2.3 쿠버네티스 첫 번째 애플리케이션 실행하기

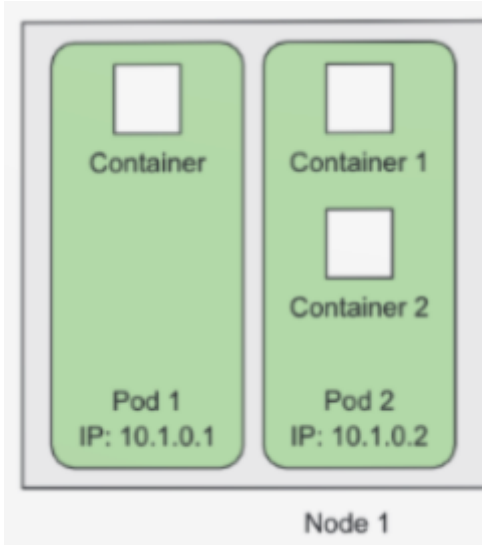
`kubectl run kubia --image=tree9295/kubia --port=8080 --generator=run/v1`

- 도커 허브에 푸시한 이미지로 오브젝트 만들어서 애플리케이션 구동하기
- 래플리케이션 컨트롤러 생성한 것

파드

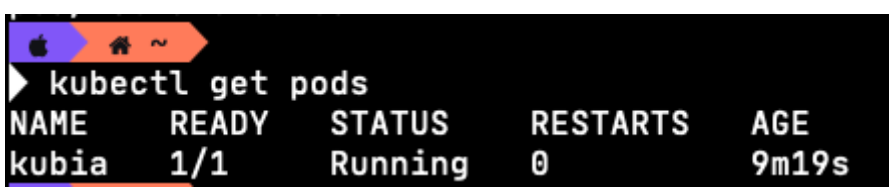
쿠버네티스는 개별 컨테이너들을 직접 다루지 않는다. 대신 함께 배치된 다수의 컨테이너 개념을 사용하는데, 이 컨테이너의 그룹을 파드(Pod)라고 한다.

- 파드는 하나 이상의 밀접하게 연관된 컨테이너의 그룹
- 같은 워커노드에서 같은 리눅스 네임스페이스로 함께 실행됨
- 각 파드는 자체IP, 호스트 이름, 프로세스 등이 있는 논리적 분리된 머신



