2장

Kubernetes 기본 명령어 다루기

전체 내용

k8s 관리도구

k8s 명령어 체계

YAML Template

k8s 관리도구

CLI 형태

Yaml 형태

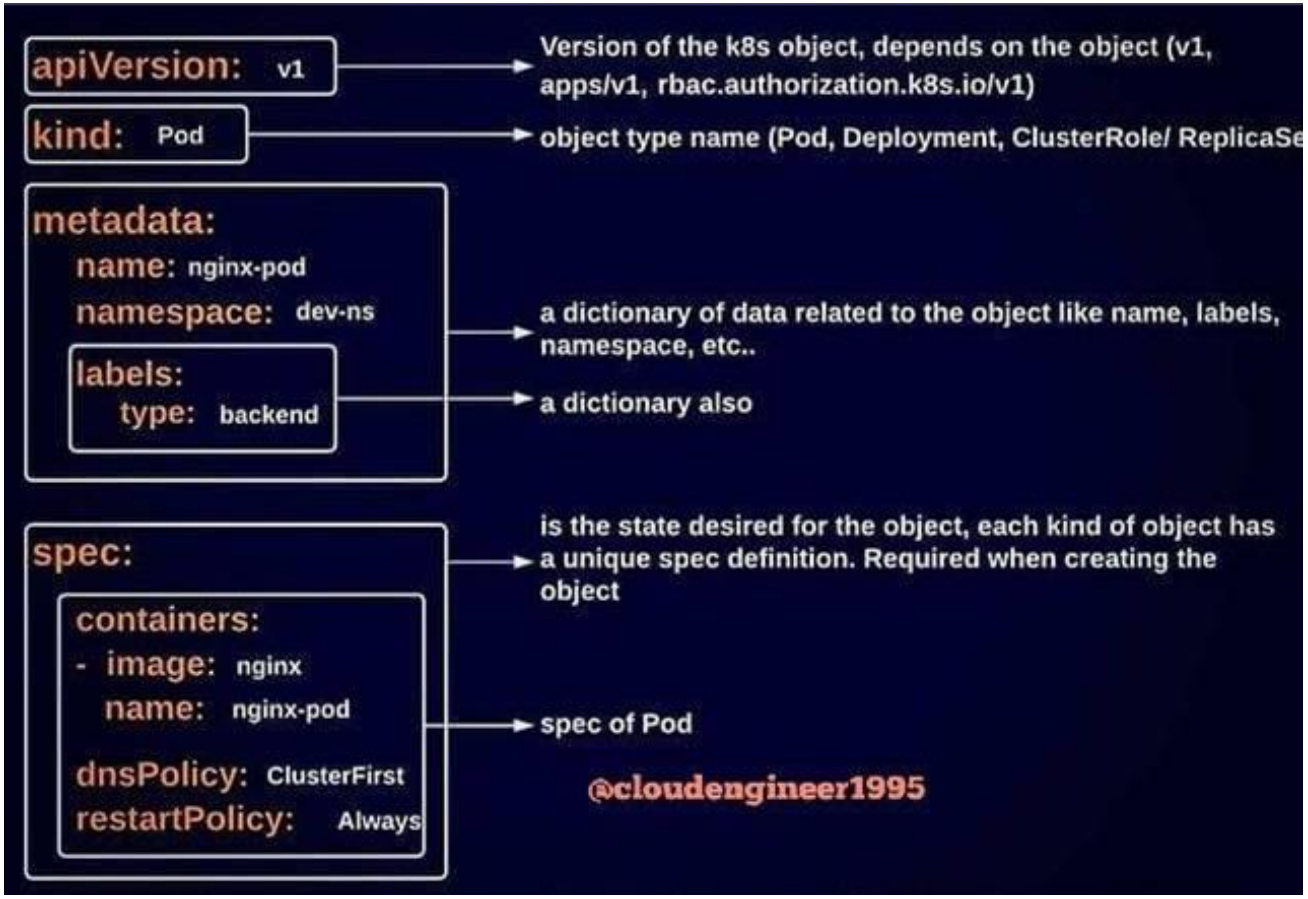
Yaml 파일 사용 예제

Kubernetes 관리도구

- **CLI 형태**
  - **kubectl** (Kubernetes 클라이언트 관리도구)
    - Master에서만 사용하고, k8s에서 관리하는 자원을 생성, 가져오기 및 삭제하는 작업을 수행한다
    - k8s API와 상호작용하기 위한 CLI 명령어 도구이다
    - Pod, ReplicaSet, Service등 대부분의 k8s resource를 관리하는데 사용한다
    - **kubectl get nodes**
    - **kubectl get pods --all-namespaces**
    - **kubectl run mygosmall --image=jesuswithme/gosmall --port 80**
  - **kubeadm**
    - On-premises 환경에서 k8s cluster를 생성 및 관리한다
    - **kubeadm init (##k8s cluster 생성하기)**
    - **kubeadm join (##k8s cluster에 가입하기)**

Kubernetes 관리도구

- **Yaml 형태**
  - **자원 파일**(yaml 또는 json)
    - pod와 replicaset, service와 같은 k8s 자원을 생성하기 위해 kubectl 명령 실행할 때, 전달해야 할 정보를 yaml 및 json 형식의 파일로 제공한다



