

# Ingress Annotator 개발기

김정민 / 삼성SDS

Cloud Native Korea  
Community Day 2025



# 발표자



김정민 / 삼성SDS / Cloud Engineer

Samsung Kubernetes Engine을 개발하는 업무를 하고 있습니다. K8s와 Go 테스트 관련 주제에 관심이 많습니다.

발표자료 <https://github.com/jmnote/slides>



## 지난 발표

<https://github.com/jmnote/slides>

### K8s LMA 'venti-stack' 개발기

김정민 - 삼성SDS

Cloud Native Korea  
Community Day 2024



'24.09

Cloud Native Korea Community Day 2024

### Helm Chart 개발과 GitHub · ArtifactHub 사용기

김정민 삼성SDS

KCD Seoul 2025



'25.05

KCD Seoul 2025

# 목차

I . 소개

II . 설계 논의

III . 관련 개념 정리

IV . 마무리

# I 소개

# Ingress Annotator란?



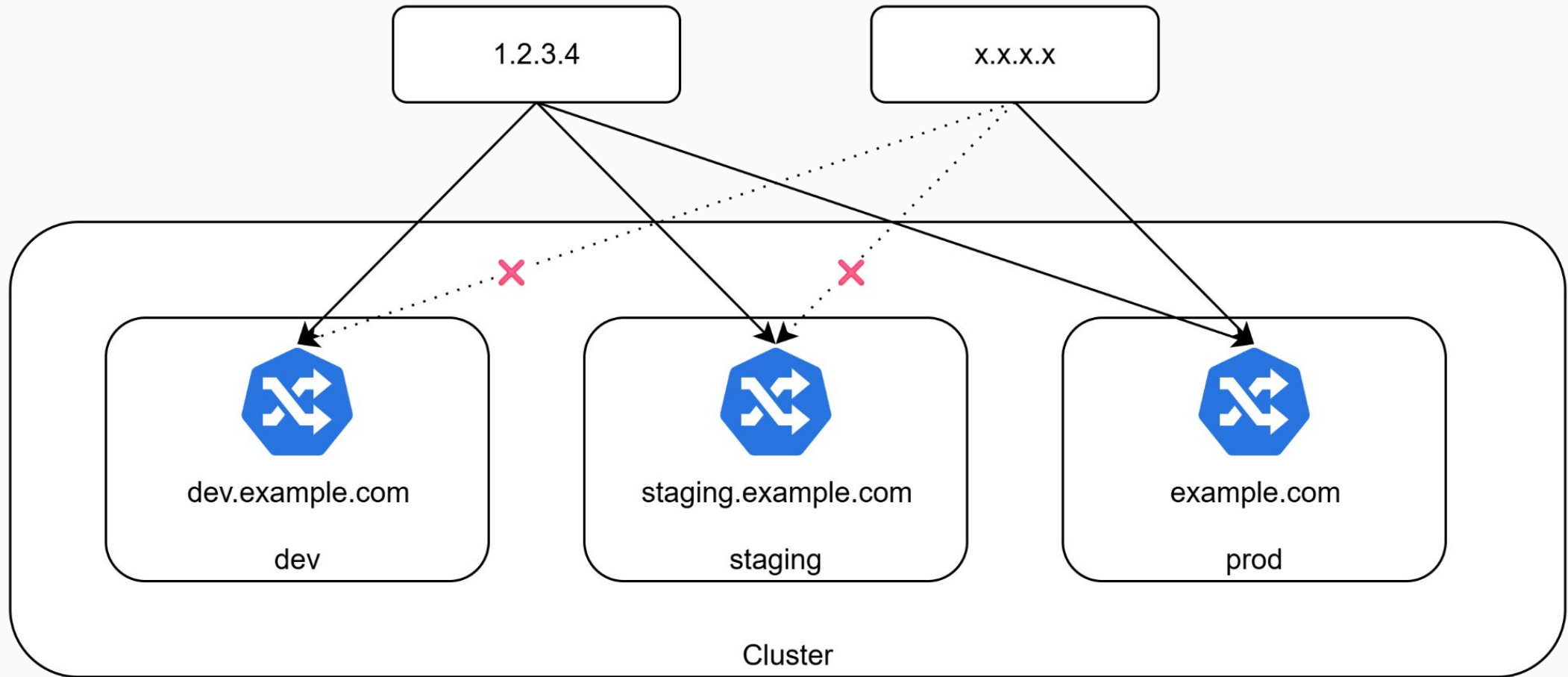
**Ingress**    클러스터 외부 트래픽을 내부 서비스로 라우팅하는 리소스

**Annotator**    어떤 리소스에 annotation을 붙이는 도구

**Ingress Annotator**    Ingress에 annotation을 붙이는 도구

표준 아님... ✨

# 사례1: IP접근제한#1 “dev, staging 네임스페이스는 특정 IP만 허용하고 싶다”



# 사례1: IP접근제한#2

“dev, staging 네임스페이스는  
특정 IP만 허용하고 싶다”

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: prod-ingress
  namespace: prod
spec:
  rules:
  - host: example.com
    ...
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dev-ingress
  namespace: dev
  annotations:
    nginx.ingress.kubernetes.io/whitelist-source-range: "1.2.3.4"
spec:
  rules:
  - host: dev.example.com
    ...
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: staging-ingress
  namespace: staging
  annotations:
    nginx.ingress.kubernetes.io/whitelist-source-range: "1.2.3.4"
spec:
  rules:
  - host: staging.example.com
    ...
```



# 사례1: IP접근제한#3 “dev, staging 네임스페이스는 특정 IP만 허용하고 싶다”

IP 추가하려면? 네임스페이스 추가하려면? ingress 추가하려면?

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress1 # ingress2, ingress3, ...
  namespace: dev1 # dev2, dev3, ...
  annotations:
    nginx.ingress.kubernetes.io/whitelist-source-range: "1.2.3.4,50.60.70.80,111.222.0.0/16,..."
spec:
  rules:
  - host: dev.example.com
    ...
```

# vs 방화벽

이런 게 왜 필요? 방화벽이 있는데...

```
nginx.ingress.kubernetes.io/whitelist-source-range: "1.2.3.4"
```

구분	방화벽	Ingress Annotation (ip-allowlist)
계층	네트워크 (L3/L4) IP:Port 기준	애플리케이션 (L7) host(domain)/path 기준
대상	Ingress Controller (연계된 Ingress 전체)	Ingress 단위
응답	응답 안함	HTTP 403 Forbidden 응답 404 아님
특징	강력함	간편함 클러스터 내부에서 간단한 설정으로 가능
공통점	출발지 IP에 따라 통신 제한	

→ 의의: 애플리케이션 보안 계층 제공 (여러 ingress controller/middleware/framework 사례)

→ 환경(예: 엔터프라이즈)에 따라 용도는 제한적일 수 있음

→ Ingress Annotator는 특정 annotation과 직접 관련은 없고, 다양한 annotation 관리 가능

# 사례2: OAuth

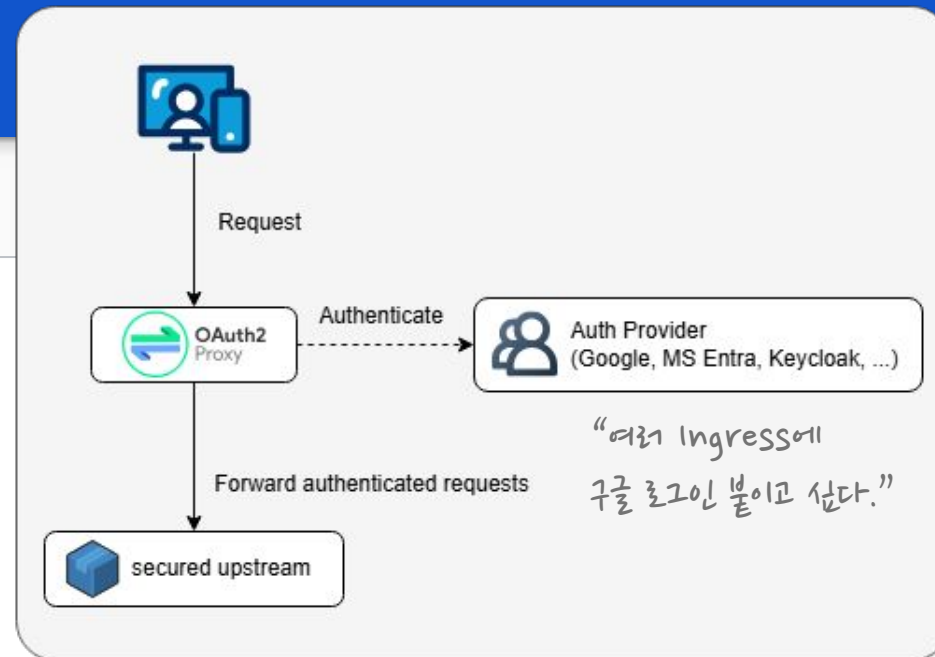
“dev/dev-secured 인그레스에  
oauth2 Proxy 적용”

