kubectl 정리

설치

- curl 다운로드

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

- 권한설정

```
chmod +x ./kubectl
```

- 커맨드로 이동

```
sudo mv ./kubectl/usr/1ocal/bin/kubectl
```

기본 사용법

kubectl [command] [type] [name] [flags]

- command : 자원에 실행하려는 동작 입니다. create, get, delete 등을 사용할 수 있다.
- type : 자원 타입입니다. pod, service, ingress 등을 사용할 수 있습니다.
- name : 자원 이름 입니다.
- flag : 부가적으로 설정할 옵션을 입력합니다.

예제 : 간단한 에코 서버 만들기

```
# 에코 파드서버 시작
$ kubectl run echoserver --generator=run-pod/v1 --image="9eJk8s.gcr.io/echoserver:1.1
0" --port=8080

# 파드 외부 연결
$ kubectl expose po echoserver --type=NodePart

# 파드 조회
$ kubectl get pods
NAMESPACE     NAME                                 READY   STATUS    RESTARTS        AGE
kube-system   coredns-565d847f94-t57s2             1/1     Running   4 (28s ago)

# 서비스 조회
$ kubect get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   7d

# 에코 서버에 접근할 수 있도록 로컬 컴퓨터로 포트포워딩 하겠음
$ kubectl port-forward svc/echoserver 8080:8080
# 브라우저에서 검증 : http://localhost:8080
# 쉘에서 검증 : curl http://localhost:8080

# 파드 로그 확인
$ kubectl logs -f echoserver

# 파드 삭제
$ kubectl delete pod echoserver
```

```
# 서비스 삭제
$ kubectl delete service echoserver

# 파드, 서비스 조회
$ kubectl get pods
$ kubectl get services
```

kubectl get pods

- NAME : 파드 이름을 표시
- READY : 숫자/숫자 형태로 파드의 준비 상태를 표시
  - 0/1 이면 파드는 생성되었으나 사용할 준비가 되지 않았음
  - 1/1 이면 파드가 생성되었고, 사용할 준비가 끝났다는 뜻
- STATUS : 파드의 현 상태를 나타낸다.
  - Running 은 파드가 실행되었다는 뜻
  - Terminating 은 컨테이너 접속 중
  - ContainerCreating 컨테이너 생성 중
- RESTARTS : 해당 파드가 몇 번 재시작 했는지를 표시
- AGE : 파드를 생성한 후 얼마나 시간이 지났는지 나타 냄

kubectl get services

NAME : 서비스 이름을 표시

TYPE : 서비스 타입을 뜻함

CLUSTER-IP : 현재 클러스터 안에서 사용되는 IP

EXTERNAL-IP : 클러스터 외부에서 접속할 때 사용하는 IP, 별도로 지정하지 않으면 NONE

PORT(S) : 해당 서비스에 접속하는 포트를 표시

AGE : 자원을 생성한 후 얼마나 시간이 지났는지 나타냄

POSIX/GNU 스타일의 명령 작성 규칙

POSIX ? Portable Operating System Interface

posix는 유닉스 운영체제에 기반을 두고 있는 표준 운영체제 인터페이스 이다.

GNU ? GNU’s Not Unix

GNU는 운영체제의 하나이자 컴퓨터 소프트웨어의 모음집이다.

- -과 함께 사용하는 옵션은 단일 알파벳/숫자 문자 인자는 짧은 옵션 입니다.
- 일부 옵션은 인자를 필요로 합니다.
- - - 과 함께 사용하는 옵션은 알파벳 두 글자 이상으로 구성한 긴 옵션 입니다.
- - - 이후에 작성하는 인자가 있다면 쿠버네티스 관련 옵션들을 모두 옵션을 종료 합니다.

```
$ kubectl -n default exec my-pod -c my-container -- ls /
```

- - n default : n은 네임스페이스를 지정하는 옵션
- exec my-pod : my-pod라는 이름의 파드에 해당 명령을 실행하라는 뜻
- -c my-container : 컨테이너를 지정하는 옵션
- — ls / : 쿠버네티스 관련 옵션들을 모두 종료한다는 뜻

플래그 (속성이나 옵션이라고 생각)

