(23.01.08) POD, Replica Set

POD

팟이란?

쿠버네티스의 궁극적인 목표는 어플리케이션을 각 노드에 컨테이너의 형태로 배포하는 것이다. 하지만, 쿠버네티스는 컨테이너를 직접 배포하지 않고 팟이라는 단위로 캡슐화 해서 배포한다.

> 팟은 쿠버네티스에서 배포될 수 있는 가장 작은 단위이다.

컨테이너를 직접 배포하면 되지, 왜 팟이라는 단위로 감싸서 배포하는 것일까?

하나의 팟은 여러개의 컨테이너를 가질 수 있다. (단, 동일한 종류의 컨테이너를 여러개 가지는 것은 불가능하다.) 같은 팟에 있는 컨테이너들은 *(쿠버네티스 배포의 최소단위가 팟이기 때문에)* 생명주기를 같이하고, 동일한 네트워크를 공유하고 있어서 로컬 호스트를 통해서 직접 통신할 수 있다.

팟을 생성하는 방법

실습 결과 캡쳐

```
kubectl run nginx
```

하지만 이 명령어는 어떤 이미지를 사용할지 명시해주지 않았기 때문에 오류가 난다.

```
kubectl run nginx --image nginx
```

```
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  4m35s  default-scheduler  Successfully assigned default/nginx to minikube
  Normal  Pulling    4m34s  kubelet            Pulling image "nginx"
  Normal  Pulled     4m25s  kubelet            Successfully pulled image "nginx" in 9.199806296s
  Normal  Created    4m25s  kubelet            Created container nginx
  Normal  Started    4m25s  kubelet            Started container nginx
```

kubectl describe pod nginx 명령어를 통해서 생성한 팟의 상세 정보를 볼 수 있다. 맨 아래의 Events 에는 팟을 생성한 후의 모든 이벤트 로그를 볼 수 있다.

1. Scheduled : 어떤 노드에 팟을 생성할지 scheduler 가 결정한다. 여기에서는 minikube 를 통해 구성했으므로 노드가 한개여서 default 노드로 할당되었다.
2. Pulling, Pulled : nginx 이미지를 도커 허브에서 (혹은 다른 도커 레지스트리에서) 내려받는다.
3. Created, Started : nginx 컨테이너가 생성되고 실행된다.

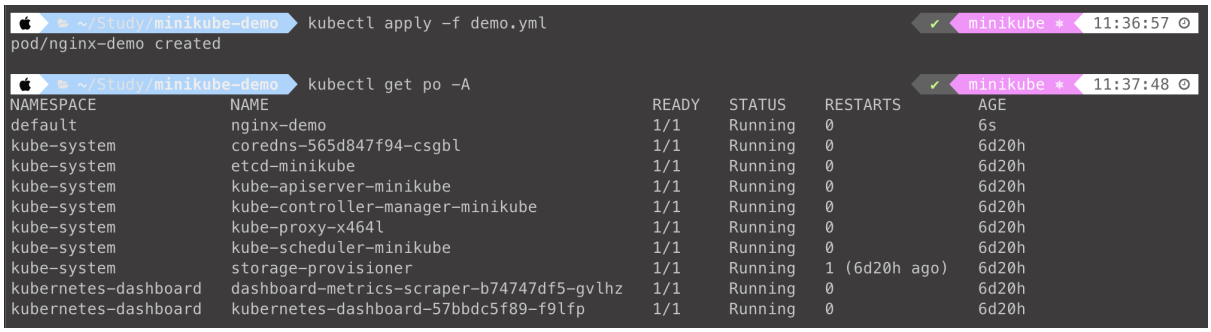
YAML 로 팟을 생성하기

쿠버네티스에서는 YAML 파일을 통해 Pod, Replicas, Deployment 등의 쿠버네티스 오브젝트들을 생성할 수 있다.

```
apiVersion:
kind:
metadata:

spec:
```

