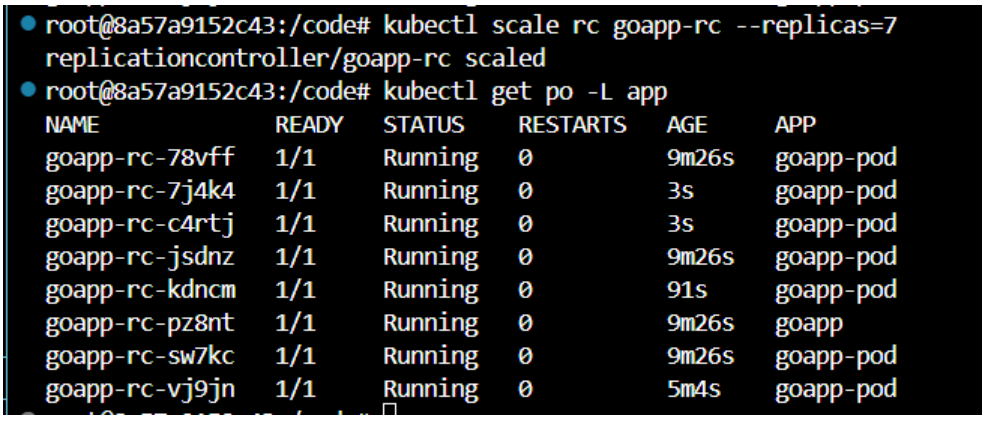
감시 중인 것이 하나가 사라졌으니 하나를 더 만듦

여기서 알 수 있는 점

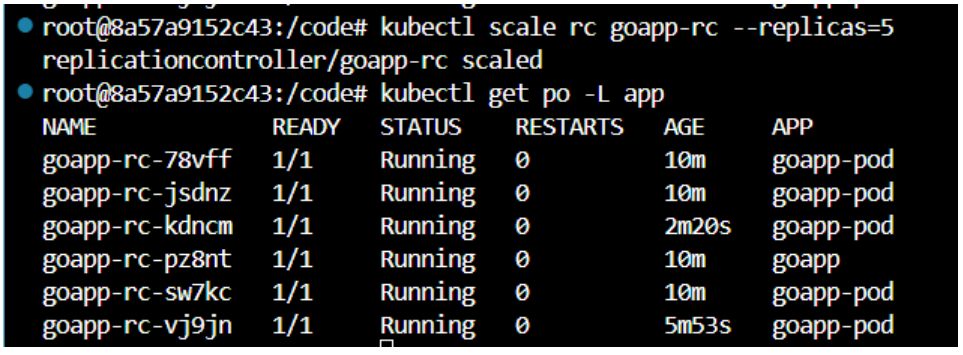
> 💡 라벨로 감시를 하고 항상 5개를 유지하려 노력한다.

scale 변경

- 늘리는 거 확인



- 줄이는 거 확인



- vi로도 수정 가능

```
  namespace: default
  resourceVersion: "39306"
  uid: 86ceb8a8-111a-4479-b1ec-8c07ed97dfd1
spec:
  replicas: 10
  selector:
    app: goapp-pod
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: goapp-pod
      name: goapp
    spec:
      containers:
      - image: dangtong/goapp
        imagePullPolicy: Always
        name: goapp-container
        ports:
```

```
root@8a57a9152c43:/code# kubectl edit rc goapp-rc
replicationcontroller/goapp-rc edited
root@8a57a9152c43:/code# kubectl get po -L app
NAME             READY   STATUS              RESTARTS   AGE     APP
goapp-rc-2xvm4   0/1     ContainerCreating   0          3s      goapp-pod
goapp-rc-78vff   1/1     Running             0          11m     goapp-pod
goapp-rc-fdzrh   0/1     ContainerCreating   0          3s      goapp-pod
goapp-rc-gvx4h   0/1     ContainerCreating   0          3s      goapp-pod
goapp-rc-jsdnz   1/1     Running             0          11m     goapp-pod
goapp-rc-kdncm   1/1     Running             0          3m26s   goapp-pod
goapp-rc-pz8nt   1/1     Running             0          11m     goapp
goapp-rc-sw7kc   1/1     Running             0          11m     goapp-pod
goapp-rc-vj9jn   1/1     Running             0          6m59s   goapp-pod
goapp-rc-wg26s   0/1     ContainerCreating   0          3s      goapp-pod
goapp-rc-x49k6   0/1     ContainerCreating   0          3s      goapp-pod
```

이미지를 변경하면서 pod의 개수를 변경하면

어떤 부분은 변경이 되고 어떤 부분은 변경이 안 되는 경우가 생긴다.

차라리 다 내리고 올려라

Replicaset

서비스를 하는데 글로벌 서비스를 한다.

컨트롤러 대체하기 위해 나옴

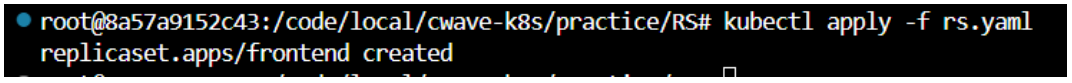
replica set은 연산자를 지원한다.

연산식을 expression으로 사용할 수 있다.