<https://oauth2-proxy.github.io/oauth2-proxy/>

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: dev-secured
  namespace: dev
  annotations:
    nginx.ingress.kubernetes.io/auth-url: >-
      https://oauth.example.com/oauth2/auth 로그인 확인
    nginx.ingress.kubernetes.io/auth-signin: >-
      https://oauth.example.com/oauth2/start?rd=https://$host$escaped_request_uri
      (안된 경우) 로그인 시작
...

```

OAuth2 Proxy as a standalone reverse-proxy



```
$ kubectl get ing -A
```

NAMESPACE	NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
oauth2-proxy	oauth2-proxy	nginx	oauth.example.com	10.20.30.40	80, 443	99d
dev	dev-ingress	nginx	dev.example.com	10.20.30.40	80, 443	99d
...						

# 일반화 #1

“어떤 조건에 맞는 Ingress에 어떤 Annotation을 붙이고 싶다.” → 정책

- **description:** IP allowlist for dev, staging

dev, staging 네임스페이스는

**selector:**

특정 IP만 허용

**include:** dev, staging

**annotations:**

**nginx.ingress.kubernetes.io/whitelist-source-range:** "1.2.3.4,50.60.70.80,111.222.0.0/16"

- **description:** OAuth2 Proxy for \*-secured

\*-secured 인그레스에

**selector:**

OAuth2 Proxy 적용

**include:** \*/\*-secured

**annotations:**

**nginx.ingress.kubernetes.io/auth-url:** "https://oauth.example.com/oauth2/auth"

**nginx.ingress.kubernetes.io/auth-signin:** "https://oauth.example.com/oauth2/start..."

“정책 = 규칙의 집합”이라고 하자.

## 일반화 #2 “어떤 조건에 맞는 Ingress에 어떤 Annotation을 붙이고 싶다.” → 정책

```
- description: Limit download speed
  selector:
    include: api/download-*
  annotations:
    nginx.ingress.kubernetes.io/limit-rate: "100k"
```

api/download-\* 인그레스에  
속도 제한 100k/s

```
- description: Enforce HTTPS
  selector:
    exclude: kube-system,default
  annotations:
    nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
```

kube-system, default 외  
모든 네임스페이스에  
https 적용

어떤 annotation이든 처리 가능 → 모든 ingress controller와 호환

# listAnnotations 기능

표현의 기능

- description: IP allowlist

selector:

include: dev

annotations:

nginx.ingress.kubernetes.io/whitelist-source-range: "1.2.3.4,50.60.70.80,111.222.0.0/16"

annotation의 값은 문자열  
→ 여러 건은 보통 comma로 구분

- description: IP allowlist

selector:

include: dev

listAnnotations:

nginx.ingress.kubernetes.io/whitelist-source-range:

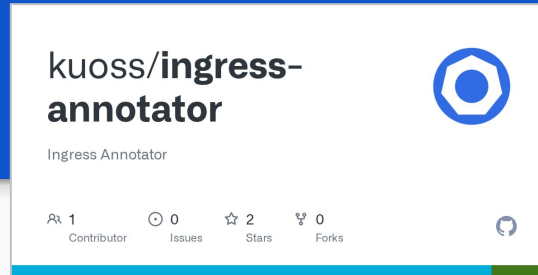
- 1.2.3.4

- 50.60.70.80

- 111.222.0.0/16

listAnnotations에는 리스트 작성 가능  
→ comma 구분 문자열로 바뀌어 적용됨

# 설치 방법



kubernetes OSS

## 방법1. Manifest 설치

```
kubectl apply -f
https://raw.githubusercontent.com/kuoss/ingress-annotator/
refs/heads/main/deploy/ingress-annotator.yaml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ingress-annotator
  namespace: ingress-annotator
data:
  rules: |
    - description: ssl-redirect
      selector:
        include: "default"
      annotations:
        nginx.ingress.kubernetes.io/ssl-redirect: "true"
    - ...
```

원하는 규칙 작성하여 apply

## 방법2. Helm Chart 설치

ArtifactHub 검색

# II 설계 논의



# Selector “어떤 조건에 맞는지” → ingress 선택

```
- description: ... for dev, staging  
  selector:  
    include: dev, staging  
  annotations:  
    ...
```

규칙

## 특징

- namespace 및 ingress 단위
- wildcard(\*) 지원
- 제외도 가능

## 예시

### # 포함

```
include: "*"           # 모든 네임스페이스  
include: "dev, staging" # dev, staging 네임스페이스  
include: "*/*-secured" # 모든 네임스페이스의 *-secured
```

### # 제외


