NHN FORWARD

Spring Cloud 기반 MSA 환경을 쿠버네티스로 전환하기

NHN 커머스DevOps팀
윤서원

다룰 내용

1. 샵바이 소개
2. 쿠버네티스로 전환하는 이유
3. 쿠버네티스 전환 준비하기
4. 쿠버네티스로 배포하기
5. 쿠버네티스로 전환하기

샵바이 소개

샵바이 소개

NHN FORWARD

클라우드 이커머스 플랫폼

shop by

1인 쇼핑몰 창업

shop by pro

중소형 쇼핑몰

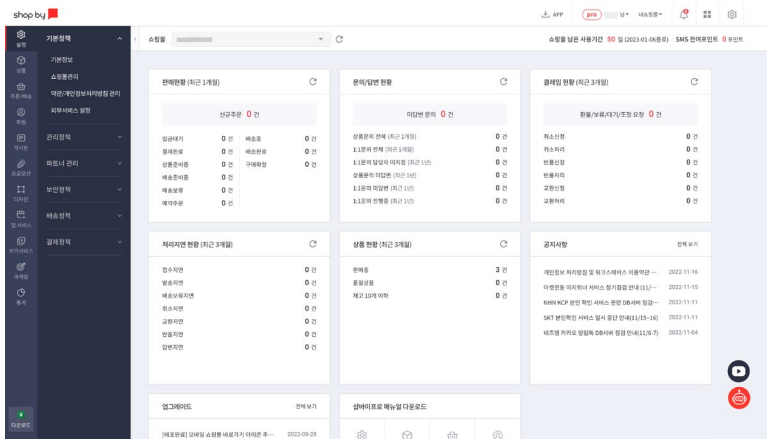
shop by premium

대형 쇼핑몰

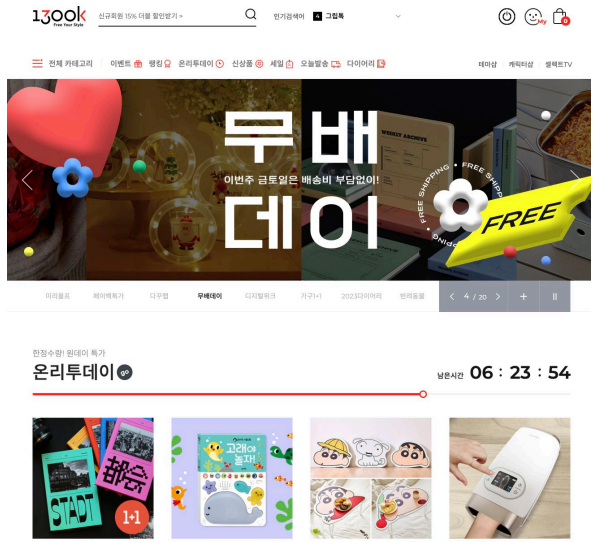
샵바이 소개

NHN FORWARD

클라우드 이커머스 플랫폼



운영자 페이지

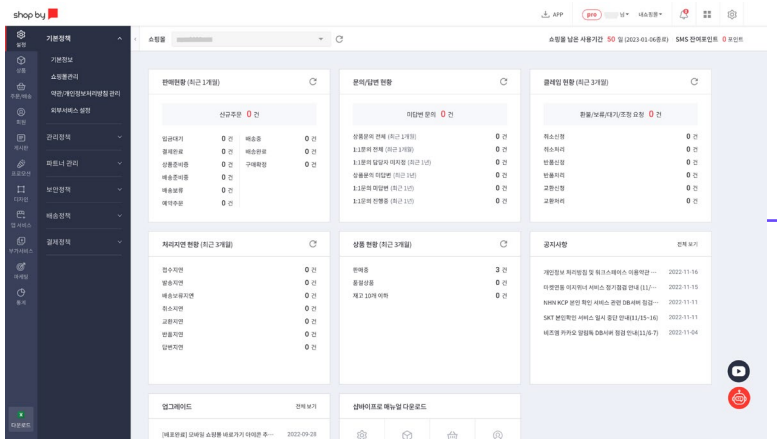


쇼핑몰 페이지

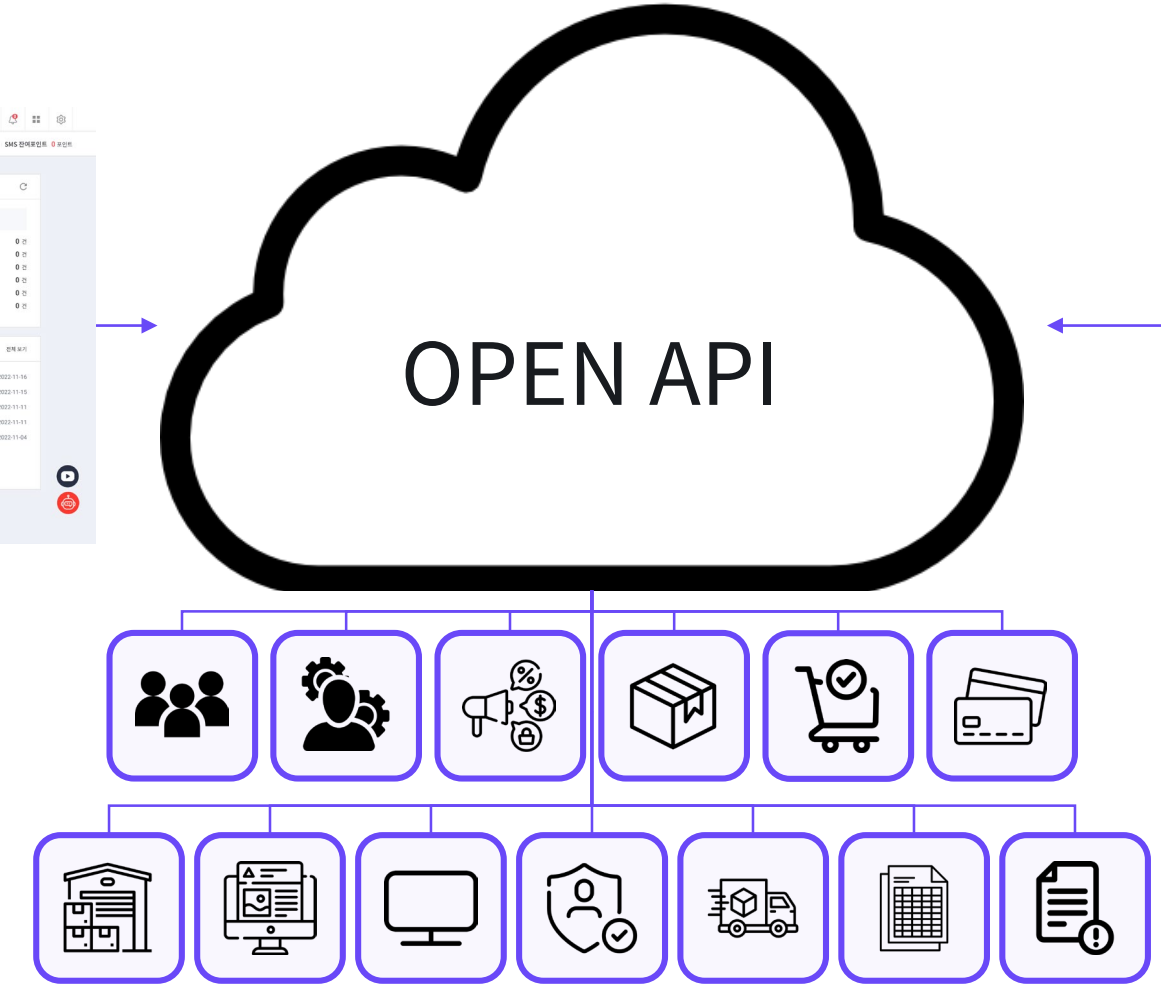
샵바이 소개

NHN FORWARD

클라우드 이커머스 플랫폼



운영자 페이지



쇼핑몰 페이지

쿠버네티스로 전환하는 이유

쿠버네티스로 전환하는 이유

NHN FORWARD

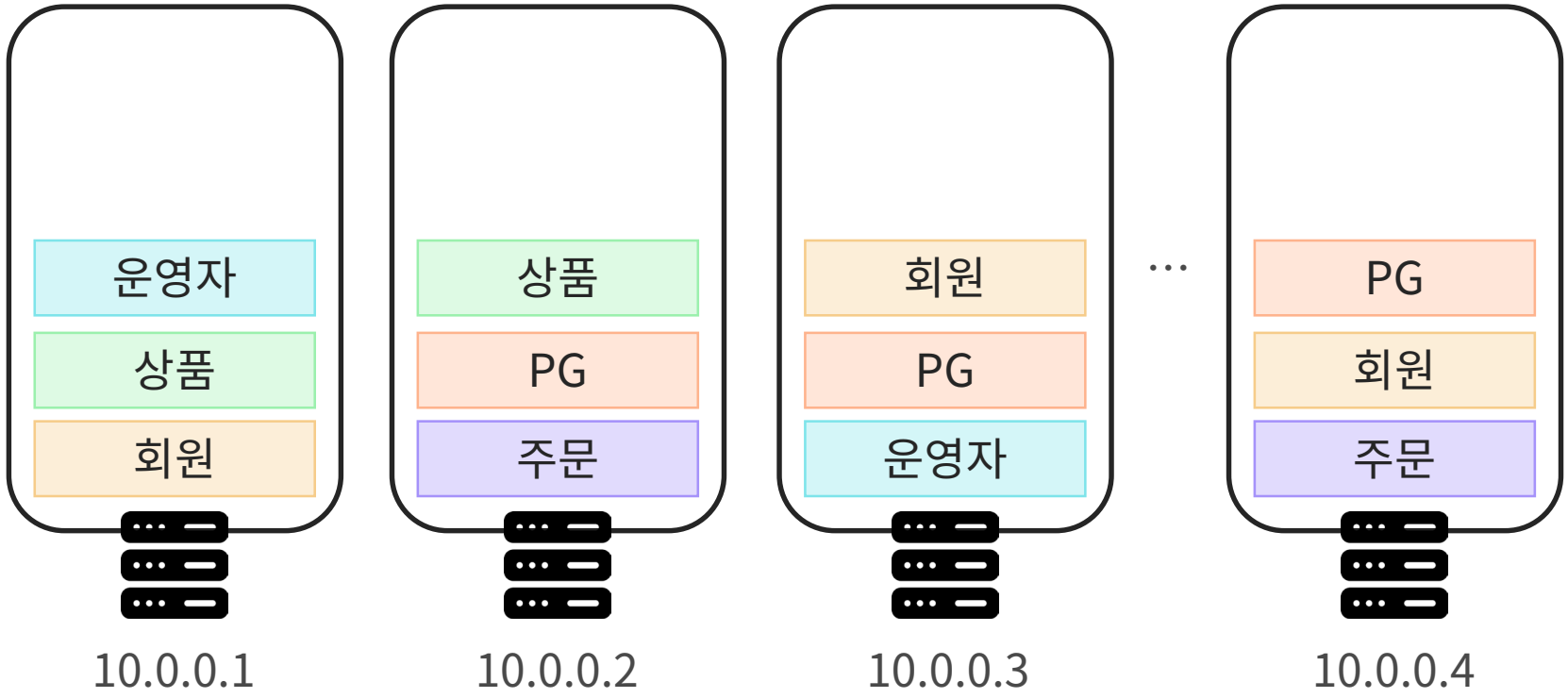
마이크로서비스가 너무 많다!

- 고정적인 스케줄링
- 잦은 스케일링
- 관리의 어려움 등…

쿠버네티스로 전환하는 이유

NHN FORWARD

기존 스케줄링

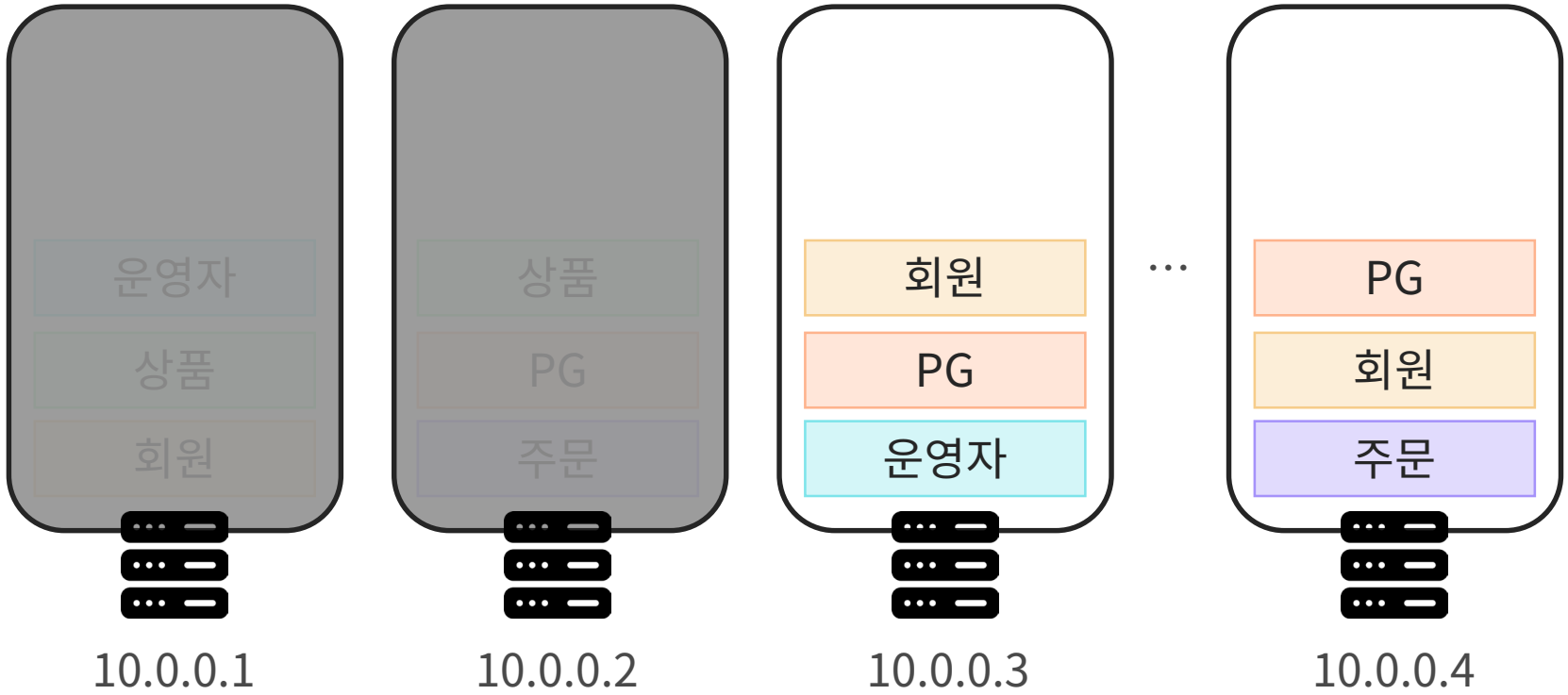


ANSIBLE

Microservice / Host
[admin] 10.0.0.1 10.0.0.3
[product] 10.0.0.1 10.0.0.2
[member] 10.0.0.1 10.0.0.3 10.0.0.4
[order] 10.0.0.2 10.0.0.4
[pg] 10.0.0.2 10.0.0.3 10.0.0.4

쿠버네티스로 전환하는 이유

기존 스케줄링



ANSIBLE

Microservice / Host

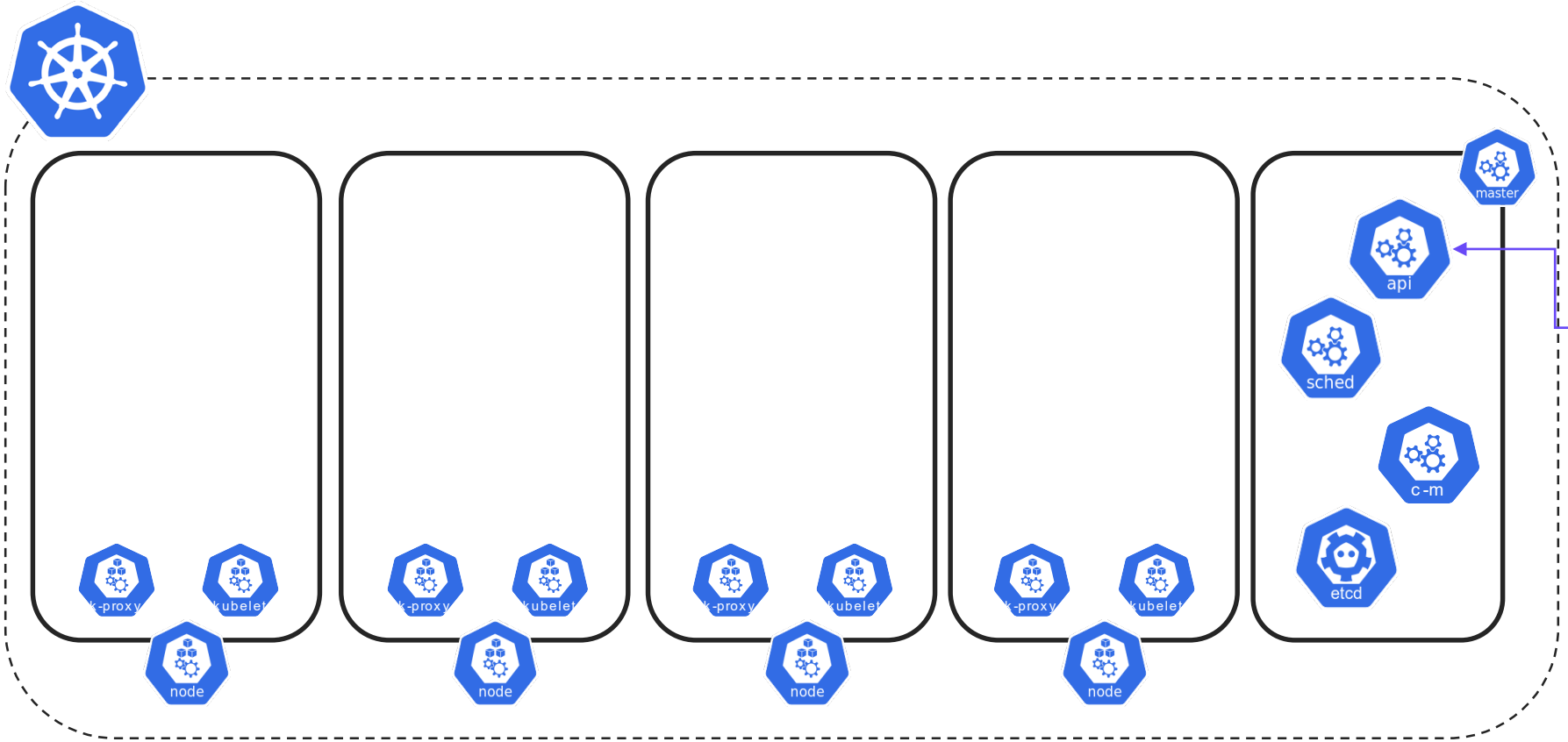
```
[admin]
10.0.0.1
10.0.0.3

[product]
10.0.0.1
10.0.0.2

[member]
10.0.0.1
10.0.0.3
10.0.0.4

[order]
10.0.0.2
10.0.0.4

[pg]
10.0.0.2
10.0.0.3
10.0.0.4
```

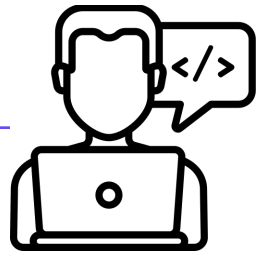
쿠버네티스로 전환하는 이유

NHN FORWARD

유연한 스케줄링

상품 서비스 2개
운영자 서비스 2개
주문 서버 3개
회원 서버 3개
PG 서버 2개
배포해 줘~

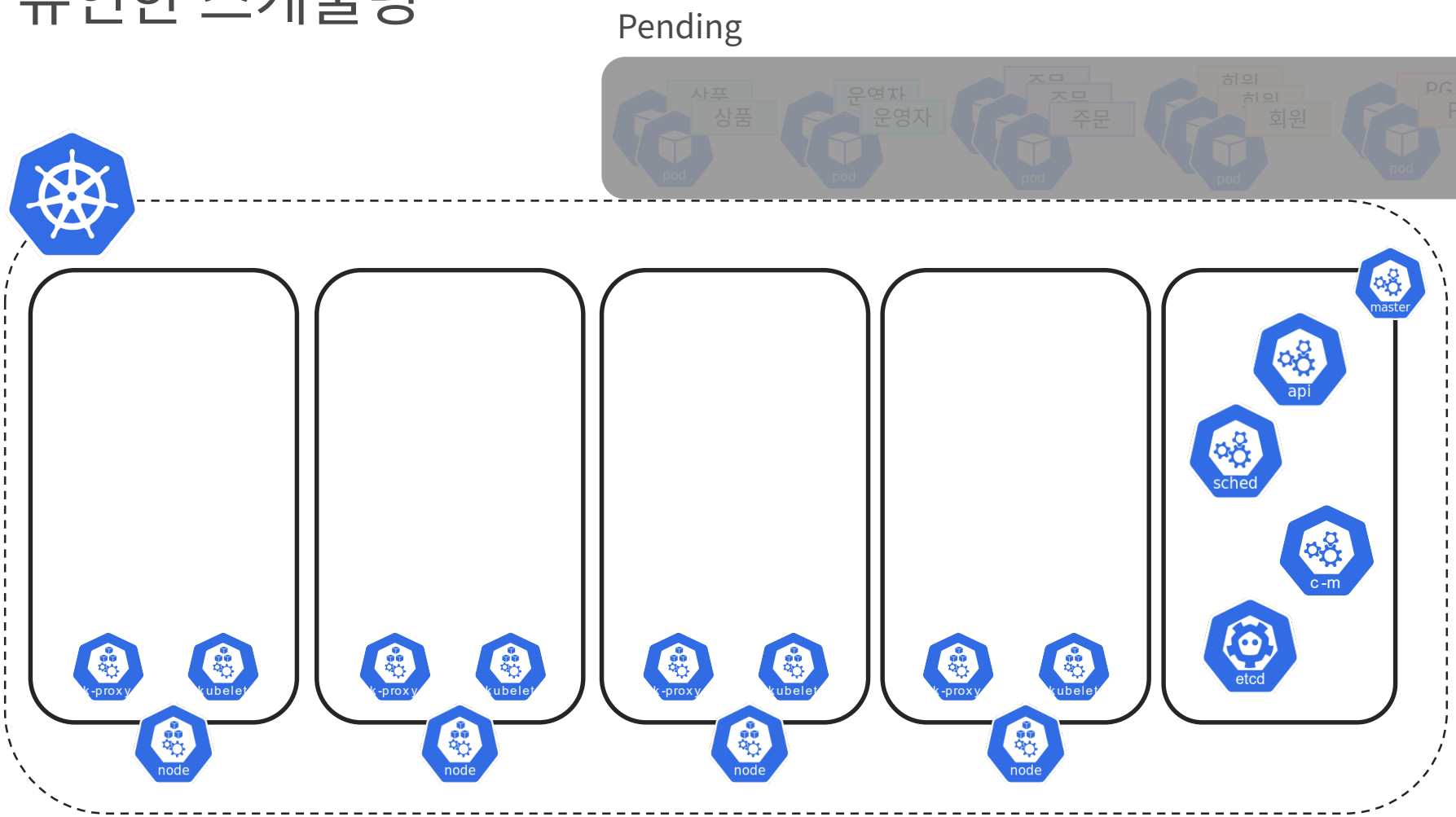




쿠버네티스로 전환하는 이유

NHN FORWARD

유연한 스케줄링

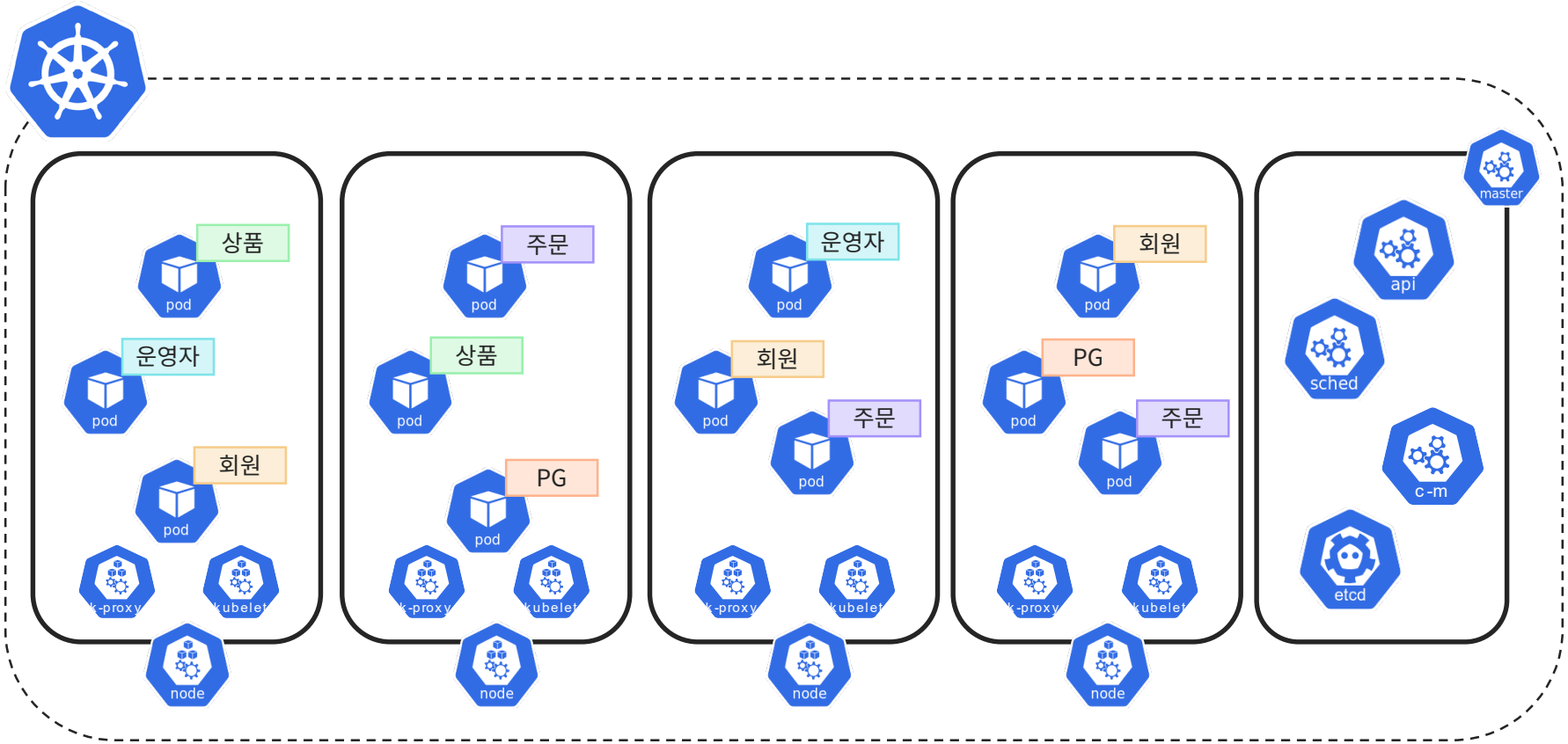


Desired State	Current State
name: product image: product replica: 2	name: product image: product replica: 0/2
name: admin image: admin replica: 2	name: admin image: admin replica: 0/2
name: order image: order replica: 3	name: order image: order replica: 0/3
name: member image: member replica: 3	name: member image: member replica: 0/3
name: pg image: pg replica: 2	name: pg image: pg replica: 0/2