실습

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  # modify replicas according to your case
  replicas: 5
  selector:
    matchLabels:
      tier: frontend
    matchExpressions:
      - {key: env, operator: In, values: [prod]}
      - {key: priority, operator: NotIn, values: [low]}
  template:
    metadata:
      labels:
        tier: frontend
        env : prod
        priority : high
    spec:
      containers:
      - name: php-redis
        image: us-docker.pkg.dev/google-samples/containers/gke/gb-frontend:v5
```

- 생성



- 확인

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/RS# kubectl get all
NAME                 READY   STATUS              RESTARTS   AGE
pod/frontend-5tp2j   0/1     ContainerCreating   0          68s
pod/frontend-8rzcw   0/1     ContainerCreating   0          68s
pod/frontend-rxd6h   0/1     ContainerCreating   0          68s
pod/frontend-tkwt5   0/1     ContainerCreating   0          68s
pod/frontend-z2dhg   0/1     ContainerCreating   0          68s

NAME                 TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.120.0.1    <none>        443/TCP   73s

NAME                         DESIRED   CURRENT   READY   AGE
replicaset.apps/frontend     5         5         0       68s
```

연습 문제 9-1

[연습문제 9-1]

1. nginx:1.9.1 Pod 3개로 구성된 Replication Controller를 작성 하세요
2. Replication Controller 만 삭제 하세요 (Pod 는 유지)
3. 남겨진 nginx Pod를 관리하는 ReplicaSet 을 작성하된 replica 4개로 구성 하세요
4. nginx Pod 를 6개로 Scale Out 하세요

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-rc
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
```

```
      image: nginx:1.9.1
      ports:
        - containerPort: 80
```

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/9-1# kubectl apply -f test.yaml
replicationcontroller/nginx-rc created
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/9-1# kubectl get po
NAME             READY   STATUS    RESTARTS   AGE
nginx-rc-f445g   1/1     Running   0          6s
nginx-rc-jd5qn   1/1     Running   0          6s
nginx-rc-mv68p   1/1     Running   0          6s
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/9-1# kubectl delete rc test-rc --cascade=orphan
Error from server (NotFound): replicationcontrollers "test-rc" not found
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/9-1# kubectl delete rc nginx-rc --cascade=orphan
replicationcontroller "nginx-rc" deleted
```

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.9.1
        ports:
        - containerPort: 80
```

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/9-1# kubectl scale rs nginx-rs --replicas=5
replicaset.apps/nginx-rs scaled
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/9-1# kubectl get po,rc
NAME                 READY   STATUS    RESTARTS   AGE
pod/nginx-rc-f445g   1/1     Running   0          8m56s
pod/nginx-rc-jd5qn   1/1     Running   0          8m56s
pod/nginx-rc-mv68p   1/1     Running   0          8m56s
pod/nginx-rs-8v2p5   1/1     Running   0          5s
pod/nginx-rs-sqcg9   1/1     Running   0          5s
```

Deamon set

로그를 따로 기록하지 말고 표준 출력으로 보내라

pod에서 로그를 기록하지 말고 표준 출력으로 pod가 있는 node에 컨테이너가 관리하는 임시 파일 영역에 기록된다.

리눅스 파일 시스템 어딘가에 표준 출력이 임시로 남는다. 임시기에 재부팅시 날라갈 수 있다.

오버레이 파일시스템 유니온 파일 시스템의 특징

아랫단에 있는 파일을 무시

따라서 docker에서 수정과 설정이 가능한 것

맨 위 layer만이 rw 가능하니까

결국엔 관심사의 분리

배포를 잘해라 후에 수정하지 말고

- **데몬 셋이 쓰이는 이유 1**

fluntd daemon으로 node에 떠있으면 node안에 있는 모든 pod안 컨테이너들의 log를 host로 보내줄 수 있음

elastic search라는 db에 이런 모든 로그를 보내고 kibana라는 tool로 대쉬보드에서 log 확인가능하다.

pod안의 컨테이너는 서로의 디스크 볼륨을 공유가 가능, 통신이 자유롭다

- **데몬셋이 쓰이는 이유 2**

pod network

SDN : 네트워크 위에 네트워크 위에 네트워크 오버레이로 만든다.

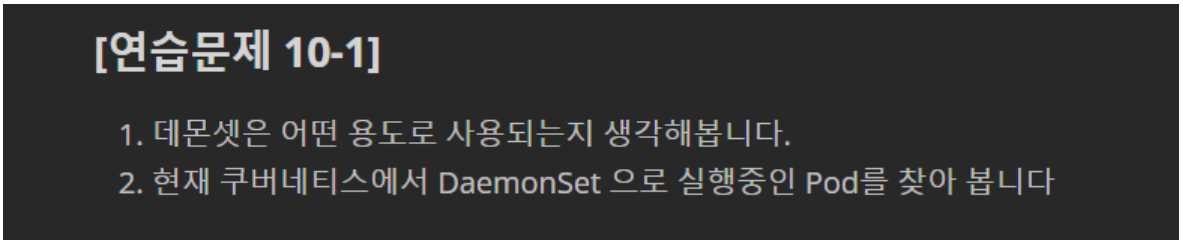
데이터 패킷에 헤더를 계속 붙이는 것 IP, MAC 등등 이렇게 해서 붙여진 것을 까는 역할을 Node안에서 하는 데몬이 하나 있다.