Kubernetes 관리도구

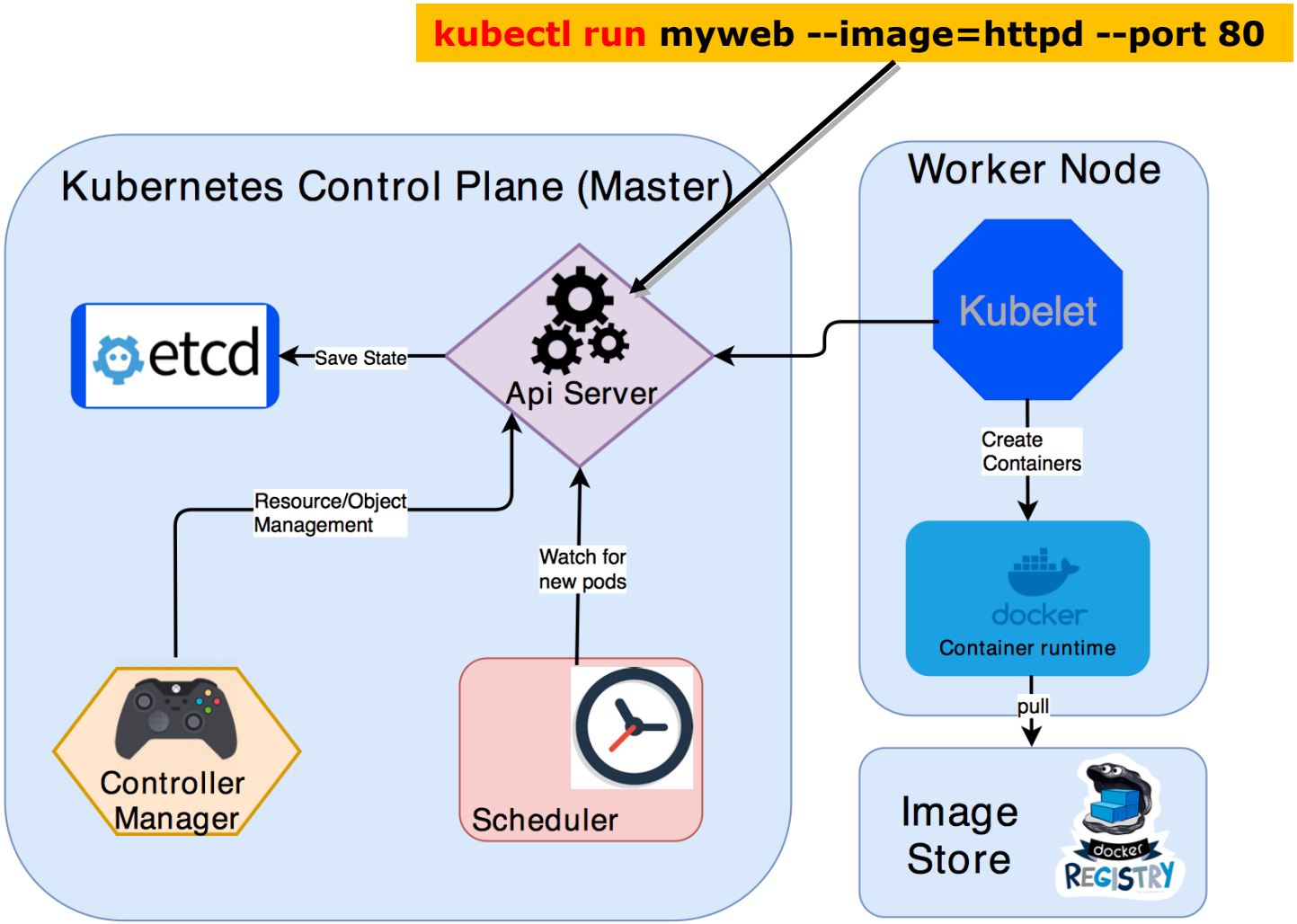
- **Yaml 파일 사용 예제**
  - **kubectl apply -f** yaml파일
    - **wget http://down.cloudshell.kr/k8s/lab/nginx-app.yaml**
    - **kubectl apply -f nginx-app.yaml**
  - **실제 실행하지 않고 테스트만 하기: --dry-run=client 옵션**
    - **kubectl run myhttp --image=nginx:1.14 --port 80 --dry-run=client**
  - **명령 실행을 yaml 파일로 확인하고 저장까지 하기: -o yaml > file.yaml 옵션**
    - **kubectl run myhttp --image=nginx:1.14 --port 80 --dry-run=client -o yaml**
    - **kubectl run myhttp --image=nginx:1.14 --port 80 --dry-run=client -o yaml > webserver-pod.yaml**
    - **kubectl apply -f webserver-pod.yaml**
  - **kubectl get pods -o wide**

k8s 명령어 체계

- kubernetes 명령어 체계
- Pod 생성 및 관리하는 명령어
- Pod을 외부에 노출하기
- Pod 수정하기
- Pod 등 개체 삭제하기
- Kubernetes Cluster 삭제

