

# Helm Chart 개발과 GitHub · ArtifactHub 사용기

김정민 삼성SDS

KCD Seoul 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>

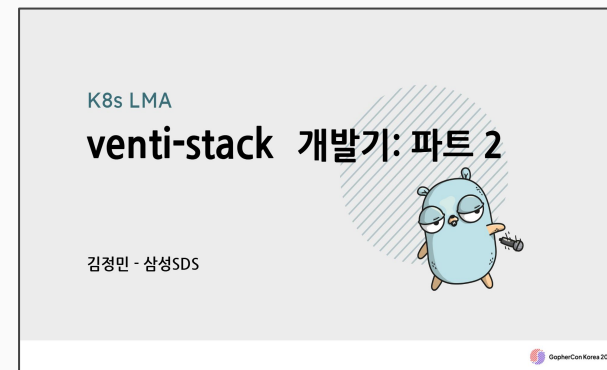


## 지난 발표



'24.09

Cloud Native Korea Community Day 2024



'24.10

GopherCon Korea 2024

<https://github.com/jmnote/slides> 지난 발표자료를 여기에

## 개발한 차트

차트	설명
ingress-annotator	정의한 규칙에 따라 ingress에 annotations 전파
lethe-stack	로깅 스택
venti-stack	로깅·모니터링 스택 <small>지난 발표자를 참고</small>

<https://github.com/kuoss/helm-charts> 참여 환영합니다.



# 목차

- I . Helm 작동 원리
- II . 다른 도구와 비교
- III . Chart 개발 실무
- IV . GitHub Actions로 Chart 릴리스
- V . ArtifactHub 등록

# I Helm 작동 원리

# Helm 기초

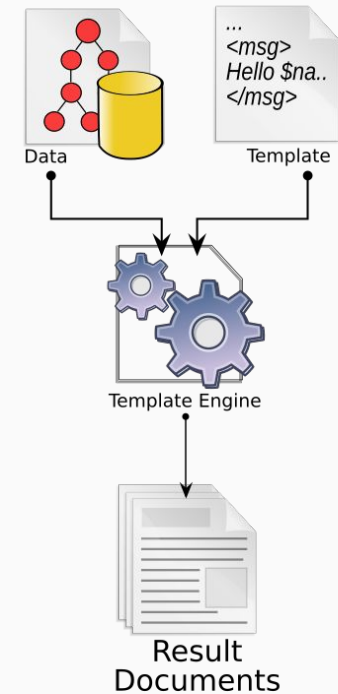
## 기초 용어

용어	설명
Helm	Kubernetes 리소스를 템플릿 기반으로 패키징·배포하는 도구
Repo	여러 Chart를 저장·배포하는 Helm 저장소 (index.yaml* 포함)
Chart	Kubernetes 리소스의 템플릿과 values.yaml을 포함한 패키지
Release	Chart를 클러스터에 설치한 것 (인스턴스, 설치의 단위)

index.yaml (repo당 하나, 차트 x 버전 목록)

## 설치 흐름

1. values.yaml 설정값으로 템플릿 렌더링
2. 렌더링된 Kubernetes Manifest를 클러스터에 적용
3. Release 기록 추적/롤백 가능



템플릿 엔진 개념도

[https://en.wikipedia.org/wiki/Template\\_processor](https://en.wikipedia.org/wiki/Template_processor)

# 자주 쓰는 Helm 명령어 (사용자 편) #1

repo 등록 `helm repo add prometheus-community https://prometheus-community.github.io/helm-charts`

repo 갱신 `helm repo update`

차트 설치 `helm install [RELEASE_NAME] prometheus-community/prometheus`

릴리스 삭제 `helm uninstall [RELEASE_NAME]`

릴리스 업그레이드 `helm upgrade [RELEASE_NAME] prometheus-community/prometheus`

- repo 등록은 로컬에 ( `~/.config/helm/repositories.yaml` )
- repo 갱신은 index.yaml 다운로드 ( `~/.cache/helm/repository/[REPO_NAME]-index.yaml` )
- 차트 설치 = 릴리스 생성
- 차트명 대신 디렉토리, tgz파일 입력 가능 ( 개발 중인 차트 디렉토리, helm pull하여 받은 차트 tgz 파일 )

# 자주 쓰는 Helm 명령어 (사용자 편) #2

## 릴리스 목록

```
$ helm ls -A
```

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART	APP VERSION
cert-manager	cert-manager	1	2025-04-13 ...	deployed	cert-manager-v1.17.1	v1.17.1
in	ingress-nginx	49	2025-03-31 ...	deployed	ingress-nginx-4.12.1	1.12.1
op	oauth2-proxy	6	2025-03-19 ...	deployed	oauth2-proxy-7.12.6	7.8.1
reflector	reflector	1	2025-04-13 ...	deployed	reflector-9.0.322	9.0.322

- 릴리스는 네임스페이스에 소속된다(namespaced).
- 업그레이드 하면 REVISION이 증가한다.
- 차트 버전, 상태, 업데이트 일시를 알 수 있다.

이 정보는 어디에 있나? ingress-nginx REVISION 49를 알아보자.



# 릴리스 정보 #1

```
$ helm history -n ingress-nginx in
```

REVISION	UPDATED	STATUS	CHART	APP VERSION	DESCRIPTION
40	Sat Jul 27 ...	superseded	ingress-nginx-4.11.1	1.11.1	Upgrade complete
41	Sat Jul 27 ...	superseded	ingress-nginx-4.11.1	1.11.1	Upgrade complete
42	Sat Jul 27 ...	superseded	ingress-nginx-4.11.1	1.11.1	Upgrade complete
43	Sat Jul 27 ...	superseded	ingress-nginx-4.11.1	1.11.1	Upgrade complete
44	Wed Oct 23 ...	superseded	ingress-nginx-4.11.3	1.11.3	Upgrade complete
45	Wed Oct 23 ...	superseded	ingress-nginx-4.11.3	1.11.3	Upgrade complete
46	Tue Nov 5 ...	superseded	ingress-nginx-4.11.3	1.11.3	Upgrade complete
47	Mon Dec 30 ...	superseded	ingress-nginx-4.11.3	1.11.3	Upgrade complete
48	Tue Jan 7 ...	superseded	ingress-nginx-4.12.0	1.12.0	Upgrade complete
49	Mon Mar 31 ...	deployed	ingress-nginx-4.12.1	1.12.1	Upgrade complete

- history(revision)은 10개가 보존된다.
- 이중에 하나로 rollback할 수 있다.
- 차트(버전)가 바뀐 경우와 그렇지 않은 경우가 있다.

## 릴리스 정보 #2

```
$ kubectl -n ingress-nginx get secret
```

NAME	TYPE	DATA	AGE
...			
sh.helm.release.v1.in.v40	helm.sh/release.v1	1	297d
sh.helm.release.v1.in.v41	helm.sh/release.v1	1	297d
sh.helm.release.v1.in.v42	helm.sh/release.v1	1	297d
sh.helm.release.v1.in.v43	helm.sh/release.v1	1	297d
sh.helm.release.v1.in.v44	helm.sh/release.v1	1	209d
sh.helm.release.v1.in.v45	helm.sh/release.v1	1	209d
sh.helm.release.v1.in.v46	helm.sh/release.v1	1	196d
sh.helm.release.v1.in.v47	helm.sh/release.v1	1	141d
sh.helm.release.v1.in.v48	helm.sh/release.v1	1	133d
sh.helm.release.v1.in.v49	helm.sh/release.v1	1	50d

→ 릴리스 정보는 secret에 저장된다.

( 지정사항에 따라 configMap에 저장도 가능 )

→ secret과 revision은 1:1 관계

# 릴리스 정보 #3

```
$ kubectl -n ingress-nginx get secret sh.helm.release.v1.in.v49 -o jsonpath="{.data.release}" \
| base64 -d | base64 -d | gzip -d | jq 'keys' -c
["chart","config","hooks","info","manifest","name","namespace","version"]
```

→ helm은 gzip압축 + base64인코딩하여 저장 (secret이므로 한번 더 인코딩됨)

키	설명
chart	차트 정보 <small>메타데이터, 템플릿, values 등</small>
config	사용자가 지정한 설정값 <small>values.yaml, --set</small>
hooks	차트에 정의된 Helm Hook 목록 <small>pre-install, post-upgrade 등</small>
info	릴리스 메타데이터 <small>상태, 배포시각 등</small>
manifest	최종 렌더링된 manifest YAML <small>텍스트 형식</small>

# Helm의 장점

- 재사용 가능한 배포 템플릿
- 차트 내 컴포넌트간 호환성 검증
- 버전 관리 및 롤백 기능

prometheus 차트 내 컴포넌트

- prometheus
- prometheus-pushgateway
- node-exporter
- kube-state-metrics
- alertmanager

```
$ helm search repo prometheus
```

NAME	CHART VERSION	APP VERSION	DESCRIPTION
prometheus-community/kube-prometheus-stack	70.7.0	v0.81.0	kube-prometheus-stack colle...
prometheus-community/prometheus	27.10.0	v3.2.1	Prometheus is a monitoring ...
prometheus-community/prometheus-adapter	4.14.1	v0.12.0	A Helm chart for k8s promet...
...			
grafana/loki-stack	2.10.2	v2.9.3	Loki: like Prometheus, but ...
grafana/snyk-exporter	0.1.0	v1.4.1	Prometheus exporter for Snyk.
kuoss/venti-stack	0.3.12	v0.2.24	Venti stack is a logging an...

- 이것 대신 ArtifaceHub에서 검색하자.

## II 다른 도구와 비교

# Kubernetes Manifest

## 장점

- k8s 리소스 명세 그 자체
- kubectl 이외 다른 도구 불필요
- 직관적 렌더링 불필요
- 높은 자유도

## 단점

- 많은 리소스 관리 어려움 *YAML 지옥?*
- 환경별 관리 어려움 *불필요한 중복 많음*
- 버전/릴리스 관리 미지원 *대안: git으로 관리*
- 컴포넌트 호환성 직접 검증 필요

→ 단순한 애플리케이션 관리에 적합

→ 모든 Manifest 변경사항을 git으로 관리하고자 할 때

```
mystack/  
├── dev/  
│   ├── prometheus/  
│   │   ├── configmap.yaml  
│   │   ├── deployment.yaml  
│   │   └── service.yaml  
│   ├── prometheus-pushgateway/  
│   │   └── ...  
│   ├── node-exporter/  
│   │   └── ...  
│   ├── kube-state-metrics/  
│   │   └── ...  
│   ├── alertmanager/  
│   │   └── ...  
└── prod/  
    ├── prometheus/  
    │   └── ...  
    └── ...
```

```
$ kubectl apply -R -f mystack/dev/
```

