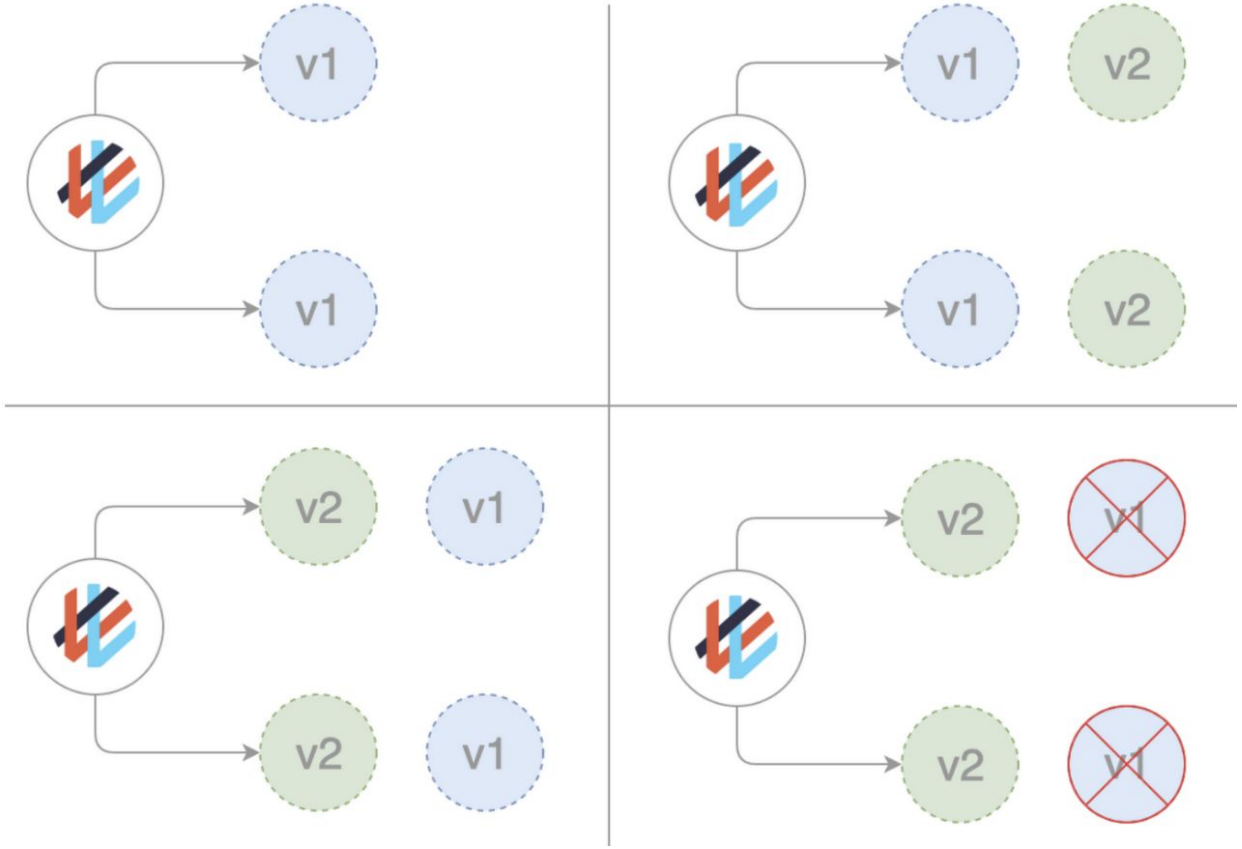
Deployments

Deployment 란?

- Deployment는 Pod 와 ReplicaSet 에 대한 **선언적 업데이트 제공하며, 배포에 대한 세분화된 기능 제공**
- Deployment 에 의도하는 상태를 기술하면, Deployment Controller는 현재 상태에서 의도하는 상태로 비율을 조정함
-



출처 : https://www.weave.works/blog/kubernetes-deployment-strategies

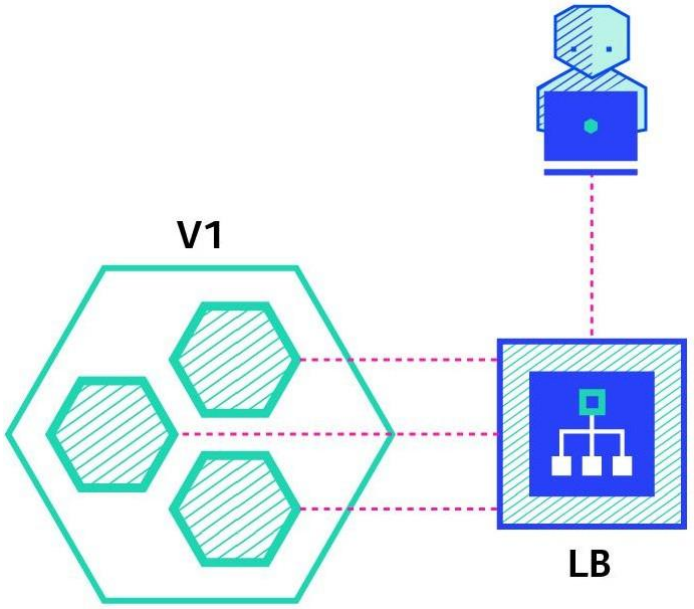
Deployment strategy

- Recreate *native*
- Ramped *native*
- Blue/Green *extra step needed*
- Canary *extra step needed*
- A/B Testing *require additional component*
- Shadow *require additional component*

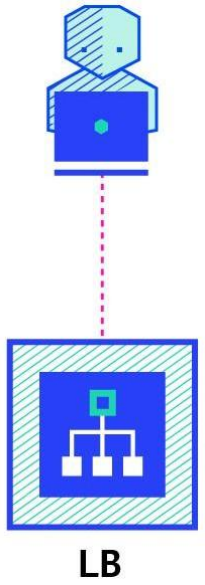
Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