쿠버네티스로 전환하는 이유

NHN FORWARD

유연한 스케줄링

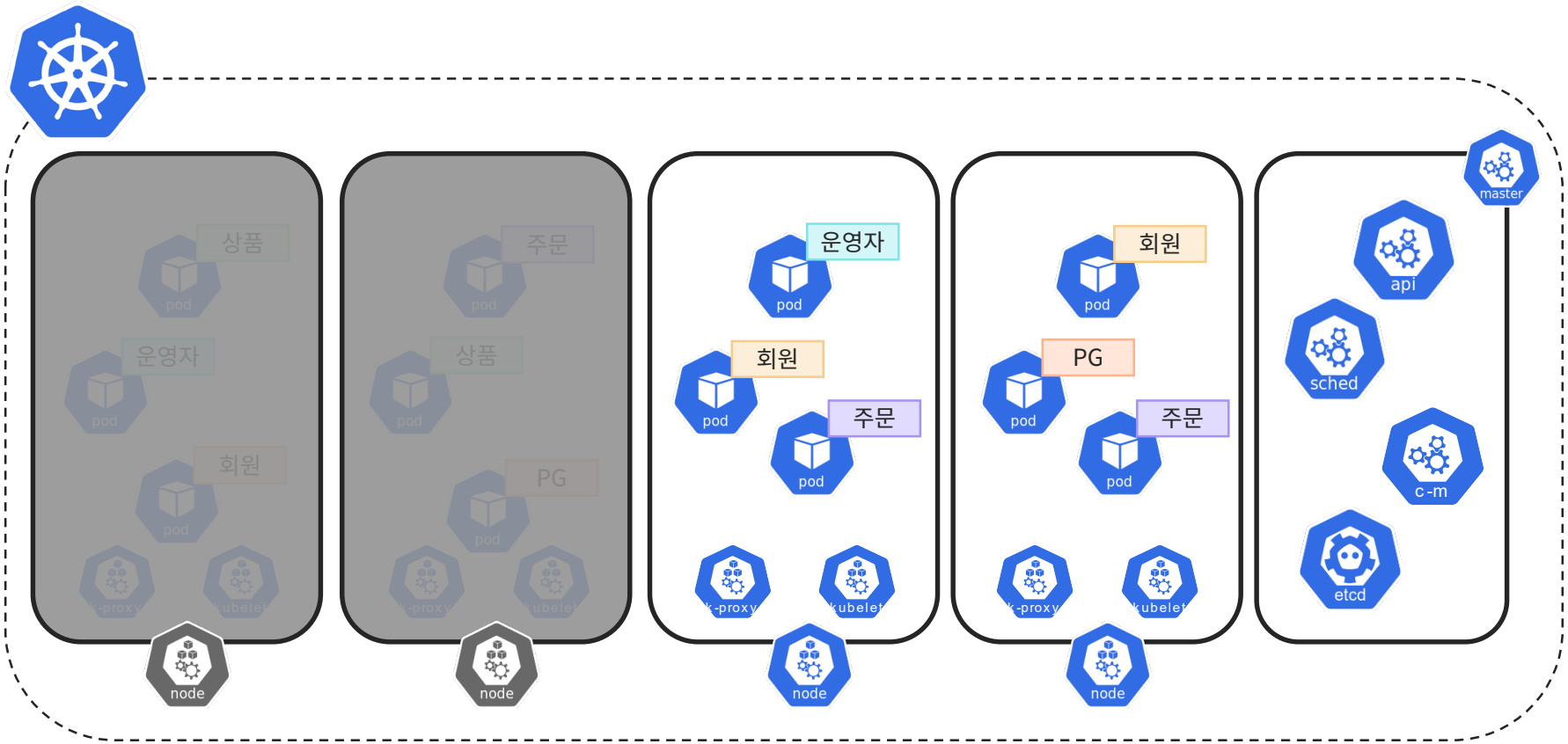


Desired State	Current State
name: product image: product replica: 2	name: product image: product replica: 2
name: admin image: admin replica: 2	name: admin image: admin replica: 2
name: order image: order replica: 3	name: order image: order replica: 3
name: member image: member replica: 3	name: member image: member replica: 3
name: pg image: pg replica: 2	name: pg image: pg replica: 2

쿠버네티스로 전환하는 이유

NHN FORWARD

유연한 스케줄링

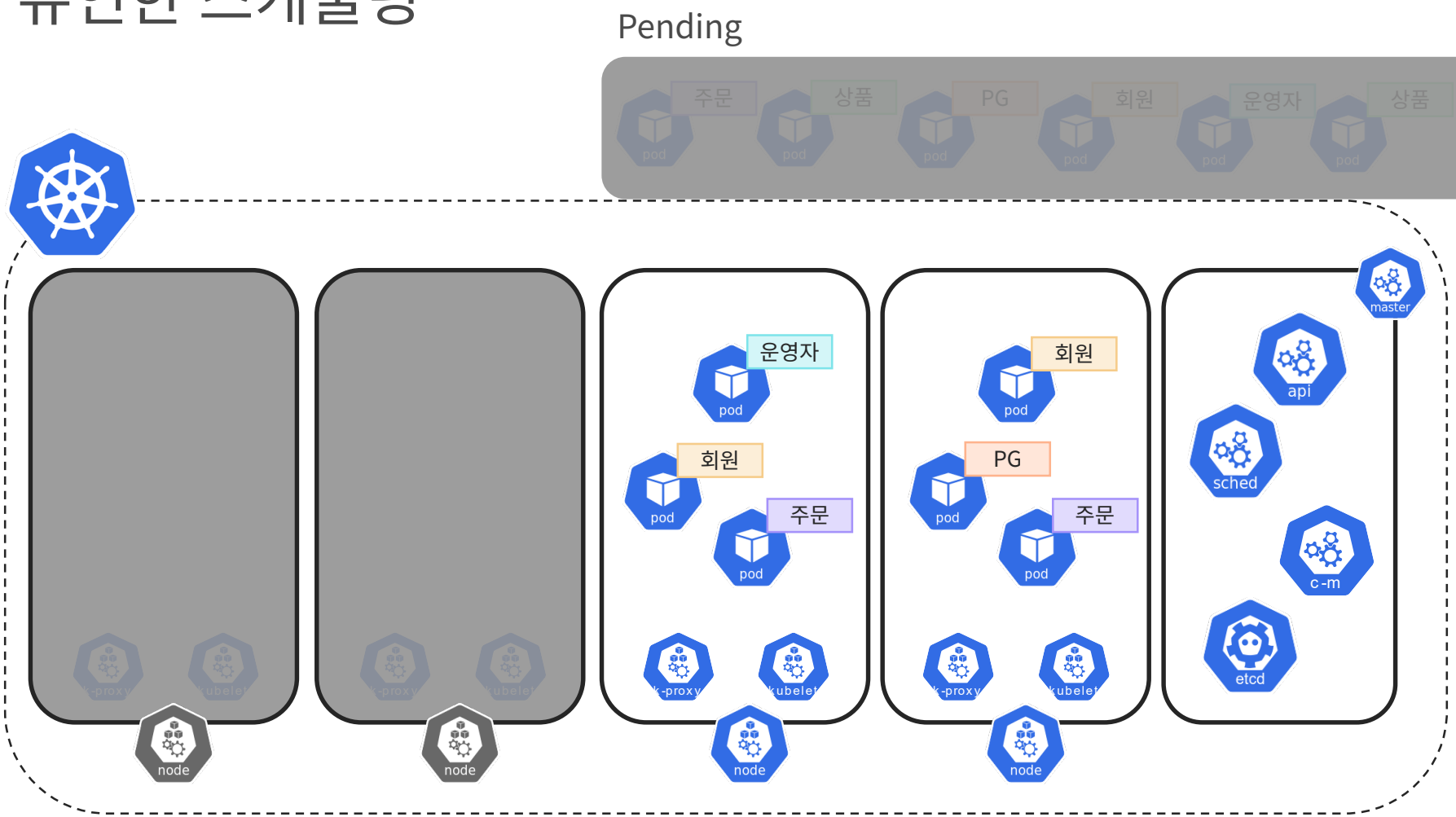


Desired State	Current State
name: product image: product replica: 2	name: product image: product replica: 0
name: admin image: admin replica: 2	name: admin image: admin replica: 1
name: order image: order replica: 3	name: order image: order replica: 2
name: member image: member replica: 3	name: member image: member replica: 2
name: pg image: pg replica: 2	name: pg image: pg replica: 1

쿠버네티스로 전환하는 이유

NHN FORWARD

유연한 스케줄링

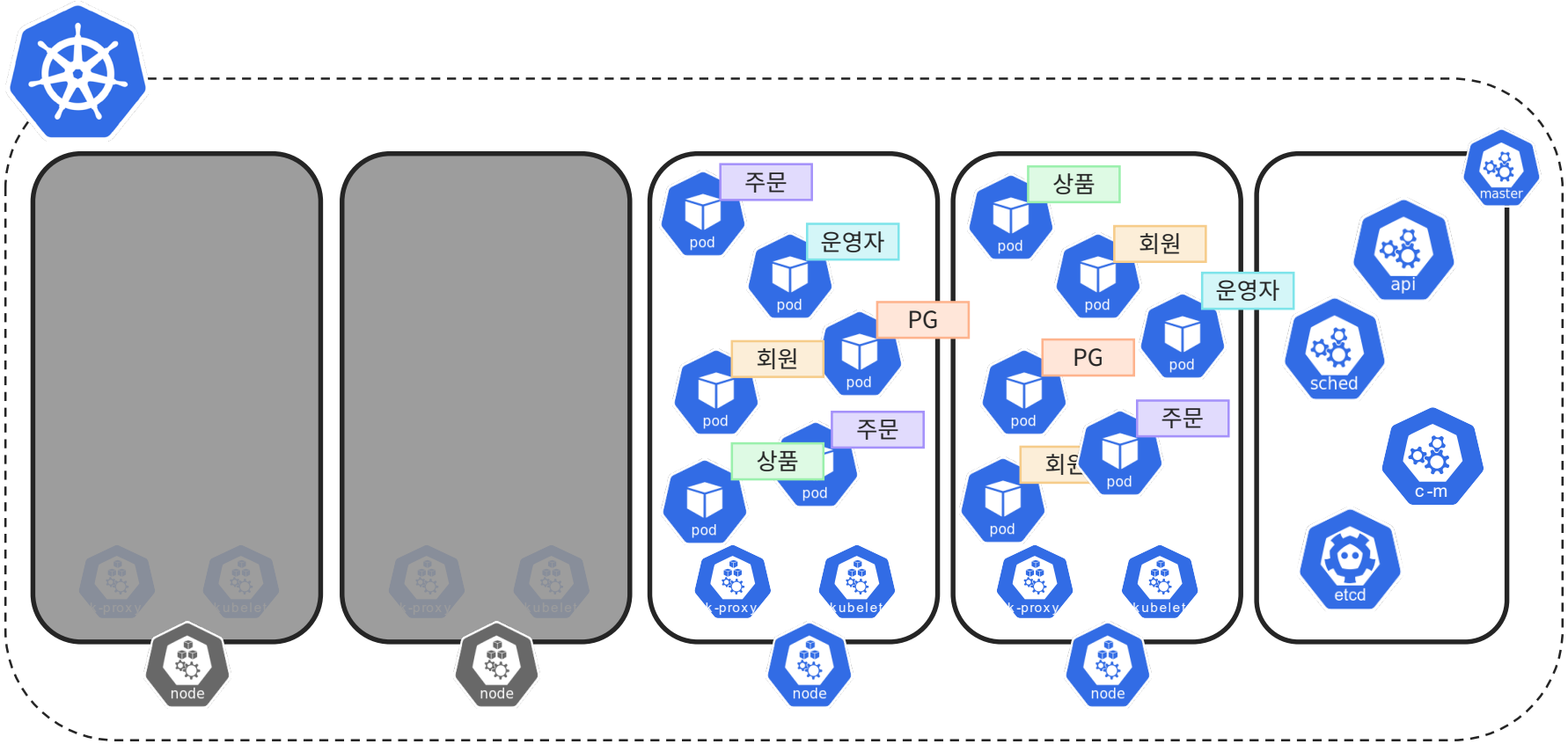


Desired State	Current State
name: product image: product replica: 2	name: product image: product replica: 0
name: admin image: admin replica: 2	name: admin image: admin replica: 1
name: order image: order replica: 3	name: order image: order replica: 2
name: member image: member replica: 3	name: member image: member replica: 2
name: pg image: pg replica: 2	name: pg image: pg replica: 1

쿠버네티스로 전환하는 이유

NHN FORWARD

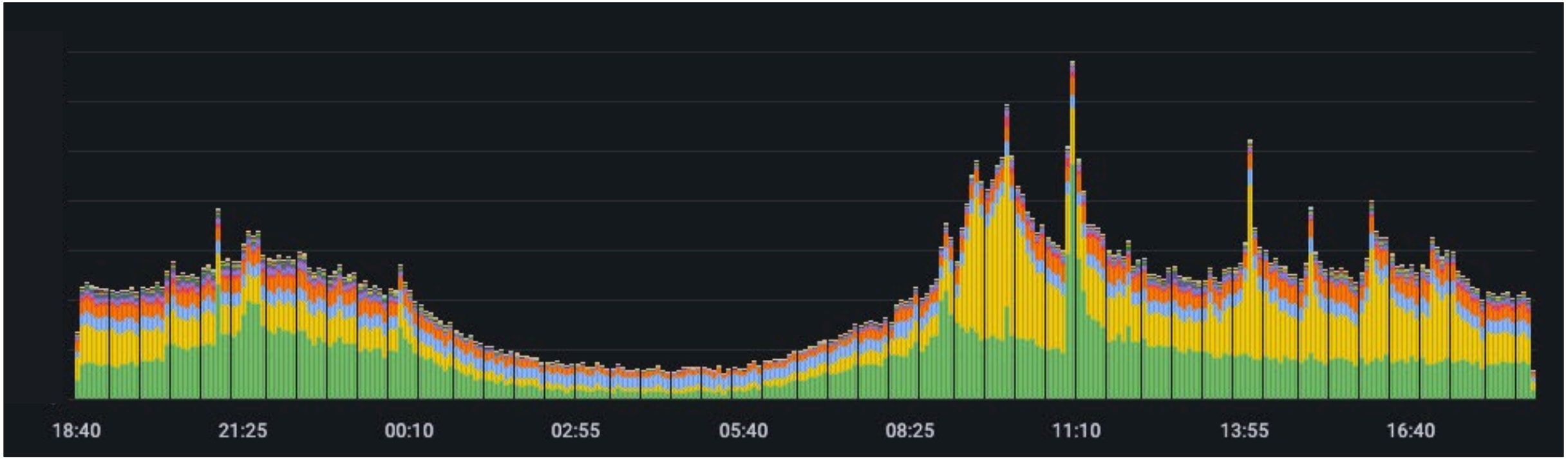
유연한 스케줄링



Desired State	Current State
name: product image: product replica: 2	name: product image: product replica: 2
name: admin image: admin replica: 2	name: admin image: admin replica: 2
name: order image: order replica: 3	name: order image: order replica: 3
name: member image: member replica: 3	name: member image: member replica: 3
name: pg image: pg replica: 2	name: pg image: pg replica: 2

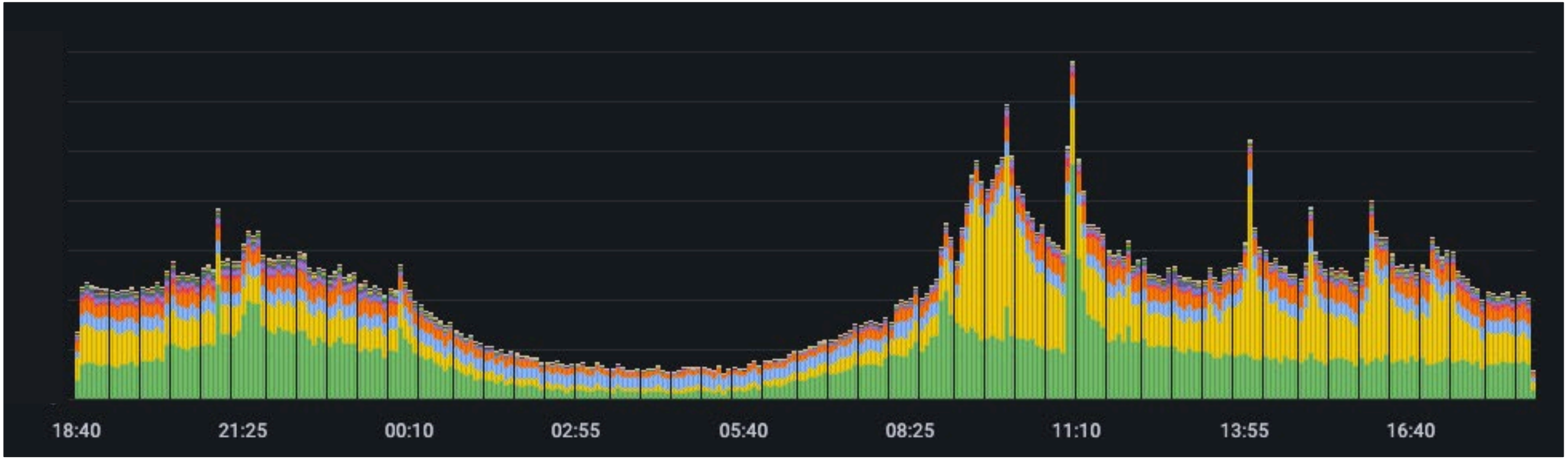
쿠버네티스로 전환하는 이유

오토 스케일링



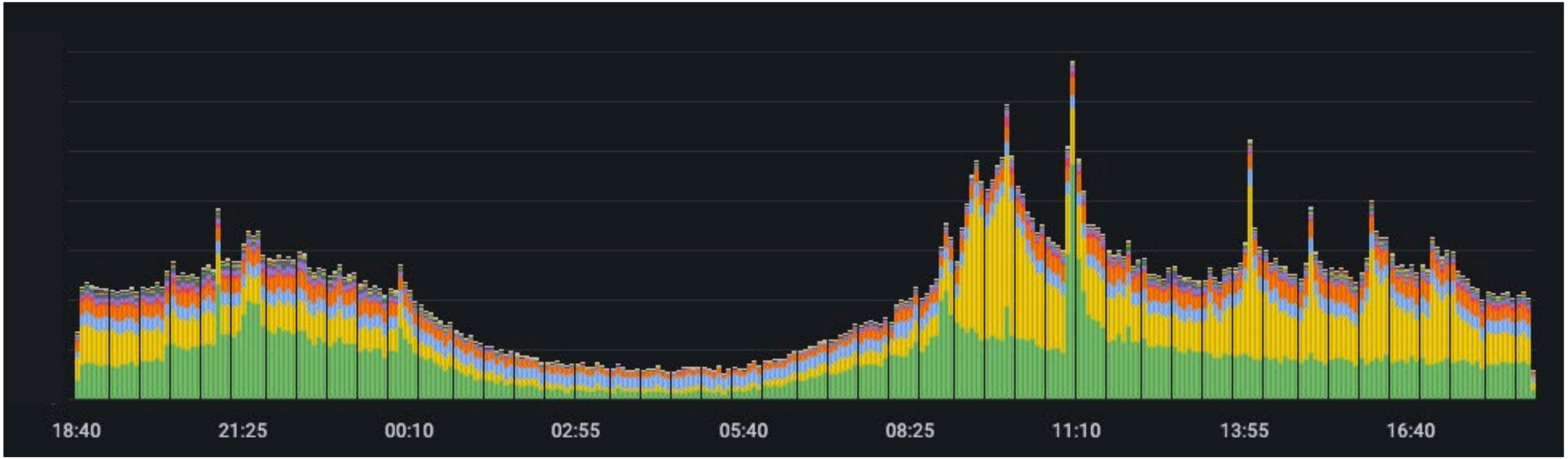
쿠버네티스로 전환하는 이유

오토 스케일링



쿠버네티스로 전환하는 이유

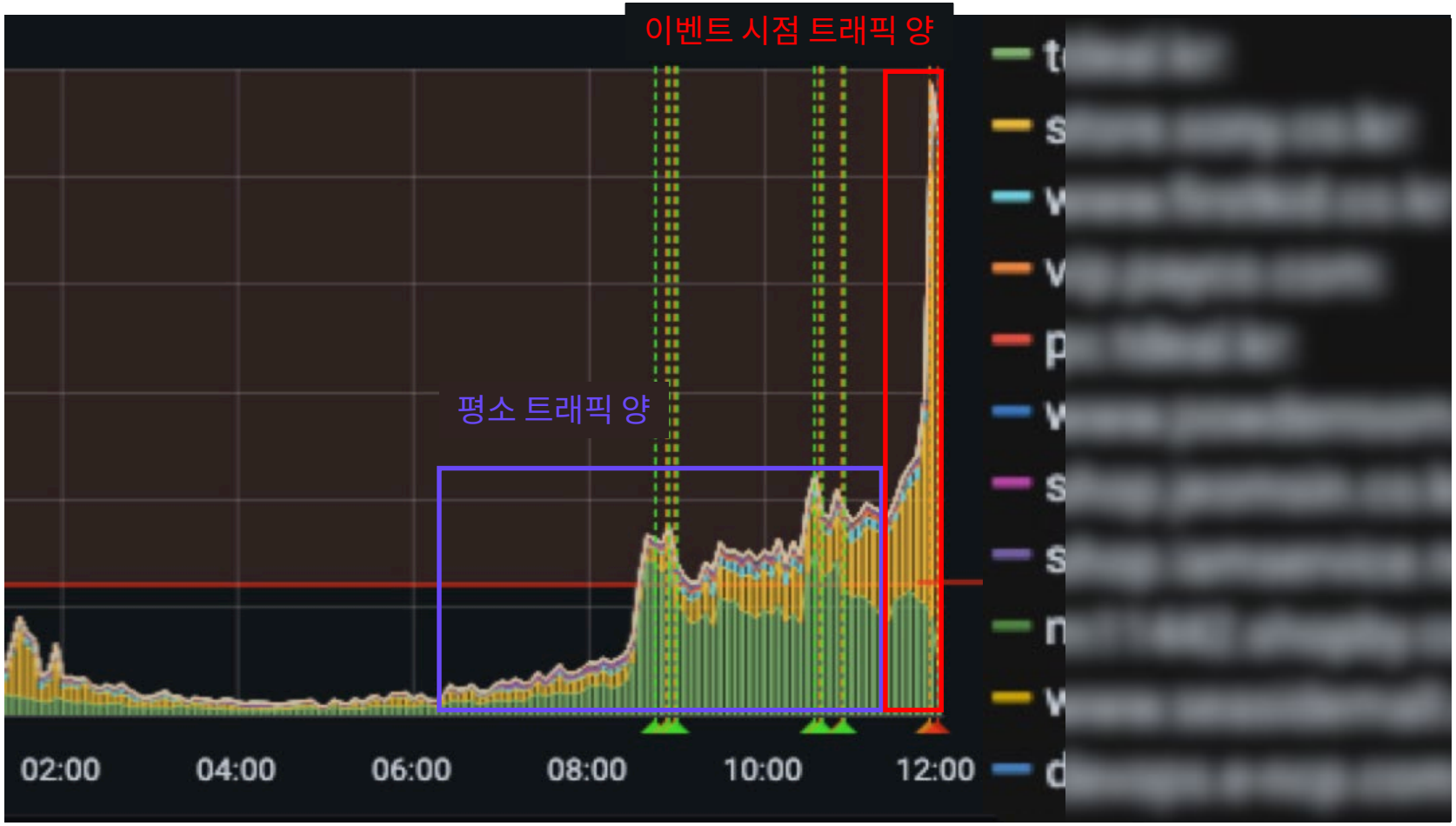
오토 스케일링



쿠버네티스로 전환하는 이유

NHN FORWARD

쇼핑몰 이벤트



상위 10개 쇼핑몰 트래픽 양

쿠버네티스로 전환하는 이유

NHN FORWARD

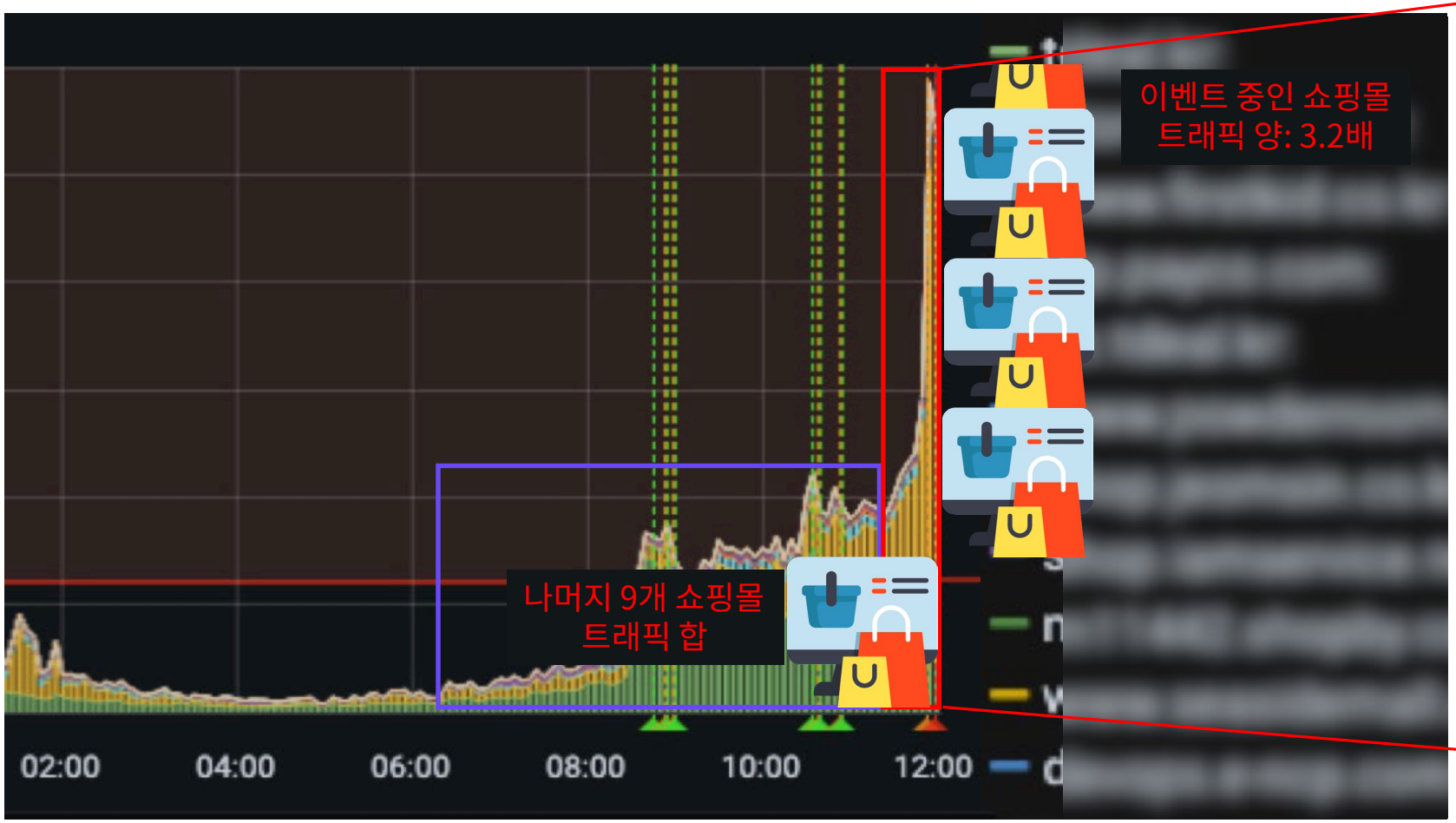
쇼핑몰 이벤트

상위 10개 쇼핑몰 트래픽 양

쿠버네티스로 전환하는 이유

NHN FORWARD

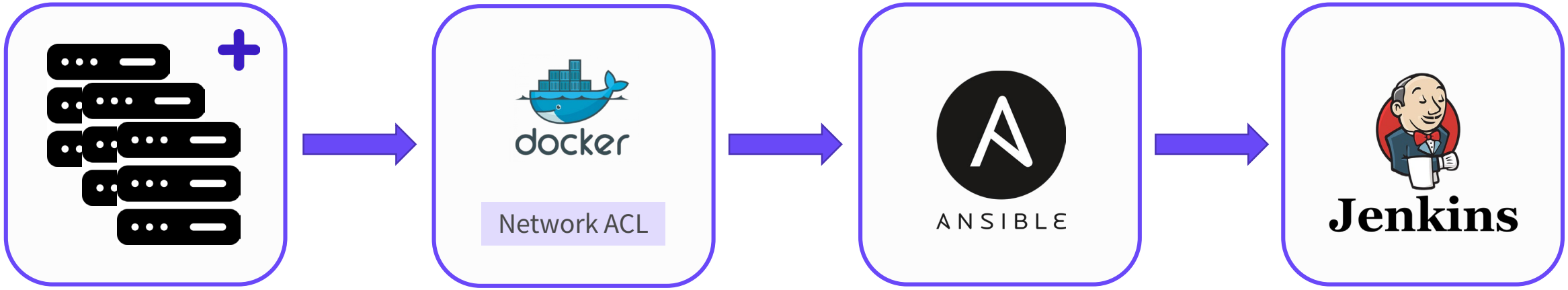
쇼핑몰 이벤트



상위 10개 쇼핑몰 트래픽 양

쿠버네티스로 전환하는 이유

쉽지 않은 서버 증설 + 스케일아웃 [Private Cloud]



