02. Kubernetes Object Model

김호진

Exported on Jun 11, 2020

Table of Contents

1 01.pod 4

1.1 Pod phase 6

2 02.Labels 8

3 03.ReplicationController / ReplicaSets 10

3.1 ReplicationController 10

3.2 ReplicaSets 10

3.3 Deployments 10

4 04.namespace 12

5 05. exercise 13

For Example

|  |
| --- |
| apiVersion: apps/v1 # the API endpoint on the API server  kind: Deployment # Deployment, Pod, Replicaset, Namespace, Service  metadata:  name: nginx-deployment #name, labels, namespace  labels:  app: nginx  spec: #the desired state of the Deployment object  replicas: 3  selector:  matchLabels:  app: nginx  template: # using the Pods Template defined in spec.template  metadata:  labels:  app: nginx  spec: # desired state of the Pod  containers:  - name: nginx  image: nginx:1.15.11  ports:  - containerPort: 80 |

# 01.pod

* Kubernetes에서의 배포 단위로 응용 프로그램의 단일 인스턴스를 나타냅니다. 포드는 하나 이상의 컨테이너로 구성된 논리적 컬렉션입니다.
  + 포드와 동일한 호스트에 함께 예약됩니다.
  + 동일한 네트워크 네임 스페이스 공유
  + 동일한 외부 저장소 (볼륨)를 마운트 포인트를 액세스

Figure 1 A pod is a collection of one or more containers

* 포드는 본질적으로 일시적(ephemeral)이며 스스로 치유 (self-heal )할 수있는 능력이 없습니다.

|  |
| --- |
| apiVersion: v1 # object definition  kind: Pod # object type  metadata: # object's name and label  name: nginx-pod  labels:  app: nginx  spec: #the block defining the desired state of the Pod object  containers:  - name: nginx  image: nginx:1.15.11  ports:  - containerPort: 80 |

-o yaml

|  |
| --- |
| [root@idc02 exercise]# kubectl get pod nginx-pod -o yaml  apiVersion: v1  kind: Pod  metadata:  annotations:  cni.projectcalico.org/podIP: 192.168.122.134/32  creationTimestamp: "2019-08-05T22:24:06Z"  labels:  app: nginx  name: nginx-pod  namespace: default  resourceVersion: "32419"  selfLink: /api/v1/namespaces/default/pods/nginx-pod  uid: ddd5d16f-b728-43b1-8339-482924ded42e  spec:  containers:  - image: nginx:1.15.11  imagePullPolicy: IfNotPresent  name: nginx  ports:  - containerPort: 80  protocol: TCP  resources: {}  terminationMessagePath: /dev/termination-log  terminationMessagePolicy: File  volumeMounts:  - mountPath: /var/run/secrets/kubernetes.io/serviceaccount  name: default-token-m9wgk  readOnly: true  dnsPolicy: ClusterFirst  enableServiceLinks: true  nodeName: k8s-n-2  priority: 0  restartPolicy: Always  schedulerName: default-scheduler  securityContext: {}  serviceAccount: default  serviceAccountName: default  terminationGracePeriodSeconds: 30  tolerations:  - effect: NoExecute  key: node.kubernetes.io/not-ready  operator: Exists  tolerationSeconds: 300  - effect: NoExecute  key: node.kubernetes.io/unreachable  operator: Exists  tolerationSeconds: 300  volumes:  - name: default-token-m9wgk  secret:  defaultMode: 420  secretName: default-token-m9wgk  status:  conditions:  - lastProbeTime: null  lastTransitionTime: "2019-08-05T22:24:06Z"  status: "True"  type: Initialized  - lastProbeTime: null  lastTransitionTime: "2019-08-05T22:24:24Z"  status: "True"  type: Ready  - lastProbeTime: null  lastTransitionTime: "2019-08-05T22:24:24Z"  status: "True"  type: ContainersReady  - lastProbeTime: null  lastTransitionTime: "2019-08-05T22:24:06Z"  status: "True"  type: PodScheduled  containerStatuses:  - containerID: docker://fc69cf847f058c959db4c03b2d780585fd36b594b98fcc2aaf4d50c33485d70f  image: nginx:1.15.11  imageID: docker-pullable://nginx@sha256:50174b19828157e94f8273e3991026dc7854ec7dd2bbb33e7d3bd91f0a4b333d  lastState: {}  name: nginx  ready: true  restartCount: 0  state:  running:  startedAt: "2019-08-05T22:24:23Z"  hostIP: 192.168.50.13  phase: Running  podIP: 192.168.122.134  qosClass: BestEffort  startTime: "2019-08-05T22:24:06Z" |