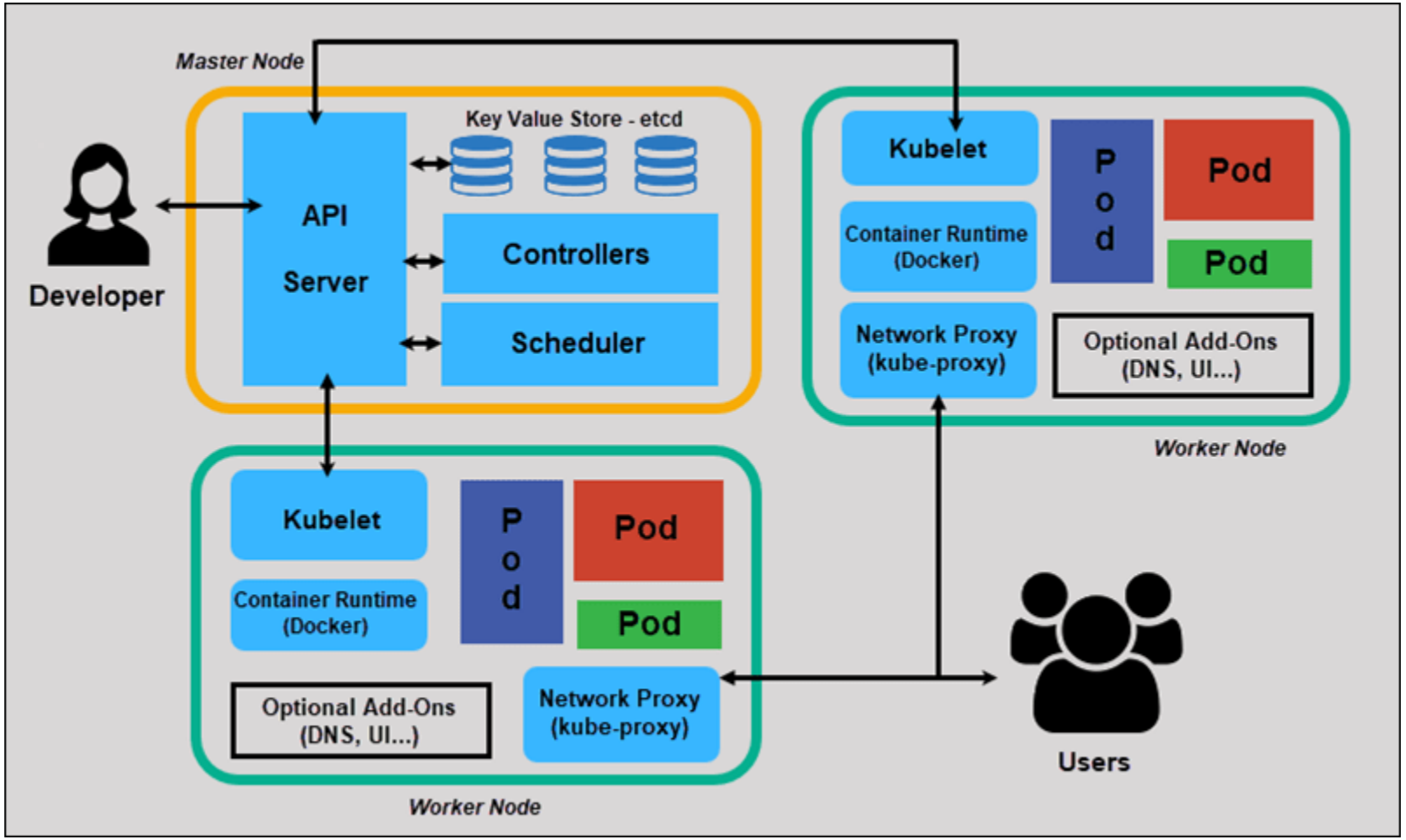
Kubernetes 명령어 체계

- k8s container 동작 순서
  - master에서 관리자 및 개발자의 명령



Kubernetes 명령어 체계

- k8s container 동작 순서
  - 사용자는 workder node로 접속



Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 사용 구문**

BASE	SYSTEM	OPERATION	RESOURCE	OPTIONS
kubectl	-n kube-system	get describe rm (delete) logs -f exec -it apply -f	pods deployment secret ingress node svc (service) ns (namespace) cm (configmap)	oyaml: -o yaml ojson: -o json owide: -o wide all: --all-namespaces w: --watch sl: --show-labels f: -f l: -l

  - 자료 참고: https://Kubernetes.io/docs/reference/kubectl/overview/ **(##강추)**
  - 도움말 이용하기
    - **kubectl --help**

Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 구문**
    - **kubectl** [command] [TYPE] [NAME] [flags] -o <output_format>
    - **command: run, create, apply, get, describe, delete**
    - **TYPE: pods, nodes, deployments, services, replicasets, configmaps 등등**
    - **NAME: resource 이름(대/소문자 구분하므로 정확하게 입력해야 한다)**
      - kubectl get pod **example-pod1 example-pod2**
      - kubectl get **pod/example-pod1 rc/example-rc1**
    - **flags (=options)**
      - 위치에 구애 받지 않는다
        - kubectl **-n kube-system** get pod (=kubectl get pod **-n kube-system**)
      - -i, -t, -n, --all, --server, -f, --image, --replicas, --port
    - **-o <output_format>**
      - **-o wide/yaml/json**
      - **kubectl get** 구문의 제일 마지막에만 사용한다
      - 현 상태를 표시하는 형식이기 때문이다

Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 구문 세부 설명**
    - **command(=operations)**
      - Resource(=object + controller)에 대하여 하고자 하는 작업(operation)
      - **run, create, apply, get, describe, delete**
    - **TYPE**
      - resource type(유형)을 지정한다
      - **단수형태, 복수형태, 축약형태**를 모두 사용할 수 있다
      - kubectl get **pod** pod_name (##단수)
      - kubectl get **pods** pod_name (##복수)
      - kubectl get **po** pod_name (##축약)

Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 구문 세부 설명**
    - **TYPE**
      - resource type의 축약형: **kubectl api-resources** **(##강추)**

전체 이름	축약형
pods	**po**
nodes	**no**
deployments	**deploy**
namespaces	**ns**
services	**svc**
replicasets	**rs**
replicationcontrollers	**rc**
daemonsets	**ds**
configmaps	**cm**
certificatesigningrequests	**csr**
componentstatuses	**cs**

전체 이름	축약형
endpoints	**ep**
events	**ev**
horizontalpodautoscalers	**hpa**
ingresses	**ing**
limitranges	**limits**
persistentvolumeclaims	**pvc**
persistentvolumes	**pv**
poddisruptionbudgets	**pdb**
podsecuritypolicies	**psp**
resourcequotas	**quota**
serviceaccounts	**sa**

Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 구문 세부 설명**
    - **NAME**
      - resource에 대한 이름을 정확히 지정한다(대,소문자 구분한다)
      - resource type과 Name을 정확히 입력한다(Name을 여러 개를 사용할 수 있다)
      - kubectl get **pod** ex-pod1 ex-pod2 ex-pod3
      - kubectl get **pod -n kube-system (##k8s 시스템과 관련된 pod만을 확인)**
      - kubectl **describe pod** coredns-5c98db65d4-792q5 **-n kube-system**
      - 여러 개의 resource type과 name을 같이 사용할 수 있다
        - TYPE1/name1 TYPE1/name2 TYPE2/name3
        - kubectl get **pod/ex-pod1 rs/ex-rs1**
      - 여러 개의 yaml 파일을 사용할 때는 -f 옵션을 사용하면 된다
        - -f file1 -f file2
        - **kubectl apply -f ./pod.yaml -f ./deploy.yaml**
    - flags(=options)
      - 이것은 아무 위치에 놓여도 된다
      - --server flags를 -s 사용해도 된다

Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 사용 예제**
    - kubectl **get node -o wide**
    - kubectl **get namespace**
      - **(= kubectl get ns)**
    - kubectl **-n kube-system** get pod
    - kubectl get pod **--all-namespaces**
    - **kubectl run** myalpine **--image alpine -it (##OS는 -it 사용)**
    - **kubectl run** mynginx **--image nginx --port 80 (##application은 --port 사용)**
    - kubectl run mygosmall **--image** jesuswithme/gosmall **--port 80 --labels** "ver=1,app=alpaca,env=prod"
      - ##**kubectl run** *pod_name* ***--image*** *image_name*
    - kubectl **get pod**
    - kubectl **delete pod --all**

Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 사용 예제**
    - kubectl **-n kube-system get pods**
    - kubectl **get pods --all-namespaces** (##모든 namespace에 실행중인 pod 확인)
    - yaml 파일을 사용하여 k8s object 생성하기
      - **wget** http://down.cloudshell.kr/k8s/ingress-controller-mandatory.yaml
      - **kubectl apply -f ingress-controller-mandatory.yaml**
      - kubectl **get ns** (##ingress-nginx라는 ns가 생성된 것을 확인)

```
root@master:~# kubectl get ns
NAME              STATUS   AGE
default           Active   27h
ingress-nginx     Active   36s
kube-node-lease   Active   27h
```

        - ##새로운 namespace에 pod을 만든 이유는 기존 것과 구별하여 관리하기 위함
      - kubectl **-n ingress-nginx get cm**
      - kubectl **-n ingress-nginx get cm nginx-configuration -o yaml**
      - kubectl **-n ingress-nginx get pods**

Kubernetes 명령어 체계

- Kubernetes 명령어 체계
  - **kubectl 사용 예제**
    - kubectl **delete -f** ingress-controller-mandatory.yaml
      - ##yaml로 생성한 모든 것을 한꺼번에 삭제하기
      - ## yaml 파일에는 여러 개체에 대한 정보가 포함될 수 있다
      - kubectl **create** -f test.yaml (##개체를 생성만 가능)
      - kubectl **apply** -f test.yaml (##개체를 생성하거나 수정하기)
      - kubectl **delete** -f test.yaml (##생성된 개체를 동일하게 삭제하기)

Pod 생성 및 관리하는 명령어

- k8s cluster에 container 생성하기**(Master에서 작업)**
  - **kubectl run** mypod1 **--image busybox** -it
    - ping www.google.com -c 3
    - **exit**
      - 연결된 세션 종료됨(container가 실행중인 상태에서 빠져 나옴)
  - **kubectl run** mypod2 **--image busybox** -it **--restart Never**
    - ## --restart=Never를 사용하면 exit를 하면 pod은 자동으로 중지된다
    - **exit**
  - **kubectl run** mypod3 **--image alpine** -it **--restart Never --rm**
    - **--rm** : exit로 빠져 나오면 자동으로 **mycon3이라는 pod까지 삭제된다**
    - ping www.google.com -c 3
    - **exit**
  - **kubectl get pods**
  - **kubectl describe pod** mypod1 | grep **Containers: -A 1**
    (##container 이름 확인하기)

Pod 생성 및 관리하는 명령어

- k8s cluster에 container 생성하기**(Master에서 작업)**
  - 세션을 끊겼지만 여전히 실행중인 Container에 **다시 접속하기**
    - **kubectl attach** mypod1 -c mypod1 **-it**
      - touch myfile.txt
      - exit
      - ## pod는 계속 실행중이어서 언제든지 container에 접속할 수 있다
    - **kubectl exec** mypod1 **-it -- sh (##pod에 container가 하나만 있어서 가능)**
    - **kubectl exec** mypod1 **-c mypod1 -it -- sh**
      - ##kubectl exec로 기존 container에 접속할 때는 꼭 제일 뒤에 실행하고자 하는 프로그램(/bin/bash, 또는 /bin/sh)을 사용해야 한다
      - ##kubectl exec mypod1 -c mypod1 **ping www.google.com**
      - ##kubectl exec mypod1 -c mypod1 **cat /etc/resolv.conf**
    - ## 실행중인 pod에 접속은 **cluster 내부에서만 일단 가능하다**

Pod 생성 및 관리하는 명령어

- k8s cluster에 container 생성하기**(Master에서 작업)**
  - 실행중인 pod 삭제하기
    - **kubectl get pods**
    - **kubectl delete pods --all** (##pod이름이 복잡하여 아예 모든 것을 삭제함)
    - **kubectl get pods**
      - ## pod가 다른 이름으로 다시 자동으로 생성된다

Pod 생성 및 관리하는 명령어

- pod 관리하기**(Master에서 작업)**
  - **kubectl get pods**
  - 실행중이거나 중지된 pod을 삭제하기
    - **kubectl delete pods** mypod2
    - **kubectl get pods**
  - **docker images | grep busybox**
    - ## busybox 도커 이미지가 master에는 없음
    - ## busybox 도커 이미지는 어떤 node에 다운로드되어 있다

- **Node에서** image와 container 변경 여부 확인하기
  - **docker images**
    - ##**첫번째 node에서만** busybox 도커 이미지가 다운로드 됨
  - **docker ps -a | grep busybox**

Pod 생성 및 관리하는 명령어

- k8s cluster에 특정한 image를 deploy
  - Kubernetes에서 Pod는 관리 또는 네트워크 목적으로 묶여 있는 하나 이상의 컨테이너들의 그룹
  - **kubectl run** 명령어는 Pod과 Container를 생성할 수 있다
    - **kubectl run hello-nodejs --image=snowdeer/hello-nodejs:v1 --port=8080**
    - **curl 10.44.0.3:8080**
  - **Deployment는** Kubernetes에서 **ReplicaSet** 및 **Pod 생성**과 **Scaling**을 위해 추천하는 방식이다
  - Pod 정보 확인하기
    - **kubectl get pods**
    - ##만약 AVAILABLE 항목이나 READY 값이 1이 되지 않는다면, **kubectl describe pods [pod name]** 명령어를 이용해서 에러 원인을 찾아야 한다

Pod 생성 및 관리하는 명령어

- k8s cluster에 특정한 image를 deploy
  - 내부 접속을 위해 IP Address 확인하기
    - **kubectl describe pods** hello-nodejs

```
root@master:~# kubectl describe pods hello-nodejs
Name:           hello-nodejs-678d94cb55-rrfqf
Namespace:      default
Priority:       0
Node:           node2/192.168.219.128
Start Time:     Wed, 26 Jun 2019 08:06:54 +0900
Labels:         pod-template-hash=678d94cb55
                run=hello-nodejs
Annotations:    <none>
Status:         Running
IP:             10.36.0.1
Controlled By:  ReplicaSet/hello-nodejs-678d94cb55
```