서버 증설

서버 세팅
(배포 환경 세팅)

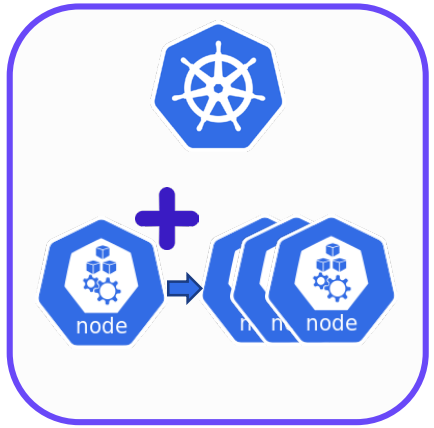
배포 설정 변경

배포

쿠버네티스로 전환하는 이유

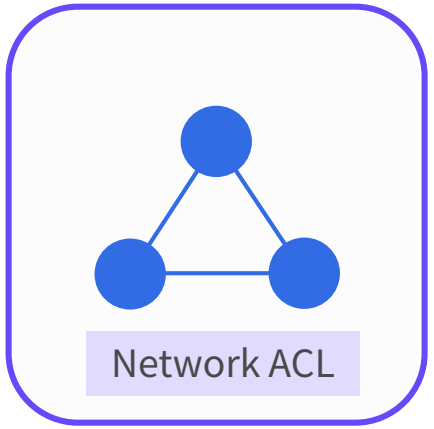
NHN FORWARD

유연한 서버 증설 + 스케일아웃 [K8s Managed Service]



노드 증설

스케일아웃



보안 그룹 설정

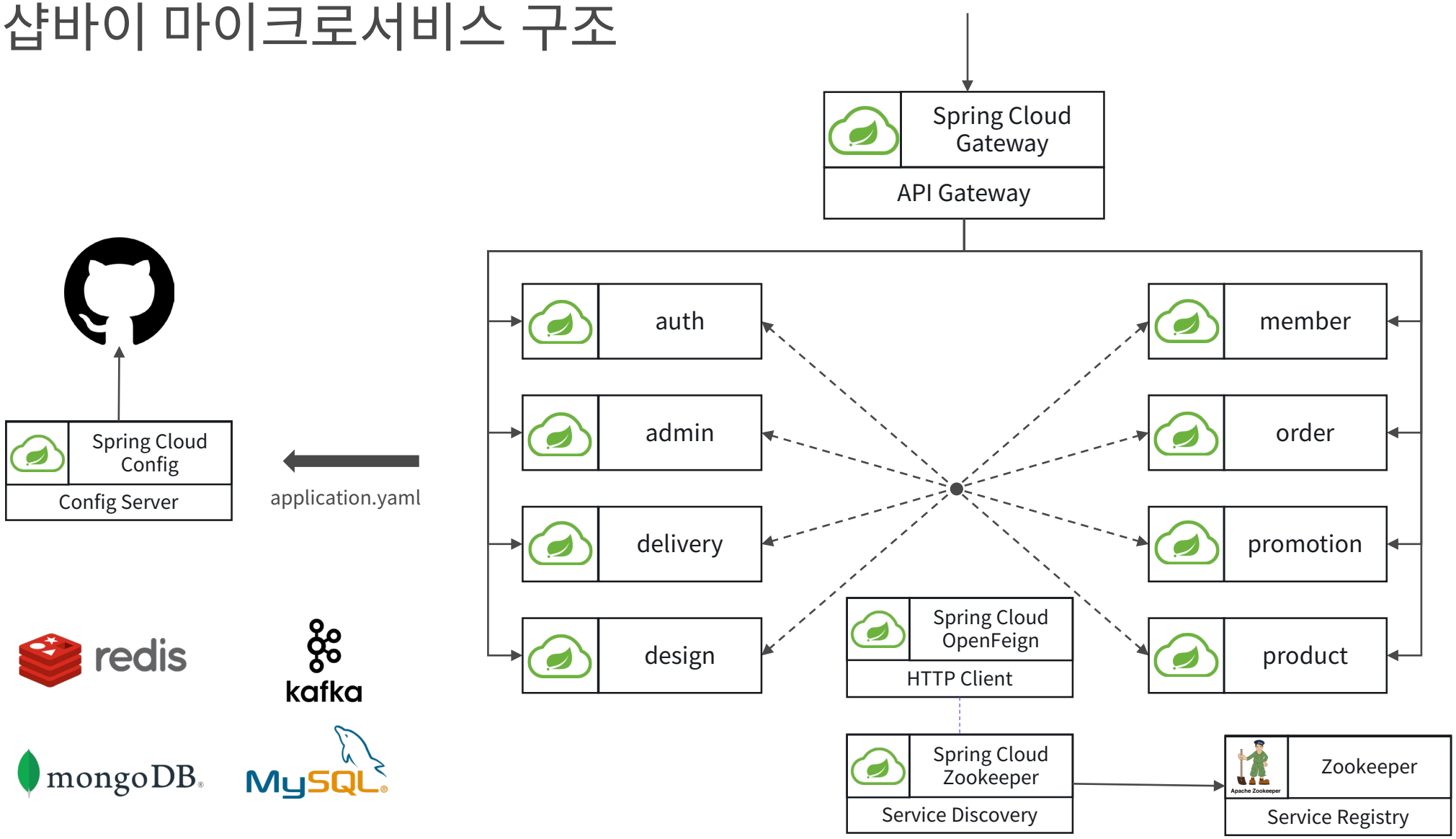
```
~ kubectl scale --replicas=5 deploy product
```

쿠버네티스 전환 준비하기

쿠버네티스 전환 준비하기

NHN FORWARD

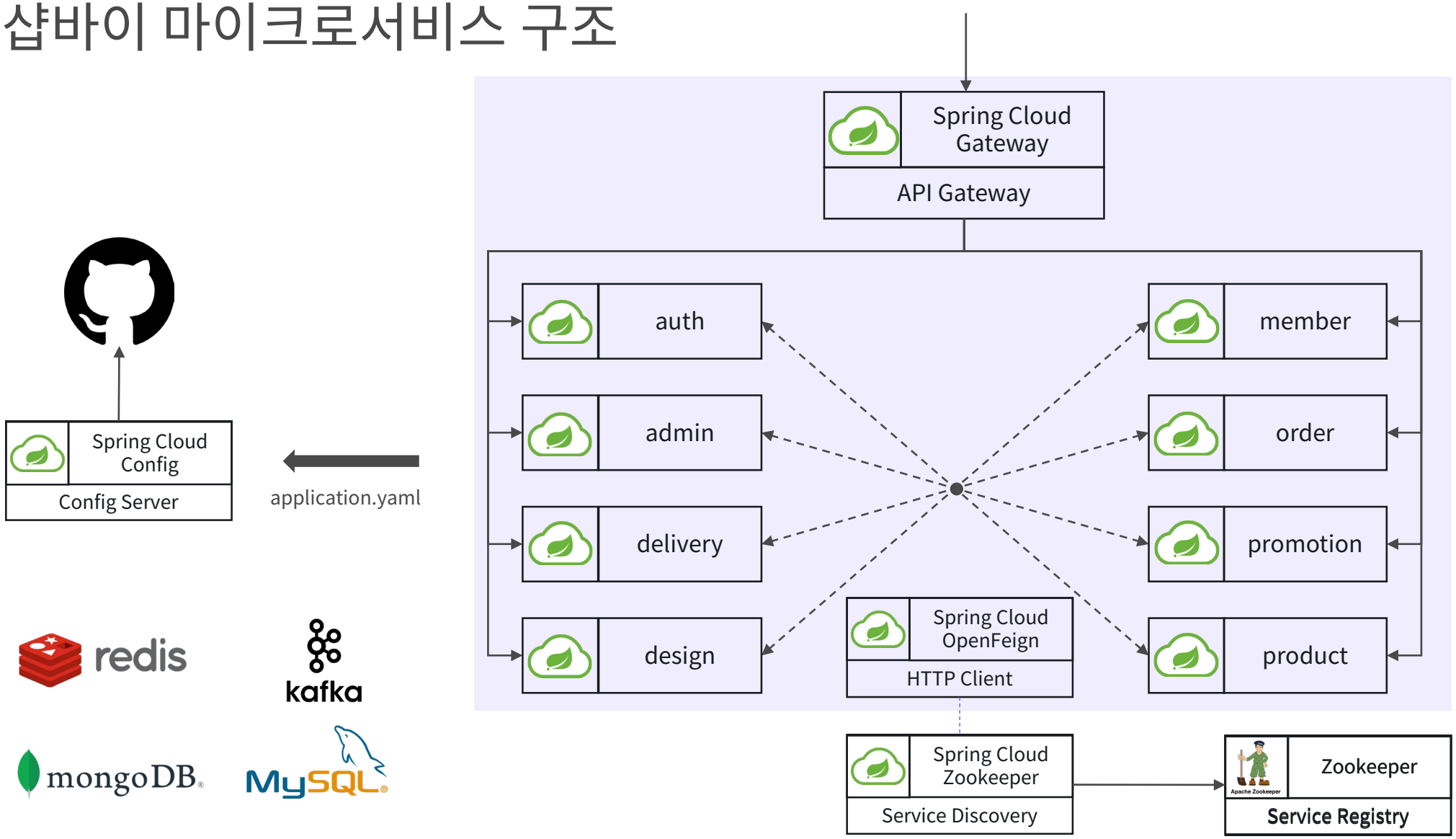
샵바이 마이크로서비스 구조



쿠버네티스 전환 준비하기

NHN FORWARD

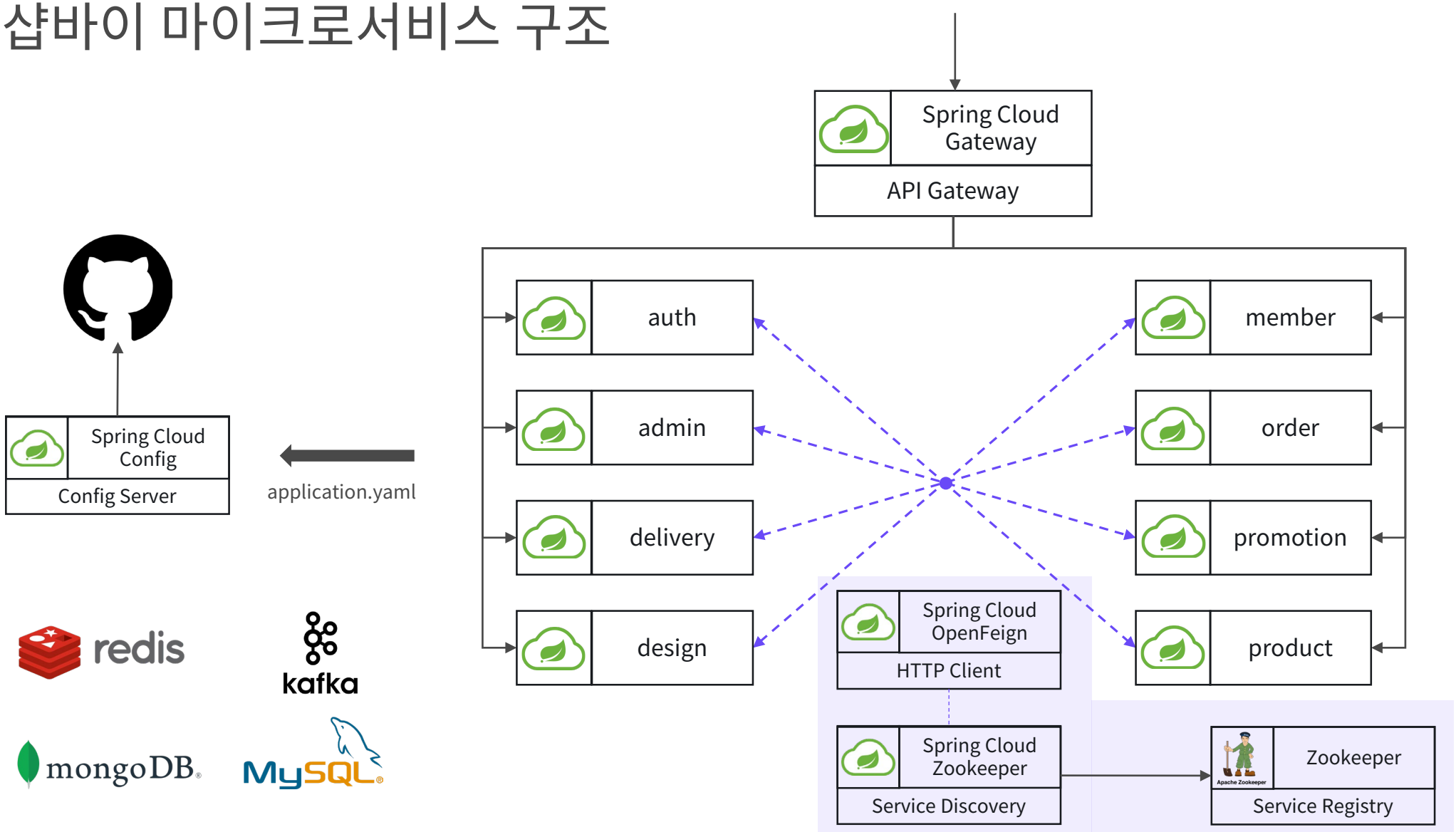
샵바이 마이크로서비스 구조



쿠버네티스 전환 준비하기

NHN FORWARD

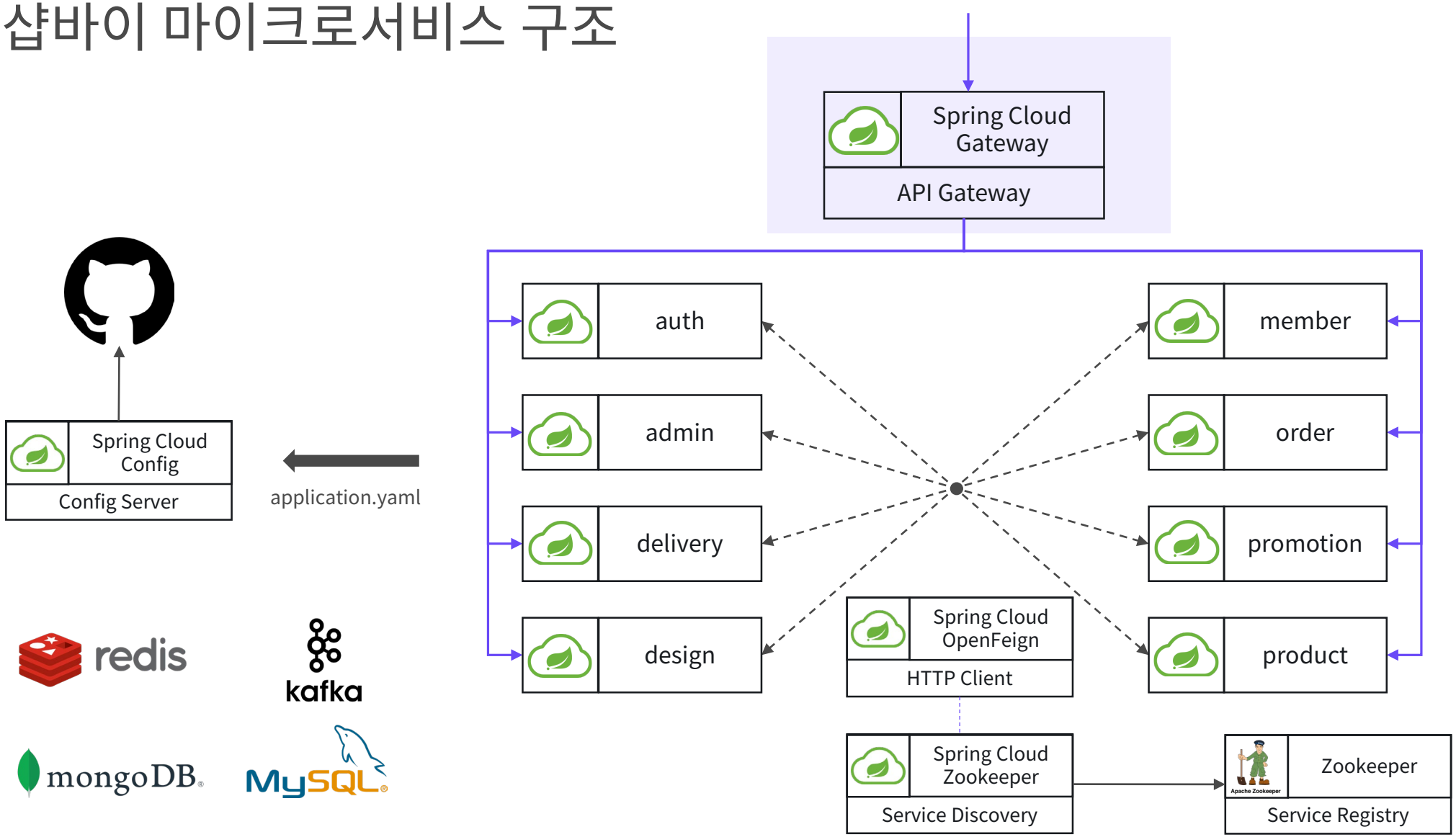
샵바이 마이크로서비스 구조



쿠버네티스 전환 준비하기

NHN FORWARD

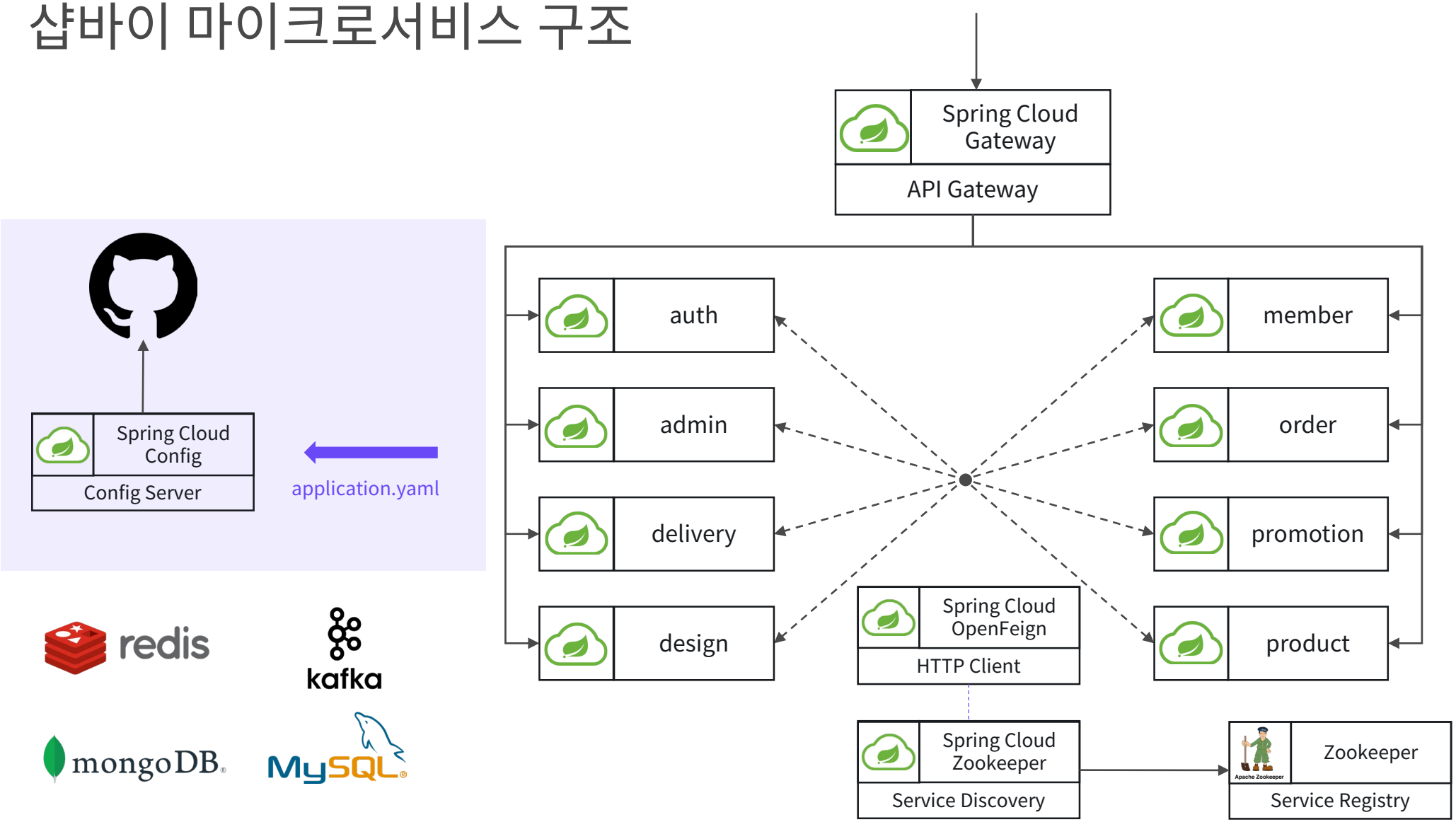
샵바이 마이크로서비스 구조



쿠버네티스 전환 준비하기

NHN FORWARD

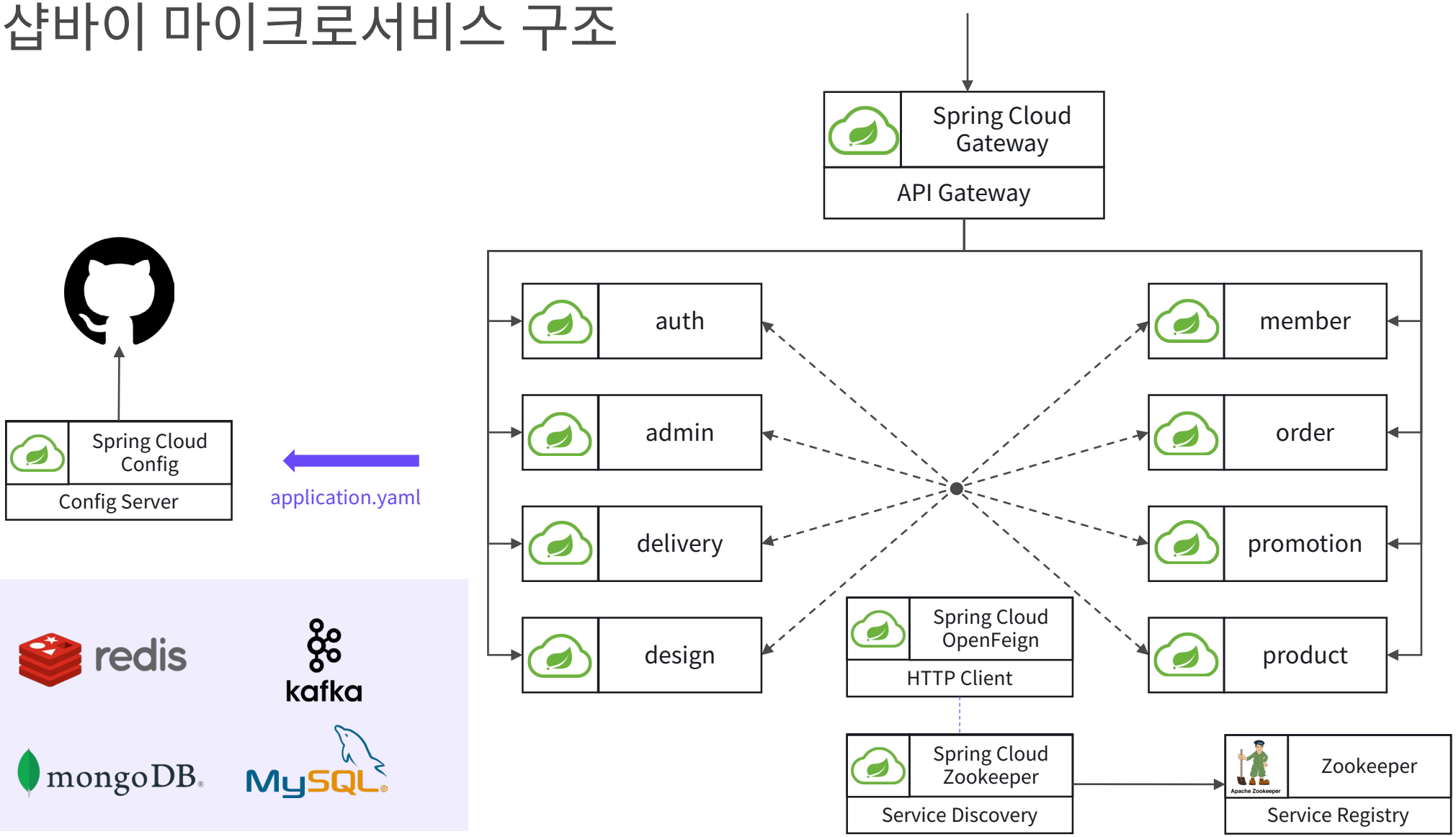
샵바이 마이크로서비스 구조



쿠버네티스 전환 준비하기

NHN FORWARD

샵바이 마이크로서비스 구조



쿠버네티스 전환 준비하기

코드 변경 없이 애플리케이션을
쿠버네티스로 전환할 수 있을까?