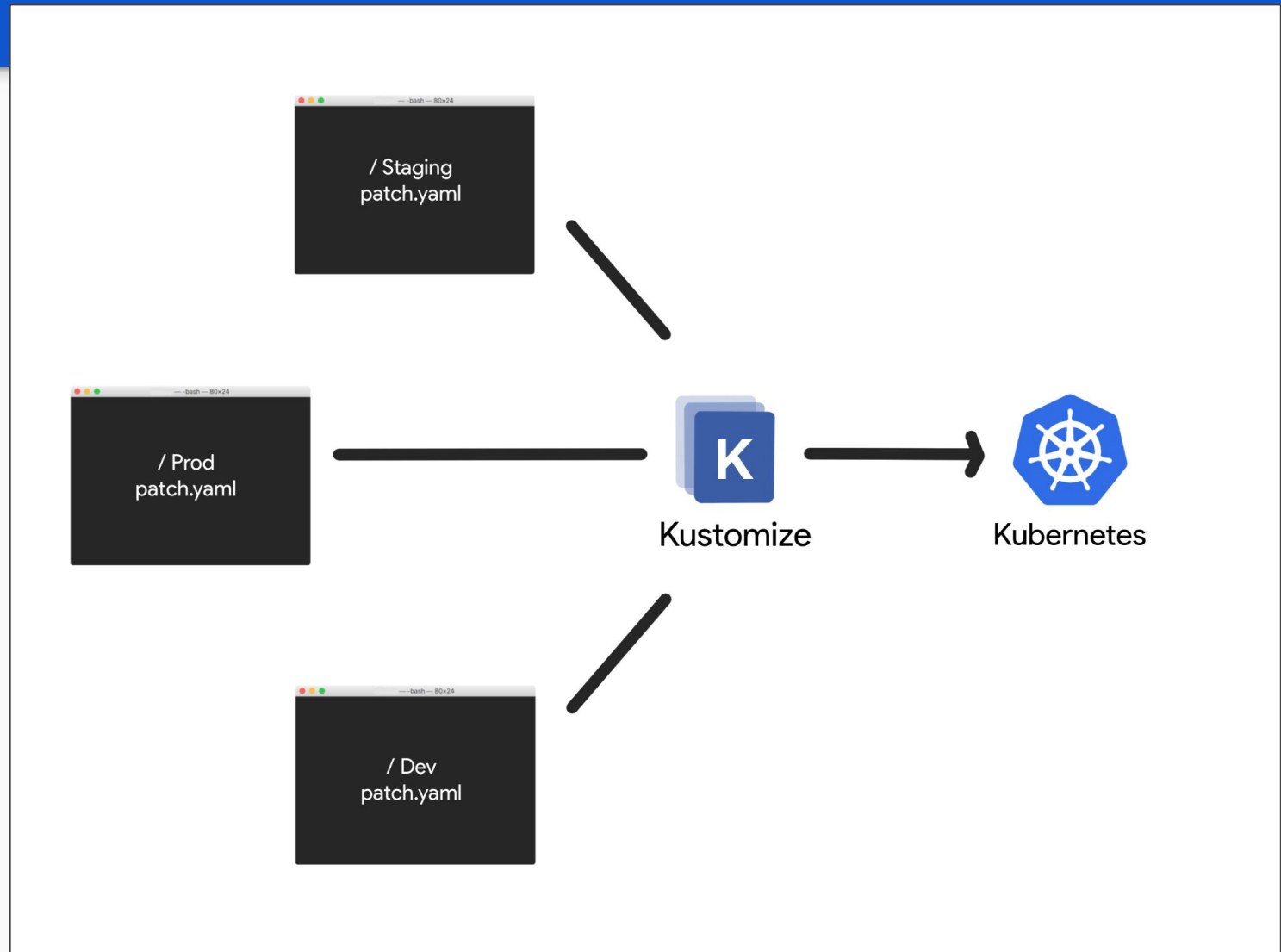
# Kustomize #1

Kubernetes Manifest를

템플릿 없이 커스터마이징하는 도구

기본 YAML 파일을 수정하지 않고도

환경별로 필요한 변경사항을 오버레이 방식으로 적용



# Kustomize #2

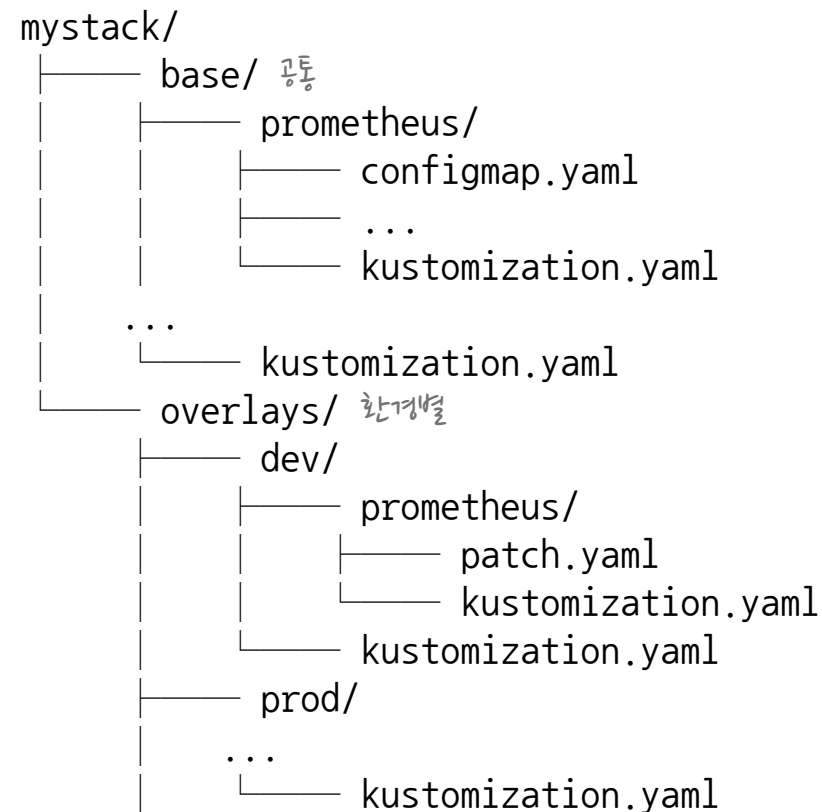
## 장점

- 기존 Manifest 활용 전환 용이함
- 별도 배포 도구 불필요 `kubectl apply -k`
- 높은 자유도 Manifest YAML 직접 조작 가능
  - 필드 값 교체/추가
  - 예: 사이드카 컨테이너 주입
- 환경별 구성 관리 용이 `base/, dev/, prod/`
- ConfigMap 변경시 롤링 업데이트 가능 (hash 부여됨)

## 단점

- 컴포넌트 호환성 직접 검증 필요
- 버전/릴리스 관리 미지원 대안: git으로 관리

→ 자체 애플리케이션의 구성 명시적·유연하게 관리

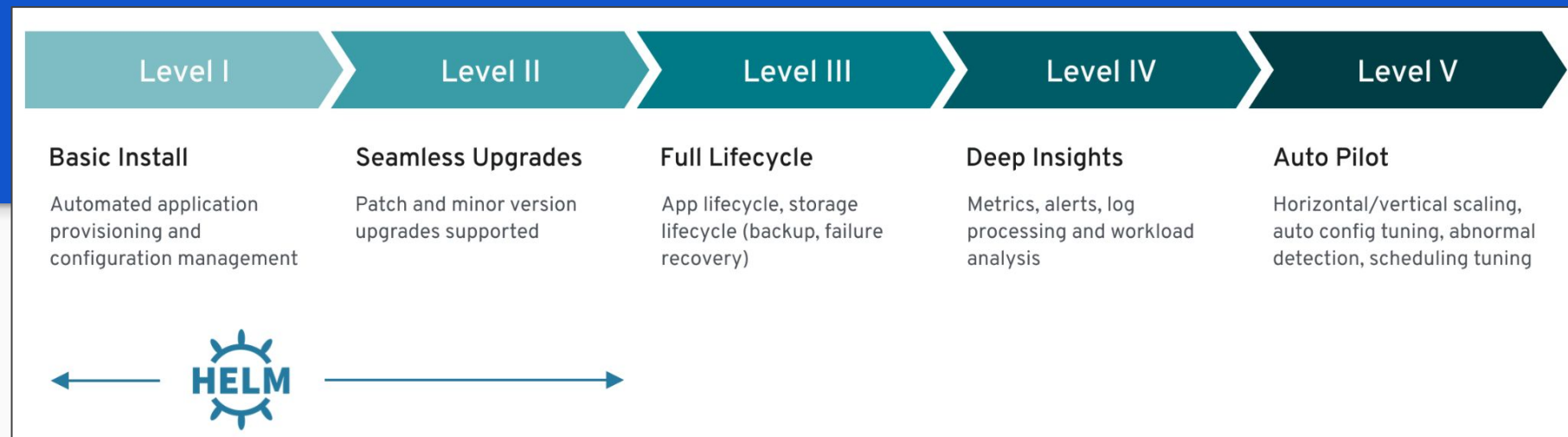


```
$ kubectl apply -k mystack/overlays/dev/
```

※ `kustomization.yaml`에서 헬름 차트를 사용하는 것도 가능  
(`values.yaml`로 설정 불가능한 항목 patch 필요시)



# Operator #1



Operator 역량 수준 <https://sdk.operatorframework.io/docs/overview/>

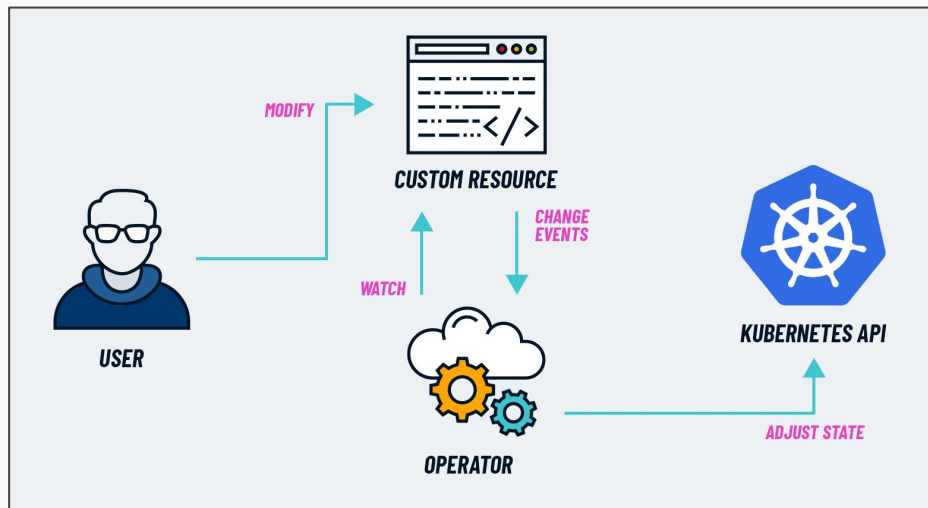
## 용어 정리

용어	설명
Operator 패턴	선언형 k8s 환경에 운영 자동화를 구현하는 설계 방식 <i>custom Resource + controller 조합</i>
Operator 도구	Operator 패턴 구현을 지원하는 도구 <i>kubebuilder, operator SDK 등</i>
Operator	Operator 패턴을 구현한 애플리케이션 <i>특정 도메인 운영 자동화 수행. 예: cert-manager, prometheus-operator</i>
OLM Operator	OLM 규격에 맞게 패키징된 Operator <i>OLM으로 설치 가능. operatorHub, ArtifactHub 등록 가능</i>
OLM	OLM Operator를 관리하는 도구 <i>operator Lifecycle Manager (operator 설치, 업데이트, 의존성, 수명주기 관리)</i>
Operator Framework	OLM, Operator SDK, Operator Registry 등을 포함하는 도구 모음 <i>생태계</i>

<https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>

<https://github.com/operator-framework>

# Operator #2



<https://www.cncf.io/blog/2022/06/15/kubernetes-operators-what-are-they-some-examples/>