      - **cluster IP: 10.36.0.1**
      - **Node IP: 192.168.219.138 (pod가 있는 곳)**
    - 또는 **kubectl get pods -o wide를 통해서 내부 접속용 IP를 알 수 있다**

```
root@master:~# kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE     IP          NODE
hello-nodejs-678d94cb55-rrfqf   1/1     Running   0          3m15s   10.36.0.1   node2
```

  - 실행중인 pod를 cluster 내부에서 접속하기
    - curl http://**10.36.0.1:8080**

```
root@master:~# curl http://10.36.0.1:8080
Hello SnowDeer!root@master:~#
```

```
root@node1:~# curl http://10.36.0.1:8080
Hello SnowDeer!root@node1:~#
```

Pod 생성 및 관리하는 명령어

- k8s cluster에 특정한 image를 deploy
  - 클러스터의 이벤트 로그를 조회하기
    - **kubectl get events**
  - kubectl 설정 정보를 조회하기
    - **kubectl config view**

Pod을 외부에 노출하기

- Service 생성하기
  - 기본적으로 Pod를 생성할 경우, service type이 ClusterIP이기 때문에 **클러스터 내부에서만 접근가능한 내부 IP Address로 설정이 된다.**
    - **kubectl get services --output=wide**
  - 외부에서 접속을 허용하는 것은 Service Type이 NodePort와 LoadBalancer
    - NodeType인 경우는 node-ip:port로 접속한다
    - LoadBalancer인 경우는 Cloud 업체에서 제공하거나 On-Prem도 가능
  - 그래서 위에서 생성한 hello-nodejs라는 Pod를 외부 네트워크에서도 접속할 수 있게 하려면 해당 Pod를 Kubernetes Service로 노출(Expose) 시켜줘야 한다
    - **kubectl expose pod hello-nodejs --type=LoadBalancer**
    - **kubectl get services**
      - --type=LoadBalancer 옵션을 이용해서 해당 Service를 외부 네트워크에 노출할 수 있다.
      - Load Balance 기능을 제공하는 **클라우드 서비스 제공자들이 해당 서비스에 외부 IP Address를 부여할 것임**
      - 이제 웹브라우저를 통해 접속을 해본다.
        위의 **kubectl get services** 명령어 출력 결과를 보면 8080 포트가 31583 포트로 포워딩된 것을 알 수 있다.
        접속 주소를 [IP Address]:31583으로 시도해본다.

Pod을 외부에 노출하기

- k8s cluster에 특정한 image를 **deploy-1**
  - **kubectl run mynginxweb1 --image=nginx --port=80**
    - --port=80: container가 서비스하는 포트 번호
    - **kubectl get pods -o wide**
    - **kubectl describe pods** mynginxweb1 | grep \\IP
    - curl IP:80 (##cluster 내부에서 접속)
  - **kubectl expose pod mynginxweb1 --port=8080 --target-port=80**
    - --port=8080: 노출된 Service의 포트 번호
    - --target-port=80: container가 서비스하는 포트 번호
    - **pod을 expose 한다**
  - **kubectl get service -o wide**

```
root@master:~# kubectl get service -o wide
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE   SELECTOR
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP    16h   <none>
mynginxweb1   ClusterIP   10.106.255.150   <none>        8080/TCP   8s    run=mynginxweb1
```

    - 이 경우에는 외부에서 접속이 안된다. Service Type이 ClusterIP이기 때문이다
    - 외부에서 접속이 되려면 Service Type이 NodePort이거나 LoadBalancer 경우 뿐이다

Pod을 외부에 노출하기

- k8s cluster에 특정한 image를 **deploy-1**
  - **kubectl delete svc mynginxweb1**
  - **kubectl expose pod mynginxweb1 --type=ClusterIP (##추천)**
  - **kubectl get service -o wide**
    - **이 경우에는 외부에서 접속이 안된다. Service Type이 ClusterIP이기 때문이다**
    - **외부에서 접속이 되려면 Service Type이 NodePort이거나 LoadBalancer 경우 뿐이다**

Pod을 외부에 노출하기

- k8s cluster에 특정한 image를 **deploy-2**
  - **kubectl run** mynginxpod **--image=nginx --port=80**
  - kubectl get pods
  - **kubectl expose pod** mynginxpod **--type=NodePort**
  - **kubectl get services -o wide**

```
root@master:~# kubectl get service -o wide
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
kubernetes    ClusterIP   10.96.0.1        <none>        443/TCP
mynginxpod    NodePort    10.109.193.184   <none>        80:31732/TCP
mynginxweb1   ClusterIP   10.106.255.150   <none>        8080/TCP
```

Pod을 외부에 노출하기

- k8s cluster에 특정한 image를 **deploy-2**
  - curl http://10.109.193.184:80 (##cluster 내부 통신)

```
root@master:~# curl http://10.109.193.184:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

  - curl -vk http://node1:31732
  - curl -vk http://192.168.219.128:31732
  - ## 모두 성공(cluster에 참여한 노드들)

```
root@master:~# curl node1:31732
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

Pod을 외부에 노출하기

- Deployment로 생성된 Pod을 scale 명령으로 Pod 확장하기
  - **kubectl create deployment** mynginxdeploy1 **--image nginx**
    - **ReplicaSet을 1개만 만듦**
    - **kubectl get all**
  - **kubectl scale deployment** mynginxdeploy1 **--replicas 4**
    - **Deployment만을 만들 때 ReplicaSet이 생성되기 때문에 pod의 scale out/in이 가능하다**
    - **##Pod만 생성할 때는 나중에 Pod을 scale out/in을 못한다**
  - **kubectl create svc nodeport mynginxdeploy1 --tcp 80:80**
  - **kubectl create deployment** mynginxdeploy2 **--image nginx --replicas=2**
    - **Deployment 생성할 때 아예 ReplicaSet을 2개를 생성함**
    - **kubectl get all**

