

목차

- **Monithub 소개**
 - 1. Monithub가 해결하려는 문제
 - 2. 관측에서 대응까지 이어지는 하나의 흐름
 - 3. 누구를 위한 제품인가요?
 - 4. 먼저 알아두면 좋은 핵심 개념
 - 5. 주요 기능을 사용자 관점에서 살펴보기
 - 6. 실제 상황에서는 어떻게 사용하나요?
 - 7. 사용할 때 기억할 점
 - 8. 역할에 맞게 시작하기
- **제품 사용 가이드**
 - 기본 사용 흐름
 - 메뉴 한눈에 보기
 - 먼저 알아둘 것
- **가입과 조직 참여**
 - 새 조직을 만들면서 가입하기
 - 기존 조직에 참여 요청하기
 - 초대 메일로 참여하기
 - 초대가 진행되지 않을 때
- **멤버 초대와 참여 요청**
 - 조직의 멤버 화면 열기
 - 이메일로 멤버 초대하기
 - 참여 요청 확인하기
 - 요청이 보이지 않거나 처리할 수 없을 때
- **홈과 기본 탐색**
 - 상단 바
 - 인시던트 활동
 - 시스템 상태
 - 좌측 메뉴
 - 홈에서 바로 판단할 것
- **대시보드**

- 대시보드 선택
- 필터와 시간 범위
- 패널 읽기
- 이상탐지
- 편집 기능
- **패널 편집**
 - 진입 방법
 - 저장과 취소
 - 타입 선택
 - Data 탭
 - Fields 탭
 - Axes & Legend
 - 타입별 문서
 - Panel & Layout
 - JSON
 - Row와 Text 패널
 - 추천 편집 순서
- **Time Series 패널**
 - Data
 - Fields
 - Visualization
 - 확인 순서
- **Bar 패널**
 - Data
 - Fields
 - Visualization
 - 확인 순서
- **Stat 패널**
 - Data
 - Fields
 - Visualization

- 확인 순서
- Gauge 패널
 - Data
 - Fields
 - Visualization
 - 확인 순서
- Table 패널
 - Data
 - Fields
 - Visualization
 - 확인 순서
- Heatmap 패널
 - Data
 - Fields
 - Visualization
 - 확인 순서
- Scatter 패널
 - Data
 - Fields
 - Visualization
 - 확인 순서
- Status Grid 패널
 - Data
 - Fields
 - Visualization
 - 확인 순서
- Trace 패널
 - Data
 - Fields
 - Visualization
 - 확인 순서

- **Gantt 패널**
 - Data
 - Fields
 - Visualization
 - 확인 순서
- **Pie 패널**
 - Data
 - Fields
 - Visualization
 - 확인 순서
- **Logs 패널**
 - Data
 - Fields
 - Visualization
 - 확인 순서
- **인시던트 및 워룸 개요**
 - 권장 대응 흐름
 - 먼저 구분해야 할 세 가지 동작
 - 문서 묶음
- **인시던트 확인**
 - 목록에서 대응할 문제 찾기
 - 상세에서 1차 판단하기
 - 담당자와 상태 관리하기
 - 필요한 신호 직접 추가하기
 - 정탐·오탐·무시로 근거 분류하기
 - 인시던트 전체 삭제하기
 - 검토를 워룸 대응으로 잇기
- **워룸 협업**
 - 채널을 찾고 현재 상황 파악하기
 - 대응 기록 남기기
 - AI 요약과 명령어 사용하기

- 오른쪽 작업 메뉴 활용하기
- 채널을 닫기 전 확인하기
- **종료와 정리**
 - 상태별 의미
 - 해결 처리 전 확인할 것
 - 종료 뒤 남길 기록
 - 삭제와 종료를 혼동하지 않기
 - 권장 마무리 순서
- **알림 확인과 설정**
 - 최근 알림 빠르게 확인하기
 - 필요한 알림 찾기
 - 어떤 알림을 받을지 정하기
 - 알림을 받을 위치 정하기
 - Monithub를 보고 있을 때 알림 받기
 - 알림이 보이거나 들리지 않을 때
- **Telegram 연결과 워룸 전송**
 - Telegram 계정 연결하기
 - Telegram에서 알림 확인하기
 - Telegram 메시지를 워룸으로 보내기
 - 연결된 조직과 도움말 확인하기
 - 계정을 바꾸거나 연결 해제하기
 - 연결이나 전송이 되지 않을 때
- **Connect 시작하기**
 - 시작하기 전에
 - 첫 연결 만들기
 - 화면 따라 연결하기: Linux 호스트 예시
 - 생성된 설정 적용하기
 - 어떤 카탈로그를 선택할까요?
 - 데이터 수집 확인하기
 - 다음 문서
- **전체 카탈로그**

- Custom OTLP
- 애플리케이션
- 인프라
- 데이터베이스
- Kubernetes
- 공식 문서
- Custom OTLP 연결
 - 설정 생성
 - Collector 실행과 송신 주소 적용
 - 연결 후 확인
- 애플리케이션 연결
 - 공통 적용 순서
 - Java
 - Node.js
 - Python
 - 연결 후 확인
- 인프라 연결
 - Linux 호스트
 - Docker
 - NGINX
 - 연결 후 확인
- 데이터베이스 연결
 - 공통 연결 순서
 - 전송 보안 선택
 - 런타임 값 적용
 - PostgreSQL
 - MySQL / MariaDB
 - Redis
 - TLS 인증서 변수
 - 여러 대상과 관리형 데이터베이스
 - 연결 후 확인

- **Kubernetes 연결**
 - 시작하기 전에
 - 1. Kubernetes 설정 계획 만들기
 - 2. 생성된 manifest 검토하기
 - 3. 클러스터에 적용하기
 - 4. 수집 범위 확인하기
 - 5. Connect에서 연결 확인하기
 - Operator 설치와의 차이
 - 연결을 삭제할 때
- **기존 설정에 적용**
 - 지원 범위
 - 1. 현재 구성 백업
 - 2. Connect 설정 복사
 - 3. 기존 구성에 병합
 - 4. 검증하고 반영
 - 5. 데이터 수집 확인
- **연결 및 수집 확인**
 - 연결 상태와 수집 상태 구분하기
 - 연결 상태 확인하기
 - 데이터 수집 상태 확인하기
 - 데이터 수집 확인 사용하기
 - 다시 선택, 재시도와 삭제
 - 조직 토큰 갱신
 - 다음 단계
- **서비스 맵**
 - 서비스 맵 열기
 - 문제 구간 분석하기
 - 서비스 맵이 보이지 않을 때
- **Monithub Sentinel**
 - Sentinel에서 하는 일
 - 화면을 보는 순서

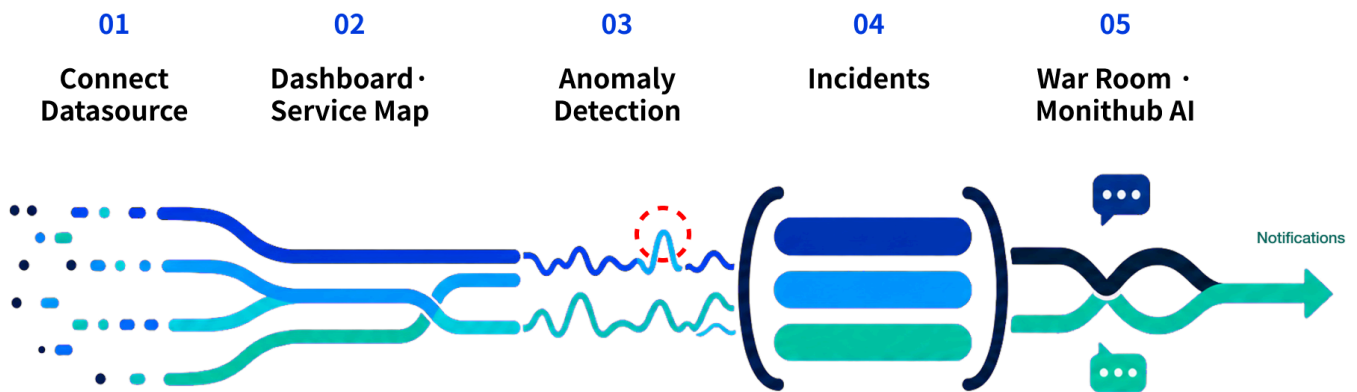
- 시작하기 전에
- 기본 사용 흐름
- 주기 분석과 최근 분석
- 다음에 볼 문서
- 결과를 믿기 전에 확인할 것
- 모델 관리
 - 모델 목록 읽기
 - 모델 상세 정보 읽기
 - 모델 개수와 활성 상태 관리하기
 - 상태별로 할 일
 - 후보를 활성화하기
 - 모델 버전 삭제
- 학습과 튜닝
 - 학습과 재학습의 차이
 - 실행 상태 확인하기
 - 탐지 성향 고르기
 - 고급 설정 조정하기
 - 고급 설정 조정 순서
 - 결과에 따라 조정하기
- 주기 분석
 - 사용 조건
 - 분석 주기 변경
 - 최근 분석 확인
 - 예정 시각이 지나도 실행되지 않을 때
- 라벨
 - 라벨의 의미
 - 라벨 목록 읽기
 - 라벨 확인 순서
 - 원본 화면에서 확인하기
 - 재학습 반영 상태 이해하기
 - 라벨 삭제하기

- 반복되는 라벨을 활용하는 방법
- **확인**과 문제 해결
 - 상태별 빠른 안내
 - 1분 점검 순서
 - 새 모델을 만들 수 없을 때
 - 활성 버전이 없을 때
 - 소스 커버리지가 낮을 때
 - 학습이 실패할 때
 - 상태가 바로 바뀌지 않을 때
 - 결과가 기대와 다를 때
 - 주기 분석이 시작되지 않을 때
 - 최근 분석 결과가 기대와 다를 때
 - 도움을 요청할 때 남길 정보
- **실습 가이드**
 - 시작 전 확인하기
 - 1. 첫 학습 시작하기
 - 2. 학습 설정 확인하고 시작하기
 - 2-1. 고급 설정 조정하기
 - 3. 학습 완료 기다리기
 - 4. 후보 확인하기
 - 5. 운영에 반영하기
 - 6. 새 버전 학습이 필요할 때
- **설정**
 - Profile
 - Settings
 - 설정 섹션
- **요금과 구독**
 - 시작하기 전에
 - 탭 한눈에 보기
 - 개요
 - 요금제 및 부가 기능

- 사용량
- 청구 요약
- 구독
- 정기 확인 흐름
- **Monithub K8S Operator 설치**
 - 설치 전 준비
 - 1. Monithub Connect 값 확인
 - 2. 클러스터 context 확인
 - 3. Namespace와 Secret 생성
 - 4. values 파일 작성
 - 5. Helm registry 로그인
 - 6. Monithub K8S Operator 설치
 - 7. 설치 상태 확인
 - 8. Monithub 화면에서 확인
 - 자주 바꾸는 설정
 - 업데이트와 롤백
 - 제거
 - Troubleshooting
 - 다음 단계
- **Metrics, Logs 및 Traces 필터링**
 - 동작 방식
 - 시작하기 전에
 - 권장 설정 예제
 - 규칙을 읽는 방법
 - Metric과 datapoint의 차이
 - Trace 필터링에서 알아둘 점
 - 지원하는 필드
 - 지원하는 연산자
 - 값 타입 작성법
 - 규칙 제한
 - 적용 상태 확인

- [안전하게 검증하는 방법](#)
- [필터 제거하기](#)
- [문제 해결](#)
- [자주 묻는 질문](#)
- [다음 단계](#)

Monithub 소개



Monithub는 대시보드, 회의 기록, 문서와 대화방에 흩어진 운영 맥락을 하나로 연결하는 관측 및 인시던트 대응 플랫폼입니다. 문제가 발생할 때마다 여러 화면에서 데이터를 찾고, 흩어진 기록을 모아 상황을 다시 설명하는 반복을 줄이기 위해 만들었습니다. 시스템에서 발생하는 신호와 대응 기록을 연결해, “**지금 어디에 문제가 있고, 무엇이 영향을 받았으며, 우리 팀은 다음에 무엇을 해야 하는가?**”라는 질문에 팀이 함께 답하도록 돕습니다. 대응 과정에서 남은 근거와 판단, 해결 결과는 다음 대응과 탐지 기준 개선에 다시 쓰여 운영 지식이 담당자의 기억에만 머물지 않게 합니다.

메트릭, 로그, 트레이스 và 네트워크 연결 정보는 시스템 상태를 이해하는 핵심 자료입니다. 그러나 데이터가 서로 다른 화면에 있고 판단과 논의가 별도의 회의록이나 대화방에 남으면, 문제를 발견할 때마다 맥락을 처음부터 다시 구성해야 합니다. 이상 징후가 나타난 시점과 영향 대상을 찾고, 근거를 모아 담당자와 대응 과정을 공유하는 일까지 하나의 흐름으로 이어져야 문제를 빠르게 해결할 수 있습니다. Monithub는 관측, 판단, 협업과 기록이 도구 사이에서 끊기지 않도록 연결합니다.

이 문서에서 알 수 있는 것

이 문서는 화면 조작 순서보다 Monithub가 해결하는 문제, 기능 간 연결과 팀의 시작 지점을 설명합니다. 실제 화면 사용 방법은 [제품 사용 가이드](#)에서 확인할 수 있습니다.

1. Monithub가 해결하려는 문제

운영 중인 시스템에서는 하나의 문제가 여러 신호로 나타납니다. 응답 시간이 느려지고 오류 로그가 늘어나며, 특정 서비스의 호출이 실패하거나 Kubernetes 리소스 상태가 달라질 수 있습니다. 신호를 따로 보면 중요한 변화인지 일시적인 흔들림인지 판단하기 어렵습니다.

문제를 발견한 뒤에도 담당자는 여러 화면에서 근거를 모으고, 메신저에서 상황을 다시 설명하며, 누가 무엇을 확인했는지 기록해야 합니다. 대응이 끝나면 흩어진 기록을 다시 정리하고, 같은 유형의 신호가 반복될 때는 탐지 기준도 조정해야 합니다.

이 과정이 개인의 경험과 기억에만 남으면 담당자가 바뀔 때마다 같은 조사와 설명을 반복하게 됩니다. 대응 중 확인한 근거와 판단, 조치 결과가 다음 문제 해결과 탐지 개선에 다시 쓰일 수 있어야 장애 대응과 운영 노하우가 조직 전체의 자산이 됩니다.

Monithub는 이 과정에서 반복되는 네 가지 어려움을 줄이는 데 초점을 둡니다.

운영 중 겪는 어려움	Monithub가 돕는 방식
데이터는 많지만 현재 상태를 한눈에 파악하기 어렵습니다.	홈, 대시보드와 서비스 맵에서 조직의 상태와 변화 흐름을 여러 관점으로 확인합니다.
여러 이상 신호 중 무엇부터 확인해야 할지 판단하기 어렵습니다.	Sentinel의 이상탐지 결과와 인시던트의 심각도, 영향 범위, 관련 근거를 함께 살펴봅니다.
분석 화면과 대응 대화가 분리되어 맥락이 끊깁니다.	인시던트에서 확인한 문제를 위룸으로 이어 담당자, 대화, 요약과 대응 기록을 같은 맥락에서 관리합니다.
대응 경험이 다음 탐지와 운영 개선으로 이어지지 않습니다.	인시던트 해결 기록과 탐지 결과 라벨을 함께 남기고, 이를 모델 학습·검토와 후속 개선에 연결해 운영 경험을 조직의 자산으로 축적합니다.

2. 관측에서 대응까지 이어지는 하나의 흐름 [🔗](#)

Monithub는 개별 기능의 모음이 아니라, 데이터를 연결하고 상태를 관측한 뒤 문제에 대응하며 그 경험을 다음 운영에 활용하는 하나의 흐름으로 이해할 수 있습니다.

2.1. 운영 데이터를 연결합니다 [🔗](#)

먼저 조직의 시스템에서 Monithub로 관측 데이터가 들어오도록 연결합니다. Connect에서는 OpenTelemetry Collector, Linux 호스트, Docker, Java 애플리케이션, Kubernetes와 Custom OTLP 같은 수집 대상을 선택하고 연결 계획을 세울 수 있습니다. Kubernetes 환경에서는 Monithub K8S Operator가 수집 컴포넌트를 설치하고 관리하도록 구성합니다.

연결한 뒤에는 설치 여부뿐 아니라 실제 데이터가 수집되는지, 어떤 서비스가 관찰되는지 확인해야 합니다. 연결 목록, 대시보드와 서비스 맵을 함께 살펴보면 수집 상태를 더 정확하게 검증할 수 있습니다.

Connect

Bare Metal과 Kubernetes 텔레메트리 소스를 위한 Agent 및 Collector 설정을 생성합니다.

텔레메트리 온보딩

Monithub 수집 파이프라인에 소스 연결

OpenTelemetry 통합을 선택하고 조직 토큰 기반 설정 계획을 생성합니다.

연결됨	대기 중	확인 필요	카탈로그 항목
1	1	2	5

카탈로그

연결 목록

소스 카탈로그

설치 위치를 먼저 선택한 뒤 Monithub에 연결할 수집 방식을 고릅니다.

통합 카탈로그 (4)

Bare Metal 또는 Kubernetes 환경에 맞는 텔레메트리 소스를 선택합니다.

Bare Metal

Linux 호스트, Docker 호스트, 애플리케이션 컨테이너에서 직접 수집합니다.

Kubernetes

Kubernetes 클러스터의 단위 수집 또는 클러스터 내부 OTLP 전송을 설정합니다.

Bare Metal 수집 방식

커스텀 OTLP

로컬 OTLP 메트릭, 로그, 트레이스 Monithub로 relay하는 OpenTelemetry Collector를 실행합니다.

Linux, macOS, Windows, Docker

Linux 호스트

Linux의 OpenTelemetry Collector를 통해 호스트 메트릭과 로그를 수집합니다.

Linux, Docker

Docker

otelcol-contrib로 Docker 내론과 컨테이너 컨테이너 메트릭을 수집합니다.

Linux, Docker

Java 애플리케이션

OpenTelemetry Java Agent를 설치하고 메트릭, 로그, 트레이스를 Monithub로 전송합니다.

Linux, macOS, Windows, Docker

설정 계획

Linux의 OpenTelemetry Collector를 통해 호스트 메트릭과 로그를 수집합니다.

Linux

Docker

수집 대상 Linux 서버에 Collector 바이너리를 설치해 호스트 메트릭과 시스템 로그를 수집합니다.

- 리소스 식별자**

이 속성은 생성된 명령어와 Collector 설정에 적용됩니다.

서비스 또는 소스 이름

instance-20260804-sldj1ks

환경 이름

dev
- 생성**

수집 대상 Linux 서버에 Collector 바이너리를 설치해 호스트 메트릭과 시스템 로그를 수집합니다.

연결 준비

2.2. 현재 상태와 변화 흐름을 관측합니다

홈 로그인 직후 현재 조직의 열린 인시던트, 최근 활동과 시스템 상태를 요약해 보여 줍니다. 더 자세한 분석이 필요하면 대시보드에서 시간 범위와 변수를 조정하고, 메트릭·로그·트레이스·이벤트를 목적에 맞는 패널로 살펴봅니다.

서비스 맵은 서비스 간 호출 관계를 토폴로지 보여 줍니다. 느리거나 실패한 연결을 중심으로 upstream과 downstream을 좁혀 보면 문제가 시작된 지점과 영향이 퍼지는 방향을 파악할 수 있습니다.

2.3. 평소와 다른 신호를 찾고 판단합니다

Monithub Sentinel은 관측 데이터에서 평소 패턴과 다른 구간을 찾습니다. 원본 데이터와 탐지 결과를 같은 시간축에 표시하므로 값의 변화와 이상 구간을 함께 비교할 수 있습니다.

이상 신호가 곧 확정된 장애를 의미하지는 않습니다. 운영자는 해당 시점의 시스템 상태와 관련 근거를 확인하고 정탐·오탐·미탐 같은 라벨을 남길 수 있습니다. 운영팀은 이 판단 기록과 실제 해결 결과를 바탕으로 새 후보 모델을 학습하고 품질을 검토한 뒤, 운영에 적합한 버전만 활성화합니다. 탐지 성향과 학습 조건을 조정하고 모델을 승격하는 일은 운영 근거에 따라 사람이 결정합니다.

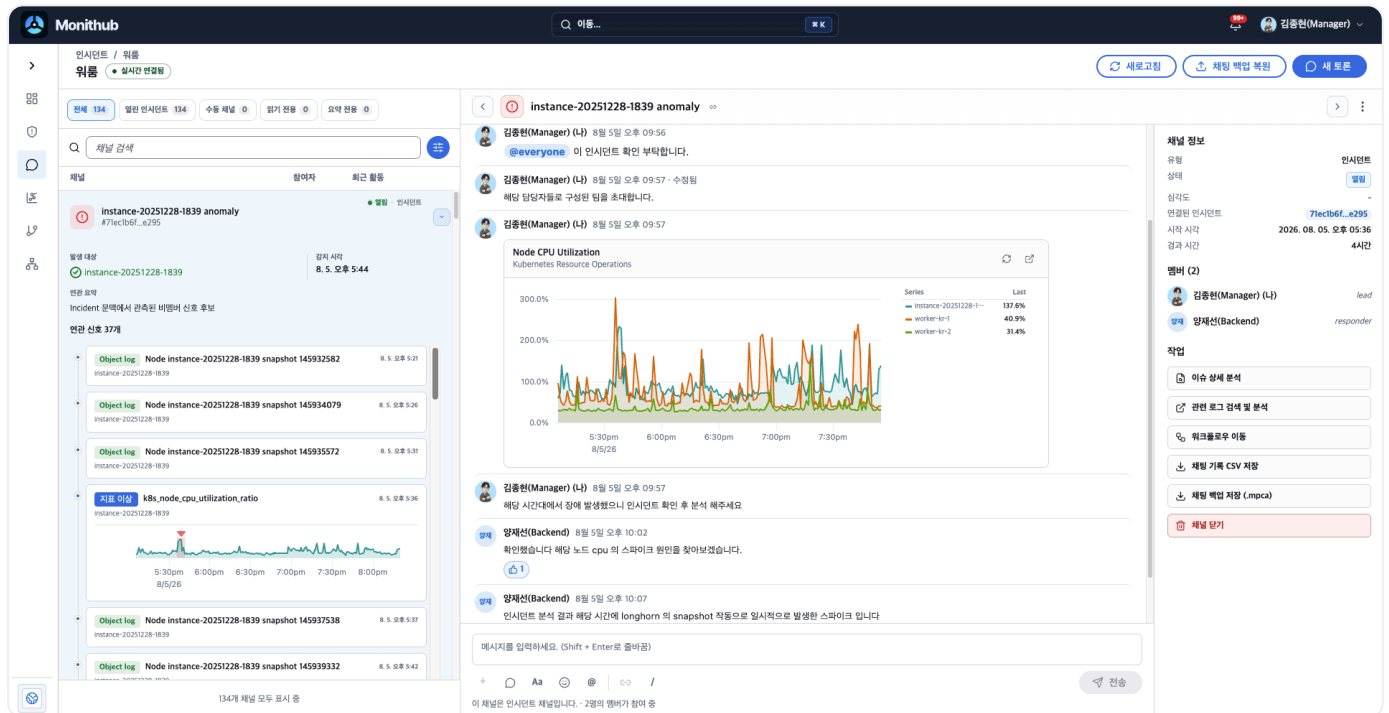
2.4. 관련 신호를 인시던트의 맥락으로 확인합니다

인시던트는 문제 대응의 시작점입니다. 개별 경고를 나열하는 대신, 함께 살펴봐야 할 이상 신호와 영향 대상, 심각도, 상태, 담당자와 시간 흐름을 한 공간에 모아 보여 줍니다.

운영자는 인시던트가 실제로 진행 중인지, 어떤 서비스나 리소스가 영향을 받았는지, 무엇을 먼저 확인해야 하는지 판단합니다. 메트릭, 로그, 트레이스와 Kubernetes 맥락을 함께 살펴보면 화면을 옮길 때마다 문제를 다시 설명하는 일을 줄일 수 있습니다. 발견 당시의 근거부터 최종 해결 결과까지 같은 인시던트에 남아 이후 유사한 문제를 검토할 때 다시 활용할 수 있습니다.

2.5. 워룸에서 함께 대응하고 기록합니다

워룸은 인시던트별 대화와 대응 기록을 관리하는 협업 공간입니다. 담당자를 부르고 확인한 사실과 가설을 공유하며, 누가 무엇을 확인했는지, 어떤 근거로 다음 행동을 정했는지, 조치 결과가 어땠는지를 한 흐름으로 남길 수 있습니다.



The screenshot displays the Monithub web application interface. On the left, a sidebar shows navigation icons and a search bar. The main area is divided into three sections: a chat window on the left with messages from '김종현(Manager)' and '양재선(Backend)', a central graph titled 'Node CPU Utilization' showing three data series (Instance-20251228-1, worker-kr-1, worker-kr-2) over time, and a right-hand panel with incident details like 'instance-20251228-1839 anomaly' and a list of actions such as '이슈 상세 분석' and '관련 로그 검색 및 분석'.

Monithub AI는 워룸 대화와 인시던트 맥락을 바탕으로 요약과 브리프 작성을 돕습니다. 긴 대화에서 확인된 내용과 남은 질문을 빠르게 파악하는 데 유용하지만, 최종 판단과 실행은 실제 시스템 상태를 확인한 담당자가 결정합니다.

이렇게 남은 기록은 종료 후 회고를 위한 자료에 그치지 않습니다. 새 담당자가 과거의 판단 근거를 파악하고, 같은 유형의 문제가 반복될 때 이전 조치와 결과를 재사용하는 조직의 운영 지식이 됩니다.

2.6. 대응 결과를 다음 운영에 반영합니다

문제를 해결한 뒤에는 원인, 영향 범위, 확인한 근거, 실행한 조치와 결과를 인시던트와 워룸의 맥락 안에서 정리합니다. 탐지 결과가 실제 상황과 달랐다면 운영팀은 해결 기록과 정탐·오탐·미탐 라벨을 바탕으로 새 모델 후보를 학습하고, 기존 모델과 비교·검토해 탐지 기준을 조정합니다.