실습

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: goapp-on-ssd
spec:
  selector:
    matchLabels:
      app: goapp-pod
  template:
    metadata:
```

```
    labels:
      app: goapp-pod
  spec:
    nodeSelector:
      disk: ssd
    containers:
    - name: goapp-container
      image: dangtong/goapp
```

연습문제 10-1




```
root@8a57a9152c43:/code/local/cwave-k8s/practice/daemon# kubens kube-system
Context "kind-cwave-cluster" modified.
Active namespace is "kube-system".
root@8a57a9152c43:/code/local/cwave-k8s/practice/daemon# kubectl get ds
```

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/daemon# kubectl get ds
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kindnet      3         3         3       3            3           kubernetes.io/os=linux   23h
kube-proxy   3         3         3       3            3           kubernetes.io/os=linux   23h
root@8a57a9152c43:/code/local/cwave-k8s/practice/daemon# kubectl get po -o wide
```

deployment

rc, rs는 pod를 생성 deployment는 rs를 생성

pod는 컨테이너에만 관심이 있다.
rs는 복제 확장 줄이기에만 관심이 있다.
deployment는 배포에만 관심이 있다.

deployment rollback 이전 버전을 기억하고 있기에 바로 돌릴 수 있다.

deployment → rs → pod

배포 전략

- 재시작

- Ramped

v2와 v1 동시 접속
maxsurge: 1이면 replica +1까지
maxunavailable : 1이면 replica -1까지

롤링 업데이트 적용
가장 일반적으로 많이 쓴다.

- blue/green red/black

개발계와 운영계가 게속 바뀜
버전 자주 변경 되는 프론트 엔드에 적합
롤백 롤아웃이 간단

2배의 리소스가 필요
릴리즈 하기전에 플랫폼에 대한 테스트 수행필요

- 카나리
  소수만 바꿔놓고 체크 후에 들어간다

쿠버네티스에는 rd가 있다.

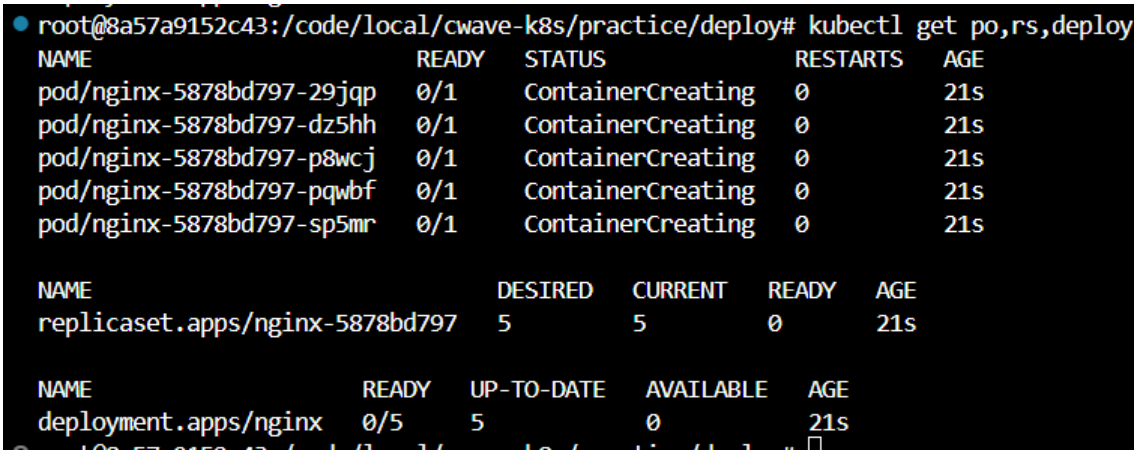
RD	CRD
Resolve definition	dec vustom des

- shadow

v1 트래픽 녹화 v2에 바로

실습

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  replicas: 5
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate # Recrate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.1
        resources:
          limits:
            memory: "128Mi"
            cpu: "500m"
        ports:
        - containerPort: 80
```



- 버전바꿔서 재 실 후 결과 확인

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl get po -w
NAME                     READY   STATUS              RESTARTS   AGE
nginx-5878bd797-29jqp    0/1     ContainerCreating   0          3m11s
nginx-5878bd797-dz5hh    0/1     ContainerCreating   0          3m11s
nginx-5878bd797-p8wcj    0/1     ContainerCreating   0          3m11s
nginx-5878bd797-pqwbf    0/1     ContainerCreating   0          3m11s
nginx-5878bd797-sp5mr    0/1     ContainerCreating   0          3m11s
nginx-545b74fc67-kcrxb   0/1     Pending             0          0s
nginx-545b74fc67-kcrxb   0/1     Pending             0          0s
nginx-5878bd797-29jqp    0/1     Terminating         0          3m35s
nginx-5878bd797-pqwbf    0/1     Terminating         0          3m35s
```

max surge가 1이기에 하나를 띄우고 available이 2이기에 2개를 죽였다.