Recreate

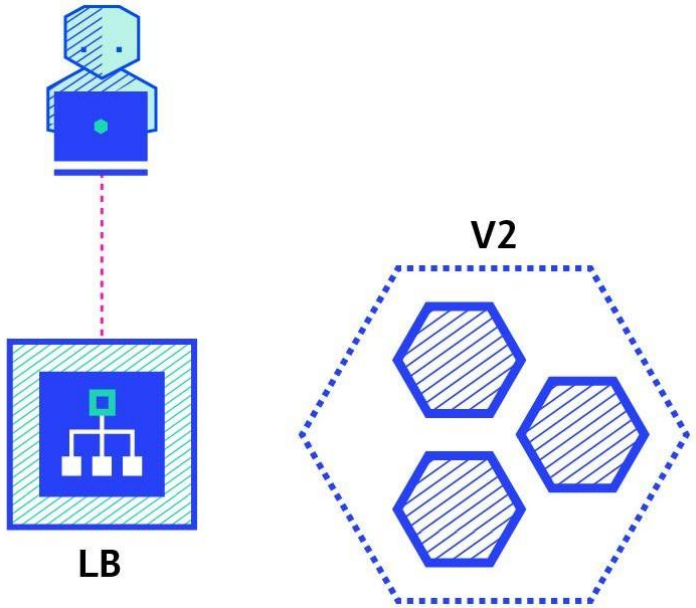
Deployment strategy - Recreate



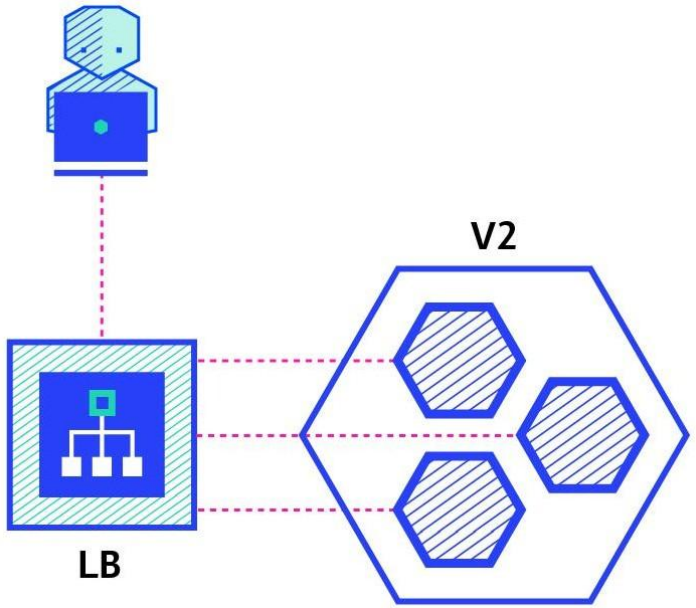
Deployment strategy - Recreate



Deployment strategy - Recreate

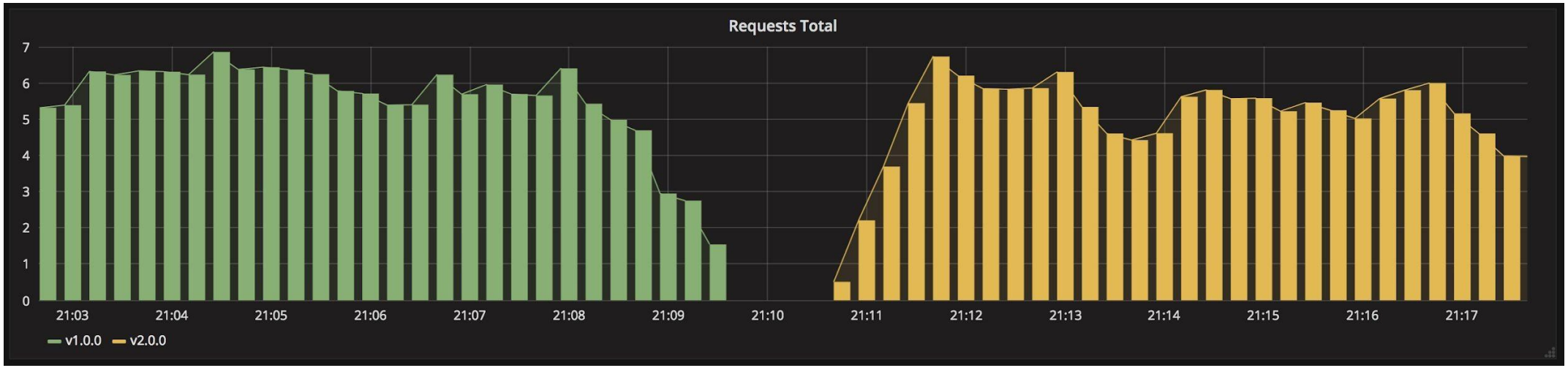


Deployment strategy - Recreate



Deployment strategy – Recreate

변경 시 트래픽 패턴



서비스 중단

Deployment strategy

장점:

- 설정 쉬움

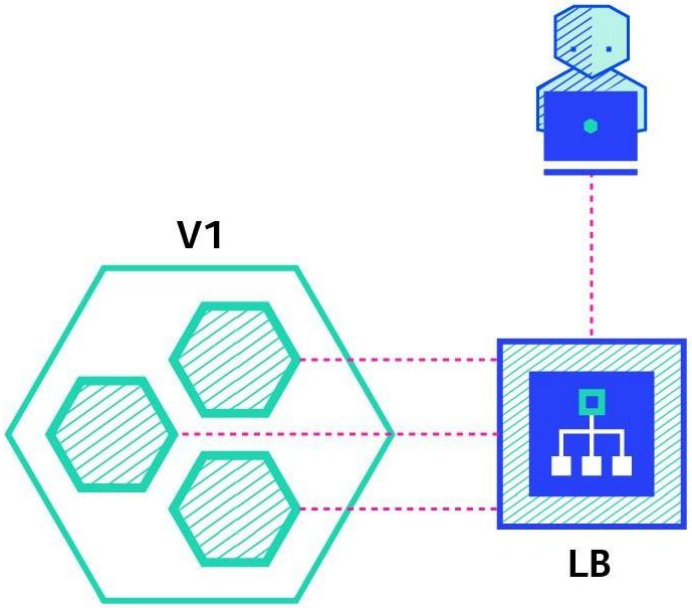
단점:

- 서비스 영향이 큼, 어플리케이션의 종료/시작 간격에 따라 서비스 다운타임이 결정됨

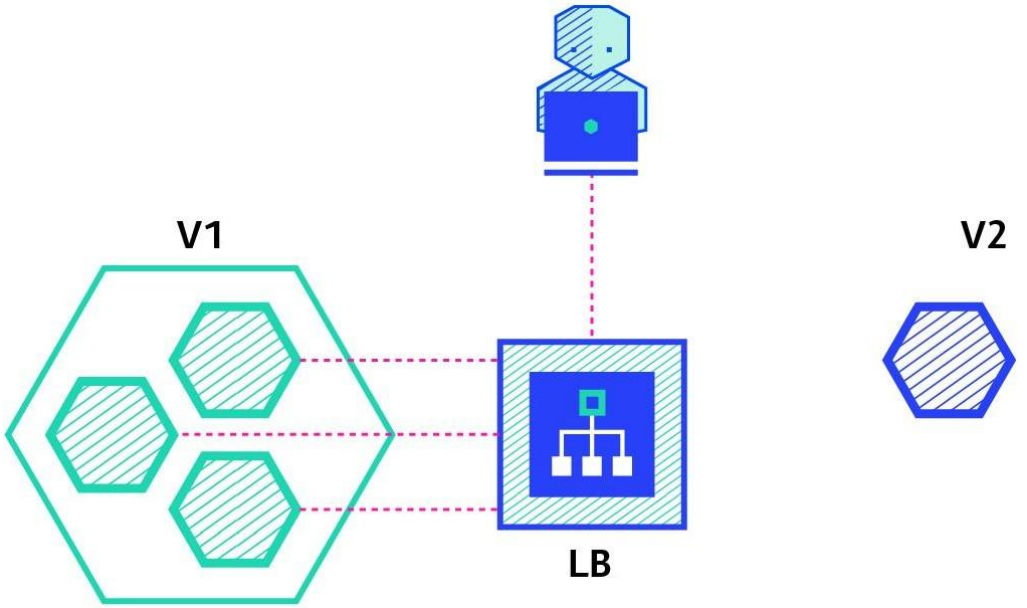
Ramped

Aka incremental, rolling update

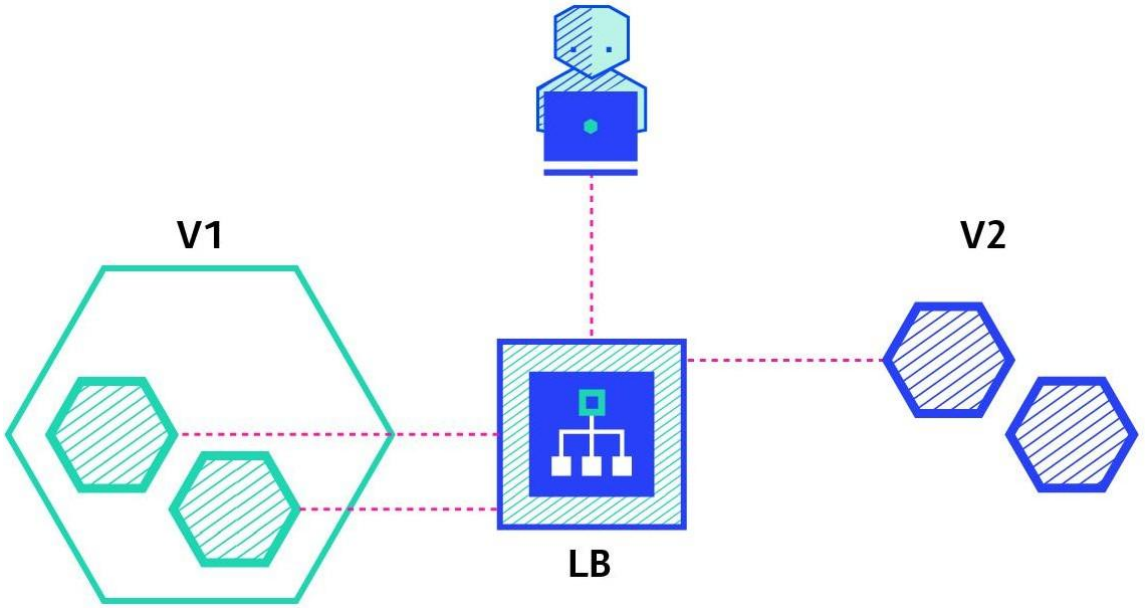
Deployment strategy



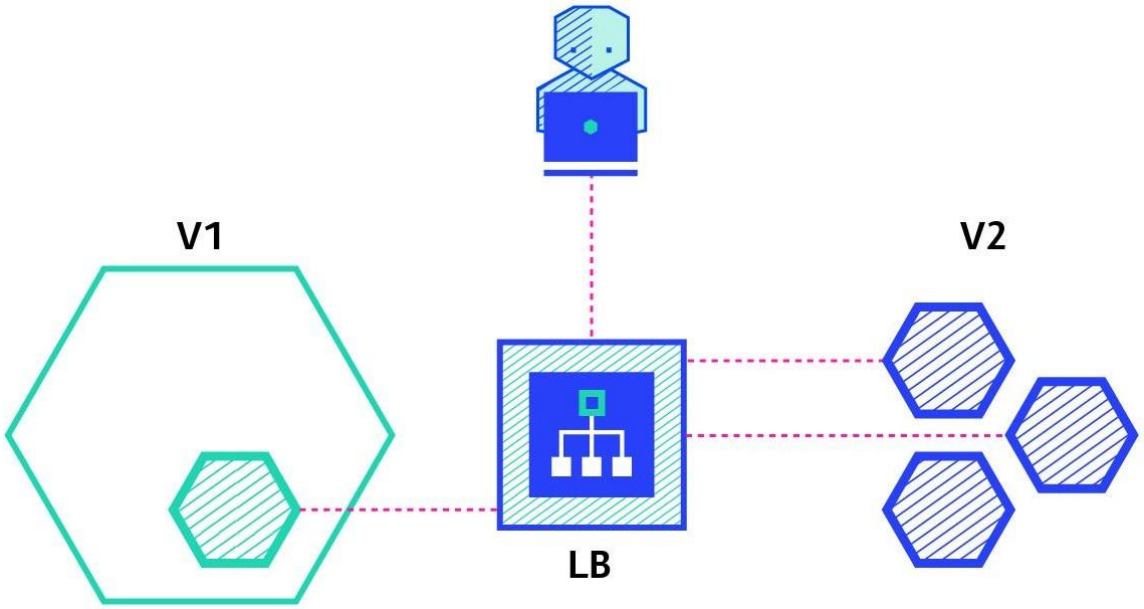
Deployment strategy



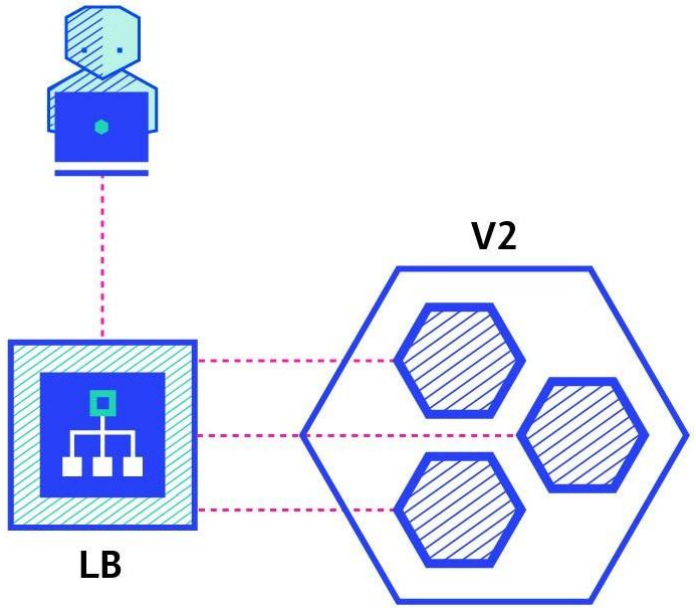
Deployment strategy



Deployment strategy



Deployment strategy

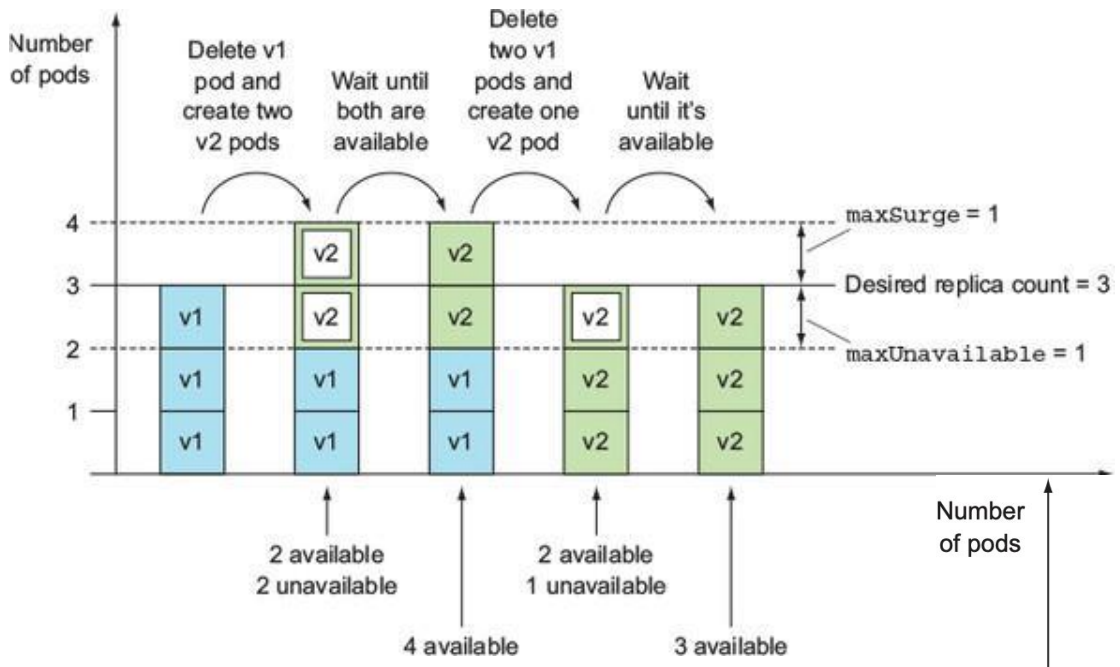


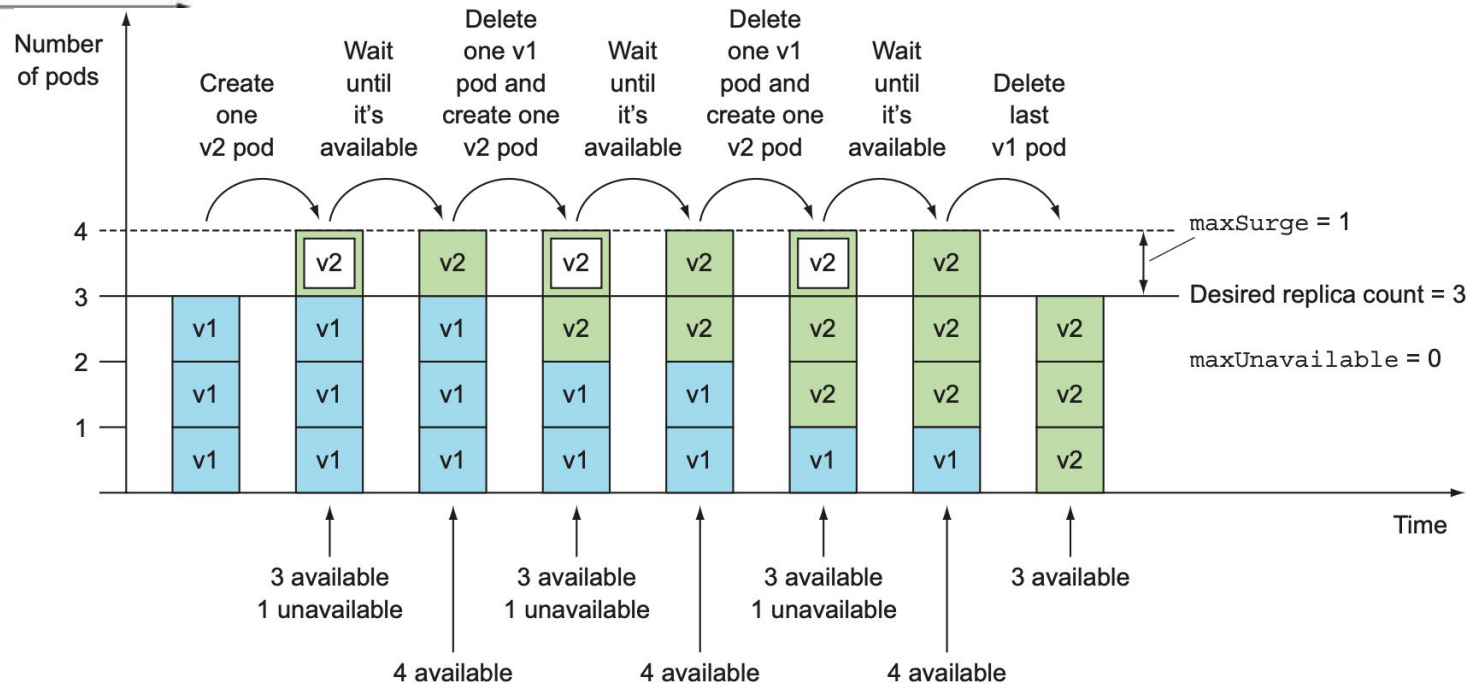
Deployment strategy

```
[...]
kind: Deployment
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2          # how many pods we can add at a time
      maxUnavailable: 0    # maxUnavailable define how many pods can be
                           # unavailable during the rolling update
[...]
```

