TKS Cloud Service

Day2: 어플리케이션 개발자가 알아야 할 쿠버네티스 기술 이야기

(feat. 쿠버네티스 기반 앱 배포 시스템 만들기)

SK telecom

Container Solution개발팀 안승규

Who am I

- **안승규 (ahnsk@sk.com)**

  SK텔레콤 Senior Software Engineer

  Kubernetes Korea Group 리더

- Java Enterprise Application 개발
- OpenStack 기반 Private Cloud 개발
- Kubernetes 기반 플랫폼 개발
- Istio 기반 MSA 플랫폼 개발
- ML Model Serving 플랫폼 개발

Discord 어플 다운로드 (@App Store, Google Play)

오늘 발표할 내용

- ✓ Image / Container 기본 개념 (cgroup, namespace, Immutable)
- ✓ Pod (이미지, 리소스, 스케줄링, 정상상태 점검)
- ✓ Controller (Reconcile)
- ✓ ReplicaSet, Deployment (Scale out, Rollout)
- ✓ 스토리지 (ConfigMap, Secret, PV, PVC)
- ✓ 네트워크 (외부 접속 - Service, Load Balancer, Ingress Controller)
- ✓ Disruption (인프라는 언제든지 장애가 발생할 수 있다)

지난 4월
Kubernetes Korea Group Meetup 발표

- ✓ **컨테이너 기반 개발/배포 프로세스**
- ✓ **어플리케이션 배포 형상**
- ✓ **배포 전략 (rolling update, blue-green, canary)**
- ✓ **배포 전략 구현 (Argo Rollout)**
- ✓ **배포 프로세스 구현 (Argo Workflow)**

Kubernetes Korea Group 참여 (#발표자료실)

지난 4월 발표 확인

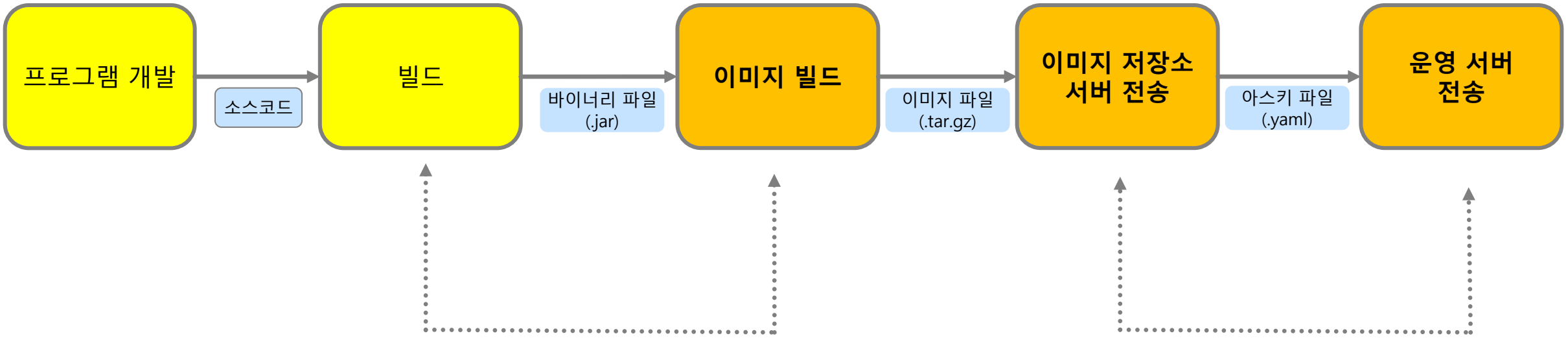
발표자료:
https://devocean.sk.com/vlog/TechDay/04_se3.pdf

발표영상:
https://devocean.sk.com/vlog/view.do?id=420&vcode=A01

컨테이너 기반 개발/배포 프로세스 (CI/CD)