1. **apiVerison** : 쿠버네티스 오브젝트를 생성할때 사용하는 쿠버네티스 API 의 버전을 명시한다. `kubectl api-verisons` 명령으로 현재 클러스터에서 사용가능한 API 버전을 확인할 수 있다.
2. **kind** : Pod, Deployment, Ingress 등 어떤 종류의 오브젝트를 생성할지 명시한다.
3. **metadata**: name, label 과 같은 오브젝트에 대한 메타데이터들을 명시한다.
4. **spec** : 어떤 이미지를 가지고 생성할지 명시한다.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-demo
  labels:
    app: nginx-demo
spec:
  containers:
    - name: nginx
      image: nginx
```



kubectl apply -f demo.yml 명령어로 팟을 배포한 모습 (여기서 -f는 file을 의미한다)

팟의 생명주기

⚠️ ——————————————————아직 작성중
——————————————————-

팟을 편집하고 싶다면?

1. `kubectl edit pod <pod-name>` 커맨드를 사용해서 수정한다.
2. 팟을 생성한 YAML이 있는 경우, YAML을 수정한 다음 `kubectl apply -f <파일명>` 커맨드를 사용해서 적용한다.
3. 팟을 생성한 YAML이 없지만, YAML을 생성해서 수정하고 싶은 경우 `kubectl get pod <pod-name> -o yaml > test.yaml` 커맨드를 사용해서 yaml을 추출한 다음 적용한다.

ReplicaSet

Replication Controller

컨트롤러는 쿠버네티스의 온도 조절기 같은 존재이다.

컨트롤러는 쿠버네티스 오브젝트들을 모니터링하고, 그에 따라 대응하는 프로세스이다.

레플리케이션 컨트롤러는 컨트롤러중 하나이다. 그럼 레플리케이션 컨트롤러는 왜 필요한 것일까?

레플리케이션 컨트롤러 (이하 rc) 는 지정된 수의 팟 복제본 (Pod replicas)이 한 번에 실행되도록 돕는다. 즉, rc는 지정된 수의 팟들이 항상 떠있고 사용 가능하도록 한다.

만약 지정한 것보다 너무 많은 팟이 있다면 extra 팟들을 종료시키고, 너무 적은 팟이 있다면 팟들을 생성한다.

`kubectl run` 을 통해 수동으로 생성된 팟과는 달리, rc에 의해 관리되는 팟들은 종료되거나 삭제되면 자동으로 교체가 된다. (따라서 단 1개의 팟만이 필요한 어플리케이션도 반드시 rc를 사용해야 한다.)

이것이 높은 가용성 (high availablity)을 가능하게 한다.

레플리케이션 컨트롤러는 부하(load)를 나눌 여러 개의 팟을 생성하는 것을 돕는다.

만약 노드의 물리적 가용성에 한계가 오면, 다른 노드에 같은 팟을 생성해야하는데, 레플리케이션 컨트롤러는 이처럼 여러개의 노드에 걸쳐 있어 팟을 생성하도록 돕는다.

ReplicaSet과 Replication Controller 의 차이

두 개는 같은 목적을 가지고 있지만, 다르다.
Replication controller 는 더 오래된 기술로, Relica Set 에 의해 대체되었다.

일단 Replica Set은 제쳐두고 Replication Controller 를 YAML 로 작성해서 배포해보자.

YAML로 Replication Controller 를 생성하기

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-rc
spec:
  template:
    metadata:
      name: nginx-pod
      labels:
        app: nginx-demo
    spec:
      containers:
        - name: nginx
          image: nginx
  replicas: 3
```

```
~/Study/minikube-demo kubectl apply -f replica-demo.yml                    minikube ⎈ 12:18:11
replicationcontroller/nginx-rc created

~/Study/minikube-demo kubectl get replicationcontroller                    minikube ⎈ 12:18:19
NAME       DESIRED   CURRENT   READY   AGE
nginx-rc   3         3         2       11s

~/Study/minikube-demo kubectl get pods                                     minikube ⎈ 12:18:30
NAME             READY   STATUS    RESTARTS   AGE
nginx-rc-8nptx   1/1     Running   0          14s
nginx-rc-98vtj   1/1     Running   0          14s
nginx-rc-xx2mj   1/1     Running   0          14s

~/Study/minikube-demo kubectl delete pod nginx-rc-8nptx                    minikube ⎈ 12:18:33
pod "nginx-rc-8nptx" deleted

~/Study/minikube-demo kubectl get pods                                     minikube ⎈ 12:22:44
NAME             READY   STATUS              RESTARTS   AGE
nginx-rc-98vtj   1/1     Running             0          4m26s
nginx-rc-q52tv   0/1     ContainerCreating   0          2s
nginx-rc-xx2mj   1/1     Running             0          4m26s
```