- 롤아웃

```
^Croot@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl annotate deployment nginx kubernetes.io/change-cause="change image to nginx
1.9.1"
deployment.apps/nginx annotated
root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl rollout history deploy nginx
deployment.apps/nginx
REVISION  CHANGE-CAUSE
1         <none>
2         change image to nginx:x1.9.1
```

롤아웃을 할 때 이렇게 log를 남겨라 나중에 어디가 변경지점인지 확인하기 쉽다

- latest로 다시 해보자

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl apply -f nginx-deploy.yaml
deployment.apps/nginx configured
root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl annotate deployment nginx kubernetes.io/change-cause="change image to nginx:x
latest"
deployment.apps/nginx annotated
root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl rollout history deploy nginx
deployment.apps/nginx
REVISION  CHANGE-CAUSE
1         <none>
2         change image to nginx:x1.9.1
3         change image to nginx:xlatest
```

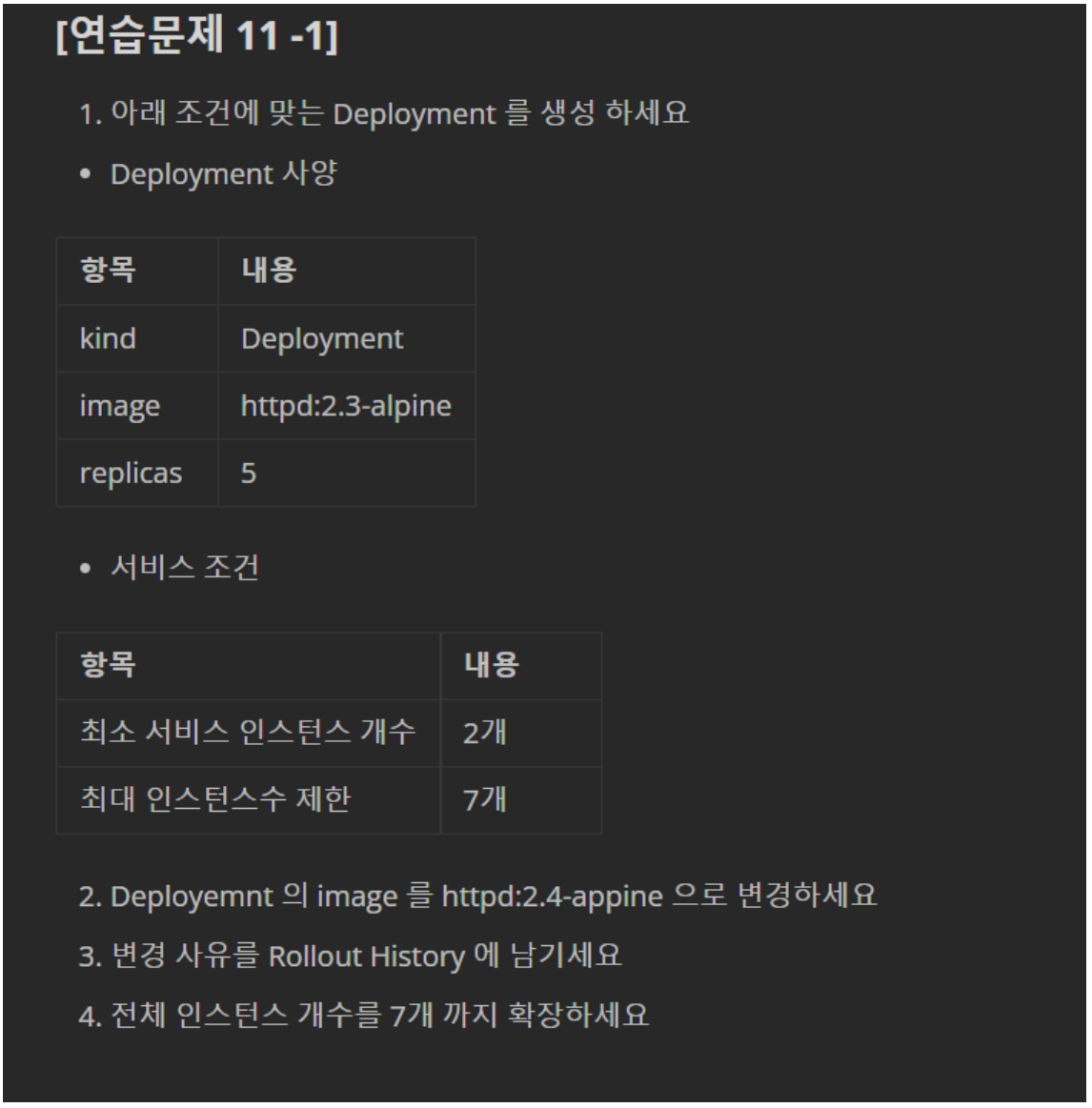
- 롤백을 하여 돌아가보자

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl rollout history deploy nginx
deployment.apps/nginx
REVISION  CHANGE-CAUSE
1         <none>
2         change image to nginx:x1.9.1
3         change image to nginx:xlatest

root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl rollout undo deploy nginx --to-revision=2
deployment.apps/nginx rolled back
root@8a57a9152c43:/code/local/cwave-k8s/practice/deploy# kubectl rollout history deploy nginx
deployment.apps/nginx
REVISION  CHANGE-CAUSE
1         <none>
3         change image to nginx:xlatest
4         change image to nginx:x1.9.1
```