```
exclude: "prod"        # prod 네임스페이스는 제외  
exclude: "prod*"       # prod* 네임스페이스는 제외  
exclude: "*/*-secured" # 모든 네임스페이스의 *-secured는 제외
```

# Selector 설계

# ingress 셀렉터 예시

include: "dev,staging" # dev, staging 네임스페이스

include: "/\*-secured" # 모든 네임스페이스의 \*-secured

항목	와일드카드(*) 	정규표현식	Label 셀렉터	CEL 셀렉터
표현력	중간	높음	제한적 <code>In/NotIn/Exists/DoesNotExist</code>	매우 높음 문자열·숫자·논리식 지원
가독성	높음	낮음	높음 명시적 키/값	중간 표현식 문법 필요
학습비용	낮음	높음	중간 K8S 표준	높음 새 언어 학습 필요
예시1	dev,staging	<code>^(dev staging)\$</code>	<pre>namespaceSelector:   matchExpressions:     - key: kubernetes.io/metadata.name       operator: In       values: ["dev","staging"]</pre>	<code>namespace in ["dev","staging"]</code>
예시2	<code>/*-secured"</code>	<code>^.*/*-secured\$</code>	불가	<code>ingressName.endsWith("-secured")</code>

※ 가독성 높고 쉽게 “[namespace]/[ingress]” 형식 선택


→ 주로 namespace 단위로 사용할 것 같아서 “[ingress]”는 생략 가능하도록 함


# Selector 사례

Label Selector가 대표적이지만, 다른 사례도 있음

유형	주요 필드	대상 및 활용	관련 리소스/도구
Label Selector	.selector .podSelector .labelSelector	Pod 선택/집합 지정, 정책 적용, 분산 배치 조건	Service, Deployment, StatefulSet, NetworkPolicy, PodDisruptionBudget, TopologySpreadConstraints
	.namespaceSelector	Namespace 선택, 정책 적용 범위 지정	NetworkPolicy
	.nodeSelector	Node 선택, Pod 배치 제어	PodSpec, Deployment, StatefulSet
CEL Selector	selectors[] .cel.expression	디바이스 조건 지정 (속성·용량 등) <small>노드에 연결된 GPU/NIC 등 리소스 세밀하게 선별</small>	ResourceClaimTemplate (DRA)
Field Selector	--field-selector	리소스 필드 기반 조회·필터링 (kubectl)	kubectl get, watch

# 구현체 비교 #1

 **ArtifactHUB**

[DOCS](#) [STATS](#) [SIGN UP](#) [SIGN IN](#) 

1 - 2 of 2 results for "ingress annotator"

Sort: Relevance Show: 20

### FILTERS

☐ Official ⓘ

☐ Verified publishers ⓘ

☐ CNCF ⓘ

### KIND

☐ Helm charts (2)

### CATEGORY


☐ Networking (1)

### LICENSE



☐ Apache-2.0 (1)


### OTHERS

☐ Only operators




**ingress-annotator**





 kuoss  kuoss


★ 1  **Helm chart**

Updated 11 days ago  
Version 0.3.0



Dynamically manages Kubernetes Ingress annotations based on ConfigMap rules, ensuring consi...


 **Networking**







**ingress-annotator**

 Helm  kiwigrd

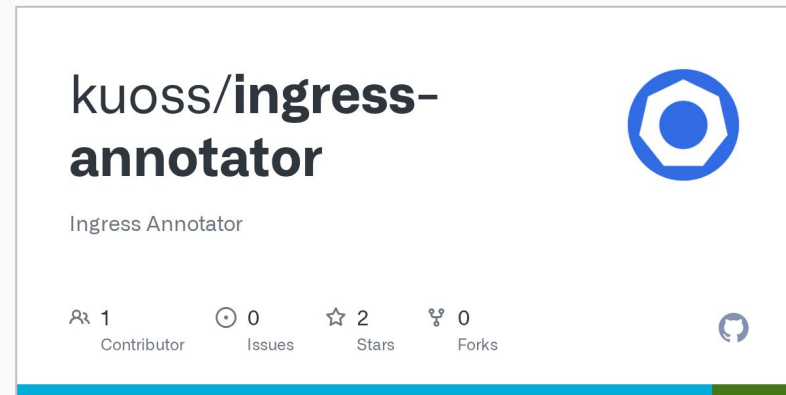
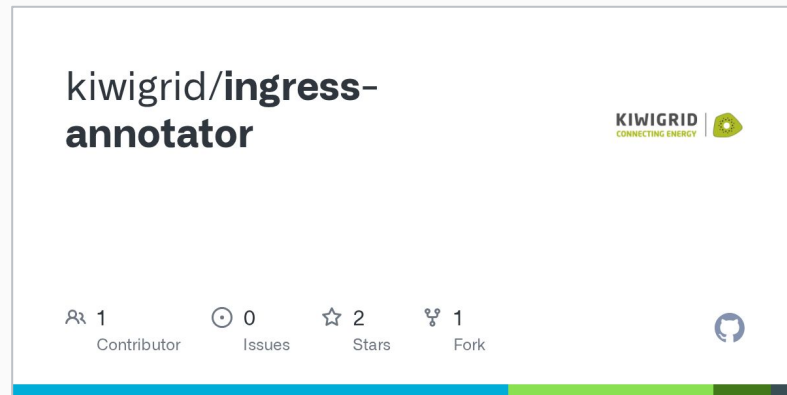
★ 0  **Helm chart**

Updated 6 years ago  
Version 0.2.0

A Helm chart for ingress annotator controller

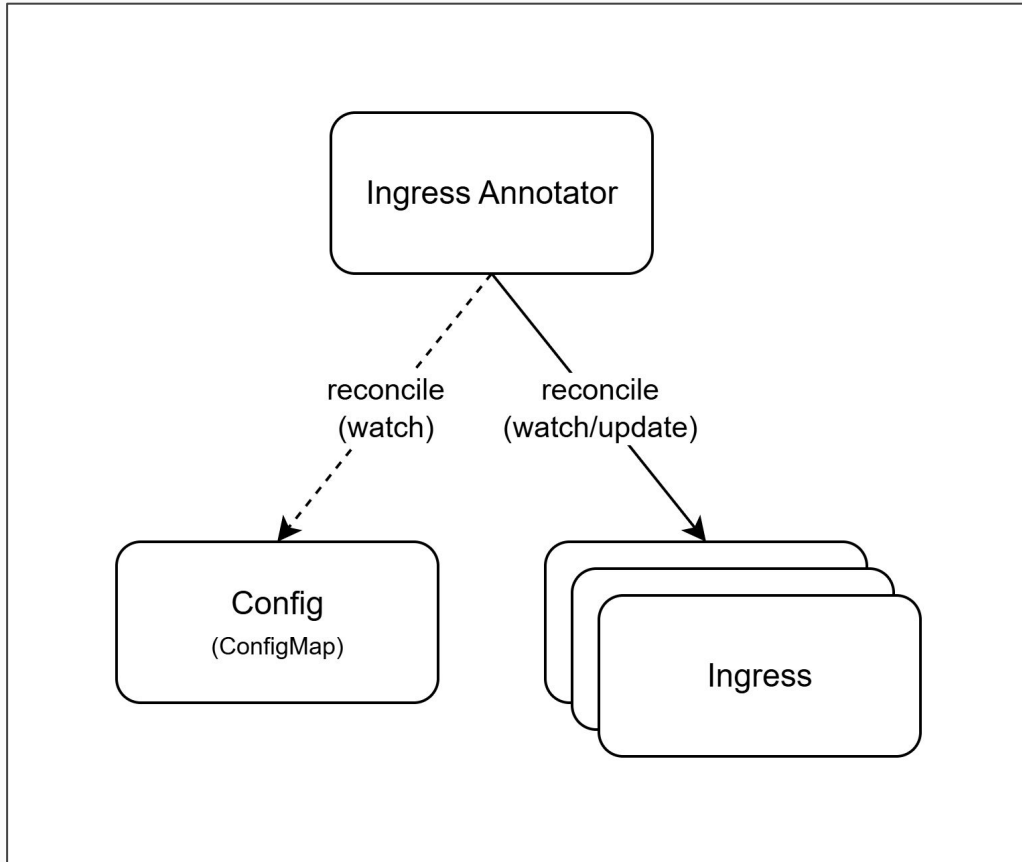
   

# 구현체 비교 #2



구분	kiwigrid	kuoss
Release	1회 v0.1.0 ('19.05)	16회 v0.1.0 ('24.08) ... v0.3.0 ('25.09)
설정 Reconcile	미지원	지원
대상 Ingress 선택	ns 1개 또는 전체 선택	ns/ing 단위 선택/제외 wildcard(*) 지원
기타	아마도 최초	listAnnotations 기능 managed-annotations 기능
공통점	<ul style="list-style-type: none"> <li>- 간단한 도구, 널리 쓰이지 않음</li> <li>- helm chart 제공</li> <li>- 모든 ingress controller 호환</li> <li>- reconciler 구현</li> </ul>	

# 구현체 비교 #3



reconcile config (kuoss)

❌ ingress-nginx 사례

# kiwigrd's config

annotations:

global:

traefik.ingress.kubernetes.io/error-pages: ...

namespaced:

default:

traefik.ingress.kubernetes.io/error-pages: ...

❌ traefik의 error-pages 어노테이션은  
폐지되고 errors 미들웨어로 교체됨

# kuoss' config (rules)

- description: error page (\*/\*)

selector:

include: "\*"

annotations:

traefik.ingress.kubernetes.io/error-pages: ...

- description: error page (default/\*, test/my-\*)

selector:

include: "default,test/my-\*"

annotations:

traefik.ingress.kubernetes.io/error-pages: ...

config의 선택터 비교

# ConfigMap

apiVersion: v1

kind: ConfigMap

metadata:

name: ingress-annotator

namespace: ingress-annotator

data:

rules: |

- description: ip-whitelist

selector:

include: dev

listAnnotations:

nginx.ingress.kubernetes.io/whitelist-source-range:

- 1.2.3.4 # devsite1

- 50.60.70.80 # devsite2

- description: oauth-secured

selector:

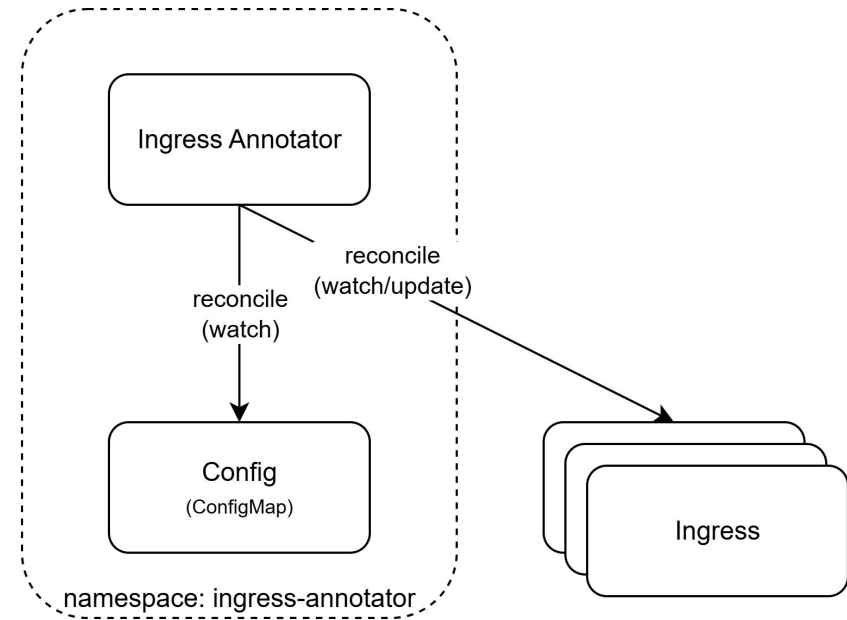
include: /\*-secured

annotations:

nginx.ingress.kubernetes.io/auth-signin: ...

nginx.ingress.kubernetes.io/auth-url: ...

단일 configMap (cluster당 1개)




정책(설정)을 변경하려면?

kubectl edit cm -n ingress-annotator  
ingress-annotator

(helm chart 설치했다면 value.yaml 수정)

# ConfigMap vs CRD


항목	ConfigMap 	CRD
성격	단순 key/value. data 필드에 YAML 문자열	리소스 스펙 정의, 구조적 필드 보유
유효성 검증	애플리케이션 수준: 컨트롤러가 직접 파싱 → 오타·형식 오류 반영될 수 있음 <i>단, Helm chart 사용 시 values 스키마로 검증</i>	API 서버 수준: 스키마 검증으로 오류 차단, 선언적 모델 강제
적용/변경 난이도	낮음, kubectl edit cm으로 즉시 수정	높음. CRD 설치/학습/버전 관리 부담
비고	가볍고 빠른 적용, 소규모 적합, 실용적	필드 확장 용이, 대규모 적합, 변경관리 공수

※ Ingress Annotator는 단순한 도구 지향  
→ 정책 저장소로 “단일 configMap” 선택



# Ingress Annotator의 Annotation 2종

구분	annotator.ingress.kubernetes.io/ managed-annotations	annotator.ingress.kubernetes.io/ reconcile
역할	기록자 (Recorder)	트리거 (Trigger)
기능	정책 annotation 자동 추가·복원, 실제 반영 상태 기록 (일관성 기준)	정책 변경 시 모든 Ingress에 재조정 이벤트 발생
시점	Reconcile Loop 수행 시	ConfigMap 정책 변경 시
지속성	지속 유지, 임의 수정 시 복원됨	임시 "true", 완료 후 제거
특징	어떤 key가 정책에 의해 관리되는지 추적 가능	전역 재조정을 위한 트리거 용도
유사사례	kubectl.kubernetes.io/ last-applied-configuration (변경 기준점 기록)	kubectl.kubernetes.io/restartedAt (트리거 신호) ※ rollout restart → 현재시각 기록 → Pod 재생성

※ 임시 트리거이므로 값 자체에 딱히 의미는 없지만,  
"true" 대신 일시 "2025-09-15T08:30:00Z" 기록하면 좋을듯 

# Reconcile 시나리오

annotator.ingress.kubernetes.io/managed-annotations

annotator.ingress.kubernetes.io/reconcile

# 1) ConfigMap 정책 설정 (👤 운영자)

allow: "10.0.0.0/8"

# 2) Ingress 생성 (👤 개발자)

limit-rate: "100k"

# 3) Ingress 조정 (🤖 컨트롤러)

managed-annotations: '{"allow":"10.0.0.0/8"}'

allow: "10.0.0.0/8"

limit-rate: "100k"

# 4) ConfigMap 정책 변경 (👤 운영자)

allow: "10.0.0.0/8"

deny: "10.1.1.1"

# 5) Ingress 조정 triggered (🤖 컨트롤러)

reconcile: "true"

managed-annotations: '{"allow":"10.0.0.0/8"}'

allow: "10.0.0.0/8"

limit-rate: "100k"

# 6) Ingress 조정 (🤖 컨트롤러)

managed-annotations: '{"allow":"10.0.0.0/8","deny":"10.1.1.1"}'

allow: "10.0.0.0/8"

deny: "10.1.1.1"

limit-rate: "100k"

# 7) ConfigMap 정책 철회 (👤 운영자, 🤖 컨트롤러)

limit-rate: "100k"

→ managed-annotations - 정책에 따라 적용된 항목 기록 (일관성을 유지하는 기준점 역할)

→ reconcile

- 정책 변경 시 전체 Ingress에 재조정 이벤트를 발생시키는 임시 트리거

# last-applied-configuration

- kubectl apply → local 매니페스트를 json 형식으로 기록함
- kubectl edit/patch/scale → 기록하지 않음