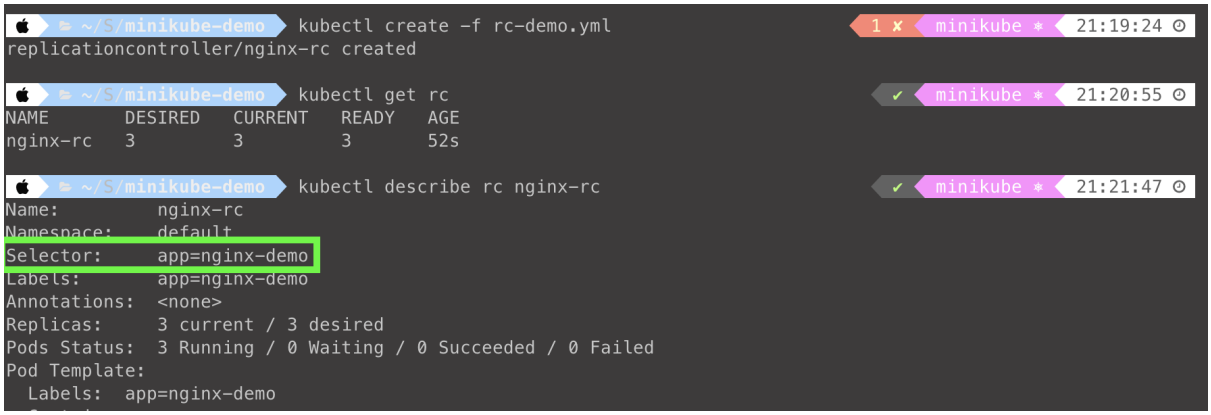
`kubectl apply -f replica-demo.yml` 명령어로 Replication Contorller를 생성한다. 생성된 Replication Contorller는 `kubectl get replicationcontroller` 명령어를 통해서 확인할 수 있다.

YAML 파일에 `replicas: 3` 로 명시하였는데, `kubectl get pods` 를 해보면 팟이 3개 떠있는 것을 볼 수 있다. 만약 이 중 한개의 팟을 죽이면 다시 팟이 생성되는 모습을 볼 수 있다.

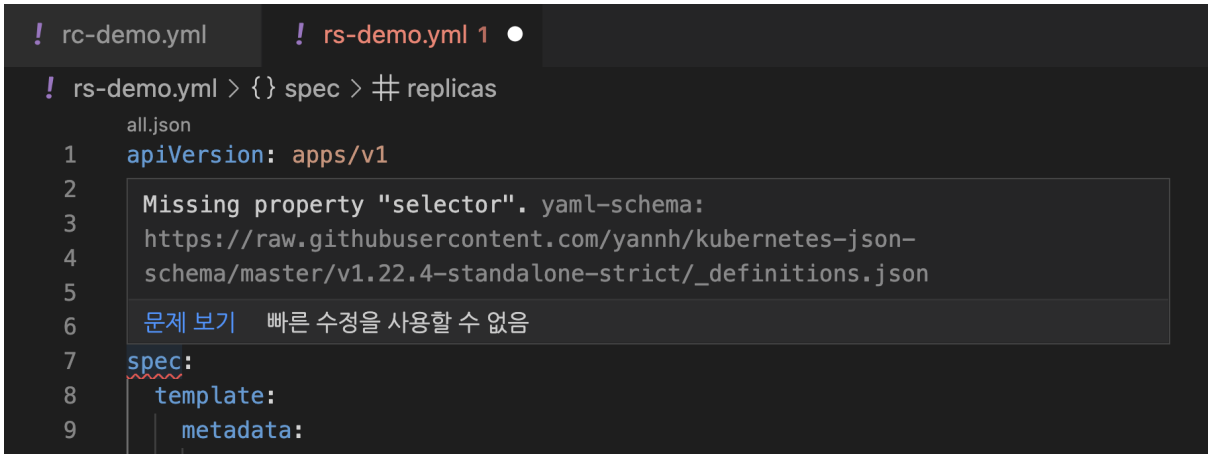
ReplicaSet

먼저 Replica Set (이하 rs)은 apiVerison을 `apps/v1` 으로 해주어야 한다. (rc 는 `v1` 이다.)