(컨테이너) 이미지 파일 = 바이너리 파일을 포함하고 있는 파일

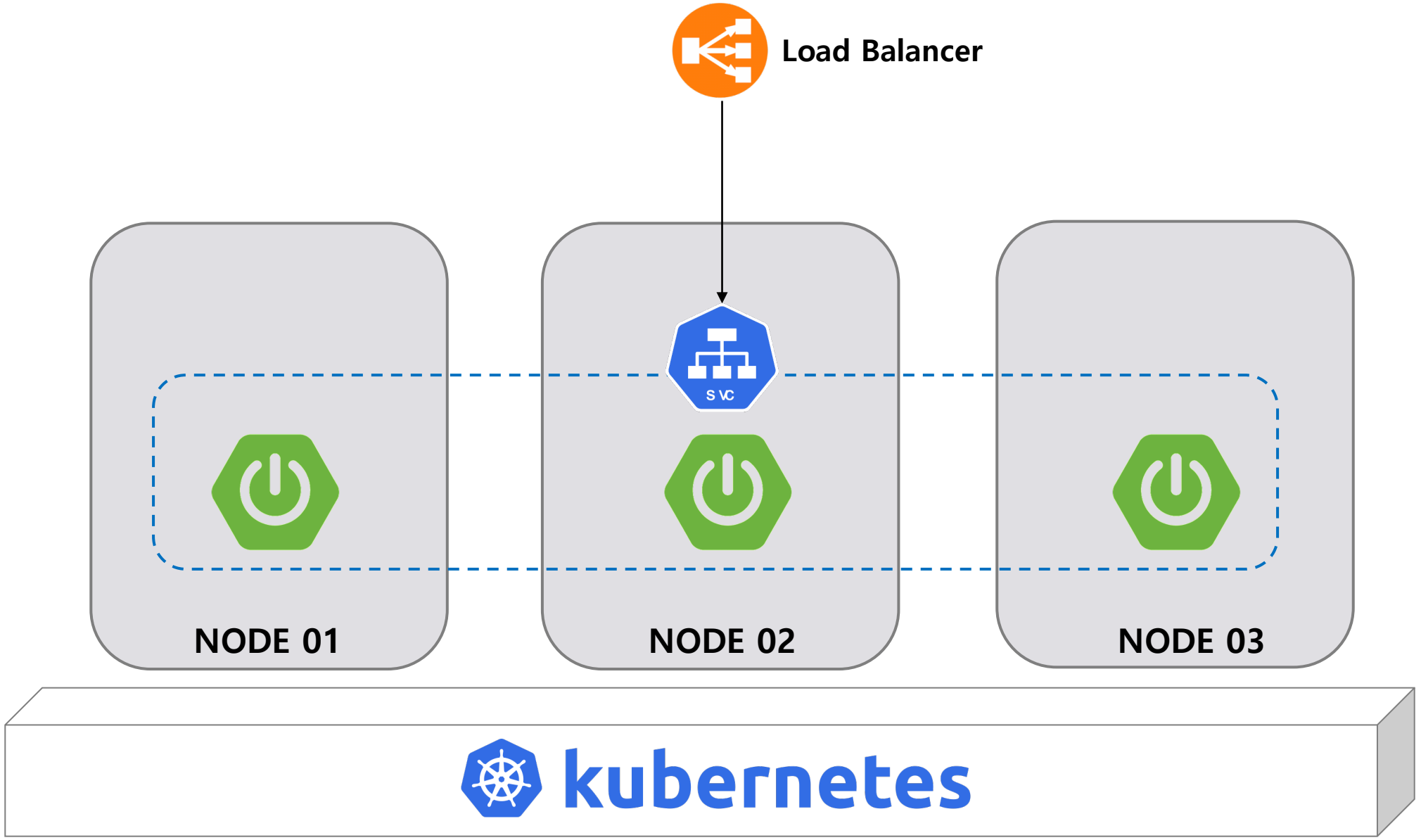
서버 대수와 상관 없이 한 번만 작업



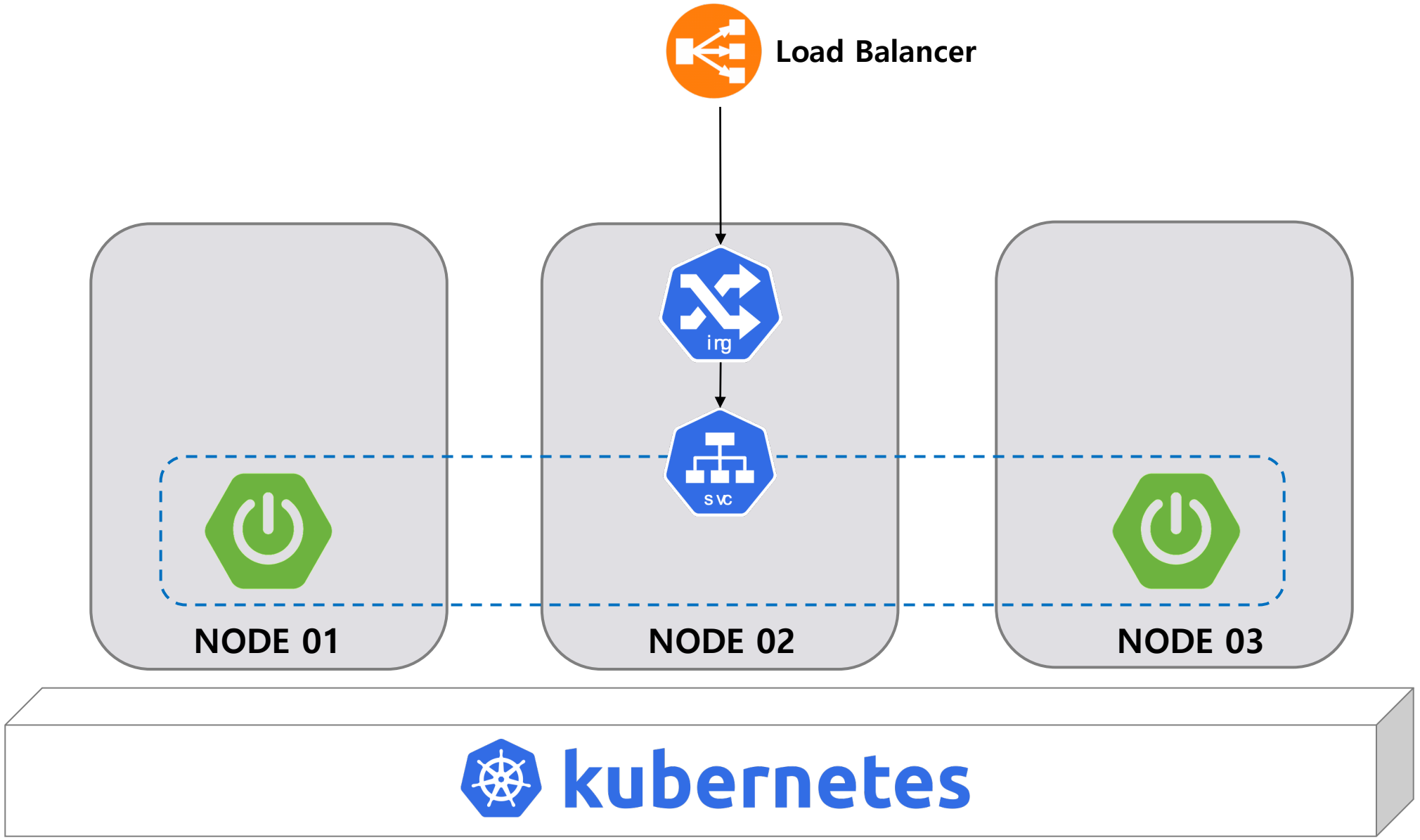
CI
(Continuous Integration)

CD
(Continuous Deployment/Delivery)

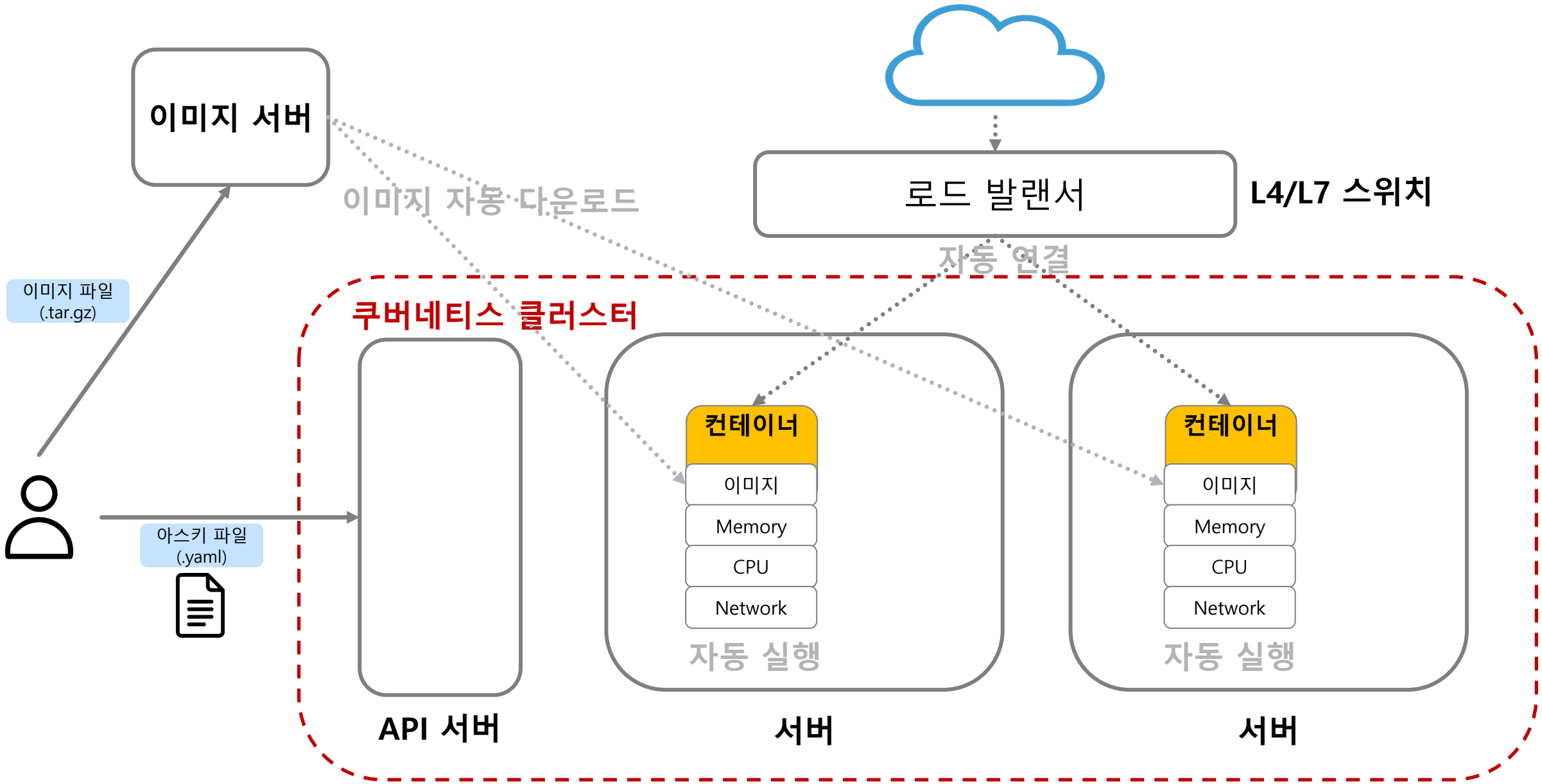
어플리케이션 배포 형상 (Load Balancer)



