디플로이먼트: 선언적 애플리케이션 업데이트

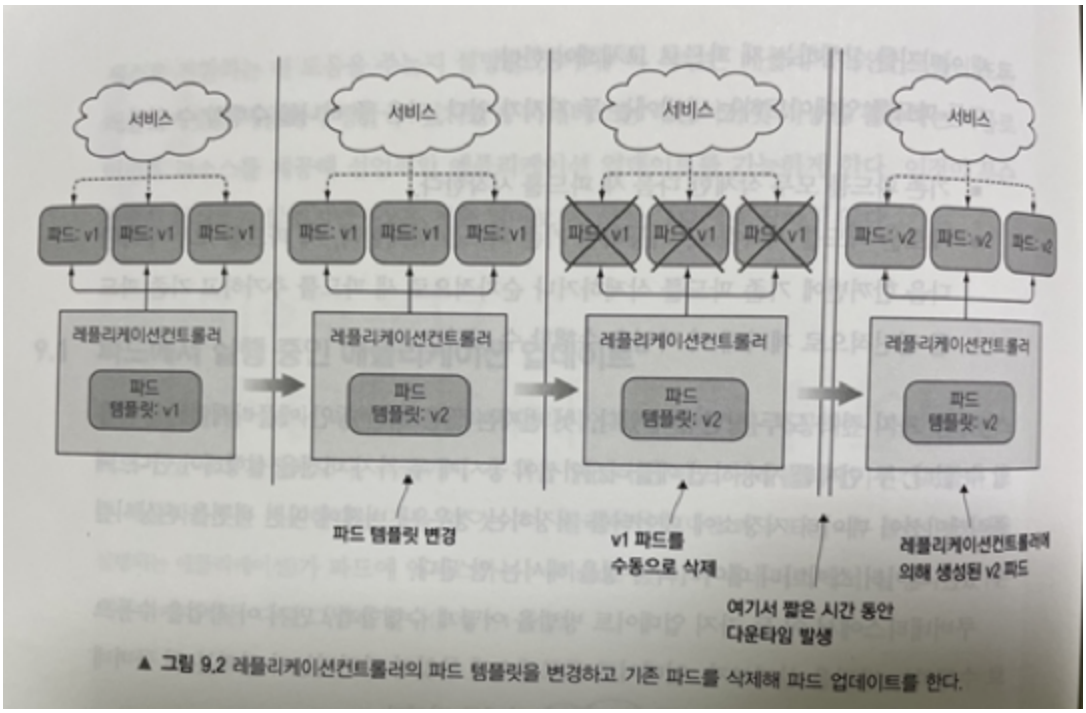
모든 파드를 업데이트하는 방법

1. 기존 파드를 모두 삭제한 다음 새 파드를 시작한다.
2. 새로운 파드를 시작하고, 기동하면 기존 파드를 삭제한다. 새 파드를 모두 추가한 다음 한꺼번에 기존 파드를 삭제한다. 새 파드를 모두 추가한 다음 한꺼번에 기존 파드를 삭제하거나 순차적으로 새 파드를 추가하고 기존파드를 점진적으로 제거해 이 작업을 수행할 수 있다.