logs nginx-pod -c nginx

|  |
| --- |
| [root@idc02 exercise]# kubectl logs nginx-pod  [root@idc02 exercise]# kubectl logs nginx-pod -c nginx |

## Pod phase

Pod는 아래와 같은 단계로 생성->실행->종료/에러 등의 단계로 구분 된다.

이러한 단계는 Pod의 status필드에 오브젝트로 정의 되어 있다.

|  |  |
| --- | --- |
| **값** | **의미** |
| Pending | 파드가 쿠버네티스 시스템에 의해서 승인되었지만, 파드를 위한 하나 또는 하나 이상의 컨테이너 이미지 생성이 아직 완료되지 않았다. 여기에는 스케줄되기 이전까지의 시간 뿐만 아니라 오래 걸릴 수 있는 네트워크를 통한 이미지 다운로드 시간도 포함된다. |
| Running | 파드가 한 노드에 결합되었고, 모든 컨테이너들의 생성이 완료되었다. 적어도 하나의 컨테이너가 동작 중이거나, 시작 또는 재시작 중에 있다. |
| Succeeded | 파드에 있는 모든 컨테이너들이 성공으로 종료되었고, 재시작되지 않을 것이다. |
| Failed | 파드에 있는 모든 컨테이너들이 종료되었고, 적어도 하나 이상의 컨테이너가 실패로 종료되었다. 즉, 해당 컨테이너는 non-zero 상태로 빠져나왔거나(exited) 시스템에 의해서 종료(terminated)되었다. |
| Unknown | 어떤 이유에 의해서 파드의 상태를 얻을 수 없다. 일반적으로 파드 호스트와의 통신 오류에 의해서 발생한다. |
| Completed | 완료된 잡(Job)과 같이 다 실행되어서 더 작동하고 있을 필요 없이 완료된 상태가 되었다. |
| CrashLoopBackOff | 파드 내 컨테이너 중 하나가 예상과는 달리 종료(exit)되었고, [restart policy](https://kubernetes.io/ko/docs/concepts/workloads/pods/pod-lifecycle/#restart-policy)에 따라 재시작된 뒤에도 0이 아닌 에러 코드가 발생했을 것이다. |

|  |
| --- |
|  |

# 02.Labels

Kubernetes 객체에 연결된 키 - 값 쌍입니다

* 왼쪽 하단의 두 개의 라벨 ( **app = frontend** 및 **env = qa)** 을 선택하여 같이 행동하도록 하고자 하기 때문에, 보통 name이나 id보다는 이 라벨을 사용하게 된다.
* **app=frontend** and **env=qa  등 여러조합으로 사용가능함.**

**Equality-Based Selectors**

* **env==dev or env=dev**

**Set-Based Selectors**

* **env in (dev,!qa)**

# 03.ReplicationController / ReplicaSets

## ReplicationController

The default controller is a [Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) which configures a [ReplicaSet](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/) to manage Pods' lifecycle.

## ReplicaSets

[ReplicaSet는](https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/)  차세대 ReplicationController입니다. ReplicaSets는 동등 및 선택 기반 선택기를 모두 지원하지만 ReplicationController는 동등기반 선택기 만 지원합니다

A Deployment automatically creates a ReplicaSet, which then creates a Pod. There is no need to manage ReplicaSets and Pods separately, the Deployment will manage them on our behalf.