어플리케이션 배포 형상 (Ingress Controller)



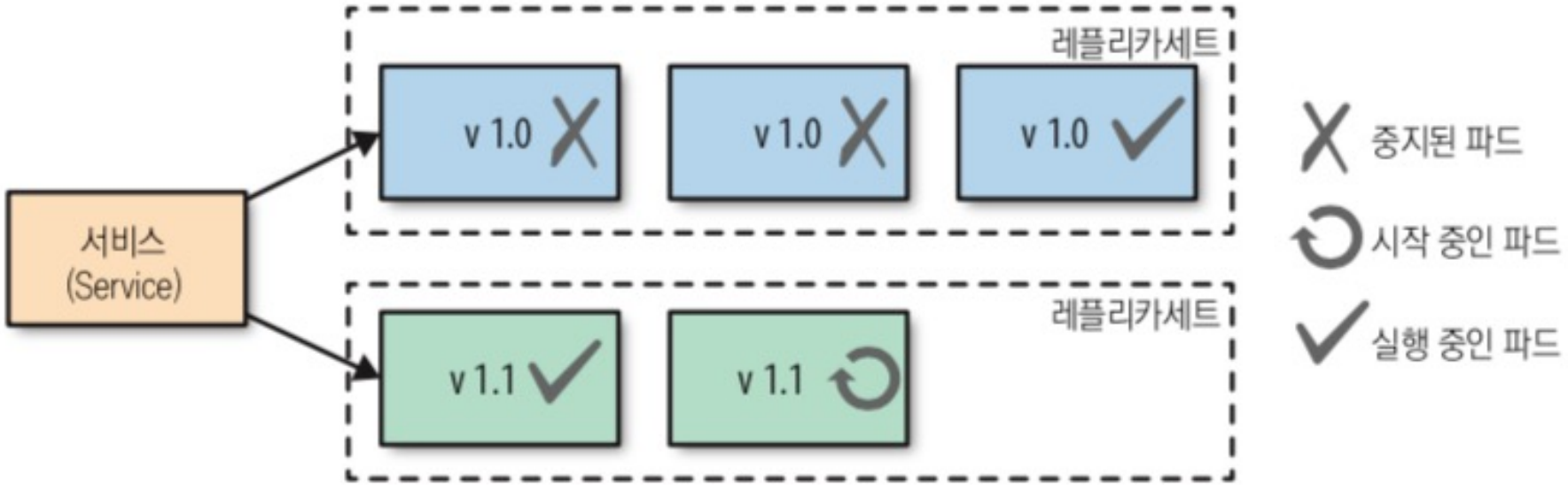
컨테이너 기반 운영 환경


이미지 서버
이미지 자동 다운로드
로드 발랜서
L4/L7 스위치
자동 연결
이미지 파일
(.tar.gz)
쿠버네티스 클러스터
컨테이너
이미지
Memory
CPU
Network
자동 실행
컨테이너
이미지
Memory
CPU
Network
자동 실행
아스키 파일
(.yaml)
API 서버
서버
서버

배포 전략 (Rolling update)

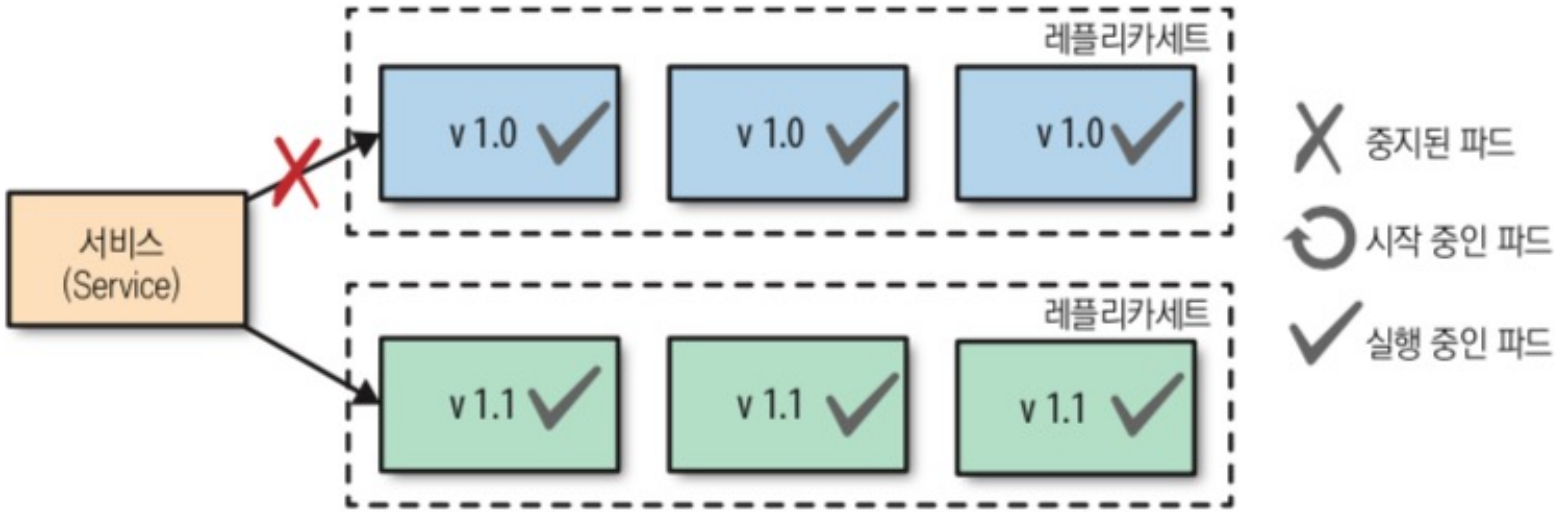
1. 롤링 배포 (Rolling update deployment)

- 두 버전이 동시에 존재
- 레이블 셀렉터를 지원하는 ReplicaSet 활용
- RollingUpdate : maxSurge, maxUnavailable