```
# demo1.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: demo
data:
  color: red
  fruit: apple
```

```
# kubectl apply -f demo1.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"color":"red","fruit":"apple"},...
data:
  color: red
  fruit: apple
```

① 매니페스트 기록

```
# demo2.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: demo
data:
  color: red
```

```
# kubectl apply -f demo2.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"color":"red"},...
data:
  color: red
```

② 예전 기록과 비교하여  
fruit 삭제됨을 인지함

③ 결과적으로 data에서 fruit 항목 삭제됨


만약 기록이 없었다면?  
→ fruit 삭제되지 않음

❌ ingress-annotator의  
managed-annotations도 같은 논리

# Client-Side Apply vs Server-Side Apply

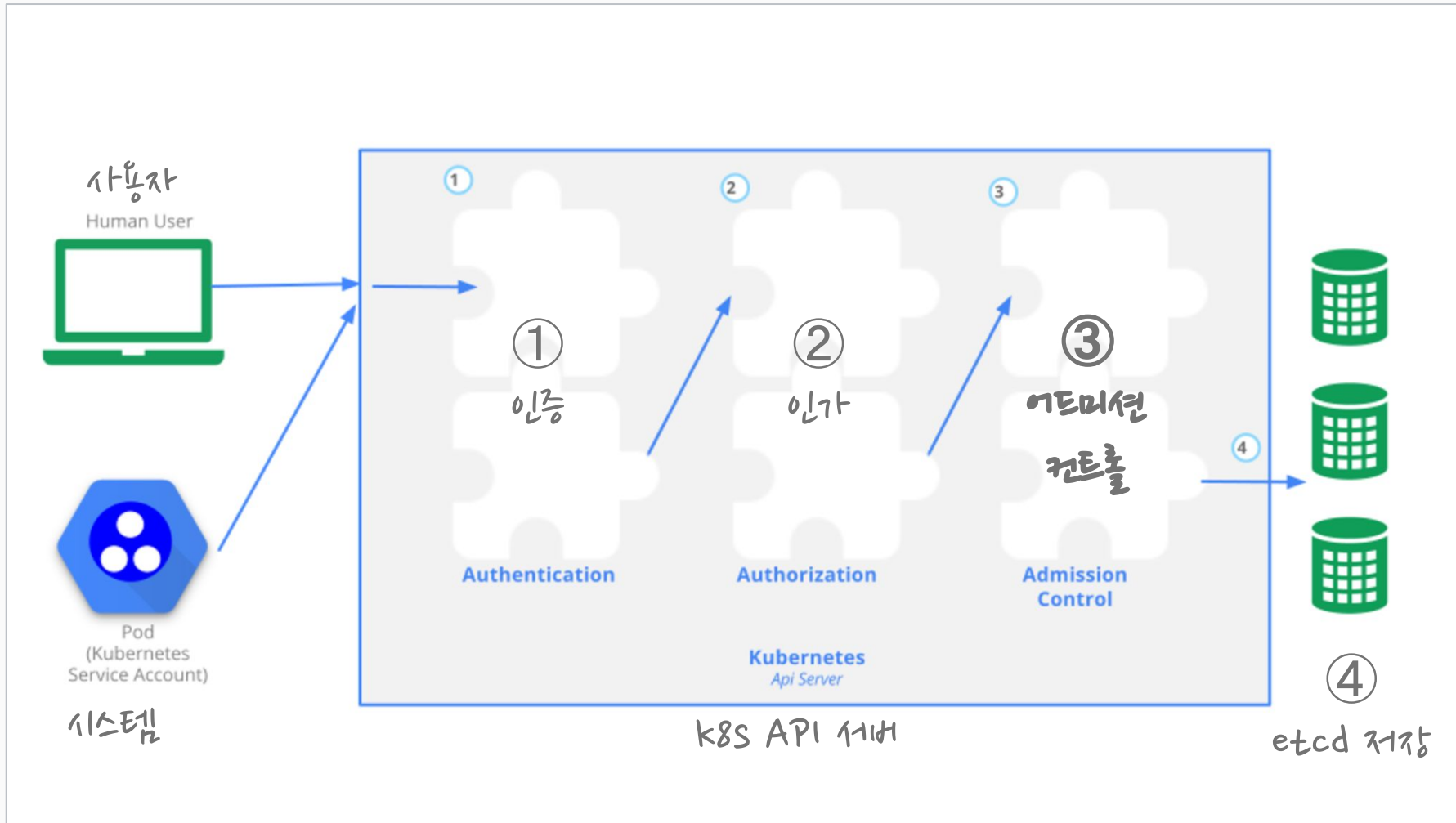
구분	Client-Side Apply (CSA) 	Server-Side Apply (SSA)
병합 위치	클라이언트 <code>last-applied + live + new</code>	서버 <code>API 서버에서 병합 수행</code>
충돌 감지	리소스 단위: 최종 작성자 wins	필드 단위 소유권 기반 충돌 감지
상태 기록	<code>last-applied-configuration</code> 어노테이션	<code>managedFields</code> 매니저, 필드별 소유권, 타임스탬프 기록
사전 비교 <code>kubectl diff</code>	클라이언트 추정치 <code>diff</code> 어노테이션 의존	실제 서버 병합 검증 <code>--dry-run=server</code>
리스트 병합	전체 교체	항목 단위 교체/병합 <code>스키마 기반 정밀 병합</code>

→ `last-applied-configuration` 한계  
항상 기록되지도 않음, 소유권 추적 불가 등

→ 추적·감사 용이한 SSA가 권장된다.  
나중에 `ingress-annotator`에도 적용하면 좋을듯 


# K8s API 접근제어

만약 Ingress Annotator에  
Admission control을 활용한다면?



# vs Admission Webhook

Admission control 구현 방법 중 하나 (외부 확장)

구분	Controller (Reconciler) 	Admission Webhook
적용 시점	사후(Eventual) 리소스 저장 후 상태 조정	사전(Sync) API 요청 시 즉시 검증/주입 (etcd 저장 전 차단)
강제력	유연한 적용 (트리프트 보정, 사후 조정) “매니저”	강제 적용 (즉시 반영) “문지기”
down시	변경 반영 지연 리소스는 남음	API 요청 자체 실패·지연
구성요소	Controller Deployment	Webhook Server + WebhookConfiguration
예시	ReplicaSet Controller replicas 수량 맞춤 Ingress Controller ingress 해석하여 LB/Proxy 설정 Ingress Annotator 규칙에 따라 annotation 부착	MutatingAdmissionWebhook Istio 사이드카 주입 ValidatingAdmissionWebhook limits 미지정 거부

만약 Ingress Annotator를 Admission Webhook으로 만든다면?

- Pros: 정책 강제력 향상 (etcd 저장 전 강제 주입)
  - Cons: 중앙 정책 변경 반영 어려움, 사후 보정 불가, 장애시 API 요청 실패 위험
- 또는 하이브리드?

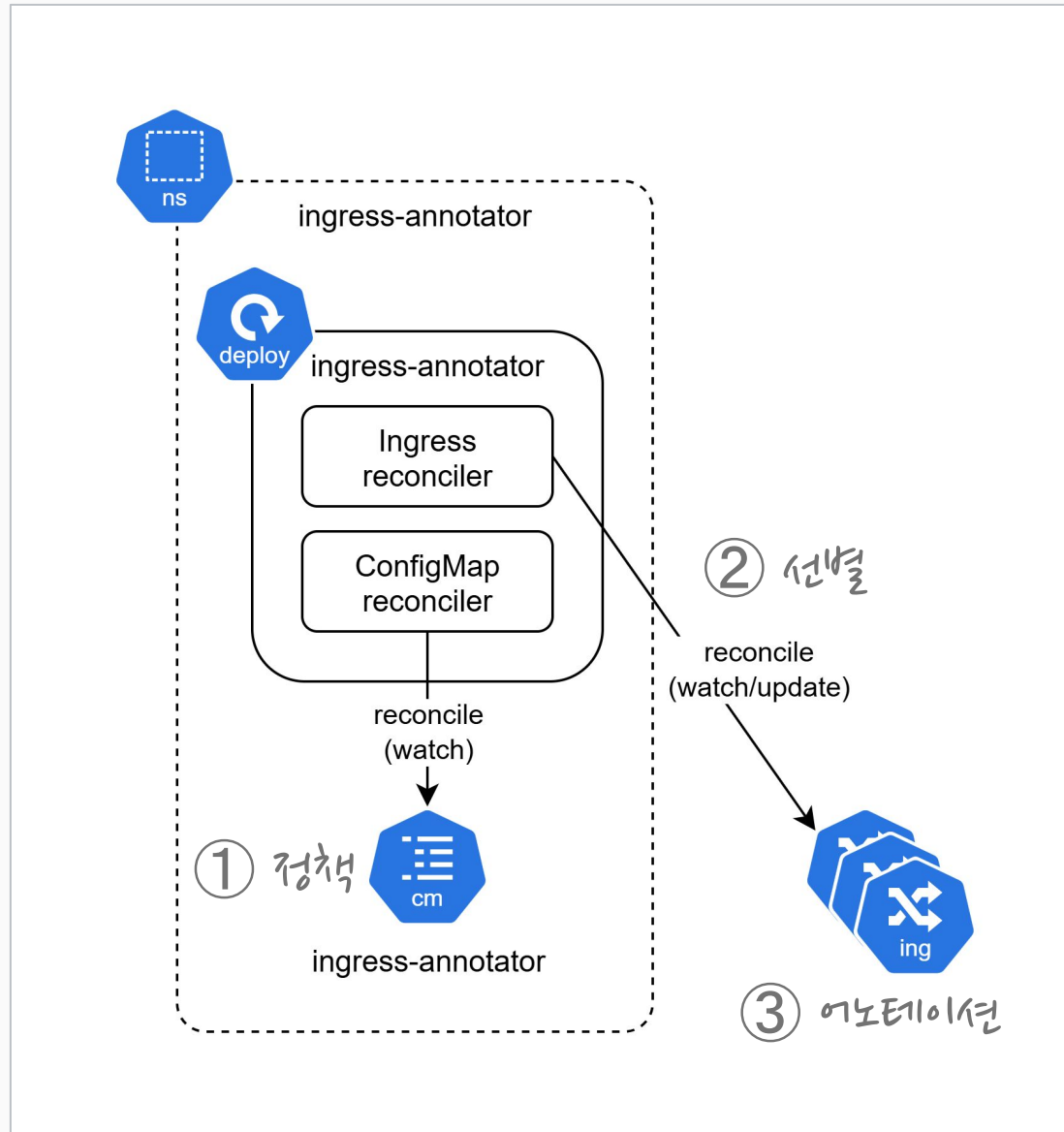
❌ 한편 validatingAdmissionPolicy는?

k8s 내장 규칙(CEL)으로 검증 (서버 불필요)

→ 검증만 가능, 수정은 불가

# 구성도

단순한 구조



이름 일치 ~~선택~~ ~~선택~~

Namespace: ingress-annotator

Deploy: ingress-annotator

configMap: ingress-annotator

① 정책에 따라

② 조건에 맞는 Ingress 선별

③ 어노테이션을 부착/제거

# III 관련 개념 정리



# Ingress #1

## Ingress

Make your HTTP (or HTTPS) network service available using a protocol-aware configuration mechanism, that understands web concepts like URIs, hostnames, paths, and more. The Ingress concept lets you map traffic to different backends based on rules you define via the Kubernetes API.

① **FEATURE STATE:** Kubernetes v1.19 [stable]

An API object that manages external access to the services in a cluster, typically HTTP.

Ingress may provide load balancing, SSL termination and name-based virtual hosting.

### Note:

Ingress is frozen. New features are being added to the [Gateway API](#).

- 클러스터 내 서비스에 대한 외부 접근을 관리하는 API 오브젝트 (주로 HTTP)
- 로드 밸런싱, SSL 종료, 이름 기반 가상 호스팅 기능 제공



example.com → Ingress → Service → Pod