쿠버네티스에서 제공하는 기능들을 적극 활용하여
쿠버네티스로 전환이 가능할까?

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	
내부 통신	Spring Cloud OpenFeign	
서비스 디스커버리	Spring Cloud Zookeeper	
서비스 레지스트리	Zookeeper	
프로퍼티 파일 관리	Spring Cloud Config	
MySQL, Mongo, Kafka, Redis등	사용 중	

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Ingress
내부 통신	Spring Cloud OpenFeign	
서비스 디스커버리	Spring Cloud Zookeeper	
서비스 레지스트리	Zookeeper	
프로퍼티 파일 관리	Spring Cloud Config	
MySQL, Mongo, Kafka, Redis등	사용 중	

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Ingress
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	
서비스 레지스트리	Zookeeper	
프로퍼티 파일 관리	Spring Cloud Config	
MySQL, Mongo, Kafka, Redis등	사용 중	

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Ingress
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	-
서비스 레지스트리	Zookeeper	Service
프로퍼티 파일 관리	Spring Cloud Config	
MySQL, Mongo, Kafka, Redis등	사용 중	

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Ingress
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	-
서비스 레지스트리	Zookeeper	Service
프로퍼티 파일 관리	Spring Cloud Config	ConfigMap, Secret
MySQL, Mongo, Kafka, Redis등	사용 중	

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Ingress
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	-
서비스 레지스트리	Zookeeper	Service
프로퍼티 파일 관리	Spring Cloud Config	ConfigMap, Secret
MySQL, Mongo, Kafka, Redis등	사용 중	그대로 사용

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Ingress
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	-
서비스 레지스트리	Zookeeper	Service
프로퍼티 파일 관리	Spring Cloud Config	ConfigMap, Secret
MySQL, Mongo, Kafka, Redis등	사용 중	그대로 사용

쿠버네티스 전환 준비하기

서비스 레지스트리

Spring Cloud OpenFeign

- 어노테이션 기반으로 작성되는 선언적 REST Client

```
@FeignClient("order-internal")
interface OrderFeignClient {

    @GetMapping(value = ["/orders/{orderNo}"])
    fun getOrderByNo(
        @PathVariable orderNo: Int
    ): OrderResponse
}
```

```
$ curl order-api/orders/1
```

쿠버네티스 전환 준비하기

NHN FORWARD

서비스 레지스트리

Spring Cloud OpenFeign

- 어노테이션 기반으로 작성되는 선언적 REST Client

```
@FeignClient("order-internal")
interface OrderFeignClient {

    @GetMapping(value = ["/orders/{orderNo}"])
    fun getOrderByNo(
        @PathVariable orderNo: Int
    ): OrderResponse
}
```

```
$ curl order-api/orders/1
```

쿠버네티스 전환 준비하기

NHN FORWARD

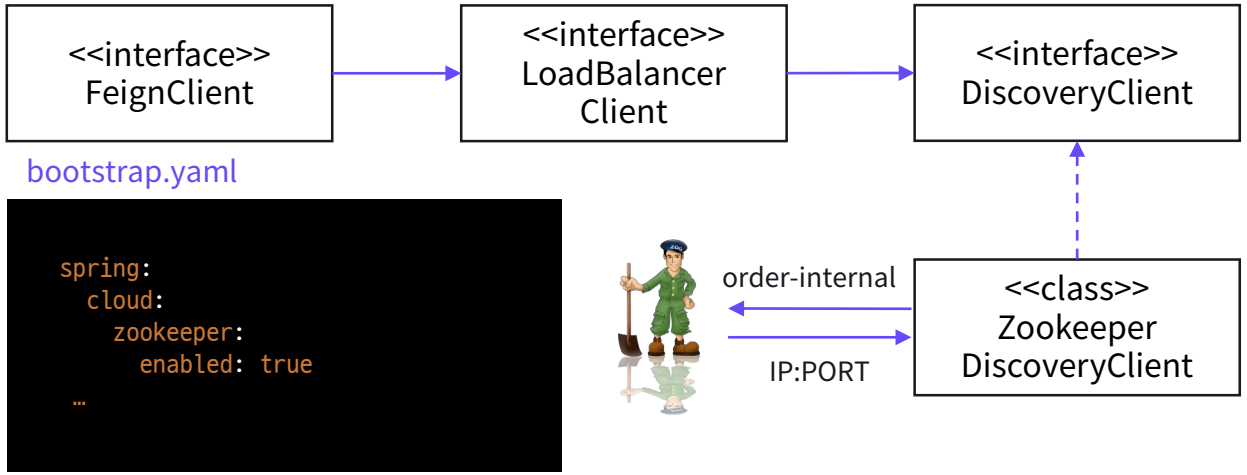
서비스 레지스트리

Spring Cloud OpenFeign

- 어노테이션 기반으로 작성되는 선언적 REST Client

```
@FeignClient("order-internal")
interface OrderFeignClient {

    @GetMapping(value = ["/orders/{orderNo}"])
    fun getOrderByNo(
        @PathVariable orderNo: Int
    ): OrderResponse
}
```

```
$ curl order-api/orders/1
```



bootstrap.yaml

```
spring:
  cloud:
    zookeeper:
      enabled: true
 …
```

build.gradle.kts

```
dependencies {

    implementation("org.springframework.cloud:spring-cloud-starter-openfeign")
    implementation("org.springframework.cloud:spring-cloud-starter-zookeeper-all")
    …
}
```

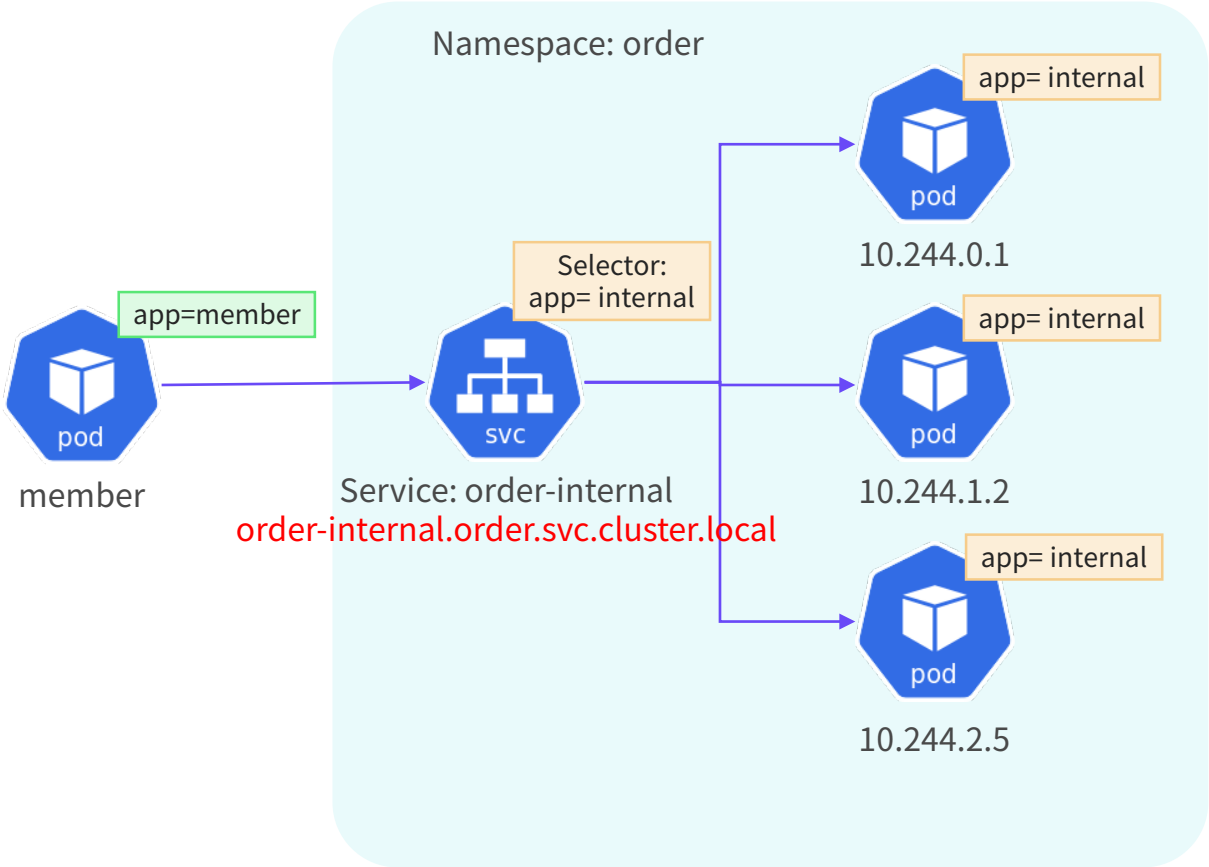
쿠버네티스 전환 준비하기

NHN FORWARD

Zookeeper? Kubernetes Service!

- K8s에는 서비스가 있다!

```
@FeignClient("order-internal")
interface OrderFeignClient {

    @GetMapping(value = ["/orders/{orderNo}"])
    fun getOrderByNo(
        @PathVariable orderNo: Int
    ): OrderResponse
}
```



쿠버네티스 전환 준비하기

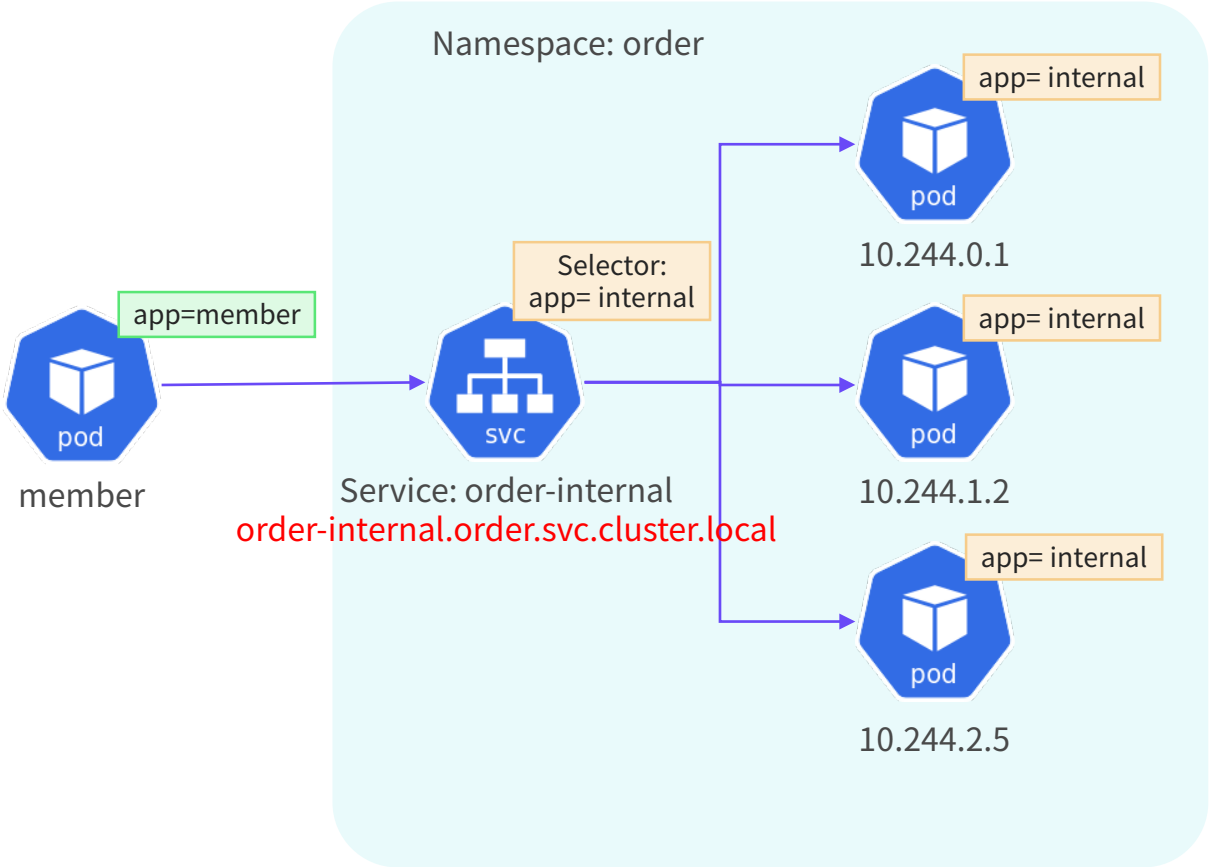
NHN FORWARD

Zookeeper? Kubernetes Service!

- K8s에는 서비스가 있다!

```
@FeignClient(url = "order-internal.order.svc.cluster.local")
interface OrderFeignClient {

    @GetMapping(value = ["/orders/{orderNo}"])
    fun getOrderByNo(
        @PathVariable orderNo: Int
    ): OrderResponse
}
```

- 코드 변경 발생
- 코드가 쿠버네티스에 의존적
- 기존 환경에서 작동 안 됨



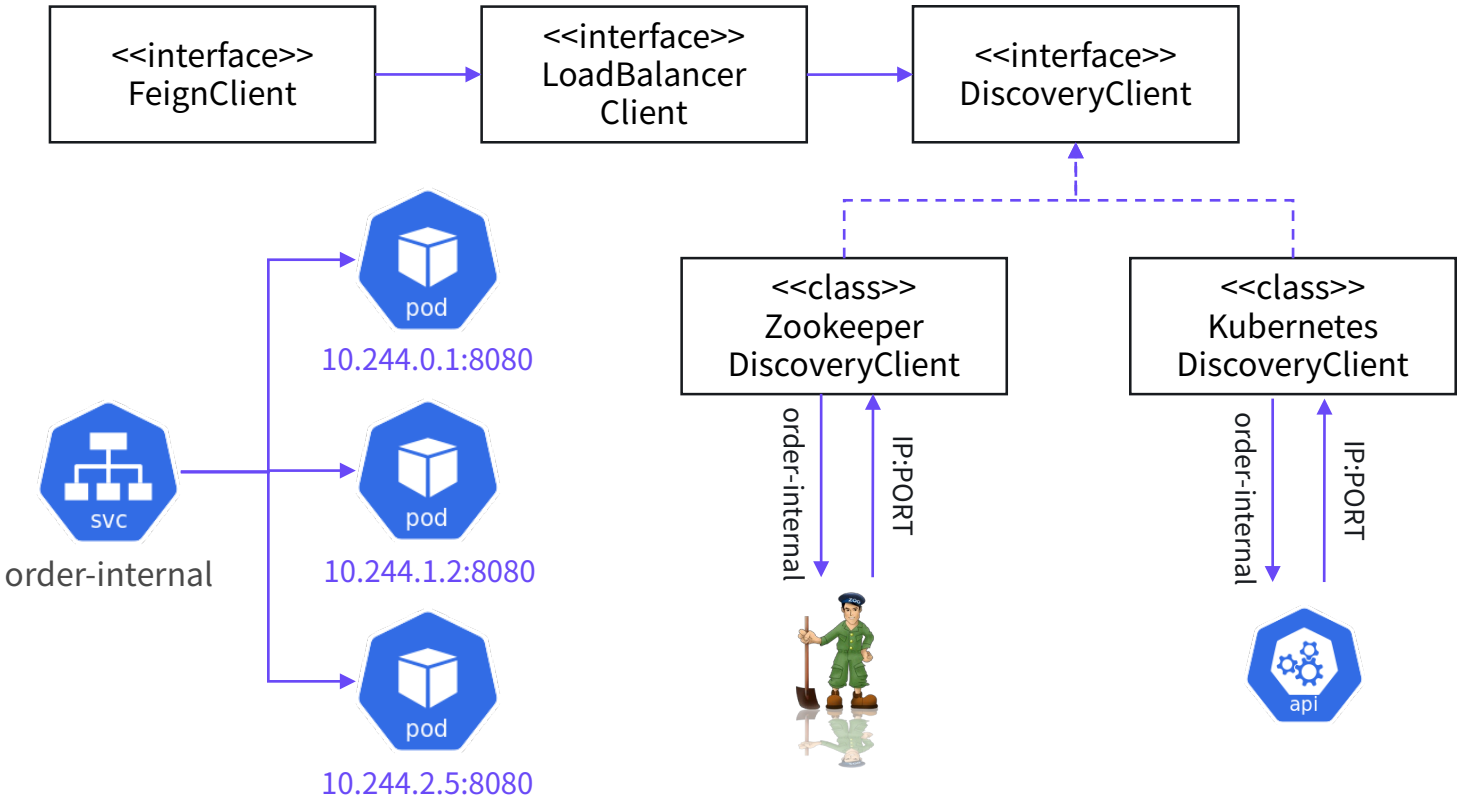
쿠버네티스 전환 준비하기

NHN FORWARD

Spring Cloud Kubernetes!

- Spring Cloud에서 제공하는 인터페이스 중 몇 가지에 대해 쿠버네티스 리소스를 활용한 구현체를 제공
  - DiscoveryClient 활용

```
@FeignClient("order-internal")
interface OrderFeignClient {

    @GetMapping(value = ["/orders/{orderNo}"])
    fun getOrderByNo(
        @PathVariable orderNo: Int
    ): OrderResponse
}
```



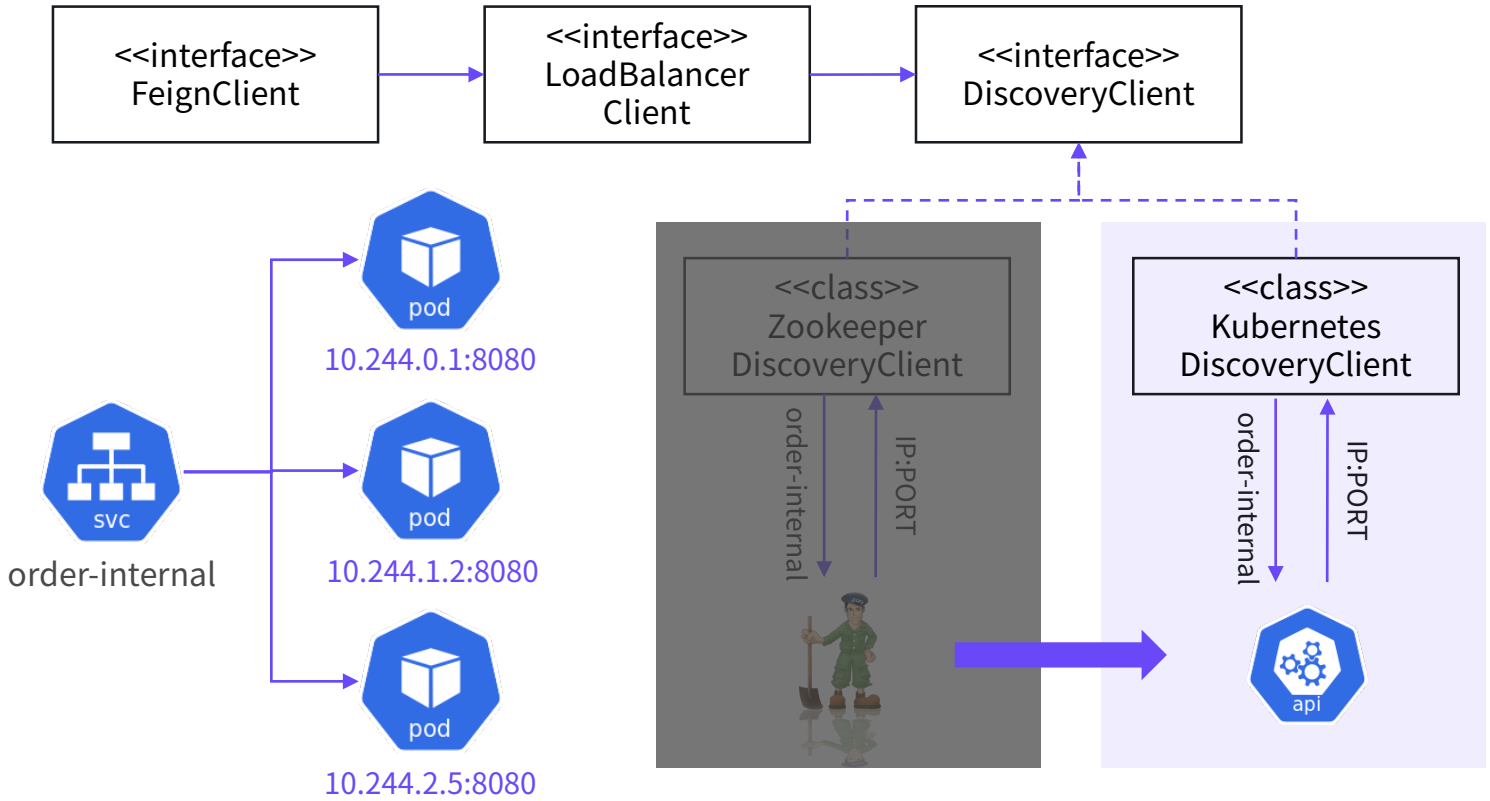
쿠버네티스 전환 준비하기

NHN FORWARD

Spring Cloud Kubernetes!

- Spring Cloud에서 제공하는 인터페이스 중 몇 가지에 대해 쿠버네티스 리소스를 활용한 구현체를 제공
  - DiscoveryClient 활용

```
@FeignClient("order-internal")
interface OrderFeignClient {

    @GetMapping(value = ["/orders/{orderNo}"])
    fun getOrderByNo(
        @PathVariable orderNo: Int
    ): OrderResponse
}
```



쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Spring Cloud Gateway
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	Spring Cloud Kubernetes
서비스 레지스트리	Zookeeper	Service
프로퍼티 파일 관리	Spring Cloud Config	ConfigMap, Secret
MySQL, Mongo, Kafka, Redis등	사용 중	그대로 사용

쿠버네티스 전환 준비하기

NHN FORWARD

Spring Cloud Gateway

- Spring Framework 5 기반
- Host & Path 기반 L7 라우팅 지원
- Custom Filter 구현 가능
- Path Rewrite 기능 제공

```yaml
spring:
  cloud:
    gateway:
      routes:
        - id: member-admin
          uri: lb://member-admin
          predicates:
            - Host=admin-api.e-ncp.com
            - Path=/members/**
          filters:
            - name: AdminAccessToken
            - RewritePath=/(?<segment>.*),     /backend/$\{segment}
        - id: product
          uri: lb://product-admin
          predicates:
            - Host=admin-api.e-ncp.com
            - Path=/products/**
          filters:
            - name: AdminAccessToken
            - RewritePath=/(?<segment>.*),     /backend/$\{segment}
```

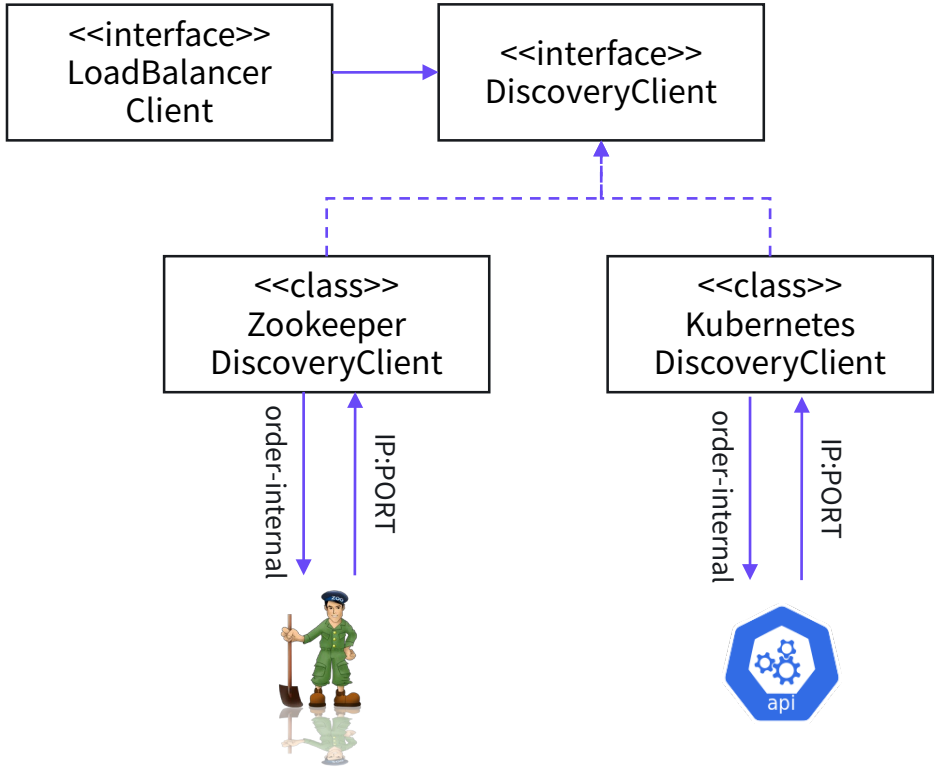
쿠버네티스 전환 준비하기

NHN FORWARD

Spring Cloud Gateway

- Spring Framework 5 기반
- Host & Path 기반 L7 라우팅 지원
- Custom Filter 구현 가능
- Path Rewrite 기능 제공

```yaml
spring:
  cloud:
    gateway:
      routes:
        - id: member-admin
          uri: lb://member-admin
          predicates:
            - Host=admin-api.e-ncp.com
            - Path=/members/**
          filters:
            - name: AdminAccessToken
            - RewritePath=/(?<segment>.*),     /backend/$\{segment}
        - id: product
          uri: lb://product-admin
          predicates:
            - Host=admin-api.e-ncp.com
            - Path=/products/**
          filters:
            - name: AdminAccessToken
            - RewritePath=/(?<segment>.*),     /backend/$\{segment}
```



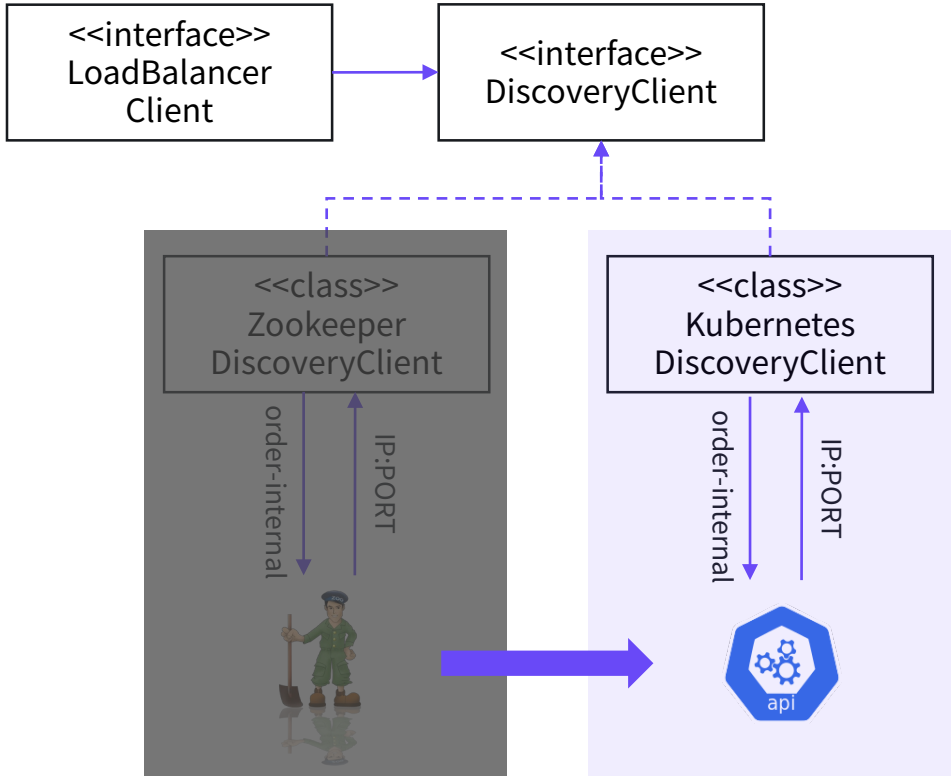
쿠버네티스 전환 준비하기

NHN FORWARD

Spring Cloud Gateway

- Spring Framework 5 기반
- Host & Path 기반 L7 라우팅 지원
- Custom Filter 구현 가능
- Path Rewrite 기능 제공

```yaml
spring:
  cloud:
    gateway:
      routes:
        - id: member-admin
          uri: lb://member-admin
          predicates:
            - Host=admin-api.e-ncp.com
            - Path=/members/**
          filters:
            - name: AdminAccessToken
            - RewritePath=/(?<segment>.*),     /backend/$\{segment}
        - id: product
          uri: lb://product-admin
          predicates:
            - Host=admin-api.e-ncp.com
            - Path=/products/**
          filters:
            - name: AdminAccessToken
            - RewritePath=/(?<segment>.*),     /backend/$\{segment}
```