## cert-manager 사례

- k8s 내 TLS 인증서 자동 관리  
certificateRequest → certificate 생성/갱신
- 주요 사용사례: ingress 인증서 연계 자동갱신
- 보통 Helm Chart로 설치 Helm 기반 operator  
❌ 차트에 들어있는 것은 operator이지만 OLM operator는 아님  
( cert-manager OLM operator도 따로 있긴 함 )

- 장점**
- 운영 자동화 설치, 백업/복구, 업그레이드, 상태 점검 스케일링 등
  - 복잡한 로직 처리 가능 헬름 차트로는 구현이 어려운 동작 가능

- 단점**
- 높은 복잡도 custom Resource, controller에 대한 이해 필요
  - 배포/디버깅 어려움 구성요소가 많음

참고) Operator Framework가 지원하는 operator

- Go 기반 operator
- Helm 기반 operator (차트 wrapping. cert-manager 사례와 다름)
- Ansible 기반 operator

일반적으로 OLM operator보다 Helm chart를 많이 사용함

- OLM 불필요. 설치 간편. 진입장벽 낮음, 커뮤니티 활성화

# 다중 비교

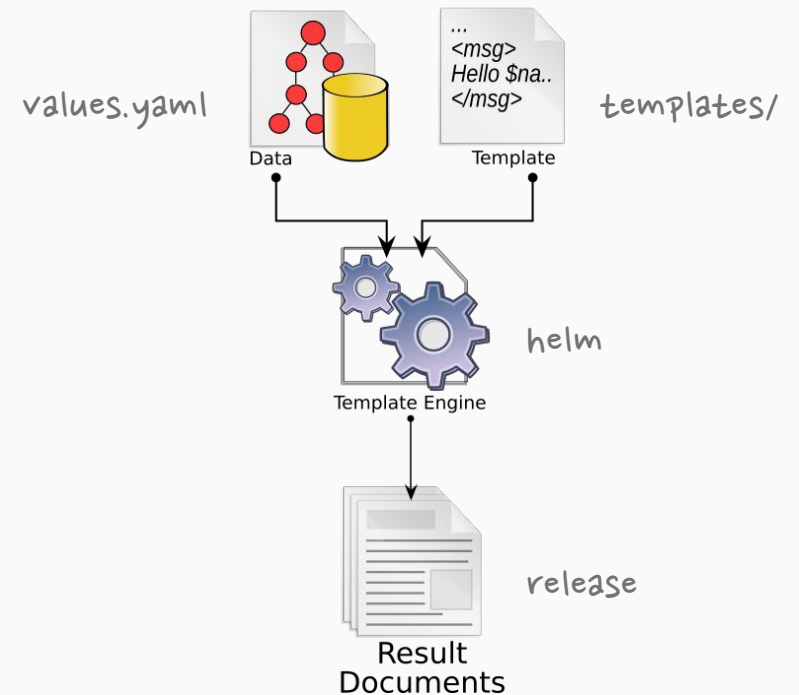
구분	Manifest	Kustomize	Helm Chart	Operator
형태	순수 YAML	YAML + Patch	템플릿 + values.yaml	CRD + Controller
템플릿 기능	✗ 없음	✗ 없음	✓ 있음 Go 템플릿	✓ 코드 기반 로직
중복 제거/재사용	✗	✓ overlay	✓ values 재사용	✓ 로직에 의한 처리
상태 관리	✗ 수동	✗ 수동	✗ 수동	✓ 자동 reconcile loop
설치 도구	kubectl apply -f	kubectl apply -k	helm install	OLM 또는 컨트롤러 배포
진입 장벽	매우 낮음	낮음	중간	높음
운영 자동화	✗	✗	✗ 일부 가능 hook	✓ 백업/복구/스케일링 등

# Chart 개발 실무

# Helm Chart 기본 구조

```
mychart/
├── Chart.yaml    차트 메타정보 (이름, 버전 등)
├── values.yaml   사용자 설정값 (기본값 정의)
├── templates/    실제 리소스
│   ├── deployment.yaml
│   ├── service.yaml
│   └── ...
│   └── _helpers.tpl 헬퍼 함수 (이름 정규화, 공통 label)
├── charts/       서브차트 (의존성)
└── README.md     차트 사용법 문서
```

- Helm은 이 구조를 기준으로 패키징 및 렌더링 수행
- charts/는 일반적으로 .gitignore 대상 필요시 helm dependency update



# Chart .yaml

```
# Chart.yaml
required
apiVersion: v2          차트 API 버전 (v2는 Helm 3 이상)
name: prometheus        차트 이름
version: 27.16.0        차트 버전
optional
appVersion: v3.4.0      앱 버전*
kubeVersion: ">=1.19.0-0"  지원되는 K8S 버전
description: Prometheus is... 차트 설명
type: application        application 또는 library
keywords: ["web", "nginx"] 검색용 키워드
dependencies:            의존성(서브차트)
- name: alertmanager
  version: "1.18.*"
  repository: https://prom...
  condition: alertmanager.enabled  서브차트 활성화 여부 변수화
```

```
# templates/deploy.yaml          앱 버전은 기본적으로는 정보성(참고용)이지만, 이미지 태그로 활용하는 경우가 많음
image: ...:{{ tpl .Values.server.image.tag . | default .Chart.AppVersion }}
```

<https://helm.sh/docs/topics/charts/#the-chartyaml-file>

<https://github.com/prometheus-community/helm-charts/blob/prometheus-27.16.0/charts/prometheus/chart.yaml>

# 차트 버전 정하기

```
# Chart.yaml Prometheus 차트
version: 27.16.0      차트 버전
appVersion: v3.4.0    (메인) 애플리케이션 버전
...
```

```
$ helm ls -A
```

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART	APP VERSION
cert-manager	cert-manager	1	2025-04-13 ...	deployed	cert-manager-v1.17.1	v1.17.1
in	ingress-nginx	49	2025-03-31 ...	deployed	ingress-nginx-4.12.1	1.12.1
prometheus	prometheus	1	2025-05-22 ...	deployed	prometheus-27.16.0	v3.4.0

차트 버전과 앱 버전 일치시킬 것인가?

- 대부분 불일치 일반적으로 차트 버전이 더 많음  
차트는 PR 올릴 때(변경사항 있을 때) 버전 up 관례
- 앱 버전이 올라가면, 차트 버전도 up
- 앱 버전이 변경 없이, 차트 버전만 up  
앱 버전만 올라가는 경우도 있을 수 있으나, 모범사례는 아님

v 접두어 사용할 것인가? 기본적으로 모두 허용되긴 함

- 차트 버전: 사용안함 권장 예: 27.16.0 (Helm 권고사항)
- 앱 버전: 사용 권장? 예: v3.4.0 (k8s 커뮤니티 관례. 단, 이미지 태그 고려)

# values .yaml

```
# values.yaml alertmanager 차트

image:
  repository: quay.io/prometheus/alertmanager
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default ...
  tag: ""
  ...

service:
  type: ClusterIP
  port: 9093
  ...

resources: {}
  # We usually recommend not to specify ...
  # limits:
  #   cpu: 100m
  #   memory: 128Mi
  # requests:
  #   ...
```

## 설계 원칙

계층 구조 사용 `image.repository`, `service.port`

일관된 키 네이밍

불필요한 중복 제거

Global values 사용 최소화

주석으로 의도와 사용법 설명

최소 설정으로 작동 가능한 기본값 제공

모듈별 enabled 고려 쉬운 on/off

`values.schema.json` 고려 validation

[https://helm.sh/docs/chart\\_best\\_practices/](https://helm.sh/docs/chart_best_practices/)



# values.schema.json

```
# values.schema.json
{
  "type": "object",
  "properties": {
    "replicaCount": {
      "type": "integer",
      "minimum": 1,
      "default": 2,
      "description": "Number of replicas",
      "examples": [1, 2, 3]
    }
  },
  "required": ["replicaCount"]
}
```

## 사용목적 한마디로 validation

- 값 누락, 타입 오류, 범위 오류 정적 검증
- 템플릿 렌더링 실패 방지
- 협업 시 기준 문서

## 알아두기

- 파일은 차트의 루트 디렉토리에 둔다.
- **helm lint** 명령어로 검사 수행 가능
- JSON Schema 스펙에 따른 것
- 단, 스펙에 default 키 있지만 Helm에서는 사용안함

helm 외 적용사례: OpenAPI(k8s CRD), VScode YAML언어서버 등

values.yaml에 작성해야 함

# 자주 쓰는 Helm 명령어 (개발자 편)

```
helm template .
```

```
helm lint
```

```
helm upgrade --install [RELEASE] .
```

```
helm test [RELEASE]
```

차트 렌더링 로직 문법 오류 확인

차트 유효성검사

설치하여 작동 확인

test pod 실행

정적 테스트 / 사전 검증 unit test 성격

동적 테스트 / 배포 후 검증 e2e test 성격

구분	helm template	helm lint
점검 항목	렌더링 오류.Values.xxx 참조 오류	Chart.yaml, 템플릿 문법, 필드 누락, 스키마 오류
YAML 출력	✅ 있음	❌ 없음

# helm test

```
# templates/tests/test-connection.yaml
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Release.Name }}-test-connection
  annotations:
    helm.sh/hook: test
spec:
  containers:
    - name: wget
      image: busybox
      command: ['wget', 'http://my-service:80']
  restartPolicy: Never
```

```
$ helm test my-release
```

## 사용목적

- 설치 후 작동 점검
- CI/CD에서 자동 검증 단계로 활용
- 예시 사례
  - HTTP 200 OK 응답 확인
  - DB 접속 확인

작성하기에 따라 다양한 점검 가능 (예: 쿼리 수행)

## 알아두기

- 테스트 리소스는 **templates/\*\*/tests/** 에 둔다.