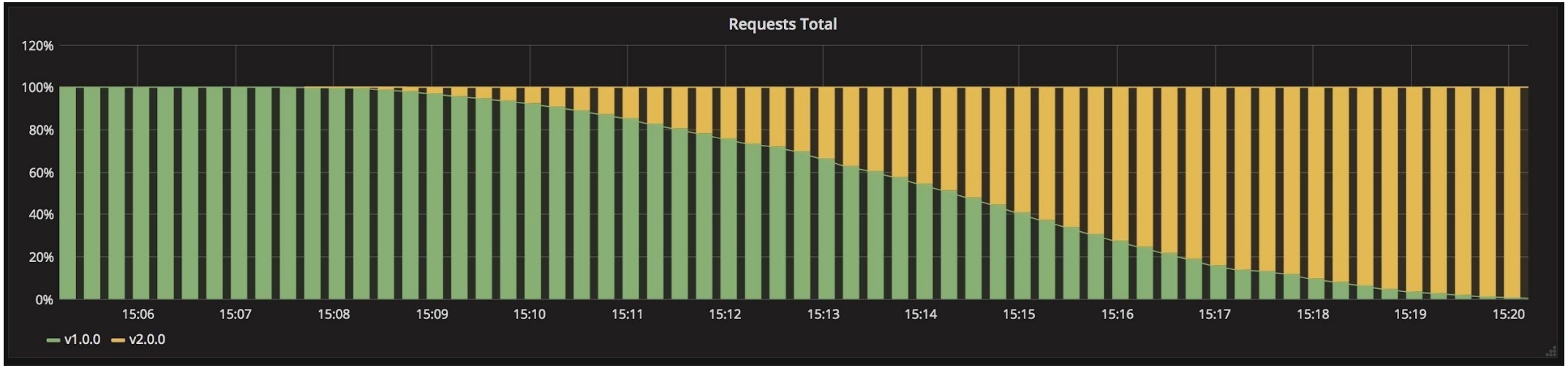
```
$ kubectl apply -f ./manifest.yaml
```

Deployment 란?





Deployment strategy



Deployment strategy

장점:

- 설정 및 사용 쉬움
- 전체 인스턴스들에 신규 버전이 천천히 배포됨
- 데이터 Rebalancing 이 진행되는 상태유지 애플리케이션에 유리

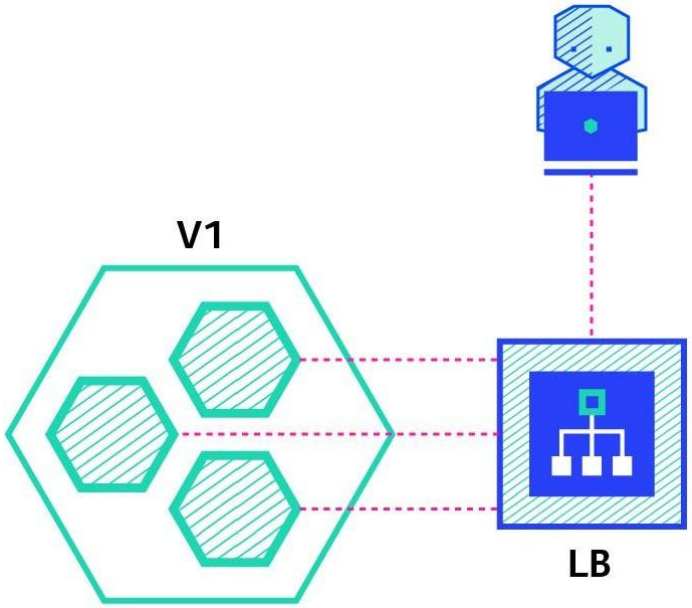
단점:

- 롤아웃/롤백 시간이 오래걸림
- 트래픽에 대한 통제는 어려움

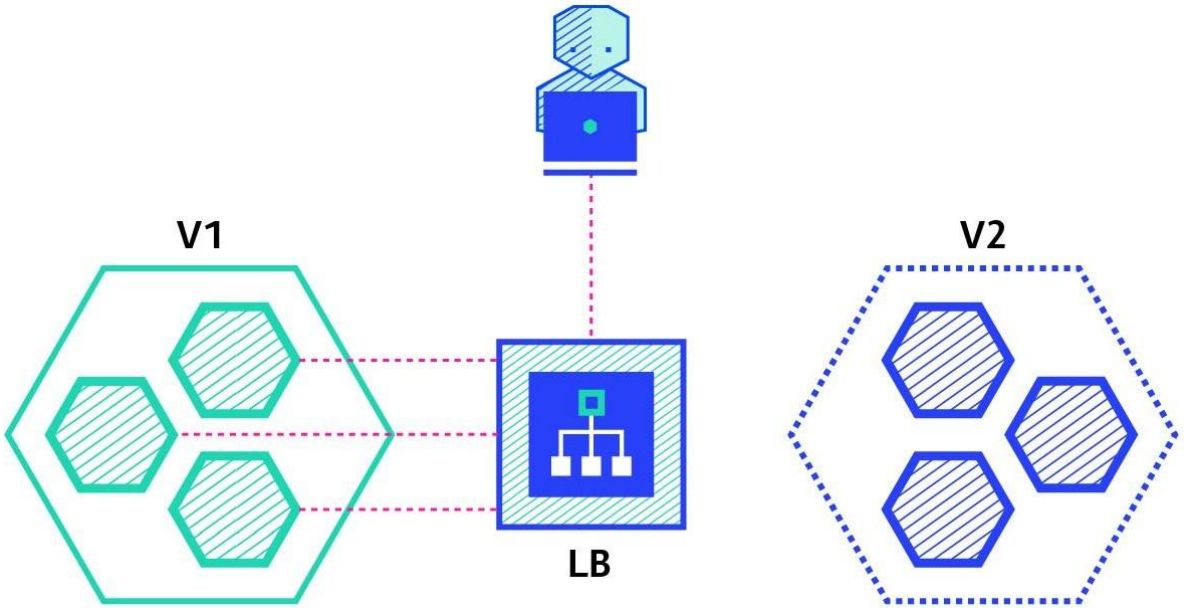
Blue/Green

Aka red/black

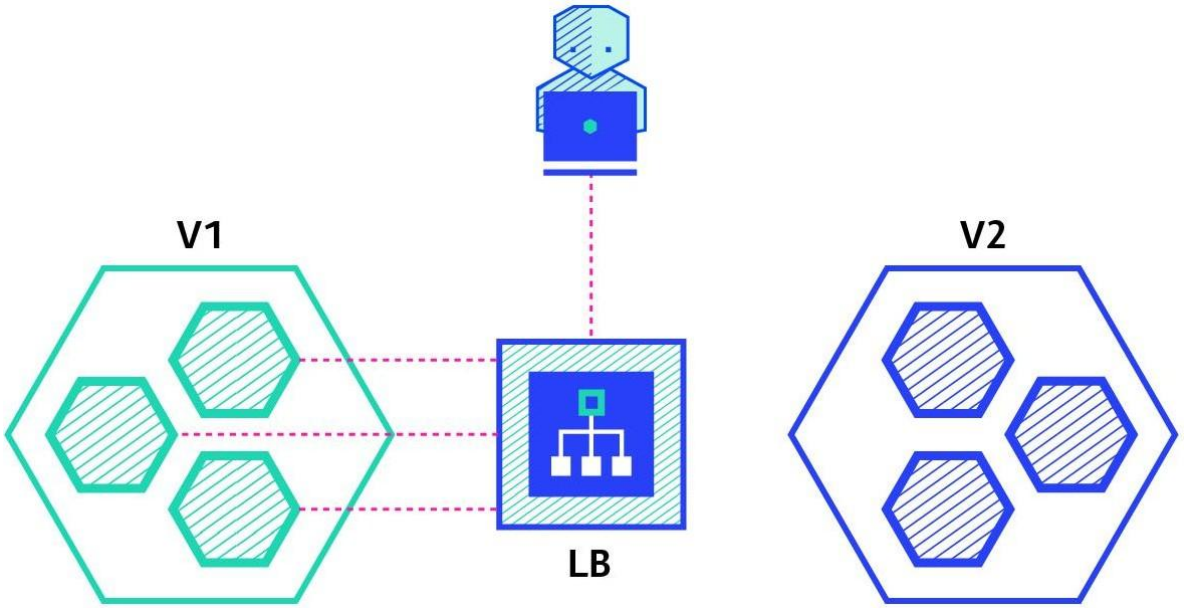
Deployment strategy



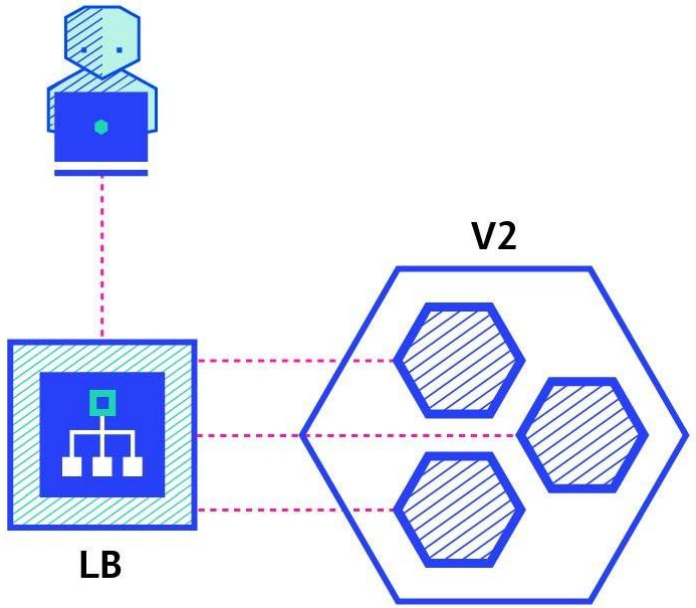
Deployment strategy



Deployment strategy



Deployment strategy



Deployment strategy

```
[...]
kind: Service
spec:
  # Note here that we match both the app and the version.
  # When switching traffic, update the label “version” with
  # the appropriate value, ie: v2.0.0
  selector:
    app: my-app
    version: v1.0.0
[...]
```

```
$ kubectl apply -f ./manifest-v2.yaml
$ kubectl patch service my-app -p \
    '{"spec":{"selector":{"version":"v2.0.0"}}}'
$ kubectl delete -f ./manifest-v1.yaml
```

Deployment strategy

```
[...]
kind:
Ingress
spec:
  rules:
  - host:
    login.domain.com
    http:
      paths:
      - backend:
          serviceName: login-v2
          servicePort: 80
  - host:
    cart.domain.com
    http:
      paths:
      - backend:
          serviceName: cart-v2
          servicePort: 80
[...]
```

```
[...]
kind:
Service
metadata:
  name: login-v2
spec:
  selector:
    app:
    login
    version:
v2.0.0  [...]
```

```
[...]
kind:
Service
metadata:
  name: cart-v2
spec:
  selector:
    app:
    cart
    version:
v2.0.0  [...]
```

```
$ kubectl apply -f ./manifest-v2.yaml
$ kubectl apply -f ./ingress.yaml
$ kubectl delete -f ./manifest-v1.yaml
```

Deployment strategy

Deployment strategy

장점:

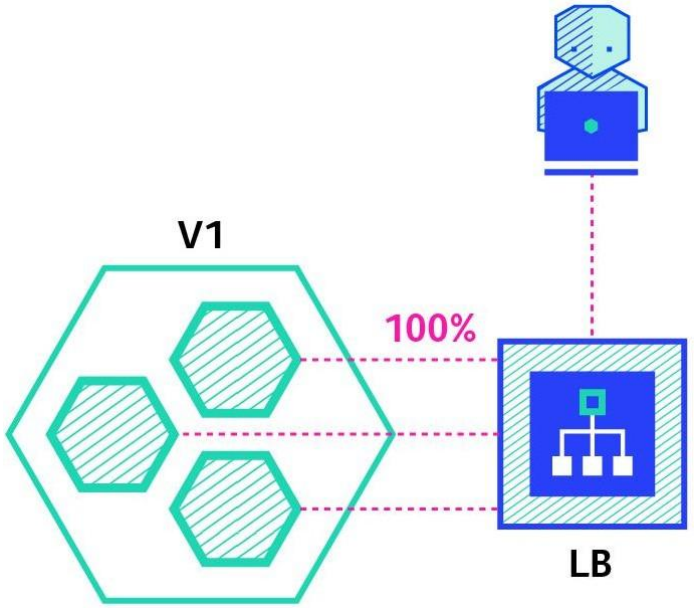
- 간단한 rollout/rollback
- 버전이 자주 변경되는 프론트엔드 서비스에 적합
- 응용 프로그램 종속성 지옥을 수정하는 더러운 방법

단점:

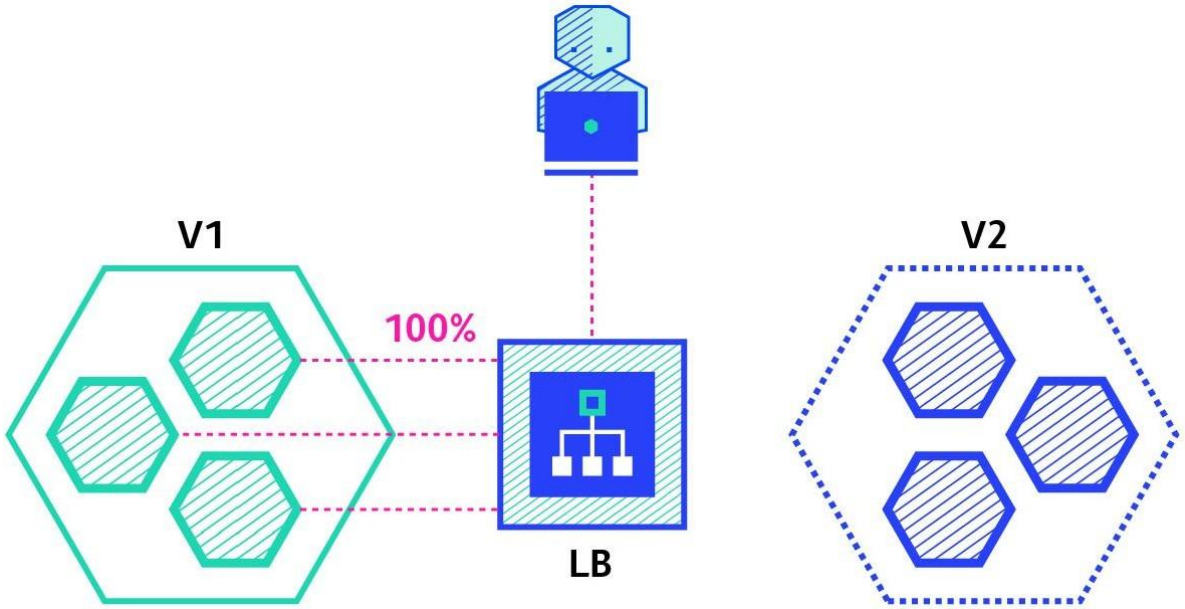
- 2배의 리소스가 필요해서 비용이 비쌈.
- 프로덕션에 릴리즈 하기 전에 전체 플랫폼에 대한 테스트 수행필요.

