Chapter 10. StatefulSets: deploying replicated stateful applications

10.1. Replicating stateful pods

각각 다른 상태를 가진 분산형 애플리케이션을 배포해야한다고 가정해보자. ReplicaSet을 이용해서 할 수 있을까? 없다.

각각 다른 상태를 가진 분산형 애플리케이션은 죽다가 다시 살아나도 변하지 않는 안정적인 ID(서비스 IP 또는 도메인)을 가져야하기 때문이다.

그렇다고 Pod 하나당 ReplicaSet과 Service를 각각 만든다면 어떻게 될까? 개별 포드들은 자신이 어떤 서비스(어떤 주소)로 노출되어있는 지 알 수 없으며, Peer를 찾기도 어렵다.

이런 상태가 있는 분산형 애플리케이션을 배포하기 위해서 Kubernetes에서는 StatefulSet이라는 것이 있다.

10.2. Understanding StatefulSets

10.2.1. Comparing StatefulSets with ReplicaSets

Stateless, Cattle, ReplicaSet

상태가 없기 때문에 인스턴스가 소멸되더라도 새 인스턴스를 만들면 문제되지 않는다.

ReplicaSet으로 관리되는 Pod들은 비유적으로 가축(cattle)과 같다. 언제든 새로운 Pod으로 교체 가능하다.

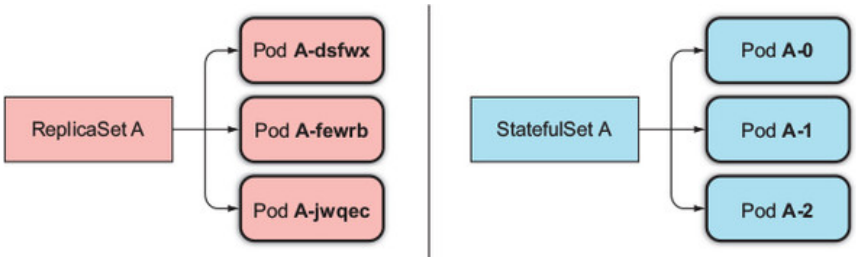
Stateful, Pet, StatefulSet

상태(예를 들어 추억)가 있기 때문에 해당 인스턴스가 소멸되면 정확히 같은 것이 아니라면 대체할 수 없다.

StatefulSet은 같은 애완동물을 부활시키듯, 관리되는 Pod들을 기존 인스턴스와 동일한 이름, 네트워크ID 등 공유정보를 가져와서 상태를 일치시킨다.

10.2.2. Providing a stable network identity

StatefulSet에 의해 생성된 Pod들은 규칙적인 이름을 갖는다. ReplicaSet이 만든 Pod의 이름은 뒤에 해시값이 붙지만, StatefulSet이 만든 Pod은 0부터 순서대로 숫자가 붙는다.



Introducing the governing Service - Headless Service

일반적으로 상태가 없는 분산형 서비스라면 그냥 앞에서 로드발란싱을 잘 해주면 되겠지만, 각각 상태가 있는 분산형 서비스는 각 인스턴스들이 개별로 동작하기 때문에 일반적으로 로드발란싱하지 않고 각각 주소를 필요로 한다. StatefulSet에서는 이러한 요구사항에 대응하기 위해서 관리용 Headless Service를 만든다.

> **Headless Service**
>
> ClusterIp를 가지지 않고 주소(DNS 이름)만 가지고 있는 서비스로 이 DNS를 조회(lookup)하면 서비스에 연결된 Pod들의 IP 주소를 반환한다.

예를 들어, 기본 네임스페이스(default)에 속하고, 서비스 이름이 foo이며, Pod들 중 첫번째 팟의 이름이 A-0일 때, 정규화된 도메인이름인 a-0.foo.default.svc.cluster.local 을 통해서 해당 Pod에 접근 가능하다. 또한 foo.default.svc.cluster.local 도메인의 SRV(Service Locator Record)를 검색하여 StatefulSet의 모든 Pod의 이름을 찾고 이를 통해서 Peer를 찾을 수 있다.

Replacing lost pets

StatefulSet에서 인스턴스가 사라지면(노드가 네트워크 연결이 끊기거나 누군가 Pod을 수동으로 삭제하면) StatefulSet은 새로운 Pod을 만들고 사라진 Pod과 동일한 이름과 호스트 이름을 갖도록 한다.