---

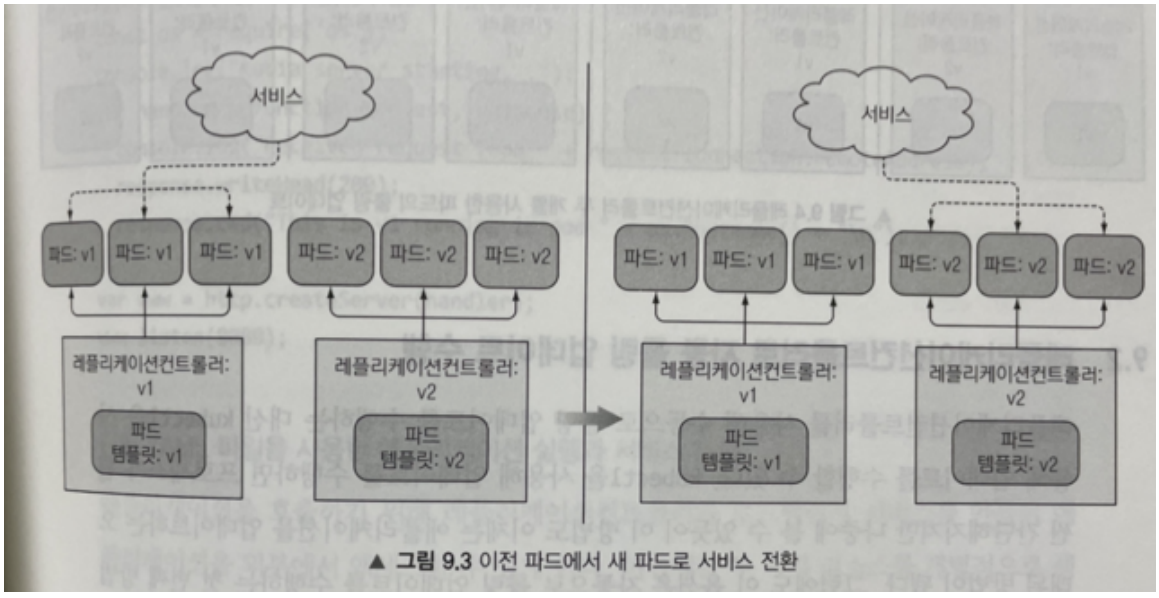
9. 1. 파드에서 실행 중인 애플리케이션 업데이트

1. ReplicationController가 있는 경우 파드 템플릿을 수정한 다음 이전 파드 인스턴스를 삭제.
    - 짧은 시간 동안 다운타임 발생.



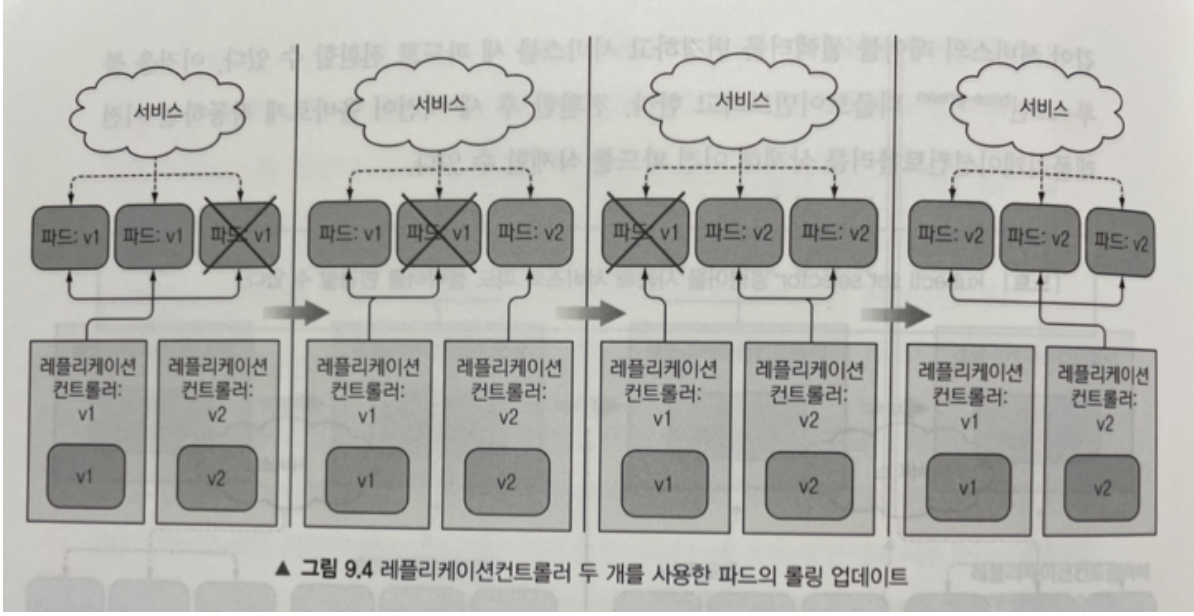
▲ 그림 9.2 레플리케이션컨트롤러의 파드 템플릿을 변경하고 기존 파드를 삭제해 파드 업데이트를 한다.

2. 블루그린 배포
    - 잠시 동안 동시에 두 배의 파드가 실행되므로 더 많은 하드웨어 리소스가 필요.



▲ 그림 9.3 이전 파드에서 새 파드로 서비스 전환

3. 롤링 업데이트
    - 파드를 단계별로 교체. ReplicationController를 천천히 스케일 다운하고 새 파드를 스케일 업해 수행.