리비전 숫자는 이미지가 변경될 때만 증가한다. deployment는 이미지에만 관심이 있기 때문

연습문제 11



[연습문제 11 -1]

1. 아래 조건에 맞는 Deployment 를 생성 하세요

- Deployment 사양

항목	내용
kind	Deployment
image	httpd:2.3-alpine
replicas	5

- 서비스 조건

항목	내용
최소 서비스 인스턴스 개수	2개
최대 인스턴스수 제한	7개

2. Deployemnt 의 image 를 httpd:2.4-appine 으로 변경하세요
3. 변경 사유를 Rollout History 에 남기세요
4. 전체 인스턴스 개수를 7개 까지 확장하세요

최소 서비스 인스턴스 수 = replica 수 - maxUnavailable 수

← 사람이 가장 적은 시간대에 사용자가 얼마만큼 있을까? 예측

최대 인스턴스 수 = replica 수 + maxSurge 수

← 부하를 측정해서 어느정도까지 리소스자원이 견딜 수가 있는가에 따라 조절

maxsurge를 많이쓰면 cpu의 메모리를 많이 사용한다. 적절히 조절해야 한다.

max surge maxunavailable 둘 다 많이 사용하면 update 시 속도가 빨라진다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```yaml
  name: alpine
spec:
  replicas: 5
  selector:
    matchLabels:
      app: alpine
  strategy:
    type: RollingUpdate # Recrate
    rollingUpdate:
      maxSurge: 2
      maxUnavailable: 3
  template:
    metadata:
      labels:
        app: alpine
    spec:
      containers:
      - name: alpine
        image: httpd:2.3-alpine
        resources:
          limits:
            memory: "128Mi"
            cpu: "500m"
        ports:
        - containerPort: 80
```

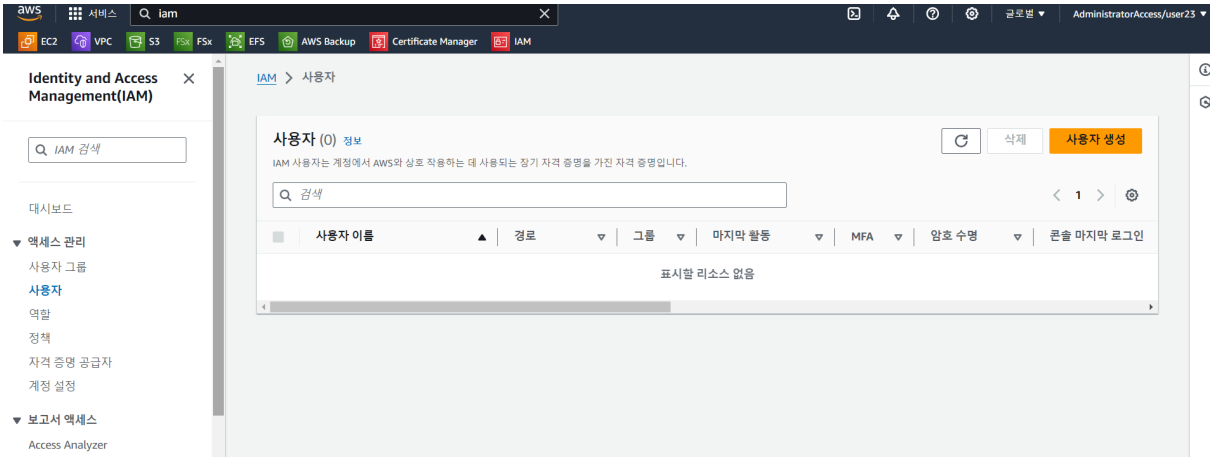
```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/11-1# kubectl get po
NAME                      READY   STATUS         RESTARTS   AGE
alpine-7f745c96d7-89kp4   0/1     ErrImagePull   0          6s
alpine-7f745c96d7-8l6mf   0/1     ErrImagePull   0          6s
alpine-7f745c96d7-ffncf   0/1     ErrImagePull   0          6s
alpine-7f745c96d7-g5bdj   0/1     ErrImagePull   0          6s
alpine-7f745c96d7-qbchz   0/1     ErrImagePull   0          6s
```

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/11-1# kubectl apply -f dp.yaml
deployment.apps/alpine configured
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/11-1# kubectl annotate deployment alpine kubernetes.io/change-cause="change image to
 httpd:2.4-alpine"
deployment.apps/alpine annotated
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/11-1# kubectl rollout history deploy alpine
deployment.apps/alpine
REVISION  CHANGE-CAUSE
1         <none>
2         change image to httpd:2.4-alpine
```

```
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/11-1# kubectl scale deployment alpine --replicas=7
deployment.apps/alpine scaled
root@8a57a9152c43:/code/local/cwave-k8s/practice/exercise/11-1# kubectl get po
NAME                      READY   STATUS             RESTARTS   AGE
alpine-7f745c96d7-8l6mf   0/1     ImagePullBackOff   0          6m27s
alpine-7f745c96d7-9xbzb   0/1     ErrImagePull       0          17s
alpine-7f745c96d7-qbchz   0/1     ImagePullBackOff   0          6m27s
alpine-f788769bb-hz2r6    0/1     ImagePullBackOff   0          5m39s
alpine-f788769bb-kjfct    0/1     ErrImagePull       0          17s
alpine-f788769bb-p9vsc    0/1     ImagePullBackOff   0          5m39s
alpine-f788769bb-rzpgk    0/1     ImagePullBackOff   0          5m39s
alpine-f788769bb-wbzkp    0/1     ImagePullBackOff   0          5m39s
alpine-f788769bb-x88pj    0/1     ImagePullBackOff   0          5m39s
```

AWS

cloudwave 제공 계정으로 aws IAM에 들어간 뒤 사용자를 생성한다.