Pod을 외부에 노출하기

- Deployment로 생성된 Pod을 scale 명령으로 Pod 확장하기
  - **kubectl create deployment mygosmall1 --image jesuswithme/gosmall**
  - **kubectl scale deployment mygosmall1 --replicas 4**
  - **kubectl create svc nodeport mygosmall1 --tcp 80:80**

```
[root@master ~]# kubectl get pod -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP           NODE
mygosmall1-54c589795c-fvpb8   1/1     Running   0          7s    10.36.0.4    node2
mygosmall1-54c589795c-j9r8c   1/1     Running   0          7s    10.44.0.2    node1
mygosmall1-54c589795c-msvt7   1/1     Running   0          7s    10.44.0.1    node1
mygosmall1-54c589795c-tl2jb   1/1     Running   0          7s    10.36.0.1    node2
[root@master ~]# kubectl get svc -o wide
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE   SELECTOR
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        22s   <none>
mygosmall1   NodePort    10.107.95.152   <none>        80:32275/TCP   15s   app=mygos
[root@master ~]# curl node1:32275
var id =  98938 ;
[root@master ~]# curl node2:32275
var id =  98938 ;
[root@master ~]# curl master:32275
var id =  287232 ;
```

Pod을 외부에 노출하기

- k8s cluster에 특정한 image를 **deploy-3**
  - **kubectl run mygosmall --image=jesuswithme/gosmall** --port=80
  - **kubectl expose pod mygosmall --type=NodePort**
    - kubectl get **services -o wide**

```
root@master:~# kubectl get services -o wide
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP
mygosmall    NodePort    10.99.106.1    <none>        80:31742/TCP
```

  - curl http://**node2**:31742
  - curl http://**node1**:31742
    - ##**이 노드에는 pod이 없는데도** pod에 접근된다

```
root@master:~# curl http://node2:31742
var id =  726303 ;
root@master:~# curl http://node1:31742
var id =  328068 ;
```

Pod 수정하기

- k8s cluster의 **pods, deployments, services 수정하기**
  - **kubectl get pod -o wide**
  - **kubectl edit *object*** *object_name*
  - **kubectl edit svc mygosmall1**
    - **##노드포트 번호를 30111로 수정하고 저장하고 빠져나온다**
  - **kubectl get svc -o wide**
    - **## 노드포트 번호가 수정된 것을 알 수 있다**
  - **kubectl edit deployment mygosmall1**
    - **##replicas를 2로 수정하고 저장하고 빠져나온다**
  - **kubectl get pod -o wide**
    - **##pod이 2개로 줄어들었다는 것을 알 수 있다**

Pod 등 개체 삭제하기

- k8s cluster의 pods, deployments, services **삭제하기**
  - **kubectl get all**

  - **kubectl delete services mynginxpod**
  - **kubectl delete deployments --all**
  - **kubectl delete pods --all**

Kubernetes Cluster 삭제

- Kubernetes Cluster 삭제하기
  - Node를 Cluster에서 탈퇴시키기(Node 제거하기)
    - **kubectl get nodes -o wide**

```
root@master:~# kubectl get nodes -o wide
NAME      STATUS   ROLES    AGE   VERSION   INTERNAL-IP
master    Ready    master   17h   v1.15.0   192.168.219.130
node1     Ready    <none>   16h   v1.15.0   192.168.219.152
node2     Ready    <none>   15h   v1.15.0   192.168.219.128
node3     Ready    <none>   15h   v1.15.0   192.168.219.136
```

    - **kubectl drain** <node name> **--delete-local-data --force --ignore-daemonsets**
    - node3를 제거해 본다

```
root@master:~# kubectl drain node3 --delete-local-data --force --ignore-daemonsets
node/node3 cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/kube-proxy-qvfmg, kube-system/weave-net-kvtkm
evicting pod "mygosmall-7d6bf9f994-nt492"
evicting pod "mygosmall-7d6bf9f994-cb8fm"
pod/mygosmall-7d6bf9f994-nt492 evicted
pod/mygosmall-7d6bf9f994-cb8fm evicted
node/node3 evicted
root@master:~#
```

Kubernetes Cluster 삭제

- Kubernetes Cluster 삭제하기
  - Node를 Cluster에서 탈퇴시키기(Node 제거하기)
    - **kubectl get nodes**

```
root@master:~# kubectl get nodes
NAME      STATUS                     ROLES
master    Ready                      master
node1     Ready                      <none>
node2     Ready                      <none>
node3     Ready,SchedulingDisabled   <none>
```

    - **다시 원상 복귀하려면,,,,,**
      - **kubectl uncordon** <node name>

```
root@master:~# kubectl uncordon node3
node/node3 uncordoned
```

      - **kubectl get nodes**

```
root@master:~# kubectl get nodes
NAME      STATUS   ROLES    AGE   VERSION
master    Ready    master   17h   v1.15.0
node1     Ready    <none>   16h   v1.15.0
node2     Ready    <none>   16h   v1.15.0
node3     Ready    <none>   16h   v1.15.0
```

Kubernetes Cluster 삭제

- Kubernetes Cluster 삭제하기
  - Node를 Cluster에서 탈퇴시키기(Node 제거하기)-**Master에서만 작업**
    - **kubectl drain node3 --delete-local-data --force --ignore-daemonsets**
    - **kubectl delete node** <node name>
    - **kubectl get nodes**

```
root@master:~# kubectl delete node node3
node "node3" deleted
root@master:~# kubectl get nodes
NAME      STATUS   ROLES    AGE   VERSION
master    Ready    master   17h   v1.15.0
node1     Ready    <none>   16h   v1.15.0
node2     Ready    <none>   16h   v1.15.0
root@master:~#
```

  - Cluster를 Reset하기(Cluster 삭제하기)-**Master와 Node에서 모두 작업**
    - **sudo kubeadm reset**

```
root@master:~# kubeadm reset
[reset] Reading configuration from the cluster...
[reset] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]:
```

Kubernetes Cluster 삭제

- Kubernetes Cluster 삭제하기
  - **Master에서** kubelet service가 중지되었는지 확인하기
    - **systemctl list-unit-files | grep kube**
    - **systemctl status kubelet**
      - ## Kubernetes가 중지된 것을 알 수 있다
  - **Master에서** Kubernetes 삭제하기
    - **yum remove** -y **kubelet kubeadm kubectl kubernetes-cni**
  - **Master에서** 다운로드된 docker image 삭제하기
    - docker images
    - **docker rmi** $(docker images -aq) -f
    - docker images

Kubernetes Cluster 삭제

- Kubernetes Cluster 삭제하기
  - **모든 Node에서** kubelet service가 중지되었는지 확인하기
    - **systemctl list-unit-files | grep kube**
    - **systemctl status kubelet**
      - ## Kubernetes가 중지된 것을 알 수 있다
  - **모든 Node에서** Kubernetes 삭제하기
    - yum remove -y **kubelet kubeadm**
  - **모든 Node에서** 다운로드된 docker image 삭제하기
    - docker images
    - **docker rmi** $(docker images -aq) -f
    - docker images

  - 모든 Node에서 "**sudo kubeadm reset" 명령어를 실행하지 않은 경우에만** 아래 작업을 진행한다 (옵션 사항)
    - **systemctl stop kubelet**
    - **sudo rm -rf /etc/kubernetes/***

YAML Template

YAML이란?