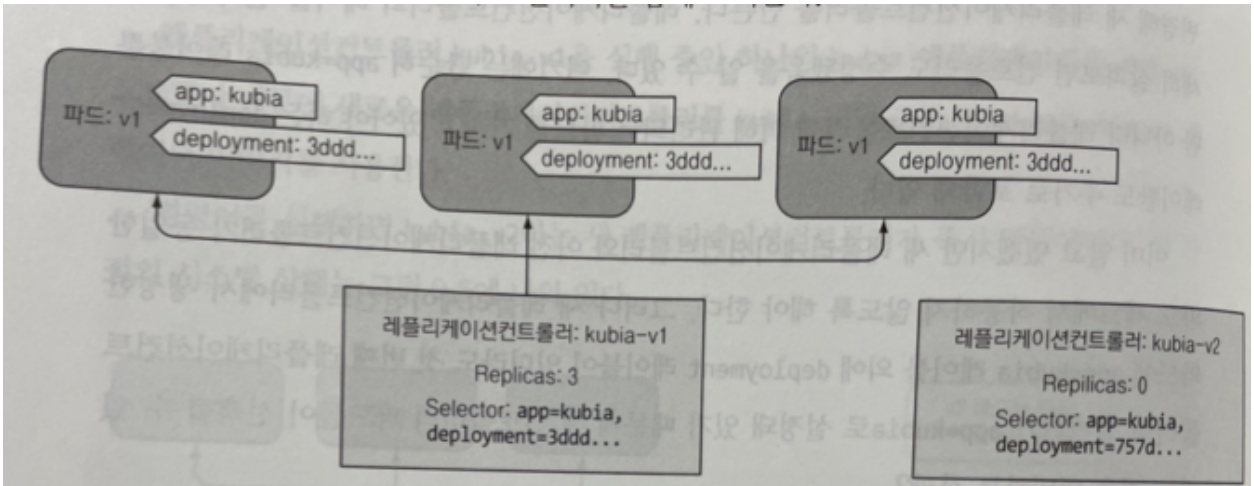
▲ 그림 9.4 레플리케이션컨트롤러 두 개를 사용한 파드의 롤링 업데이트

9.2. ReplicationController로 자동 롤링 업데이트 실행

- kubectl을 사용해서 업데이트 수행.

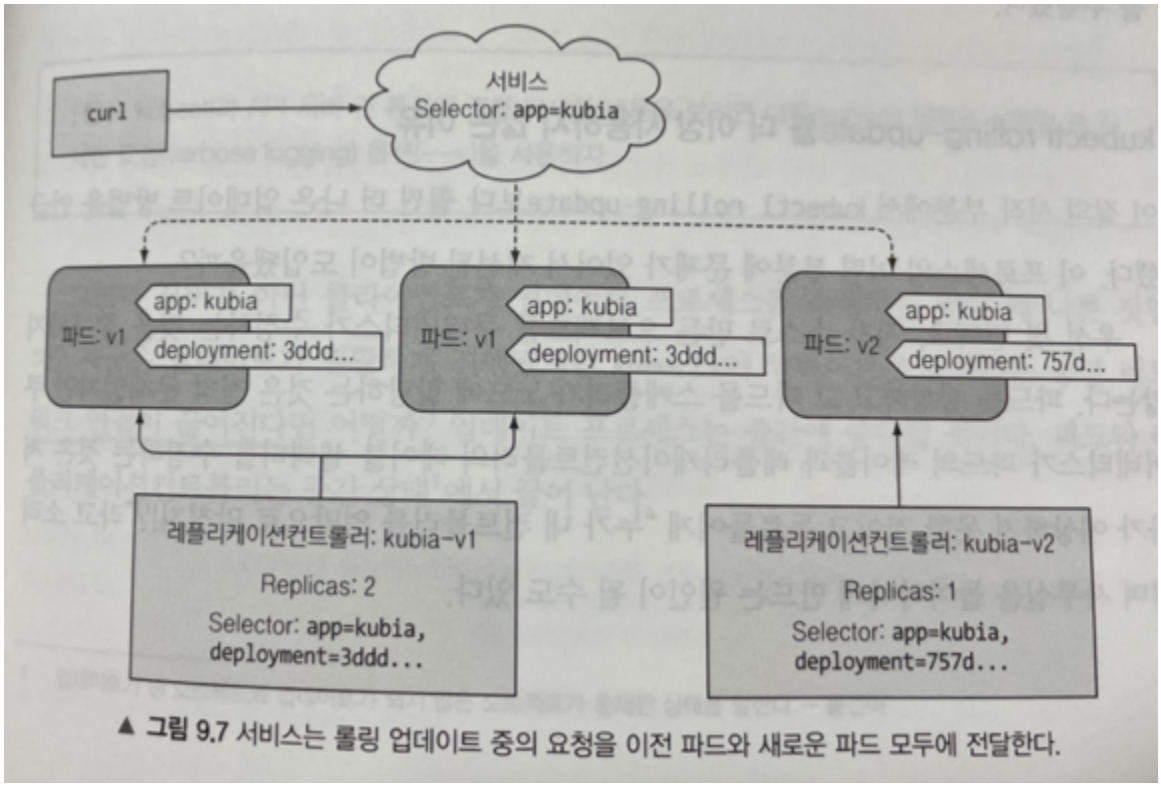
```
kubectl rolling-update kubia-v1 kubia-v2 --image=luksa/kubia:v2
```