Canary

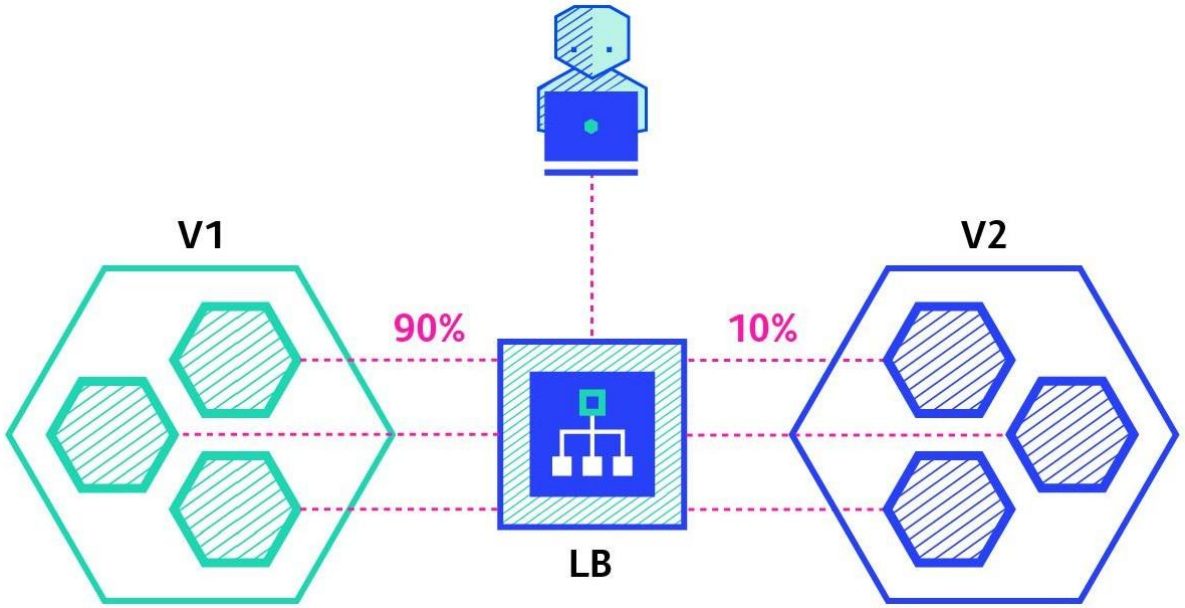
Deployment strategy



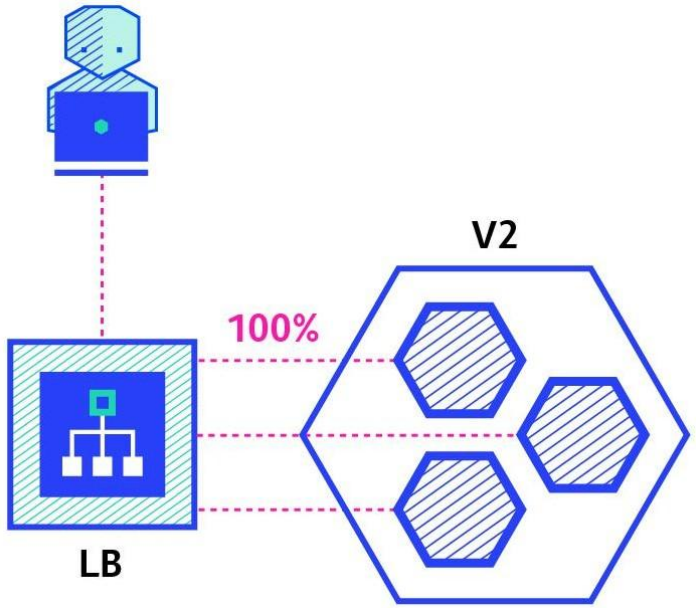
Deployment strategy



Deployment strategy



Deployment strategy



Deployment strategy

```
[...]
kind: Deployment
metadata:
  name: my-app-v1
spec:
  replicas: 9
  template:
    labels:
      app: my-app
      version: v1.0.0
[...]
```

```
[...]
kind: Deployment
metadata:
  name: my-app-v2
spec:
  replicas: 1
  template:
    labels:
      app: my-app
      version:
      v2.0.0
[...]
```

```
[...]
kind: Service
metadata:
  name: my-app
spec:
  selector:
    app: my-app
[...]
```

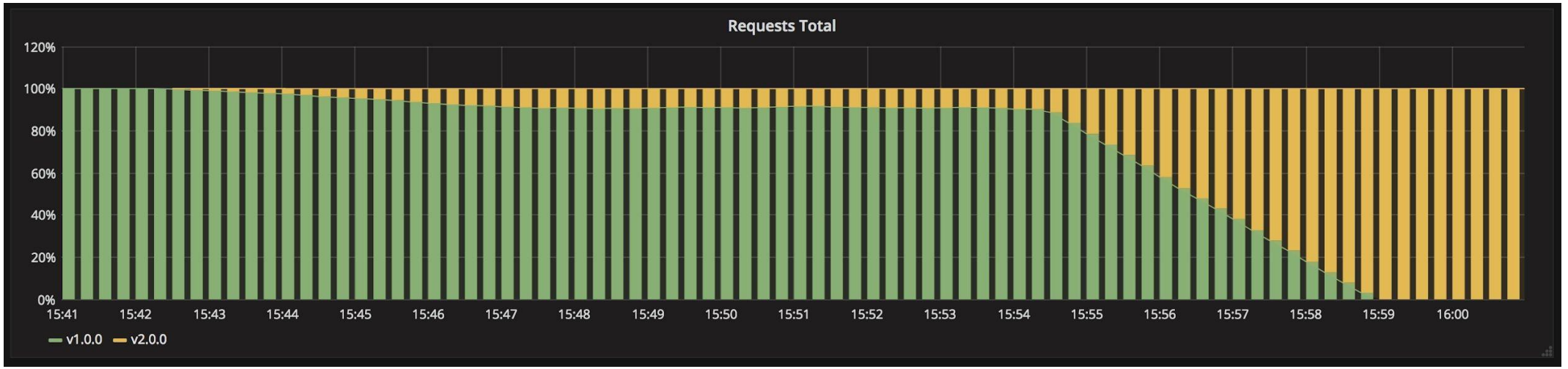
```
$ kubectl apply -f ./manifest-v2.yaml
$ kubectl scale deploy/my-app-v2 --replicas=10
$ kubectl delete -f ./manifest-v1.yaml
```

Deployment strategy

```
[...]
kind:
RouteRule
metadata:
  name:
my-app  spec:
  destination:
    name:
    my-app
  route:
  - labels:                 # 90%
      version:              traffic
      v1.0.0
    weight: 90              # 10%
  - labels:                 traffic
      version:
      v2.0.0
    weight: 10
[...]
```

```
$ kubectl apply -f ./manifest-v2.yaml
$ kubectl apply -f ./routerule.yaml
```

Deployment strategy



Deployment strategy

장점:

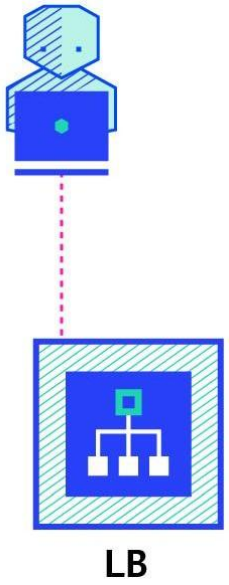
- 일부 사용자를 위해 출시된 버전을 사용할 경우
- 오류율 및 성능 모니터링에 편리
- 비교적 빠른 롤백

단점:

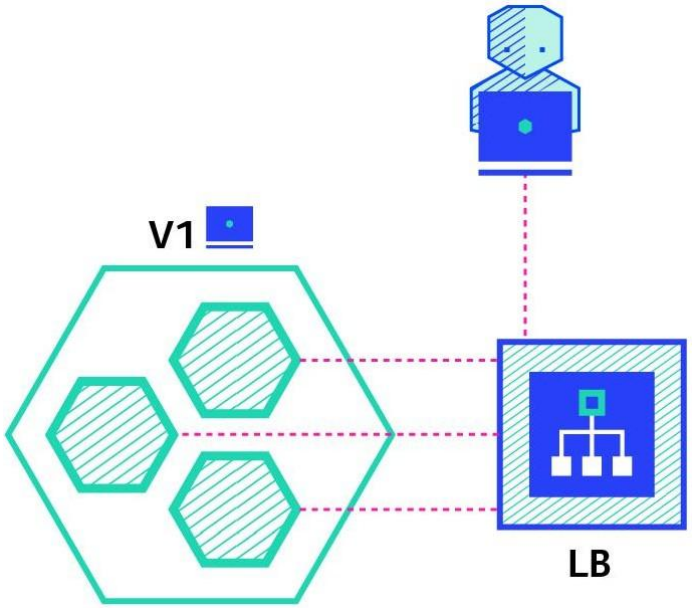
- 느린 롤아웃 (일부 트래픽에 대해 검증해야 하기 때문에)
- sticky sessions 필요
- 정확한 트래픽 이동에는 Istio 또는 Linkerd와 같은 추가 도구가 필요.