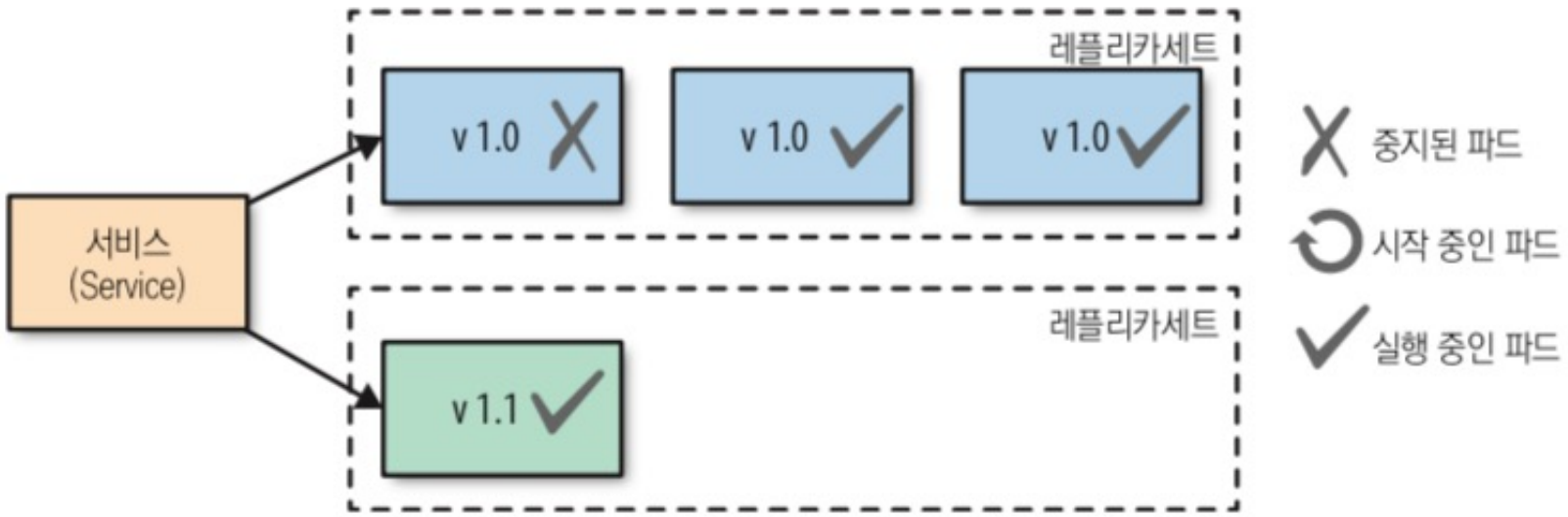
2. 블루-그린 배포 (Blue-Green deployment)

- Service 에서 Selector 를 활용
- 블루와 그린 컨테이너가 모두 실행되는 동안
  애플리케이션 용량은 2배로 필요

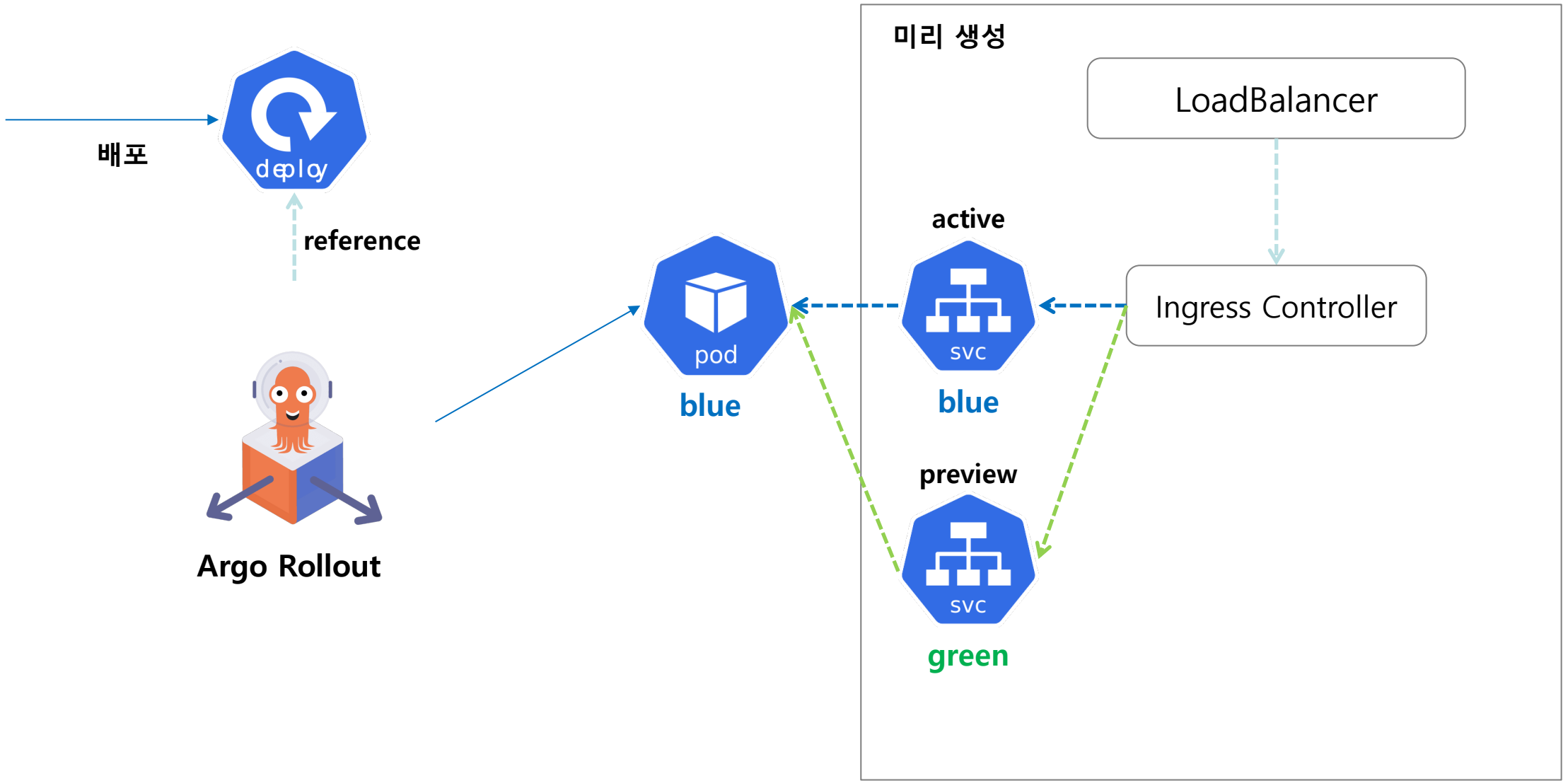


3. 카나리아 릴리스 (Canary release)

- 이전 인스턴스의 작은 하위집합만 새로운 인스턴스로 교체
- 운영에 새 버전을 도입할 때 위험을 줄여줌



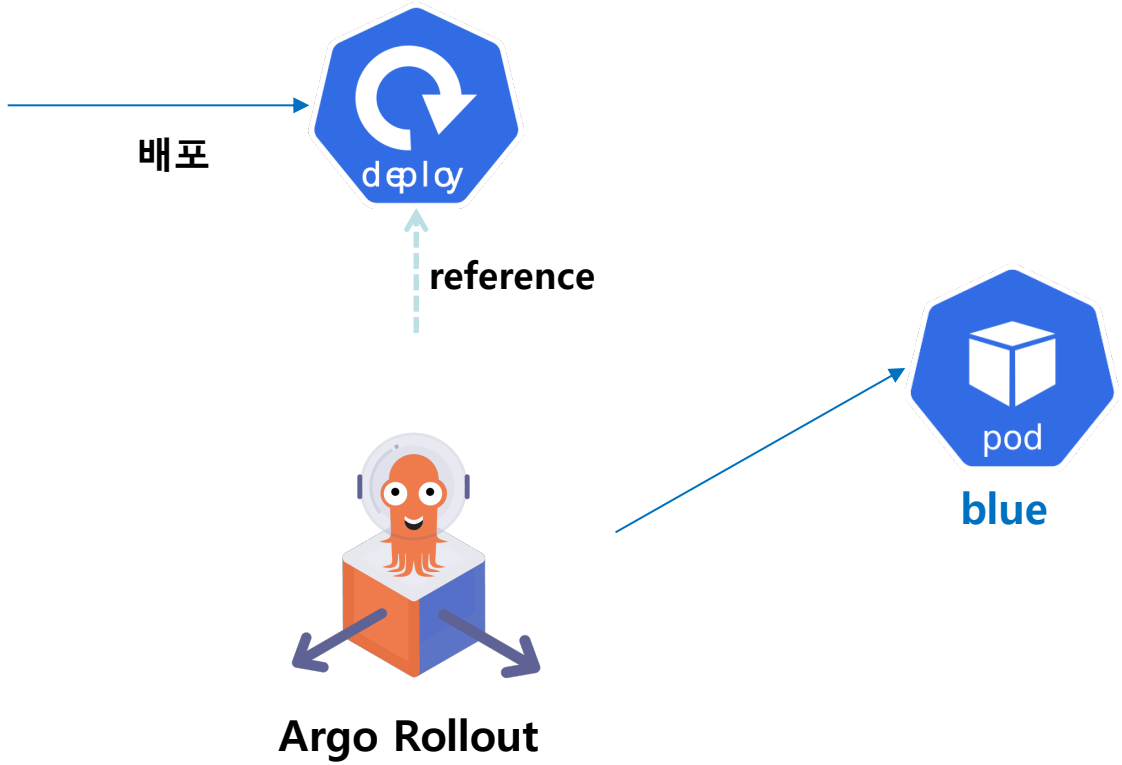
배포 전략 구현 (Argo Rollout – Blue 배포)