IAC(Infra at code) → Terraform, 플루밍

테라폼은 사용사례도 많고 자체 언어사용(hcl) ← 어렵지 않다.

시장의 90%가 이걸 사용한다. 따라서 테라폼을 사용

가상머신, 홈서버 구축에도 테라폼 사용 가능

proxmox ← 홈 서버 구축 오픈소스 이마저도 테라폼으로 짤 수 있다.

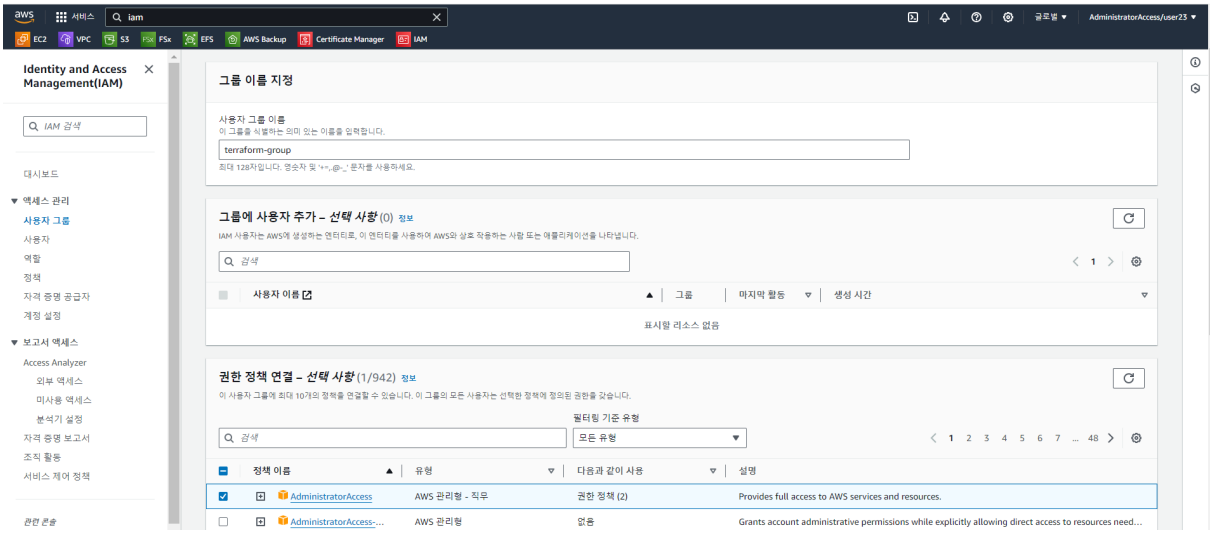
플루민은 파이썬 자바 노드js 다 제공

amazon eks를 code로 짜겠다.

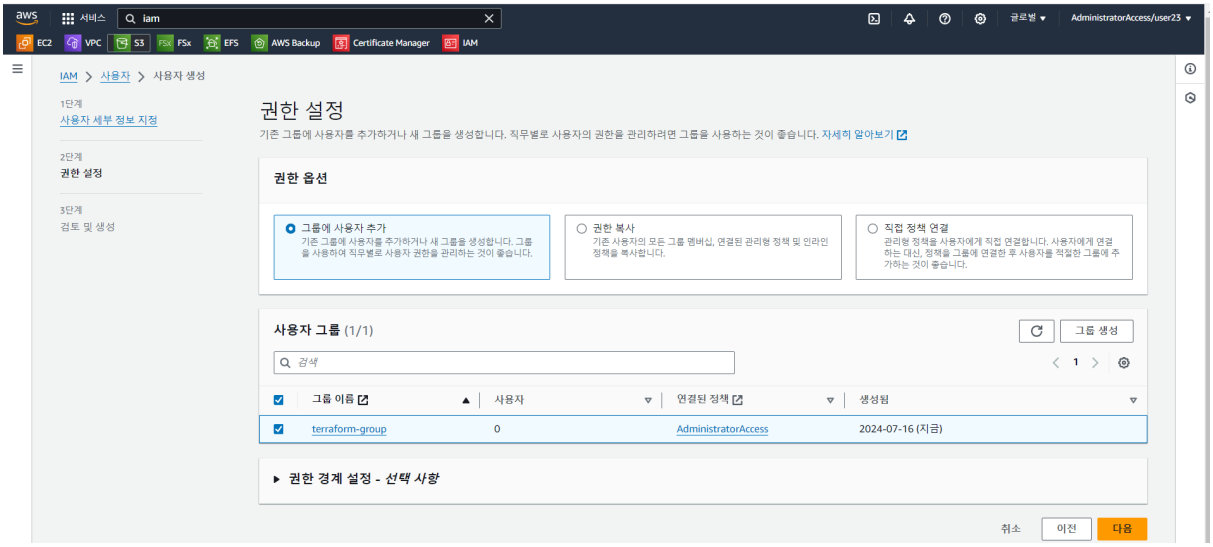
aws는 azure와 google과 비교해서 지저분하고 어렵다 예전에 나와서

테라폼이 사용할 계정을 만들겠다.