A/B Testing

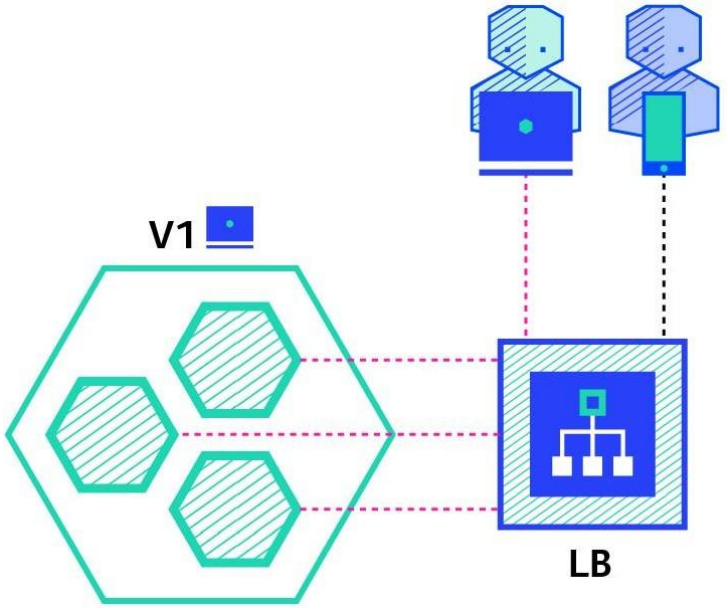
Deployment strategy



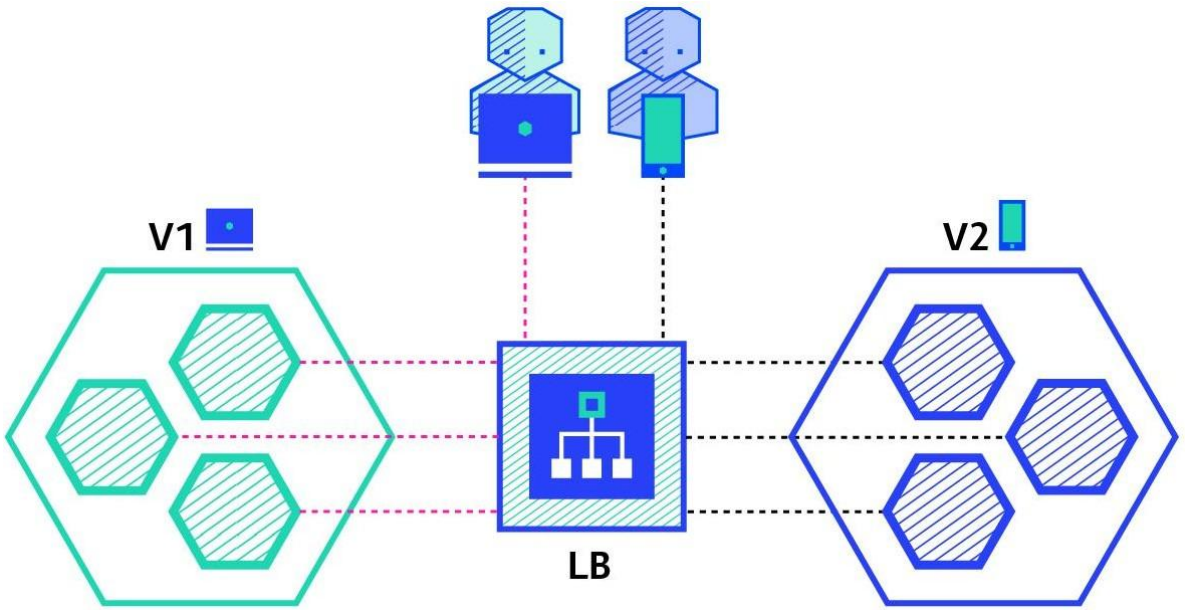
Deployment strategy



Deployment strategy



Deployment strategy



Possible conditions:

- *Geolocalisation*
- *Language*
- *Cookie*
- *User Agent (device, OS, etc.)*
- *Custom Header*
- *Query parameters*

Deployment strategy (with Istio)

```
[...]
kind:
RouteRule
metadata:
  name:
my-app-v1  spec:
  destination:
    name:
    my-app
  route:
  - labels:
      version: v1.0.0
  match:
    request:
      headers
      :
        x-api-version:
          exact: "v1.0.0"
[...]
```

```
[...]
kind:
RouteRule
metadata:
  name:
my-app-v2  spec:
  destination:
    name:
    my-app
  route:
  - labels:
      version: v2.0.0
  match:
    request:
      headers
[...] :
        x-api-version:
          exact: "v2.0.0"
```

```
$ kubectl apply -f ./manifest-v2.yaml
$ kubectl apply -f ./routerule.yaml
```

Deployment strategy

Deployment strategy

장점:

- 여러가지 버전의 어플리케이션을 동시 서비스
- 트래픽 분배에 대한 완전한 제어 가능
- 전환율을 개선하기 위해 비즈니스 목적으로 사용할 수 있는 훌륭한 도구

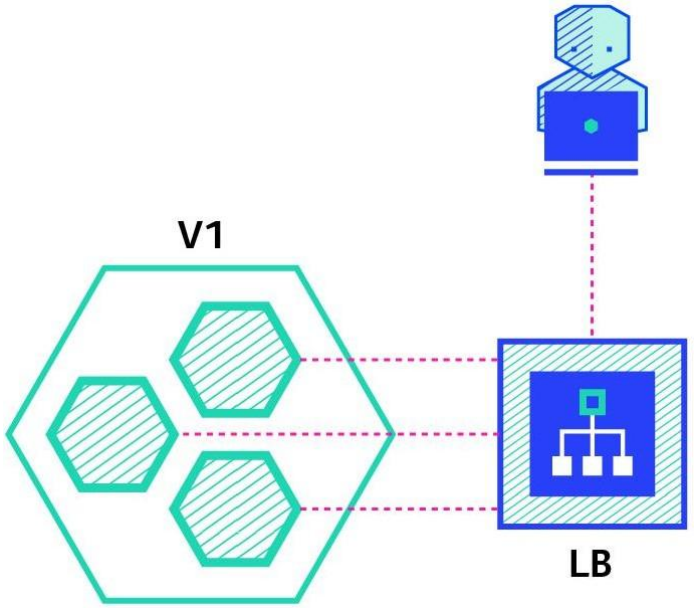
단점:

- 지능형 로드 밸런서 필요(Istio, Linkerd 등)
- 세션에 대한 오류를 해결하기 어렵기 때문에 분산 추적 기술이 필수.