`kubectl get pods`

- 파드 조회하기



`kubectl describe pod`

- 파드 상세 조회
- IP, container id, image, host,

위 과정을 전체적인 그림으로 보면

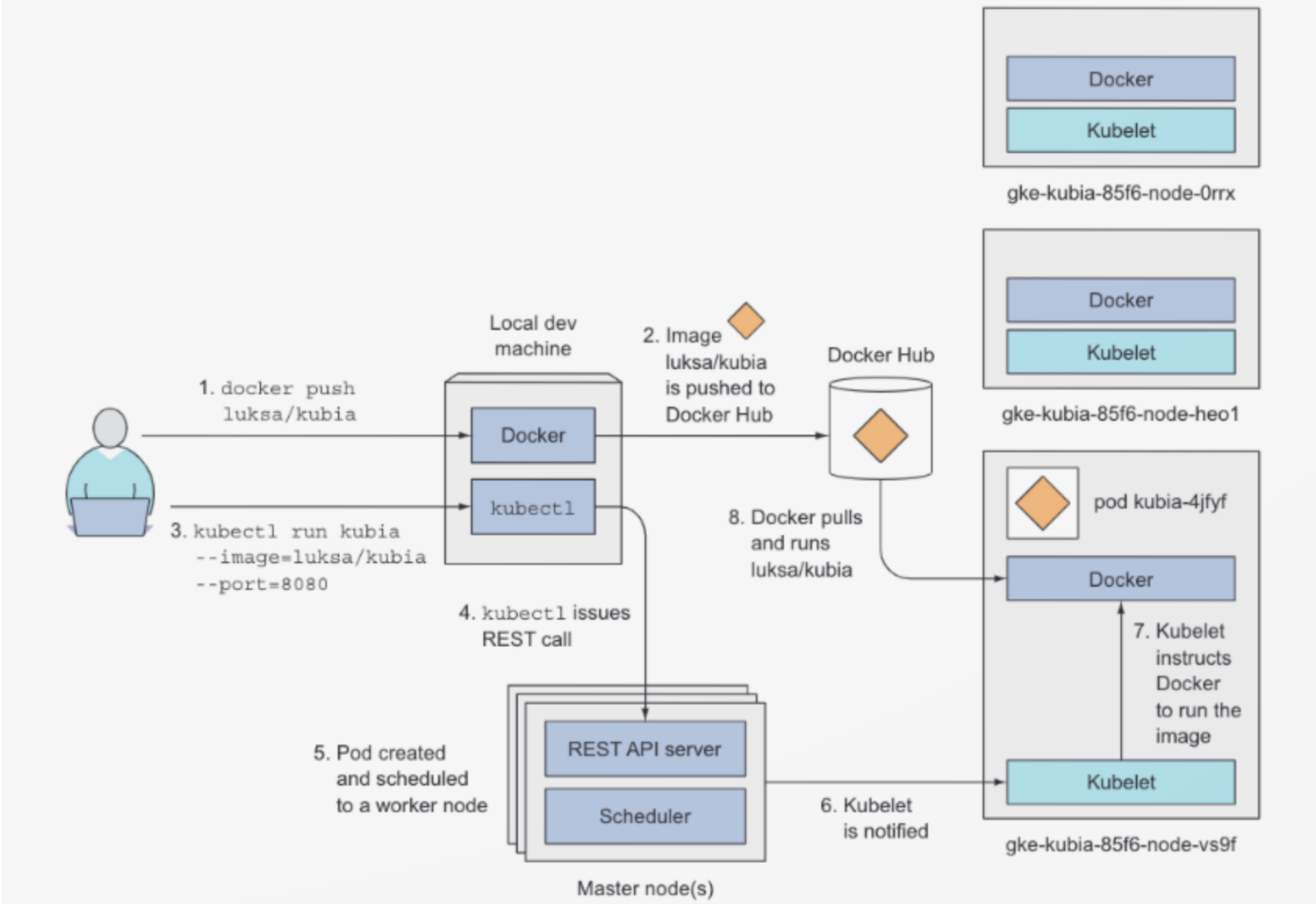


Figure 2.6 Running the `luksa/kubia` container image in Kubernetes

두 부분으로 나눌 수 있을 것 같은데

- 도커 허브에 이미지 컨테이너를 푸시하는 과정
- 쿠버네티스에서 이미지를 받아 실행하는 과정

파드에 접근하기

- 각 파드는 자체 IP주소를 갖고 있지만 내부 주소여서 외부에서 접근이 불가능하다.
- 서비스 오브젝트를 통해 노출해야만 한다. → LoadBalancer 유형의 서비스를 활용한다.

`kubectl expose rc kubia --type=LoadBalancer --name kubia-http`

- 생성한 레플리케이션 컨트롤러 노출 (rc = replicationcontroller)

`kubectl get services`

- 쿠버네티스 서비스 조회
- 위 과정을 수행하면 여기서 서비스의 external IP가 조회된다.
- 해당 IP를 통해 서비스로 접근할 수 있다.

중간 정리

파드와 컨테이너? 레플리케이션컨트롤러? 서비스?

파드와 컨테이너

파드가 원하는 만큼의 컨테이너를 포함하고 있다. 파드는 자체 고유한 IP 주소와 호스트 이름을 갖는다.

레플리케이션컨트롤러

항상 정확히 의도하는 레플리카의 수 만큼의 파드 인스턴스를 실행하도록 지정한다. (위 예제에서는 1개)

보통 파드를 복제하고 항상 실행상태로 만든다. 즉, 어떤 이유로 파드가 사라지거나 죽으면, 사라진 파드를 대체하기 위해 새로운 파드를 생성한다.

서비스

파드는 일시적인 존재 → 언제든 사라질 수 있다.

하지만 사라지면 레플리케이션컨트롤러가 새로 파드를 생성해서 그 파드가 기존 파드를 대체하게 된다. 하지만 새로운 파드의 경우 다른 IP주소를 할당받기에 기존 파드의 주소와 불일치가 생기고 외부로 여러 IP주소가 노출 될 수도 있는 문제를 발생시킨다.

이러한 것을 막기 위해 서비스에서 어떤 파드가 어떤 IP주소를 갖는지에 상관없이 외부의 접근을 하나의 파드로 연결해 요청을 처리하도록 한다.

즉, 서비스는 동일한 서비스를 제공하는 하나 이상의 파드 그룹의 정적 위치(IP)를 나타낸다. (extern

애플리케이션 수평확장

쿠버네티스는 간단하게 확장이 가능하다.

→ 확장하고자 하는 레플리케이션컨트롤러의 의도하는 레플리카 수(desired replica)를 원하는 만큼 늘려주면 된다.

`kubectl get replicationcontrollers`

- 레플리케이션 컨트롤러 조회
- 각 레플리케이션 컨트롤러마다의 DESIRED, CURRENT 레플리카의 수 정보를 조회할 수 있다.