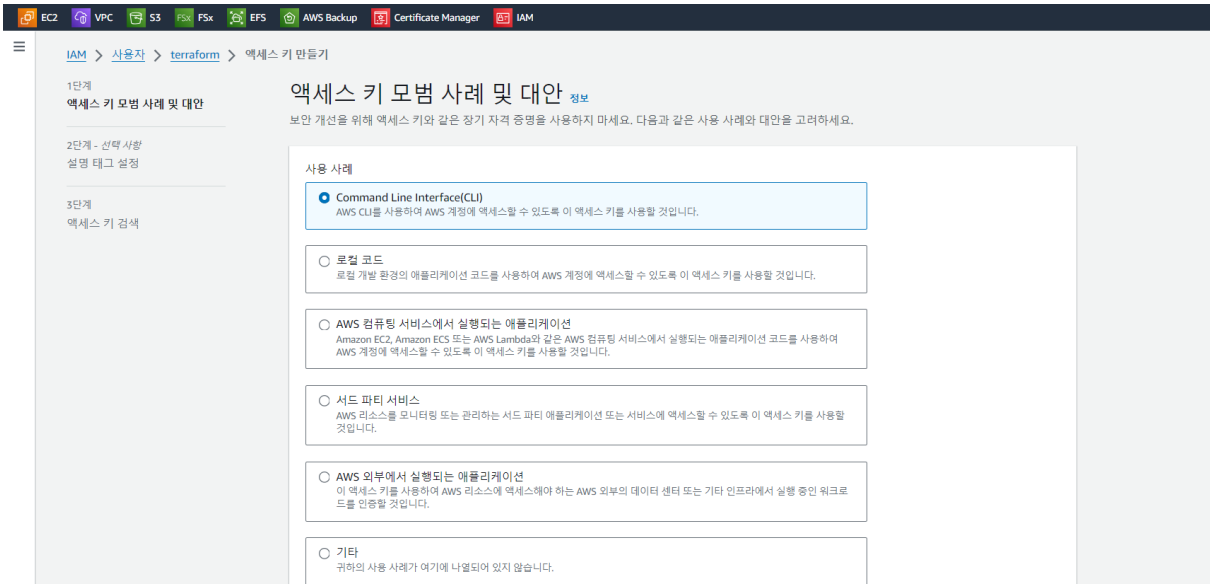
- 그룹 생성



- 그룹에 할당



- 보안 자격 증명에서 사용자 액세스 키 생성



이때 키 다운로드 받아야한다.

안받으면 시크릿 키 볼 방법 x

- 받은 키 값과 아이디를 .env 파일에 넣어주고 추가 변수도 설정해준다.

- 그 후 다시 ide 생성한다

```
PS C:\kkk\local_code\cwave-k8s\makeENV\IDE> docker compose down
time="2024-07-16T16:31:13+09:00" level=warning msg="C:\\kkk\\local_code\\cwave-k8s\\makeENV\\IDE\\docker-compose.yaml: `version` is obsolete"
[+] Running 1/1
 ✓Container ide  Removed                                                                                    4.5s
PS C:\kkk\local_code\cwave-k8s\makeENV\IDE> docker compose up -d
time="2024-07-16T16:31:39+09:00" level=warning msg="C:\\kkk\\local_code\\cwave-k8s\\makeENV\\IDE\\docker-compose.yaml: `version` is obsolete"
[+] Running 1/1
 ✓Container ide  Started                                                                                    0.6s
PS C:\kkk\local_code\cwave-k8s\makeENV\IDE>
```

```
root@8348ca531c4f:/code# env | grep AWS
AWS_PROFILE=cwave
AWS_DEFAULT_REGION=ap-northeast-2
```

env 잡았을 때 aws 들어가 있는 것을 확인 가능

- 웹 ide에 연결됐음 확인 가능하다.

```
aws configure --profile cwave
```

를 통해 아이디와 키 리젼 아웃풋 형식을 입력하고 연결을 한다.

```
root@8348ca531c4f:/code# aws sts get-caller-identity
Account: '211125410568'
Arn: arn:aws:iam::211125410568:user/terraform
UserId: AIDATCKAODMEDZ6DVBUTP
```

```
root@8348ca531c4f:/code# aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                    cwave              env    ['AWS_PROFILE', 'AWS_DEFAULT_PROFILE']
access_key     ****************EQ5S              env
secret_key     ****************dEnL              env
    region                ap-northeast-2         env    ['AWS_REGION', 'AWS_DEFAULT_REGION']
```