쿠버네티스 전환 준비하기

API Gateway

- Ingress vs Spring Cloud Gateway

기능	Ingress (Controller)		Spring Cloud Gateway
	Nginx-ingress-controller	Istio-ingressgateway	Spring Cloud Gateway
L7 라우팅	O	O	O
Path Rewrite	O	O	O
커스텀 필터	Lua	Lua	Java/Kotlin
리로딩	O	O	Spring Cloud Config

쿠버네티스 전환 준비하기

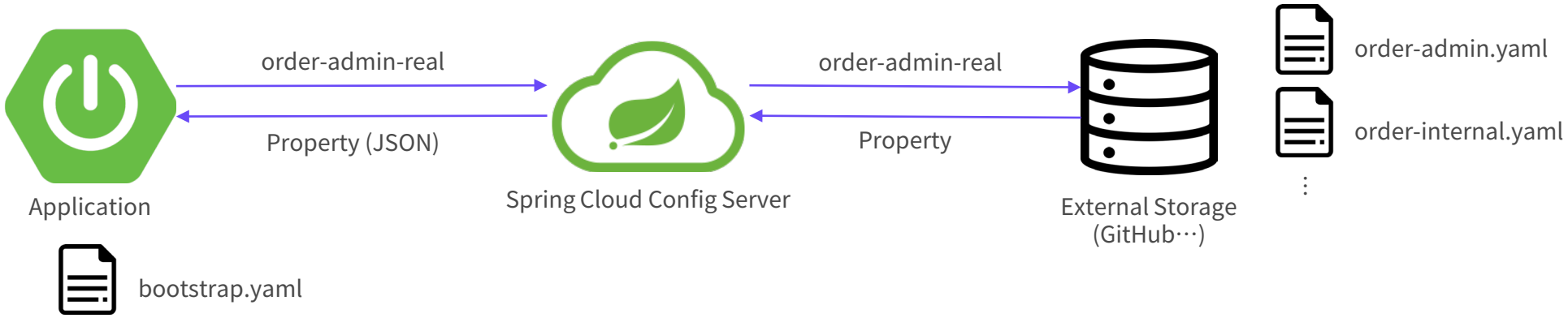
Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Spring Cloud Gateway
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	Spring Cloud Kubernetes
서비스 레지스트리	Zookeeper	Service
프로퍼티 파일 관리	Spring Cloud Config	ConfigMap, Secret
MySQL, Mongo, Kafka, Redis등	사용 중	그대로 사용

쿠버네티스 전환 준비하기

NHN FORWARD

프로퍼티 파일 관리



build.gradle.kts

```
dependencies {

    implementation("org.springframework.cloud:spring-cloud-starter-config")
    …
}
```

bootstrap.yaml

```
spring:
  application:
    name: order-admin
  cloud:
    config:
      uri: http://config-server.com:8080
      enabled: true
…
```

쿠버네티스 전환 준비하기

NHN FORWARD

프로퍼티 파일 관리

- Kubernetes ConfigMap & Secret


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: application
  namespace: test
data:
  application-k8s.yaml: |
    config:
      stage: k8s
```

쿠버네티스 전환 준비하기

NHN FORWARD

프로퍼티 파일 관리

- Kubernetes ConfigMap & Secret


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: application
  namespace: test
data:
  application-k8s.yaml: |
    config:
      stage: k8s
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: springboot
spec:
  imagePullSecrets:
    - name: harbor
  containers:
    - image: private-registry/test/webservice
      imagePullPolicy: Always
      name: spring-boot
      env:
        - name: SPRING_PROFILES_ACTIVE
          value: k8s
      volumeMounts:
        - name: application
          mountPath: /app/resources/application-k8s.yaml
          subPath: application.yaml
          readOnly: true
  volumes:
    - name: application
      configMap:
        name: application
```

쿠버네티스 전환 준비하기

NHN FORWARD

프로퍼티 파일 관리

- Kubernetes ConfigMap & Secret


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: application
  namespace: test
data:
  application-k8s.yaml: |
    config:
      stage: k8s
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: springboot
spec:
  imagePullSecrets:
    - name: harbor
  containers:
    - image: private-registry/test/webservice
      imagePullPolicy: Always
      name: spring-boot
      env:
        - name: SPRING_PROFILES_ACTIVE
          value: k8s
      volumeMounts:
        - name: application
          mountPath: /app/resources/application-k8s.yaml
          subPath: application.yaml
          readOnly: true
  volumes:
    - name: application
      configMap:
        name: application
```

쿠버네티스 전환 준비하기

NHN FORWARD

프로퍼티 파일 관리

- Kubernetes ConfigMap & Secret


```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: application
  namespace: test
data:
  application-k8s.yaml: |
    config:
      stage: k8s
```

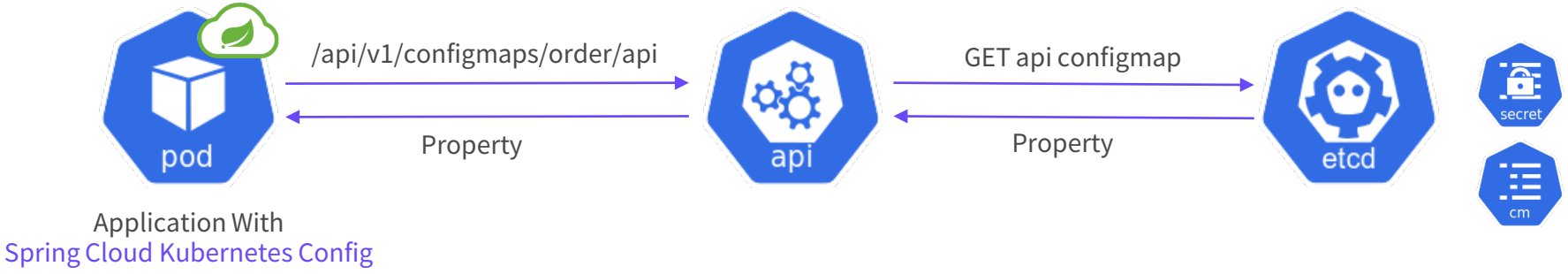
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: springboot
spec:
  imagePullSecrets:
    - name: harbor
  containers:
    - image: private-registry/test/webservice
      imagePullPolicy: Always
      name: spring-boot
      env:
        - name: SPRING_PROFILES_ACTIVE
          value: k8s
      volumeMounts:
        - name: application
          mountPath: /app/resources/application-k8s.yaml
          subPath: application.yaml
          readOnly: true
  volumes:
    - name: application
      configMap:
        name: application
```

쿠버네티스 전환 준비하기

NHN FORWARD

Spring Cloud Kubernetes!

- Spring Cloud에서 제공하는 여러 인터페이스 중 몇 가지에 대해 쿠버네티스 리소스를 활용한 구현체를 제공
  - PropertySource




```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: admin
  namespace: order
data:
  application.yaml: |-
    spring:
      application:
        name: order-admin
```

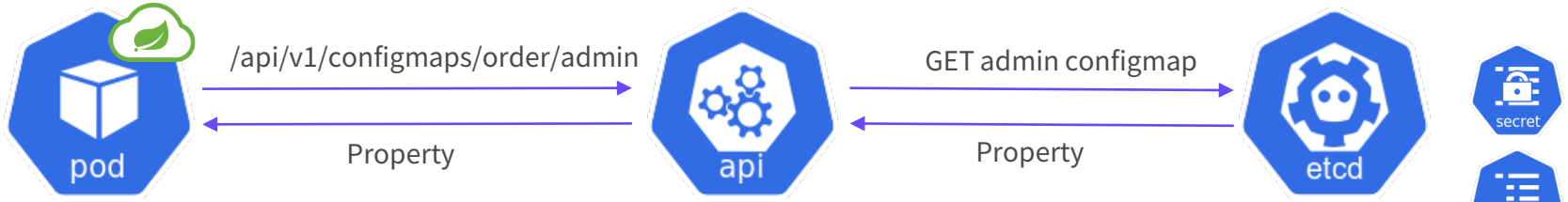
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: kafka
  namespace: common-config
data:
  application.yaml: |-
    spring:
      kafka:
        bootstrap-servers: IP:PORT
```

쿠버네티스 전환 준비하기

NHN FORWARD

Spring Cloud Kubernetes!

- Spring Cloud에서 제공하는 여러 인터페이스 중 몇 가지에 대해 쿠버네티스 리소스를 활용한 구현체를 제공
  - PropertySource



Application With
Spring Cloud Kubernetes Config

bootstrap.yaml

```yaml
spring:
  cloud:
    kubernetes:
      enabled: true
      config:
        enabled: true
        sources:
          - name: admin
            namespace: order
          - name: kafka
            namespace: common-config
```

build.gradle.kts

```gradle
dependencies {

    implementation("org.springframework.cloud:spring-cloud-starter-kubernetes-client-all")
    …
}
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: admin
  namespace: order
data:
  application.yaml: |-
    spring:
      application:
        name: order-admin
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: kafka
  namespace: common-config
data:
  application.yaml: |-
    spring:
      kafka:
        bootstrap-servers: IP:PORT
```

쿠버네티스 전환 준비하기

Dependencies

Dependency	VM (On-Prem)	Kubernetes
API Gateway	Spring Cloud Gateway	Spring Cloud Gateway
내부 통신	Spring Cloud OpenFeign	Spring Cloud OpenFeign
서비스 디스커버리	Spring Cloud Zookeeper	Spring Cloud Kubernetes
서비스 레지스트리	Zookeeper	Service
프로퍼티 파일 관리	Spring Cloud Config	ConfigMap, Secret
MySQL, Mongo, Kafka, Redis등	사용 중	그대로 사용

쿠버네티스 전환 준비하기

NHN FORWARD

기존 환경과 호환: 프로파일을 사용!

build.gradle.kts

```
dependencies {
    implementation("org.springframework.cloud:spring-cloud-starter-openfeign")
    implementation("org.springframework.cloud:spring-cloud-starter-kubernetes-client-all")
    implementation("org.springframework.cloud:spring-cloud-starter-zookeeper-all")
    implementation("org.springframework.cloud:spring-cloud-starter-config")
    ...
}
```

쿠버네티스 전환 준비하기

NHN FORWARD

기존 환경과 호환: 프로파일을 사용!

build.gradle.kts

```
dependencies {
    implementation("org.springframework.cloud:spring-cloud-starter-openfeign")
    implementation("org.springframework.cloud:spring-cloud-starter-kubernetes-client-all")
    implementation("org.springframework.cloud:spring-cloud-starter-zookeeper-all")
    implementation("org.springframework.cloud:spring-cloud-starter-config")
    …
}
```

bootstrap.yaml (Private Cloud)

```
spring:
  application:
    name: order-admin
  cloud:
    config:
      uri: http://config-server.com:8080
      enabled: true
    zookeeper:
      enabled: true
    kubernetes:
      enabled: false
```

쿠버네티스 전환 준비하기

NHN FORWARD

기존 환경과 호환: 프로파일을 사용!

build.gradle.kts

```
dependencies {
    implementation("org.springframework.cloud:spring-cloud-starter-openfeign")
    implementation("org.springframework.cloud:spring-cloud-starter-kubernetes-client-all")
    implementation("org.springframework.cloud:spring-cloud-starter-zookeeper-all")
    implementation("org.springframework.cloud:spring-cloud-starter-config")
    …
}
```

bootstrap.yaml (Private Cloud)

```yaml
spring:
  application:
    name: order-admin
  cloud:
    config:
      uri: http://config-server.com:8080
      enabled: true
    zookeeper:
      enabled: true
    kubernetes:
      enabled: false
```

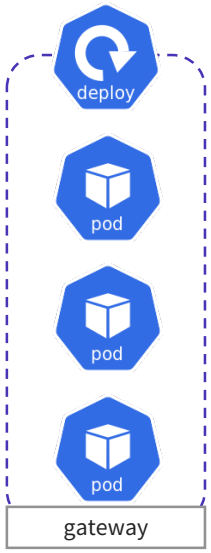
bootstrap-kubernetes.yaml (K8S)

```yaml
spring:
  profiles: kubernetes
  cloud:
    zookeeper:
      enabled: false
    config:
      enabled: false
    kubernetes:
      enabled: true
      loadbalancer:
        enabled: true
        mode: service # default: pod
      discovery:
        enabled: true
        all-namespaces: true
      config:
        enabled: true
        sources:
          - name: admin
            namespace: order
          - name: kafka
            namespace: common-config
```

쿠버네티스에 배포하기

쿠버네티스에 배포하기

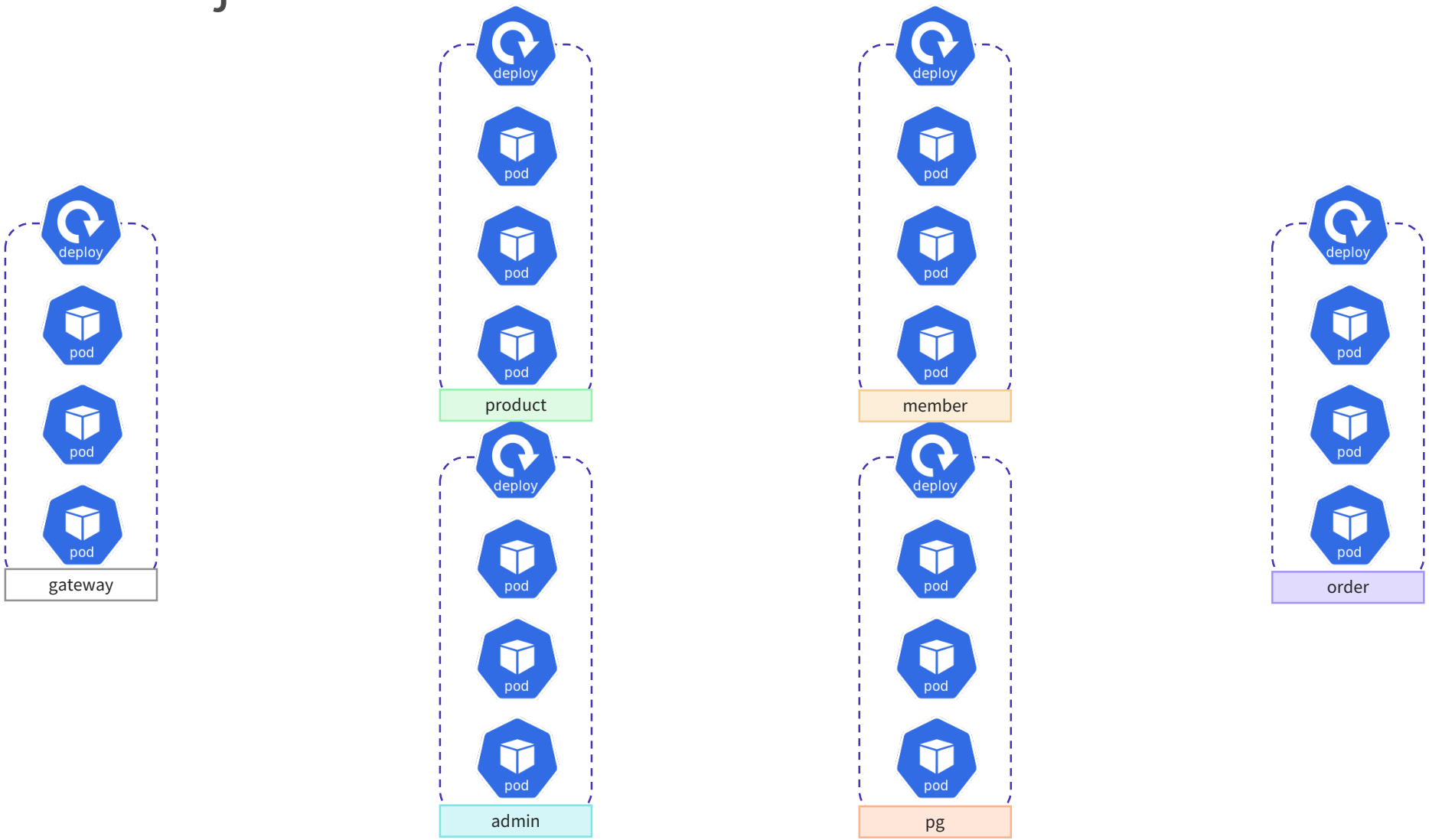
Kubernetes Objects



쿠버네티스에 배포하기

NHN FORWARD

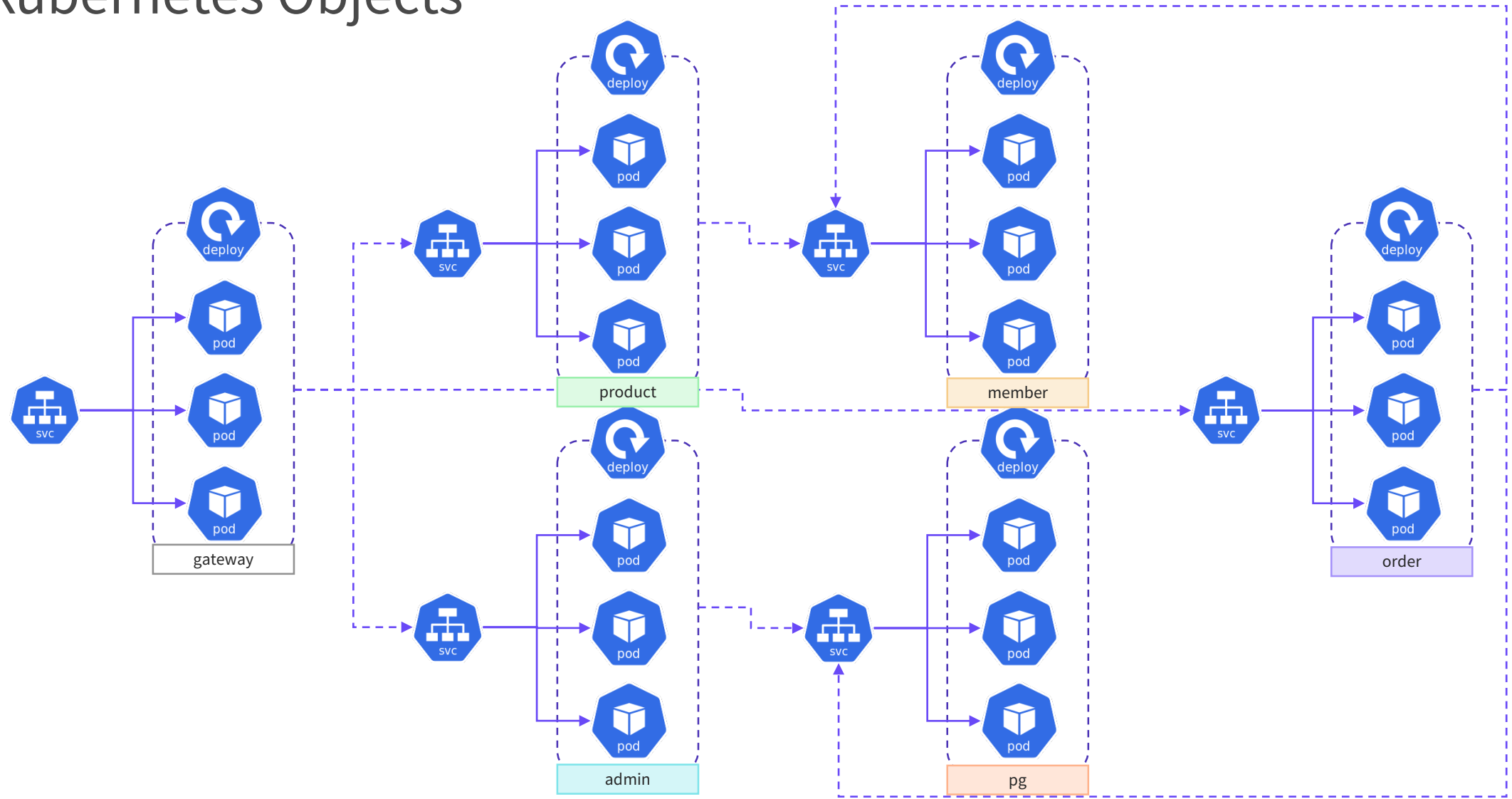
Kubernetes Objects



쿠버네티스에 배포하기

NHN FORWARD

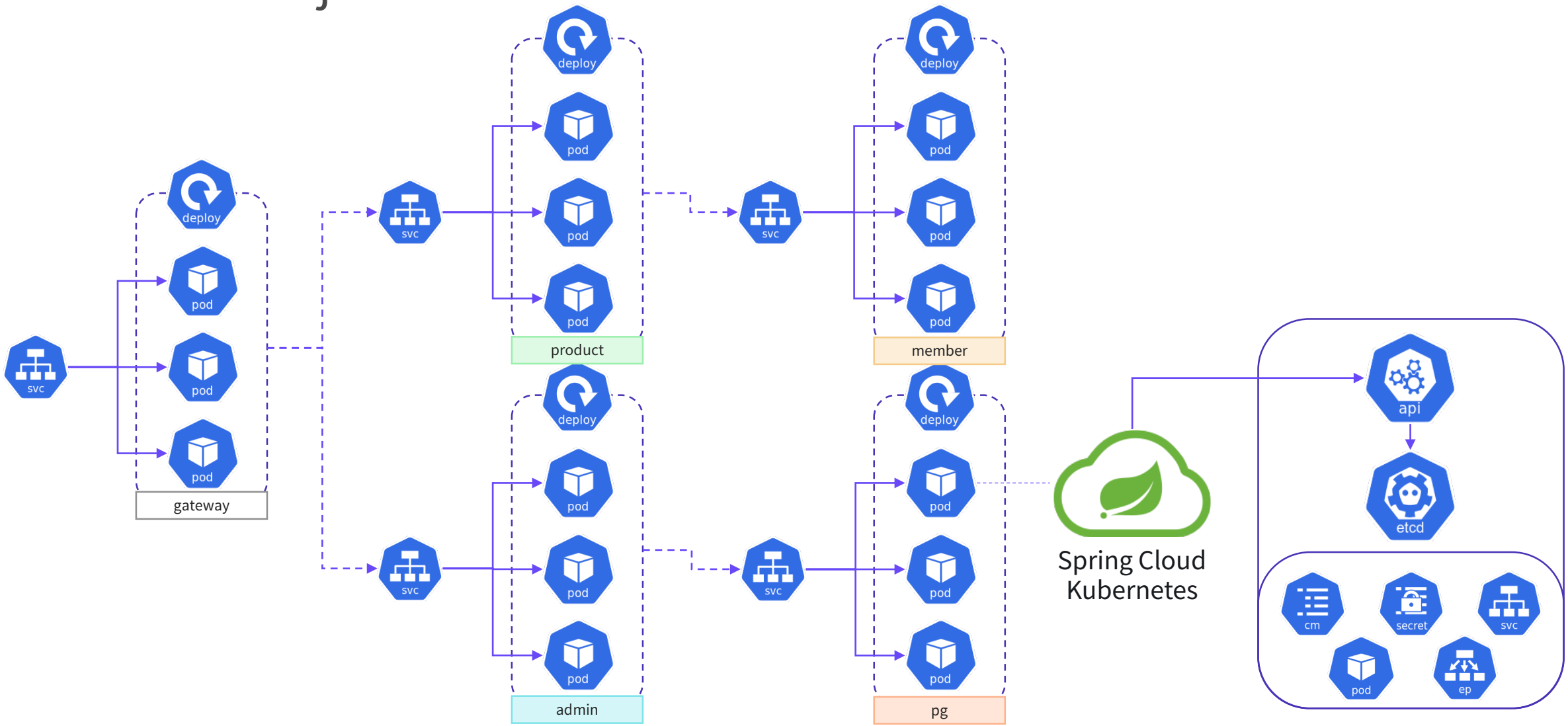
Kubernetes Objects



쿠버네티스에 배포하기

NHN FORWARD

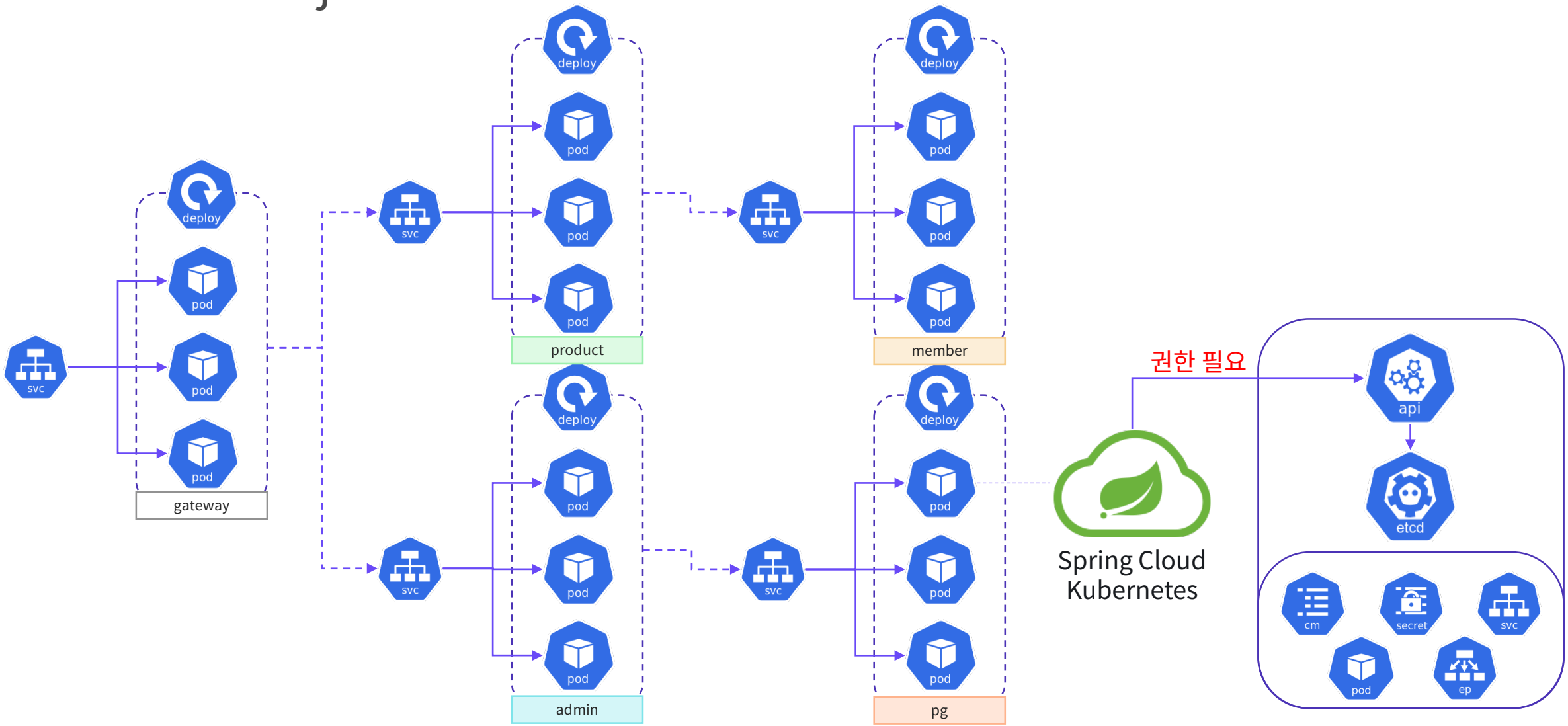
Kubernetes Objects



쿠버네티스에 배포하기

NHN FORWARD

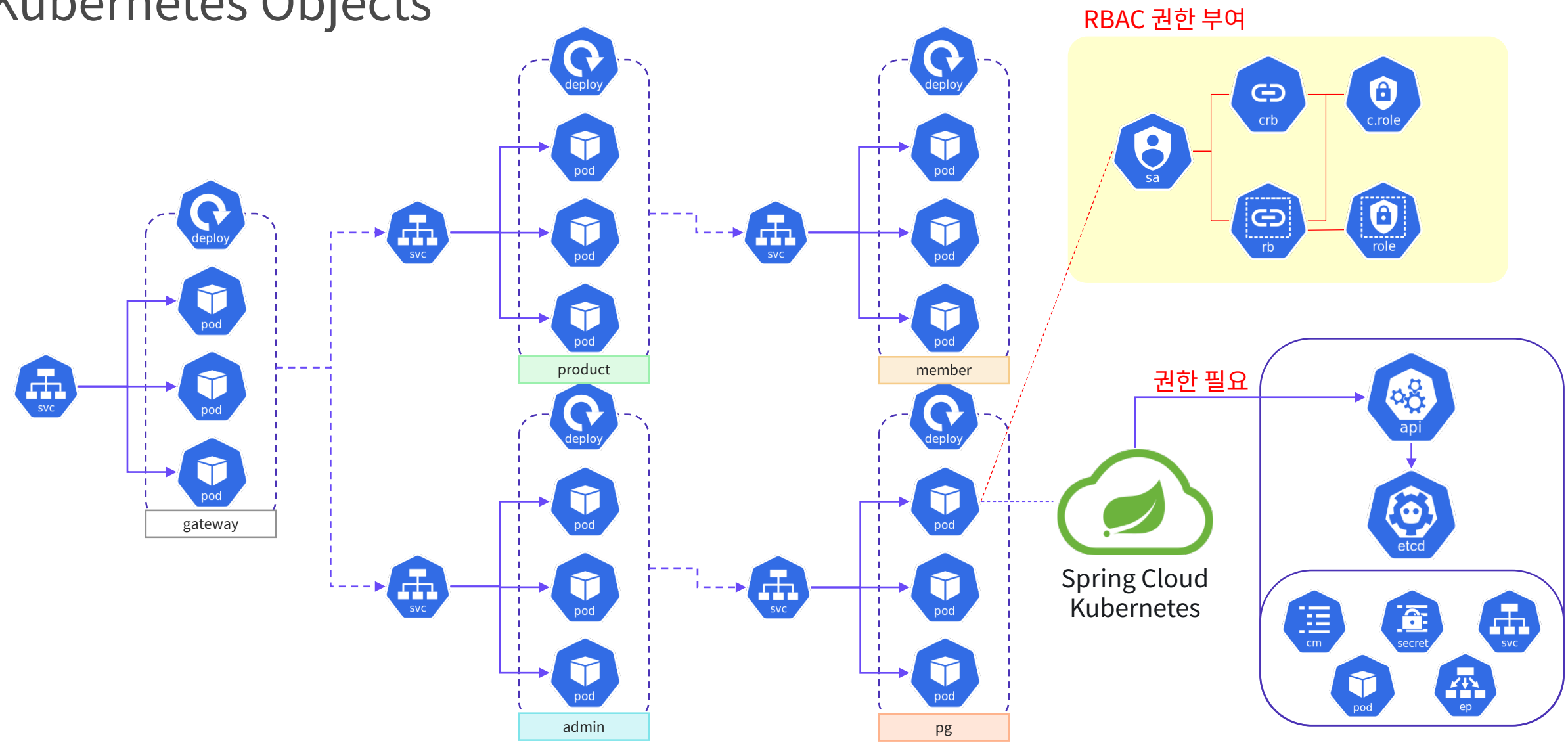
Kubernetes Objects



쿠버네티스에 배포하기

NHN FORWARD

Kubernetes Objects



쿠버네티스에 배포하기

Kubernetes Objects

Workloads	Networking	Configuration	RBAC
deploy pod	svc	cm secret	rb c.role sa role crb

쿠버네티스에 배포하기

NHN FORWARD

애플리케이션 배포 시 필요한 필수 리소스들…

cm

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: internal
  namespace: order
data:
  bootstrap.yaml: |-
    server:
      servlet:
        …
    …
```