배포 전략 구현 (Argo Rollout) – Deployment

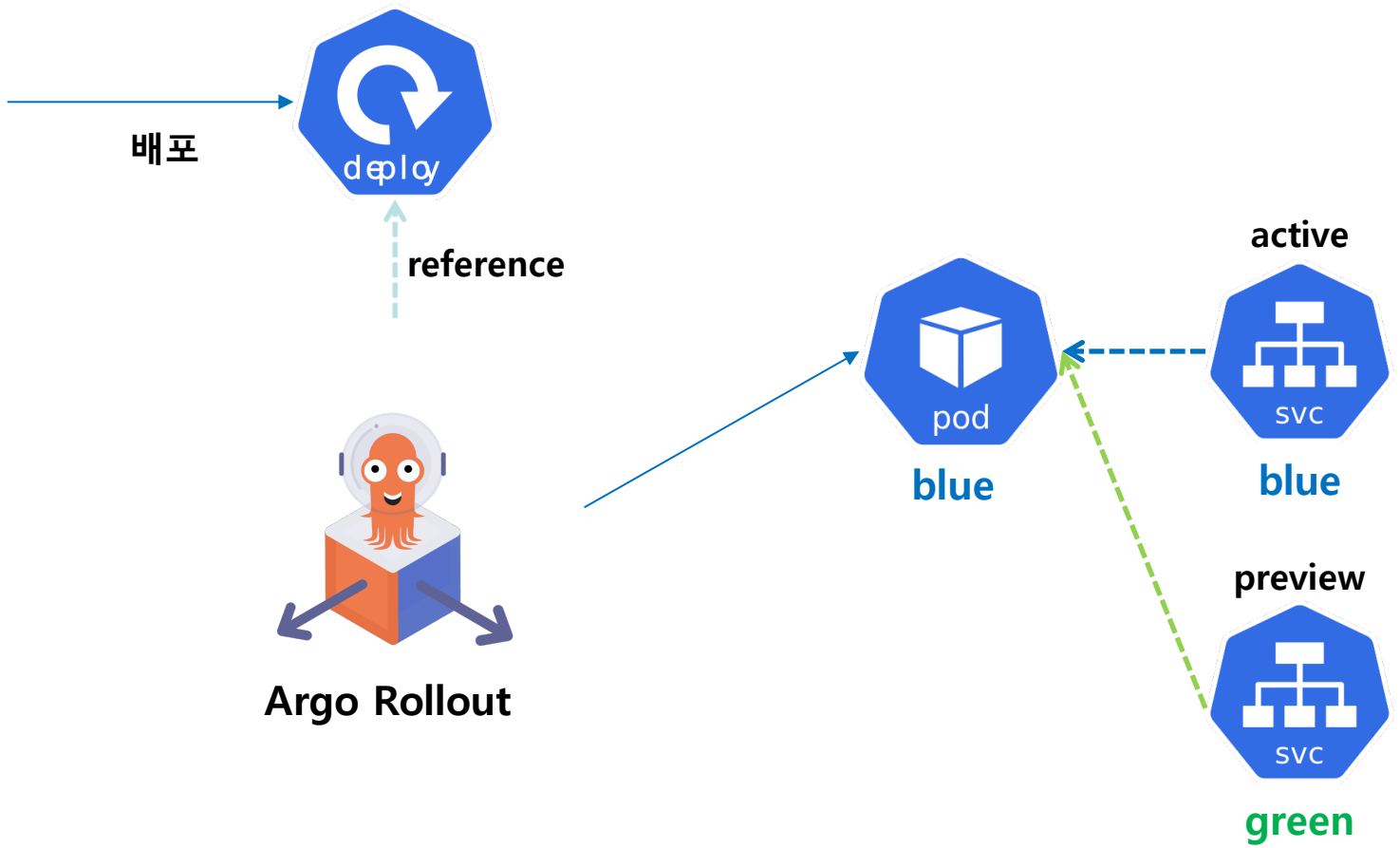

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 0
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: seungkyua/nginx:blue
        name: nginx
```

배포 전략 구현 (Argo Rollout) – Rollout




```yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: nginx
spec:
  replicas: 3
  revisionHistoryLimit: 5
  selector:
    matchLabels:
      app: nginx
  workloadRef:
    apiVersion: apps/v1
    kind: Deployment
    name: nginx
  strategy:
    blueGreen:
      activeService: nginx-svc
      previewService: nginx-preview-svc
      previewReplicaCount: 1
      autoPromotionEnabled: false
```

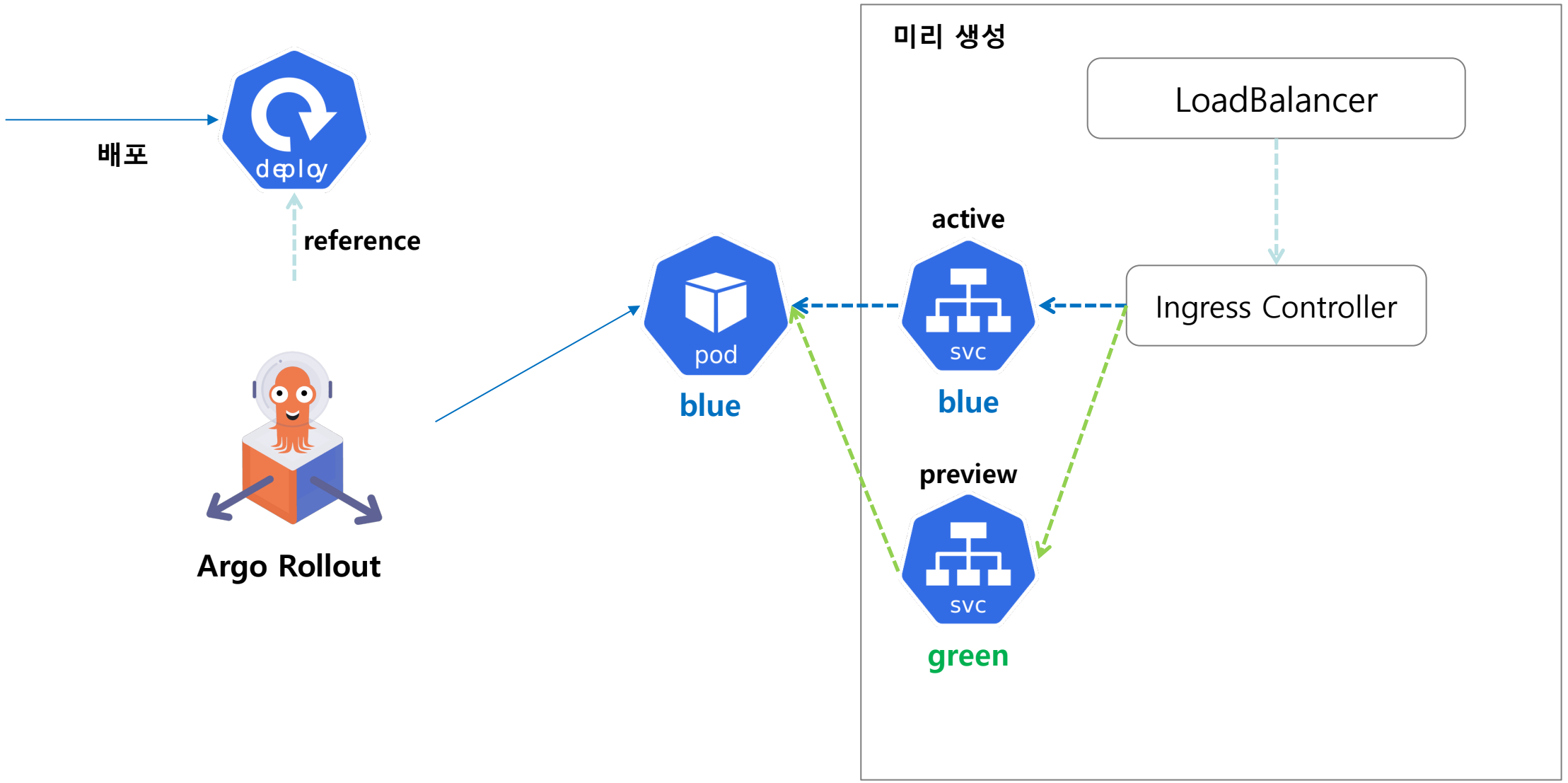
배포 전략 구현 (Argo Rollout) - Service



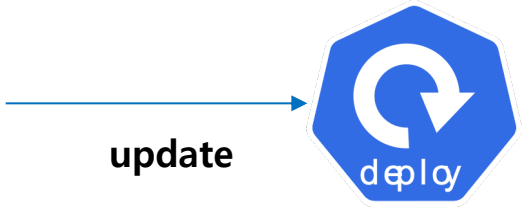

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-svc
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: ClusterIP
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-preview-svc
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: nginx
  type: ClusterIP
```

배포 전략 구현 (Argo Rollout – Blue 배포)

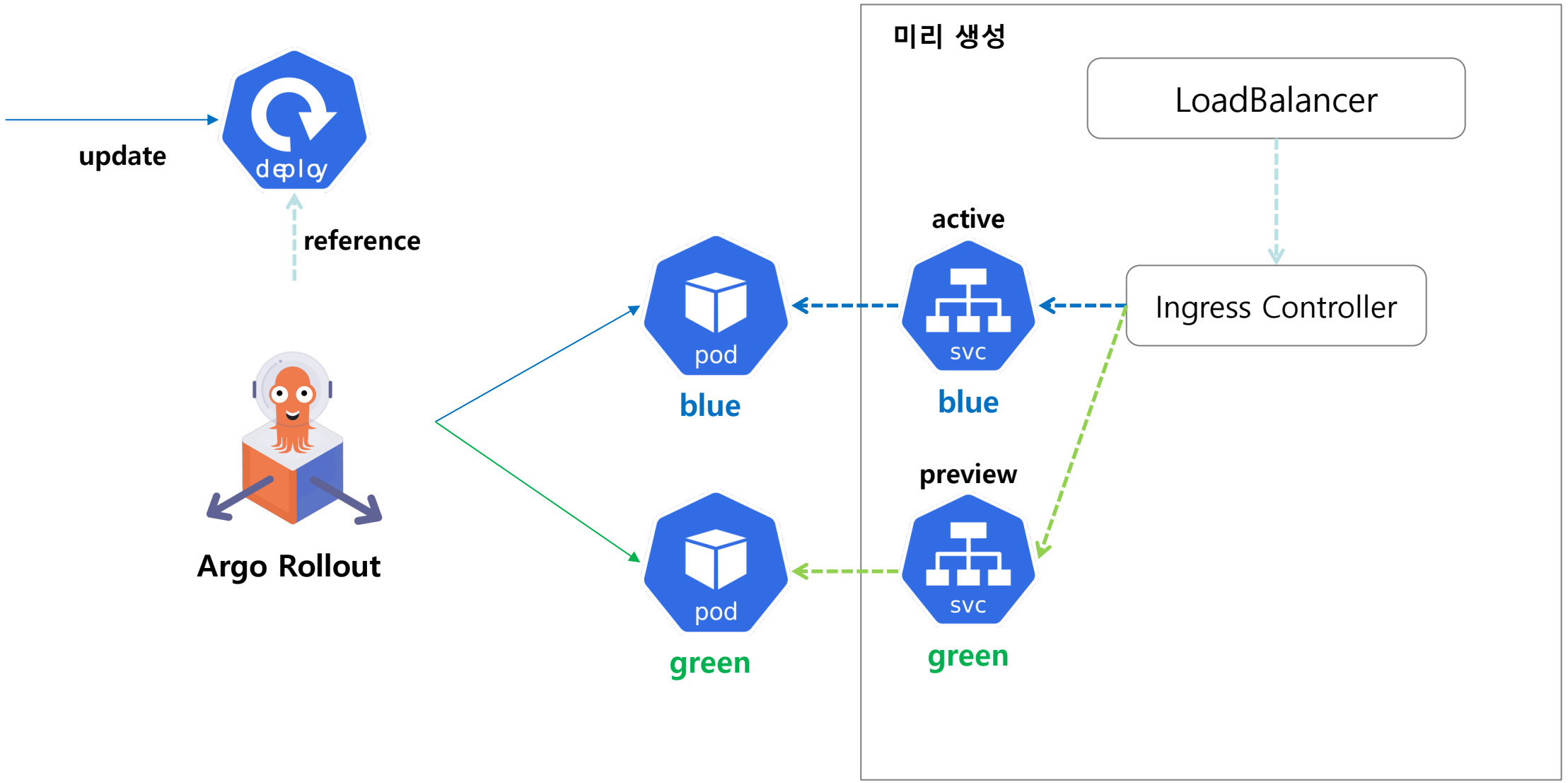


배포 전략 구현 (Argo Rollout – Green 배포)