Scaling a StatefulSet

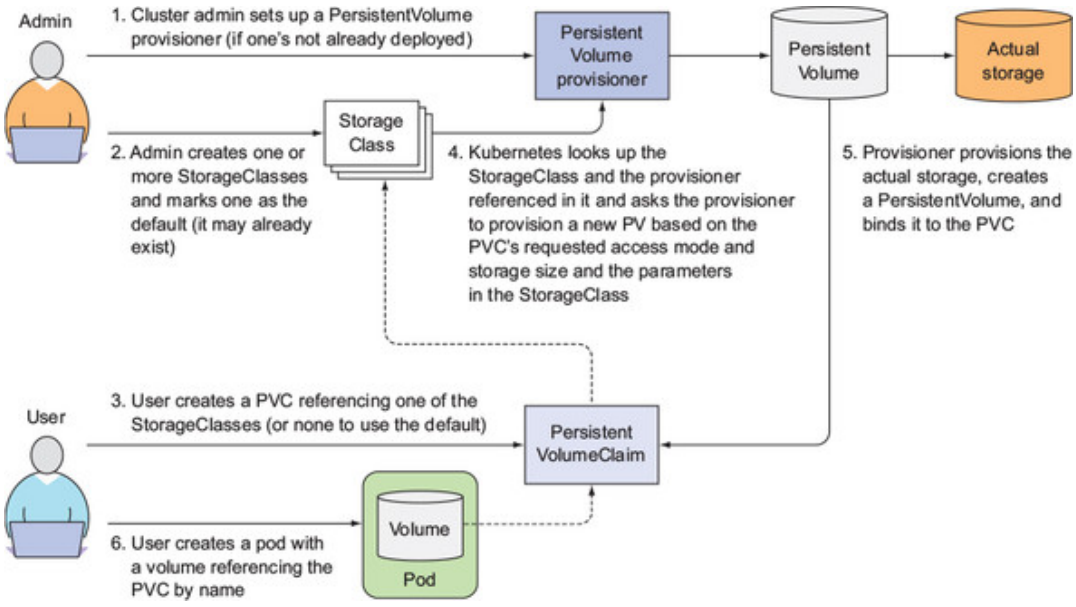
StatefulSet에서 Pod을 scale-out하면 사용되지 않은 다음 인덱스 번호를 사용해서 새로운 Pod을 생성한다. 그리고 scale-in하면 마지막 인덱스 부터 ***하나씩*** 제거(drain)된다.

예를 들어, A-0, A-1 이렇게 두개의 Pod이 있었다면 scale-out 이후에는 A-2 Pod이 생성된다. 그리고 빠질 때는 A-2 부터 빠진다.

10.2.3. Providing stable dedicated storage to each stateful instance

(각 스테이트풀 인스턴스에 안정적인 전용 스토리지 제공)

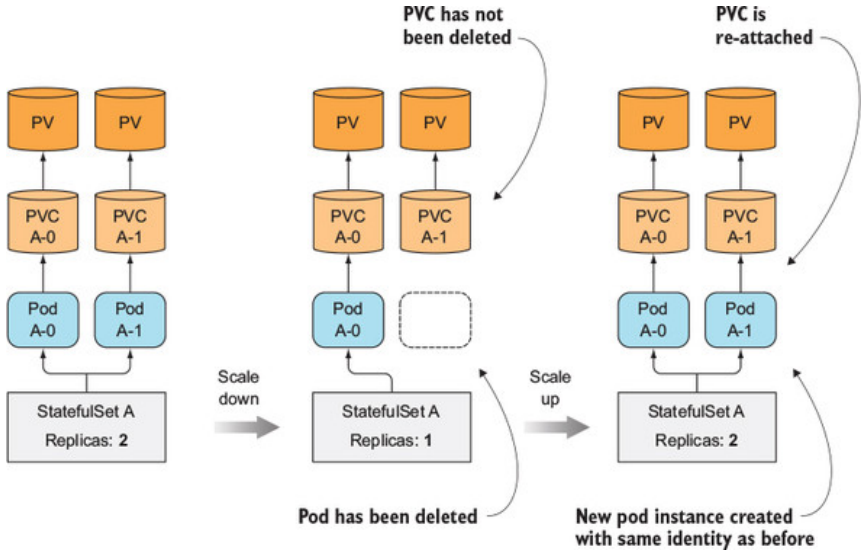
복습 - PersistentVolume & PersistentVolumeClaim



각각 상태가 있기 때문에 각 Pod들의 스토리지는 다른 Pod들과 분리되어야 한다. 그리고 6장에서 PersistentVolume과 PersistentVolumeClaim을 통해서 Pod에 영구 스토리지를 매핑하는 것을 배웠다.