관측, 판단, 대응과 피드백이 이렇게 연결되면 유사한 문제가 생겼을 때 과거의 근거와 해결 방법을 바로 찾아 더 일관된 기준으로 대응할 수 있습니다. 장애 대응과 운영 노하우가 조직 전체의 운영 자산으로 축적되어 특정 사람의 기억에 의존하지 않는 운영 관리를 돕는 것이 이 흐름의 목적입니다.

3. 누구를 위한 제품인가요?

Monithub는 운영 데이터를 직접 분석하는 사람뿐 아니라, 문제 상황에서 같은 맥락을 공유해야 하는 팀 전체를 위한 제품입니다.

사용자	Monithub에서 주로 하는 일
SRE·DevOps·운영 담당자	홈과 대시보드에서 상태를 확인하고, 서비스 맵과 인시던트 근거를 바탕으로 문제 범위를 좁힙니다.
개발자	담당 서비스의 메트릭, 로그와 트레이스를 확인하고, 워룸에서 원인 가설과 수정 결과를 공유합니다.
인시던트 리더	심각도와 영향 범위를 판단하고, 담당자와 다음 행동을 정하며, 대응 상황을 요약하고 종료 조건을 확인합니다.
플랫폼·조직 관리자	조직과 멤버, 데이터 연결, 수집 상태, 알림 채널, 사용량과 요금제를 관리합니다.
기술 책임자·팀 리더	현재 위험과 반복되는 문제, 대응 진행 상황을 파악하고 우선순위와 후속 개선 작업을 결정합니다.

한 사람이 여러 역할을 맡을 수도 있습니다. 소규모 팀에서는 관리자가 데이터 연결과 인시던트 대응을 함께 담당하고, 규모가 커지면 관측, 분석, 대응과 조직 관리 역할을 나눌 수 있습니다.

역할이나 담당자가 바뀌더라도 같은 인시던트 맥락과 해결 기록을 이어서 볼 수 있으므로, 팀은 개인의 경험을 조직의 공통된 운영 기준으로 발전시킬 수 있습니다.

4. 먼저 알아두면 좋은 핵심 개념

처음에는 메뉴 이름보다 아래 개념이 어떻게 이어지는지 이해하는 것이 중요합니다.

개념	쉽게 말하면
조직	사용자, 권한, 연결과 운영 데이터를 구분하는 작업 공간입니다. 상단에서 조직을 바꾸면 화면에 보이는 데이터 범위도 달라집니다.
관측 데이터	시스템 상태를 이해하기 위해 수집하는 메트릭, 로그, 트레이스, 이벤트와 네트워크 관계 정보입니다.
Connect	어떤 환경과 수집 대상을 Monithub에 연결할지 선택하고 설정 계획을 만드는 기능입니다.
대시보드	여러 관측 데이터를 목적에 맞는 패널로 배치해 함께 보는 화면입니다.
패널	숫자, 시계열, 표, 로그, 트레이스 등 하나의 질문에 답하도록 데이터를 표현하는 대시보드 구성 단위입니다.
서비스 맵	서비스 간 호출 방향과 연결 상태를 그래프로 보여 주는 화면입니다.
이상 신호	평소 패턴과 달라 추가 확인이 필요한 데이터 구간입니다. 이상 신호가 항상 실제 장애를 뜻하는 것은 아닙니다.
Sentinel 모델	이상 신호를 찾는 기준입니다. 운영자가 해결 기록과 탐지 결과 라벨을 바탕으로 새 후보를 학습·검토한 뒤 활성 버전을 운영에 사용합니다.
인시던트	관련 신호와 영향 대상을 묶어 문제의 근거, 판단, 조치와 해결 결과를 관리하는 작업 단위입니다.
위험	인시던트 참여자가 누가 무엇을 확인하고 결정했는지 대화, 행동과 결과를 함께 남기는 협업 공간입니다.
알림	새 인시던트, Connect 수집 상태와 위험 메시지처럼 사용자가 확인해야 할 변화를 전달하는 기능입니다.
Gateway	AI Gateway의 상태, 등록된 서비스와 모델, 라우팅 및 요청 테스트를 확인하는 운영 화면입니다.

5. 주요 기능을 사용자 관점에서 살펴보기 [🔗](#)

5.1. 홈: 어디에서부터 확인할지 정합니다 [🔗](#)

홈은 여러 운영 화면 중 첫 번째 확인 지점입니다. 열린 인시던트 수, 최근 활동, 영향 대상과 시스템 상태를 함께 보며 지금 확인해야 할 문제가 있는지 판단할 수 있습니다.

정상 여부를 단정하기보다 **오늘의 운영 상태에서 어디를 먼저 볼지 정하는 화면**입니다. 위험 또는 주의 상태가 보이거나 특정 대상이 반복해서 영향을 받았다면 대시보드나 인시던트로 이동해 근거를 확인합니다.

[홈과 기본 탐색 방법 알아보기 →](#)

5.2. 대시보드와 패널: 질문에 맞게 데이터를 구성합니다 [🔗](#)

대시보드는 Kubernetes 리소스, 애플리케이션, 로그, 트레이스, Gateway, 비용과 수집량 등 서로 다른 운영 주제를 패널 단위로 구성합니다. 시간 범위와 Namespace, Node, Pod 같은 변수를 사용하면 같은 대시보드에서도 분석 대상을 빠르게 좁힐 수 있습니다.

운영자는 큰 숫자로 현재 상태를 확인하고, 시계열로 변화 시점을 찾으며, 표와 로그에서 구체적인 근거를 살펴봅니다. 필요한 권한이 있으면 데이터 쿼리, 필드 표현, 시각화 방식과 패널 배치를 업무에 맞게 편집할 수 있습니다.

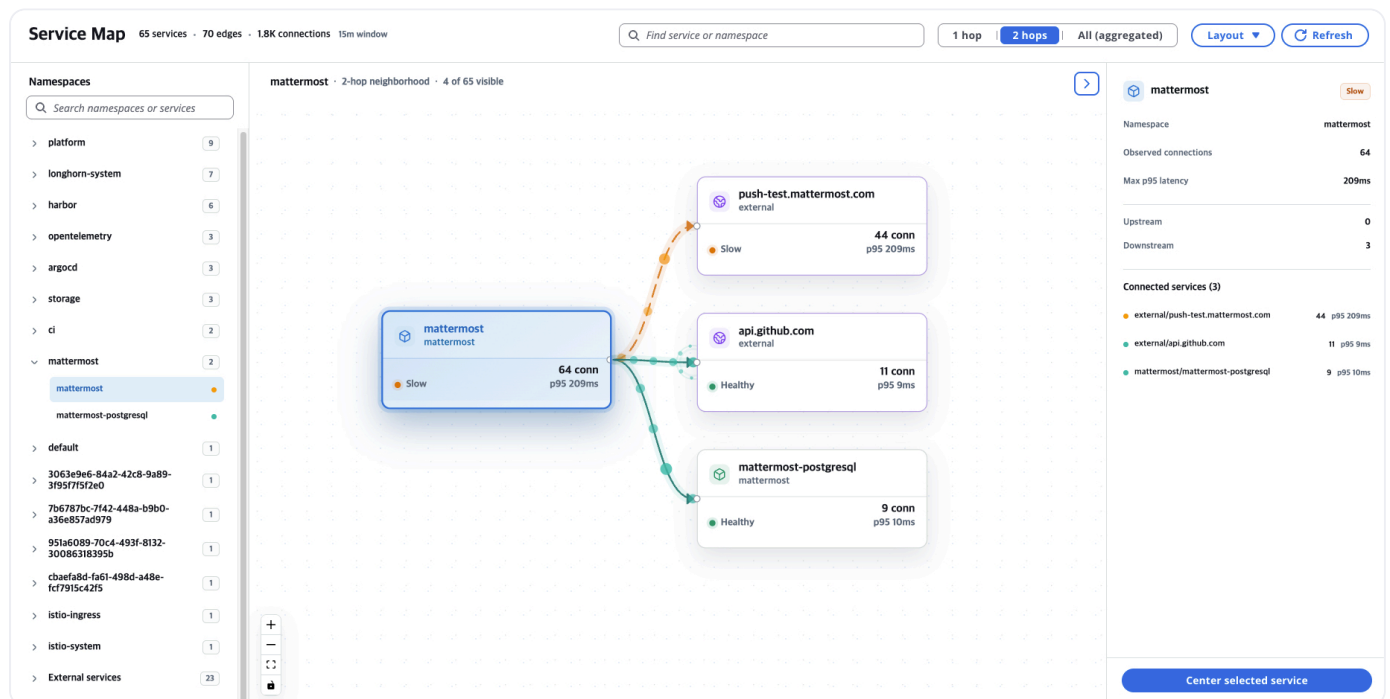
- [대시보드 사용 방법 →](#)
- [패널 편집과 시각화 유형 →](#)

5.3. 서비스 맵: 영향이 퍼지는 방향을 이해합니다 [🔗](#)

서비스 맵은 선택한 서비스를 중심으로 직접 연결된 upstream과 downstream을 보여 줍니다. 서비스 상태, 호출 방향과 지연 시간을 함께 비교하면 “어느 서비스가 느린가?”뿐 아니라 “어느 연결에서 문제가 시작되었고 다음에 무엇이 영향을 받는가?”까지 살펴볼 수 있습니다.

연결이 복잡한 환경에서는 1-hop부터 확인하고 필요할 때 2-hop으로 넓히는 것이 좋습니다. 전체 토폴로지 큰 흐름을 비교하되, 실제 문제는 영향받은 서비스를 중심으로 범위를 좁혀 분석합니다.

서비스 맵에서는 왼쪽에서 네임스페이스와 서비스를 선택하고, 가운데에서 연결 흐름을 살펴보며, 오른쪽에서 연결 수와 지연 시간 같은 핵심 정보를 확인할 수 있습니다.



[서비스 맵 사용 방법 알아보기 →](#)

5.4. Sentinel: 이상탐지를 운영 경험에 맞게 관리합니다 [🔗](#)

Sentinel은 관측 데이터에서 이상 징후를 찾고 모델 버전을 관리하는 기능입니다. 운영자는 Sentinel이 제시한 탐지 결과를 실제 인시던트 해결 기록과 비교하고 라벨을 남긴 뒤, 이를 근거로 새 모델 후보를 학습·검토해 탐지 기준을 운영 환경에 맞게 다듬습니다.

모델 튜닝 화면에서는 활성·후보 버전, 다음 학습 시각과 라벨의 재학습 반영 여부를 한눈에 확인할 수 있습니다. 주기 분석 일정과 최근 실행 결과도 같은 화면에서 살펴보며 학습과 탐지가 계획대로 이어지는지 점검합니다. 탐지가 지나치게 민감하거나 중요한 변화를 놓친다면 운영자가 라벨과 해결 결과를 근거로 학습 조건과 분석 주기를 조정하면 새 후보를 다시 검증합니다.

Settings

프로필 Incident Prometheus ClickHouse Organization **Sentinel**

모델 버전 + 새 버전 학습 ↺ 새로고침

모델 이름	상태	품질	모델 생성 시간
V6	활성화	주의	2026. 8. 3. 오전 1:50:34
V5	후보	주의	2026. 8. 1. 오후 5:59:20
V4	후보	주의	2026. 8. 1. 오후 3:21:19

모델 상세 정보

모델 이름 V6 Sentinel Schema 0.1.0

소스 커버리지 79.5% (70/88)

다음 학습 시간 2026. 8. 17. 오전 1:50:34

최근 활성화 시간 2026. 8. 3. 오전 2:08:40

학습 당시 탐지 상황 균형

▼ 라벨 (3)

수정	라벨	대상 유형	패널	메트릭	재학습 반영	열기	Actions
2026. 8. 5. 오후 7:40:09	무시됨	이상	ts-node-cpu	k8s_node_cpu_utilization_ratio	미반영	열기	×
2026. 8. 5. 오후 5:45:12	오함	이상	pod-file-system	k8s_pod_filesystem_usage_bytes	미반영	열기	×
2026. 8. 5. 오후 5:45:09	오함	이상	pod-file-system	k8s_pod_filesystem_usage_bytes	미반영	열기	×

주기 분석 분석이 예약되어 있습니다

수집된 데이터를 바탕으로 정기적으로 이상 징후를 분석합니다.

현재 분석 주기 6시간마다

다음 분석 예정 2026. 8. 5. 오후 11:34:00

분석 주기 운영 주기 또는 개별을 단위 주기를 선택합니다.

6시간마다 변경 내용 확인

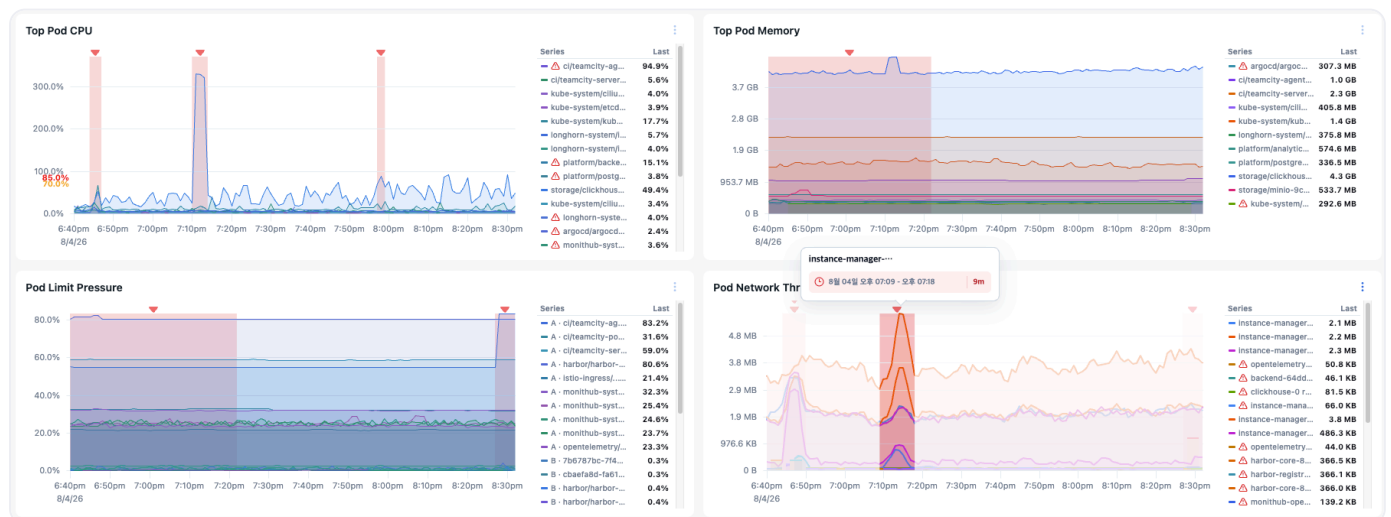
공식 사이트 Monithub v1.0.0
재출 품제 이미지 및 업데이트
monithub.org 방문

최근 분석

최근에 실행된 주기 분석 결과를 확인합니다.

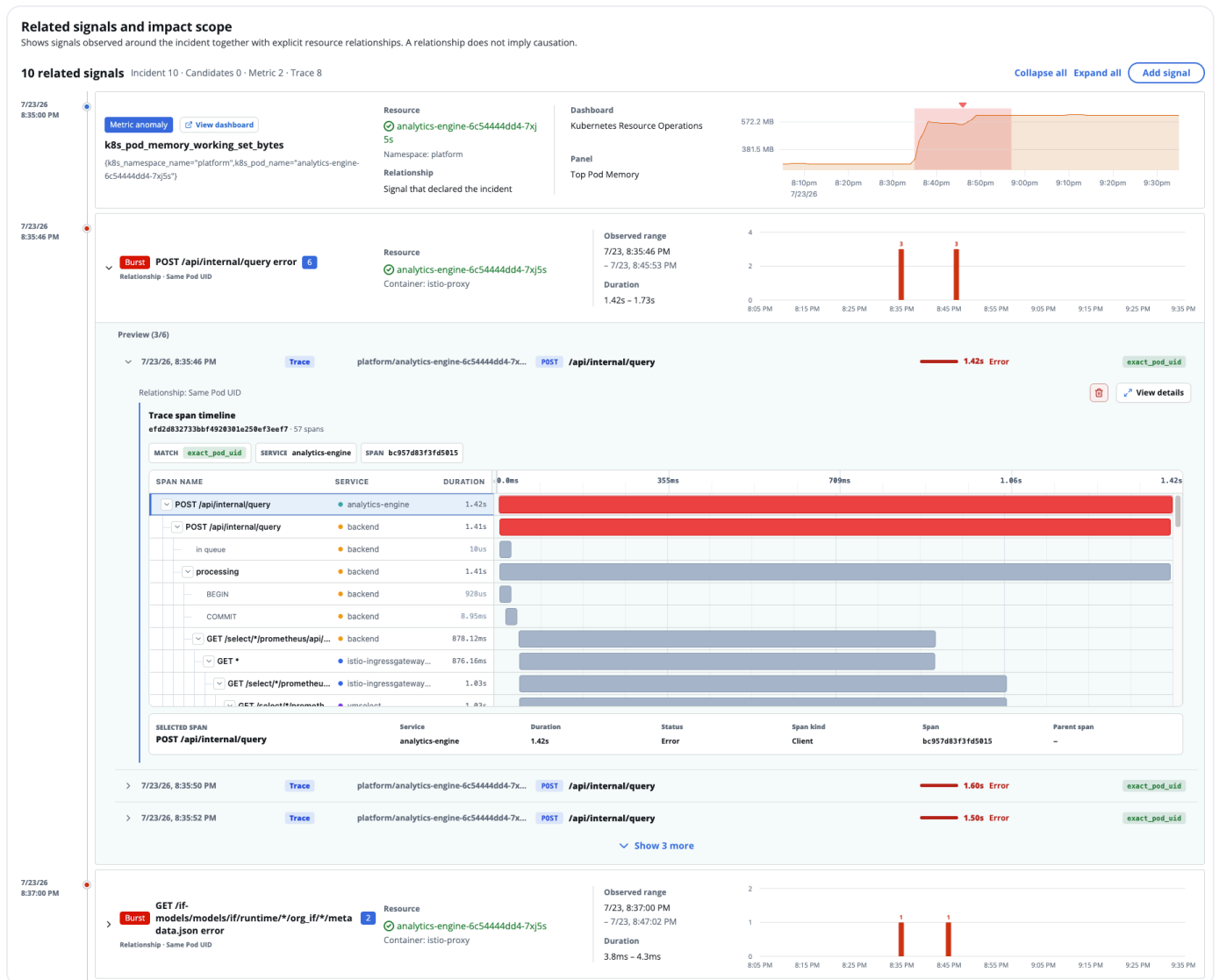
분석 시각	분석 대상 범위	상태
2026. 8. 5. 오후 5:34:03	2026. 8. 5. 오전 5:34:00 ~ 2026. 8. 5. 오후 5:34:00	실패
2026. 8. 5. 오전 9:34:02	2026. 8. 4. 오후 9:34:00 ~ 2026. 8. 5. 오전 9:34:00	실패
2026. 8. 5. 오전 1:34:03	2026. 8. 4. 오후 1:34:00 ~ 2026. 8. 5. 오전 1:34:00	실패
2026. 8. 4. 오후 5:34:02	2026. 8. 4. 오전 5:34:00 ~ 2026. 8. 4. 오후 5:34:00	실패

대시보드에서는 탐지된 이상 구간을 원본 시계열 위에 표시합니다. 구간을 선택하면 대상 시계열, 시작·종료 시각과 지속 시간을 확인할 수 있어 값의 변화와 탐지 결과를 같은 맥락에서 판단하기 쉽습니다.



5.5. 인시던트: 신호를 실제 문제의 맥락으로 정리합니다 [🔗](#)

인시던트에서는 문제의 상태와 심각도, 영향 대상, 담당자와 관련 근거를 함께 확인합니다. 목록에서 우선 순위를 판단하고 상세 화면에서 시간 흐름과 텔레메트리 맥락을 살펴본 뒤, 필요하면 문제를 분리하거나 관련 항목을 함께 다룰 수 있습니다.



목표는 단순히 경고 개수를 줄이는 것이 아닙니다. 팀이 같은 근거를 바탕으로 **지금 대응해야 하는 문제인지**, 누가 무엇을 확인할지, 언제 해결되었다고 볼지 합의하고 그 판단과 결과를 다음 대응에 남기는 것이 중요합니다.

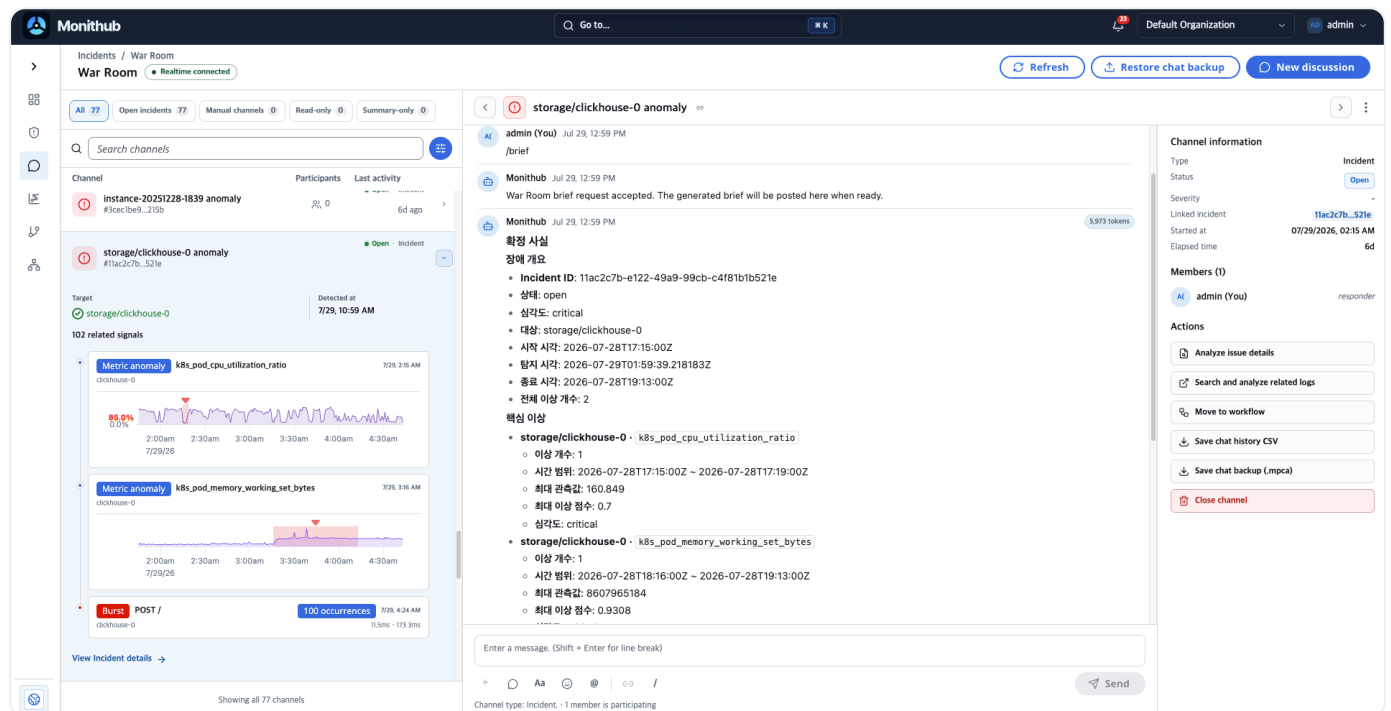
[인시던트와 워룸의 전체 흐름 알아보기 →](#)

5.6. 워룸과 Monithub AI: 대응 맥락을 공유합니다 [🔗](#)

워룸에서는 인시던트별로 대화를 나누고 조직 멤버를 부르며, 확인한 사실과 다음 행동을 기록합니다. 단순한 대화방이 아니라 담당자, 판단 근거, 조치와 결과를 인시던트에 연결해 대응 과정을 조직이 다시 활용할 수 있는 기록으로 남기는 공간입니다.

Monithub AI의 요약과 브리프는 새로 참여한 사람이 긴 대화를 모두 읽지 않고도 현재 상황을 파악하도록 돕습니다. 관찰된 문제, 지금까지 확인한 내용과 남은 질문을 정리한 뒤 실제 근거와 비교해 활용합니다.

대응이 끝난 뒤에도 이 기록을 통해 새 담당자는 당시 상황을 다시 설명받지 않고 원인과 해결 과정을 파악할 수 있습니다. 반복되는 문제에서는 과거에 효과가 있었던 조치와 남은 후속 작업을 바로 찾아볼 수 있습니다.



[워룸에서 협업하는 방법 알아보기 →](#)

5.7. 알림: 필요한 변화를 놓치지 않게 합니다 [🔗](#)

알림함은 새 인시던트, Connect 수집 상태와 워룸 메시지를 한곳에서 확인하는 기본 채널입니다. 읽음 상태, 유형, 심각도, 기간과 키워드로 항목을 좁혀 보고 해당 인시던트나 워룸으로 바로 이동할 수 있습니다.

사용 환경에 따라 Web Push, Telegram과 앱 알림음을 설정할 수 있습니다. 인시던트는 항상 받되, 워룸 메시지는 멘션만 받을지, 모든 메시지를 받을지, 받지 않을지 선택해 불필요한 알림을 줄일 수 있습니다.

알림

Default Organization

모두 읽음

알림 목록

설정

Q 제목 또는 내용 검색

필터 0

미읽음만

최근 30일

초기화

알림 목록

138개 알림 · 미읽음 133개

오늘 - 2026. 8. 5.