sa

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: internal
  namespace: order
```

svc

```yaml
apiVersion: v1
kind: Service
metadata:
  name: order-internal
  namespace: order
spec:
  type: ClusterIP
  selector:
    app: internal
  ports:
    - port: 80
      protocol: TCP
      targetPort: 7000
```

crb

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: order-internal
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: spring-k8s
subjects:
  - kind: ServiceAccount
    name: internal
    namespace: order
```

deploy

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: internal
  name: internal
  namespace: order
spec:
  replicas: 4
  selector:
    matchLabels:
      app: internal
  template:
    metadata:
      labels:
        app: internal
    spec:
      containers:
        - name: internal
          image: private-registry-url/order-internal:alpha
          imagePullPolicy: Always
          restartPolicy: Always
          serviceAccount: internal
          ports:
            - containerPort: 7000
              name: http
              protocol: TCP
```

쿠버네티스에 배포하기

NHN FORWARD

반복…반복

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: internal
  namespace: admin
data:
  bootstrap.yaml: |-
    server:
      servlet:
        …
    …
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: admin-internal
  namespace: admin
spec:
  type: ClusterIP
  selector:
    app: internal
  ports:
    - port: 80
      protocol: TCP
      targetPort: 8000
```

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: internal
  namespace: admin
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-internal
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: spring-k8s
subjects:
  - kind: ServiceAccount
    name: internal
    namespace: admin
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: internal
  name: internal
  namespace: admin
spec:
  replicas: 2
  selector:
    matchLabels:
      app: internal
  template:
    metadata:
      labels:
        app: internal
    spec:
      containers:
        - name: internal
          image: private-registry-url/admin-internal:1.0.1
          imagePullPolicy: IfNotPresent
          restartPolicy: Always
          serviceAccount: internal
          ports:
            - containerPort: 8000
              name: http
              protocol: TCP
```

쿠버네티스에 배포하기

NHN FORWARD

마이크로서비스가 몇 갠데!!

cm

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: shop
  namespace: admin
data:
  bootstrap.yaml: |-
    server:
      servlet:
        …
    …
```

sa

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: shop
  namespace: admin
```

svc

```yaml
apiVersion: v1
kind: Service
metadata:
  name: admin-shop
  namespace: admin
spec:
  type: ClusterIP
  selector:
    app: shop
  ports:
    - port: 80
      protocol: TCP
      targetPort: 9000
```

crb

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-shop
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: spring-k8s
subjects:
  - kind: ServiceAccount
    name: shop
    namespace: admin
```

deploy

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: shop
  name: shop
  namespace: admin
spec:
  replicas: 3
  selector:
    matchLabels:
      app: shop
  template:
    metadata:
      labels:
        app: shop
    spec:
      containers:
        - name: internal
          image: private-registry-url/admin-shop:1.0.2
          imagePullPolicy: IfNotPresent
          restartPolicy: Always
          serviceAccount: internal
          ports:
            - containerPort: 9000
              name: http
              protocol: TCP
```

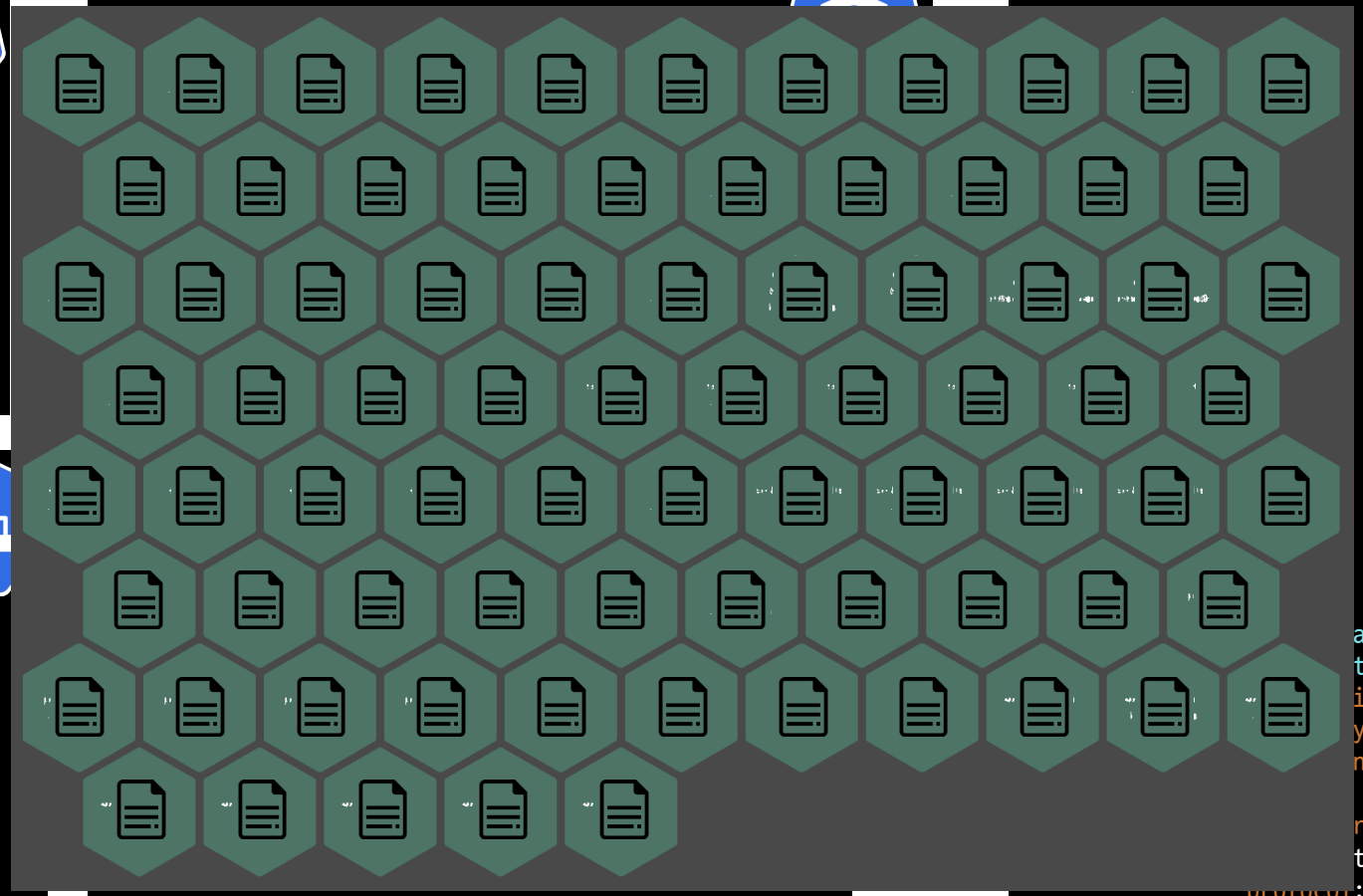
쿠버네티스에 배포하기

NHN FORWARD

마이크로서비스가 몇 갠데!!

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: shop
  namespace: admin
data:
  bootstrap.yaml: |-
    server:
      servlet:
        …
    …
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: admin-shop
  namespace: admin
spec:
  type: ClusterIP
  selector:
    app: shop
  ports:
    - port: 80
      protocol: TCP
      targetPort: 9000
```



```yaml
al
te-registry-url/admin-shop:1.0.2
icy: IfNotPresent
y: Always
nt: internal

rPort: 9000
tp
: TCP
```

쿠버네티스에 배포하기

NHN FORWARD

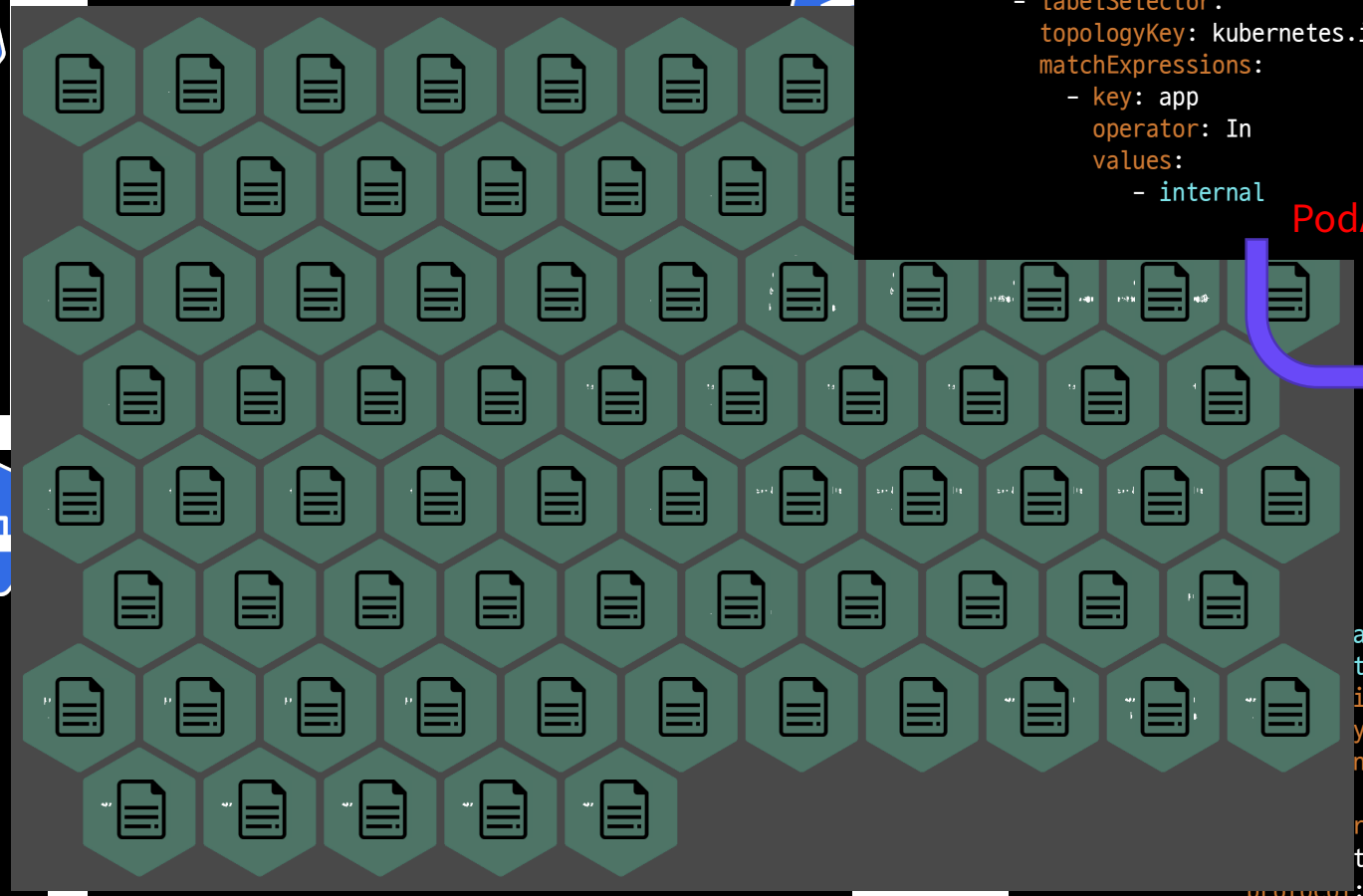
마이크로서비스가 몇 갠데!!

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: shop
  namespace: admin
data:
  bootstrap.yaml: |-
    server:
      servlet:
        …
    …
```

cm

```yaml
apiVersion: v1
kind: Service
metadata:
  name: admin-shop
  namespace: admin
spec:
  type: ClusterIP
  selector:
    app: shop
  ports:
    - port: 80
      protocol: TCP
      targetPort: 9000
```

svc



```yaml
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
          topologyKey: kubernetes.io/hostname
          matchExpressions:
            - key: app
              operator: In
              values:
                - internal
```

PodAntiAffinity 추가해야 해요!

deploy

```
al
te-registry-url/admin-shop:1.0.2
icy: IfNotPresent
y: Always
nt: internal

rPort: 9000
tp
protocol: TCP
```

쿠버네티스에 배포하기

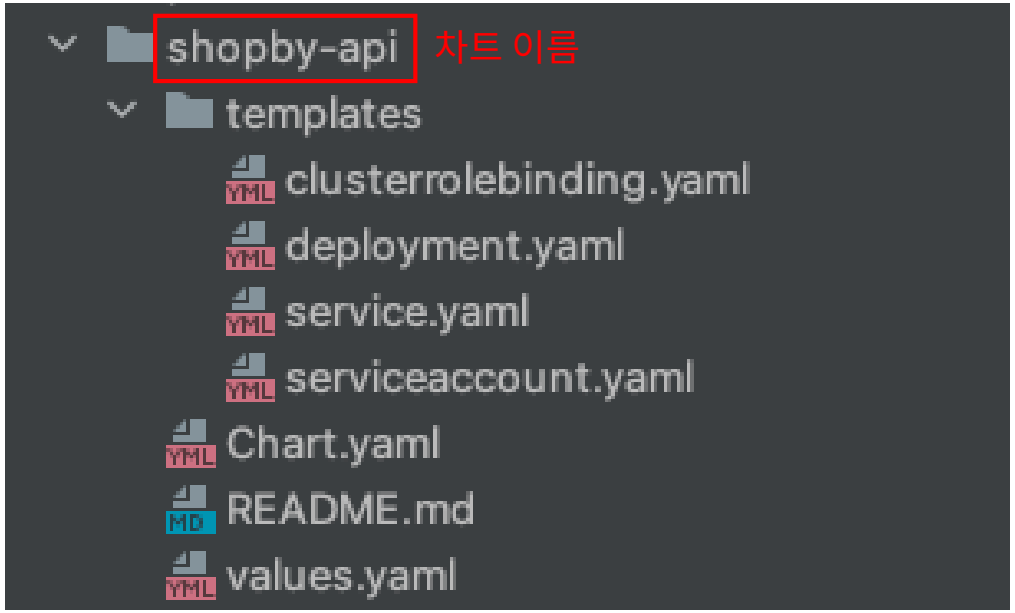
Helm으로 패키징!

- 쿠버네티스 리소스 패키지 도구
- Manifest 파일 묶음: template
- Template 변수 바인딩은 yaml파일로
- Chart로 관리

쿠버네티스에 배포하기

Helm으로 패키징!

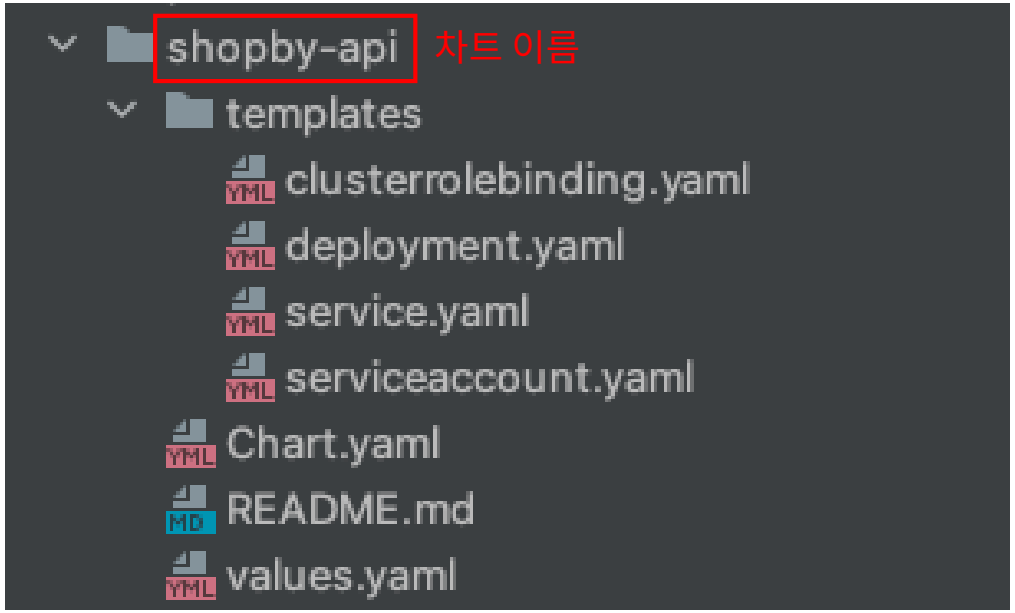
- 쿠버네티스 리소스 패키지 도구
- Manifest 파일 묶음: template
- Template 변수 바인딩은 yaml파일로
- Chart로 관리



쿠버네티스에 배포하기

Helm으로 패키징!

- 쿠버네티스 리소스 패키지 도구
- Manifest 파일 묶음: template
- Template 변수 바인딩은 yaml파일로
- Chart로 관리



프로퍼티 파일용 ConfigMap은 외부에서 로드
→ 배포와 별개 프로세스

쿠버네티스에 배포하기

NHN FORWARD

Helm Chart

service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.namespace }}-{{ .Valu
es.app.name }}
  namespace: {{ .Release.namespace }}
spec:
  type: ClusterIP
  selector:
    app: {{ .Values.app.name }}
  ports:
    - port: 80
      protocol: TCP
      targetPort: {{ .Values.app.port }}
```

clusterrolebinding.yaml

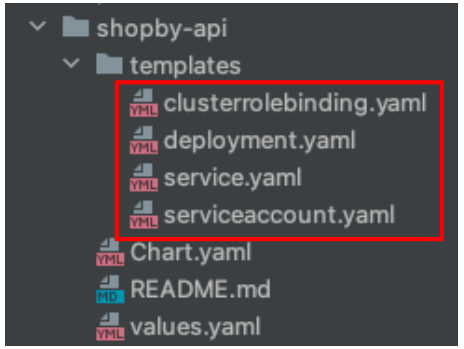
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: {{ .Release.namespace }}-{{ .Valu
es.app.name }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: spring-k8s
subjects:
  - kind: ServiceAccount
    name: {{ .Values.app.name }}
    namespace: {{ .Release.namespace }}
```

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: {{ .Values.app.name }}
  name: {{ .Values.app.name }}
  namespace: {{ .Release.namespace }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.app.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.app.name }}
    spec:
      containers:
        - name: {{ .Values.app.name }}
          image: {{ .Values.container.image.name }}
:{{ .Values.container.image.tag }}
          imagePullPolicy: {{ ternary "Always" "If
NotPresent" (eq .Values.container.image.tag "alpha
") }}
          restartPolicy: Always
          serviceAccount: {{ .Values.app.name }}
          ports:
            - containerPort: {{ .Values.app.port }}
              name: http
              protocol: TCP
```

serviceaccount.yaml

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: {{ .Values.app.name }}
  namespace: {{ .Release.namespace }}
```



order-internal.yaml (values.yaml)

```yaml
app:
  name: internal
  profile: kubernetes
  port: 8231
  pinpoint:
    enabled: true
    mainClassName: com.ncp.order.InternalApplicationKt
    samplingRate: 1
  heapdump:
    enabled: true
  options:
    metaspaceSize: "128m"
    maxMetaspaceSize: "192m"
    minRamPercentage: "60.0"
    maxRamPercentage: "60.0"
container:
  image:
    name: pivate-repo-url/order-internal
    tag: alpha
replicas: 1
resources:
  requests:
    cpu: "0.5"
    memory: "800Mi"
  limits:
    cpu: "0.5"
    memory: "800Mi"
```

쿠버네티스에 배포하기

NHN FORWARD

Helm Chart

service.yaml

svc

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.namespace }}-{{ .Valu
es.app.name }}
  namespace: {{ .Release.namespace }}
spec:
  type: ClusterIP
  selector:
    app: {{ .Values.app.name }}
  ports:
    - port: 80
      protocol: TCP
      targetPort: {{ .Values.app.port }}
```

clusterrolebinding.yaml

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: {{ .Release.namespace }}-{{ .Valu
es.app.name }}
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: spring-k8s
subjects:
  - kind: ServiceAccount
    name: {{ .Values.app.name }}
    namespace: {{ .Release.namespace }}
```

crb

deployment.yaml

deploy

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: {{ .Values.app.name }}
  name: {{ .Values.app.name }}
  namespace: {{ .Release.namespace }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.app.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.app.name }}
    spec:
      containers:
        - name: {{ .Values.app.name }}
          image: {{ .Values.container.image.name }}
:{{ .Values.container.image.tag }}
          imagePullPolicy: {{ ternary "Always" "If
NotPresent" (eq .Values.container.image.tag "alpha
") }}
          restartPolicy: Always
          serviceAccount: {{ .Values.app.name }}
          ports:
            - containerPort: {{ .Values.app.port }}
              name: http
              protocol: TCP
```

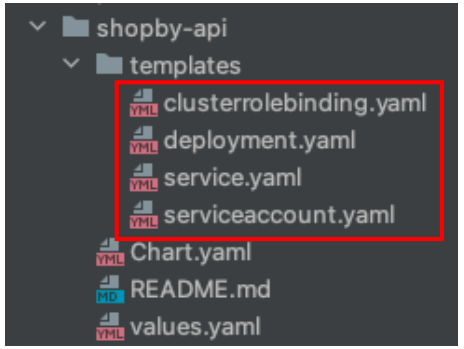
serviceaccount.yaml

sa

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: {{ .Values.app.name }}
  namespace: {{ .Release.namespace }}
```

배포!