보통 templates/tests/

- 어노테이션 **helm.sh/hook: test** 필수

helm.sh/hook-delete-policy 테스트 후 리소스 삭제

helm.sh/hook-weight 순서 지정

# **IV** GitHub Actions로 Chart 릴리스

# 차트를 어디에 둘까? 어느 git repo?

구분	1안) 앱 repo <small>애플리케이션 repo에 함께 둔다.</small>	2안) 차트 repo <small>별도의 차트 전용 repo에 둔다.</small>
장점	<ul style="list-style-type: none"> <li>• 앱 코드와 차트 변경 동시 관리</li> <li>• PR 하나로 전체 반영</li> <li>• 초기 설정 간단</li> </ul>	<ul style="list-style-type: none"> <li>• 여러 차트를 모아 집중 관리</li> <li>• <b>chart-releaser 활용 쉬움</b></li> <li>• <b>외부 배포에 적합</b> <small>ArtifactHub</small></li> </ul>
단점	<ul style="list-style-type: none"> <li>• chart-releaser 설정 복잡</li> <li>• gh-pages 브랜치 충돌 가능성</li> </ul>	<ul style="list-style-type: none"> <li>• 앱과 차트 버전 관리 분리</li> <li>• 릴리스 타이밍 동기화 필요 <small>CI/CD 구성 필요</small></li> </ul>
사례	<p>ingress-nginx</p> <p>→ 단일 애플리케이션 중심인 경우 고려해볼 수 있다.</p>	<p>prometheus 등 다양한 OSS 프로젝트</p> <p>→ git repo = 차트 repo (1:1 관계). 일반적으로 권장된다. ★</p>

# 차트 자동화 도구 *ct, cr, action*

## ct/cr 사용목적

- 여러 차트를 한곳에서 집중 관리
- 변경된 차트만 필터링하여 작업

*반면 helm lint/install/test는 단일 차트 대상*

## chart-testing (ct)

- 변경된 차트를 검증하는 도구 (CI)
- Lint / Install / Test 자동화

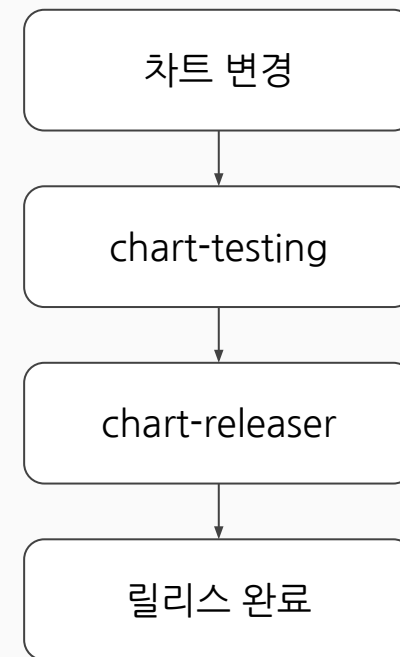
## chart-releaser (cr)

- 변경된 차트를 릴리스하는 도구 (CD)
- .tgz 패키징, index.yaml 생성, Helm Repo 배포

*gh-pages 브랜치를 Helm Repo로 활용*

## GitHub Actions

단계	트리거	GitHub Actions	관련 명령어	workflows 파일명 예시
검증	PR	chart-testing-action	ct lint, ct install	lint-test.yaml
릴리스	Merge	chart-releaser-action	cr package, cr index, cr upload	release.yaml



*함께 사용하면  
CI/CD 파이프라인*

# lint-test.yaml #1

<https://github.com/helm/chart-testing-action>

**name:** Lint and Test Charts

**on:** pull\_request PR 발생 시 자동 실행

**jobs:**

**lint-test:**

**runs-on:** ubuntu-latest 실행 환경 구성 - ubuntu 리눅스 환경에서 실행

**steps:**

- **uses:** actions/checkout@v3
- **uses:** azure/setup-helm@v4.2.0
- **uses:** actions/setup-python@v5.3.0
- **uses:** helm/chart-testing-action@v2.7.0
- ...

① 실행 환경 준비

git checkout 수행

도구 설치 #1 helm

도구 설치 #2 python (chart-testing에 필요)

도구 설치 #3 chart-testing (ct CLI)

# lint-test.yaml #2

- **name:** Run chart-testing (list-changed) ② 변경된 차트 필터링  
**id:** list-changed  
**run:** |  
    changed=\$(ct list-changed --target-branch \${ github.event.repository.default\_branch })  
    if [[ -n "\$changed" ]]; then  
        echo "changed=true" >> "\$GITHUB\_OUTPUT"    ② 변경된 차트만 필터링하여 다음 단계 실행 조건으로 사용  
    fi
- **name:** Run chart-testing (lint) ③ 차트 lint 검사 (유효성 검사)  
**if:** steps.list-changed.outputs.changed == 'true'  
**run:** ct lint --target-branch \${ github.event.repository.default\_branch }  
    ct lint = helm lint + 메타데이터, 버전 증가 여부 등 확인



# lint-test.yaml #3

- **name:** Create kind cluster      ④ kind 클러스터 생성  
**if:** steps.list-changed.outputs.changed == 'true'  
**uses:** helm/kind-action@v1.12.0
- **name:** Run chart-testing (install)      ⑤ 설치 테스트  
**if:** steps.list-changed.outputs.changed == 'true'  
**run:** ct install --target-branch \${ github.event.repository.default\_branch }

ct.yaml에 run-tests: true가 설정되어 있다면 helm test도 실행된다.

# release.yaml #1

<https://github.com/helm/chart-releaser-action>

```
name: Release Charts

on:
  push:
    branches:
      - main  main 브랜치에 변경사항이 push(PR merge 포함)될 때 실행

jobs:
  release:
    permissions:
      contents: write  릴리스 생성하므로 쓰기 권한 필요
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      ...
```

# release.yaml #2

- **name**: Configure Git

① GitHub Actions 환경에서 Git 커밋 작성자 정보 구성

**run**: |

git config user.name "\$GITHUB\_ACTOR"

git config user.email "\$GITHUB\_ACTOR@users.noreply.github.com"

- **name**: Run chart-releaser

② 릴리스 변경된 차트 감지 → .tgz 생성 → index.yaml 갱신

**uses**: helm/chart-releaser-action@v1.7

→ gh-pages 브랜치로 Helm Repo 구성

**env**:

**CR\_TOKEN**: "\${{ secrets.GITHUB\_TOKEN }}" GitHub 인증용 ( 기본적으로 secrets.GITHUB\_TOKEN 사용 )

잘 진행되었다면, Git Repo의 gh-pages 브랜치, Helm Repo (GitHub Pages) 확인

- gh-pages 브랜치: <https://github.com/kuoss/helm-charts/tree/gh-pages>

- Helm Repo: <https://kuoss.github.io/helm-charts/>

<https://kuoss.github.io/helm-charts/index.yaml>

# Helm 차트 배포 방식

지금까지 살펴본 방식은 chart-releaser 기반 배포 방식이다.

새로 도입된 OCI 레지스트리를 활용한 차트 배포 방식도 있다.

구분	chart-releaser 방식 전통적 Helm Repo	OCI 방식 OCI 레지스트리
저장소	GitHub Pages 정적 웹	OCI 레지스트리 예: GHCR, DockerHub, Harbor
배포 파일	.tgz + index.yaml	.tgz index.yaml 없음
URL 형식	https://user.github.io/repo	oci://ghcr.io/org/chart-name
명령어	cr package, cr upload 등	helm push, helm pull
인증	GitHub Token (GITHUB_TOKEN)	OCI 인증 (helm registry login)
검색	브라우저 가능, index.yaml 기반	브라우저 불가, 주소 직접 입력

“전통적”이라고는 했지만 기존 Helm Repo 방식은 앞으로도 계속 지원된다.

대부분의 오픈소스 프로젝트들이 기존 방식을 계속 사용하고 있다.

OCI (Open Container Initiative)

컨테이너 이미지 및 아티팩트를 표준 형식으로 저장할 수 있는 공통 인터페이스

OCI 차트설치 예시

```
helm registry login ghcr.io (필요 시) 로그인 — 프라이빗 레지스트리인 경우
helm install my-release oci://ghcr.io/my-org/my-chart --version 1.2.3 차트 설치
```

# ArtifactHub 등록

# ArtifactHub

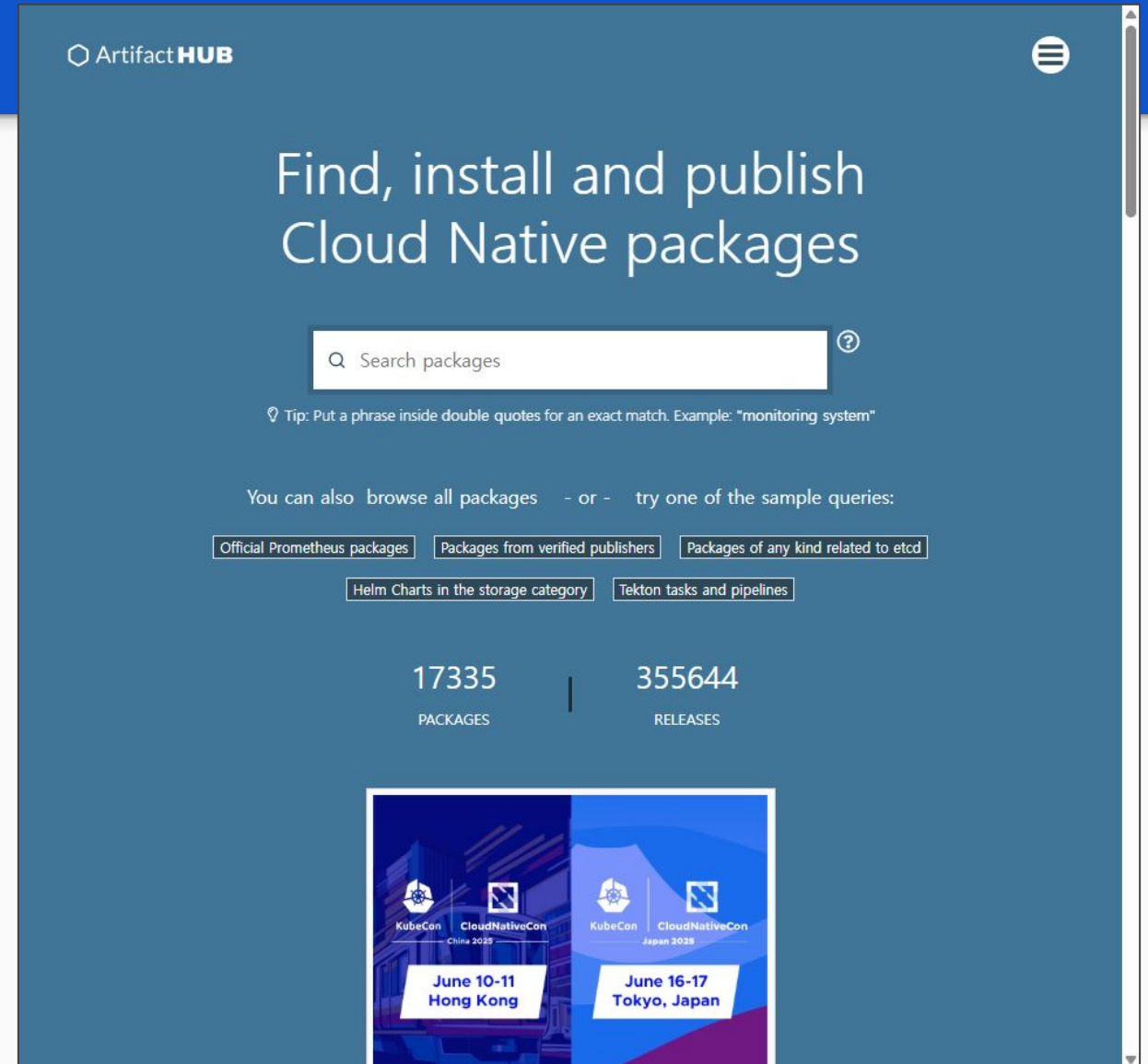
## ArtifactHub

- <https://artifacthub.io>
- CNCF에서 운영하는 클라우드 네이티브 패키지 허브
- 다양한 패키지를 하나의 통합된 플랫폼에서 검색
- 지원 패키지 20종 이상

Helm 차트, OLM 오퍼레이터, 컨테이너 이미지, OPA 정책, Falco 규칙,

kubectl 플러그인(Krew), Argo 템플릿, Backstage 플러그인 등

<https://github.com/artifacthub/hub>



1 - 20 of 634 results for "prometheus"

Sort: Relevance

Show: 20



## FILTERS

☐ Official ⓘ☐ Verified publishers ⓘ☐ CNCF ⓘ

## KIND

☐ Containers images (20)☐ Headlamp plugins (1)☐ Helm charts (587)☐ Helm plugins (1)☐ KCL modules (2)☐ Keptn integrations (2)☐ Krew kubectrl plugins (1)☐ Kyverno policies (1)☐ Meshery designs (12)☐ OLM operators (5)☐ Tekton tasks (2)

## CATEGORY

☐ Database (12)☐ Integration and delivery (1)☐ Monitoring and logging (360)☐ Not a category

## prometheus

Prometheus prometheus-community

★ 474

Helm chart

Updated 2 days ago

Version 27.16.0

Prometheus is a monitoring system and time series database.