# Ingress #2

Ingress is frozen...

## Ingress

Make your HTTP (or HTTPS) network service available using a protocol-aware configuration mechanism, that understands web concepts like URIs, hostnames, paths, and more. The Ingress concept lets you map traffic to different backends based on rules you define via the Kubernetes API.

**FEATURE STATE:** Kubernetes v1.19 [stable]

An API object that manages external access to the services in a cluster, typically HTTP.

Ingress may provide load balancing, SSL termination and name-based virtual hosting.

### Note:

Ingress is frozen. New features are being added to the [Gateway API](#).



- Ingress는 동결되었다.
- 새로운 기능은 Gateway API에 추가된다.

# Ingress #3

## 인그레스란?

인그레스는 클러스터 외부에서 클러스터 내부 서비스로 HTTP와 HTTPS 경로를 노출한다. 트래픽 라우팅은 인그레스 리소스에 정의된 규칙에 의해 컨트롤된다.

다음은 인그레스가 모든 트래픽을 하나의 서비스로 보내는 간단한 예시이다.

논리적 매핑!



그림. 인그레스

인그레스는 외부에서 서비스로 접속이 가능한 URL, 로드 밸런싱 트래픽, SSL / TLS 종료 그리고 이름-기반의 가상 호스팅을 제공하도록 구성할 수 있다. 인그레스 컨트롤러는 일반적으로 로드 밸런서를 사용해서 인그레스를 수행할 책임이 있으며, 트래픽을 처리하는데 도움이 되도록 에지 라우터 또는 추가 프론트 엔드를 구성할 수도 있다.

<https://kubernetes.io/ko/docs/concepts/services-networking/ingress/>

vs 실제 트래픽 경로

클라이언트 → LB/프록시 → Pod

- Service는 Endpoints 정보 통해 Pod 매핑!
- 패킷이 Ingress나 Service를 경유하는 것 아님

# Ingress #4

## Ingress Annotation

### 인그레스 리소스

최소한의 인그레스 리소스 예제:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io
spec:
  ingressClassName: nginx-examp
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

<https://kubernetes.io/ko/docs/concepts/services-networking/ingress/>

최소한(minimal)이라는데 아님..

암튼 annotation 사용은 흔함

annotation은 k8s 메타데이터 표준  
단, key/value는 구현체마다 다름

구현체는 자체 기능(옵션)을

annotation으로 노출

→ 사용자는 리소스에 지정하여 기능 활용

# Ingress Controller #1

Annotator 같은 기능 없나?

- Ingress 리소스가 실제로 동작하도록 해주는 컨트롤러
- Ingress를 해석하고, 클러스터 외부 HTTP/HTTPS 트래픽을 내부 Service로 라우팅하는 로드밸런서 구성·관리
- Ingress는 추상화 수준의 규칙만 정의하므로, 실제 동작은 Ingress Controller 구현체에 의해 결정된다.

구분	프록시 내장형 <i>controlPlane + DataPlane</i>	클라우드 LB 연계형 <i>controlPlane only</i>
예시	<ul style="list-style-type: none"><li>• Ingress Nginx Controller</li><li>• Traefik</li><li>• HAProxy Ingress</li></ul>	<ul style="list-style-type: none"><li>• AWS Load Balancer Controller</li><li>• GCE Ingress Controller (GLBC)</li></ul>
동작 방식	컨트롤러 자체가 프록시로 동작하며 설정 갱신	클라우드 LB를 생성·관리하고, 트래픽은 클라우드 LB가 처리
특징	<ul style="list-style-type: none"><li>• 클라우드/온프레미스 모두 사용 가능</li><li>• 성능은 Pod 수평 확장에 의존</li></ul>	<ul style="list-style-type: none"><li>• 클라우드 종속적</li><li>• 관리형 LB의 확장성·안정성 활용</li></ul>

# Ingress Controller #2

Annotator 같은 기능 없나?

<https://kubernetes.github.io/ingress-nginx/user-guide/nginx-configuration/configmap/>

구분	Annotation	ConfigMap
범위	개별 설정 (Ingress 리소스별)	전역 설정 (컨트롤러 전체)
예시	nginx.ingress.kubernetes.io/whitelist-source-range	whitelist-source-range
	nginx.ingress.kubernetes.io/auth-url	global-auth-url
	nginx.ingress.kubernetes.io/auth-signin	global-auth-signin
	nginx.ingress.kubernetes.io/limit-rate	limit-rate

→ 일반적으로 Annotation(개별 설정)에 대응되는 configMap(전역 설정)이 있다.

→ 전체에 적용될 뿐, 원하는 Ingress 선택은 불가

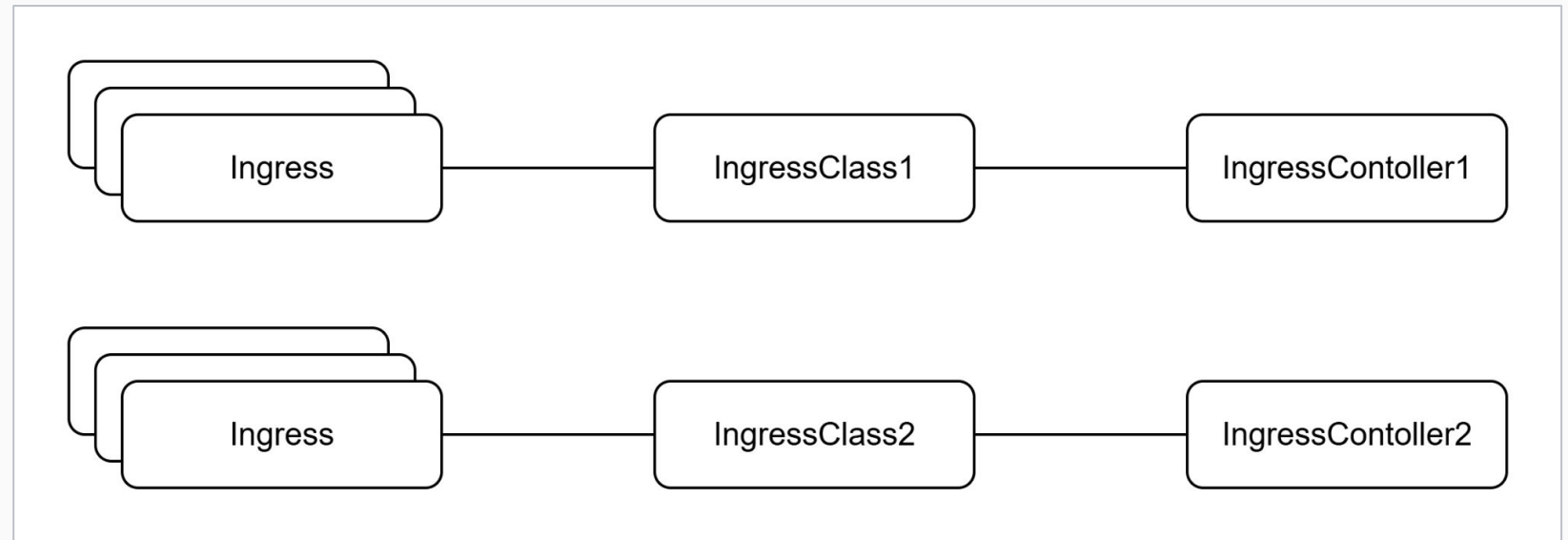
그렇다면 ingress controller를 여러 개 사용하면 어떨까?

# IngressClass

Ingress 리소스가 어떤 Ingress controller에 의해 처리될지 지정하는 리소스