```
$ helm install order-internal shopby-api -f order-internal.yaml -n order
$ helm install product-shop shopby-api -f product-shop.yaml -n order
$ helm install member-admin shopby-api -f member-admin.yaml -n order
```



order-internal.yaml (values.yaml)

```yaml
app:
  name: internal
  profile: kubernetes
  port: 8231
  pinpoint:
    enabled: true
    mainClassName: com.ncp.order.InternalApplicationKt
    samplingRate: 1
  heapdump:
    enabled: true
  options:
    metaspaceSize: "128m"
    maxMetaspaceSize: "192m"
    minRamPercentage: "60.0"
    maxRamPercentage: "60.0"
container:
  image:
    name: pivate-repo-url/order-internal
    tag: alpha
replicas: 1
resources:
  requests:
    cpu: "0.5"
    memory: "800Mi"
  limits:
    cpu: "0.5"
    memory: "800Mi"
```

쿠버네티스에 배포하기

Helm Chart

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: {{ .Values.app.name }}
  name: {{ .Values.app.name }}
  namespace: {{ .Release.namespace }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.app.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.app.name }}
    spec:
      containers:
        - name: {{ .Values.app.name }}
          image: {{ .Values.container.image.name }}:{{ .Values.container.image.tag }}
          imagePullPolicy: {{ ternary "Always" "IfNotPresent" (eq .Values.container.image.tag "alpha") }}
          restartPolicy: Always
          serviceAccount: {{ .Values.app.name }}
          ports:
            - containerPort: {{ .Values.app.port }}
              name: http
              protocol: TCP
```

변경 사항

```yaml
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
          topologyKey: kubernetes.io/hostname
          matchExpressions:
            - key: app
              operator: In
              values:
                - internal
```

쿠버네티스에 배포하기

Helm Chart

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: {{ .Values.app.name }}
  name: {{ .Values.app.name }}
  namespace: {{ .Release.namespace }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.app.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.app.name }}
    spec:
      containers:
        affinity:
          podAntiAffinity:
            requiredDuringSchedulingIgnoredDuringExecution:
              - labelSelector:
                topologyKey: kubernetes.io/hostname
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - {{ .Values.app.name }}
        - name: {{ .Values.app.name }}
          image: {{ .Values.container.image.name }}:{{ .Values.container.image.tag }}
          imagePullPolicy: {{ ternary "Always" "IfNotPresent" (eq .Values.container.image.tag "alpha") }}
          restartPolicy: Always
          serviceAccount: {{ .Values.app.name }}
          ports:
            - containerPort: {{ .Values.app.port }}
              name: http
              protocol: TCP
```

Chart에 추가

변경 사항

```yaml
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
          topologyKey: kubernetes.io/hostname
          matchExpressions:
            - key: app
              operator: In
              values:
                - internal
```

쿠버네티스에 배포하기

Helm Chart

deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
  labels:
    app: {{ .Values.app.name }}
  name: {{ .Values.app.name }}
  namespace: {{ .Release.namespace }}
spec:
  replicas: {{ .Values.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.app.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.app.name }}
    spec:
      containers:
        affinity:
          podAntiAffinity:
            requiredDuringSchedulingIgnoredDuringExecution:
              - labelSelector:
                topologyKey: kubernetes.io/hostname
                matchExpressions:
                  - key: app
                    operator: In
                    values:
                      - {{ .Values.app.name }}
        - name: {{ .Values.app.name }}
          image: {{ .Values.container.image.name }}:{{ .Values.container.image.tag }}
          imagePullPolicy: {{ ternary "Always" "IfNotPresent" (eq .Values.container.image.tag "alpha") }}
          restartPolicy: Always
          serviceAccount: {{ .Values.app.name }}
          ports:
            - containerPort: {{ .Values.app.port }}
              name: http
              protocol: TCP
```

Chart에 추가

변경 사항

```yaml
spec:
  affinity:
    podAntiAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
          topologyKey: kubernetes.io/hostname
          matchExpressions:
            - key: app
              operator: In
              values:
                - internal
```

반영

```
$ helm upgrade -i order-internal shopby-api -f order-internal.yaml -n order
$ helm upgrade -i product-shop shopby-api -f product-shop.yaml -n order
$ helm upgrade -i member-admin shopby-api -f member-admin.yaml -n order
```

쿠버네티스에 배포하기

프로세스화

Developer

배포 유발

Jenkins

nexus repository

Docker Repository

Github Repository

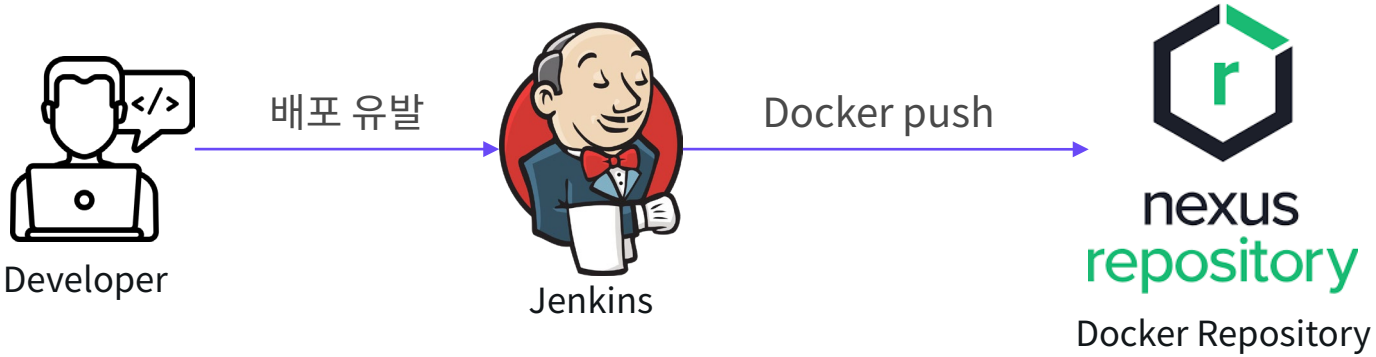
- values.yaml (Helm Chart)

ArgoCD

Kubernetes

쿠버네티스에 배포하기

프로세스화



Github Repository

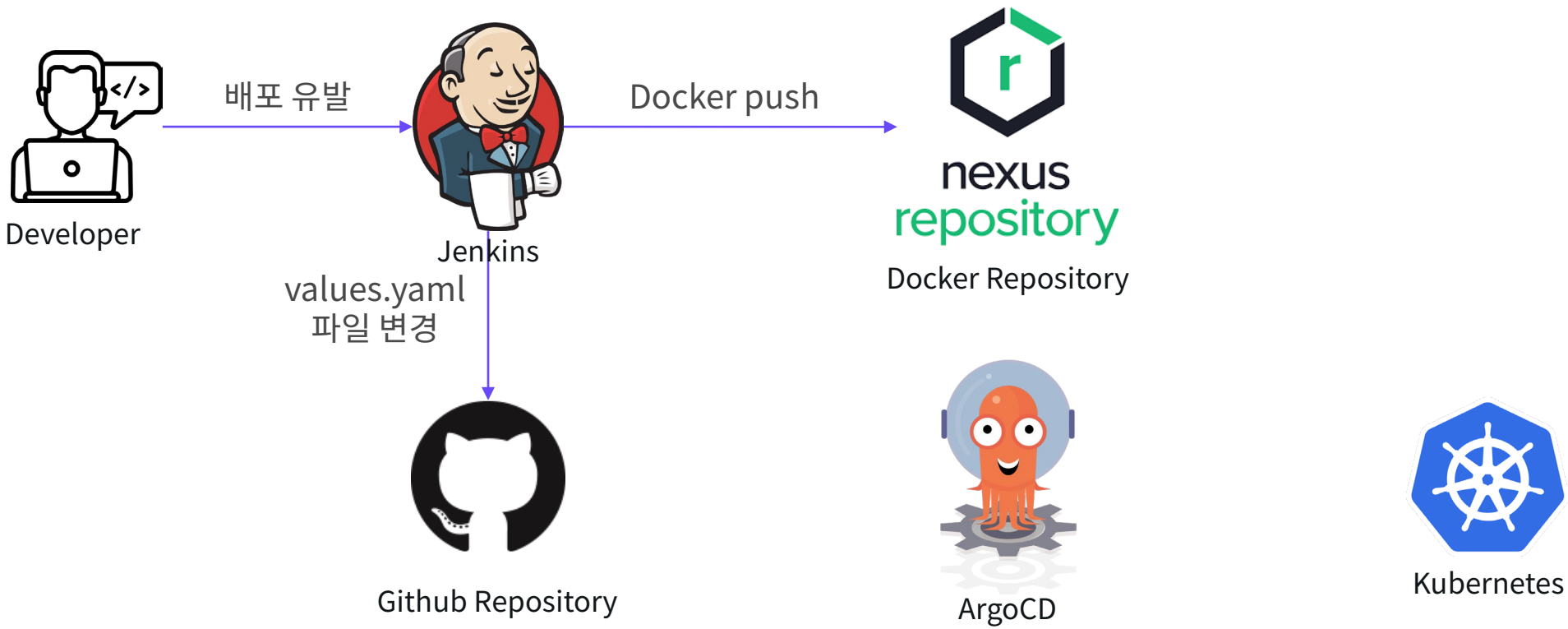
- values.yaml (Helm Chart)

ArgoCD

Kubernetes

쿠버네티스에 배포하기

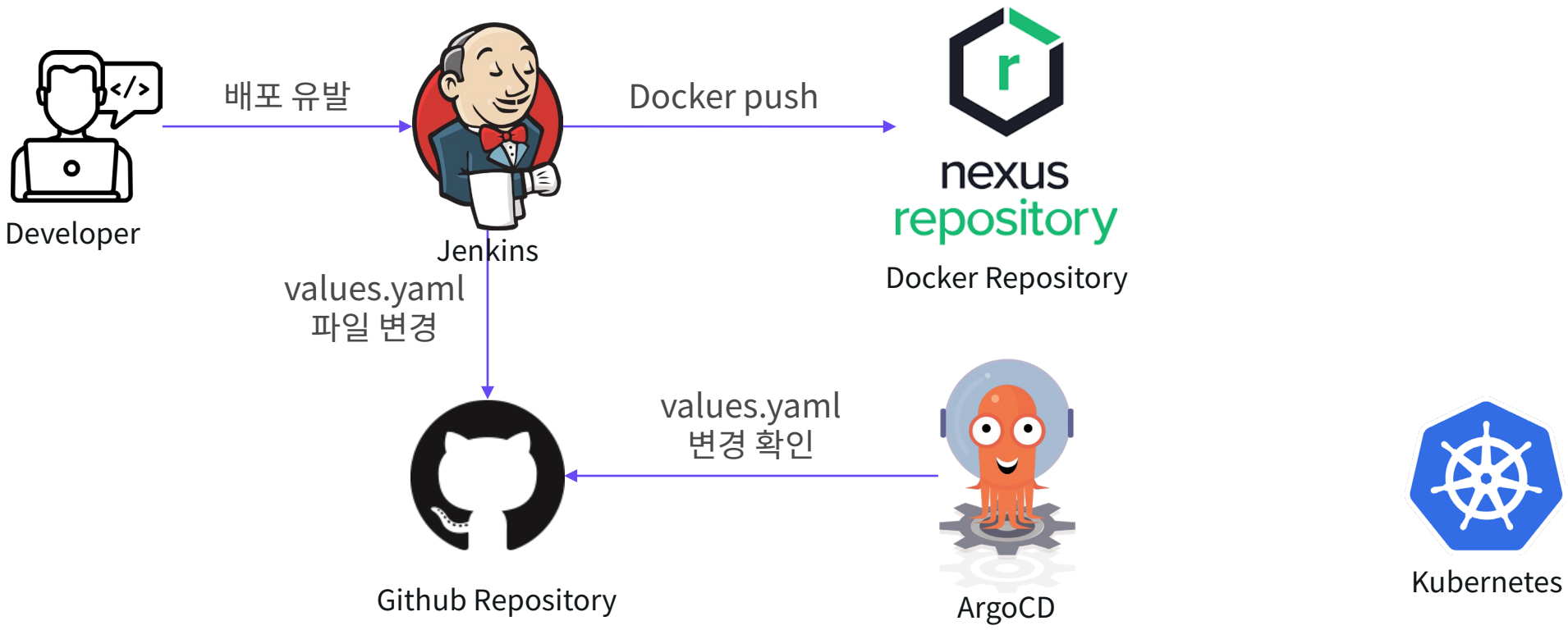
프로세스화



- values.yaml (Helm Chart)

쿠버네티스에 배포하기

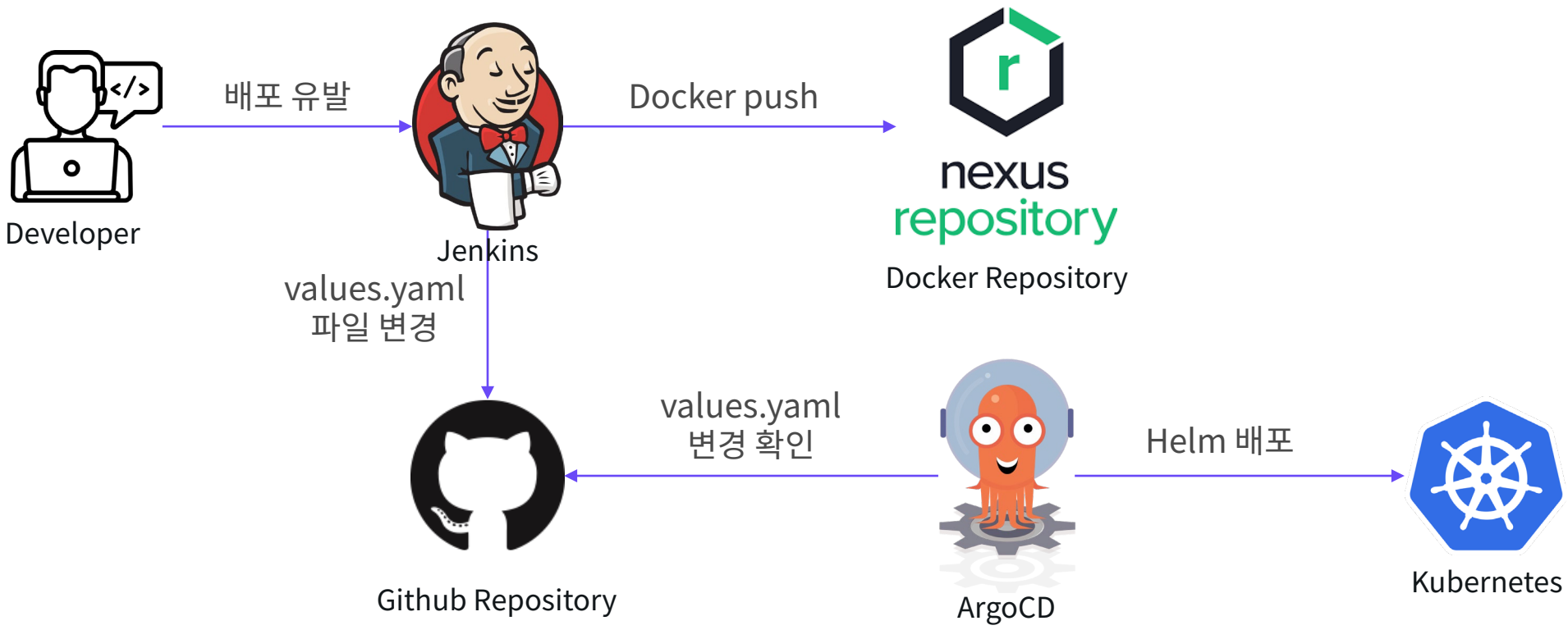
프로세스화



- values.yaml (Helm Chart)

쿠버네티스에 배포하기

프로세스화

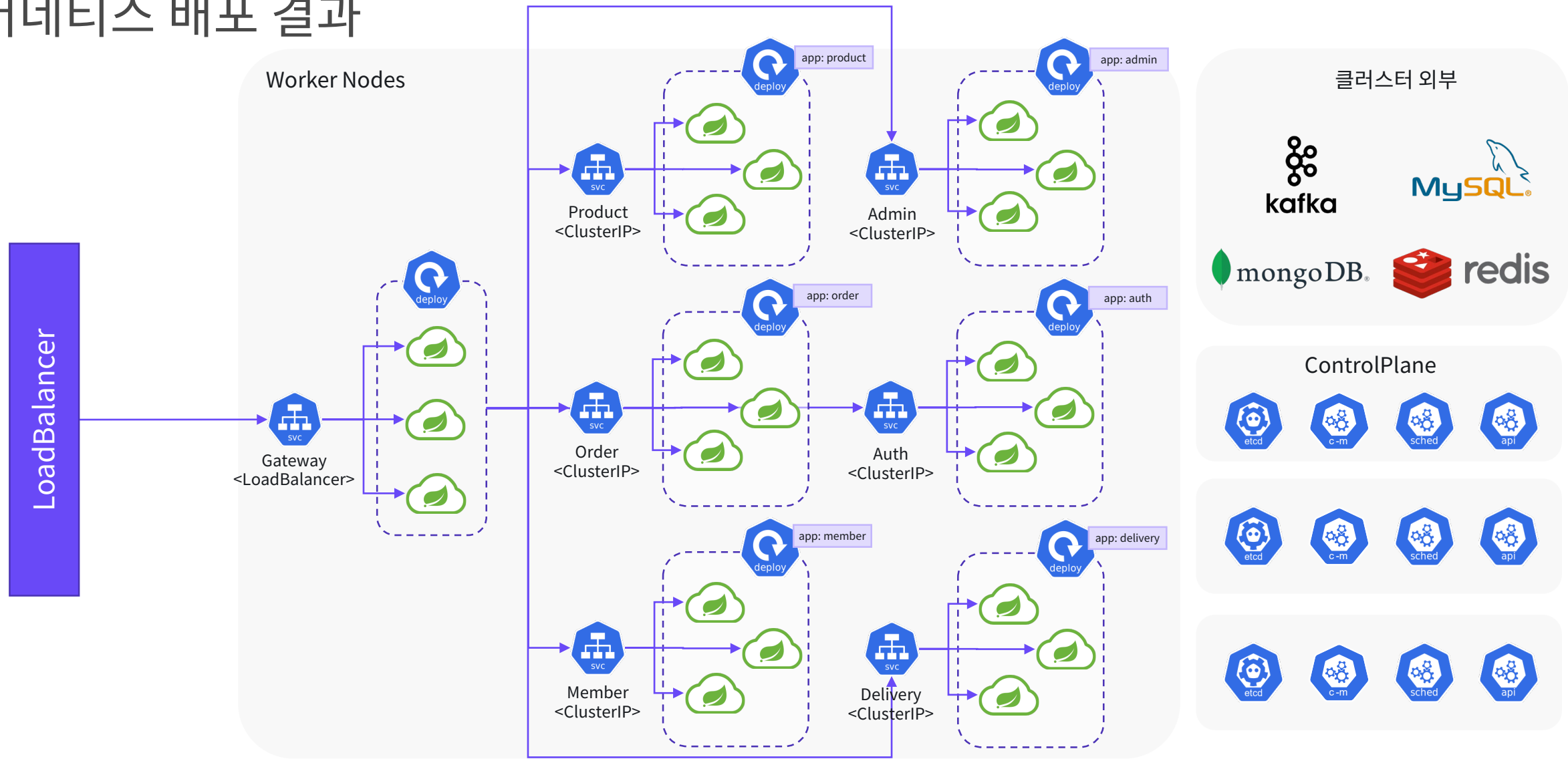


- values.yaml (Helm Chart)

쿠버네티스에 배포하기

NHN FORWARD

쿠버네티스 배포 결과



쿠버네티스로 전환하기

쿠버네티스로 전환하기

트래픽 전환 전략

- 8,000여 개의 쇼핑몰 운영 중
- 중단하면 손실이 큼
- 무조건 무중단으로…

쿠버네티스로 전환하기

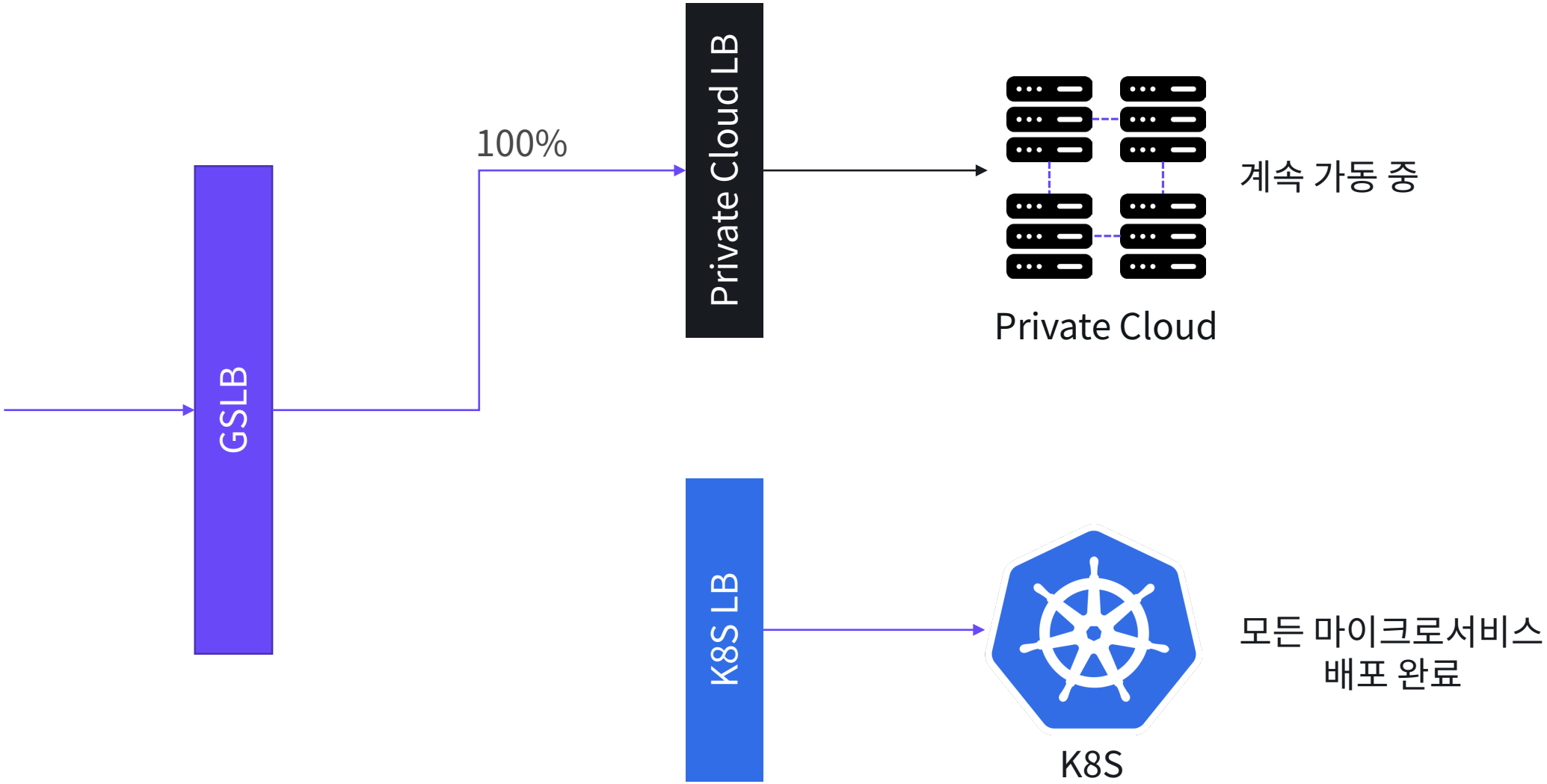
트래픽 전환 전략

- 8,000여 개의 쇼핑몰 운영 중
- 중단하면 손실이 큼
- 무조건 무중단으로…

- GSLB
- 먼저 쿠버네티스에 모든 마이크로서비스 배포
- 전환기에는 두 환경 동시에 구동
- 리소스 비용이 많이 들긴 하지만 안전이 우선!