War Room에서 나를 댄설했습니다

@김종현(Manager)

오후 10:07

backend-5bcb46f94-2csv2 이상 감지

platform/backend-5bcb46f94-2csv2 · receive

오후 5:44

worker-kr-2 이상 감지

worker-kr-2

오후 5:44

worker-kr-1 이상 감지

worker-kr-1

오후 5:44

instance-20251228-1839 이상 감지

instance-20251228-1839

오후 5:44

worker-kr-1 이상 감지

worker-kr-1

오후 5:37

worker-kr-1 이상 감지

worker-kr-1

오후 5:37

instance-20251228-1839 이상 감지

instance-20251228-1839

오후 5:37

instance-20251228-1839 이상 감지

instance-20251228-1839

오후 5:37

backend-7b67f94c99-4lsq6 이상 감지

platform/backend-7b67f94c99-4lsq6

오후 5:35

War Room에서 나를 댄설했습니다

생성 시간 2026. 8. 5. 오후 10:07:54

@김종현(Manager)

신호 정보

심각도

Info

유형

워룸

대화 ID

71ca62f1-00d1-48f7-bdd2-1d658af97c7f

메시지 ID

2855e855-f66d-4808-af97-b5ab2f484749

소스

chat_message

▶ 페이로드 (JSON)

읽음으로 표시

War Room 열기

- [알림 확인과 설정 →](#)
- [Telegram 연결과 전송 →](#)

5.8. Connect와 Monithub K8S Operator: 관측 기반을 마련합니다 [🔗](#)

Connect에서는 수집 환경과 대상을 구분해 연결 계획을 만듭니다. Bare Metal과 Kubernetes 중 환경을 선택하고, OpenTelemetry Collector, Linux, Docker, Java 애플리케이션이나 Custom OTLP 등 필요한 대상을 고를 수 있습니다.

Kubernetes 환경에서는 Monithub K8S Operator가 Collector와 Agent를 설치하고 관리합니다. 처음에는 수집 범위를 작게 설정하고 데이터가 정상적으로 들어오는지 확인한 뒤, 필요한 신호와 범위를 점진적으로 넓히는 것이 좋습니다.

- [Connect와 Gateway 화면 알아보기 →](#)
- [Monithub K8S Operator 설치하기 →](#)
- [Telemetry 필터링하기 →](#)

5.9. 조직과 설정: 운영 범위와 사용 환경을 관리합니다 [🔗](#)

Monithub의 데이터와 권한은 조직을 기준으로 나뉩니다. 사용자는 새 조직을 만들거나 기존 조직에 참여하고, 관리자는 멤버를 초대하거나 참여 요청을 처리합니다. 화면 상단에서 현재 조직을 바꾸면 대시보드, 인시던트, 워룸과 요금·구독의 데이터 범위도 함께 달라집니다.

Monithub Docs · docs.monithub.org

22 / 225

Settings에서는 프로필, 인시던트, 데이터소스, 조직과 Sentinel 관련 설정을 관리합니다.

- [가입과 조직 참여 →](#)
- [멤버 초대와 참여 요청 →](#)
- [설정 →](#)

5.10. 요금과 구독: 사용 현황과 청구 상태를 관리합니다 [🔗](#)

요금과 구독에서는 현재 요금제와 갱신일, 사용량과 한도를 확인합니다. 운영자는 요금제와 부가 기능을 조정하고, 수동 청구 요약을 생성·확정·내보내거나 구독 상태를 관리할 수 있습니다.

- [요금과 구독 →](#)

6. 실제 상황에서는 어떻게 사용하나요? [🔗](#)

6.1. 서비스 응답이 갑자기 느려진 경우 [🔗](#)

1. 홈에서 열린 인시던트와 영향 대상, 최근 활동을 확인합니다.
2. 관련 대시보드에서 시간 범위를 문제 발생 시점으로 맞추고 응답 시간, 오류와 리소스 변화를 비교합니다.
3. 서비스 맵에서 느린 서비스를 선택해 직접 연결된 upstream과 downstream 상태를 확인합니다.
4. 인시던트 상세에서 관련 메트릭, 로그, 트레이스와 Kubernetes 맥락을 함께 살펴봅니다.
5. 워룸을 열어 담당자를 부르고, 확인된 사실과 원인 가설, 다음 행동을 기록합니다.
6. 대응 대화가 길어지면 AI 요약이나 브리프로 현재 상황을 정리하고 실제 근거와 비교합니다.
7. 복구 후 원인, 영향 범위, 확인 근거, 조치 결과와 반복 방지 작업을 정리하고, 탐지 결과 라벨과 모델 학습·검토 필요 여부를 남깁니다.

이 흐름의 목적은 모든 분석을 자동화하는 것이 아니라, 사람이 판단하는 데 필요한 근거와 협업 맥락을 끊임 없이 연결하는 것입니다. 축적된 해결 기록은 다음 담당자와 다음 인시던트가 다시 사용할 수 있는 조직의 운영 자산이 됩니다.

6.2. 이상탐지 알림이 너무 자주 발생하는 경우 [🔗](#)

1. 이상 구간이 실제 운영 문제였는지 대시보드와 인시던트 근거로 확인합니다.
2. 정탐, 오탐 또는 무시됨 라벨을 남겨 판단 결과를 구분합니다.

3. 같은 유형의 오탐이 반복되는지 기간과 데이터 소스별로 살펴봅니다.
4. 반복된 오탐의 해결 기록과 라벨을 근거로 운영자가 Sentinel의 탐지 성향이나 학습 조건을 조정해 새 후보 버전을 학습합니다.
5. 후보의 품질과 소스 커버리지를 검토한 뒤, 기존 활성 버전보다 적합할 때만 승격합니다.
6. 변경 후 대시보드의 이상 구간과 새 인시던트 흐름을 관찰해 개선 여부를 확인합니다.

6.3. 새 Kubernetes 클러스터를 연결하는 경우 [🔗](#)

1. 올바른 조직을 선택하고 Connect에서 Kubernetes 연결 계획을 만듭니다.
2. 발급된 식별 정보와 토큰, OTLP endpoint와 registry 인증 정보를 안전하게 준비합니다.
3. 클러스터에 Monithub K8S Operator와 기본 수집 컴포넌트를 설치합니다.
4. Monithub K8S Operator, Collector와 Agent 상태가 정상인지 확인합니다.
5. Monithub 연결 목록과 서비스 맵에서 실제 데이터가 들어오는지 검증합니다.
6. 기본 수집이 안정된 뒤 필요한 telemetry preset과 필터를 조정합니다.

7. 사용할 때 기억할 점 [🔗](#)

- **현재 조직을 먼저 확인합니다.** 조직이 다르면 같은 메뉴에서도 보이는 데이터와 권한이 달라집니다.
- **시간 범위와 필터를 함께 확인합니다.** 데이터가 없거나 평소와 다르게 보일 때는 수집 문제로 단정하기 전에 조회 범위를 확인합니다.
- **이상 신호와 AI 결과는 판단을 돕는 근거입니다.** 실제 장애 여부와 실행할 조치는 시스템 상태와 팀의 운영 기준을 바탕으로 결정합니다.
- **대응 기록은 다음 대응에서 다시 쓸 수 있게 남깁니다.** 확인 근거, 판단 이유, 조치 결과와 후속 작업을 인시던트와 위룸의 같은 맥락에 정리하면 특정 사람의 기억에 의존하지 않고 운영 경험을 이어갈 수 있습니다.
- **보이는 기능은 권한과 연결 상태에 따라 달라질 수 있습니다.** 필요한 메뉴가 없거나 데이터를 볼 수 없다면 조직 관리자에게 권한과 데이터소스 상태를 확인합니다.
- **민감한 연결 정보는 안전하게 관리합니다.** Connect token과 registry token은 문서, 이슈나 메신저에 남기지 않고 Secret으로 주입합니다.

8. 역할에 맞게 시작하기 [🔗](#)

지금 하려는 일

추천 시작 문서

처음 가입하고 화면을 익히고 싶습니다.

[가입과 조직 참여](#) → [홈과 기본 탐색](#)

우리 시스템의 상태를 관측하고 싶습니다.

[대시보드](#) → [서비스 맵](#)

인시던트 대응 흐름을 준비하고 싶습니다.

[인시던트 및 워룸 개요](#) → [알림 확인과 설정](#)

이상탐지 모델을 운영하고 싶습니다.

[Monithub Sentinel](#) → [실습 가이드](#) → [학습과 튜닝](#)

Kubernetes 데이터를 처음 연결하고 싶습니다.

[Connect와 Gateway](#) → [Monithub K8S Operator 설치](#)

프로필, 데이터소스와 조직 설정을 관리하고 싶습니다.

[설정](#)

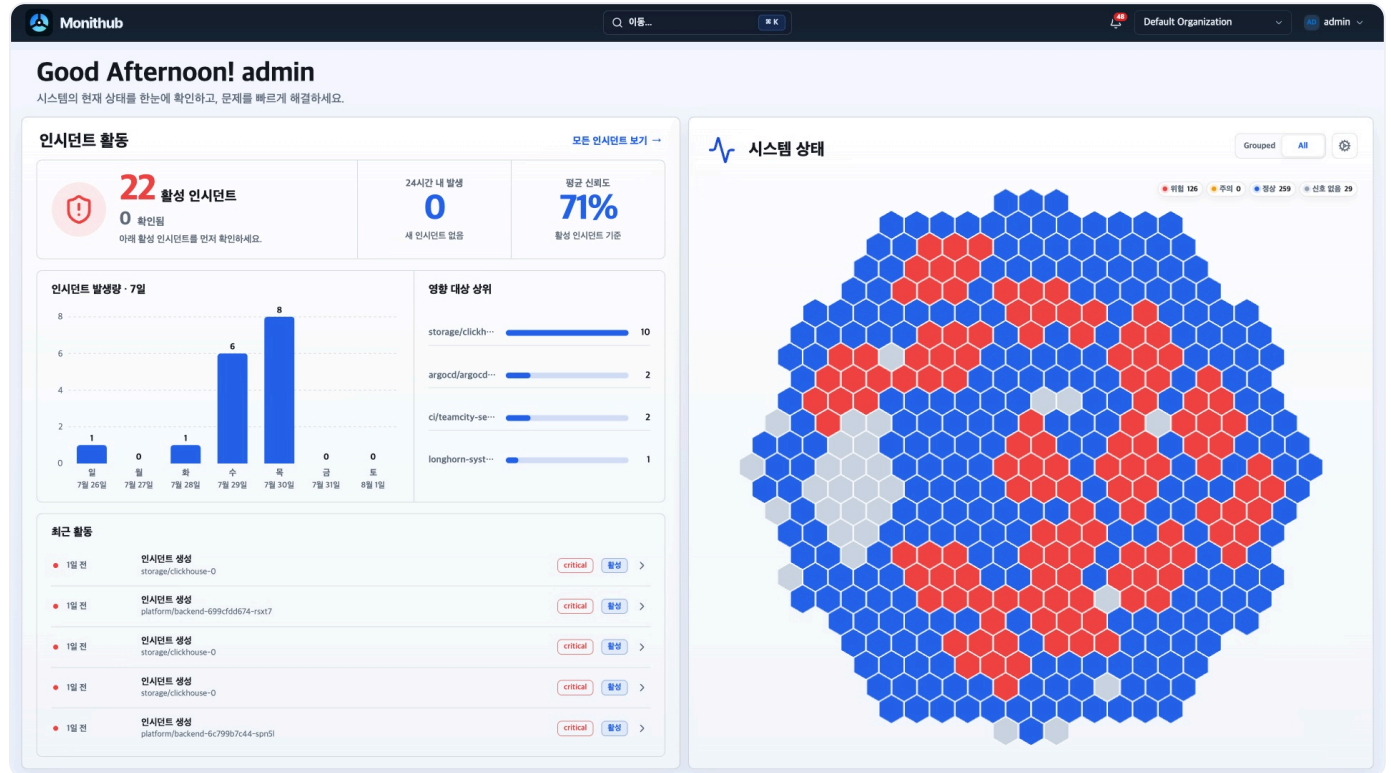
요금제, 사용량, 청구 요약과 구독 상태를 확인하고 싶습니다.

[요금과 구독](#)

제품의 역할과 전체 흐름을 이해했다면 [제품 사용 가이드](#)에서 필요한 작업을 골라 시작해 보세요.

제품 사용 가이드

이 가이드는 Monithub 화면에서 원하는 작업을 실제로 수행할 수 있도록 메뉴 위치, 확인할 내용과 실행 순서를 안내합니다. 제품의 목적과 기능이 서로 어떻게 이어지는지 먼저 알고 싶다면 [Monithub 소개](#)를 읽어 보세요.



기본 사용 흐름 [🔗](#)

1. 처음 사용하는 경우 [새 조직을 만들거나 기존 조직에 참여](#)합니다.
2. 로그인 후 상단에서 사용할 조직을 선택합니다.
3. 홈에서 현재 상태와 주요 메뉴를 확인합니다.
4. 대시보드에서 메트릭, 로그, 이벤트, 이상탐지 상태를 봅니다.
5. 필요한 경우 패널 편집 화면에서 쿼리, 표시 형식, 시각화 옵션, 레이아웃을 조정합니다.
6. 인시던트에서 문제 후보를 확인하고 상세 화면으로 들어갑니다.
7. 워룸에서 대응 대화, 분석 요청, 채팅 백업을 관리합니다.
8. [알림](#)에서 새 인시던트, Connect 수집 상태, 워룸 메시지를 확인하고 받을 방식을 설정합니다.
9. [Connect](#)에서 데이터 수집 연결과 최근 수집 상태를 확인합니다.
10. Monithub Sentinel에서 이상탐지 모델, 재학습, 라벨을 관리합니다.
11. [설정](#)에서 운영 환경, 데이터소스, 조직과 프로필을 관리합니다.

12. [요금과 구독](#)에서 요금제, 사용량, 청구 요약과 구독 상태를 관리합니다.

메뉴 한눈에 보기

메뉴	사용하는 때
계정과 조직	처음 가입하거나 기존 조직에 참여하고, 관리자가 멤버 초대와 참여 요청을 처리할 때
홈	로그인 직후 인시던트 활동 추이, 영향 대상, 최근 활동과 시스템 상태를 빠르게 볼 때
대시보드	Kubernetes, 로그, 트레이스, Gateway 등 운영 지표를 패널 단위로 보고 패널 편집으로 표시 방식을 조정할 때
인시던트	이상 신호가 묶인 문제 후보를 확인하고 담당자, 상태, 근거를 추적할 때
워룸	인시던트별 채팅방에서 대화, 요약, 백업, 분석 요청을 처리할 때
알림	새 인시던트, Connect 수집 상태, 워룸 메시지를 확인하고 Web Push, Telegram, 앱 알림음을 설정할 때
Connect	Collector, Java, Linux, Docker, Kubernetes, Custom OTLP 연결 계획을 만들 때
Gateway	Gateway 헬스, 서비스 레지스트리, 라우팅, 요청 테스트를 확인할 때
Service Map	서비스와 네트워크 edge 상태를 그래프로 볼 때
Monithub Sentinel	이상탐지 모델을 만들고 재학습, 후보 검토, 활성 승격, 라벨을 관리할 때
설정	데이터소스, 인시던트, 조직, Sentinel, 프로필과 화면 환경을 관리할 때
요금과 구독	현재 요금제, 갱신일, 사용량, 한도, 청구 요약과 구독 상태를 확인할 때

먼저 알아둘 것

- 상단의 조직 선택값에 따라 대시보드, 인시던트, 워룸, 요금과 구독 데이터가 달라집니다.
- [이동...](#) 또는 [⌘ K](#)를 사용하면 주요 메뉴와 대시보드를 빠르게 찾을 수 있습니다.
- 좌측 메뉴에서는 주요 기능과 조직의 대시보드를 바로 선택할 수 있습니다.
- 대부분의 화면에는 [Refresh](#) 또는 [새로고침](#) 버튼이 있어 최신 상태를 다시 불러올 수 있습니다.
- War Room을 사용하려면 Platform의 워룸 메뉴에서 원하는 방을 선택합니다.

가입과 조직 참여

Monithub에 처음 가입할 때는 새 조직을 만들지, 이미 만들어진 조직에 참여할지 먼저 선택합니다. 회사나 팀에서 초대 메일을 받았다면 메일의 링크로 시작하는 것이 가장 빠릅니다.

내 상황

선택할 항목

새 팀이나 프로젝트의 공간이 필요함

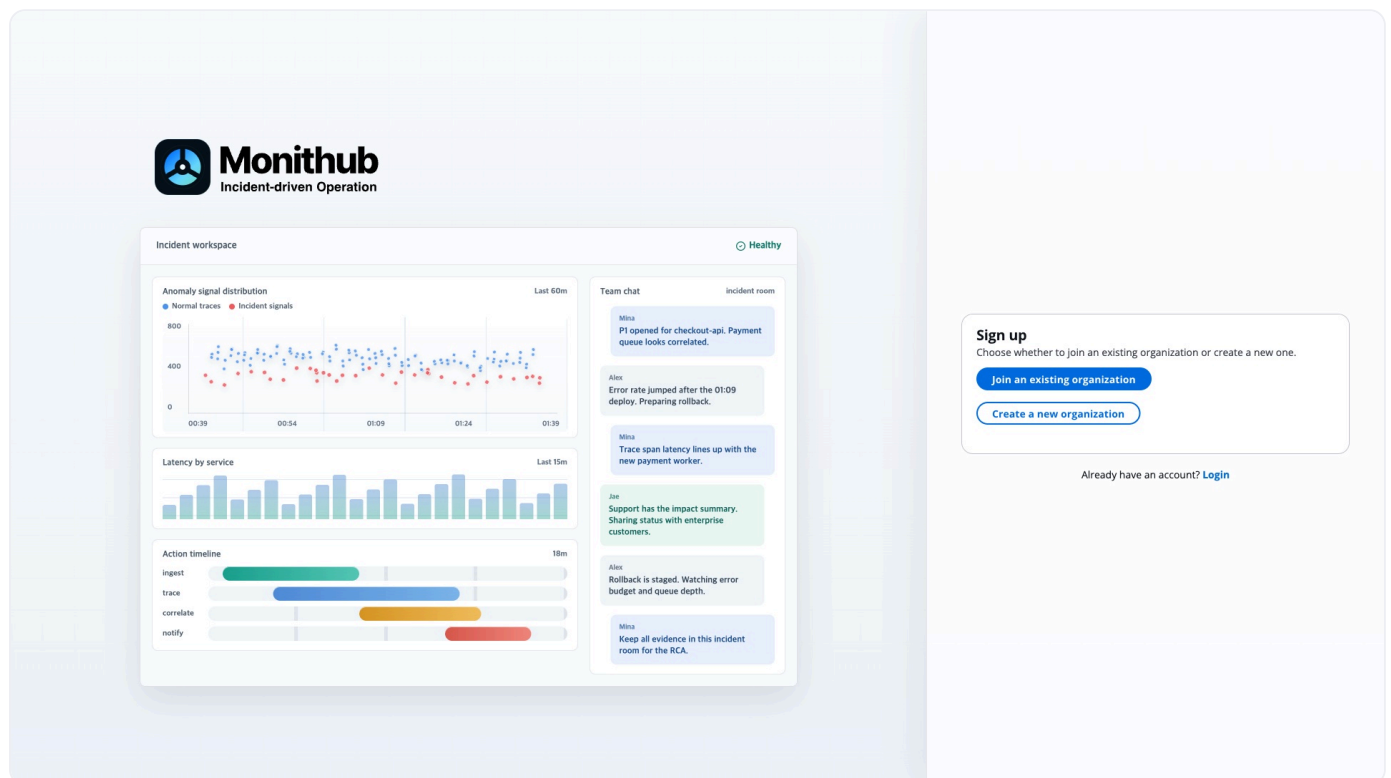
[Create a new organization](#)

관리자가 알려 준 조직에 참여하려고 함

[Join an existing organization](#)

조직 초대 메일을 이미 받음

메일의 초대 링크



새 조직을 만들면서 가입하기 [🔗](#)

1. 새 조직 만들기 선택 [🔗](#)

가입 화면에서 [Create a new organization](#)을 누릅니다. [Create organization](#) 화면이 열리면 조직과 계정 정보를 입력합니다.

입력 항목	입력할 내용
Organization name	화면에 표시할 팀이나 회사 이름
Organization slug	조직 주소에 사용할 고유한 이름
Email	인증 메일을 받을 이메일 주소
Display name	다른 멤버에게 표시할 이름. 비워 두면 이메일의 @ 앞부분이 사용됩니다.
Password	8자 이상의 비밀번호
Confirm password	위에서 입력한 비밀번호

[Organization slug](#)에는 영문 소문자, 숫자, 하이픈(-)만 사용할 수 있습니다. 조직 이름을 입력하면 추천값이 자동으로 채워지며, 필요하면 직접 바꿀 수 있습니다.

2. 인증 메일 받기 [🔗](#)

입력을 마치면 [Send verification email](#)을 누릅니다. [Check your email](#) 화면이 보이면 입력한 이메일의 받은편지함을 확인합니다.

메일의 인증 링크를 열거나, 메일에 표시된 8자리 코드를 [Verification code](#)에 입력한 뒤 [Verify code](#)를 누릅니다.

- 메일이 오지 않으면 스팸함과 입력한 이메일 주소를 확인한 뒤 [Resend email](#)을 누릅니다.
- 이메일이나 조직 정보를 고쳐야 하면 [Edit details](#)를 누릅니다.
- 이미 사용 중인 [Organization slug](#)라는 안내가 나오면 다른 값을 입력합니다.

인증이 끝나고 새 조직의 대시보드가 열리면 가입과 조직 생성이 완료된 것입니다.

기존 조직에 참여 요청하기 [🔗](#)

조직 초대 링크가 없지만 관리자가 조직의 [Organization slug](#)를 알려 준 경우 다음 순서로 참여를 요청합니다.

1. 가입 화면에서 [Join an existing organization](#)을 누릅니다.
2. [Organization slug](#)에 관리자가 알려 준 값을 입력합니다.
3. [Email](#)에 초대받을 이메일 주소를 입력합니다.

4. [Send invitation email](#)을 누릅니다.

5. 다음 단계가 메일로 안내된다는 메시지가 보이면 받은편지함을 확인합니다.

화면에는 조직이나 요청 상태를 구분하지 않고 같은 안내가 표시됩니다. 초대 메일이 바로 오지 않으면 조직 관리자에게 참여 요청을 확인해 달라고 요청하세요. 관리자가 승인하면 초대 메일이 발송됩니다.

Note

전에 받은 초대 메일을 잃어버렸다면 [Organization slug](#)를 비워 두고 초대받은 이메일만 입력해 다시 요청할 수 있습니다.

초대 메일로 참여하기

초대 메일의 링크를 열면 [Organization 초대](#) 화면에서 조직 이름, 초대받은 이메일, 초대 만료일을 확인할 수 있습니다.

Monithub 계정이 없는 경우

1. [가입해서 수락](#)을 누릅니다.
2. 초대받은 이메일이 맞는지 확인합니다. 이 이메일은 다른 주소로 바꿀 수 없습니다.
3. [Display name](#), [Password](#), [Confirm password](#)를 입력합니다.
4. [Send verification email](#)을 누릅니다.
5. 메일의 링크를 열거나 8자리 코드를 입력해 이메일 인증을 마칩니다.
6. 초대받은 조직의 대시보드가 열렸는지 확인합니다.

Monithub 계정이 있는 경우

1. [로그인하여 수락](#)을 누릅니다.
2. 초대받은 이메일과 같은 계정으로 로그인합니다.
3. 초대 화면으로 돌아오면 [초대 수락](#)을 누릅니다.
4. “초대를 수락했습니다”라는 메시지가 보이면 [대시보드로 이동](#)을 누릅니다.

⚠ Caution

다른 이메일의 계정으로 로그인하면 초대를 수락할 수 없습니다. 로그아웃한 뒤 초대 메일을 받은 계정으로 다시 로그인하세요.

초대가 진행되지 않을 때 [🔗](#)

화면 또는 상황

다음에 할 일

인증 메일이나 초대 메일이 보이지 않음

스팸함을 확인하고 이메일 주소가 맞는지 확인합니다. 계속 오지 않으면 조직 관리자에게 새 초대를 요청합니다.

만료된 초대입니다. 또는 취소된 초대입니다.

기존 링크 대신 조직 관리자에게 새 초대 메일을 요청합니다.

이미 수락된 초대입니다.

대시보드로 이동을 눌러 조직 화면을 확인합니다.

이미 이 organization의 멤버입니다.

대시보드로 이동을 누릅니다.

“이 초대는 다른 이메일 주소로 발송되었습니다.”

현재 계정에서 로그아웃하고 초대받은 이메일의 계정으로 다시 로그인합니다.

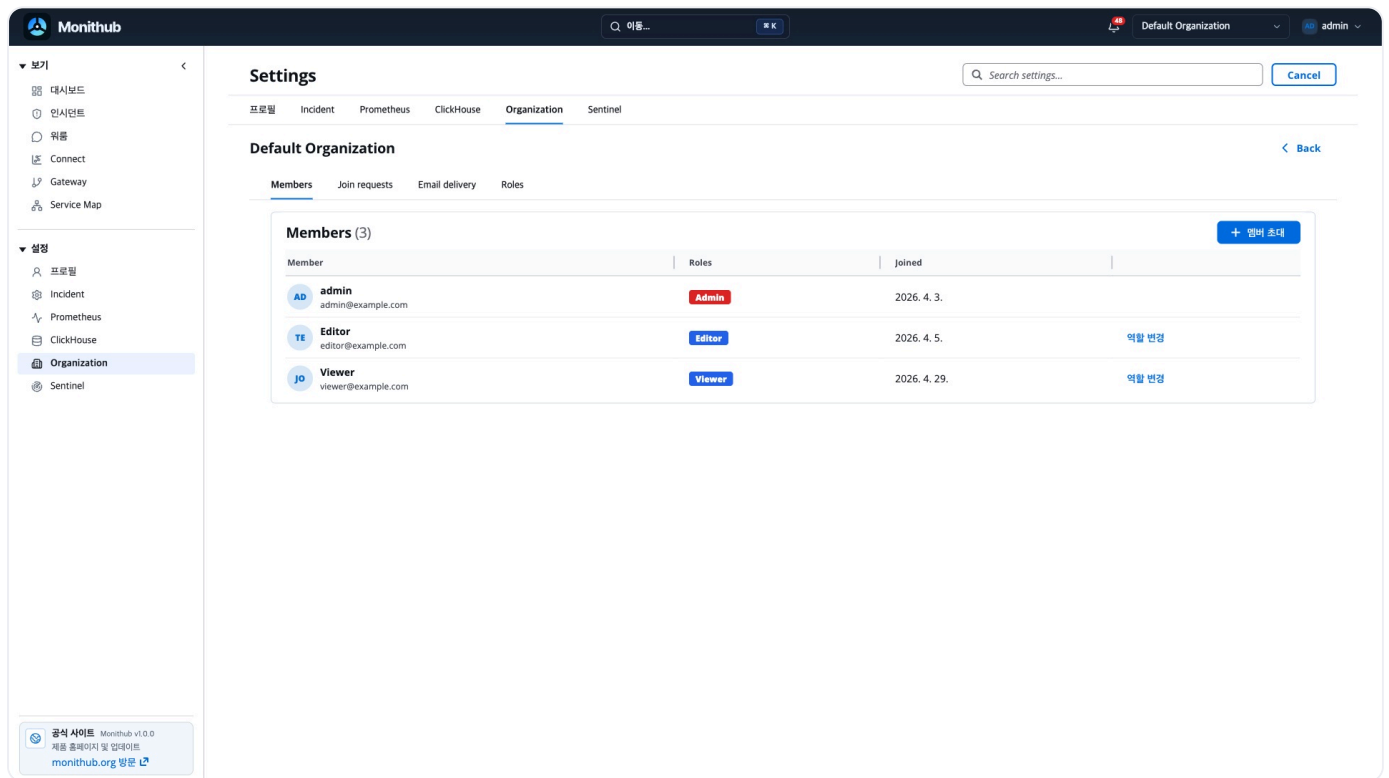
조직 관리자가 멤버를 직접 초대하거나 참여 요청을 처리하는 방법은 [멤버 초대와 참여 요청](#)에서 확인할 수 있습니다.