YAML 기본 문법

Field 및 Field의 세부 사항

YAML이란?

- **YAML이란?**
  - YAML이라는 이름은 **"YAML은 마크업 언어가 아니다 (YAML Ain't Markup Language)"** 라는 재귀적인 이름에서 유래되었다

```
# What does YAML mean?
YAML:
    - Y: YAML
    - A: Ain't
    - M: Markup
    - L: Language
```

  - 원래 YAML의 뜻은 **"또 다른 마크업 언어 (Yet Another Markup Language)"**였으나, YAML의 핵심은 문서 마크업이 아닌 **데이터 중심**에 있다는 것을 보여주기 위해 이름을 바꾸었다
  - 사림이 쉽게 읽을 수 있는 데이터 직렬화 양식을 갖고 있다
    - XML과 JSON이 데이터 직렬화에 주로 쓰이기 시작하면서, 많은 사람들이 YAML을 '가벼운 마크업 언어'로 사용하려 하고 있다

YAML 기본 문법

- **YAML 기본 문법**
  - 구조화된 데이터를 표현하기 위한 데이터 형식을 갖고 있다
  - Python처럼 들여쓰고 데이터의 계층을 표현한다
  - 들여쓰기는 Tab을 사용하지 않고 Space Bar를 사용한다
  - 가독성이 좋아서 설정 파일에 적합하다
  - #을 사용하여 주석 처리하고 여러 줄의 주석은 지원하지 않는다
  - "---"은 성격이 다른 YAML 형식의 문서 여러 개가 있을 때 구분자로 사용한다
  - Scalar 문법
    - 문자열이나 숫자를 표현하며 **":"**을 기준으로 "**key: value**"를 설정한다
  - Array 문법
    - Array 또는 List를 표현하며, **"-"** 문자로 **여러 개**를 나열한다

YAML 기본 문법

- **YAML 기본 문법**
  - 기본 형식: **Field**
    - **apiVersion:**
    - **kind:**
    - **metadata:**
    - **spec:**

```
[root@master ~]# cat pod-nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
   - image: nginx:1.14
     name: nginx
     ports:
     - containerPort: 80
     - containerPort: 443
```

  - YAML 구문의 유효성 확인
    - http://www.yamllint.com/

Field 및 Field의 세부 사항

- **Field 및 Field의 세부 사항**
  - 현재 Cluster에서 사용 가능한 API 버전 확인하기
    - **kubectl api-versions**
  - Object 및 Controller의 사용 가능한 Field가 어떤 것들이 있는지 확인하기
    - **kubectl explain pods**
    - **kubectl explain service**
    - **kubectl explain deployment**
    - **kubectl explain statefulset**
  - 각 Object 및 Controller의 **현재 사용하는 버전 확인하기** (##강추)
    - **kubectl explain pods | grep VERSION**
    - **kubectl explain deployment | grep VERSION**

```
[root@master ~]# kubectl explain pods | grep VERSION
VERSION:  v1
[root@master ~]# kubectl explain deployment | grep VERSION
VERSION:  apps/v1
```

object의 yaml 파일을 생성할 때 apiVersion이 다르면 오류가 발생하니 주의한다

Field 및 Field의 세부 사항

- **Field 및 Field의 세부 사항**
  - 각 Field의 하위 Field가 어떤 것이 있고 **상세한 설명 확인하기(metadata, spec)**
    - **kubectl explain pods.metadata**
    - **kubectl explain pods.spec**
    - **kubectl explain deployment.metadata**
    - **kubectl explain deployment.spec**

  - **Field의 설명 없이** 특정 Field와 그 아래에 속한 모든 하위 Field를 한꺼번에 보기
    - **kubectl explain pods --recursive**
    - **kubectl explain namespace --recursive**
    - --recursive를 사용하여 metadta, spce의 하위 필드를 알 수 있다

```
FIELDS:
   apiVersion   <string>
   kind <string>
   metadata     <Object>
      annotations       <map[string]string>
      clusterName       <string>
      creationTimestamp <string>
      deletionGracePeriodSeconds        <integer>
      deletionTimestamp <string>
      finalizers        <[]string>
      generateName      <string>
      generation        <integer>
      labels    <map[string]string>
      managedFields     <[]Object>
         apiVersion     <string>
         fieldsType     <string>
         fieldsV1       <map[string]>
         manager        <string>
         operation      <string>
         time   <string>
      name      <string>
```

Field 및 Field의 세부 사항

- **Field 및 Field의 세부 사항**
  - 각 Field의 하위 Field가 어떤 것이 있고 **상세한 설명 확인하기(metadata, spec)**
    - **kubectl api-resources**
    - **kubectl explain rs**
    - **kubectl explain rs.spec**
    - **kubectl explain rs.spec.selector**

```
[root@master ~]# kubectl explain rs.spec.selector
KIND:     ReplicaSet
VERSION:  apps/v1

RESOURCE: selector <Object>

DESCRIPTION:
     Selector is a label query over pods that should match the replica count.
     Label keys and values that must match in order to be controlled by this
     replica set. It must match the pod template's labels. More info:
     https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors

     A label selector is a label query over a set of resources. The result of
     matchLabels and matchExpressions are ANDed. An empty label selector matches
     all objects. A null label selector matches no objects.

FIELDS:
   matchExpressions     <[]Object>
     matchExpressions is a list of label selector requirements. The requirements
     are ANDed.

   matchLabels  <map[string]string>
     matchLabels is a map of {key,value} pairs. A single {key,value} in the
     matchLabels map is equivalent to an element of matchExpressions, whose key
     field is "key", the operator is "In", and the values array contains only
```