결론적으로 StatefulSet은 VolumeClaimTemplate를 통해서 자신만의 PersistentVolumeClaim을 갖는다. 그리고 각각 별개의 PersistentVolume에 스토리지를 요청한다.

StatefulSet에서 스케일다운 같은 상황을 통해 Pod이 종료되더라도 PersistentVolumeClaim은 삭제되지 않는다. 만약 다시 스케일업 할때 새로 만들어진 인스턴스가 기존 PersistentVolumeClaim에 연결된다. 만약 지우고 싶다면 수동으로 지워야한다.



10.2.4. Understanding StatefulSet guarantees

StatefulSet은 안정적인 ID와 스토리지를 보장하며, 절대 동일한 ID의 인스턴스가 1개임을 보장해야 한다.

10.3. Using a StatefulSet (With GKE)

10.3.1. 앱 및 컨테이너 이미지 만들기

POST로 받은 요청을 파일로 저장하는 NodeJS 앱을 만든다. 저장된 파일이 상태를 의미한다. 편하게 저자가 만든 이미지를 그냥 사용한다.

```
docker pull luksa/kubia-pet
```

10.3.2. 스테이트풀 셋을 통한 애플리케이션 배포

애플리케이션 배포를 위해 다음의 객체들을 만들어 두어야한다.

- 데이터 파일을 저장하는 PersistentVolume
- StatefulSet에 대한 관리서비스 (Headless Service)
- StatefulSet

PersistentVolume 생성

```
gcloud compute disks create --size=1GiB --zone=asia-northeast1-a pv-a
gcloud compute disks create --size=1GiB --zone=asia-northeast1-a pv-b
gcloud compute disks create --size=1GiB --zone=asia-northeast1-a pv-c
```

코드 : https://github.com/luksa/kubernetes-in-action/blob/master/Chapter10/persistent-volumes-gcepd.yaml

관리 서비스 생성

코드 : https://github.com/luksa/kubernetes-in-action/blob/master/Chapter10/kubia-service-headless.yaml

StatefulSet 매니페스트 작성

코드 : https://github.com/luksa/kubernetes-in-action/blob/master/Chapter10/kubia-statefulset.yaml

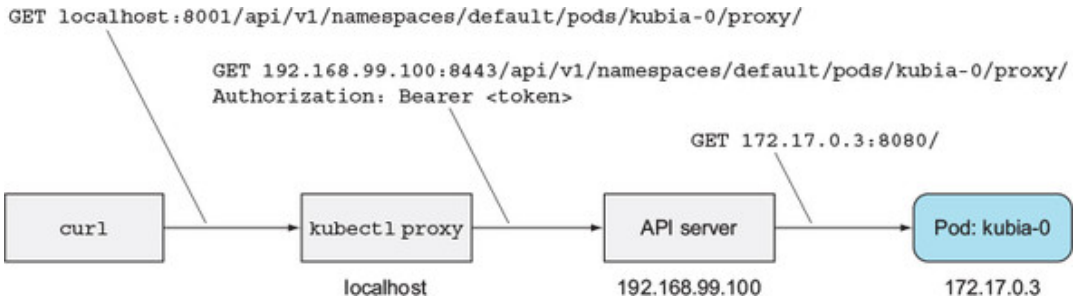
```
...
  volumeClaimTemplates:
  - metadata:
      name: data
    spec:
      resources:
        requests:
          storage: 1Mi
      accessModes:
      - ReadWriteOnce
```

```
kubectl create -f kubia-statefulset.yaml
```

⚠️ 중요포인트! Pod이 동시에 생성되는 것이 아니고 하나씩 생성된다.