멤버 초대와 참여 요청

조직 관리자는 이메일로 멤버를 직접 초대하거나, 가입 화면에서 들어온 참여 요청을 확인할 수 있습니다. 두 방법 모두 상대방이 초대 메일을 열고 수락해야 조직 참여가 완료됩니다.

조직의 멤버 화면 열기 [🔗](#)

1. Monithub에 로그인한 뒤 우측 상단의 사용자 메뉴를 엽니다.
2. **Settings**를 누릅니다.
3. 설정 메뉴에서 **Organization**을 선택합니다.
4. **My Organizations**에서 관리할 조직의 **Manage →**를 누릅니다.
5. **Members** 탭이 선택되어 있는지 확인합니다.



이메일로 멤버 초대하기 [🔗](#)

1. **Members** 탭에서 **멤버 초대**를 누릅니다.
2. **Email address**에 초대할 사람의 이메일을 입력합니다.
3. **Send Invitation**을 누릅니다.

4. **Invitation Sent** 창에 입력한 이메일과 “초대 메일을 발송했습니다”라는 메시지가 보이는지 확인합니다.
5. **Done**을 눌러 창을 닫습니다.

초대를 보낸 직후에는 상대방이 멤버 목록에 나타나지 않습니다. 상대방이 메일의 링크에서 초대를 수락한 뒤 **Members** 목록을 다시 확인하세요.

Note

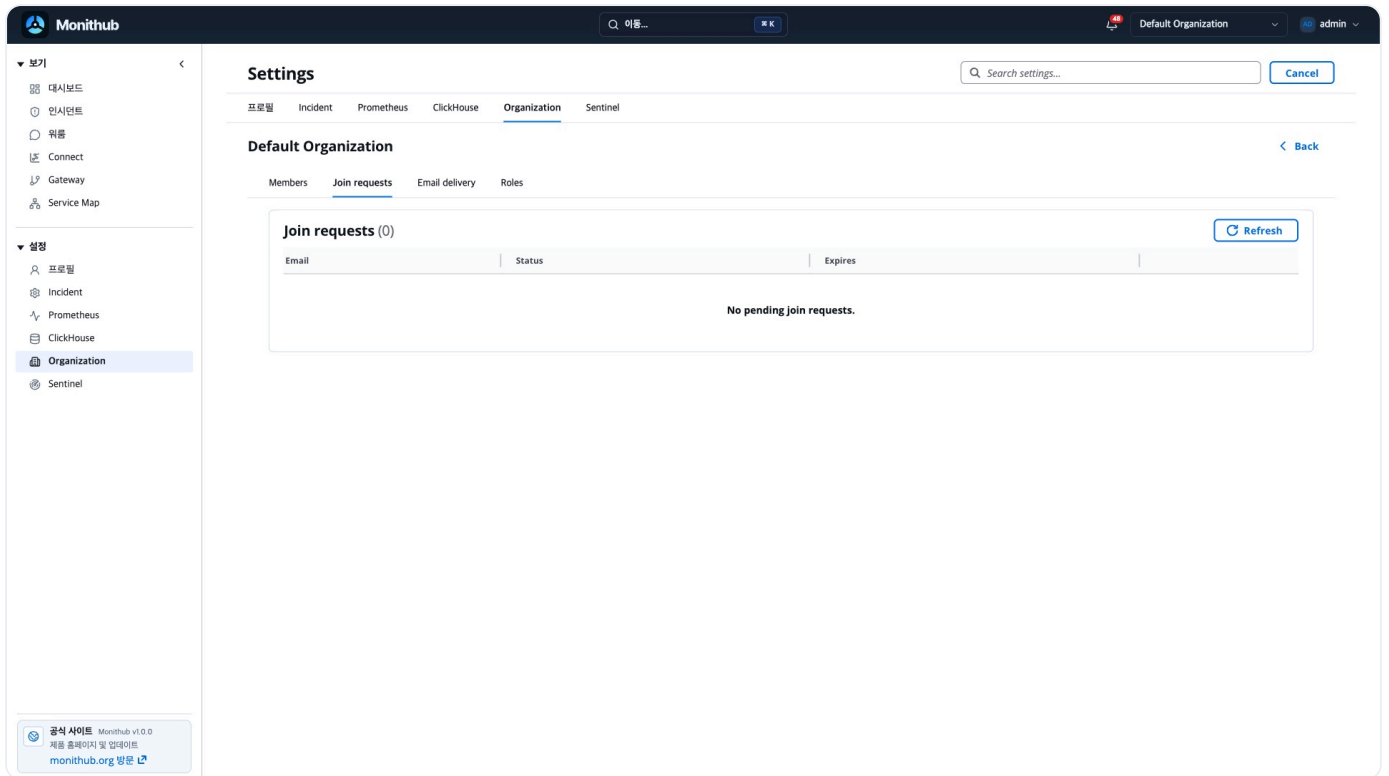
“이미 대기 중인 초대가 있는 이메일입니다”라는 안내가 나오면 같은 주소로 새 초대를 만들 수 없습니다. 초대받은 사람은 가입 화면에서 **Join an existing organization**을 선택하고 이메일을 입력해 다음 단계 메일을 다시 요청할 수 있습니다.

초대받은 사람이 계정 유무에 따라 진행하는 방법은 [가입과 조직 참여](#)에서 확인할 수 있습니다.

참여 요청 확인하기

가입 화면에서 조직의 **Organization slug**와 이메일을 입력한 사용자의 요청은 관리자의 **Join requests** 탭에 표시됩니다.

1. **Settings → Organization**에서 관리할 조직의 **Manage →**를 누릅니다.
2. **Join requests** 탭을 누릅니다.
3. 요청한 이메일, 요청 시각, 상태, 만료 시각을 확인합니다.
4. 조직에 참여해도 되는 사용자라면 **Approve**, 그렇지 않으면 **Reject**를 누릅니다.



승인한 경우 [🔗](#)

“Invitation email sent”라는 메시지가 보이고 해당 요청이 목록에서 사라지면 처리가 완료된 것입니다. 승인된 이메일로 초대 메일이 발송됩니다.

승인은 멤버 추가를 즉시 완료하지 않습니다. 요청한 사람이 메일의 초대 링크를 열어 로그인 또는 가입한 뒤 [초대 수락](#)까지 마쳐야 [Members](#) 목록에 나타납니다.

거절한 경우 [🔗](#)

거절되었다는 메시지가 보이고 해당 요청이 목록에서 사라지는지 확인합니다. 잘못 거절했다면 사용자가 가입 화면에서 다시 참여를 요청해야 합니다.

요청이 보이지 않거나 처리할 수 없을 때 [🔗](#)

화면 또는 상황

다음에 할 일

[Join requests](#) 탭이 보이지 않음

조직 관리자 계정으로 로그인했는지 확인합니다.

[No pending join requests.](#)가 표시됨

대기 중인 요청이 없습니다. 요청자에게 [Organization slug](#)와 이메일을 확인한 뒤 다시 요청해 달라고 안내합니다.

승인 후에도 멤버 목록에 사용자가 없음

요청자에게 초대 메일을 열고 수락까지 마쳤는지 확인해 달라고 안내합니다.

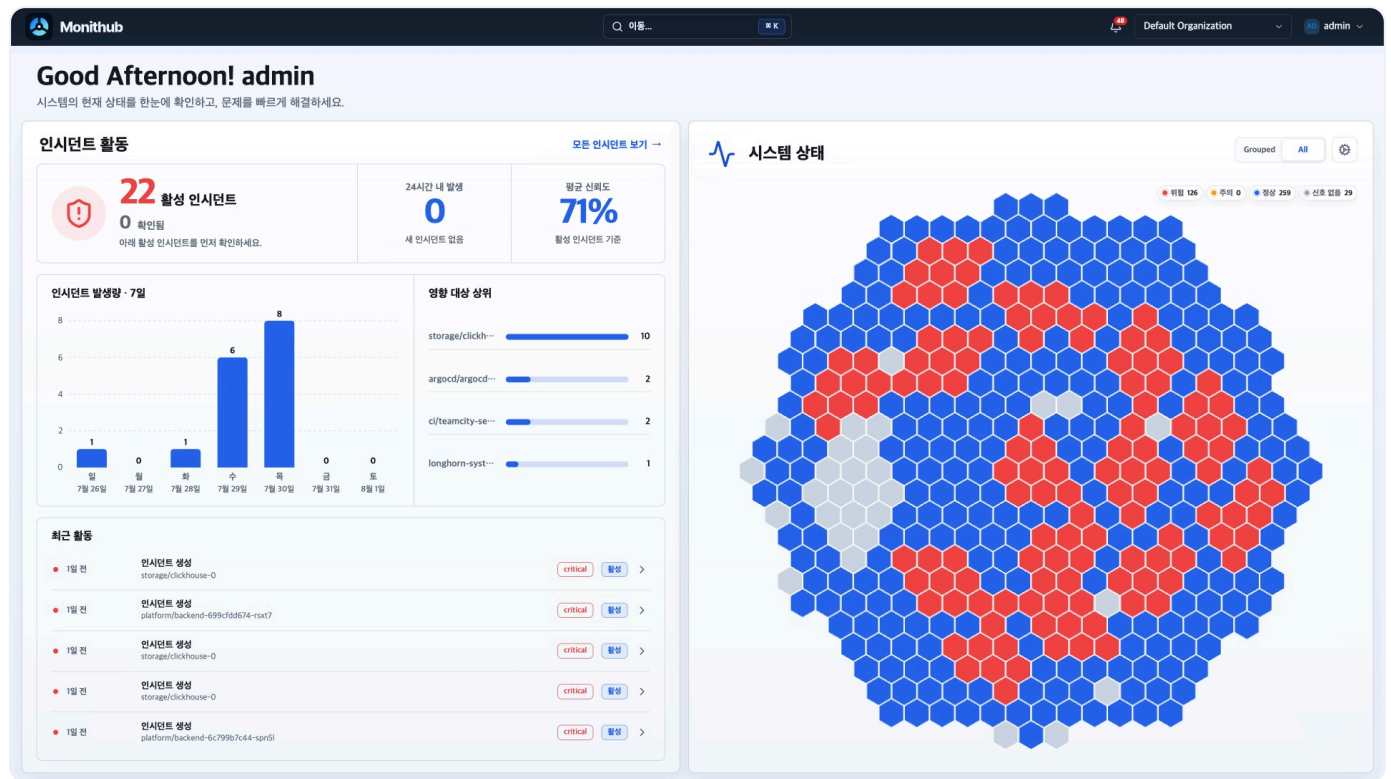
초대 메일이 도착하지 않음

요청자에게 스팸함과 입력한 이메일 주소를 확인하도록 안내합니다. 필요하면 가입 화면에서 다음 단계 메일을 다시 요청합니다.

처리가 끝나면 [Members](#) 탭으로 돌아가 새 멤버의 이름과 이메일이 목록에 표시되는지 확인합니다.

홈과 기본 탐색

홈은 Monithub에 들어온 직후 가장 먼저 보는 운영 콘솔입니다. 현재 조직에서 발생한 인시던트 활동과 리소스 상태를 한 화면에서 요약합니다.



상단 바 [🔗](#)

영역 설명

빠른 이동	이동... 을 누르거나 ⌘ K 를 입력해 메뉴와 대시보드를 찾습니다.
알림	읽지 않은 알림 수를 표시합니다.
조직 선택	현재 보고 있는 조직을 바꿉니다. 조직을 바꾸면 주요 데이터 범위도 바뀝니다.
사용자 메뉴	로그인 사용자와 계정 메뉴를 확인합니다.

인시던트 활동 [🔗](#)

인시던트 활동은 현재 열린 인시던트와 최근 변화량을 빠르게 판단하는 영역입니다.

영역	확인할 내용
Active incidents	현재 해결되지 않은 인시던트 수를 확인합니다.
Acknowledged	담당자가 확인한 인시던트 수를 확인합니다.
Past 24h	최근 24시간에 감지된 활동량을 봅니다.
Avg confidence	육인 이상 신호의 평균 신뢰도를 참고합니다.
인시던트 발생량 · 7일	최근 7일 동안 인시던트가 증가하거나 감소한 시점을 봅니다.
영향 대상 상위	반복해서 영향을 받은 리소스를 우선 확인합니다.
최근 활동	상태 변경, 담당자 지정 등 최근 대응 이력을 확인합니다.

시스템 상태

시스템 상태는 조직의 리소스를 계층형 도형으로 묶어 보여 줍니다. 색상은 **위험**, **주의**, **정상**, **신호 없음** 상태를 나타냅니다.

- **Grouped**는 리소스를 그룹별로 묶어 보고, **All**은 전체를 한 번에 펼쳐 봅니다.
- 중앙 또는 바깥쪽 조각을 선택해 해당 그룹과 하위 리소스 상태를 좁혀 봅니다.
- **Options**에서 표시 기준을 조정한 뒤 상태 분포를 다시 확인합니다.

좌측 메뉴

좌측 아이콘 메뉴는 모든 주요 화면으로 이동하는 기본 경로입니다.

1. 메뉴가 접혀 있으면 상단의 펼치기 버튼을 누릅니다.
2. **대시보드**, **인시던트**, **위험**, **Connect**, **Gateway**, **Service Map** 중 필요한 화면을 선택합니다.
3. 대시보드 화면에서는 좌측 목록에서 원하는 대시보드도 바로 선택할 수 있습니다.

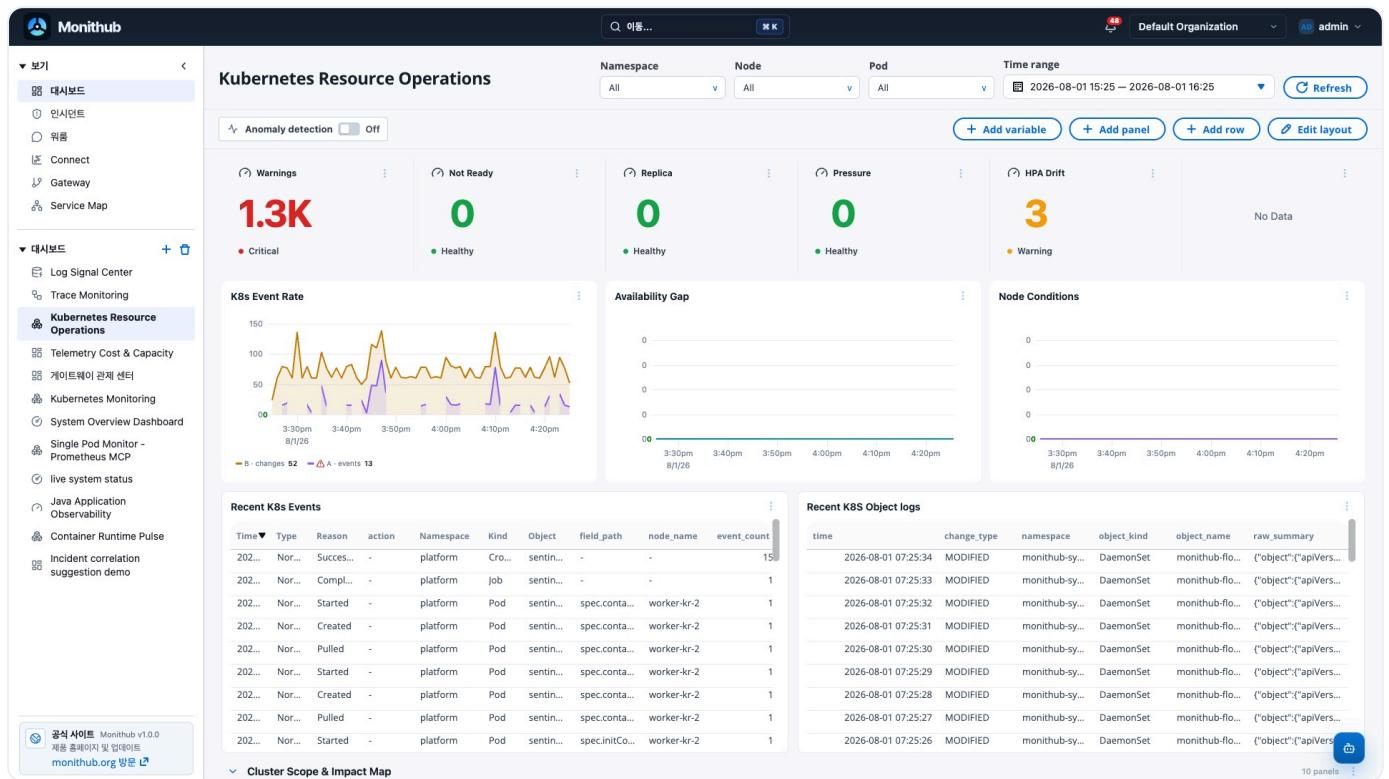
홈에서 바로 판단할 것

- 7일 추이에서 인시던트가 급증한 구간이 있는지 확인합니다.
- **영향 대상 상위**에 중요한 리소스가 반복해서 등장하는지 봅니다.

- 시스템 상태에 **위험** 또는 **주의** 조각이 있는지 확인합니다.
- 최근 활동에서 아직 담당자가 없거나 장시간 이어진 인시던트를 찾습니다.

대시보드

대시보드는 운영 지표를 패널 단위로 모아 보는 화면입니다. 왼쪽 목록에서 대시보드를 선택하고 상단 변수와 시간 범위로 필요한 데이터만 좁혀 봅니다.



대시보드 선택

좌측 대시보드 목록에서 원하는 대시보드를 선택합니다. 확인한 화면에는 아래 대시보드가 보였습니다.

- Kubernetes Resource Operations
- Trace Monitoring
- Log Signal Center
- Telemetry Cost & Capacity
- Single Pod Monitor - Prometheus MCP
- 게이트웨이 관제 센터
- System Overview Dashboard
- Container Runtime Pulse
- Java Application Observability
- live system status
- Incident correlation suggestion demo

필터와 시간 범위

대시보드 상단 필터에서 데이터를 좁힙니다.

필터	설명
Namespace	Kubernetes namespace 기준으로 패널 데이터를 좁힙니다.
Node	특정 노드만 봅니다.
Pod	특정 Pod만 봅니다.
Time range	조회 시간을 바꿉니다.
Refresh	현재 필터와 시간 범위로 데이터를 다시 불러옵니다.

패널 읽기

대시보드는 row와 panel로 구성됩니다.

- 큰 숫자 패널은 현재 상태를 즉시 판단하는 데 사용합니다.
- 차트 패널은 시간 흐름에 따른 변화와 급증 구간을 봅니다.
- 테이블 패널은 이벤트, 로그, object 변경 내역처럼 행 단위 근거를 확인합니다.
- 패널 우측 메뉴에서 세부 동작을 열 수 있습니다.

이상탐지

Anomaly detection 토글이 켜져 있으면 지원하는 패널에 이상탐지 결과가 겹쳐 표시됩니다. 결과가 갱신 중이면 현재 시간 범위와 변수를 유지한 채 잠시 뒤 **Refresh**로 다시 확인합니다.

편집 기능

권한이 있으면 상단 버튼으로 대시보드를 수정할 수 있습니다.

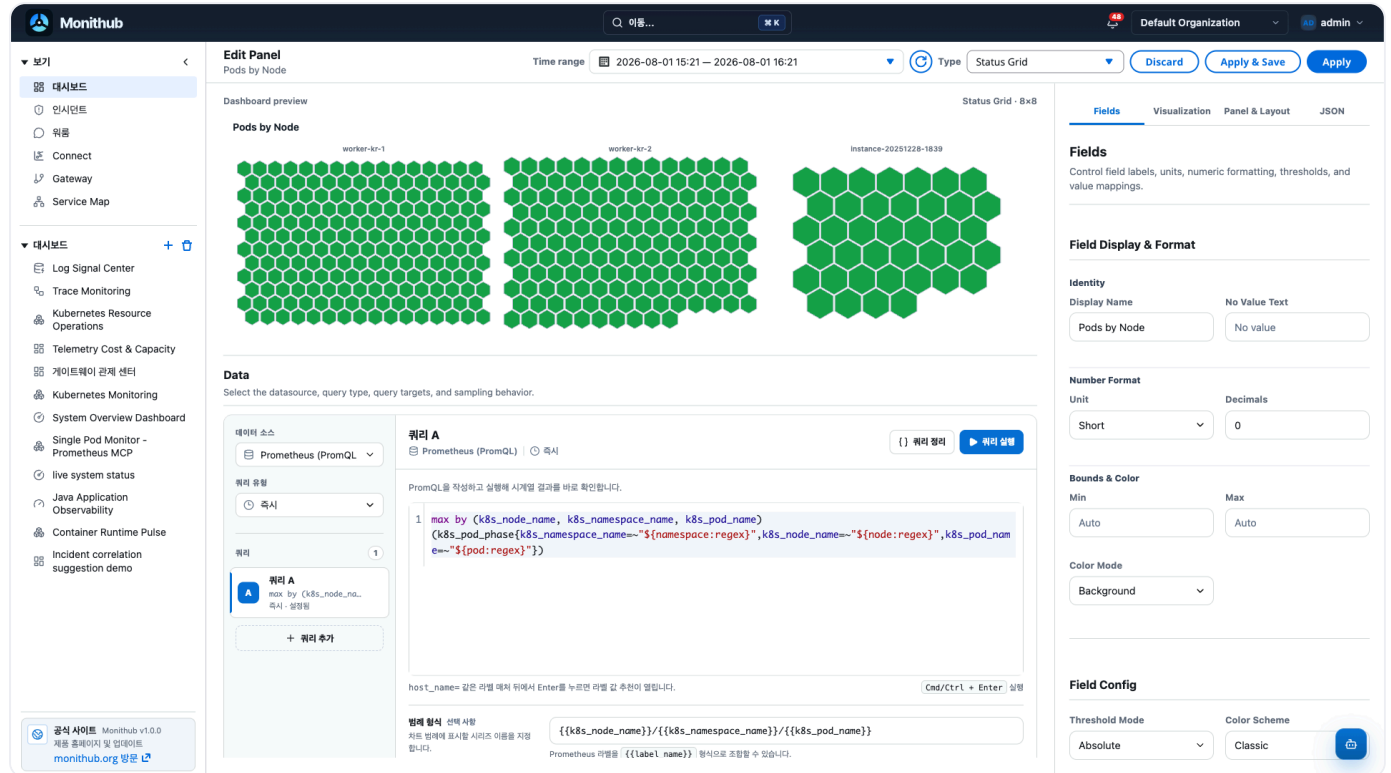
버튼	용도
Add variable	Namespace, Node, Pod 같은 재사용 필터를 추가합니다.
Add panel	새 차트, 숫자, 테이블 패널을 추가합니다.
Add row	패널 묶음을 추가합니다.
Edit layout	패널 배치를 편집합니다.

패널 타입별 쿼리, 표시 형식, 시각화 옵션은 [패널 편집](#) 문서에서 확인합니다.

운영 중인 대시보드를 수정할 때는 먼저 현재 필터와 시간 범위가 맞는지 확인한 뒤 저장합니다.

패널 편집

패널 편집 화면은 대시보드 패널의 데이터 쿼리, 값 표시 형식, 시각화 옵션, 레이아웃, JSON 설정을 한 곳에서 조정하는 화면입니다. 타입별 조작법은 하위 문서로 분리했고, 예시는 현재 조직의 Kubernetes, 로그, 트레이스 대시보드 패널을 기준으로 캡처했습니다.



진입 방법 [🔗](#)

대시보드에서 기존 패널 우측 상단의 **Panel actions** 메뉴를 열고 **Edit**를 선택합니다. 새 패널을 만들 때는 대시보드 상단의 **Add panel**을 사용합니다. 패널 묶음은 **Add row**로 만들며, row는 일반 지표 패널과 달리 데이터 쿼리보다 그룹 접기와 배치가 중심입니다.

편집 화면은 크게 다섯 영역으로 나뉩니다.

영역	역할
상단 바	패널 제목, 편집용 시간 범위, 새로고침, 패널 타입, 저장/취소 버튼을 제공합니다.
미리보기	현재 draft 설정으로 패널이 어떻게 보이는지 확인합니다.
Data	datasource, query type, query target, query 실행 테스트를 조작합니다.
오른쪽 탭	Fields , Visualization , Panel & Layout , JSON 설정을 전환합니다.
저장 버튼	Apply , Apply & Save , Discard 로 draft 반영 범위를 정합니다.