Monitoring and logging



## Prometheus Operator

Operator Framework Community Operators

★ 35

OLM operator

Updated a year ago

Version 0.70.0

Manage the full lifecycle of configuring and managing Prometheus and Alertmanager servers.

Monitoring and logging



## prometheus

Bitnami Bitnami

★ 11

Helm chart

Updated 2 days ago

Version 2.0.8

Prometheus is an open source monitoring and alerting system. It enables sysadmins to monitor their infrastructures by collecting metrics from conf...

Monitoring and logging





prometheus

Helm chart

Monitoring and logging

Prometheus

prometheus-community

Prometheus is a monitoring system and time series database.



SUBSCRIPTIONS: 168 WEBHOOKS: 17 PRODUCTION USERS: 7

☆ Star

474



## Prometheus

Prometheus, a [Cloud Native Computing Foundation](#) project, is a systems and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true.

This chart bootstraps a [Prometheus](#) deployment on a [Kubernetes](#) cluster using the [Helm](#) package manager.

## Prerequisites

- Kubernetes 1.19+
- Helm 3.7+

## Get Repository Info

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

See [helm repository](#) for command documentation.

## Install Chart

↓ INSTALL

TEMPLATES

DEFAULT VALUES

VALUES SCHEMA

CHANGELOG



APPLICATION VERSION

v3.4.0





venti-stack

Helm chart

Monitoring and logging

kuoss kuoss

Star

3



Venti stack is a logging and monitoring system.



SUBSCRIPTIONS: 2 PRODUCTION USERS: 1

## venti-stack

Artifact Hub

venti-stack

venti

Installs the venti-stack, a collection of Kubernetes manifests to provide easy to operate end-to-end Kubernetes cluster logging, monitoring and visualizing.

### Prerequisites

- Kubernetes 1.16+
- Helm 3+

### Get Helm Repository Info

```
helm repo add kuoss https://kuoss.github.io/helm-charts
helm repo update
```

See `helm repo` for command documentation.

### Install Helm Chart

INSTALL

TEMPLATES

DEFAULT VALUES

VALUES SCHEMA

CHANGELOG



APPLICATION VERSION

v0.2.24

# ArtifactHub에 Chart 등록

## 전제조건

Helm Repo에 공개 접근가능해야 함

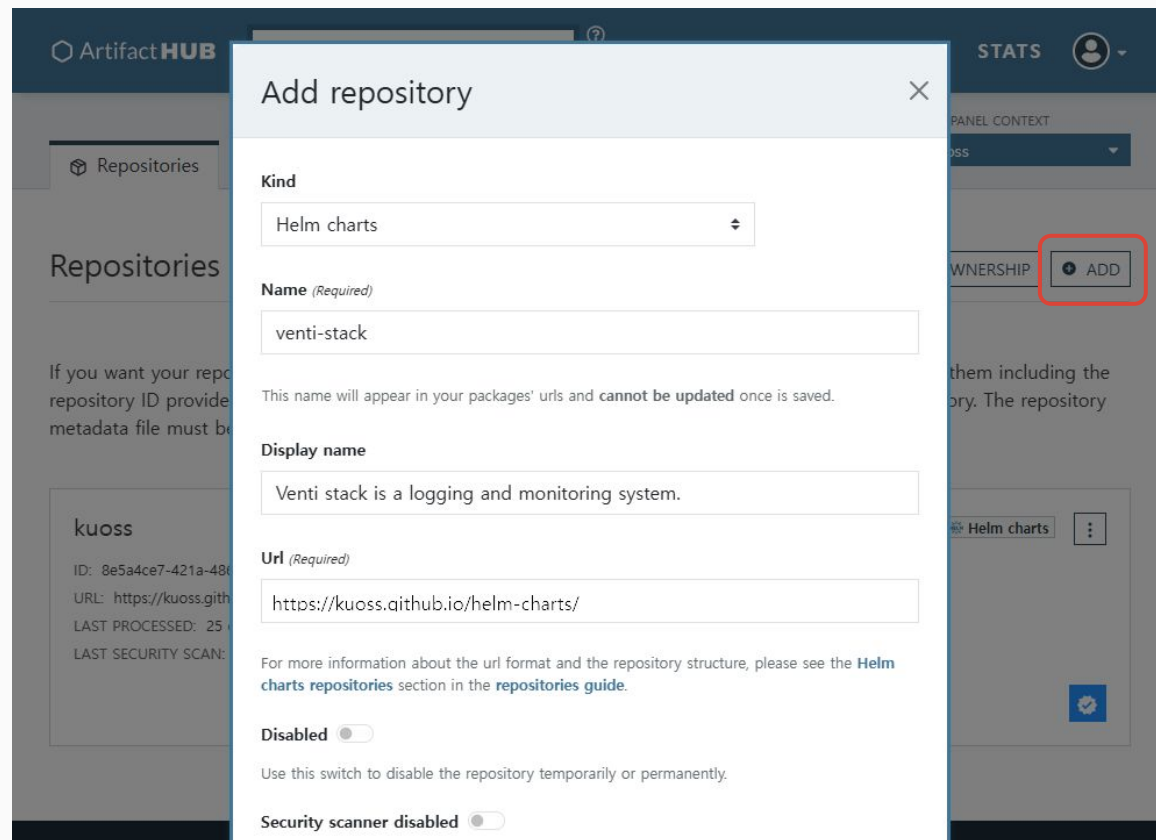
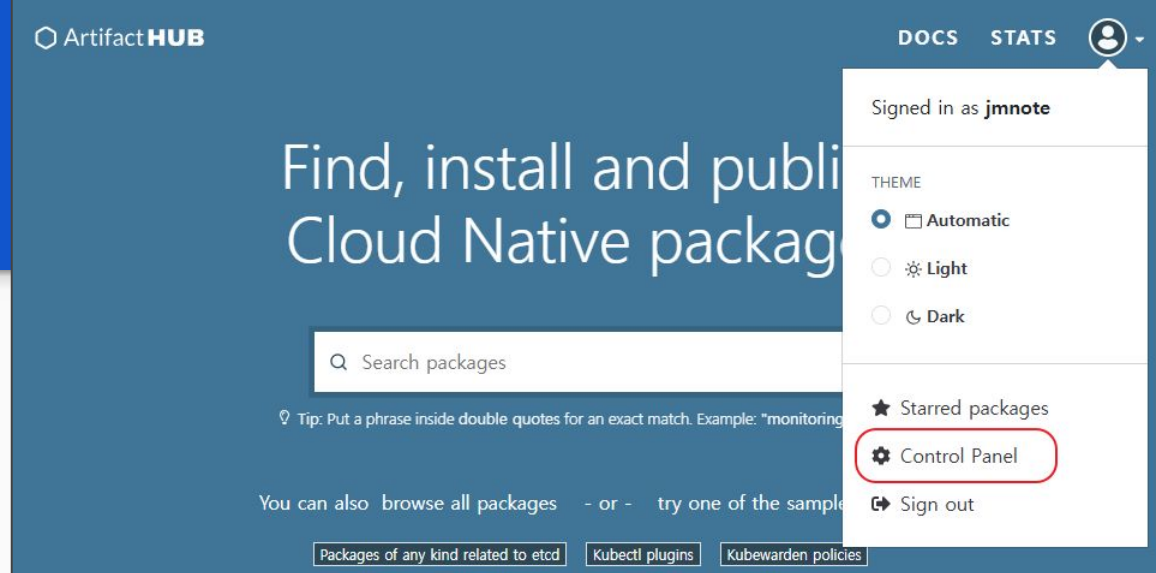
Helm Repo (GitHub Pages), Git Repo의 gh-pages 브랜치 확인

## 등록방법

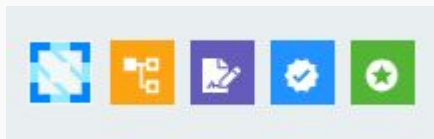
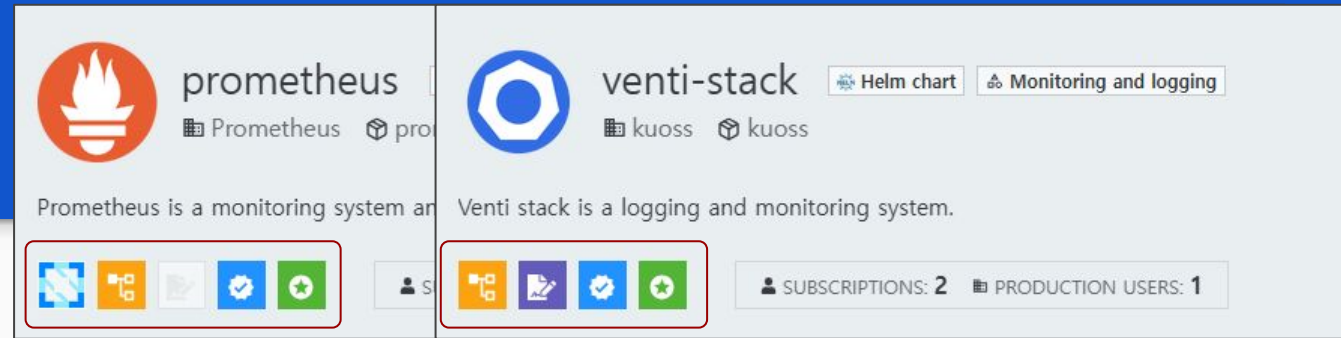
- artifacthub.io 로그인 GitHub 계정 권장
- Control Panel → Repositories → [ADD]
  - Kind: Helm charts 선택
  - Name: venti-stack
  - URL: <https://kuoss.github.io/helm-charts/>

※ 최초 등록 이후 해당 Repo의 신규 차트 및 버전은 자동 등록된다.

<https://artifacthub.io/docs/topics/repositories/helm-charts/>



# 배지 5종



배지 이름	설명
CNCF	CNCF 프로젝트가 배포한 패키지입니다.
Values schema	이 차트는 <b>values.schema.json</b> 파일을 제공하여 값의 유효성을 정의합니다.
Signed	이 패키지는 Helm signing 기능으로 서명되었습니다.
Verified publisher	신뢰할 수 있는 게시자로 검증된 소스에서 제공된 패키지입니다.
Official	공식적으로 인정된 패키지로, 해당 프로젝트 또는 조직에서 직접 유지 관리합니다.