ReplicaSets 및 포드는 별도로 관리 할 필요가 없으며 Deployment가 이를 대신하여 관리합니다.

## Deployments

[Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) objects provide declarative updates to Pods and ReplicaSets

a new **Deployment** creates **ReplicaSet** **A** which then creates **3 Pods**,

This particular state is recorded as **Revision 1**. ==> a Deployment**rolling update ==>** Deployment, Revision 2

일단 **ReplicaSet B**  와 **3 개 포드는**  버전 **1.9.1은** 준비의 **배포** 적극적으로 관리를 시작합니다. 그러나 Deployment는 Deployment의 **롤백** 기능 에서 중요한 요소 인 Revision으로 저장된 이전 구성 상태를 유지합니다

==> change the Pods' Template==>

# 04.namespace

**kube-system**,**kube-public**,**kube-node-lease**, **default**

|  |
| --- |
| hojinuicBookPro:~ hojinkim$ kubectl get namespaces  NAME STATUS AGE  default Active 14h ==> 관리자와 개발자에 의해 생성 된 객체와 리소스가 포함  kube-node-lease Active 14h ==> 마스터 노드가 문제가 생겼거나(네트워크) 등의 문제를 해결하기 위해서 새로 포함된 것으로, 기존의 NodeStatus에 nodelease가 추가되어 둘다 heartbeat역할을 하며, 경량으로 구현되었고, 더 자주 개신된다.  kube-public Active 14h ==> 클러스터에 대한 보안이 필요업는 공개 네임 스페이스  kube-system Active 14h ==> Kubernetes 시스템, 주로 컨트롤 플레인 에이전트에 의해 생성 된 개체를 포함 |

|  |
| --- |
| **What about kube-node-lease?**   * Starting with Kubernetes 1.14, there is a kube-node-lease namespace (or in Kubernetes 1.13 if the NodeLease feature gate is enabled) * That namespace contains one Lease object per node * *Node leases* are a new way to implement node heartbeats (i.e. node regularly pinging the control plane to say "I'm alive!") |