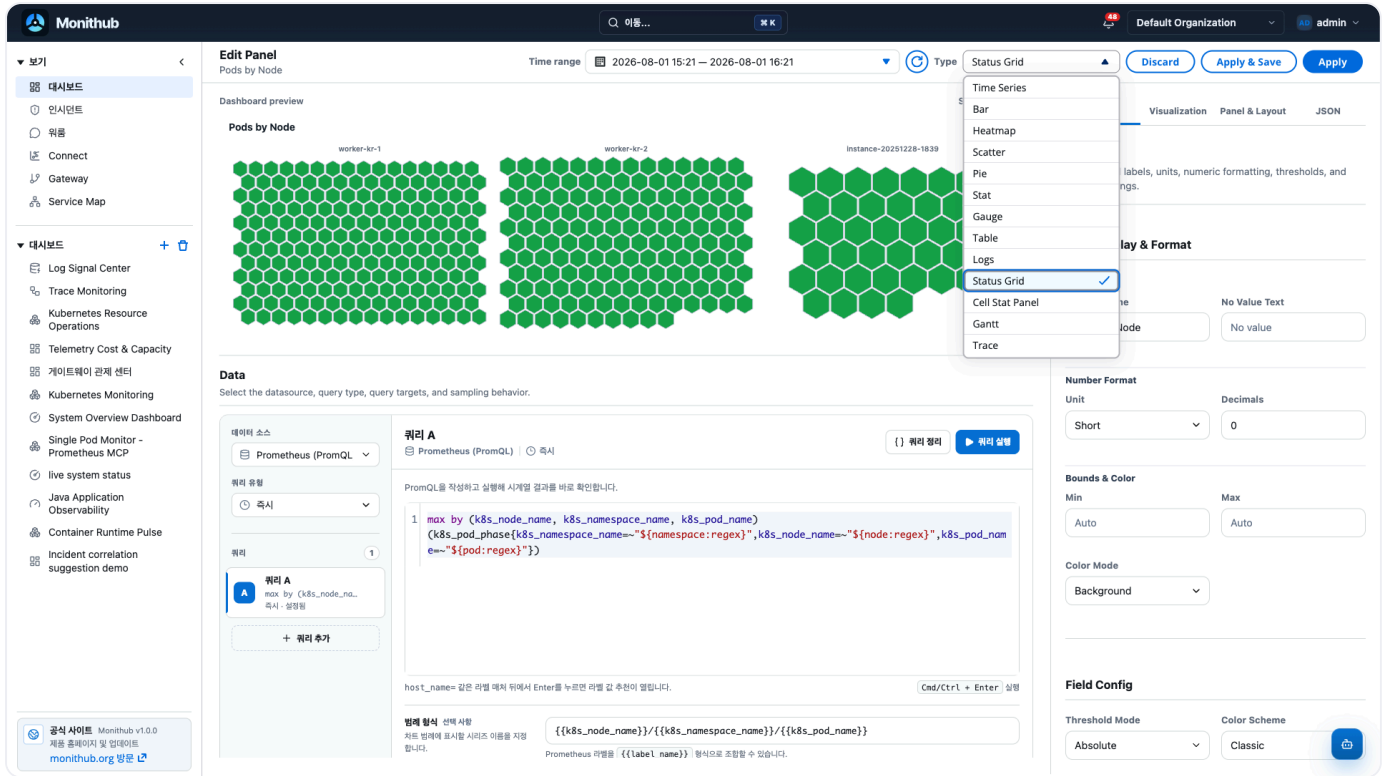
저장과 취소

버튼	동작
Discard	현재 편집 draft를 버리고 편집 화면을 닫습니다.
Apply	편집 화면의 draft를 현재 대시보드 화면에 반영합니다. 대시보드 저장까지 필요한 화면에서는 별도 저장 흐름을 확인합니다.
Apply & Save	draft를 적용하고 backend 저장까지 요청합니다.
Refresh	현재 편집용 시간 범위로 미리보기와 쿼리 결과를 다시 불러옵니다. 패널 정의 자체를 저장하지는 않습니다.

패널 타입을 바꾸면 같은 패널 id 안에서 모델이 새 타입에 맞게 정규화됩니다. 예를 들어 [Trace](#) 타입은 ClickHouse 계열 데이터에 맞게 datasource가 바뀔 수 있습니다. 타입 변경도 draft이므로 저장 버튼을 누르기 전까지 운영 대시보드에 확정되지 않습니다.

타입 선택

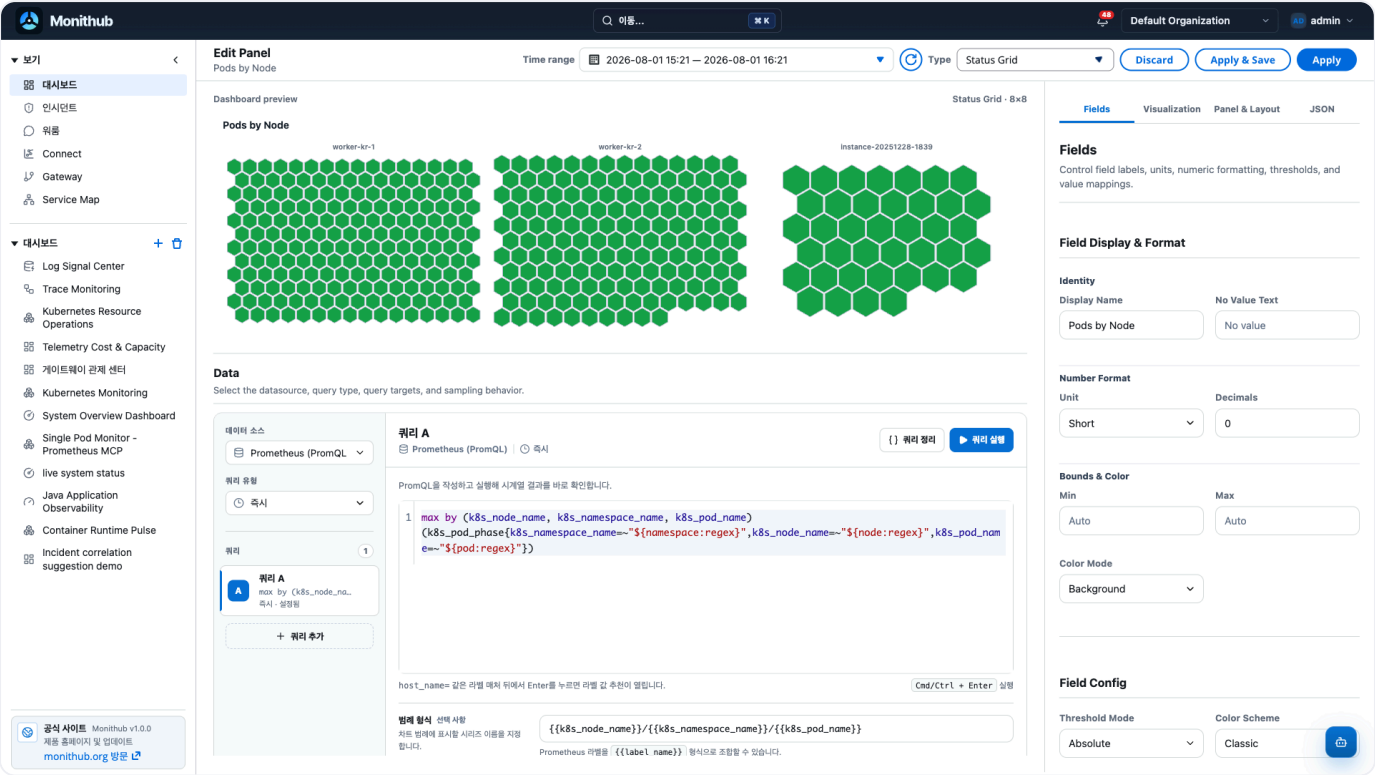
상단 [Type](#) 선택에서 패널 종류를 바꿉니다. 타입을 바꾸면 오른쪽 [Visualization](#) 탭의 입력 항목이 타입에 맞게 달라집니다.



타입

주 사용 상황

Time Series	시간 흐름에 따른 메트릭 변화, 이상탐지 overlay, series 비교를 볼 때
Bar	Time Series 계열 데이터를 막대 스타일로 비교할 때
Heatmap	시간과 bucket, label, value가 있는 밀도/분포 데이터를 볼 때
Scatter	trace duration처럼 행 단위 관측치를 x/y 점으로 볼 때
Pie	적은 수의 category 비중을 볼 때
Stat	하나 또는 여러 series의 최신값, 합계, 평균 같은 대표값을 볼 때
Gauge	단일 값이 기준 범위 안에서 어디에 있는지 볼 때
Table	로그, 이벤트, trace, ClickHouse 결과처럼 행과 열을 자세히 볼 때
Logs	로그성 텍스트 결과를 표시할 때
Status Grid	여러 대상의 정상/비정상 상태를 타일로 볼 때
Cell Stat Panel	작은 셀 단위로 여러 상태 값을 요약해 비교할 때
Gantt	시작/종료/기간이 있는 작업, span, 배치 흐름을 시간 막대로 볼 때
Trace	trace 목록과 trace 요약을 볼 때



Data 영역은 모든 데이터 패널의 출발점입니다. 먼저 datasource와 query type을 정한 뒤, query를 실행해서 반환 형태가 패널 타입과 맞는지 확인합니다.

설정	역할
Queries	패널이 사용하는 query target 목록입니다. A, B 같은 refId로 구분합니다.
Datasource	Prometheus (PromQL) 또는 ClickHouse (SQL)을 선택합니다. 상단 datasource를 바꾸면 현재 query target에도 같은 datasource가 적용됩니다.
Query type	Prometheus에서는 Instant와 Range를 고릅니다. ClickHouse는 table query로 동작합니다.
Add Query	같은 패널에 query target을 추가합니다. Time Series, Table, Status Grid처럼 여러 query를 비교하거나 join할 때 사용합니다.
Query editor	PromQL 또는 SQL을 작성합니다. 대시보드 변수와 시간 매크로를 함께 사용할 수 있습니다.
Format / snippet 버튼	PromQL/SQL 포맷 정리, trace query template, group by, time filter, limit snippet을 넣습니다.
Run Query	저장하지 않고 현재 query를 실행해 결과 형태를 확인합니다.

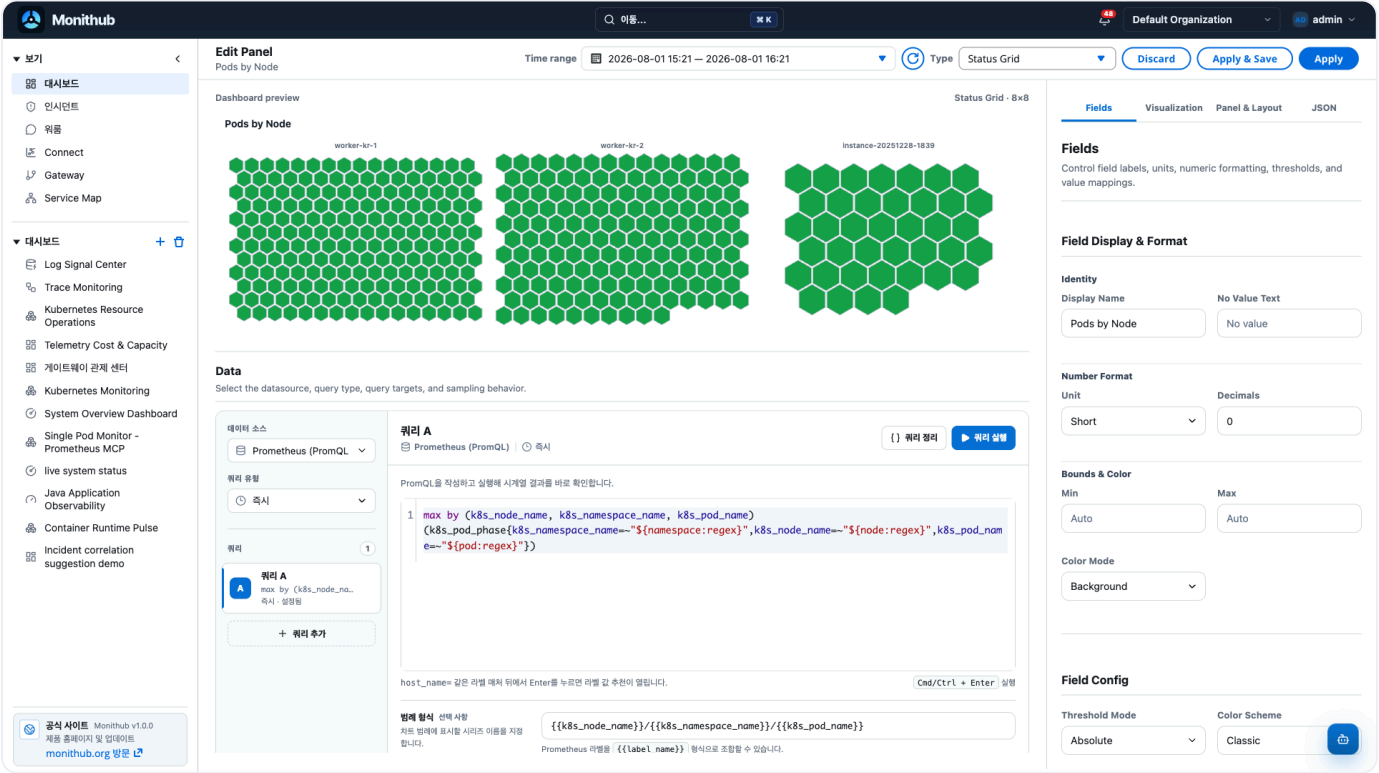
ClickHouse 쿼리는 "\$_fromTime", "\$_toTime" 매크로를 지원합니다. 이 값은 편집 화면 상단의 Time range 기준으로 치환됩니다. Prometheus range query는 시간 범위와 start/end가 중요하고, instant query는 특정 시점의 대표값을 가져옵니다.

Table 타입에서 query가 둘 이상이면 join 설정이 나타납니다.

Join 설정	설명
Enable join across queries	여러 query 결과를 하나의 table로 합칩니다.
Join Queries	join에 포함할 refId를 고릅니다. 선택하지 않으면 전체 query가 대상입니다.
Join Key Field	여러 결과를 같은 행으로 묶을 기준 field입니다.
Join Type	inner, left, outer join 중 의도에 맞는 방식을 고릅니다.

중복 설정을 피하려면 datasource는 가능한 한 query target에서 한 번에 맞춥니다. 패널 타입 전환 후 datasource가 자동 변경된 경우, query editor와 datasource 표시가 서로 맞는지 먼저 확인합니다.

Fields 탭



Fields는 값이 화면에 표시되는 방식을 정합니다. 타입별 시각화가 달라도 단위, 소수점, min/max, threshold, value mapping은 대부분 공통으로 영향을 줍니다.

Field Display & Format [🔗](#)

설정	역할
Display Name	legend, tooltip, stat title처럼 field 이름을 사람이 읽기 쉬운 이름으로 바꿉니다.
No Value Text	값이 없거나 null일 때 표시할 문구입니다. 기본값은 <code>No value</code> 입니다.
Unit	숫자 단위를 지정합니다. <code>percent</code> , <code>bytes</code> , <code>ms</code> , <code>short</code> 등으로 값 formatter가 달라집니다.
Decimals	소수점 자리수를 고정합니다. 비워두면 자동으로 정합니다.
Min / Max	값 범위를 정합니다. Gauge, axis, threshold 색상 판단의 기준과 함께 쓰입니다.
Color Mode	값 자체 색상, 배경 색상, 색상 미사용 여부를 정합니다.

`Unit`, `Decimals`, `Min`, `Max`, `Display Name`, `No Value Text`는 내부적으로 `fieldConfig.defaults`에도 같이 반영됩니다. 같은 의미의 값을 JSON에서 따로 바꾸면 UI 값과 JSON 값이 어긋날 수 있으므로, 일반 편집에서는 Fields 탭을 기준으로 관리합니다.

Field Config [🔗](#)

설정	역할
Threshold Mode	threshold step 값을 절대값으로 볼지 percentage로 볼지 정합니다.
Color Scheme	threshold나 value 색상 기본 팔레트를 정합니다.
Series Palette	Time Series/Bar 계열에서 여러 series 색상 순서를 정합니다.
Threshold Steps	값 범위별 색상을 순서대로 정의합니다. 각 step은 해당 값 이상부터 적용됩니다.
Value Mappings	특정 값 또는 범위를 텍스트와 색상으로 바꿉니다. 상태 코드, boolean, sentinel 상태를 사람이 읽는 문구로 바꿀 때 사용합니다.

중복 관계는 아래 원칙으로 정리합니다.

상황	권장 기준
Field Unit과 Axis Unit이 모두 있는 경우	값 표시와 tooltip은 Field Unit, 축 눈금은 Axis Unit을 따릅니다. 특별한 이유가 없으면 같은 단위로 맞추습니다.
Field Min/Max와 Axis Min/Max가 모두 있는 경우	Axis 입력은 chart 축에 직접 영향을 주며, 현재 UI는 min/max/decimals를 Field/Standard 값에도 동기화합니다. 한 곳에서 바꾼 뒤 다른 탭에서 값이 맞는지 확인합니다.
Global threshold와 Table column threshold가 모두 있는 경우	Table column threshold가 해당 column 표시를 더 구체적으로 제어합니다. 공통 기준은 Fields, 특정 column 예외는 Table column에서 둡니다.
Value Mapping과 Threshold가 모두 있는 경우	Mapping은 표시 텍스트/색상을 바꾸고, Threshold는 값 범위 기반 상태 색상을 만듭니다. 같은 값을 서로 다른 의미로 색칠하지 않도록 색상 기준을 하나로 맞추습니다.

Axes & Legend

Visualization 그룹 안에는 타입에 따라 **Axes & Scale**과 **Legend**도 함께 나타납니다.

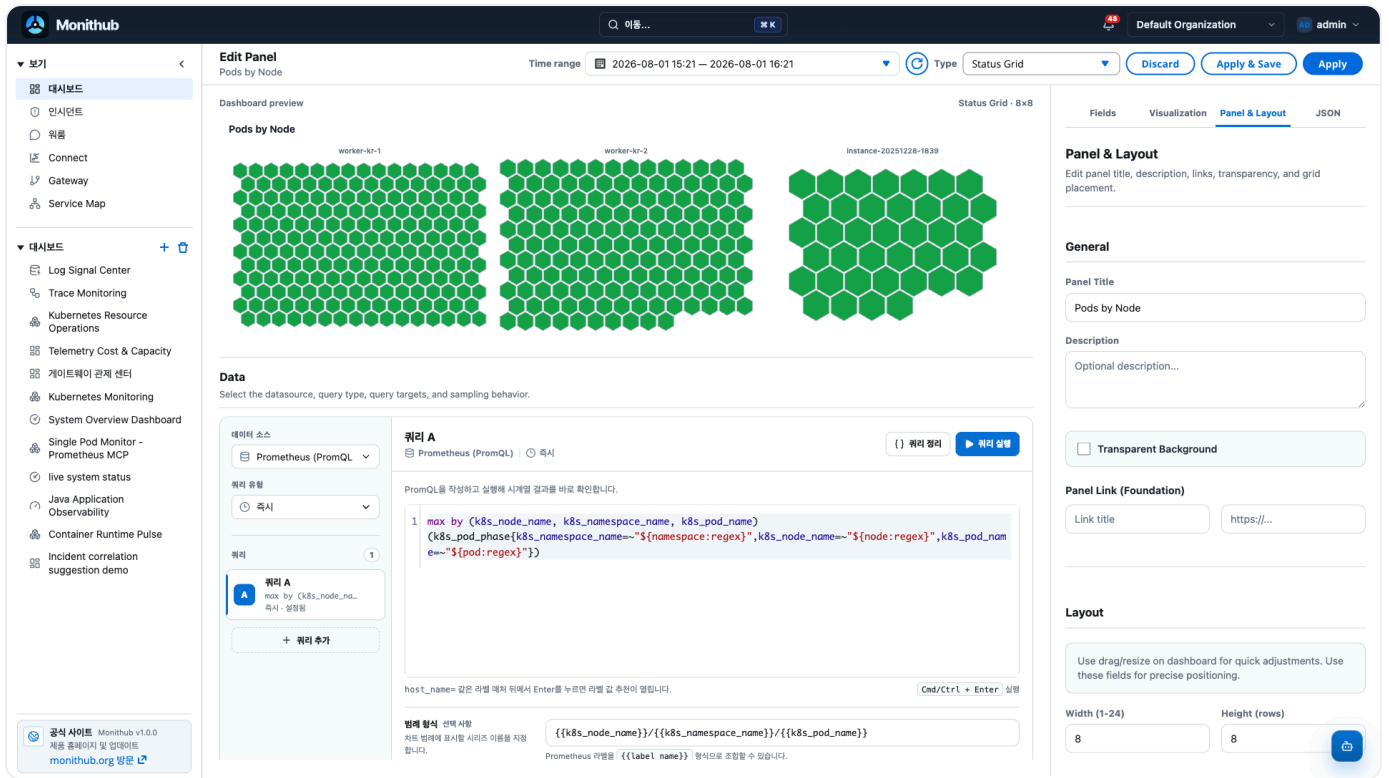
설정	적용 타입	설명
Axis Label	Bar, Scatter, Time Series, Pie, Gauge	축에 표시할 이름입니다.
Scale	Bar, Scatter, Time Series, Pie, Gauge	Linear 또는 Logarithmic 을 고릅니다. 0 또는 음수가 포함된 데이터에는 log scale을 피합니다.
Placement	Bar, Scatter, Time Series, Pie, Gauge	축을 왼쪽 또는 오른쪽에 둡니다.
Axis Unit	Bar, Scatter, Time Series, Pie, Gauge	축 눈금 단위를 정합니다. Field Unit과 맞추는 것이 기본입니다.
Decimals / Min / Max	Bar, Scatter, Time Series, Pie, Gauge	축과 값 범위의 정밀도를 정합니다.
Soft Min / Soft Max	Bar, Scatter, Time Series, Pie, Gauge	데이터가 범위를 벗어나면 자동 확장할 수 있는 느슨한 범위입니다.
Show Legend	Bar, Scatter, Time Series, Pie	legend 표시 여부입니다.
Mode	Bar, Scatter, Time Series, Pie	List 또는 Table legend를 고릅니다.
Placement	Bar, Scatter, Time Series, Pie	legend를 아래 또는 오른쪽에 둡니다.
Values	Time Series	table legend에 Last, Min, Max, Avg 같은 계산값을 column으로 표시합니다. 최소 하나는 남아야 합니다.

타입별 문서 [🔗](#)

타입별 조작 설명은 개별 페이지에서 확인합니다. 각 페이지는 query 결과 형태를 맞추는 [Data](#), 단위와 threshold를 맞추는 [Fields](#), 타입 전용 렌더링을 조정하는 [Visualization](#) 순서로 읽을 수 있게 구성했습니다.

타입 문서	대표 사용 상황	먼저 확인할 설정
Time Series	시간 흐름에 따른 메트릭 추세, series 비교	range query, draw style, null 처리, 축, legend
Bar	bucket, category, series를 막대 스타일로 비교	draw style, stack, fill, 축 단위
Stat	최신값, 평균, 합계 같은 대표값 표시	instant query, reducer, trend, color mode
Gauge	현재값이 기준 범위 안에서 어디에 있는지 표시	reducer, min/max, threshold, unit
Table	행과 열로 근거 데이터 검토	join, detected columns, cell type, filter
Heatmap	시간과 bucket별 밀도/분포 확인	x/y/value field, color scale, empty cell
Scatter	row 단위 관측치를 x/y 점으로 비교	x field, y field, series field
Status Grid	여러 대상의 정상/비정상 상태 스캔	reducer, reverse status, group, true/false text
Trace	trace 목록과 요약 확인	summary card, trace columns, column visibility
Gantt	시작/종료/기간이 있는 작업을 시간 막대로 표시	time unit, row grouping, row height
Pie	적은 수의 category 비중 표시	category query, unit, decimals, legend
Logs	로그성 텍스트 결과 표시	query, field formatting, panel layout, JSON

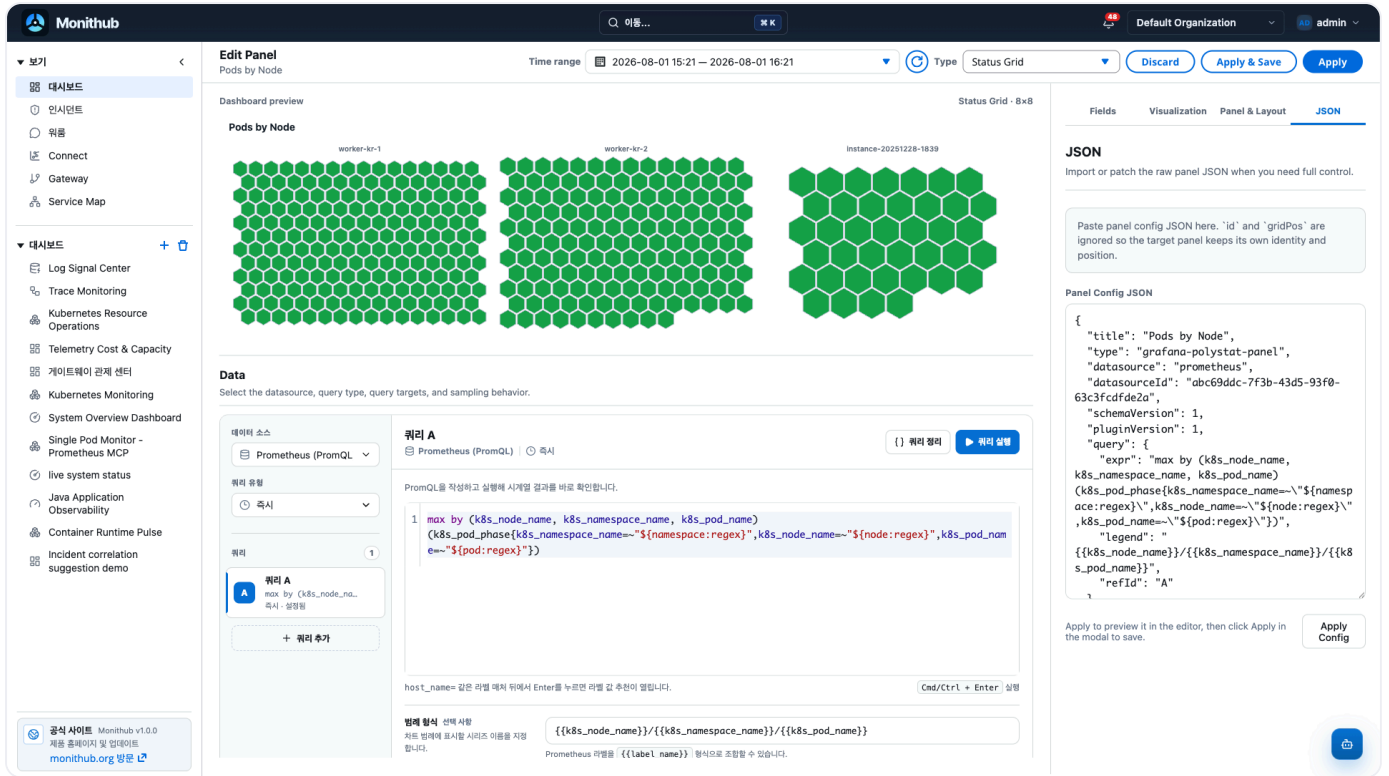
Panel & Layout



설정	역할
Panel Title	대시보드와 패널 header에 표시되는 제목입니다.
Description	패널 설명입니다. 미리보기와 문서화 목적에 사용됩니다.
Transparent Background	패널 배경 투명 여부입니다. 주변 row 배경과 맞춰야 합니다.
Panel Link	패널 메뉴에서 열 수 있는 외부 또는 내부 링크입니다.
Width	24-column grid 기준 너비입니다.
Height	row 단위 높이입니다.
X	0부터 시작하는 좌우 위치입니다.
Y	0부터 시작하는 세로 위치입니다.