- 레플리케이션컨트롤러 kubia-v1을 실행 중인 하나의 kubia 애플리케이션을 버전2로 교체했기 때문에 새로운 ReplicationController를 kubia-v2라고 하고 luksa/kubia:v2 컨테이너 이미지를 사용한다.
  명령어를 실행하면 kubia-v2라는 새 ReplicationController가 즉시 만들어진다.



▲ 그림 9.6 롤링 업데이트 시작 시 기존 레플리케이션컨트롤러와 새 레플리케이션컨트롤러의 자세한 파드 상태

- kubectl이 ReplicationController의 셀렉터를 변경하기 전에 실행 중인 파드의 레이블을 먼저 수정한다.



▲ 그림 9.7 서비스는 롤링 업데이트 중의 요청을 이전 파드와 새로운 파드 모두에 전달한다.

kubectl rolling-update를 더 이상 사용하지 않는 이유

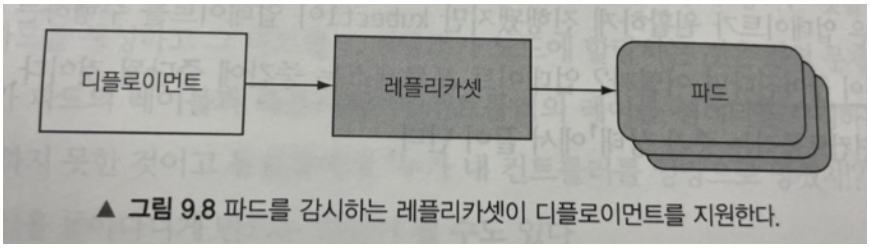
1. 쿠버네티스가 파드의 레이블과 ReplicationController의 레이블 셀렉터를 수정한다.

2. kubectl 클라이언트가 쿠버네티스 마스터 대신 스케일링을 수행한다.
   kubectl이 업데이트를 수행하는 동안 네트워크 연결이 끊어진다면 프로세스는 중간에 중단된다.
3. 실제 명령을 나타낸다.
   쿠버네티스에게 의도하는 시스템의 상태를 선언하고 쿠버네티스가 그것을 달성하는 방식이 아니다.

9.3. 애플리케이션을 선언적으로 업데이트하기 위한 Deployment 사용하기

Deployment

- 애플리케이션을 배포하고 선언적으로 업데이트하기 위한 높은 수준의 리소스.
- 디플로이먼트를 사용하는 경우 실제 파드는 디플로이먼트가 아닌 디플로이먼트의 Replica Set에 의해 생성되고 관리된다.



▲ 그림 9.8 파드를 감시하는 레플리카셋이 디플로이먼트를 지원한다.