# Signed 뱃지 얻기

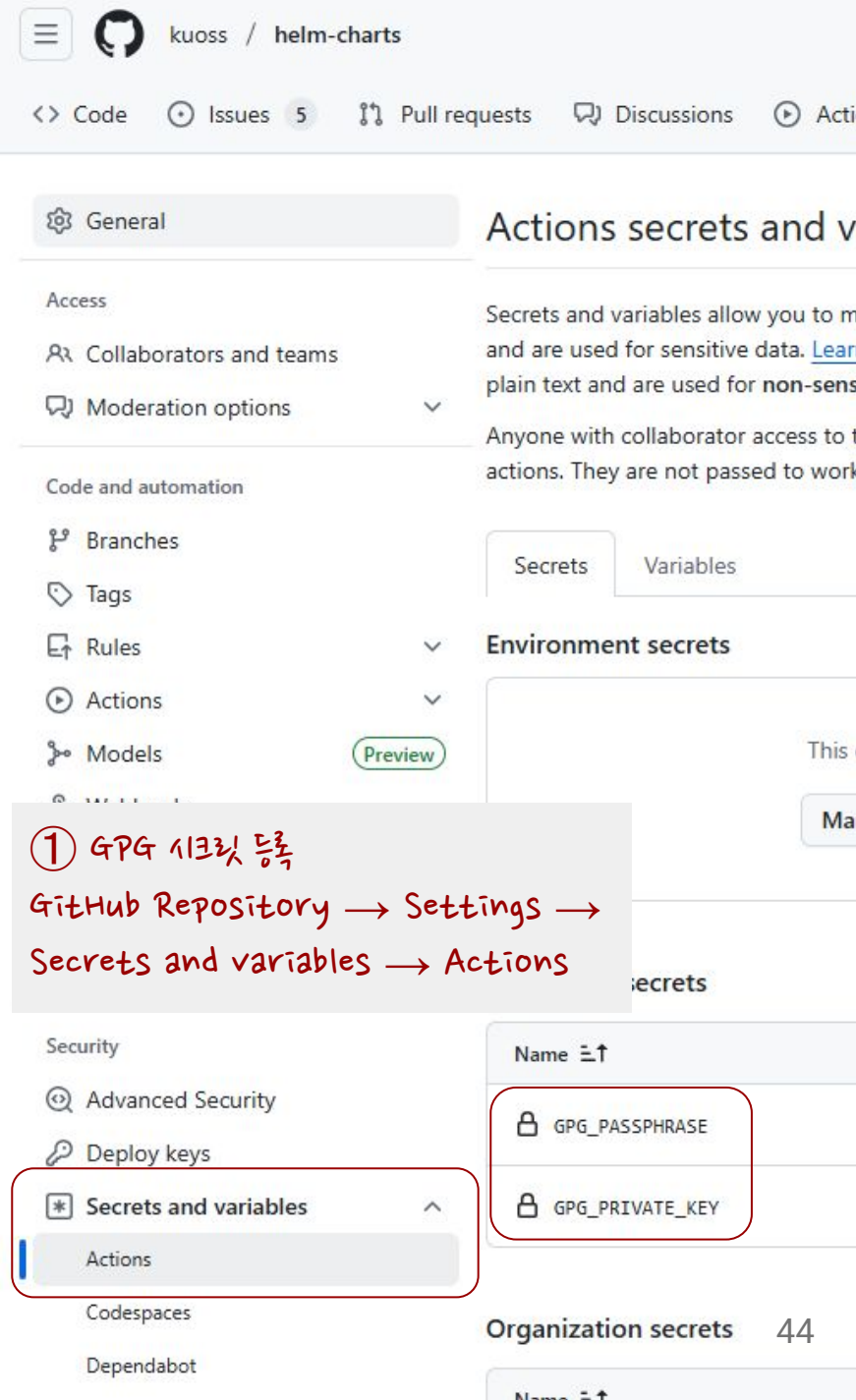
참고) 컨테이너 서명에는 흔히 Dockerfile에 서명 작업을 추가하며, 차트 서명과 달라 Signed 뱃지 관련 없음

## ② .github/workflows/release.yaml 수정

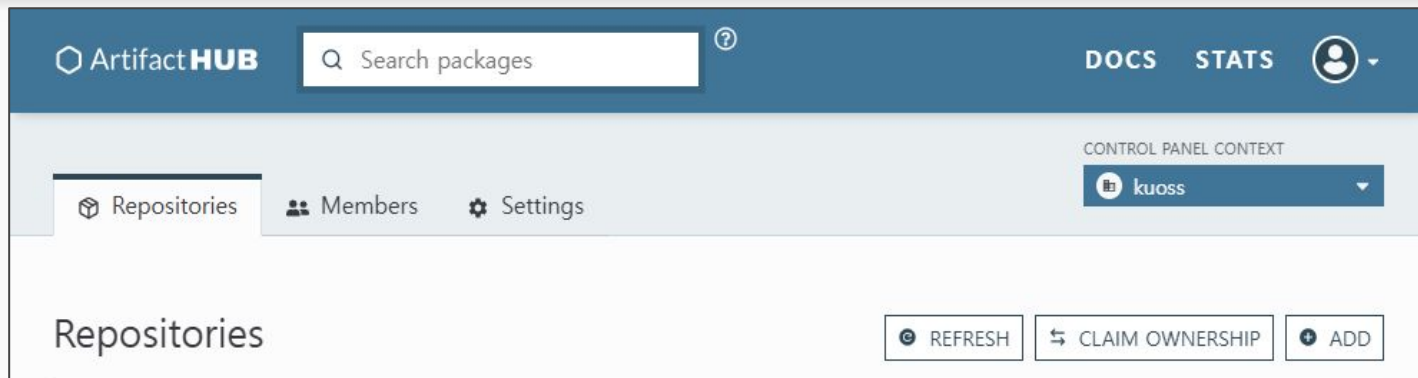
- **name:** Import GPG key  
**env:**  
GPG\_PRIVATE\_KEY: "\${{ secrets.GPG\_PRIVATE\_KEY }}"  
GPG\_PASSPHRASE: "\${{ secrets.GPG\_PASSPHRASE }}"  
**run:** |  
echo "\$GPG\_PRIVATE\_KEY" | gpg --dearmor --output keyring.gpg  
echo "\$GPG\_PASSPHRASE" > passphrase-file.txt
- **name:** Run chart-releaser  
**uses:** helm/chart-releaser-action@v1.7  
**env:**  
CR\_TOKEN: "\${{ secrets.GITHUB\_TOKEN }}"  
CR\_KEY: jmnote <opcore@gmail.com>  
CR\_KEYRING: keyring.gpg  
CR\_PASSPHRASE\_FILE: passphrase-file.txt  
CR\_SIGN: true

이외에 crazy-max/ghaction-import-gpg를 사용하는 방법도 있음

<https://github.com/artifactory/hub/blob/master/.github/workflows/release.yaml> 참고



# Verified publisher 뱃지 얻기



If you want your repositories to be labeled as **Verified Publisher**, you need to provide a repository ID provided below. This label will let users know that your metadata file must be located at the path used in the repository URL.

① ArtifactHub에서 repositoryID 확인

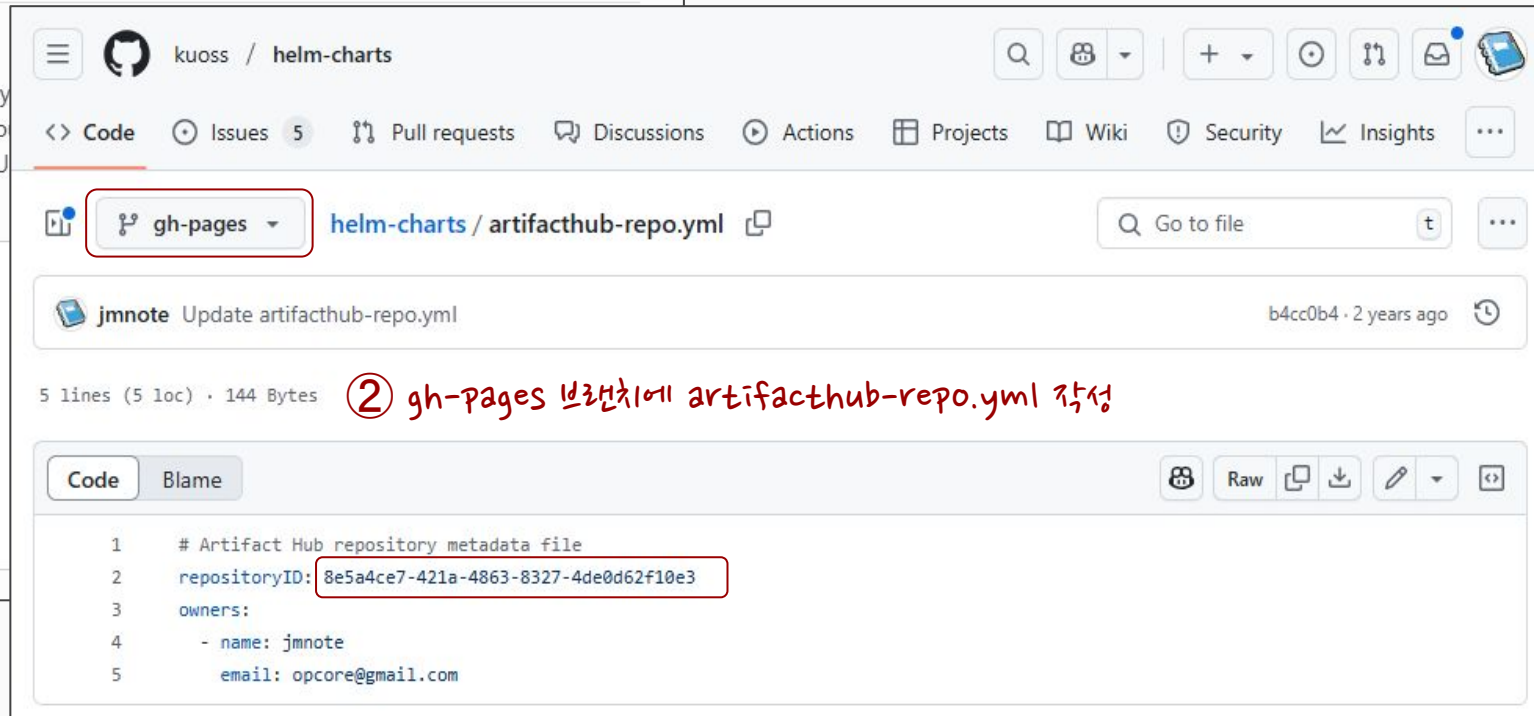
kuoss

ID: **8e5a4ce7-421a-4863-8327-4de0d62f10e3**

URL: <https://kuoss.github.io/helm-charts>

LAST PROCESSED: 25 days ago ✓ (next check in ~ 26 minutes)

LAST SECURITY SCAN: 4 hours ago ✓ (next scan in ~ 11 minutes)





