kubectl run mynginx --image nginx --restart Never

kubectl get pod mynginx --> pod 확인

kubectl get pod ---> 전체 pod 리스트 출력

kubectl get pod mynginx -o yaml --> 실제 쿠버네티스 팟 리소스의 상세

kubectl describe pod mynginx

kubectl exec mynginx -- apt update

kubectl exec mynginx -- apt install -y curl

==> mynginx 에 명령어 수행

kubectl logs mynginx

kubectl cp ~/.bashrc mynginx:/. 쿠베로 보내기

kubectl exec mynginx -- cat /.bashrc 반대로 빼내기

kubectl edit pod mynginx // 파드 정보 수정 삭제 가능

kubectl delete pod mynginx // 삭제 가능

kubectl get pod --namespace kube-system --> 특정 네임스페이스에 대한 pod 살펴보기

-n kube-system도 가능

default 기본이 되는 네임스페이스 , get pod 와 같음

kube-system 은 쿠버네틱스를 이루는 주요 pod들이 들어있음

kubectl get namespace 네임스페이스들 얻을 수 있음

kubectl label node master color=red 라벨링

kubectl get node master -oyaml | grep -A 18 labels 라벨링 확인

kubectl get <RESOURCE>

kubectl get namespace

kubectl get node

kubectl get pod

kubectl get node master -o yaml

kubectl get pod mynginx -o jsonpath=”{ .spec.containers[0].image }”

jq 패키치를 통해 가능

sudo apt install -y jq

kubectl get pod mynginx -o json | jq -r .spec.containers[0].image

kubectl auto completion

==> echo “source <(kubectl completion bash)” >> ~/.bashrc

source ~/.bashrc

kubectl config view

cat ~/.kube/config 과 위는 동일

kubectl config 는 config를 보여주는 역할 크리덴셜을 제공

다양한 유저와 다양한 클러스트를 조합하여 context를 만들고 스위칭해서 사용가능

kubectl config get-contexts 컨텍스트들을 보여줌

kubectl config current-context 현제 컨텍스트를 보여줌

kubectl config use-context my-cluster-name 컨텍스트 스위치

kubectl cluster-info

치트시트 확인하기

Pod 쿠버네티스 최소 단위 Pod

한개 이상의 컨테이너 실행

Pod내에 localhost로 통신

다른 Pod에서 NAT없이 직접 접근

yaml정의서 -> 선언형 명령

apiVersion 네임스페이스나 스코프 결정

kind 리소스 타입 정의

metadata : 메타정보

label : 라벨정보

run : mynginx 키는 런이고 라벨은 mynginx

name : 이름정보 ==> kubectl run mynginx 에 myningx 정의

spec : 리소스 정의

containers : 컨테이너 스펙 정의

- name : nginx => - 는 리스트 정의 name은 nginx 이고 image 는 nginx 인 컨테이너

image : nginx

yaml을 통해 Pod 생성하기

kubectl apply -f mynginx.yaml => apply 선언형 명령어 전달

cat redirection

cat << EOF | kubectl apply -f -

apiVersion: v1

kind : Pod .... cat으로 만들기

kubectl exec mynginx -- curl localhost

kubectl get pod -L run --> 특정 키값에 대한 쿼리를 하고 싶을때 -L하면 됨

kubectl get pod mynginx --show-labels ==> 전체 라벨 조회

kubectl get pod --show-labels ==> 모든 팟의 전체 정보 조회

다시 kubectl apply -f mynginx.yaml 하면 변경포인트에 수정해줌. 선언형 명령의 장점

Filtering

kubectl get pod -l run ==> run을 가진 pod들이 조회됨

kubectl get pod -l run=mynginx ==> key 가 run이고 value가 mynginx인 pod 조회

node select 설정

kubectl label node master disktype=ssd

nodeSelector: ==> 어떤 라벨을 가진 노드에 할당할지

disktype: ssd ==> 해당 pod 는 disktype이 ssd인 노드를 선택하겠다는 의미

특정 노드에 나의 팟 할당할 수 있음.

host local volume 연결

kubectl exec volum-nginx -- ls -alh /test-volume ==> 홈디렉토리에 연결됨 // volume-nginx.yaml 참고

매개변수 전달

예제 파일

apiVersion: v1

kind: Pod

metadata:

name: env-pod

spec:

containers:

- name: nginx

image: nginx

command: [ “echo” ] ==> 명령어

args: [ “abc”, “def”] ==> 매개변수

kubectl logs arg-pod

command echo 가 하번만 실행되고 말기때문에 상태가 completed으로 됨

환경변수를 통한 정보 전달

env 를 통한 전달

apiVersion: v1

kind: Pod

metadata:

name: env-pod

spec:

containers:

- name: nginx

image: nginx

env:

- name: my\_env ==> 키 값

value: “this is nginx!” ==> value

kubectl exec env-pod -- printenv 로 출력

CONFIG - MAP

kubectl create <RESOURCE> <name>

kubectl create namespace new-ns

kubectl create configmap new-cm

Config mma 생성 및 활용

kubectl configmapp <name> <data-source>

game.properties파일

enemies=aliens

lives=3

enemies.cheat=true

enemies.cheat.level=noGoodRotten

secret.code.passphrase=UUDDLRLRABAS

secret.code.allowed=true

kubectl create configmap game-config --from-env-file=game.properties

==> 환경변수형태로 키 밸류 타입을 넣는다.

kubectl get configmapp game-config

kubectl get configmapp game-config -o yaml

configmap ==> 데이터를 저장하는 그릇

레퍼런스 하기 , 컨피그맵 사용하기

valueFrom프로퍼티 사용

apiVersion: v1

kind: Pod

metadata:

name: cat-env

spec:

containers:

- name: cat-env

image: k8s.gcr.io/busybox

command: [ “printenv” ]

args: [ “CONFIG\_KEY” ]

env:

- name: CONFIG\_KEY

valueFrom:

configMapKeyRef:

name: game-config

key: enemies.cheat

envFrom 프로퍼티 사용

apiVersion: v1

kind: Pod

metadata:

name: cat-env

spec:

containers:

- name: cat-env

image: k8s.gcr.io/busybox

command: [ “printenv” ]

envFrom: ==> 전체 컨피그맵 레프

- configMapRef:

name: game-config

kubectl logs cat-env하면 전부 출력

환경변수를 전체 파일로 전달 -- > volumeMounts

apiVersion: v1

kind: Pod

metadata:

name: volume-config

spec:

containers:

- name: volume-config

image: nginx

volumeMounts:

- name: config-volume

mounPath: /etc/conf

volumes:

-name: config-volume

configMapp:

name: game-config

kubectl exec volume-config -- cat /etec/conf/enemies ==> aliens

kubectl exec volume-config -- cat /etec/conf/enemies..cheat ==> true

이 방법은 파일로 만들어서 키와 밸류를 파일로 풀어냄

안풀고 하는 법

kubectl create configmap game-config --from-env-file=game.properties

==> env때문에 환경변수레퍼런싱으로서 작용

kubectl create configmap game-config-file --from-file=game.properties

==> Configmap안에 파일채로 저장

apiVersion: v1

kind: Pod

metadata:

name: volume-config

spec:

containers:

- name: volume-config

image: nginx

volumeMounts:

- name: config-volume

mounPath: /etc/conf

volumes:

-name: config-volume

configMapp:

name: game-config-file

리소스 관리

여기서 리소는 CPU MEM 에 관한것

apiVersion: v1

kind: Pod

metadata:

name: request

spec:

containers:

- name: nginx

image: nginx

resources:

requests:

cpu: 250m

memory: 500Mi

requests를 limits으로 바꾸면 상한선 제한 가능

-o yaml로 확인해보기

requests: 컴퓨터가 사용할 최소 250m 500mi 메모리 필요하다는것 얘기

1000m = 약 1core

250m = 0.25 core

limits: 컴퓨터가 사용할 최대 리소스양 제한

여러 컨테이너 실행

apiVersion: v1

kind: Pod

metadata:

name: mynginx

spec:

containers:

- name: mynginx

image: nginx

- name: curl

image: curlimages/curl

command: [ “sh” ]

args:

- “-c”

- “sleep 5 && curl localhost”

컨테이너 내는 서로 로컬호스트로 통신가능

kubectl logs two-container -c mynginx == > 컨테이너 두개이기 때문에 하나 선택

컨테이너끼리의 선후 관계를 보장할 수 없어서 sleep 5를 해줌

initContainers : 초기화 컨테이너

apiVersion: v1

kind: Pod

metadata:

name: init-container

spec:

containers:

- name: buysbox

image: k8s.gcr.io/busybox

command:[ “ls” ]

args: [ “/work-dir/moby” ]

volumeMounts:

- name: workdir

mountPath: /work-dir

initContainers:

- name: git

image: alpine/git

command: [ “sh” ]

args:

- “-c”

- “git clone <https://github.com/moby/moby.git> \ /work-dir/moby”

volumeMounts:

- name: wordir

mountPath: “/work-dir”

volumes:

- name: workdir

emptyDir: {}

emptyDir==> 임시 데이터 저장소 리눅스의 /tmp Pod에 따라 생성및 삭제됨

kubectl logs init-container -c git -f // -f 주면 테일링이 된다.

pod의 건강상태?