빠른 배치는 대시보드의 [Edit layout](#)에서 drag/resize로 조정하고, 정확한 좌표가 필요할 때 이 탭에서 숫자를 맞춥니다. Width와 X의 합이 24를 넘지 않도록 관리합니다.

JSON [🔗](#)



JSON 탭은 UI에 아직 노출되지 않은 panel config를 가져오거나 수정할 때 사용합니다.

설정	역할
Panel Config JSON	패널 config payload를 직접 편집합니다.
Apply Config	JSON을 현재 미리보기 draft에 적용합니다.
error message	JSON parse 또는 적용 실패 이유를 표시합니다.

Apply Config는 편집기 내부 draft에만 반영합니다. 실제 대시보드에 확정하려면 이후 **Apply** 또는 **Apply & Save**를 눌러야 합니다. JSON 적용 시 **id**와 **gridPos**는 대상 패널의 identity와 위치를 유지하기 위해 무시됩니다.

JSON을 쓸 때는 아래 순서를 권장합니다.

1. 현재 패널 JSON을 백업합니다.
2. type, queryModel, visualization, options, fieldConfig 중 필요한 부분만 바꿉니다.
3. **Apply Config**로 미리보기와 오른쪽 탭 값이 맞는지 확인합니다.
4. **Apply** 또는 **Apply & Save**로 확정합니다.

Row와 Text 패널 [🔗](#)

Row 패널은 대시보드의 **Add row**로 추가하는 패널 묶음입니다. 일반 지표 패널처럼 query를 실행하지 않고, row 제목과 접힘 상태, 묶인 패널 배치가 핵심입니다. Row 편집에서는 **General**, **Layout**, **JSON** 중심으로 확인합니다.

Text 패널은 현재 타입 선택 메뉴에는 기본 노출되지 않을 수 있습니다. 기존 대시보드에 Text 패널이 있다면 markdown/text 표시 모드가 JSON에 들어있을 수 있으므로, **Panel & Layout**과 **JSON**을 함께 확인합니다.

추천 편집 순서

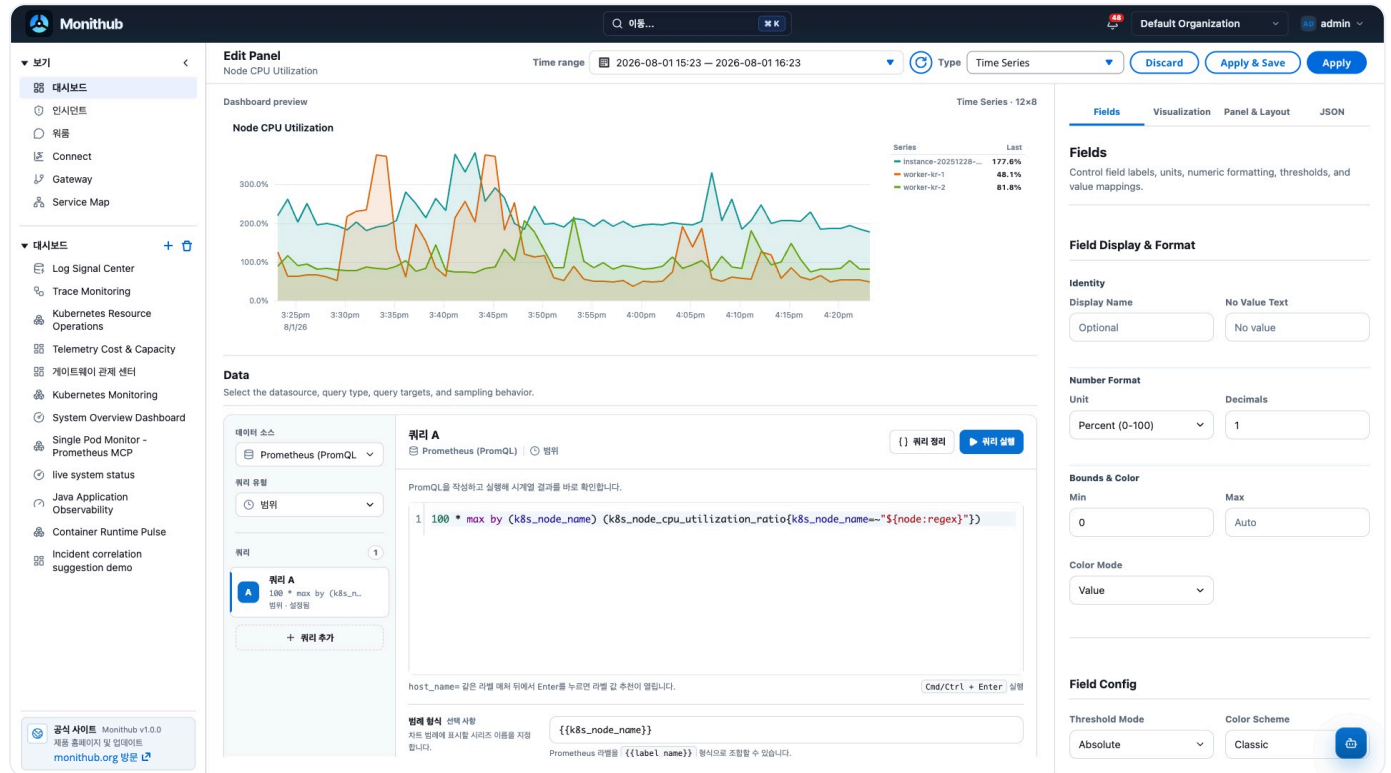
1. **Type**을 먼저 고릅니다.
2. **Data**에서 datasource와 query 결과 형태를 맞춥니다.
3. **Fields**에서 unit, decimals, threshold, value mapping을 맞춥니다.
4. **Visualization**에서 타입별 표현 방식을 조정합니다.
5. **Axes & Scale**과 **Legend**가 있는 타입은 축과 legend를 맞춥니다.
6. **Panel & Layout**에서 제목, 설명, 링크, grid 위치를 정리합니다.
7. 필요한 경우에만 **JSON**을 사용합니다.
8. 미리보기를 확인한 뒤 **Apply & Save**로 저장합니다.

운영 대시보드에서는 query 변경과 type 변경을 한 번에 크게 바꾸지 않는 것이 좋습니다. 먼저 query 결과가 원하는 형태인지 확인하고, 그 다음 시각화와 field 설정을 조정하면 원인을 추적하기 쉽습니다.

Time Series 패널

Time Series는 시간 흐름에 따른 메트릭 변화, 이상탐지 overlay, 여러 series 비교를 볼 때 사용합니다. 예시는 [Node CPU Utilization](#) 편집 화면입니다.

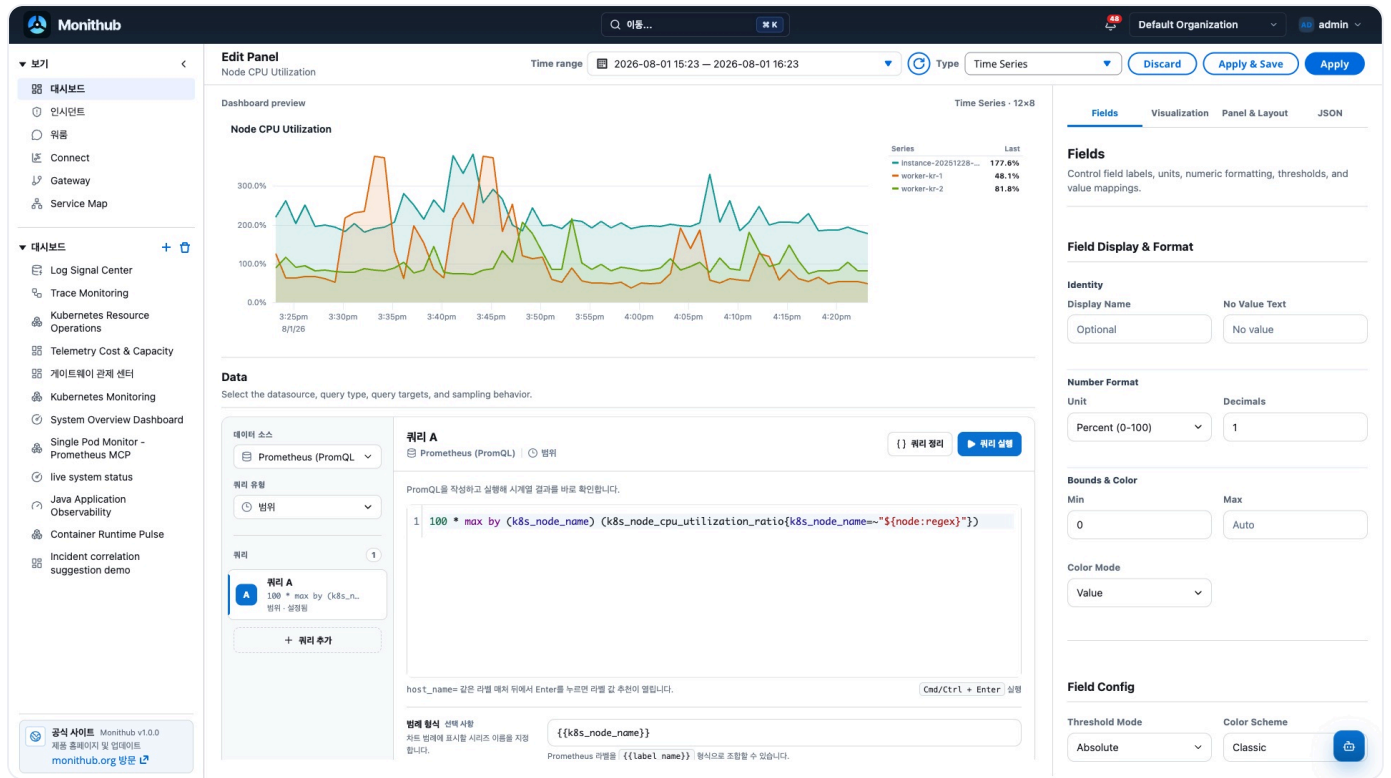
Data [🔗](#)



Data에서는 시간축을 만들 query를 먼저 맞춥니다. 추세를 보는 패널이므로 Prometheus는 보통 [Range](#) query를 사용하고, ClickHouse를 쓰면 시간 bucket column이 결과에 포함되어야 합니다.

설정	역할
Datasource	Prometheus (PromQL) 또는 ClickHouse (SQL) 을 선택합니다.
Query type	추세는 Range , 현재값은 Instant 를 사용합니다.
Queries	여러 query를 추가해 series를 겹쳐 비교합니다.
Query editor	PromQL/SQL을 작성하고 Run Query 로 반환 series를 확인합니다.
Legend Format	<code>{{pod}}</code> , <code>{{service}}</code> 처럼 legend 이름을 읽기 쉽게 정합니다.

Fields [🔗](#)



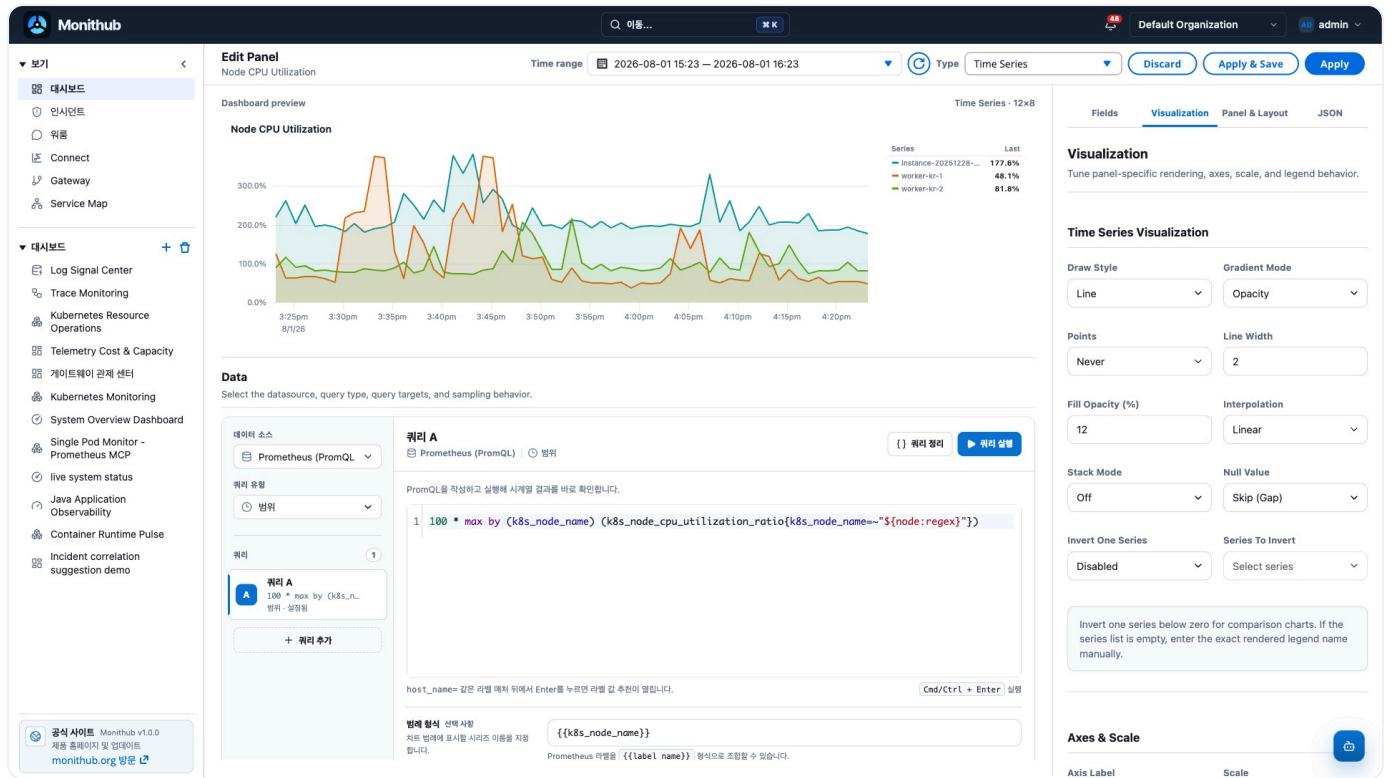
Fields는 line 값이 어떤 단위와 상태 색으로 보이는지를 정합니다. 축 단위와 tooltip 단위가 다르면 운영자가 값을 잘못 읽기 쉬우므로 여기서 기본 포맷을 먼저 맞춥니다.

설정

역할

Display Name / No Value Text	legend, tooltip, 빈 값 표시를 정리합니다.
Unit / Decimals	tooltip, legend table, axis와 맞춰야 하는 기본 단위입니다.
Min / Max	값 범위와 threshold 판단 기준입니다.
Threshold Steps	warning/critical 같은 값 구간 색상을 정의합니다.
Value Mappings	특정 값이나 범위를 텍스트와 색상으로 바꿉니다.
Series Palette	여러 series의 색상 순서를 정합니다.

Visualization [🔗](#)



Visualization은 같은 query 결과를 어떤 그래프 형태로 그릴지 정합니다. 값 자체의 의미는 Fields에서, 선/점/면/stack 표현은 여기서 조정합니다.

설정	역할
Draw Style	line, bars, points 중 표현 방식을 고릅니다.
Gradient Mode	단색, 투명도 gradient, hue gradient를 선택합니다.
Points / Line Width	점 표시와 선 두께를 조정합니다.
Fill Opacity	area처럼 채우는 정도입니다.
Interpolation	linear, smooth, step before, step after 중 선 연결 방식을 고릅니다.
Stack Mode	series를 쌓아서 총량 또는 비중을 비교합니다.
Null Value	누락 구간을 0으로 채울지, 끊을지, 연결할지 정합니다.
Invert One Series	inbound/outbound처럼 한 series를 0 아래로 뒤집습니다.

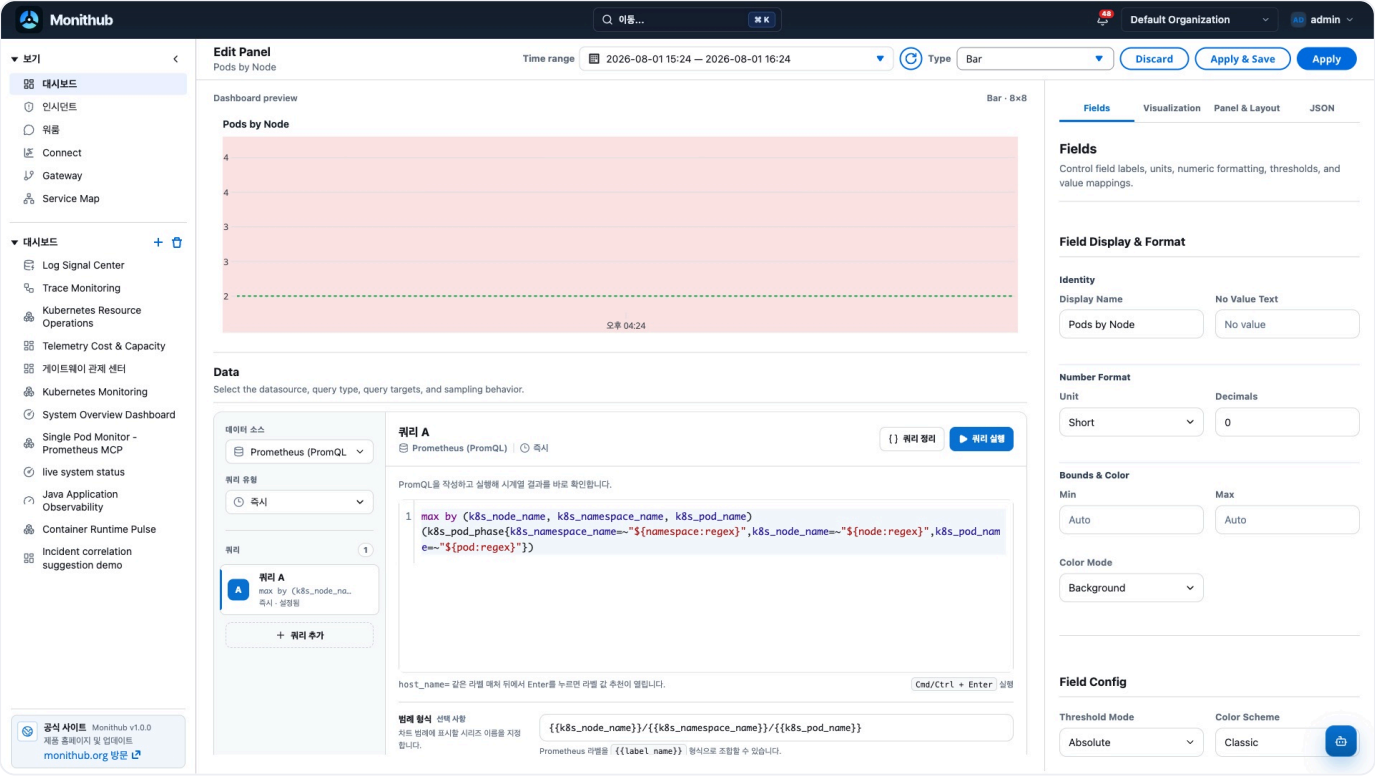
확인 순서 [🔗](#)

먼저 Data에서 range query와 legend를 안정화하고, Fields에서 unit과 threshold를 맞춘 뒤, Visualization에서 null 처리와 stack 여부를 정합니다.

Bar 패널

Bar는 Time Series 계열 데이터를 막대 스타일로 비교할 때 사용합니다. 같은 query라도 막대 화면에서는 구간별 크기와 series 간 비교가 더 중요합니다.

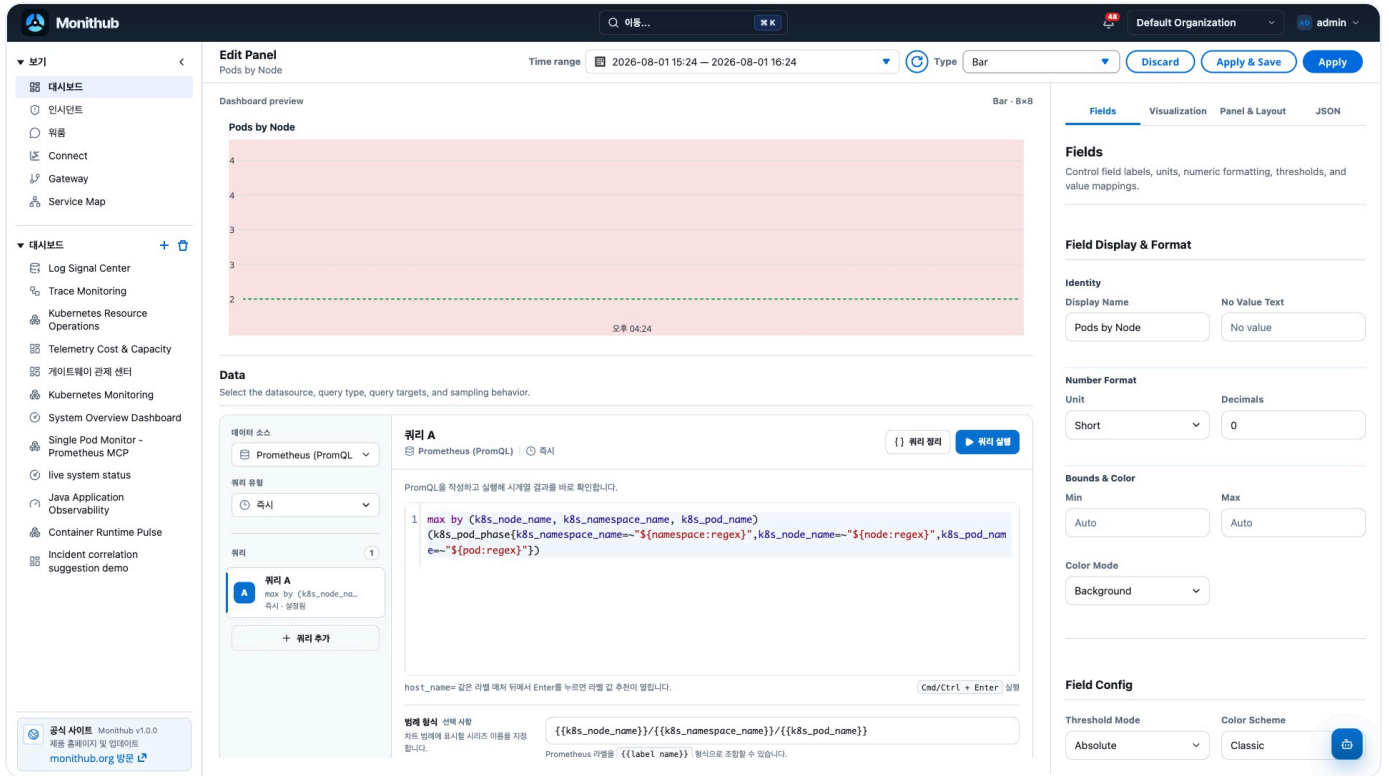
Data



Data에서는 막대 하나가 무엇을 의미하는지 먼저 정합니다. 시간 bucket 비교라면 range query, category 비교라면 query에서 category별 집계를 만들어야 합니다.

설정	역할
Query type	시간 bucket 막대는 Range , category 집계는 table query나 PromQL 집계를 사용합니다.
Queries	여러 series를 비교할 때 query를 추가합니다.
Query editor	bucket 크기와 time range가 너무 촘촘하지 않은지 확인합니다.
Legend Format	막대 색상이 어떤 series인지 바로 읽히도록 label을 줄입니다.