- 롤링 업데이트 경우, 애플리케이션을 업데이트할 때는 추가 Replication Controller를 도입하고 두 컨트롤러가 잘 조화되도록 조정해야 한다 ⇒ Deployment

Deployment 생성

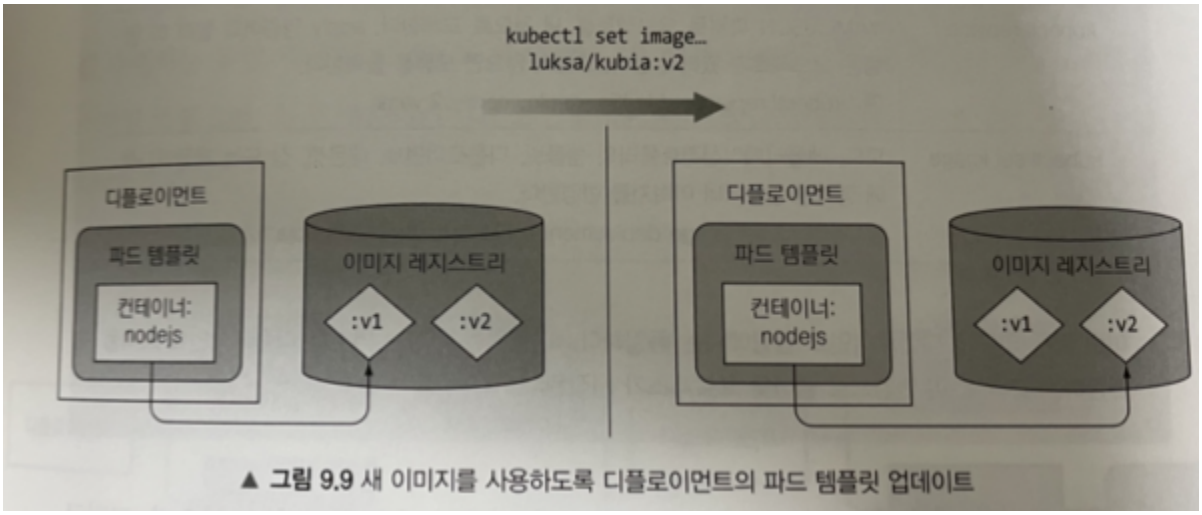
```
# kubia-deployment-v1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubia   # 디플로이먼트 이름에 버전을 포함할 필요가 없다.
```

```
spec:
  replicas: 3
  template:
    metadata:
      name: kubia
      labels:
        app: kubia
    spec:
      containers:
      - image: luksa/kubia:v1
        name: nodejs
```

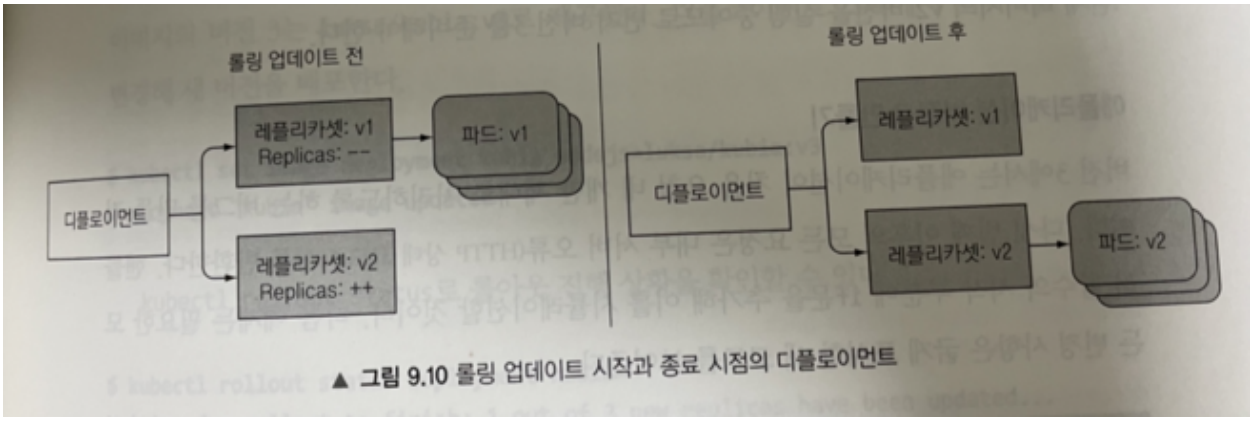
```
kubectl create -f kubia-deployment-v1.yaml --record
```

- `create`를 사용할 때는 반드시 `--record` 옵션을 포함시켜 개정 이력(revision history)에 명령어를 기록하도록 한다.

```
# Deployment 롤아웃 상태 출력
kubectl rollout status deployment kubia
```