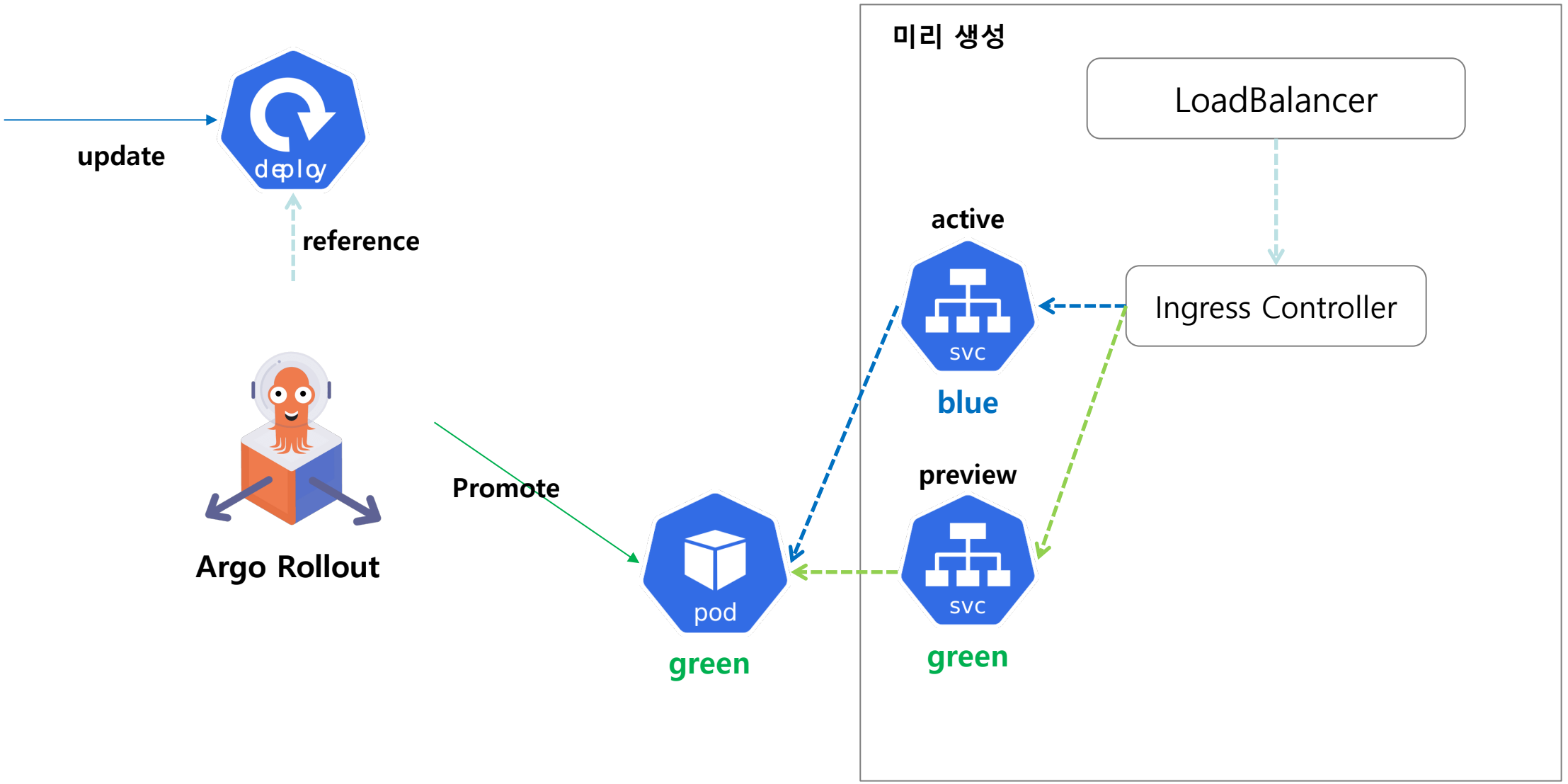



```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: nginx
  name: nginx
spec:
  replicas: 0
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - image: seungkyua/nginx:Green
        name: nginx
```

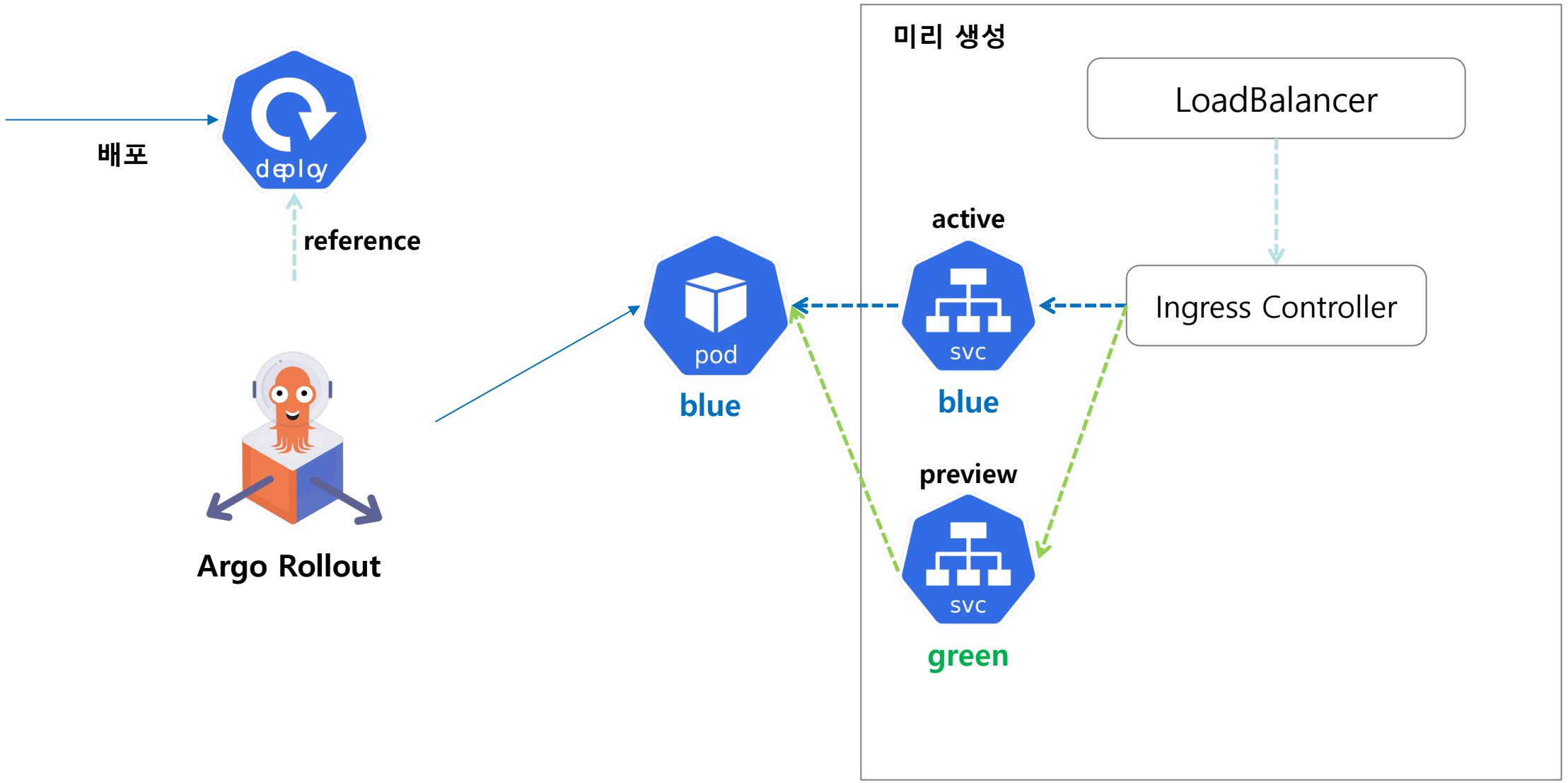
배포 전략 구현 (Argo Rollout – Green 배포)