# Official 뱃지 얻기

## 조건

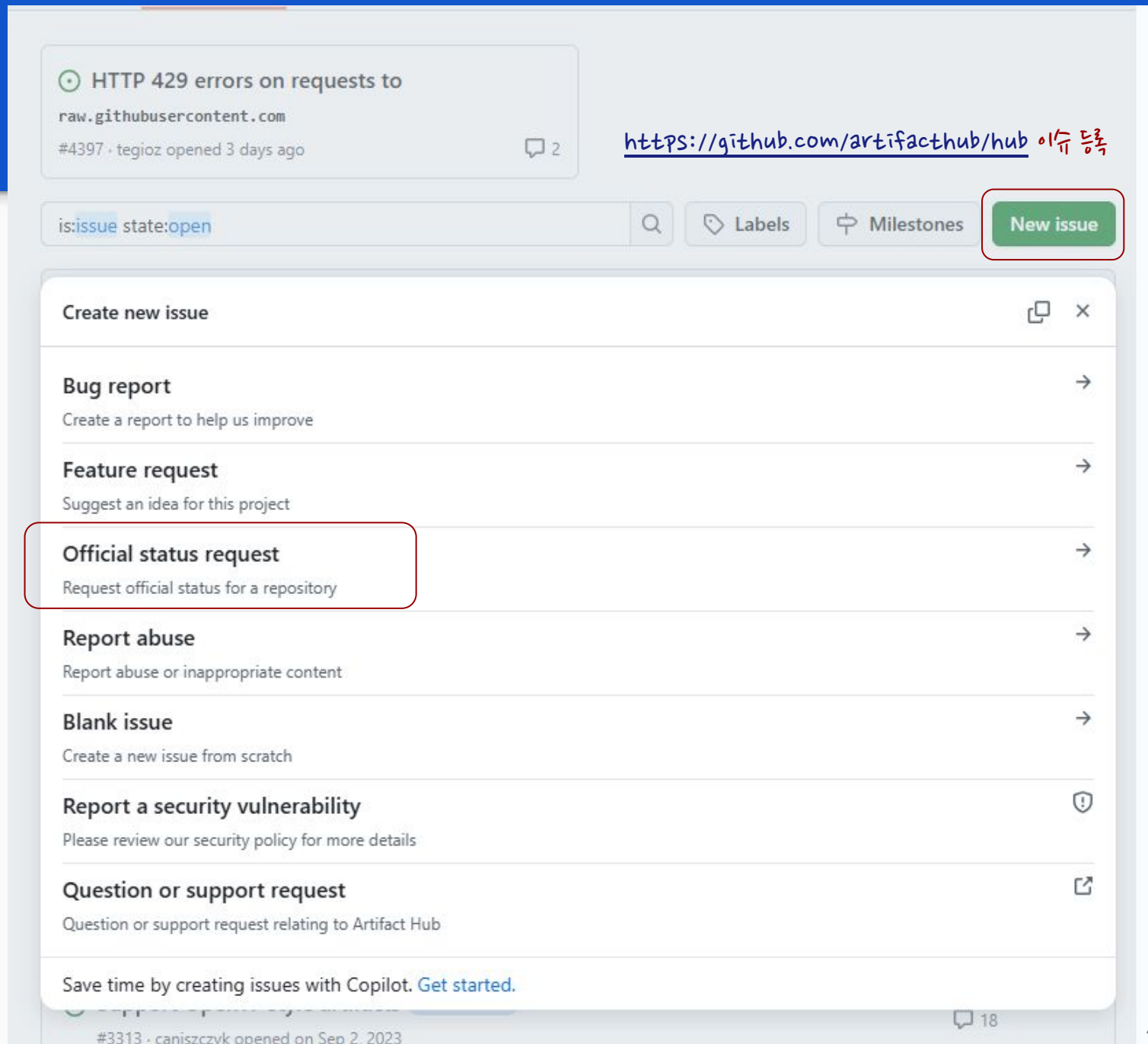
- 소프트웨어의 소유자여야 함


헬름 차트의 메인 어플리케이션을 소유한 사람이어야 함

- Verified publisher 뱃지를 먼저 획득해야 함
- 공식 패키지 모두에 README.md 필수


<https://artifacthub.io/docs/topics>

[/repositories/#official-status](https://artifacthub.io/docs/topics/repositories/#official-status)





DOCSSTATSSIGN UPSIGN IN



# prometheus

Prometheus is a monitoring system and time series database.

[Helm chart](#) [Monitoring and logging](#)

Star 474

## Prometheus

Prometheus, a Cloud Native Computing Foundation project, is a monitoring system and time series database. It targets at given intervals, evaluates rules, and stores the data in a time series database.

This chart bootstraps a Prometheus deployment on a Kubernetes cluster.

### Prerequisites

- Kubernetes 1.19+
- Helm 3.7+

### Get Repository Info

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo update
```

See [helm repository](#) for command documentation.

### Install Chart

## prometheus

### Helm v3

Add repository

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

Install chart

```
helm install my-prometheus prometheus-community/prometheus --version 27.16.0
```

*my-prometheus corresponds to the release name, feel free to change it to suit your needs. You can also add additional flags to the `helm install` command if you need to.*

You can also download this package's content using [this link](#).

Need Helm?

[CLOSE](#)


### INSTALL

### TEMPLATES

### DEFAULT VALUES

### VALUES SCHEMA

### CHANGELOG



APPLICATION VERSION  
v3.4.0

## Templates

Compare to version ...



Search by template or resou... 🔍

This chart version contains 18 templates

TEMPLATE: clusterrole.yaml

RESOURCE: ClusterRole

TEMPLATE: clusterrolebinding.yaml

RESOURCE: ClusterRoleBinding

TEMPLATE: cm.yaml

RESOURCE: ConfigMap

TEMPLATE: deploy.yaml

RESOURCE: Deployment

TEMPLATE: extra-manifests.yaml

RESOURCE: -

TEMPLATE: headless-svc.yaml

RESOURCE: Service

TEMPLATE: httproute.yaml

RESOURCE: -

```
1  {{- if and .Values.rbac.create (empty .Values.server.useExistingClusterRoleName) -}}
2  apiVersion: {{ template "rbac.apiVersion" . }}
3  kind: ClusterRole
4  metadata:
5    labels:
6      {{- include "prometheus.server.labels" . | nindent 4 }}
7    name: {{ include "prometheus.clusterRoleName" . }}
8  rules:
9    {{- if and .Values.podSecurityPolicy.enabled (.Capabilities.APIVersions.Has "policy/v1beta1/PodSecurityPolicy") }}
10   - apiGroups:
11     - extensions
12     resources:
13     - podsecuritypolicies
14     verbs:
15     - use
16     resourceNames:
17     - {{ template "prometheus.server.fullname" . }}
18   {{- end }}
19   - apiGroups:
20     - ""
21     resources:
```



TIP: some extra info may be displayed when hovering over values entries and other built-in objects and functions.

CLOSE





Prometheus is



Prom

Prometheus, a  
targets at give

This chart boo

Prerequisite

- Kubernetes
- Helm 3.7-

Get Repo

helm repo a  
helm repo u

See helm repo

Install Chart

## Default values

Compare to version ...



Search path



```
1 # yaml-language-server: $schema=values.schema.json
2 # Default values for prometheus.
3 # This is a YAML-formatted file.
4 # Declare variables to be passed into your templates.
5
6 rbac:
7   create: true
8
9 podSecurityPolicy:
10   enabled: false
11
12 imagePullSecrets: []
13 # - name: "image-pull-secret"
14
15 ## Define serviceAccount names for components. Defaults to component's fully qualified name.
16 ##
17 serviceAccounts:
18   server:
19     create: true
```



X CLOSE

APPLICATION VERSION  
v3.4.0



Prometheus is



Prom

Prometheus, a  
targets at give

This chart boo

Prerequisite

- Kubernetes
- Helm 3.7-

Get Repo

helm repo a  
helm repo up

See helm repo

Install Chart

APPLICATION VERSION  
v3.4.0

## Values schema reference



Search path



rbac:

create:

server:

env:

name:

tsdb:

image:

tag:

digest:

pullPolicy:

repository:

global:

scrape\_timeout:

scrape\_interval:

evaluation\_interval:

▶ TYPE: object

▶ TYPE: boolean

▶ TYPE: object

▶ TYPE: array

▶ TYPE: string

▶ TYPE: object

▶ TYPE: object

▶ TYPE: string

▶ TYPE: string

▶ TYPE: string

▶ TYPE: string

▶ TYPE: object

▶ TYPE: string

▶ TYPE: string

▶ TYPE: string

× CLOSE

&lt; Back to "ingress-nginx" results

NGINX

ingress-nginx

Kubernetes

ingr

Ingress controller for Kubernetes using



SUBSCR



ingress-nginx

ingress-nginx Ingress controller for Ku

Version

4.12.2

AppVersion

1.12.2

To use, add `ingressClassName: nginx`

This chart bootstraps an ingress-nginx

## Requirements

Kubernetes: `>=1.21.0-0`

## Get Repo Info

`helm repo add ingress-nginx https:``helm repo update`

## Install Chart

## Changelog

차트에 changelog가 포함된 경우에만 나옴  
( prometheus 차트에는 없음 )

4.12.2  
30 Apr, 2025

4.12.2

Released 21 days ago

- Update Ingress-Nginx version controller-v1.12.2

4.12.1  
25 Mar, 2025

4.12.1

Released 2 months ago

- Update Ingress-Nginx version controller-v1.12.1

4.12.0  
31 Dec, 2024

4.12.0

Released 5 months ago

- CI: Fix chart testing. (#12258)
- Update Ingress-Nginx version controller-v1.12.0

4.12.0-beta.0  
26 Nov, 2024

4.12.0-beta.0

Pre-release

Released 6 months ago

- Update Ingress-Nginx version controller-v1.12.0-beta.0

4.11.6  
1 May, 2025

4.11.6

Released 21 days ago

- Update Ingress-Nginx version controller-v1.11.6

4.11.5  
25 Mar, 20254.11.4  
31 Dec, 20244.11.3  
9 Oct, 20244.11.2  
16 Aug, 20244.11.1  
19 Jul, 2024

4.11.0

GET MARKDOWN

CLOSE

☆ Star

764



INSTALL

TEMPLATES

DEFAULT VALUES

CHANGELOG



APPLICATION VERSION

1.12.2

kubernetes / ingress-nginx

Q Type / to search

<> Code Issues 342 Pull requests 78 Actions Projects Security Insights

Files

main

Go to file

.github

build

changelog

charts/ingress-nginx

changelog

helm-chart-2.10.0.md

helm-chart-2.11.0.md

helm-chart-2.11.1.md

helm-chart-2.11.2.md

helm-chart-2.11.3.md

helm-chart-2.12.0.md

helm-chart-2.12.1.md

helm-chart-2.13.0.md

helm-chart-2.14.0.md

helm-chart-2.15.0.md

helm-chart-2.16.0.md

helm-chart-2.9.0.md

helm-chart-2.9.1.md

helm-chart-3.0.0.md

helm-chart-3.10.0.md

ingress-nginx / charts / ingress-nginx / changelog /

Add file

...

Gacko Release controller v1.12.2/v1.11.6 & chart v4.12.2/v4.11.6. (#13318) ✓

1489f51 · 3 weeks ago History

Name	Last commit message	Last commit date
..		
helm-chart-2.10.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.11.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.11.1.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.11.2.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.11.3.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.12.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.12.1.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.13.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.14.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.15.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.16.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.9.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-2.9.1.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-3.0.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago
helm-chart-3.10.0.md	Chart: Split CHANGELOG.md into changelog/helm-chart-*.md.	2 years ago



Starting with version 16.0, the Prometheus chart requires Helm 3.7+ in order to install successfully. Please check your `helm` release before installation.