▲ 그림 9.9 새 이미지를 사용하도록 디플로이먼트의 파드 템플릿 업데이트



▲ 그림 9.10 롤링 업데이트 시작과 종료 시점의 디플로이먼트

- 추가 레플리카셋이 생성됐고, 그 후 천천히 스케일 업했으며, 이전 레플리카셋의 크기를 0으로 스케일 다운했다.
- 롤링 업데이트 프로세스와는 다르게, 기존 레플리카셋도 여전히 존재된다. ⇒ 롤백 가능

Deployment 전략

1. RollingUpdate (기본)
    - 이전 파드를 하나씩 제거하고 동시에 새 파드를 추가.
2. Recreate
    - 한 번에 기존 모든 파드를 삭제한 뒤 새로운 파드를 만든다.
    - 애플리케이션이 여러 버전을 병렬로 실행하는 것을 지원하지 않고 새 버전을 시작하기 전에 이전 버전을 완전히 중지해야 하는 경우 사용.
    - 짧은 서비스 다운타임 발생.

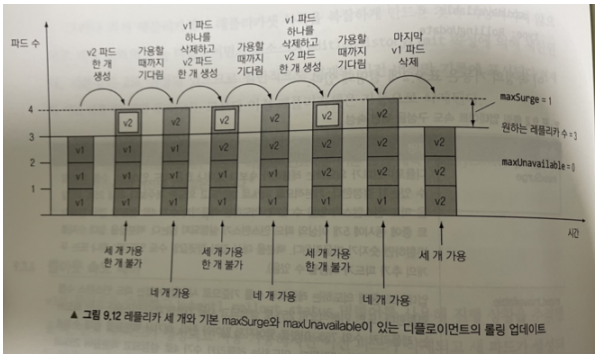
Deployment Rollback

```
# 디플로이먼트를 이전 버전으로 롤백하기
kubectl rollout undo deployment kubia

# 특정 디플로이먼트 개정(버전)으로 롤백하기
kubectl rollout undo deployment kubia --to-revision=1
```

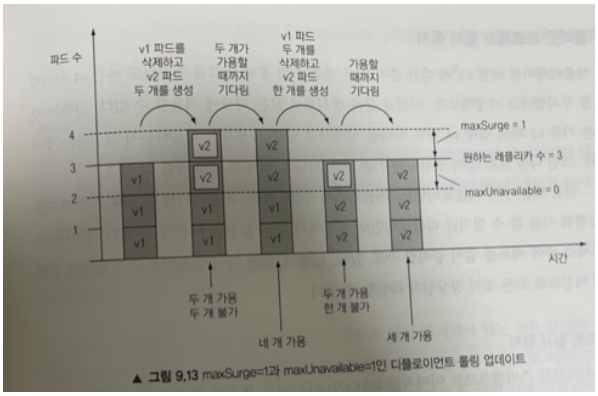
- 개정 내역(revision history)의 수는 디플로이먼트 리소스의 editionHistoryLimit 속성에 의해 제한된다. 기본값은 10(apps/v1)로 설정돼 있으므로 일반적으로 현재와 이전 버전만 기록에 표시된다.

Rollout 속도 제어

- **maxSurge**
  - 디플로이먼트가 의도하는 레플리카 수보다 얼마나 많은 파드 인스턴스 수를 허용할 수 있는지.
- **maxUnavailable**
  - 업데이트 중에 의도하는 레플리카 수를 기준으로 사용할 수 없는 파드 인스턴스 수.