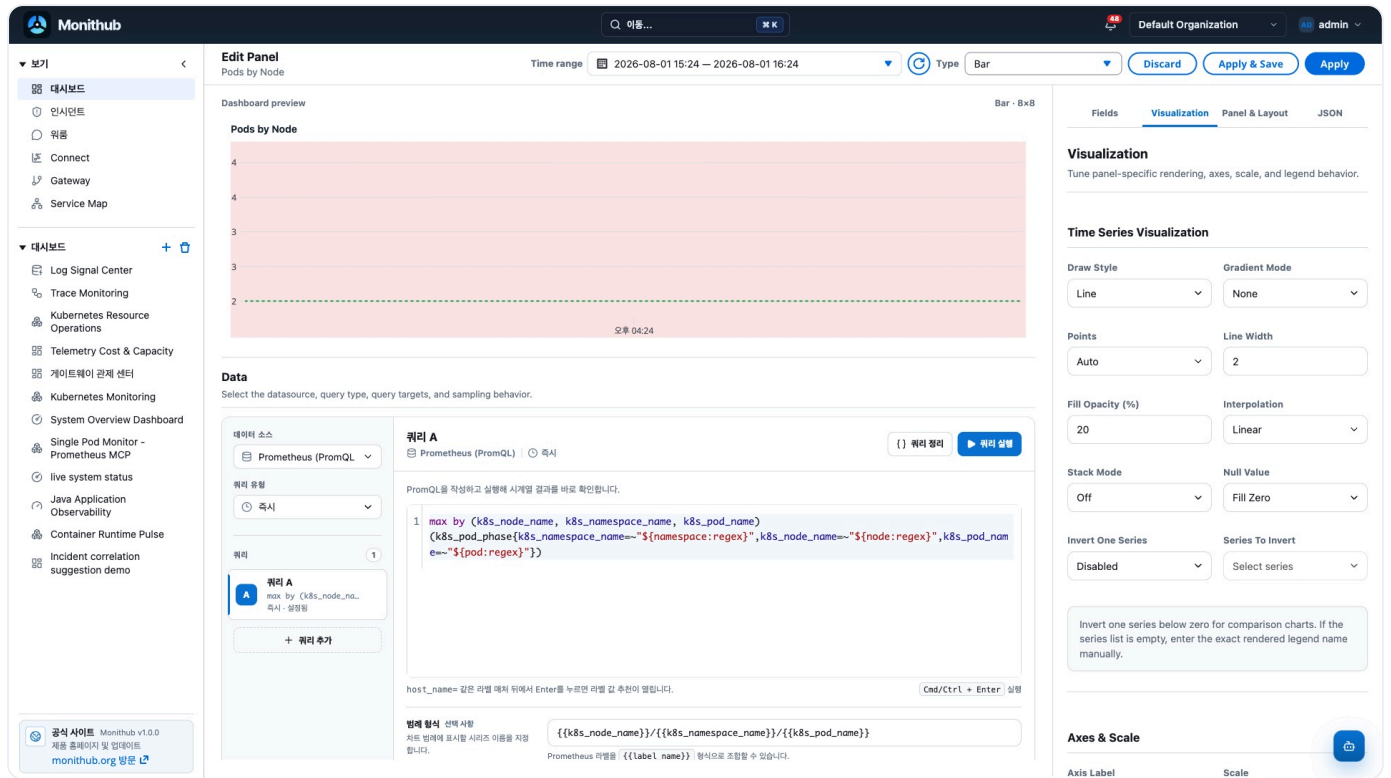
Fields



Fields에서는 막대 높이와 tooltip 값의 단위를 맞추니다. Bar는 축 단위가 시각적 판단을 크게 좌우하므로 Field Unit과 Axis Unit을 같은 기준으로 두는 편이 좋습니다.

설정	역할
Unit / Decimals	막대 높이와 tooltip 값을 읽는 기준입니다.
Min / Max	막대 축과 threshold 색상 기준에 영향을 줍니다.
Threshold Steps	값 구간별 상태 색상을 정합니다.
Series Palette	여러 막대 series 색상 순서를 통일합니다.

Visualization [🔗](#)



Visualization은 Time Series와 같은 설정을 공유하지만, Bar에서는 **Draw Style**과 **Stack Mode**가 핵심입니다.

설정	역할
Draw Style	Bar에서는 Bars 로 둡니다.
Stack Mode	series를 쌓아서 총량 또는 비중을 비교합니다.
Fill Opacity	막대 채움 정도입니다.
Gradient Mode	운영 화면에서는 단색이 가장 안정적입니다.
Null Value	누락 구간을 0으로 채우면 실제 0과 수집 공백이 섞일 수 있습니다.

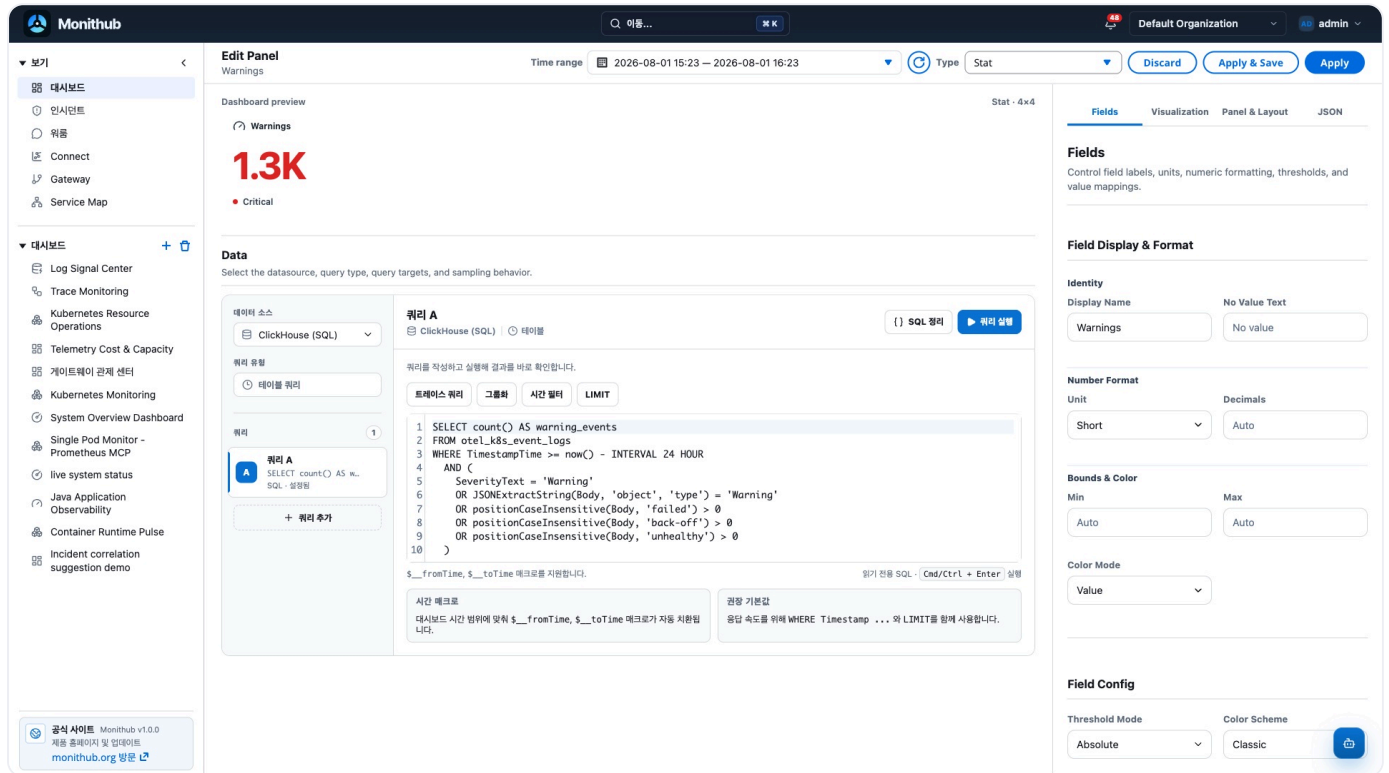
확인 순서 [🔗](#)

Data에서 비교 단위를 정하고, Fields와 Axes에서 단위를 맞춘 뒤, Visualization에서 stack 여부를 결정합니다.

Stat 패널

Stat은 하나의 대표값 또는 여러 series의 대표값을 간결하게 보여줍니다. 예시는 [Node CPU](#) 계열 Stat 편집 화면입니다.

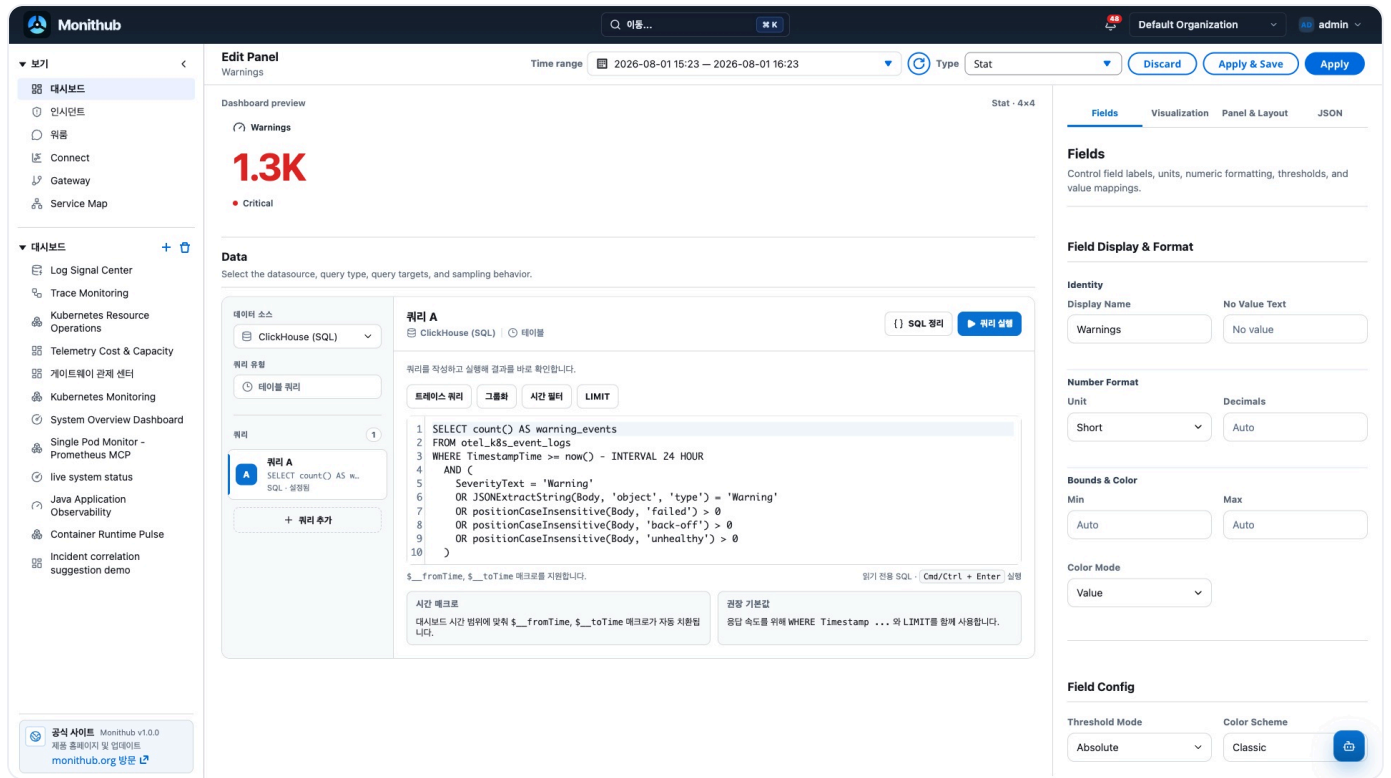
Data



Data에서는 Stat에 들어갈 값의 개수와 query type을 결정합니다. 현재값만 필요하면 [Instant](#), 작은 sparkline이나 변화 badge가 필요하면 [Range](#)를 사용합니다.

설정	역할
Query type	현재값은 Instant , 추세와 변화 비교는 Range 를 사용합니다.
Queries	여러 series가 반환되면 Visualization의 multi-series mode가 표시 방식을 정합니다.
Run Query	series 개수와 값 타입을 확인합니다.
Legend Format	multi-series grid에서 카드 이름으로 쓰입니다.

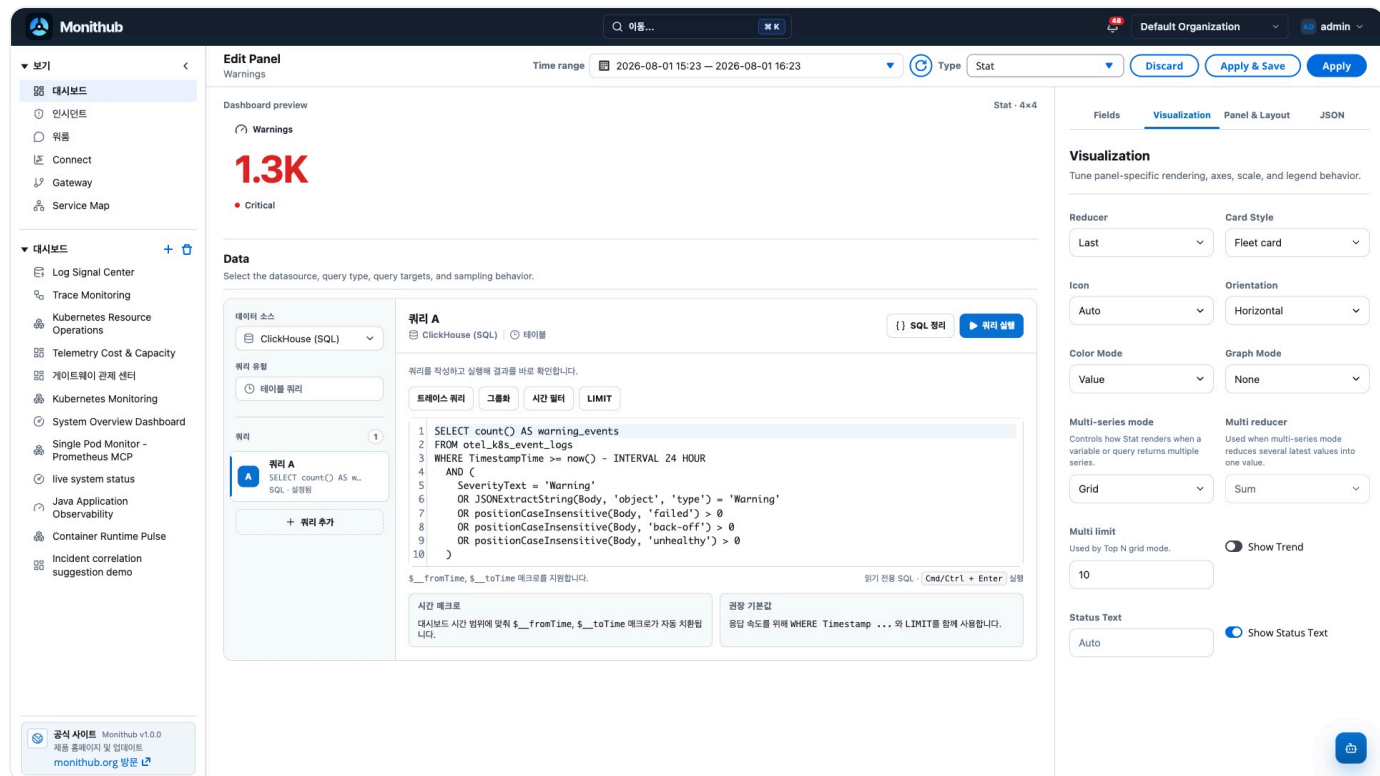
Fields



Fields는 큰 숫자 하나를 어떻게 읽을지 정하는 곳입니다. unit, decimals, threshold가 맞지 않으면 Stat은 가장 빠르게 오해를 만듭니다.

설정	역할
Display Name	단일 Stat 카드의 제목 또는 값 이름입니다.
Unit / Decimals	큰 숫자의 단위와 소수점 표시입니다.
Min / Max	background color와 threshold 판단 기준입니다.
Threshold Steps	값 색상과 상태 색을 나눕니다.
Value Mappings	0/1, 상태 코드, sentinel 상태값을 텍스트로 바꿉니다.

Visualization [🔗](#)



Visualization은 series를 어떤 대표값으로 줄이고, 카드 안에서 어떻게 표현할지 정합니다.

설정	역할
Reducer	Last, Mean, Sum, Min, Max 중 대표값을 선택합니다.
Card Style / Icon	카드 밀도와 아이콘 표현을 정합니다.
Orientation	값과 보조 그래프 배치를 정합니다.
Color Mode / Graph Mode	값 색상과 sparkline 표시 여부를 정합니다.
Multi-series mode	여러 series를 grid, reduce, top N, single only로 처리합니다.
Show Trend / Status Text	이전 값 대비 변화와 상태 문구를 표시합니다.

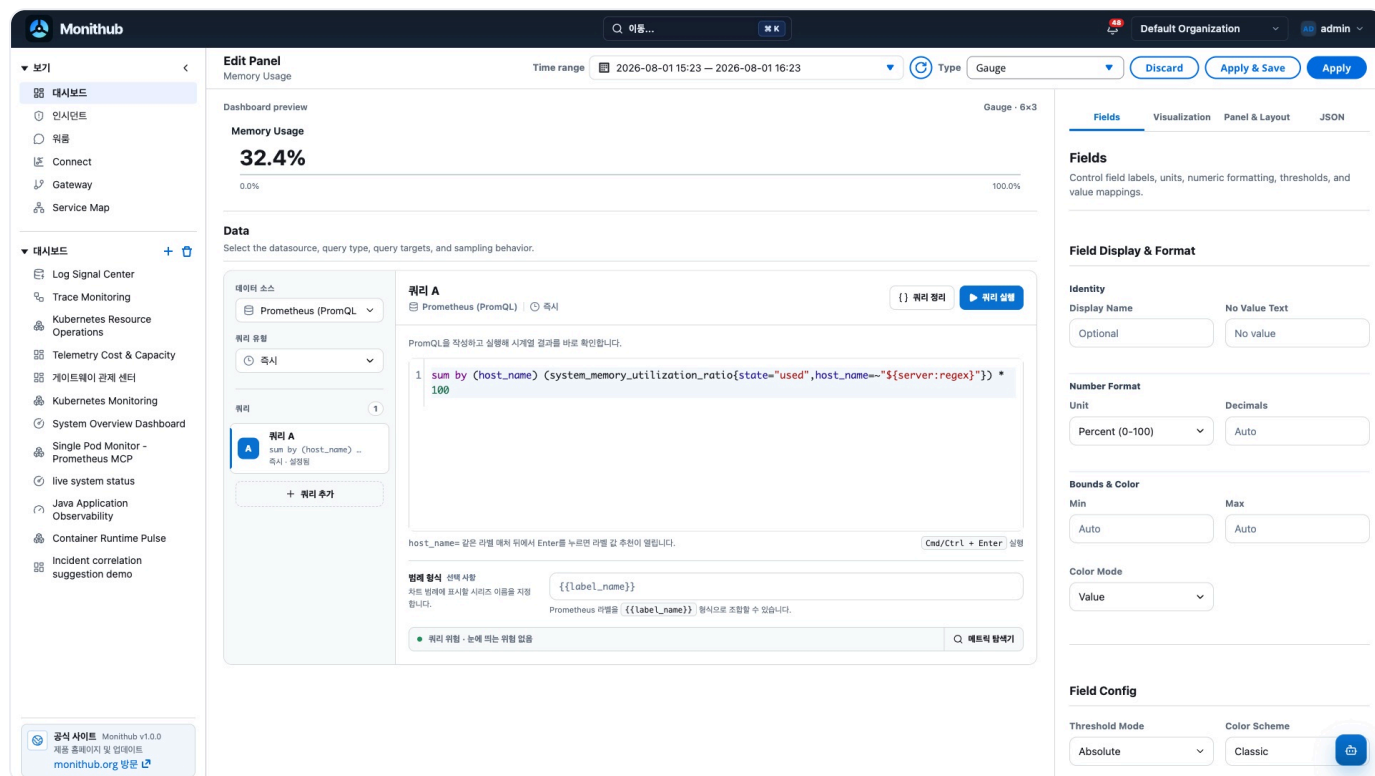
확인 순서 [🔗](#)

먼저 query 결과가 하나인지 여러 series인지 확인하고, Fields에서 단위를 맞춘 뒤, Reducer와 multi-series mode를 정합니다.

Gauge 패널

Gauge는 현재값이 기준 범위 안에서 어느 위치인지 보여줍니다. 단일 값의 상태를 빠르게 판단해야 할 때 사용합니다.

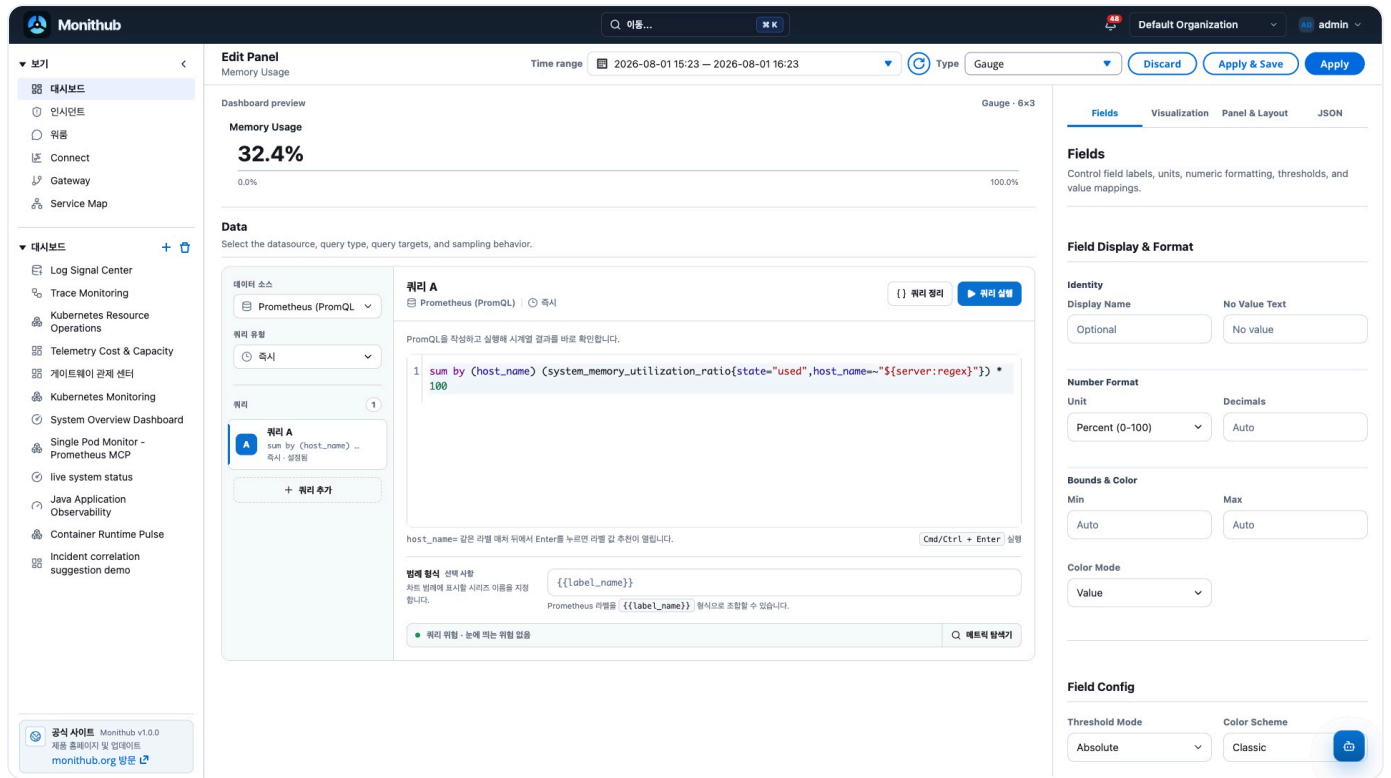
Data [🔗](#)



Data에서는 gauge에 들어갈 값을 하나의 대표값으로 만들 수 있는지 확인합니다. 여러 series가 남으면 reducer가 어느 값에 적용되는지 혼란스러울 수 있습니다.

설정	역할
Query type	현재 상태만 보면 Instant , 최근 구간 대표값이면 Range 를 사용합니다.
Run Query	결과가 하나의 series로 줄어드는지 확인합니다.
Legend Format	여러 series가 남을 수 있다면 대표값 이름을 명확히 둡니다.

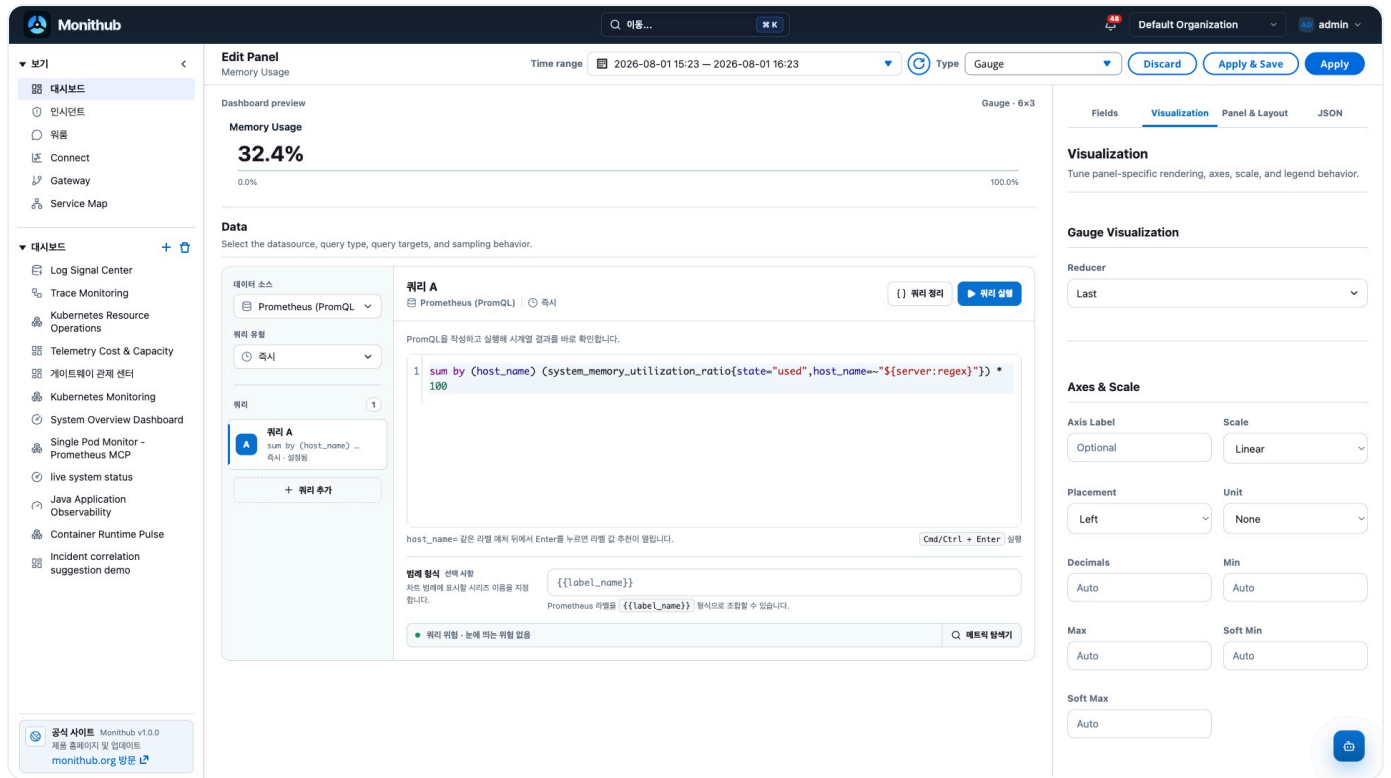
Fields [🔗](#)



Fields에서는 gauge의 범위와 색상 기준을 맞춥니다. Gauge는 **Min**, **Max**, threshold가 화면 의미를 거의 결정합니다.