```
apiVersion: networking.k8s.io/v1
kind: IngressClass
metadata:
  name: nginx-2
spec:
  controller: k8s.io/ingress-nginx-2
```

Ingress class와  
Ingress controller를  
여러 개 사용한다면?



필요한 규칙 수만큼 Ingress controller 필요 → 너무 비효율적



# ※ Class 사례

분야	Kind/.field	대상	설명
Network	IngressClass	Ingress	Ingress 컨트롤러 선택
	GatewayClass	Gateway	Gateway 구현체 선택
	.spec.loadBalancerClass	Service (LB)	LoadBalancer 컨트롤러 선택
Compute	RuntimeClass	Pod	컨테이너 런타임 선택
	PriorityClass	Pod	스케줄링 우선순위 정의
	.status.qosClass	Pod	리소스 요청·제한에 따른 QoS
Storage	StorageClass	PVC	스토리지 백엔드 및 파라미터 지정
	VolumeAttributesClass	PV	CSI 볼륨 옵션 설정
	VolumeSnapshotClass	Snapshot	스냅샷 드라이버 정의
Cluster	ClusterClass	Cluster	템플릿화된 클러스터 프로비저닝
Service Catalog	ServiceClass ClusterServiceClass	Service Catalog	외부 서비스 정의 (폐지됨)



# Traefik 사례

## Ingress controller

### 1) Ingress 사용시

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress1
  namespace: staging
  annotations:
    traefik.ingress.kubernetes.io/router.middlewares: dev-ip-allowlist@kubernetescrd
spec:
  ingressClassName: traefik
  ...
```

→ 여러 인그레스에 일괄적용하는 기능은 없음

→ ingress-annotator 사용하면 ok

```
apiVersion: traefik.io/v1alpha1
kind: Middleware
metadata:
```

```
  name: ip-allowlist
  namespace: dev
```

```
spec:
```

```
  ipAllowList:
    sourceRange:
      - 1.2.3.4/32
      - 50.60.70.80/32
      - 111.222.0.0/16
```

IP 허용목록을 Middleware로  
정의하여 재사용 가능

### 2) IngressRoute 사용시

인그레스 대용품

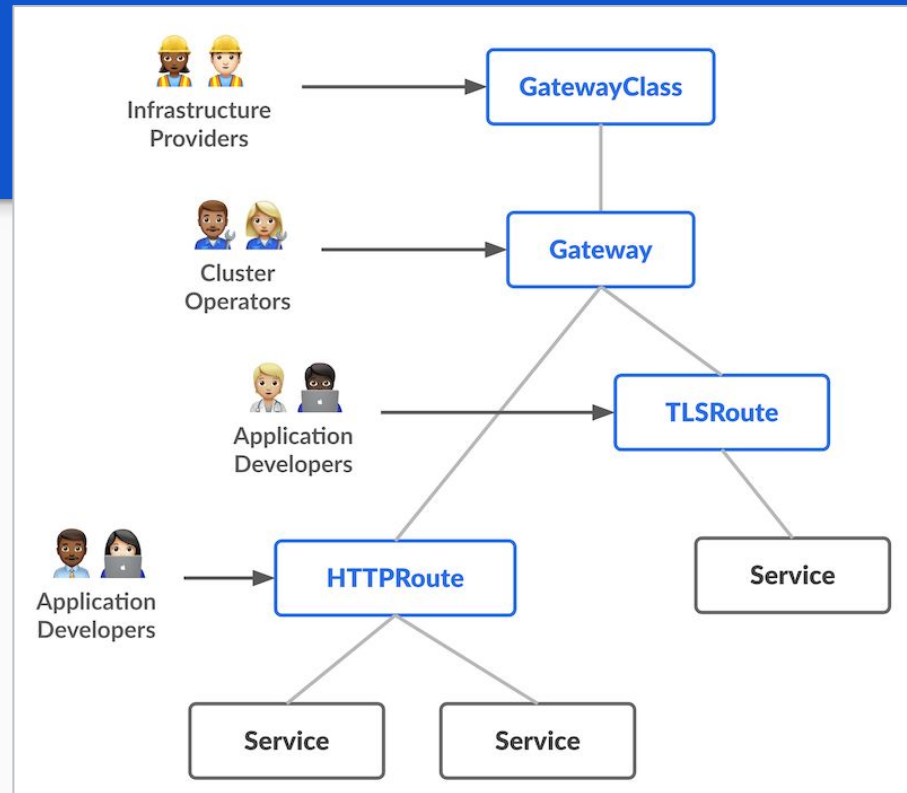
```
apiVersion: traefik.io/v1alpha1
kind: IngressRoute
metadata:
  name: ingressroute1
  namespace: staging
spec:
  routes:
    - middlewares:
      - name: ip-allowlist
        namespace: dev
```