- **kubectl explain pod.spec.volumes**

```
[root@master ~]# kubectl explain pod.spec.volumes | more
KIND:     Pod
VERSION:  v1

RESOURCE: volumes <[]Object>

DESCRIPTION:
     List of volumes that can be mounted by containers belonging to the pod.
     More info: https://kubernetes.io/docs/concepts/storage/volumes

     Volume represents a named volume in a pod that may be accessed by any
     container in the pod.

FIELDS:
   awsElasticBlockStore <Object>
     AWSElasticBlockStore represents an AWS Disk resource that is attached to a
     kubelet's host machine and then exposed to the pod. More info:
     https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore

   azureDisk    <Object>
     AzureDisk represents an Azure Data Disk mount on the host and bind mount to
     the pod.

   azureFile    <Object>
     AzureFile represents an Azure File Service mount on the host and bind mount
```

Field 및 Field의 세부 사항

- **Field 및 Field의 세부 사항**
  - **하위 Field의 상세 설명과 또 그 아래 Field 확인하기**
    - **kubectl api-resources**
    - **kubectl explain rs --recursive**
    - **kubectl explain rs.metadata --recursive**
    - **kubectl explain rs.metadata.managedFields --recursive**

```
[root@master ~]# kubectl explain rs.metadata.managedFields --recursive
KIND:     ReplicaSet
VERSION:  apps/v1

RESOURCE: managedFields <[]Object>

DESCRIPTION:
     ManagedFields maps workflow-id and version to the set of fields that are
     managed by that workflow. This is mostly for internal housekeeping, and
     users typically shouldn't need to set or understand this field. A workflow
     can be the user's name, a controller's name, or the name of a specific
     apply path like "ci-cd". The set of fields is always in the version that
     the workflow used when modifying the object.

     ManagedFieldsEntry is a workflow-id, a FieldSet and the group version of
     the resource that the fieldset applies to.

FIELDS:
   apiVersion   <string>
   fieldsType   <string>
   fieldsV1     <map[string]>
   manager      <string>
   operation    <string>
   time <string>
```

- **kubectl explain rs.metadata.managedFields.manager --recursive**

쉬어가는 코너

- ImagePullBackOff 상태
- kubernetes 자원 종류 확인하기
- Pod을 Expose할 때 주의사항

ImagePullBackOff 상태

- **ImagePullBackOff 상태**
  - Pod을 만들라고 명령을 내린 후 kubectl get pod을 하면 STATUS에 **ContainerCreating ➔ ErrImagePull ➔ ImagePullBackOff** 로 나타나는 경우가 있다

```
[root@master ~]# kubectl get pod
NAME              READY   STATUS
hostpath-nginx    0/1     ContainerCreating
hostpath-redis    0/1     ContainerCreating
shared-volumes    2/2     Running
[root@master ~]# kubectl get pod
NAME              READY   STATUS             REST
hostpath-nginx    0/1     ErrImagePull       0
hostpath-redis    1/1     Running            0
shared-volumes    2/2     Running            0
```

```
[root@master ~]# kubectl get pod -o wide
NAME              READY   STATUS
 GATES
hostpath          0/1     ImagePullBackOff
shared-volumes    2/2     Running
```

  - 여기서 **ImagePullBackOff**란 **k8s가 컨테이너 이미지를 다운로드하지 못했기 때문에 Pod이 시작하지 못했다**는 뜻이다
  - 여기서 **BackOff**란 k8s가 계속해서 이미지를 다운로드하려고 시도하는데 back-off delay 기간을 증가시키면서 다운로드를 계속 시도한다는 것이다
  - 나중에 해당 Pod이 정상적으로 Running이 된 후에 **kubectl describe pod** *pod_name*으로 접속하면 상세한 과정을 알 수가 았다
  - 네트워크 상태가 좋지 못하거나 image가 저장된 장소에 원하는 이미지가 없는 경우에 이런 현상이 나타난다

Kubernetes 자원 종류 확인하기

- **Kubernetes 자원 종류 확인하기**
  - **kubectl api-resources (##강추)**

```
[root@master ~]# kubectl api-resources
NAME                              SHORTNAMES   APIVERSION   NAMESPACED   KIND
bindings                                       v1           true         Binding
componentstatuses                 cs           v1           false        ComponentStatus
configmaps                        cm           v1           true         ConfigMap
endpoints                         ep           v1           true         Endpoints
events                            ev           v1           true         Event
limitranges                       limits       v1           true         LimitRange
namespaces                        ns           v1           false        Namespace
nodes                             no           v1           false        Node
persistentvolumeclaims            pvc          v1           true         PersistentVolumeClaim
persistentvolumes                 pv           v1           false        PersistentVolume
pods                              po           v1           true         Pod
podtemplates                                   v1           true         PodTemplate
replicationcontrollers            rc           v1           true         ReplicationController
resourcequotas                    quota        v1           true         ResourceQuota
secrets                                        v1           true         Secret
serviceaccounts                   sa           v1           true         ServiceAccount
services                          svc          v1           true         Service
```

  - 종류(Kind)를 알 수 있고, 그것들이 특정한 namespace에 속하는지도 알 수 있고, 축약어 및 version을 정확히 알 수 있다

Pod을 Expose할 때 주의사항

- **Deployment 및 Pod을 Expose하여 접속하는 방법**
  - **Deployment 생성 및 Service 생성하기**
    - k create deployment poly1 --image httpd --replicas 2
    - k create service loadbalancer poly1 --tcp 80:80
  - **Pod 생성 및 Service 생성하기**
    - k run poly2 --image httpd --port 80
    - k create service loadbalancer poly2 --tcp 80:80
    - (##접속할 때 문제 발생... k edit svc poly4를 실행하여 selector를 수정하면 문제 해결됨)
  - **Deployment 생성 및 노출하기(Service 생성됨)**
    - k create deployment poly3 --image httpd --replicas 2
    - k expose deployment poly3 --type LoadBalancer --port 80
  - **Pod 생성 및 노출하기(Service 생성됨)**
    - k run poly4 --image httpd --port 80
    - k expose pod poly4 --type LoadBalancer --port 80
-