10.3.3. 실제 포드로 동작해보기

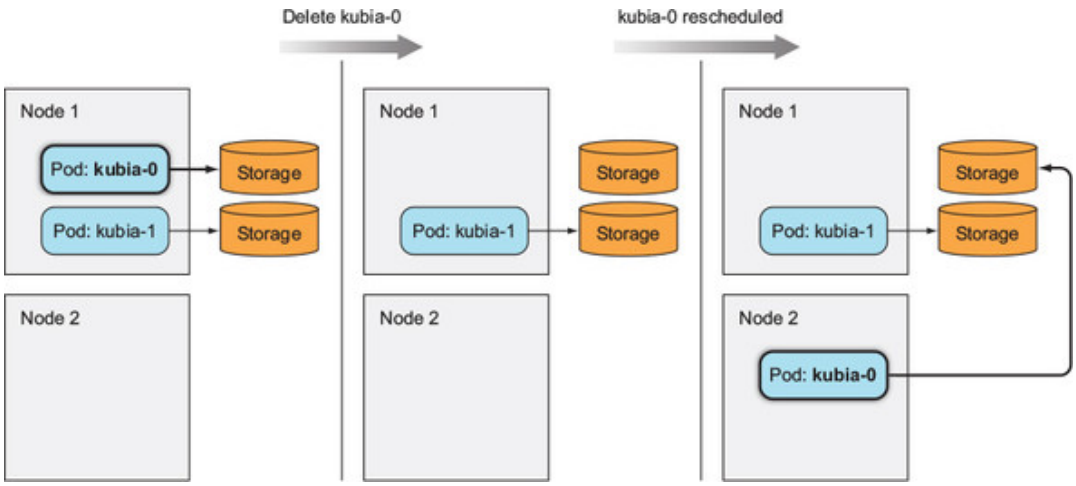
API 서버를 통해서 포드와 통신한다.




```
# kube proxy 실행
kuberctl proxy
# GET test
curl localhost:8001/api/v1/namespaces/default/pods/kubia-0/proxy/
# POST test
curl -X POST -d "Hey there! This greeting was submitted to kubia-0." \
localhost:8001/api/v1/namespaces/default/pods/kubia-0/proxy/
```

강제로 팟을 삭제해보고 같은 스토리지에 붙는지 테스트해본다.

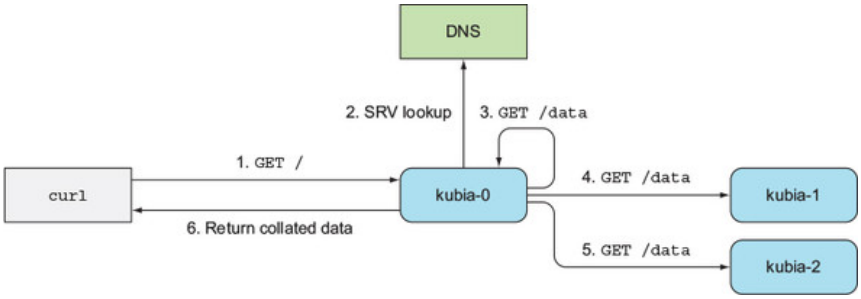
```
kubectl delete po kubia-0
```



10.4. DNS를 이용한 Peer 검색

SRV 레코드는 특정 서비스를 제공하는 서버의 호스트 이름 및 포트를 가리키는데 사용된다. 쿠버네티스는 헤드리스 서비스를 지원하는 포드의 호스트 이름을 가리키도록 SRV 레코드를 만든다.

애플리케이션(팟)은 DNS에 SRV lookup을 통해서 Peer의 주소를 가져올 수 있다.



10.5. 스테이트풀셋 장애 제거

노드가 네트워크 장애가 생겼을 때, Pod은 제거되지 않고 Unknown 상태가 된다. kubelet이 Pod의 컨테이너가 종료되었음을 API 서버에 알릴 수 없기 때문이다. 그래서 만약 실제로 이런 상황이 생겨서 새로운 팟을 띄워야한다면 장애가 난 팟을 강제로 지워야 한다.

```
kubectl delete po kubia-0 --force --grace-period 0
```

참고 링크

- 소스코드 : https://github.com/luksa/kubernetes-in-action/tree/master/Chapter10
- StatefulSet으로 MySql 배포하기 : https://kubernetes.io/docs/tasks/run-application/run-replicated-stateful-application/#deploy-mysql