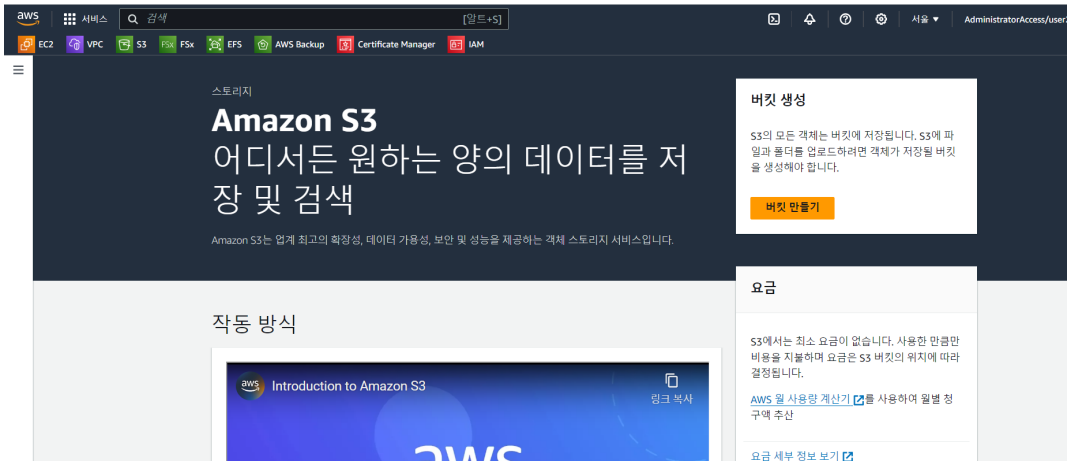
terraform

terraform 코드는 module과 state, code를 들고 있다.

state 파일이 중요 이 파일이 날라가면 인스턴스 삭제 및 재생성이 굉장히 힘들다.

```
17  backend "s3" {
18    bucket = "cloudwave-tf-admin"
19    key    = "cwave/terraform.tfstate"
20    region = "ap-northeast-2"
21  }
```

이런 파일을 aws s3 버킷에 보관하겠다..



ap-northeast-2는 서울 상암동에 위치

zone resource는 8층 9층 10층으로 나뉜다

s3는 글로벌 리소스이기에 다 보인다.

```
17    backend "s3" {
18      bucket = "cloudwave-tf-admin23"
19      key    = "cwave/terraform.tfstate"
20      region = "ap-northeast-2"
21    }
22  }
```

- terraform init

```
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

- terraform plan

```
Plan: 74 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + bastion-public-ip = (known after apply)
  + pem_location      = "./cwave.pem"

Warning: Argument is deprecated

  with module.eks.aws_eks_addon.this["kube-proxy"],
  on .terraform/modules/eks/main.tf line 400, in resource "aws_eks_addon" "this":
 400:   resolve_conflicts        = try(each.value.resolve_conflicts, "OVERWRITE")

The "resolve_conflicts" attribute can't be set to "PRESERVE" on initial resource creation. Use "resolve_conflicts_on_create" and/or
"resolve_conflicts_on_update" instead

(and 2 more similar warnings elsewhere)
```

- terraform apply

- terraform destroy

```
module.eks.aws_security_group_rule.node["ingress_cluster_6443_webhook"]: Destroying... [id=sgrule-4224966247]
module.eks.module.kms.aws_kms_key.this[0]: Destruction complete after 1s
module.eks.aws_security_group_rule.node["egress_all"]: Destroying... [id=sgrule-2284395502]
module.eks.aws_security_group_rule.node["ingress_cluster_8443_webhook"]: Destruction complete after 1s
module.eks.aws_security_group_rule.node["ingress_cluster_4443_webhook"]: Destroying... [id=sgrule-848411121]
module.eks.aws_iam_role_policy_attachment.this["AmazonEKSVPCResourceController"]: Destruction complete after 1s
module.eks.aws_iam_role_policy_attachment.this["AmazonEKSClusterPolicy"]: Destruction complete after 1s
module.eks.aws_iam_role.this[0]: Destroying... [id=cwave-cluster-20240716082244645100000004]
module.eks.aws_security_group_rule.node["ingress_nodes_ephemeral"]: Destruction complete after 1s
module.eks.aws_security_group_rule.node["ingress_self_coredns_udp"]: Destruction complete after 2s
module.eks.aws_security_group_rule.node["ingress_self_coredns_tcp"]: Destruction complete after 2s
module.eks.aws_iam_role.this[0]: Destruction complete after 2s
module.eks.aws_security_group_rule.node["ingress_cluster_9443_webhook"]: Destruction complete after 3s
module.eks.aws_security_group_rule.node["ingress_cluster_kubelet"]: Destruction complete after 3s
module.eks.aws_security_group_rule.node["ingress_cluster_6443_webhook"]: Destruction complete after 4s
module.eks.aws_security_group_rule.node["egress_all"]: Destruction complete after 3s
module.eks.aws_security_group_rule.node["ingress_cluster_4443_webhook"]: Destruction complete after 4s
module.eks.aws_security_group.cluster[0]: Destroying... [id=sg-0967607e6e94657f5]
module.eks.aws_security_group.node[0]: Destroying... [id=sg-0e24a148857555958]
module.eks.aws_security_group.cluster[0]: Destruction complete after 0s
module.eks.aws_security_group.node[0]: Destruction complete after 0s
aws_vpc.vpc: Destroying... [id=vpc-03696eccbe6ab7cab]
aws_vpc.vpc: Destruction complete after 1s

Destroy complete! Resources: 74 destroyed.
```