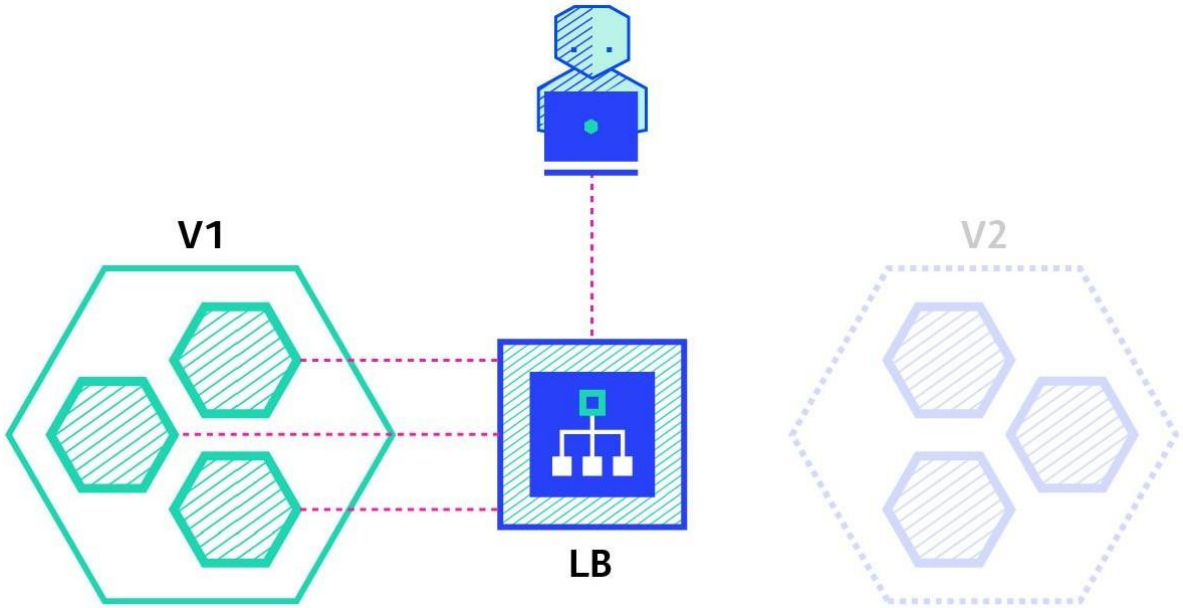
Shadow

Aks Mirrored, Dark

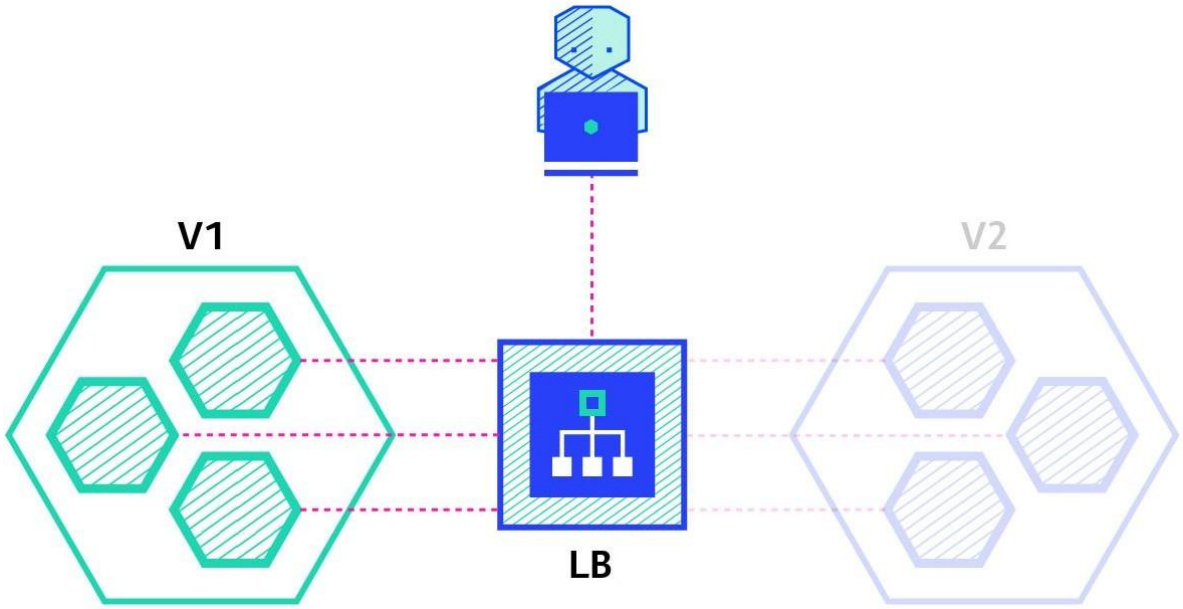
Deployment strategy



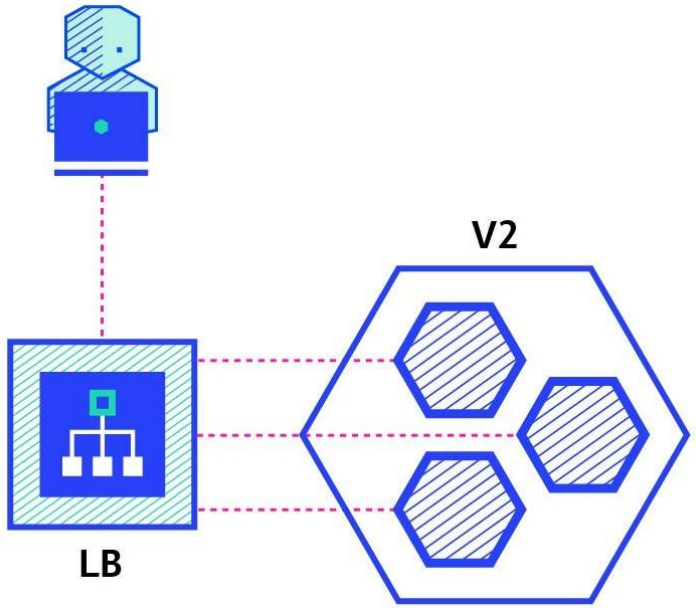
Deployment strategy



Deployment strategy



Deployment strategy



Deployment strategy (with Istio)

```
[...]
kind:
RouteRule
spec:
  destination:
    name:
    my-app
  route:
  - labels:
      version:
    v1.0.0  weight:
    100
  - labels:
      version: v2.0.0
    weight: 0
  mirror:
    name: my-app-v2
    labels:
      version: v2.0.0
[...]
```

```
$ kubectl apply -f ./manifest-v2.yaml
$ kubectl apply -f ./routerule.yaml
```

Deployment strategy (with Istio)

Deployment strategy (with Istio)

장점:

- 운영계 트래픽에 대한 애플리케이션 성능 테스트에 좋음
- 사용자 영향도 없음
- 애플리케이션의 안정성과 성능이 요구 사항을 충족할 때까지 롤 아웃 없음

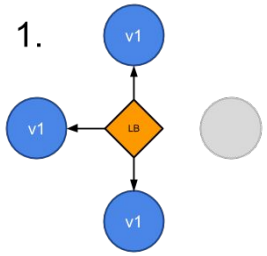
단점:

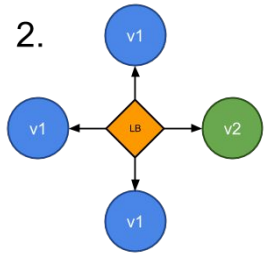
- 설정이 다소 복잡
- 2배의 리소스가 필요해 비용이 비쌈
- 특정 서비스에 대한(결제, 외부채널 연동) mocking/stubbing 서비스가 필요합니다.

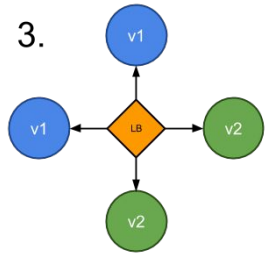
Deployment strategy - Summary

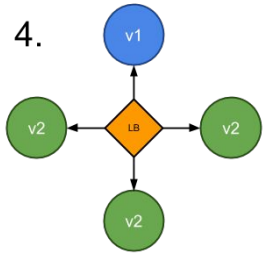
- 다운타임을 허용해도 된다면 **Recreate**
- **recreate** 와 **ramped** 는 Deployment 기능에 설정으로 포함되어 있음.(kubectl apply 로 충분)
- **ramped** 와 **blue/green** 배포는 일반적으로 사용성이 좋고 적용하기 쉬움.
- **blue/green** 배포는동일한 서버에서 버전이 변경되는 어플리케이션을 로드하는 프런트 엔드에 적합.
- **blue/green** 와 **shadow** 는 추가적인 인스턴스를 필요로 하기때문에 비용이 비쌈.
- **canary** 와 **A/B Testing** 릴리즈에 대한 확신이 없거나, 대규모 업데이트일 때 사용하기 좋음
- **canary**, **A/B Testing** , **shadow** 모두 Istio 와 같은 추가적인 컴포넌트가 필요 합니다.

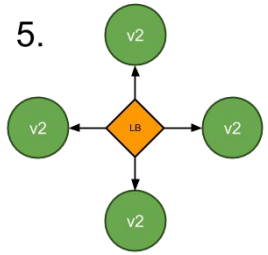
Deployment strategy - Quiz

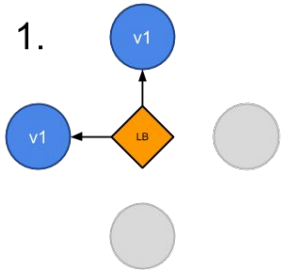


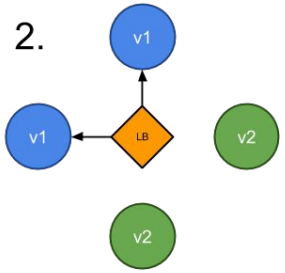


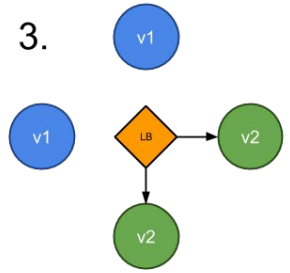


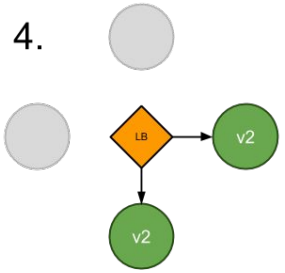


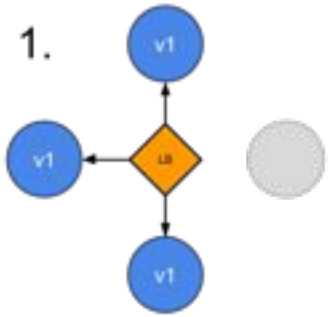


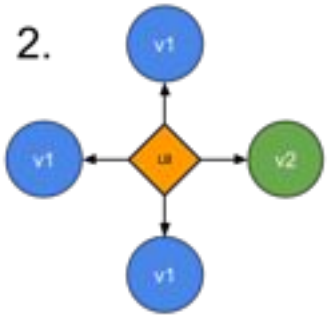


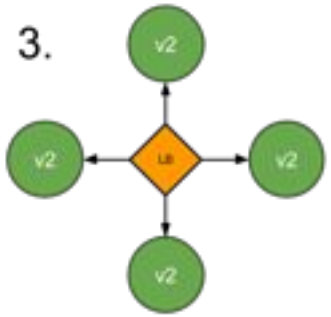


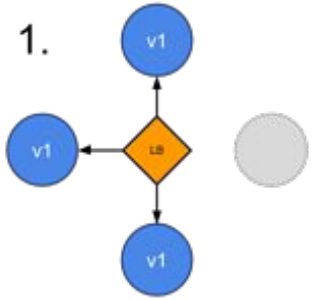












Deployment Strategy

- spec.strategy.type==Recreate 이면 새 파드가 생성되기 전에 죽는다.
- spec.strategy.type==RollingUpdate 이면 파드를 롤링 업데이트 방식으로 업데이트. 한다.
- 기본은 RollingUpdate

Deployment 생성

- spec.template 에는 배포할 Pod를 기술
- Label 은 관리 대상 Pod를 정의 함
- Replicas 는 복제 노드를 의미하며 내부적으로 ReplicaSet 이 생성

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

셀렉터는 다른 컨트롤러(다른 디플로이먼트와 스테이트풀 셋 포함)와 겹치지 않아야 한다. 쿠버네티스는 겹치는 것을 막지 않으며, 만약 다중 컨트롤러가 겹치는 셀렉터를 가지는 경우 해당 컨트롤러의 충돌 또는 예기치 않은 동작을 야기할 수 있다.

Deployment Sclaout

- Deployment 로 배포 후에는 ReplicaSet 을 직접 관리 하지 않고 deployment 통해서 제어 합니다.

```
$ kubectl edit deploy <deploy-name>

$ kubectl scale deploy <deploy-name> -replicas=<replica-count>
```