# 05. exercise

|  |
| --- |
| [root@idc01 ~]# kubectl create deployment mynginx --image=nginx:1.15-alpine  deployment.apps/mynginx created  [root@idc01 ~]# kubectl get deploy,rs,po -l app=mynginx  NAME READY UP-TO-DATE AVAILABLE AGE  deployment.extensions/mynginx 1/1 1 1 34s    NAME DESIRED CURRENT READY AGE  replicaset.extensions/mynginx-86f7d8cfcc 1 1 1 34s    NAME READY STATUS RESTARTS AGE  pod/mynginx-86f7d8cfcc-5zq2m 1/1 Running 0 34s    [root@idc01 ~]# kubectl scale deploy mynginx --replicas=3  deployment.extensions/mynginx scaled    [root@idc01 ~]# kubectl get deploy,rs,po -l app=mynginx  NAME READY UP-TO-DATE AVAILABLE AGE  deployment.extensions/mynginx 3/3 3 3 3m26s    NAME DESIRED CURRENT READY AGE  replicaset.extensions/mynginx-86f7d8cfcc 3 3 3 3m26s    NAME READY STATUS RESTARTS AGE  pod/mynginx-86f7d8cfcc-5zq2m 1/1 Running 0 3m26s  pod/mynginx-86f7d8cfcc-tqlx6 1/1 Running 0 27s  pod/mynginx-86f7d8cfcc-vc9qc 1/1 Running 0 27s    [root@idc01 ~]# kubectl describe deploy  Name: mynginx  Namespace: default  CreationTimestamp: Fri, 02 Aug 2019 23:29:31 -0400  Labels: app=mynginx  Annotations: deployment.kubernetes.io/revision: 1  Selector: app=mynginx  Replicas: 3 desired | 3 updated | 3 total | 3 available | 0 unavailable  StrategyType: RollingUpdate  MinReadySeconds: 0  RollingUpdateStrategy: 25% max unavailable, 25% max surge  Pod Template:  Labels: app=mynginx  Containers:  nginx:  Image: nginx:1.15-alpine  Port: <none>  Host Port: <none>  Environment: <none>  Mounts: <none>  Volumes: <none>  Conditions:  Type Status Reason  ---- ------ ------  Progressing True NewReplicaSetAvailable  Available True MinimumReplicasAvailable  OldReplicaSets: <none>  NewReplicaSet: mynginx-86f7d8cfcc (3/3 replicas created)  Events:  Type Reason Age From Message  ---- ------ ---- ---- -------  Normal ScalingReplicaSet 4m45s deployment-controller Scaled up replica set mynginx-86f7d8cfcc to 1  Normal ScalingReplicaSet 106s deployment-controller Scaled up replica set mynginx-86f7d8cfcc to 3    [root@idc01 ~]# kubectl rollout history deploy  deployment.extensions/mynginx  REVISION CHANGE-CAUSE  1 <none>    [root@idc01 ~]# kubectl rollout history deploy mynginx --revision=1  deployment.extensions/mynginx with revision #1  Pod Template:  Labels: app=mynginx  pod-template-hash=86f7d8cfcc  Containers:  nginx:  Image: nginx:1.15-alpine  Port: <none>  Host Port: <none>  Environment: <none>  Mounts: <none>  Volumes: <none>    [root@idc01 ~]# kubectl set image deploy mynginx nginx=nginx:1.16-alpine  deployment.extensions/mynginx image updated    [root@idc01 ~]# kubectl rollout history deploy  deployment.extensions/mynginx  REVISION CHANGE-CAUSE  1 <none>  2 <none>    [root@idc01 ~]# kubectl rollout history deploy mynginx  deployment.extensions/mynginx  REVISION CHANGE-CAUSE  1 <none>  2 <none>    [root@idc01 ~]# kubectl rollout history deploy mynginx --revision=2  deployment.extensions/mynginx with revision #2  Pod Template:  Labels: app=mynginx  pod-template-hash=5d7f998444  Containers:  nginx:  Image: nginx:1.16-alpine  Port: <none>  Host Port: <none>  Environment: <none>  Mounts: <none>  Volumes: <none>    [root@idc01 ~]# kubectl rollout undo deploy mynginx --to-revision=1  deployment.extensions/mynginx rolled back  [root@idc01 ~]# kubectl rollout history deploy mynginx  deployment.extensions/mynginx  REVISION CHANGE-CAUSE  2 <none>  3 <none>    [root@idc01 ~]# kubectl get deploy,rs,po -l app=mynginx  NAME READY UP-TO-DATE AVAILABLE AGE  deployment.extensions/mynginx 3/3 3 3 30m    NAME DESIRED CURRENT READY AGE  replicaset.extensions/mynginx-5d7f998444 0 0 0 4m35s  replicaset.extensions/mynginx-86f7d8cfcc 3 3 3 30m    NAME READY STATUS RESTARTS AGE  pod/mynginx-86f7d8cfcc-9k77c 1/1 Running 0 14s  pod/mynginx-86f7d8cfcc-gz2qx 1/1 Running 0 12s  pod/mynginx-86f7d8cfcc-kxwsx 1/1 Running 0 10s |

|  |
| --- |
| kubectl create deployment mynginx --image=nginx:1.15-alpine  kubectl get deploy,rs,po -l app=mynginx  kubectl scale deploy mynginx --replicas=3  kubectl get deploy,rs,po -l app=mynginx    kubectl describe deploy/mynginx  kubectl rollout history deploy mynginx  kubectl rollout history deploy mynginx --revision=1  kubectl set image deploy mynginx nginx=nginx:1.16-alpine  kubectl rollout history deploy mynginx --revision=2  kubectl get deploy,rs,po -l app=mynginx  kubectl rollout undo deploy mynginx --to-revision=1 |