rc와 rs의 가장 큰 차이점은 rs는 spec 에 selector 를 기입해주어야 한다는 것이다. rc에도 selector 를 기입해줄 수 있지만, 필수는 아니다. 만약 rc를 생성할 때 selector 를 기입해주지 않으면 pod 정의 template 에 정의된 label 이 selector 로 지정이 된다.



selector를 기입하지 않고 replica controller 를 생성하였더니 selector 가 자동으로 지정이 되었다.



반면 ReplicaSet은 selector를 기입하지 않으면 오류가 난다.

ReplicaSet은 selector 기입이 필수로 rc에서는 지원하지 않는 다양한 레이블 매칭 옵션을 지원한다.

Label과 Selector

우리는 팟이나 다른 쿠버네티스 오브젝트에 label을 붙일 수 있다. label은 왜 붙이는 것일까? ReplicaSet 은 지정된 숫자의 팟 replica가 살아있을 수 있도록 팟을 모니터링한다고 했다. 하지만 쿠버네티스 클러스터에는 수많은 팟이 있다. 팟을 생성할때 label을 다는 것이 유용한 이유가 바로 이것이다.

selector 필드에 우리가 팟을 생성할 때 달았던 label을 명시해서 일종의 필터 역할을 하게 하는 것이다. 이러한 컨셉은 단순히 ReplicaSet에서 뿐만 아니라 쿠버네티스 내에서 두루 쓰인다.

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

만약에 팟 3개가 이미 생성이 되어있고, replicas 옵션이 3인 ReplicaSet을 배포한다면 어떻게 될까? 팟이 이미 3개 생성되어 있기 때문에, ReplicaSet은 추가적인 팟을 생성하지 않는다. 그러면 위의 template 영역은 불필요한게 아닐까? 아니다. 만약 3개 중 어떤 팟이 죽는다면 다시 살려야 하기 때문에, template는 필요하다. **따라서 template에 명시된 pod의 label과 selector의 label은 반드시 동일해야한다.**

단, 주의할 점이 있는데 ReplicaSet은 템플릿에 지정한 팟에 국한되지 않고, (ReplicaSet이 생성되기 직전) 이미 생성되어있던 팟도 label을 보고 관리대상에 추가한다. 예를 들면, 팟이 1개가 이미 생성되어있고, replicas 옵션이 3인 ReplicaSet을 배포할 경우 - ReplicaSet은 템플릿을 가지고 2개의 팟만을 새로 생성한다.