# Gateway API #1

## Introduction



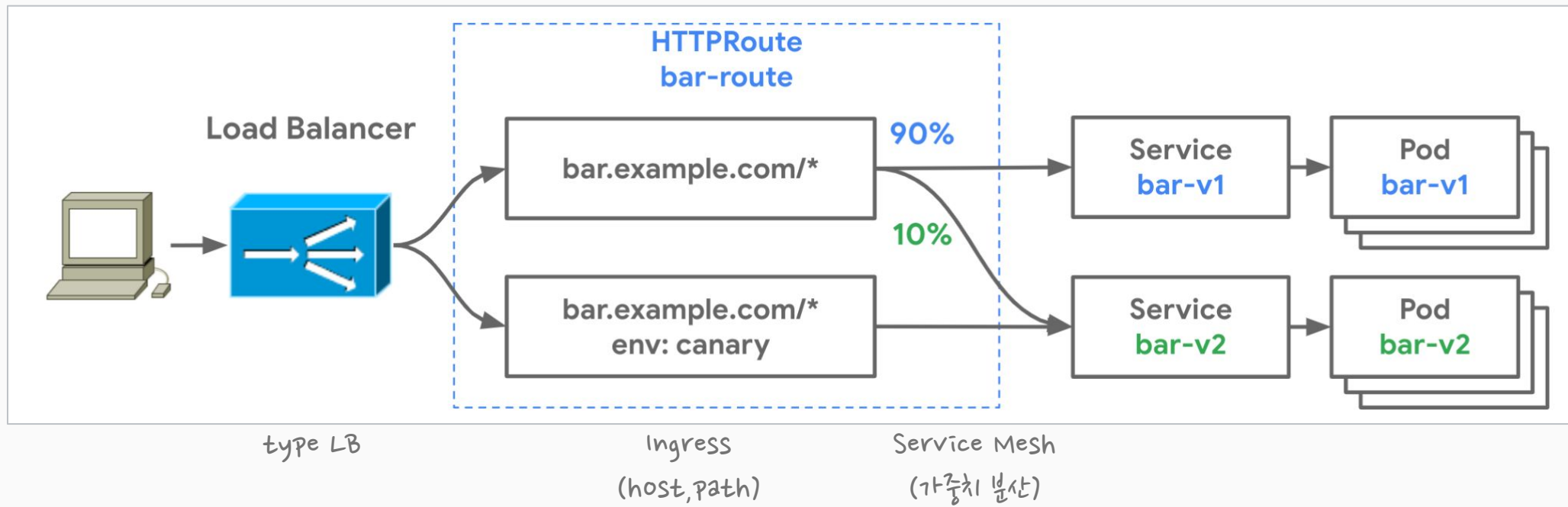
# gateway api



- Gateway API는 쿠버네티스 공식 프로젝트로, L4/L7 라우팅을 다룬다.
- Ingress, 로드밸런싱, 서비스 메시 API의 차세대 모델을 대표한다.
- 처음부터 범용적·표현력 풍부·역할 지향적으로 설계되었다.

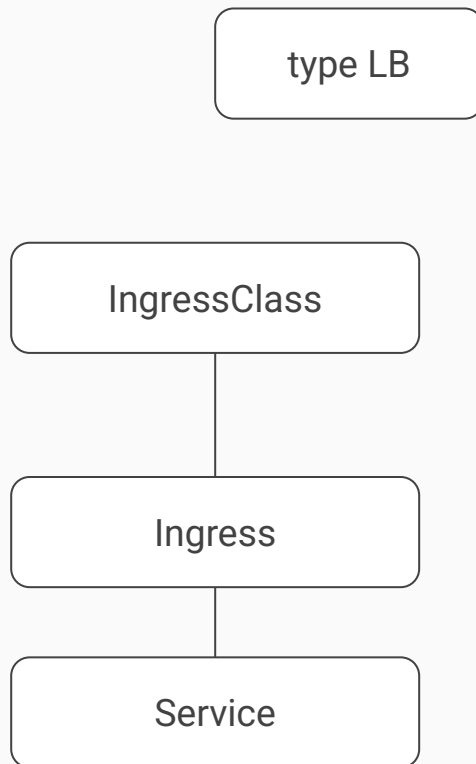
# Gateway API #2

- k8s 네트워킹 차세대 표준
- type LB, Ingress, Service Mesh 아우르는 대통합 후속 모델

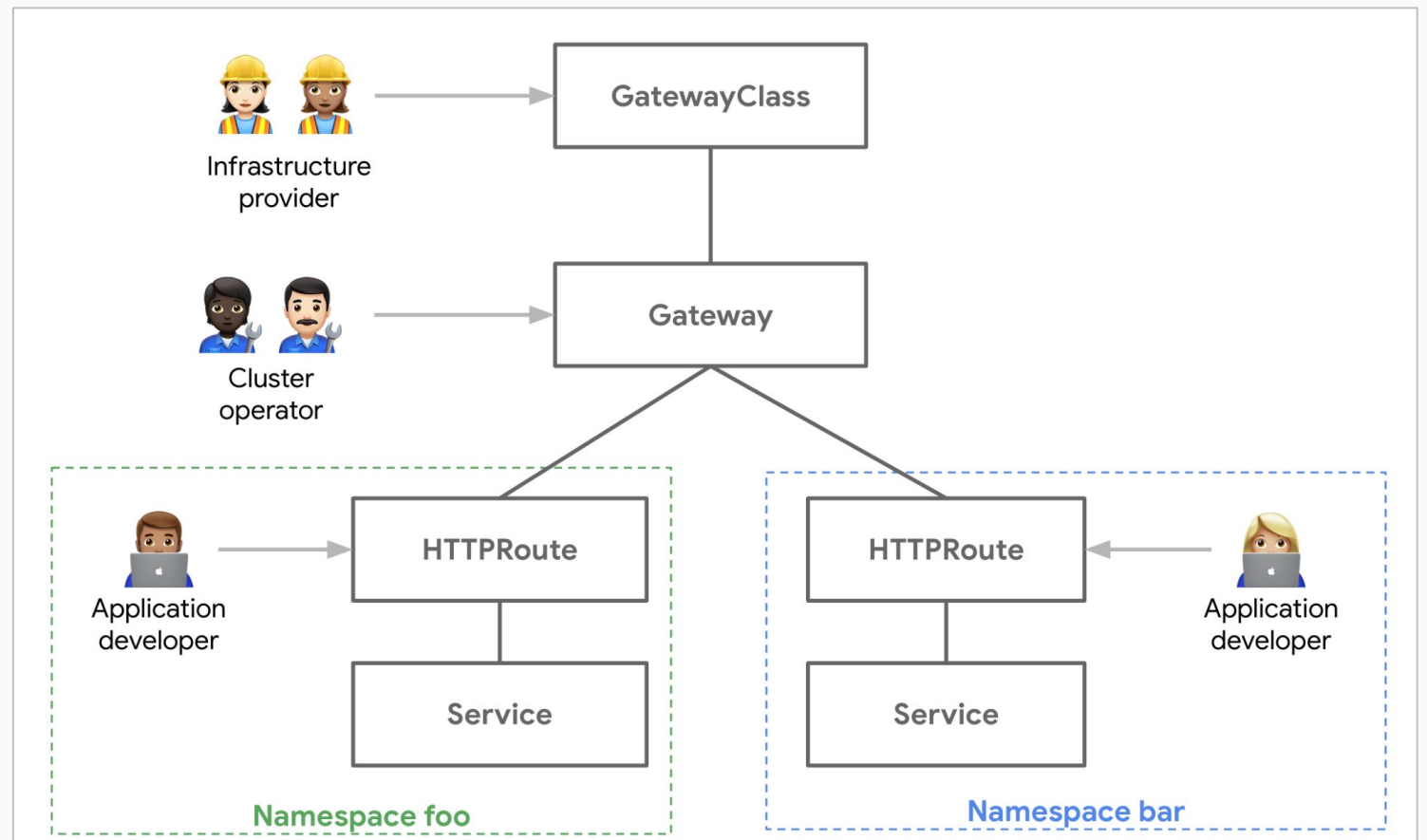


# Gateway API #3 vs Ingress

Ingress with



Gateway API with



<https://kubernetes.io/blog/2021/04/22/evolving-kubernetes-networking-with-the-gateway-api/>

# Gateway API #4 *vs Ingress*

Ingress의 한계를 보완하고, 정책 기반·역할 기반 접근 제공

구분	Ingress	Gateway API
목적	단순 HTTP(S) 진입점	다양한 L4/L7 네트워크 트래픽 관리
구조	단일 리소스(Ingress)	역할 분리된 다중 리소스(Gateway, Route, Policy 등)
확장성	Annotation 중심	spec 기반 확장 (표준 필드)
표현력	제한적 (host·path 기반 라우팅)	풍부 (다중 리스너, TCP/UDP, TLS, 필터 체인)
생태계	널리 사용됨	점진적 확산

# Gateway API #5 Policy Attachment?

Ingress 시대

Gateway API 시대

구분	Ingress Annotation	Policy Attachment
형태	단순 key/value (메타데이터)	CRD 객체 (스키마 기반)
표준화	없음 (컨트롤러별 제각각)	Gateway API 표준 패턴
유효성 검증	불가 <small>단순 문자열</small>	가능 <small>스키마/CRD</small>
대상	Ingress 리소스	Gateway/Route 리소스
관리 범위	개별 리소스 단위	Direct / Inherited (계층적)
상태	안정적, 광범위 사용	실험적

왜 비교하나? Ingress Annotator의 “정책”을 대신할 수 있을까 해서...