배포 전략 구현 (Argo Rollout) – Promote

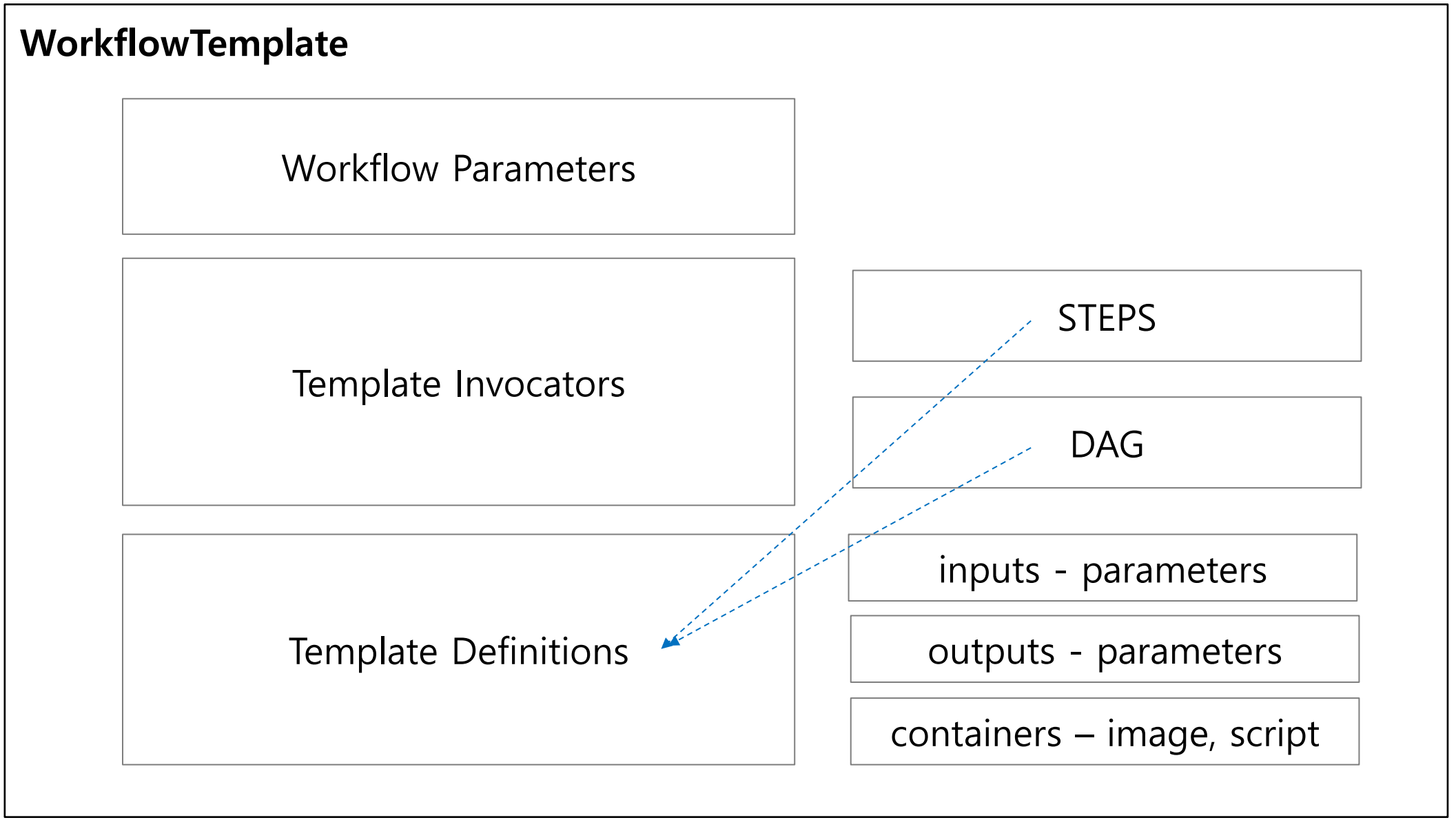


배포 전략 구현 (Argo Rollout) – Blue 배포 상태



1. WorkflowTemplate 작성 (Custom Resource)
2. WorkflowTemplate 저장 (kubectl apply –f xxx)
3. WorkflowTemplate 실행 (argo summit --from xx/xx)

배포 프로세스 구현 (Argo Workflow) – WorkflowTemplate 구조



배포 프로세스 구현 (Argo Workflow) – WorkflowTemplate 구조

```yaml
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: serve-java-app
  namespace: argo
spec:
  entrypoint: main
  arguments:
    parameters:
    - name: app_type
      value: "springboot"

  templates:
  - name: main
    steps:
    - - name: build-image
        template: build-image
    - - name: deploy-app
        templateRef:
          name: deploy-app
          template: deploy-java-app
        arguments:
          parameters:
          - name: app_type
            value: "{{workflow.parameters.app_type}}"
```

Template Parameters

Template Invocators

배포 프로세스 구현 (Argo Workflow) – WorkflowTemplate 구조

```yaml
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: serve-java-app
  namespace: argo
spec:
...
  templates:
  - name: main
    steps:
    - - name: build-image
        template: build-image
...
...
  - name: build-image
    container:
      image: harbor-cicd.taco-cat.xyz/tks/appserving-worker:latest
      command:
      - /bin/sh
      - '-exc'
      - |
        BUILD_LOG='/mnt/out/build_output.log'
        mkdir -p /apps && cd /apps/
        ...
```

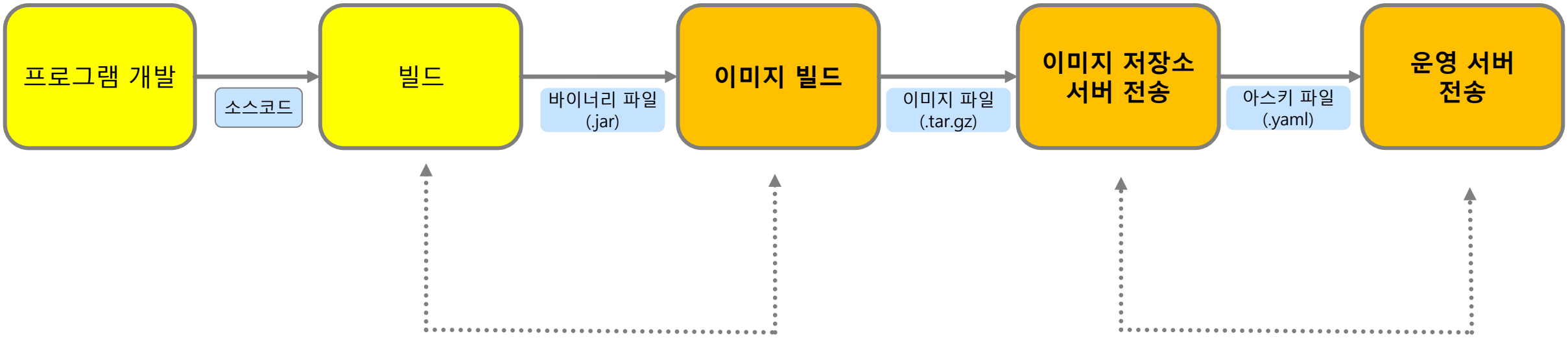
Template Invocators

Template Definitions

컨테이너 기반 개발/배포 프로세스 (CI/CD)

단계별로 WorkflowTemplate 을 만들어서 통합하여 한 번에 실행

서버 대수와 상관 없이 한 번만 작업



CI
(Continuous Integration)

CD
(Continuous Deployment/Delivery)

7/4(화) 1:30 - 3:30 pm, 209A

쿠버네티스 환경에 어플리케이션 배포하기

- TKS 클라우드 서비스를 사용하여 AWS 의 EKS 를 손쉽게 만들수 있습니다.

- 사용자가 만든 자바 샘플 앱을 쿠버네티스 위에 배포하는 방법을 배웁니다.

- Argo Rollout 을 활용하여 자바 샘플 앱을 다양한 배포 전략을 활용하여 배포합니다. (Rolling Update, Blue-Green)

- TKS 의 앱서빙을 활용하여 자바 앱을 더욱 손쉽게 배포하는 방법을 배웁니다.

Q & A