그럼 이미 생성되어있는 팟이 label만 ReplicaSet의 selector와 동일하고, 팟을 만드는데 생성한 컨테이너 이미지가 다르면 어떻게 될까? 아래 예시를 보자.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-nginx-demo
  labels:
    app: nginx-demo
spec:
  containers:
    - name: hello
      image: gcr.io/google-samples/hello-app:2.0
```

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

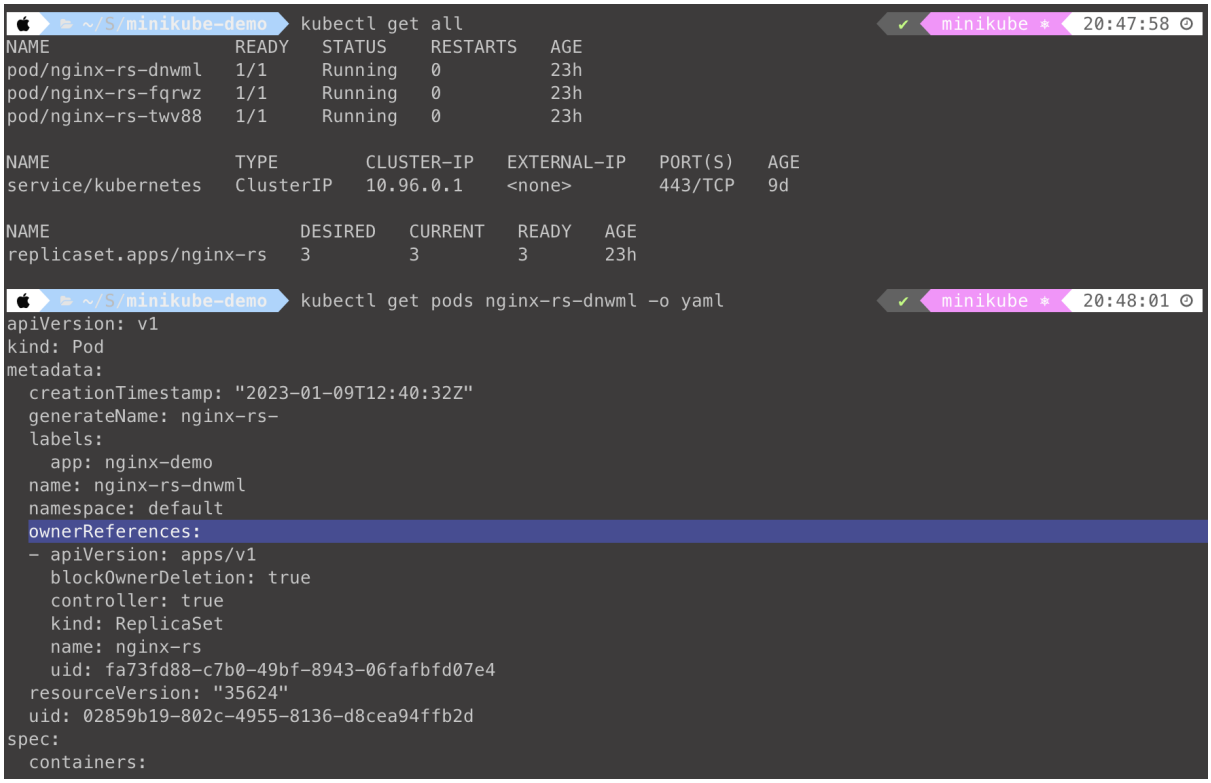
```
~/S/minikube-demo  kubectl create -f hello-pod-demo.yml                    minikube ⎈ 20:55:50
pod/hello-nginx-demo created

~/S/minikube-demo  kubectl create -f rs-demo.yml                           minikube ⎈ 20:55:56
replicaset.apps/nginx-rs created

~/S/minikube-demo  kubectl get all                                         minikube ⎈ 20:56:13
NAME                 READY   STATUS              RESTARTS   AGE
pod/hello-nginx-demo   1/1     Running             0          21s
pod/nginx-rs-9qvpm     0/1     ContainerCreating   0          4s
pod/nginx-rs-dswqp     0/1     ContainerCreating   0          4s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   9d

NAME                         DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-rs   3         3         1       4s
```

hello 이미지로 생성된 팟 1개, nginx 이미지로 생성된 팟 2개가 같은 ReplicaSet에 잡혀 있다.

위 예시에서 볼 수 있듯이 ReplicaSet이 기존에 떠있던 팟을 관리대상에 추가할때는 템플릿과 동일한 팟인지를 검사하지 않는다. 따라서 우리는 ReplicaSet을 배포하기 전에 selector와 일치하는 label을 가지는 팟이 없는지를 체크하는 편이 좋다.

ReplicaSet과 Pods

ReplicaSet과 팟은 팟의 `metadata.ownerReferences` 필드를 통해 이어져 있다. 위 예시의 YAML의 템플릿에 `metadata.ownerReferences` 필드를 명시해준적이 없는데, 무엇을 의미하는 걸까?



ReplicaSet에 의해 관리되는 팟은 metadata.ownerReferences 필드를 가진다. 여기에는 ReplicaSet의 정보가 있는데 말그대로 owner 의 정보를 가지고 있다고 생각하면 된다.

ReplicaSet은 이 링크를 통해 팟들의 상태를 관리한다.

ReplicaSet Scaling

- YAML을 수정하고, `kubectl replace -f rc-demo.yml` 명령어로 재배포한다.
- `kubectl scale --replicas=6 -f rc-demo.yml` 로 scale한다.
- yml을 명시하지 않고 `kubectl scale --replicas=6 replicaset ngix-rs` 처럼 replicaset의 이름을 명시해서 sacle 할 수도 있다.