`kubectl scale rc [rc이름] --replicas=[NUM]`

- 레플리카의 수를 변경할 rc이름과 수를 입력하면 의도하는 대로 바뀌게 된다.
- 단순히 쿠버네티스에게 인스턴스 몇개를 유지해야함을 알려주는 것

위 과정을 통해서 시스템의 의도하는 상태를 선언적으로 변경하고 쿠버네티스가 실제 현재 상태를 검사해 의도한대로 조정하게 된다.

ex)

`kubectl scale rc kubia --replicas=3` 을 수행하면

- `kubectl get rc` : DESIRED, CURRNET가 3개로 변함
- `kubectl get pods` : 파드가 3개로 늘어남

→ rc가 파드를 생성하고, 새로운 파드들은 컨테이너 이미지를 다운로드 받아 수분 내에 준비상태가 되고 작동하게 된다.

서비스 호출하면 어떤 파드가 받을까?

서비스 IP로 요청을 보내보자

- `kubectl get services` or `kubectl get svc` 로 서비스 IP 조회 가능

그럼 무작위로 늘어난 파드들을 호출하고 있는 것을 확인할 수 있다.

- 다수 파드 앞에서 로드밸런서 역할을 하는 것

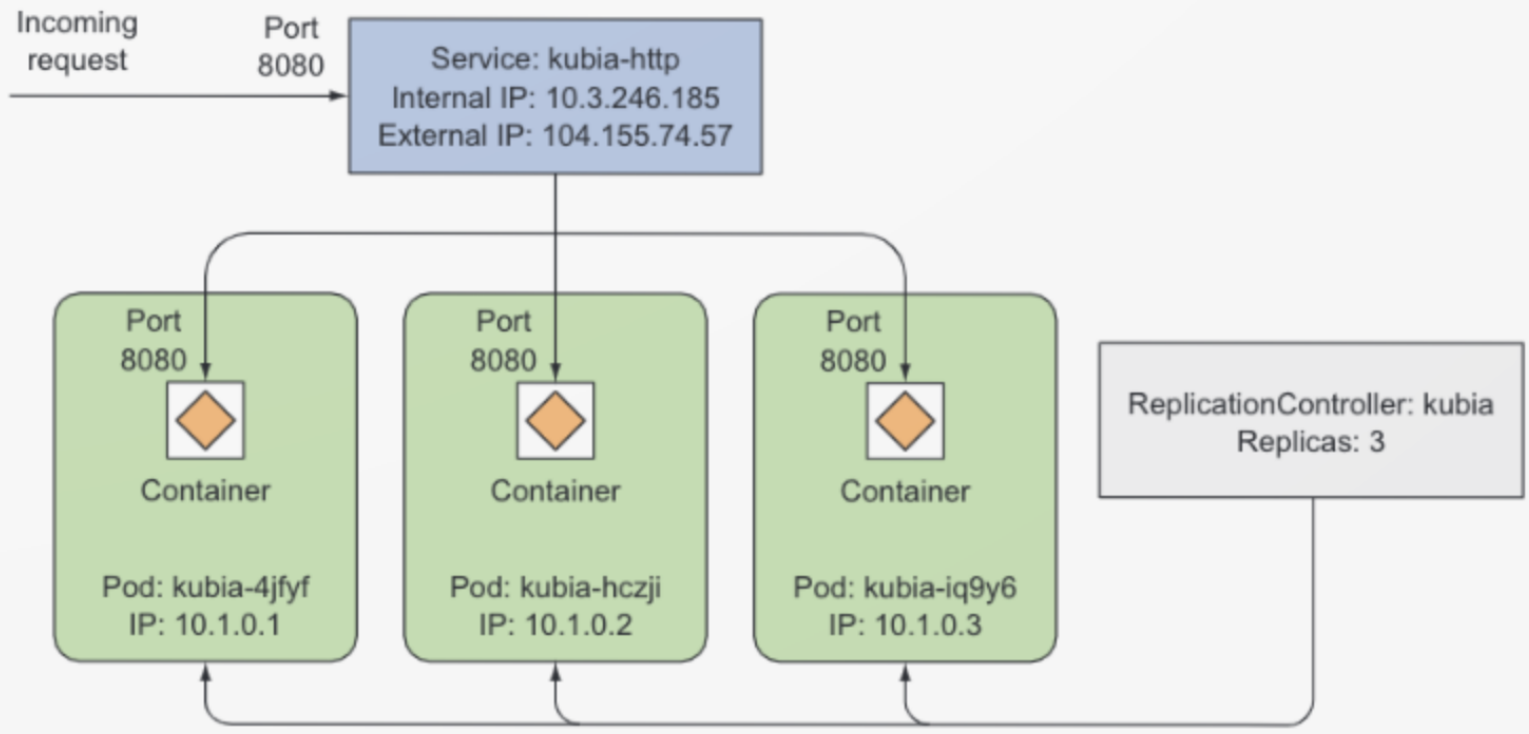


Figure 2.8 Three instances of a pod managed by the same ReplicationController and exposed through a single service IP and port.

- 그림에서 볼 수 있듯, 서비스는 맨 앞에서 하나의 IP주소로 받아들이고 내부적으로는 여러 파드들을 무작위로 호출해 request를 처리하게 된다.