헬스체크 기능 !

livenessProbe

apiVersion: v1

kind: Pod

metadata:

name: liveness

spec:

containers:

- name: mynginx

image: nginx

livenessProbe:

httpGet:

path: /

port: 80

쿠버네티스가 80포트를 통해 지속적으로 핑을 날려 건강 검사

살아있는지 검사

readinessProbe

apiVersion: v1

kind: Pod

metadata:

name: liveness

spec:

containers:

- name: mynginx

image: nginx

readinessProbe:

httpGet:

path: /

port: 80

쿠버네티스 Ready 체크

httpGet: return 200 Code

잘해줌.

file을 이용한 체크

apiVersion: v1

kind: Pod

metadata:

name: readiness-command

spec:

containers:

- name: mynginx

image: nginx

readinessProbe:

exec:

command: [ “ls”, “/work-dir/moby” ]

volumeMounts:

- name: workdir

mountPath: “/work-dir”

initContainers:

- name: git

image: alpine/git

command: [ “sh” ]

args:

- “-c”

- “git clone <https://github.com/moby/moby.git> \ /work-dir/moby”

volumeMounts:

- name: wordir

mountPath: “/work-dir”

volumes:

- name: workdir

emptyDir: {}

readinessProbe

exec: 명령어 실행 후, return code 0

즉 ls 를 실행했을때 성공 0 이면 성공했다 판단

그러나 0이외의 수면 제대로 안됬다고 판단

서비스 리소스 = 쿠버네티스 네트워킹을 담당하는 리소스

apiVersion: v1

kind: Service

metadata:

name: mynginx

spec:

ports:

- port: 80 // 외부 포트

protocol: TCP // 프로토콜

targetPort: 80 // 내부 포트

selector: // 라벨 셀럭터 특정 Pod에게 트레픽을 전달하는 연결고리

run: mynginx

불안정한 Pod와 안정적인 Service

ephemeral ==> Pod는 언제든지 삭제 될 수 있다.

Service는 Stable함.

특정 Pod이 죽어도 Service로 안정적인 일을 할 수 있음.

Service as 도메인 네임

$SERVICE\_NAME.$NAMESPACE.svc.cluster.local

ex) curl mynginx.default.svc.cluster.local == curl mynginx.default

$SERVICE\_NAME.$NAMESPACE == $SERVICE\_NAME.$NAMESPACE.svc.cluster.local

$SERVICE\_NAME == $SERVICE\_NAME.<호출한 네임스페이스>.svc.cluster.local

Cluster Ip

기본 서비스 타입

Cluster 외부 트래픽 전달 X

Cluster 내부에서만 동작

port: 서비스 포트

targetPort 컨테이너 포트

apiVersion: v1

kind: Service

metadata:

name: mynginx

spec:

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

run: mynginx

가장 기본적인 ,, 서비스 타입생략시 생성

라벨 셀렉터로 run: mynginx라는 pod에 제공.

curl.yaml

apiVersion: v1

kind: Pod

metadata:

name: curl

spec:

containers:

- name: curl

image: curlimages/curl

command: [”curl”]

args: [ “10.42.0.44” ]

10.42.0.44 를 mynginx.default.svc.cluster.local 로 대체할수도 있음 or mynginx로

service.yaml

apiVersion: v1

kind: Service

metadata:

name: mynginx

spec:

ports:

- port: 80

protocol: TCP

targetPort: 80

selector:

run: mynginx

pod ip 10.42.0.44

kubectl get service(svc)

service 이름으로 레퍼렌싱

kubectl run busybox --image busybox --restart Never -- sleep infinity

busybox 여러 가지 툴들어있음

kubectl exec busybox -- nslookup mynginx

nodePort

Cluster 외부 트래픽 전달 가능

노드의 port를 통해서 traffic 전달

nodePort: 노출되는 노드의 port

apiVersion: v1

kind: Service

metadata:

name: mynginx

spec:

type: NodePort

ports:

- port: 80

protocol: TCP

targetPort: 80

nodePort: 30080

selector:

run: mynginx

curl <ip>:30080

curl ifconfig.io 하면 외부아이피도 알려줌