```
helm install [RELEASE_NAME] prometheus-community/prometheus
```

See [configuration below](#).

See [helm install](#) for command documentation.

## Dependencies

By default this chart installs additional, dependent charts:

- [alertmanager](#)
- [kube-state-metrics](#)
- [prometheus-node-exporter](#)
- [prometheus-pushgateway](#)

To disable the dependency during installation, set `alertmanager.enabled`, `kube-state-metrics.enabled`, `prometheus-node-exporter.enabled` and `prometheus-pushgateway.enabled` to `false`.

See [helm dependency](#) for command documentation.

## Uninstall Chart

```
helm uninstall [RELEASE_NAME]
```

This removes all the Kubernetes components associated with the chart and deletes the release.

See [helm uninstall](#) for command documentation.

## Updating values.schema.json

A `values.schema.json` file has been added to validate chart values. When `values.schema.json` file has a structure change (i.e. add a new field

### CHART VERSIONS

27.16.0 (20 May, 2025)  
27.15.0 (20 May, 2025)  
27.14.0 (20 May, 2025)

[See all](#) (485)

### LAST YEAR ACTIVITY



### LAST 30 DAYS VIEWS



[See details](#) (all versions)

### TYPE

application

### KUBERNETES VERSION

>= 1.19.0-0

### SECURITY REPORT

17 vulnerabilities found

HIGH 3  
 MEDIUM 14

LAST SCAN: 17 hours ago

[OPEN FULL REPORT](#)

### LINKS

- [Homepage](#)
- [Source](#)
- [Source](#)
- [Source](#)
- [Source](#)

See `helm repository` for command documentation.

## Install Chart

Starting with  
installation.

`helm inst`

See `configu`

See `helm in`

## Depend

By default t

- alertma
- kube-st
- promet
- promet

To disable t

`prometheus-`

See `helm de`

## Uninsta

`helm unin`

This remove

See `helm uninstall` for command documentation.

Updating values schema is on

APPLICATION VERSION

### Security report

#### SUMMARY

Image	Rating	Critical	High	Medium	Low	Unknown	Total
quay.io/prometheus/prometheus:v3.4.0	A	0	0	0	0	0	0
quay.io/prometheus/pushgateway:v1.11.1	C	0	0	2	0	0	2
quay.io/prometheus/alertmanager:v0.28.1	D	0	2	6	0	0	8
quay.io/prometheus/node-exporter:v1.9.1	C	0	0	2	0	0	2
registry.k8s.io/kube-state-metrics/kube-stat...	D	0	1	4	0	0	5
quay.io/prometheus-operator/prometheus-...	A	0	0	0	0	0	0

**17** vulnerabilities (**5** unique) have been detected in this package's **images**.

UNIQUE VULNERABILITIES (5)



× CLOSE

★ Homepage

🔗 Source

🔗 Source

🔗

May '25

22 May  
(all versions)

3  
14

Starting with version 16.0, the Prometheus chart requires Helm 3.7+ in order to install successfully. Please check your `helm` release before installation.

## Security report

### VULNERABILITIES DETAILS

IMAGE: `quay.io/prometheus/prometheus:v3.4.0`

▶ TARGET: `bin/prometheus` RATING: A

▶ TARGET: `bin/promtool` RATING: A

IMAGE: `quay.io/prometheus/pushgateway:v1.11.1`

▶ TARGET: `bin/pushgateway` RATING: C [Show vulnerabilities](#)

IMAGE: `quay.io/prometheus/alertmanager:v0.28.1`

▶ TARGET: `bin/alertmanager` RATING: D [Show vulnerabilities](#)

▶ TARGET: `bin/amttool` RATING: D [Show vulnerabilities](#)

IMAGE: `quay.io/prometheus/node-exporter:v1.9.1`

▶ TARGET: `bin/node_exporter` RATING: C [Show vulnerabilities](#)

IMAGE: `registry.k8s.io/kube-state-metrics/kube-state-metrics:v2.15.0`

▶ TARGET: `debian 12.9` RATING: A

▶ TARGET: `kube-state-metrics` RATING: D [Show vulnerabilities](#)

IMAGE: `quay.io/prometheus-operator/prometheus-config-reloader:v0.82.2`

× CLOSE

## Dependencies

By default the

- alertmanager
- kube-state-metrics
- prometheus
- prometheus-operator

To disable the

`prometheus-`

See `helm docs`

## Uninstall

`helm uninstall`

This removes

See `helm uninstall`

## Updating values.schema.json

A `values.schema.json` file has been added to validate chart values. When `values.yaml` file has a structure change (i.e. add a new field,

Source

Source

Upstream Project

Starting with version 16.0, the Prometheus chart requires Helm 3.7+ in order to install successfully. Please check your `helm` release before installation.

## Security report

📦 IMAGE: **quay.io/prometheus/alertmanager:v0.28.1**

▶ TARGET: `bin/alertmanager` RATING: D [Show vulnerabilities](#)

▼ TARGET: `bin/amtool` RATING: D [Hide vulnerabilities](#)

Critical	High	Medium	Low	Unknown	Total
0	1	3	0	0	4

ID	SEVERITY	PACKAGE	VERSION	FIXED IN
▶ CVE-2025-22869 <a href="#">🔗</a>	<span style="color: red;">HIGH</span>	golang.org/x/crypto	v0.31.0	0.35.0
▶ CVE-2025-22872 <a href="#">🔗</a>	<span style="color: orange;">MEDIUM</span>	golang.org/x/net	v0.33.0	0.38.0
▶ CVE-2025-22871 <a href="#">🔗</a>	<span style="color: orange;">MEDIUM</span>	stdlib	1.23.7	1.23.8, 1.24.2
▶ CVE-2025-22870 <a href="#">🔗</a>	<span style="color: orange;">MEDIUM</span>	golang.org/x/net	v0.33.0	0.36.0

🔗 IMAGE: **quay.io/prometheus/node-exporter:v1.9.1**

▶ TARGET: `bin/node_exporter` RATING: C [Show vulnerabilities](#)

📦 IMAGE: **registry.k8s.io/kube-state-metrics/kube-state-metrics:v2.15.0**

▶ TARGET: `debian 12.9` RATING: A

▶ TARGET: `kube-state-metrics` RATING: D [Show vulnerabilities](#)

× CLOSE

## Dependencies

By default the

- alertmanager
- kube-state-metrics
- prometheus
- prometheus

To disable the

prometheus-

See helm de

## Uninstall

helm unin

This remove

See helm un

## Updating values.schema.json

A `values.schema.json` file has been added to validate chart values. When `values.yaml` file has a structure change (i.e. add a new field,

🔗 Source

🔗 Source

🔗 Upstream Project



# 보안 보고서 기능

- 패키지에서 사용하는 컨테이너 이미지 취약점 자동 분석
- 분석 도구: Trivy
- 스캔 주기
  - 신규 버전: 즉시 또는 30분 이내 스캔
  - 최신 버전: 매일 스캔
  - 이전 버전: 주 1회 재스캔

→ 시간이 지나면 기존의 보안등급이 바뀔 수 있음
- 1년 이상 경과 버전, 이미지 15개 초과시 스캔 제외

Go로 빌드된 바이너리는 어떻게 점검할 수 있었을까?

→ 바이너리에 사용된 모듈 정보가 들어있음

```
$ go version -m /usr/local/bin/kubectl
```

IMAGE: registry.k8s.io/kube-state-metrics/kube-state-metrics:v2.15.0

TARGET: debian 12.9 RATING: A

TARGET: kube-state-metrics RATING: D [Show vulnerabilities](#)

## Trivy

- Aqua Security에서 개발한 오픈소스 취약점 스캐너
- 컨테이너 이미지, 파일시스템 등 스캔 가능
- OS 패키지, 언어별 라이브러리 취약점 탐지

## 보안 강화 전략

### 1. Base 이미지 보안 강화

- 취약한 베이스 이미지 사용금지
- scratch, distroless 이미지로 대체

### 2. OS 패키지 최소화 불필요한 패키지 설치 지양

### 3. Go 의존성 관리 강화

- 취약한 Go 모듈 패치
- govulncheck, GitHub의 취약점 알림 활용 dependabot 등

If NetworkPolicy is enabled for Prometheus' scrape targets, you may also need to manually create a networkpolicy which allows it.

## Views over the last 30 days

30일간 차트 조회 수

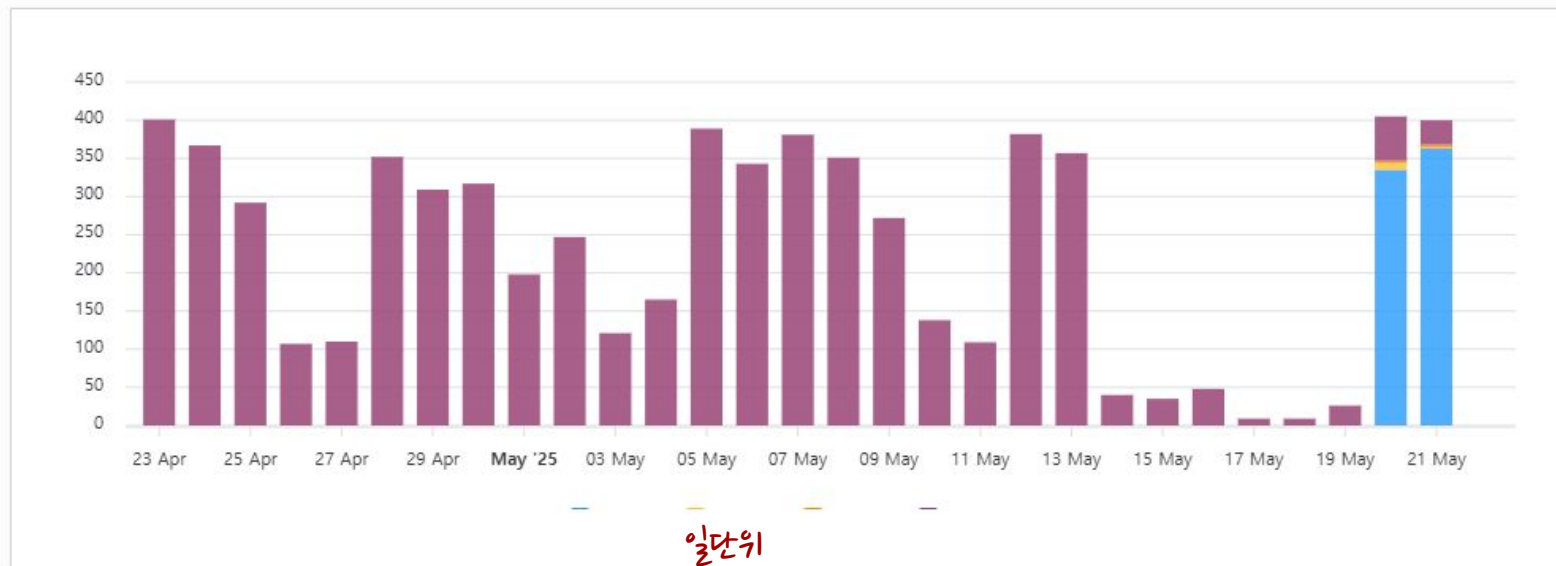


차트 버전별로  
색상 다름

### PROJECT

[Documentation](#)

[Blog](#)

[Statistics](#)

### COMMUNITY

[GitHub](#)

[Slack](#)

[Twitter](#)

### ABOUT

Artifact Hub is an

**Open Source**

project licensed under the  
Apache License 2.0



Copyright © The Artifact Hub Authors

# EOF

## 요약

- Helm 작동 원리, 다른 도구와 비교
- 차트 개발, 테스트, 릴리스
- ArtifactHub 사용 및 차트 등록

## What's next 기회가 된다면...

- ingress-annotator 소개
- lethe 로깅 성능 테스트