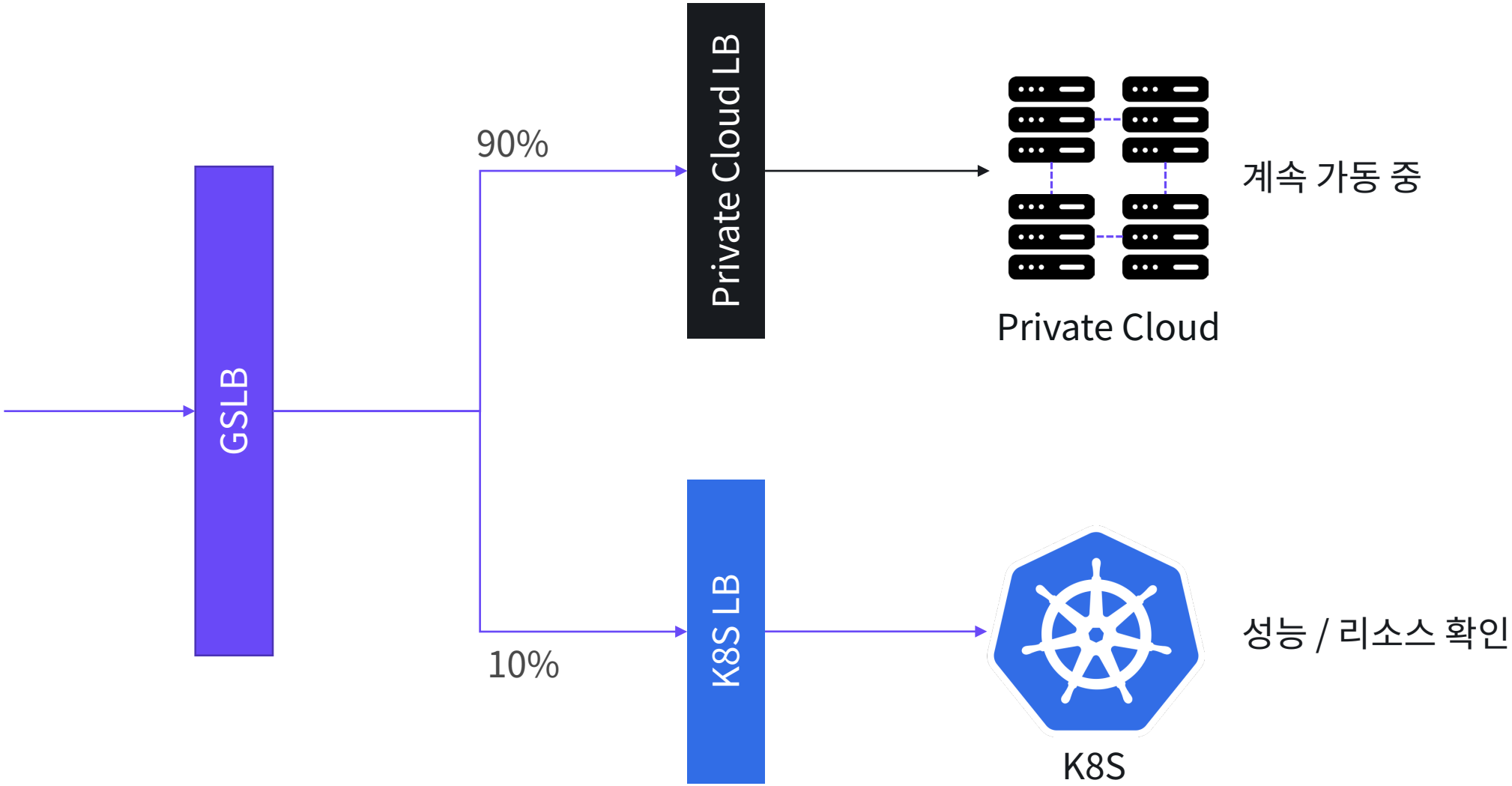
쿠버네티스로 전환하기

전략



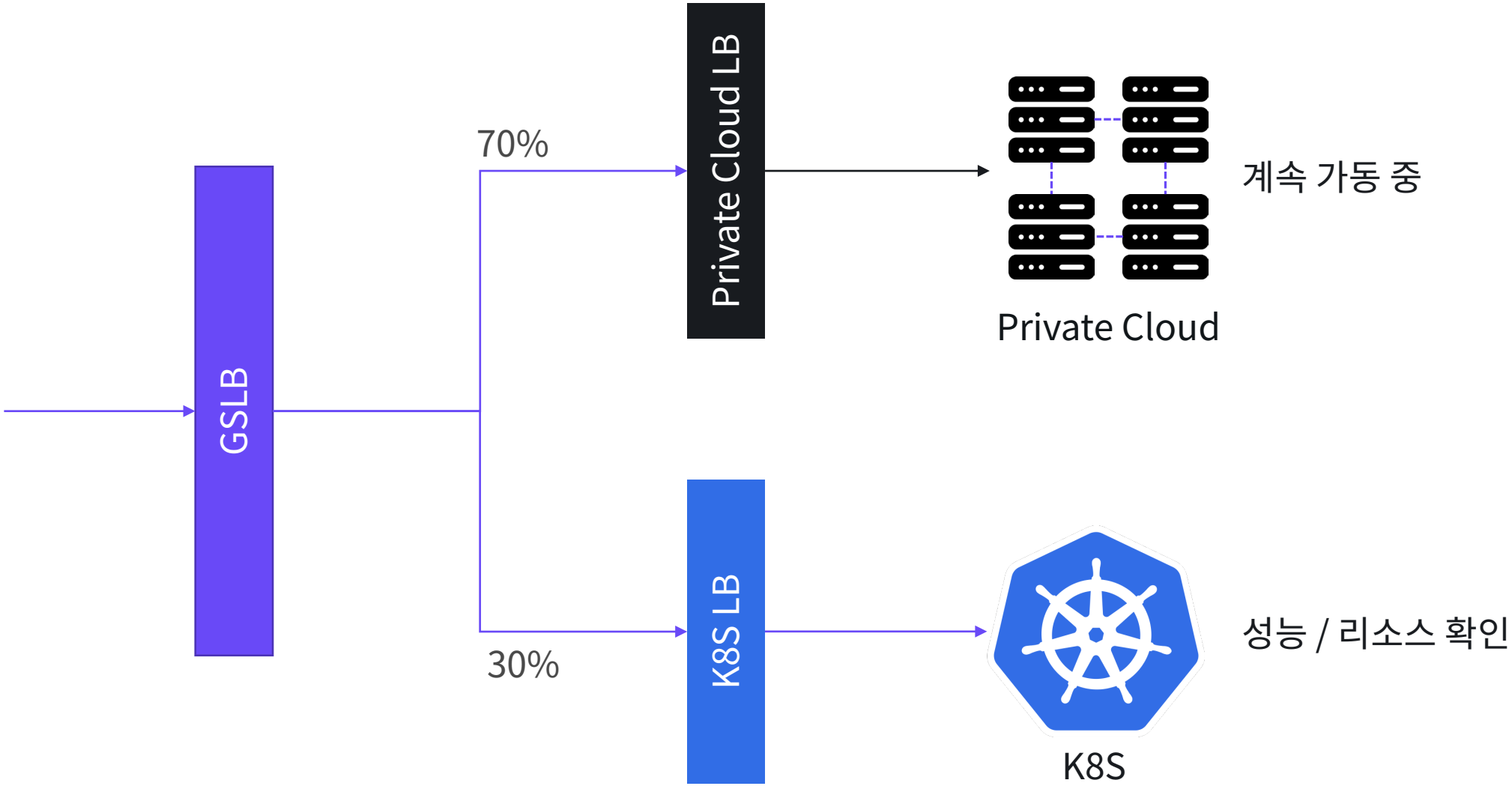
쿠버네티스로 전환하기

전략



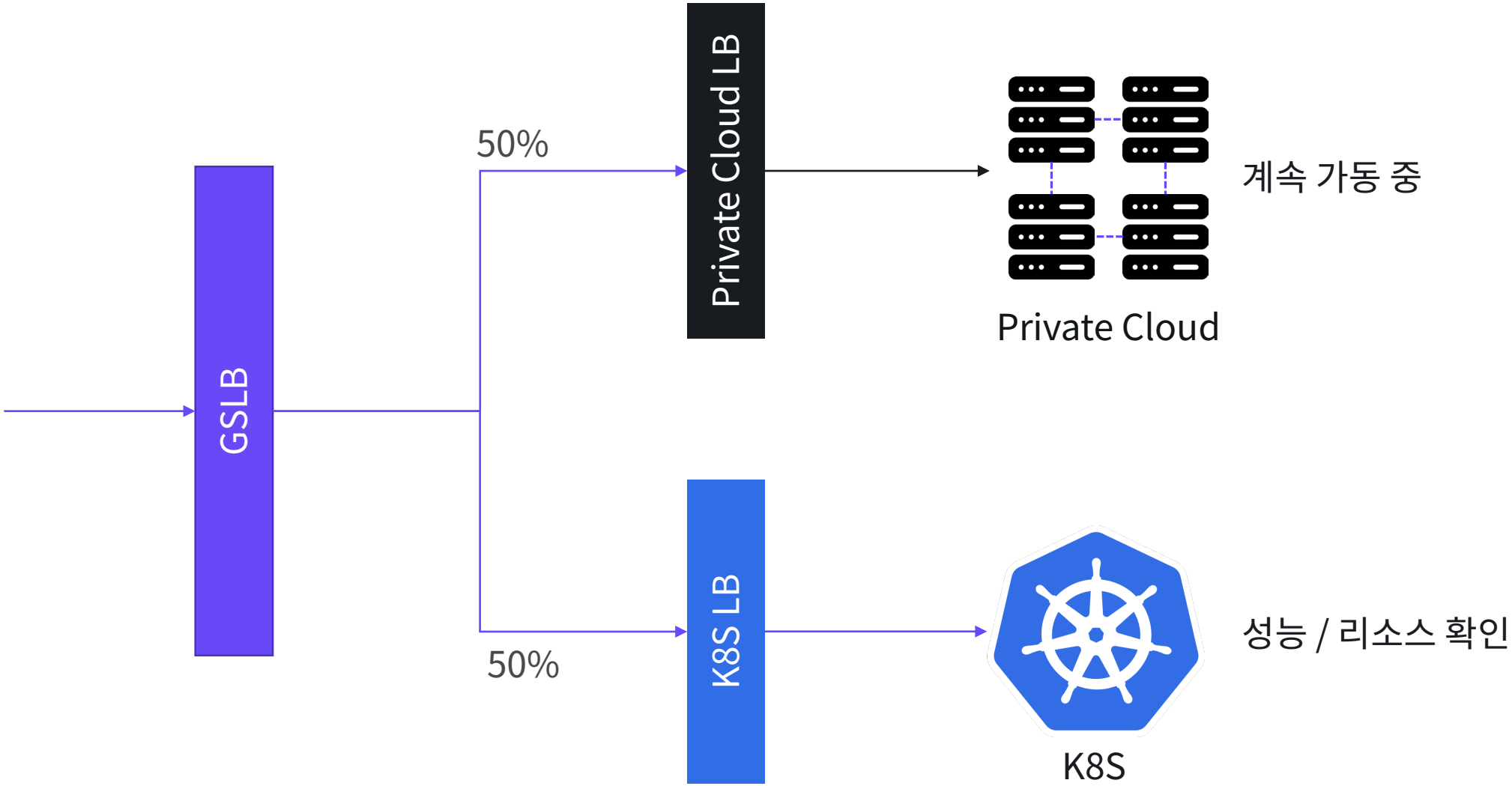
쿠버네티스로 전환하기

전략



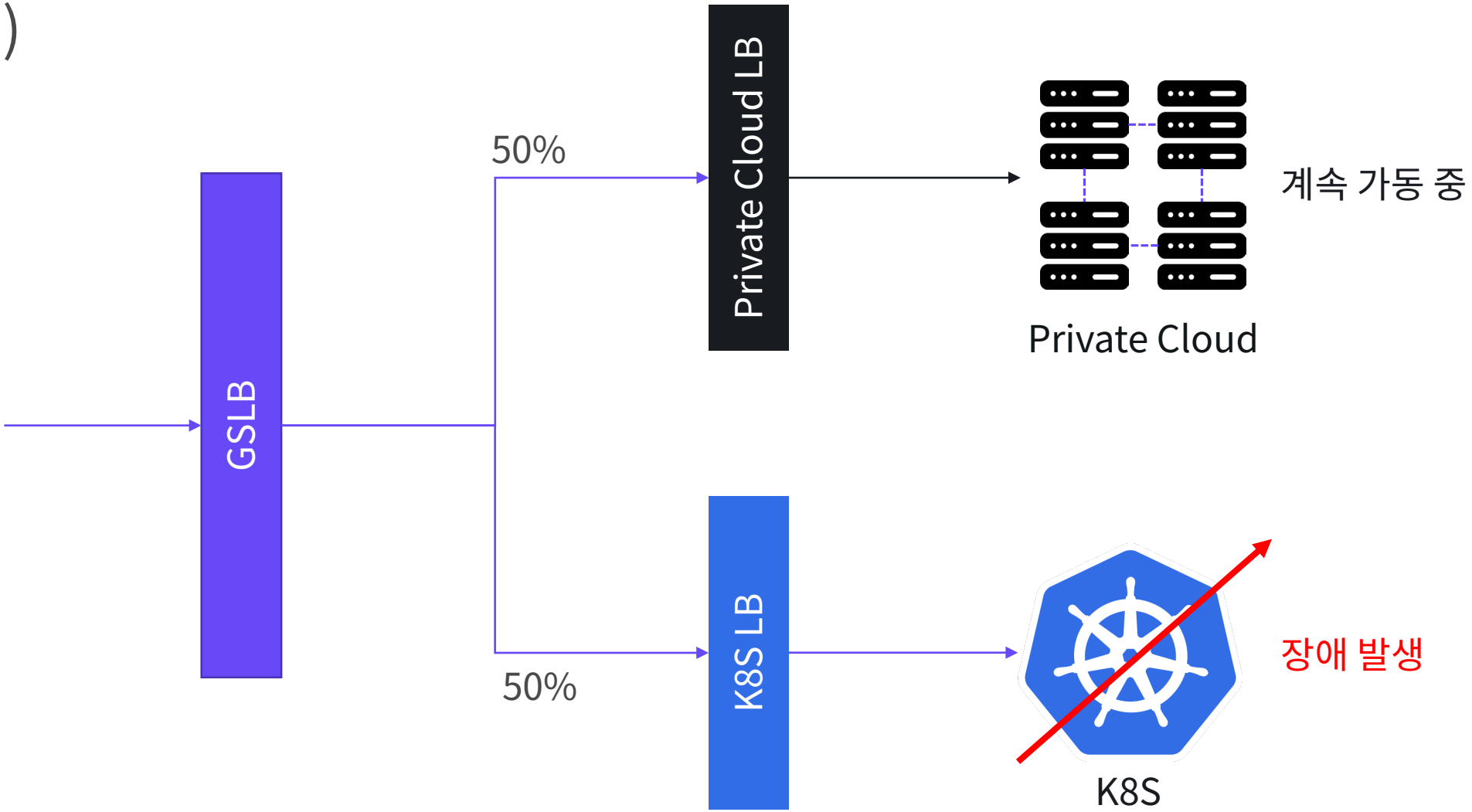
쿠버네티스로 전환하기

전략



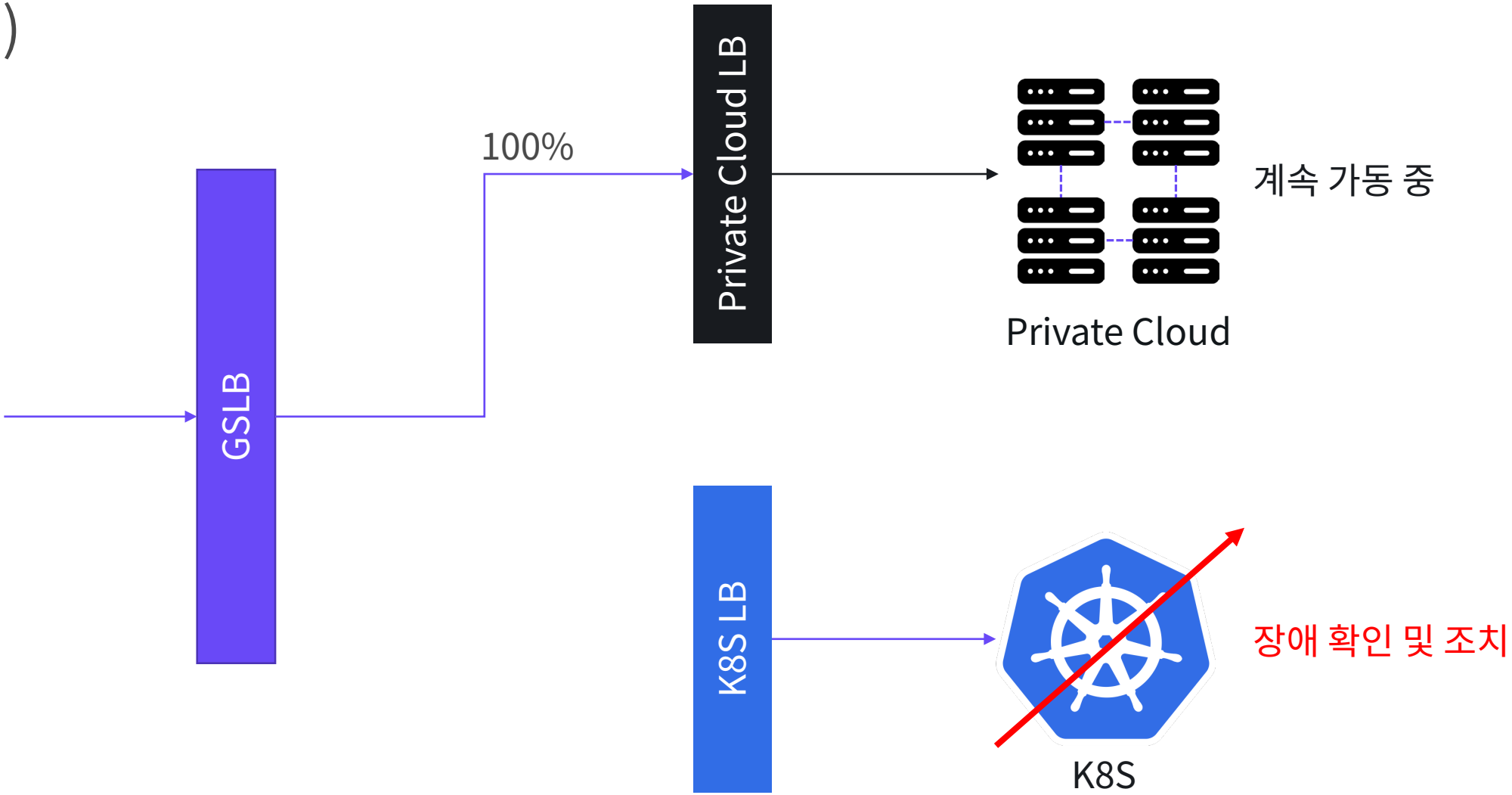
쿠버네티스로 전환하기

전략(롤백)



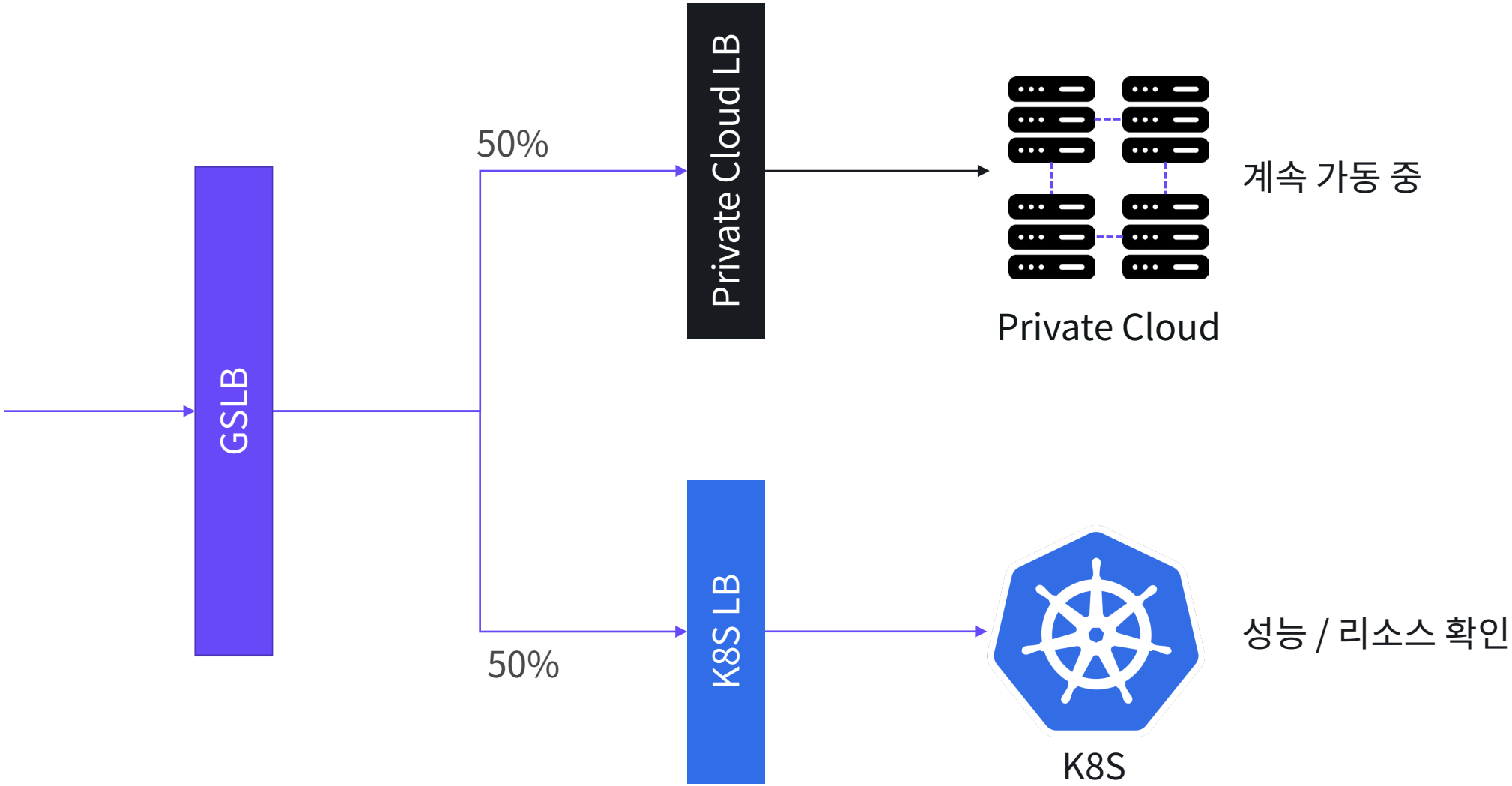
쿠버네티스로 전환하기

전략(롤백)



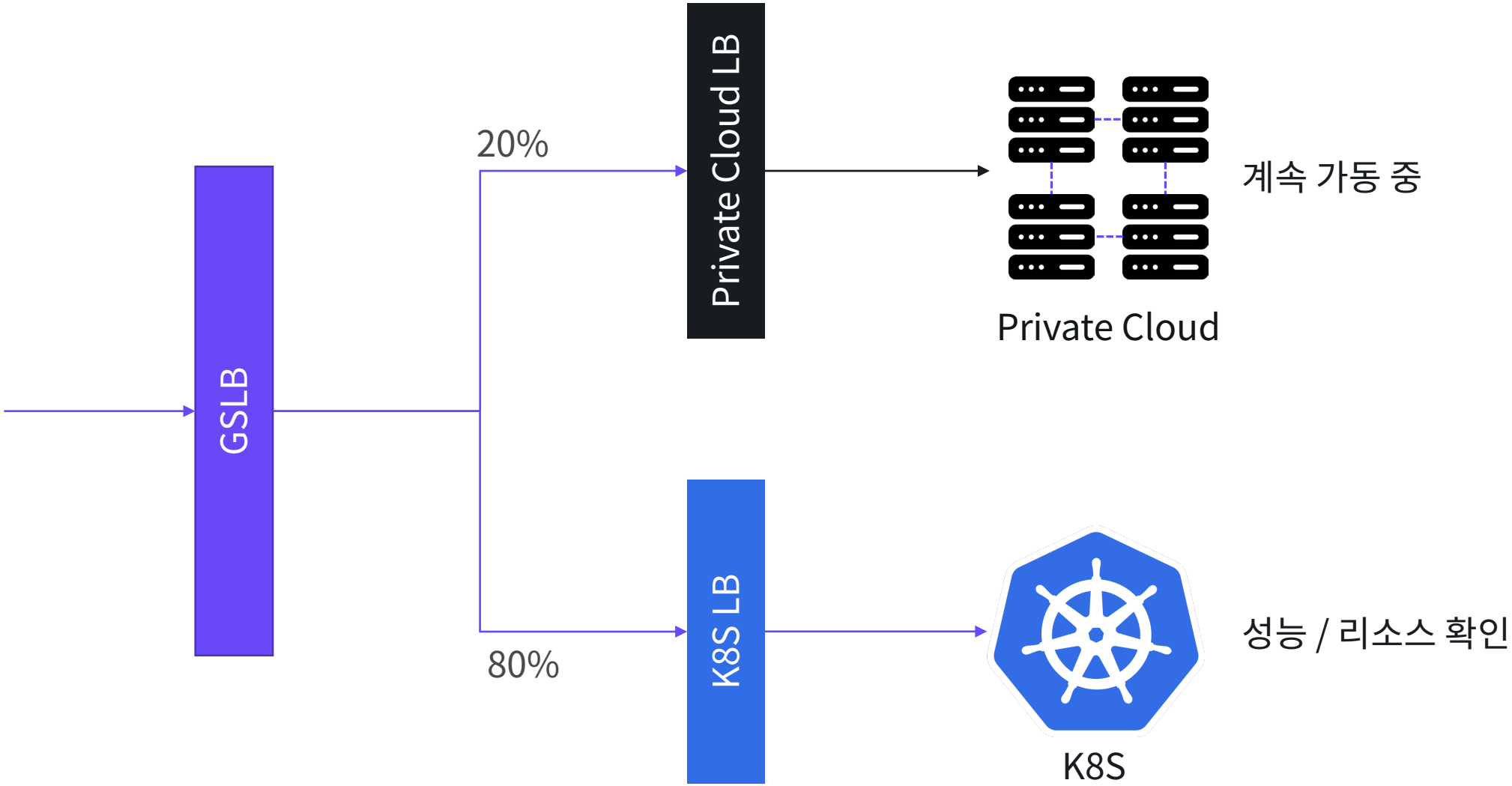
쿠버네티스로 전환하기

전략



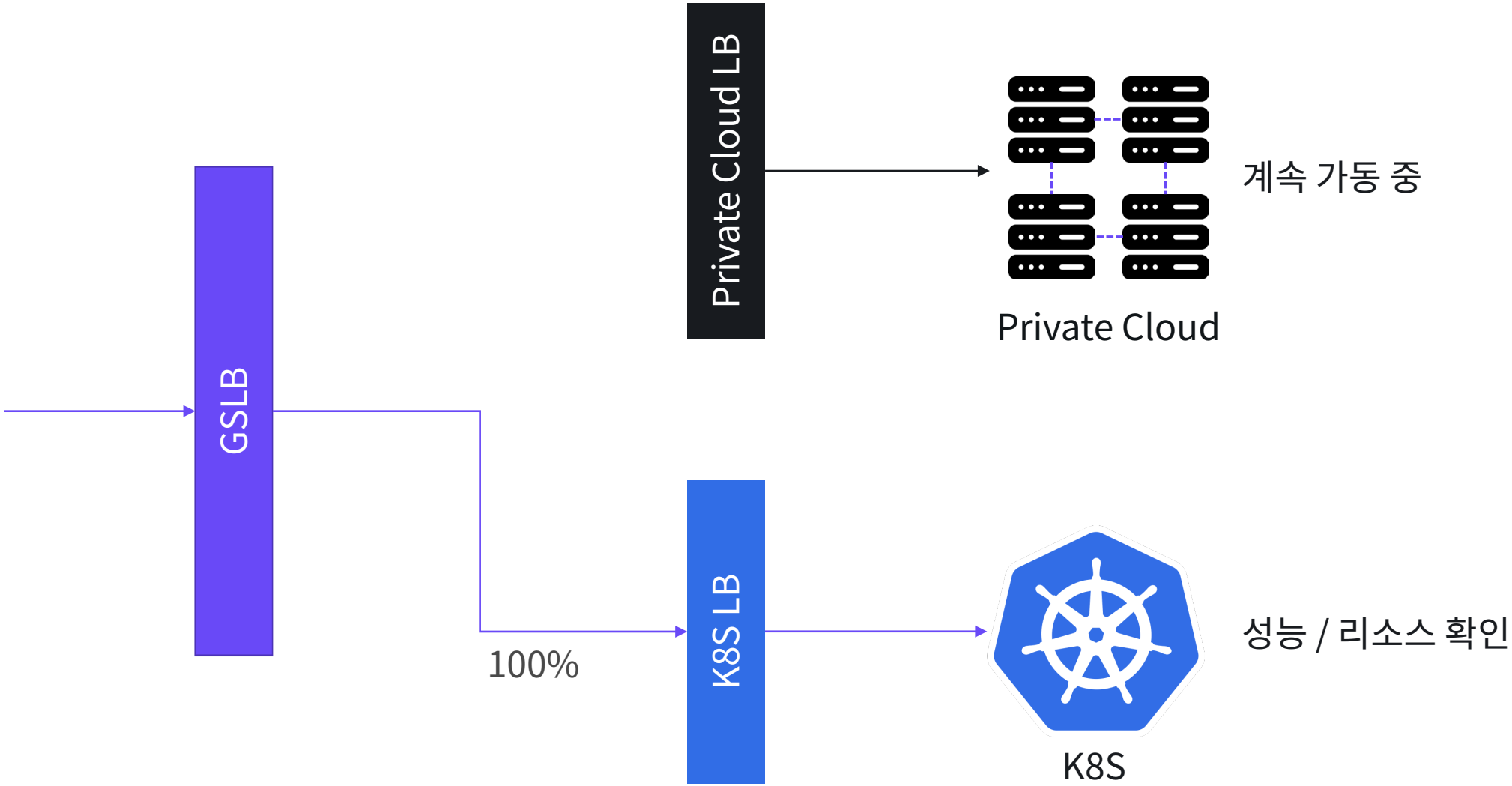
쿠버네티스로 전환하기

전략



쿠버네티스로 전환하기

전략



쿠버네티스로 전환하기

마무리 및 회고

- 작년 10월부터 쿠버네티스 시작
- 3 + 1 DevOps팀
- 신규 서비스들은 쿠버네티스로 배포 및 운영 중
- 샵바이는 베타 환경까지 운영

쿠버네티스로 전환하기

마무리 및 회고

- 로컬에서도 대부분의 작업이 가능하다.
- 셀프힐링 (선 조치 후 보고)
- 오픈소스에서 쿠버네티스로의 배포 지원

- 롤링 배포 이외에도 다른 배포 방식 적용 계획 중
- 멀티 클러스터 멀티 리전 운영 계획 중

Q & A

NHN FORWARD

고맙습니다.