▲ 그림 9.12 레플리카 세 개와 기본 maxSurge와 maxUnavailable이 있는 디플로이먼트의 롤링 업데이트



▲ 그림 9.13 maxSurge=1과 maxUnavailable=1인 디플로이먼트 롤링 업데이트

Rollout 프로세스 일시 중지

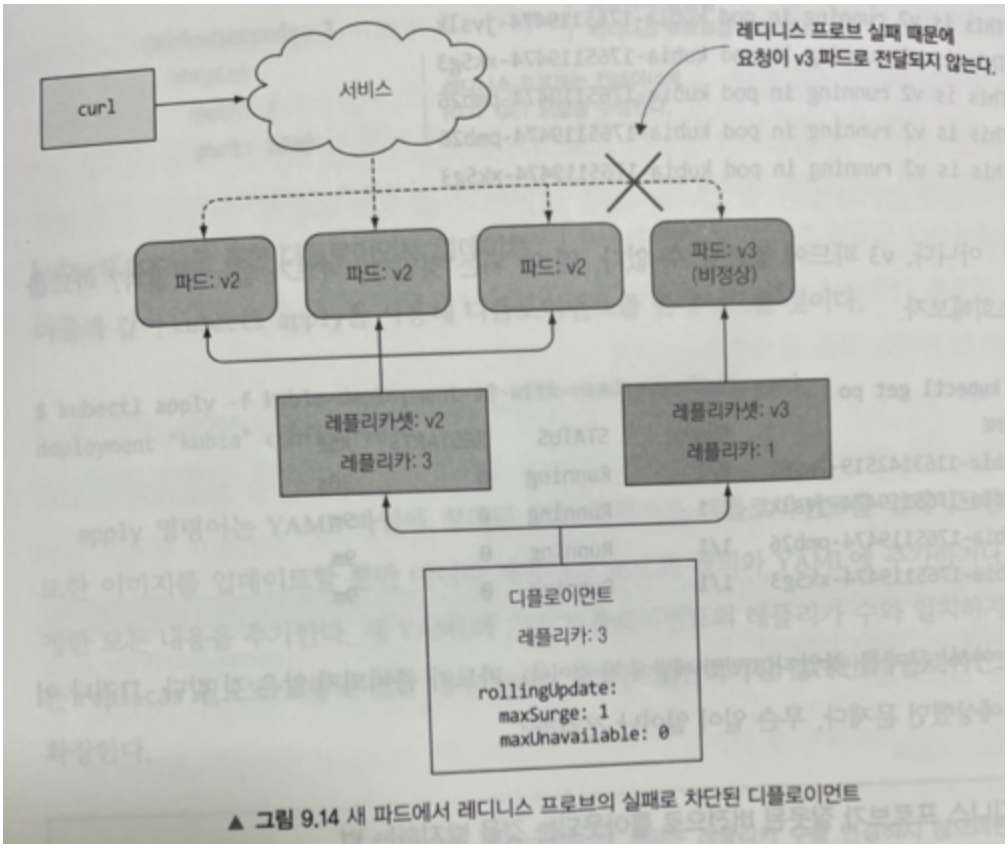
```
# 롤아웃 일시 정지
kubectl rollout pauser deployment kubia

# 롤아웃 재개
kubectl rollout resume deployment kubia
```

- 카나리 배포 효과적 사용 가능.
- 새 버전을 모든 사람에게 롤아웃하는 대신 하나 또는 적은 수의 이전 파드만 새 버전으로 바꾼다.

잘못된 버전의 롤아웃 방지

- **minReadySeconds**
  - 오작동 버전의 배포를 방지.

- 적절한 Readiness Probe와 minReadySeconds 설정으로 잘못된 애플리케이션 버전 배포 방지.
- 새 파드가 시작되자마자 Readiness Probe가 매초마다 시작된다. Readiness Probe가 실패하면 결과적으로 파드는 서비스의 엔드포인트에서 제거된다. 새 파드를 사용할 수 없으므로 롤아웃 프로세스가 계속되지 않고 배포가 중단된다.



▲ 그림 9.14 새 파드에서 레디니스 프로브의 실패로 차단된 디플로이먼트

- 기본적으로 롤아웃이 10분동안 진행되지 않으면 실패한 것으로 간주된다. Deployment가 실패한 것으로 간주되는 시간은 Deployment 스펙의 profressDeadlineSeconds 속성으로 설정할 수 있다.
- 롤아웃이 계속 진행되지 않기 때문에 rollout undo를 사용하여 롤아웃을 취소해 중단한다.

💡 **kubectl patch** 명령어는 텍스트 편집기에서 정의를 편집하기 않고도 리소스 속성 한 두 개 정도를 수정하는 데 유용하다.

하나의 YAML 파일에 여러 리소스를 정의하려면 대시 3개를 구분 기호로 사용한다.

보이지 않는 곳에서 정확히 무엇을 하는지 확인하려면 kubectl의 자세한 로깅옵션(—v 옵션)을 켠다.