→ Traefik의 Middleware처럼 일부 기능을 대신할 가능성이 있어 보이기도 하는데 아직 잘 모르겠음

# Gateway API #6

Gateway API 구현체 및 ingress의 미래

관련 구현체	제공 주체	설명
InGate	SIG-Network	Ingress + Gateway API 통합 컨트롤러 <small>아직 release 없음</small>
Traefik Proxy	Traefik Labs	Ingress + Gateway API 모두 지원
NGINX Gateway Fabric	F5 NGINX	NGINX 기반 Gateway API 구현
Istio	Istio Community	Service Mesh + Gateway API, 정책/보안 기능 풍부
Cilium	Isovalent	eBPF 기반 네트워킹, Gateway API 지원
Contour	VMware	Envoy 기반, Gateway API 지원

## ※ ingress2gateway (SIG-Network)

- Ingress → Gateway API 리소스 변환 도구 (Mig 가능)
- 사용자는 Ingress를 사용하다가 고급기능 필요시 전환 고려

## Ingress는 계속 공존

- 강점: 단순성/편리함, 높은 보급률
- 많은 Gateway API 구현체들이 Ingress도 함께 지원
- deprecate 계획 없음 (Gateway API FAQ)

# Label vs Annotation

구분	Label	Annotation
용도	리소스 선택·분류·식별 <i>selector, scheduling, grouping</i>	세부 정책·도구 메타데이터 저장 <i>컨트롤러 확장, 운영 기록, 부가 설정</i>
변경시	스케줄링·서비스 연결에 영향	컨트롤러 동작 제어 및 동작에 영향
키 길이	최대 317자 <i>prefix(253)/name(63)</i>	최대 317자 <i>prefix(253)/name(63)</i>
값 길이	최대 63자	최대 262144자 (256KiB) <i>JSON 문자열 저장 가능</i>
예시	<p>app: mysql  app.kubernetes.io/name: mysql  apps.kubernetes.io/pod-index: "0" <i>sts</i></p> <p>kubernetes.io/os: linux  disktype: ssd</p>	<p>xxx.ingress.kubernetes.io/...  <del>kubernetes.io/ingress-class</del> .spec.ingressClassName  ingressclass.kubernetes.io/is-default-class</p> <p>kubectl.kubernetes.io/default-container "logs/exec"  kubectl.kubernetes.io/last-applied-configuration</p>



# Annotation 사례 #1

cert-manager



```
apiVersion: v1
kind: Secret
metadata:
  name: example-com-tls
  namespace: cert-manager
  annotations:
    cert-manager.io/alt-names: '*.example.com,example.com'
    cert-manager.io/certificate-name: example-com-tls
    cert-manager.io/common-name: example.com
    cert-manager.io/issuer-kind: Issuer
    cert-manager.io/issuer-name: letsencrypt-prod
data:
  tls.crt: LS0tLS1CRUd...
  tls.key: LS0tLS1CRUd...
type: kubernetes.io/tls
```

certificate가 발급되면  
tls 타입의 Secret이 생성됨

certificate 관련 정보가  
annotation으로 기록됨

# Annotation 사례 #2 *reflector*

```
apiVersion: v1
kind: Secret
metadata:
  name: example-com-tls
  namespace: cert-manager
  annotations:
    cert-manager.io/alt-names: '*.example.com,example.com'
    cert-manager.io/certificate-name: example-com-tls
    cert-manager.io/common-name: example.com
    cert-manager.io/issuer-kind: Issuer
    cert-manager.io/issuer-name: letsencrypt-prod
    reflector.v1.k8s.emberstack.com/reflection-allowed: "true"
    reflector.v1.k8s.emberstack.com/reflection-auto-enabled: "true"
data:
  tls.crt: LS0tLS1CRUd...
  tls.key: LS0tLS1CRUd...
type: kubernetes.io/tls
```

## emberstack/kubernetes-reflector



Custom Kubernetes controller that can be used to replicate secrets, configmaps and certificates.

24  
Contributors

3  
Issues

26  
Discussions

1k  
Stars

108  
Forks



certificate, Secret은

네임스페이스 범위

reflector로 Secret/configMap을

다른 네임스페이스에 복제 가능

→ 여러 네임스페이스에서 인증서  
간편하게 사용

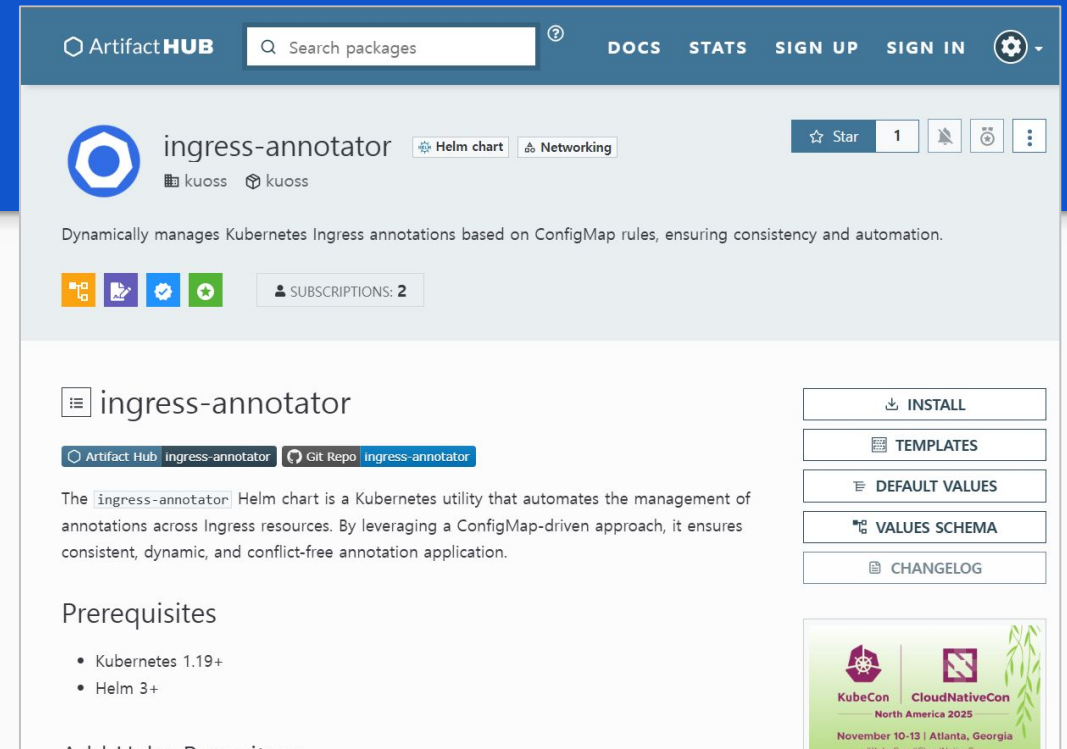
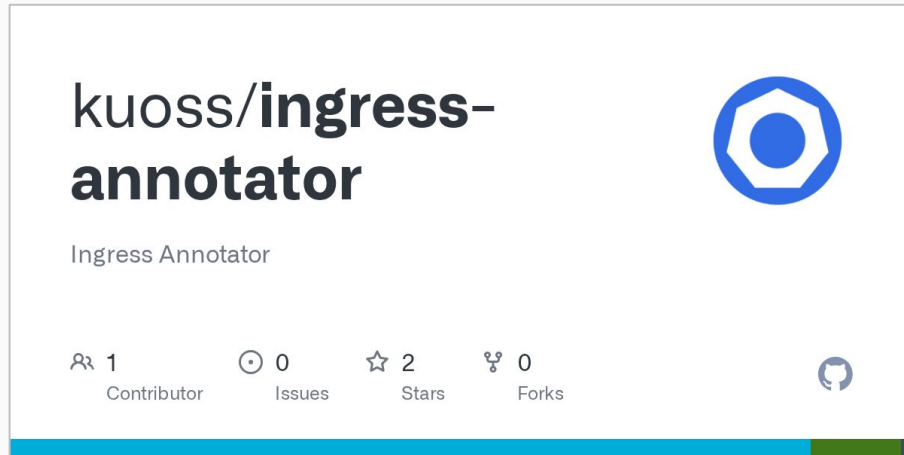
cert-manager — reflector

ingress-controller — ingress-annotator

약간 유사한 관계 (보조 역할)

# IV 마무리

# Ingress Annotator



<https://github.com/kuoss/ingress-annotator>

정책에 따라 Ingress에 Annotation을 붙여주는 도구(컨트롤러)

- 사용방법이 간단함. 그러저럭 잘 설치되됨

- 상황에 따라 유용한 도구. 사용 및 참여 바랍니다.

# EOF

## 요약

- 1. 소개 Ingress Annotator 개발 배경, 설치/사용 방법
- 2. 설계 논의 Selector, config, Reconcile, Admission webhook
- 3. 관련 개념 정리 Ingress, Gateway API, Annotation 사례

## What's next 기회가 된다면...

- Lethe 로깅 성능 테스트
- Kubernetes the Easy Way 다시보기