- 전역 플래그
  - kubectl, options, 명령별 플래그는 명령 각각의 도움말에서 확인할 수 있다.
  - -h (—help) : kubectl [command] —help 형태로 사용합니다.
  - -v [log level] : 명령을 실행하는 과정의 로그를 출력하거나 로그 레벨을 설정 합니다. 디버깅할 때 유용
- 개별 플래그

kubeconfig 환경 변수

$HOME/.kube/config 파일에서 클러스터, 인증, 컨텍스트 정보를 읽어 들입니다.

```
(base) ➜  ls -al $HOME/.kube/config
-rw-------  1 yongyeonkim  staff  843  1  1 20:54 /Users/yongyeonkim/.kube/config

(base) ➜  cat $HOME/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /Users/yongyeonkim/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Sun, 01 Jan 2023 20:54:23 KST
        provider: minikube.sigs.k8s.io
        version: v1.28.0
      name: cluster_info
    server: https://127.0.0.1:63117
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Sun, 01 Jan 2023 20:54:23 KST
        provider: minikube.sigs.k8s.io
        version: v1.28.0
      name: context_info
    namespace: default
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /Users/yongyeonkim/.minikube/profiles/minikube/client.crt
    client-key: /Users/yongyeonkim/.minikube/profiles/minikube/client.key

$ kubectl api-resources
```

도커 데스크톱으로 쿠버네티스를 사용한다면 자동으로 kubeconfig가 설정 됨.

```
kubectl config use-context docker-desktop

# 다른 설정 지정하는 방법
kubectl -kubeconfig=AWSConfig get pods
kubectl -kubeconfig=GCPConfig get pods
```

자동완성

```
# 배시 쉘
echo 'source <(kubectl completion bash)' >> ~/.bashrc

# Z 쉘
echo 'source <(kubectl completion zsh') >> ~/.zshrc
```

kubectl 별칭 사용하기

```
# alias kubectl to k
echo 'alias k=kubectl' >> ~/.bashrc
echo 'complete -F __start_kubectl k' >> ~/.bashrc
```

kubectl 약어 사용하기

```
# node 확인
k get no
# pod 확인
k get po
# deploy 삭제
k delete deploy nginx
```

다양한 사용예

파이프라인 조합

```
# awk 조합 예시
kubectl get nodes -o wide --no-headers | awk '{pring $6}'

# jq 조합 예시
kubectl get nodes -o json | jq -r '.items[].status[].addresses[] | select(.type=="Inte
rnalIP") | .address'

# JSONPath 템플릿 조합 예시
kubectl get nodes -o jsonpath='{.items[*].status.addresses[?@.type=="InternalIP")].add
ress}'

# vim 조합
kubectl get nodes - json | vim -c 'set ft=json' -

# gron을 이용한 jsonpath 구조 파악하기
k get pods etcd-minikube -n kube-system -o json | gron
json = {};
json.apiVersion = "v1";
```

```
json.kind = "Pod";
json.metadata = {};
json.metadata.annotations = {};
json.metadata.annotations["kubeadm.kubernetes.io/etcd.advertise-client-urls"] = "http
s://192.168.49.2:2379";
json.metadata.annotations["kubernetes.io/config.hash"] = "bd495b7643dfc9d3194bd002e968
bc3d";
json.metadata.annotations["kubernetes.io/config.mirror"] = "bd495b7643dfc9d3194bd002e9
68bc3d";
json.metadata.annotations["kubernetes.io/config.seen"] = "2022-12-25T11:34:11.78618904
8Z";
json.metadata.annotations["kubernetes.io/config.source"] = "file";
json.metadata.creationTimestamp = "2022-12-25T11:34:12Z";
json.metadata.labels = {};

# go-template 예시
kubectl get pod etcd-minikube -o go-template="{{range .status.containerStatuses}}{{.la
stState.terminated.message}}{{end}}" -n kube-system
<no value>%

# netshoot 컨테이너 실행
kubectl run my-shell --rm -i --tty --image nicolaka/netshoot -- bash

# 호스트 네트워크 네임스페이스
apiVersion: v1
kind: Pod
metadata:
  name: my-shell-hostnet
  namespace: default
spec:
  hostNetwork: true
  dnsPolicy: ClusterFirstWithHostNet
  containers:
  - name: my-shell-hostnet
    image: nicolaka/netshoot
    args:
    - sleep
    - infinity

# 호스트 네트워크 네임스페이스를 이용한 netshoot pod 생성
kubectl exec -it my-shell-hostnet -- bash
```

보통 컨테이너 이미지를 만들 때는 용량을 줄이고 보안을 강화하려고 실행에 필요한 최소한의 환경만 설정 한다.

그래서 여러 가지 테스트용 명령어들을 제외 한다.

이 때문에 쿠버네티스 클러스터에서 직접 네트워크 문제들을 추적하기는 어렵다.

그래서 네트워크 문제를 추적할 때 필요한 여러가지 도구를 포함한 별도의 컨테이너인 netshoot가 있다.