설정	역할
Unit / Decimals	중앙 값과 gauge tick의 기본 단위입니다.
Min / Max	gauge 시작/끝 범위입니다.
Threshold Steps	warning/critical 색상 구간입니다.
Value Mappings	특수 상태값을 텍스트로 바꿀 때 사용합니다.

Visualization [🔗](#)



Visualization에서는 어떤 대표값을 gauge에 올릴지 정합니다. 범위와 색상은 Fields, 축 단위는 Axes & Scale과 함께 확인합니다.

설정	역할
Reducer	Last, Mean, Sum, Min, Max 중 gauge에 표시할 대표값을 선택합니다.
Axes & Scale	axis unit, min/max, scale을 조정합니다.
Fields Min / Max	gauge 범위와 색상 판단 기준으로 함께 작동합니다.

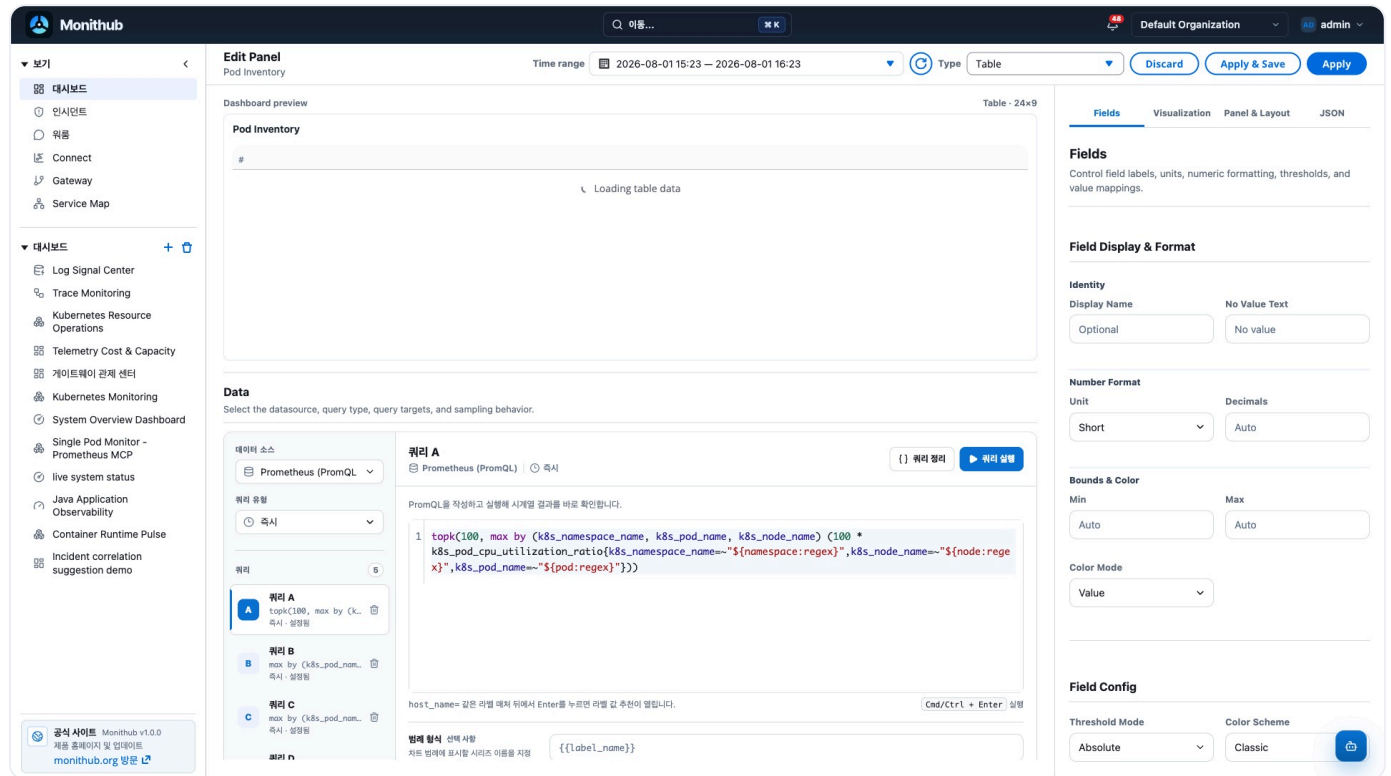
확인 순서 [🔗](#)

Data에서 단일 값으로 줄어드는지 확인하고, Fields에서 min/max와 threshold를 맞춘 뒤, Visualization의 reducer를 확정합니다.

Table 패널

Table은 query 결과를 row/column으로 검토하는 패널입니다. ClickHouse SQL 결과, Kubernetes event, 로그 요약, trace inventory처럼 근거 데이터를 보여줄 때 사용합니다.

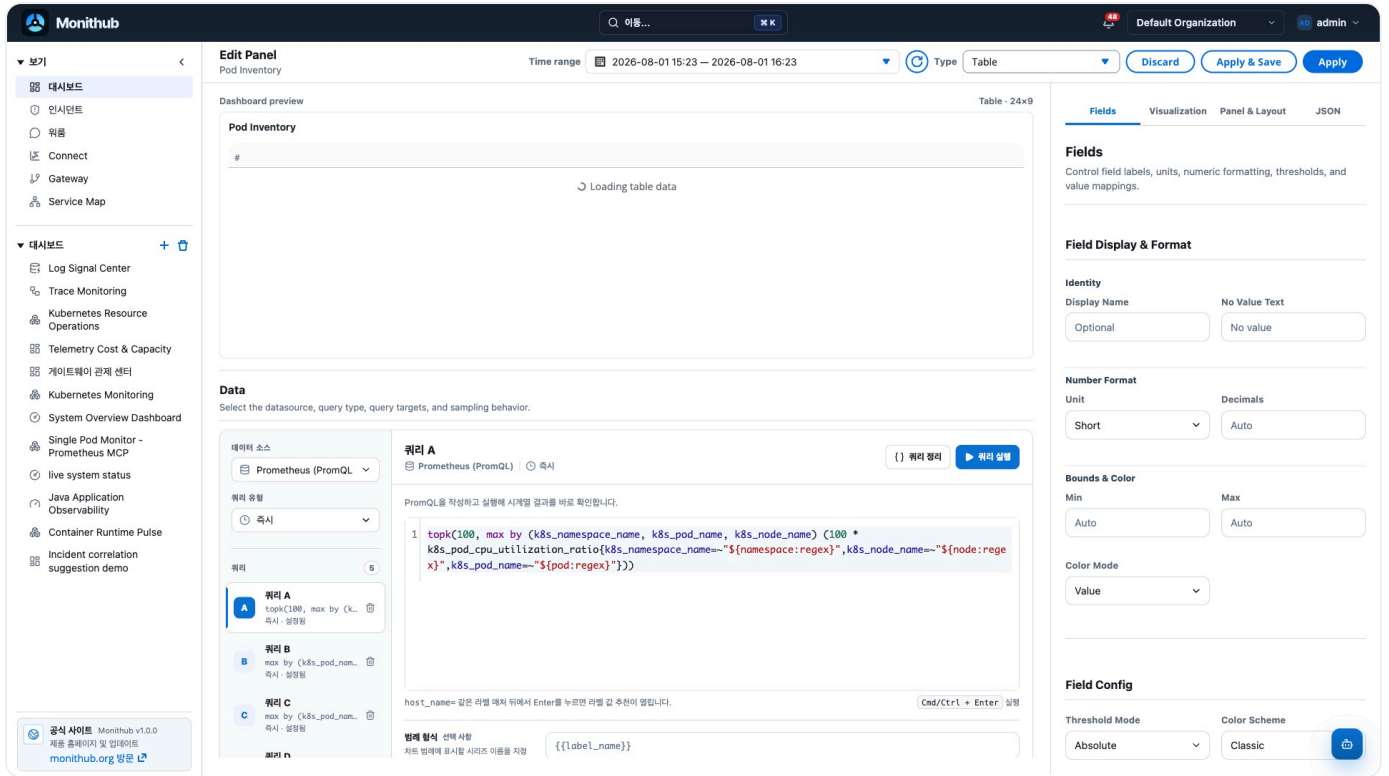
Data



Data에서는 row grain과 column 이름을 먼저 안정화합니다. column alias가 바뀌면 Visualization의 Detected Columns 설정도 다시 확인해야 합니다.

설정	역할
Datasource	로그, trace, inventory는 ClickHouse SQL을 자주 사용합니다.
Queries	A, B처럼 여러 query를 추가하고 join할 수 있습니다.
Query editor	column alias를 명확히 붙입니다.
Run Query	row 수, column 이름, null 비율을 확인합니다.
Join	query가 둘 이상이면 join 대상, key, type을 정합니다.

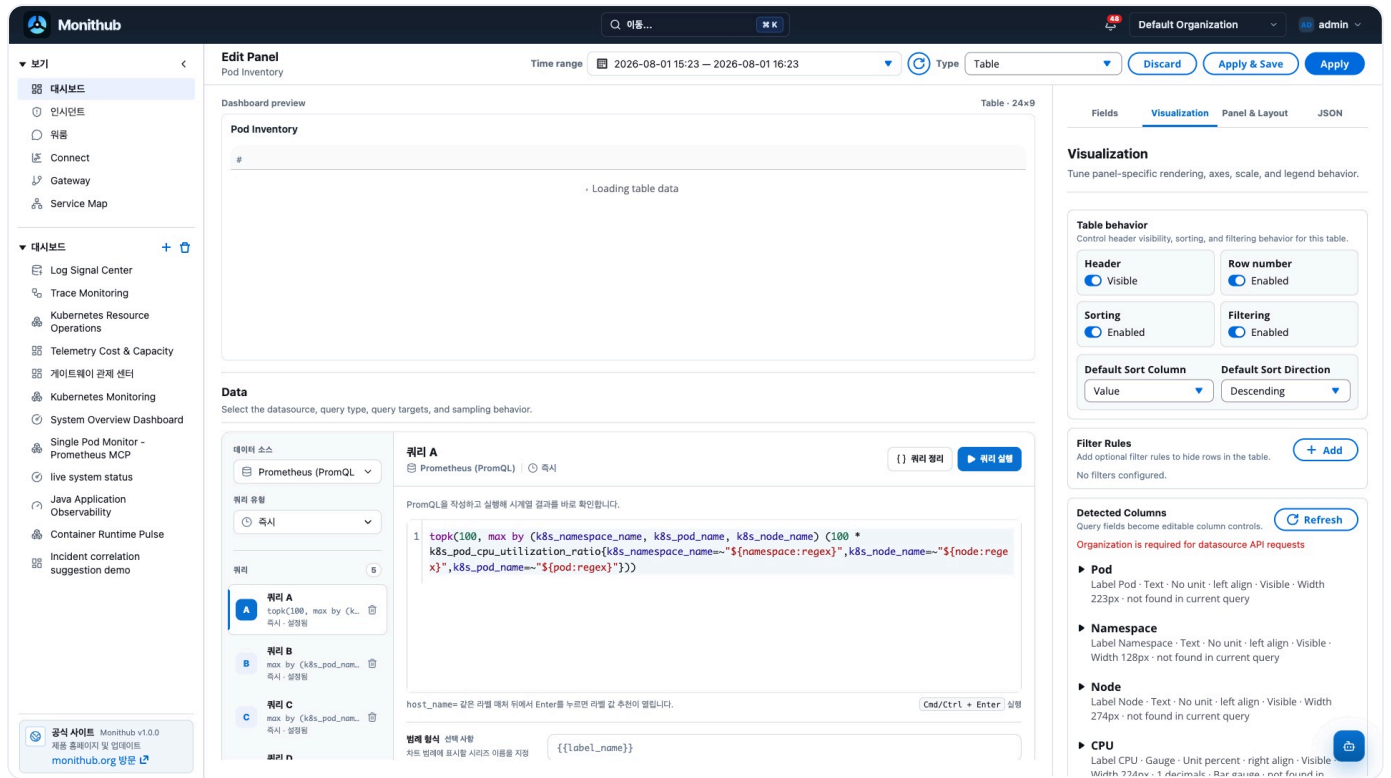
Fields



Fields는 table 전체의 기본 표시 규칙입니다. 특정 column만 다르게 보여야 하면 Visualization의 Detected Columns에서 더 구체적으로 덮어씁니다.

설정	역할
Display Name / No Value Text	기본 field 이름과 빈 값 문구입니다.
Unit / Decimals	column별 unit이 없을 때 쓰는 기본 포맷입니다.
Threshold Steps	column별 threshold가 없을 때 공통 색상 기준입니다.
Value Mappings	상태 코드나 boolean 값을 table cell 문구로 바꿉니다.

Visualization [🔗](#)



Visualization은 table behavior와 column별 표시 방식을 조정합니다. 이 탭이 Table 사용 가이드의 핵심입니다.

설정	역할
Header / Row number	table header와 행 번호 표시 여부입니다.
Sorting / Filtering	사용자가 정렬하거나 filter rule로 row를 숨길 수 있게 합니다.
Detected Columns	현재 query 결과의 field를 column 설정으로 엮니다.
Rename Field / Cell Type	column 이름과 text, gauge, sparkline, background color 표시를 정합니다.
Unit / Decimals / Min / Max	column별 숫자 포맷과 gauge 기준입니다.
Thresholds / Data link	column 전용 색상 구간과 drill-down 링크입니다.

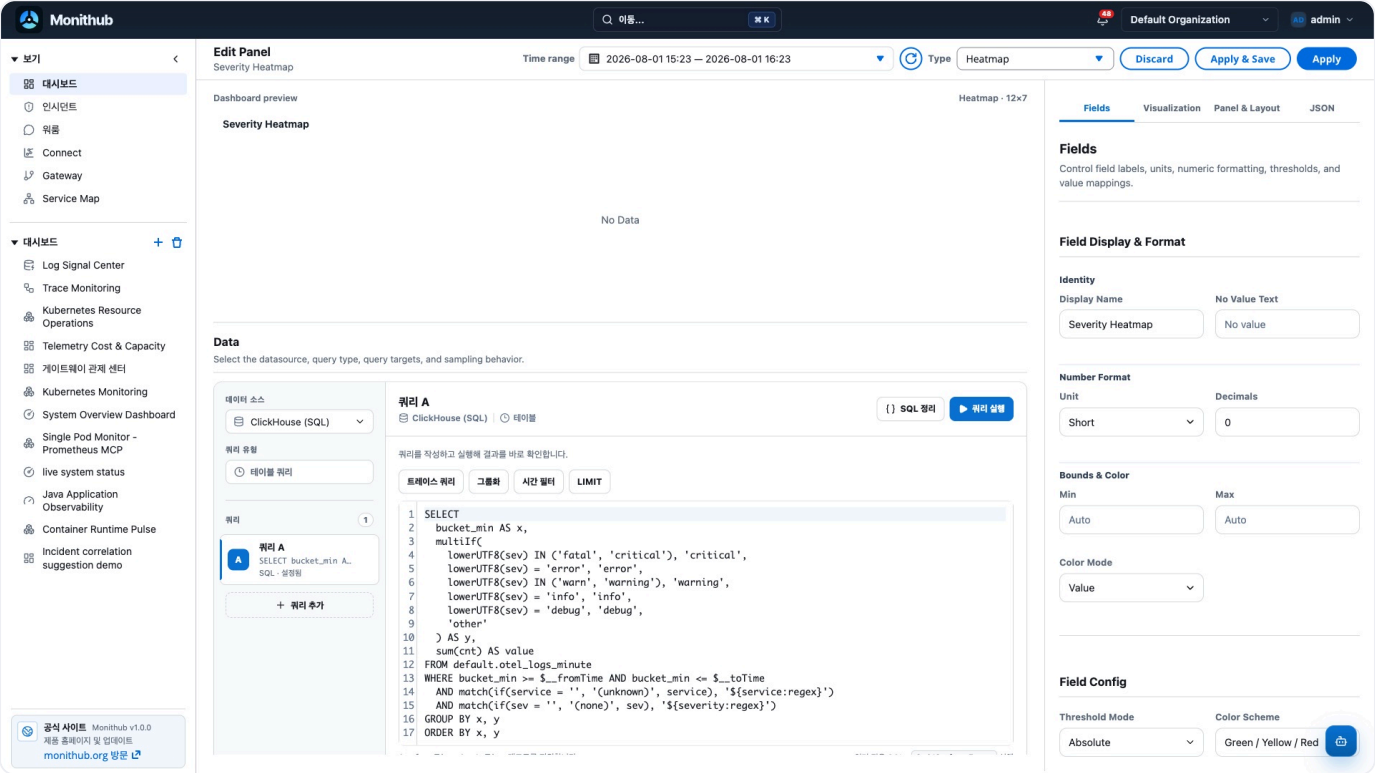
확인 순서 [🔗](#)

Data에서 column 이름을 고정하고, Fields에서 공통 포맷을 맞춘 뒤, Visualization에서 column별 예외를 정리합니다.

Heatmap 패널

Heatmap은 시간과 bucket, series label, value로 밀도 변화를 볼 때 사용합니다. latency histogram, error bucket, resource distribution처럼 값의 분포가 중요한 화면에 적합합니다.

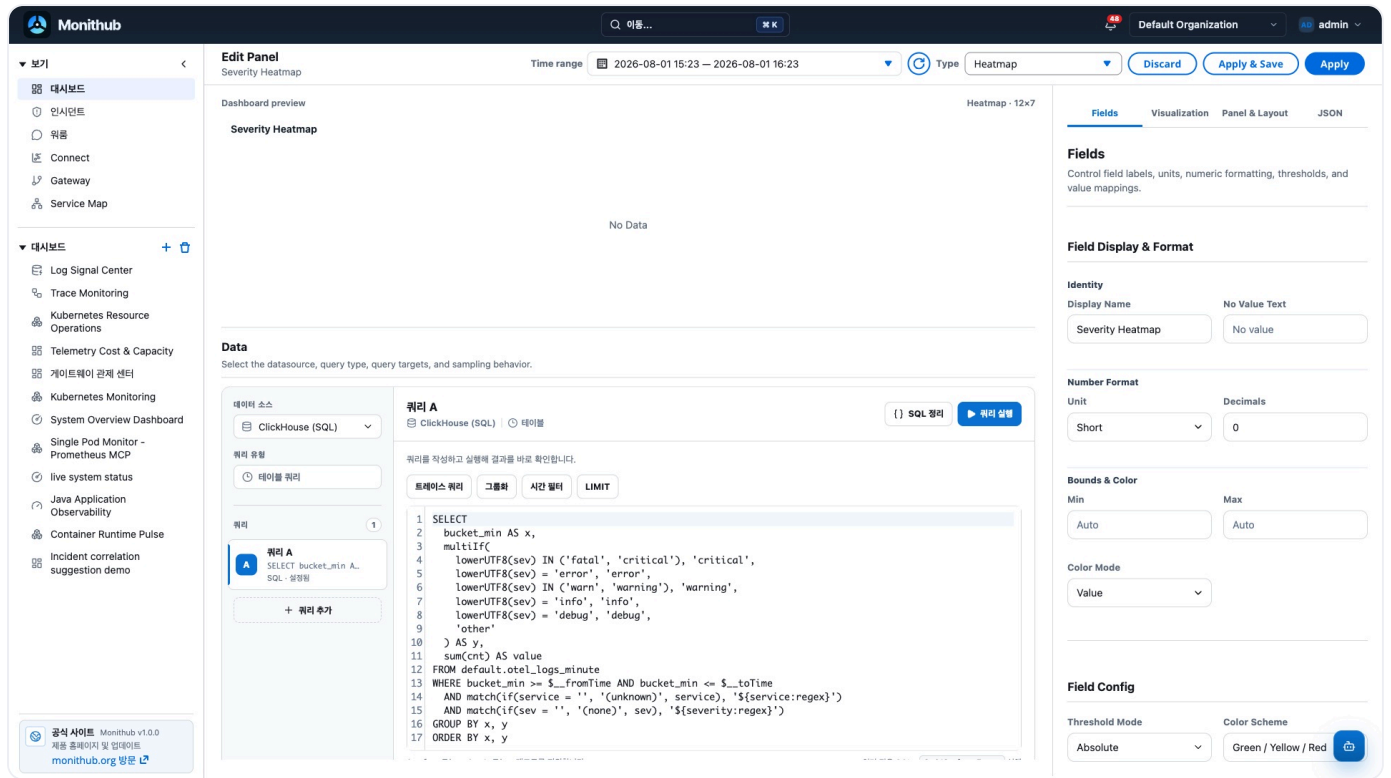
Data



Data에서는 x축 시간, y축 bucket, 색상 값이 한 row에 함께 나오도록 query를 구성합니다. ClickHouse SQL에서는 `$__fromTime`, `$__toTime` 매크로를 time range와 맞춥니다.

설정	역할
Datasource	trace latency는 <code>ClickHouse (SQL)</code> , Prometheus histogram은 <code>Prometheus (PromQL)</code> 을 사용합니다.
Query type	ClickHouse는 table query, Prometheus histogram은 보통 <code>Range</code> query입니다.
반환 field	<code>x</code> , <code>bucket</code> , <code>value</code> 또는 <code>time</code> , <code>le</code> , <code>value</code> 처럼 이름을 명확히 둡니다.
Run Query	row가 비어 있지 않은지, bucket label이 너무 많지 않은지 확인합니다.

Fields



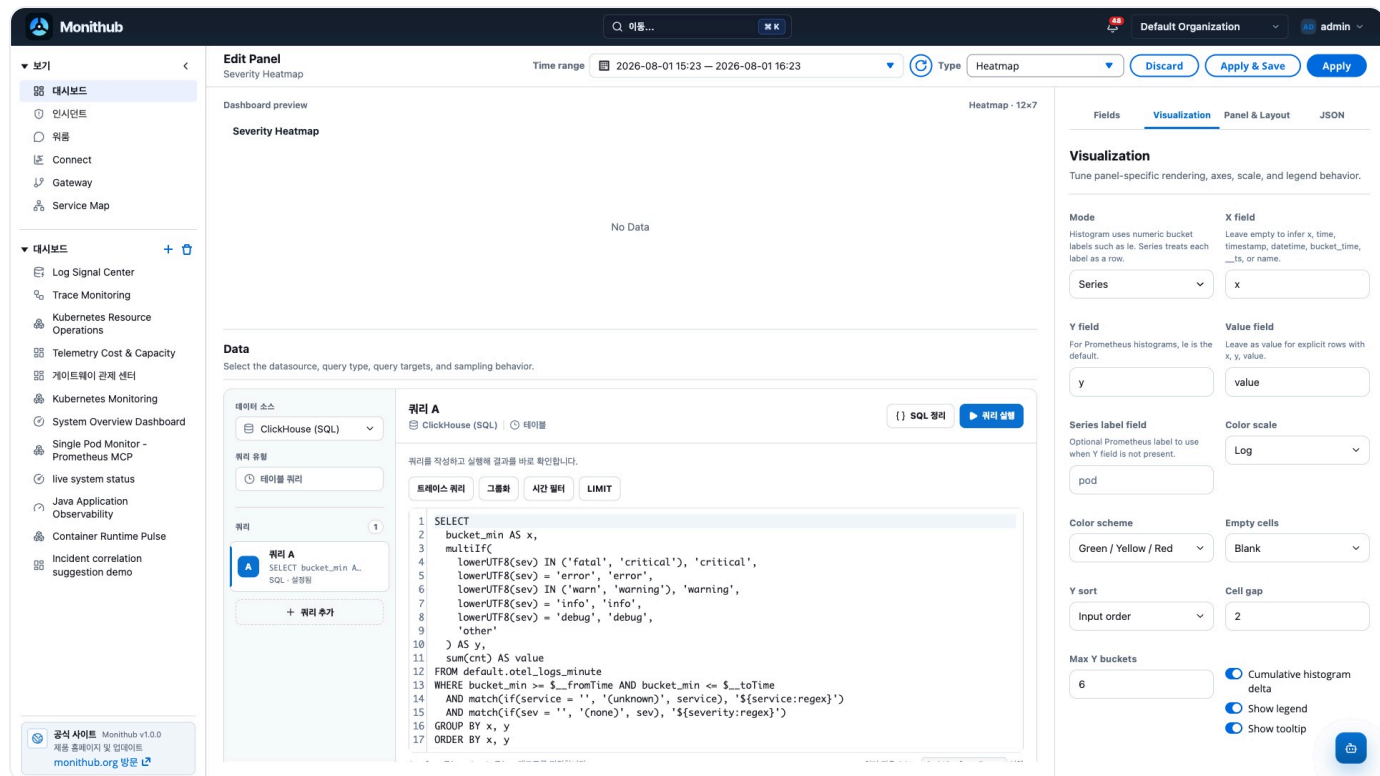
Fields는 색상 legend와 tooltip 값의 단위, 소수점, 상태 색 기준을 정합니다. Heatmap cell의 위치는 Visualization이 정하지만 값이 어떻게 읽히는지는 Fields가 좌우합니다.

설정

역할

Display Name / No Value Text	tooltip과 빈 cell 문구를 정합니다.
Unit / Decimals	count면 Short , latency면 ms 처럼 맞춥니다.
Min / Max	극단값 때문에 색이 눌릴 때 범위를 조정합니다.
Threshold Steps	특정 값 이상을 warning/critical처럼 읽을 때 사용합니다.
Value Mappings	0, null, 특수 상태값을 사람이 읽는 텍스트나 색상으로 바꿉니다.

Visualization [🔗](#)



Visualization에서는 query 결과의 어떤 field를 x/y/value로 쓸지와 heatmap 렌더링 방식을 정합니다.

설정	역할
Mode	histogram, series, value-binning 중 해석 방식을 고릅니다.
X / Y / Value field	시간, bucket, 색상 값 field를 지정합니다.
Series label field	Y field가 없을 때 행 이름으로 쓸 label입니다.
Color scale / scheme	linear/log와 색상 팔레트를 정합니다.
Empty cells	빈 칸을 blank로 둘지 0으로 채울지 정합니다.
Y sort / Max Y buckets	row 정렬과 bucket 수 상한입니다.
Cumulative histogram delta	누적 histogram을 interval delta처럼 해석합니다.

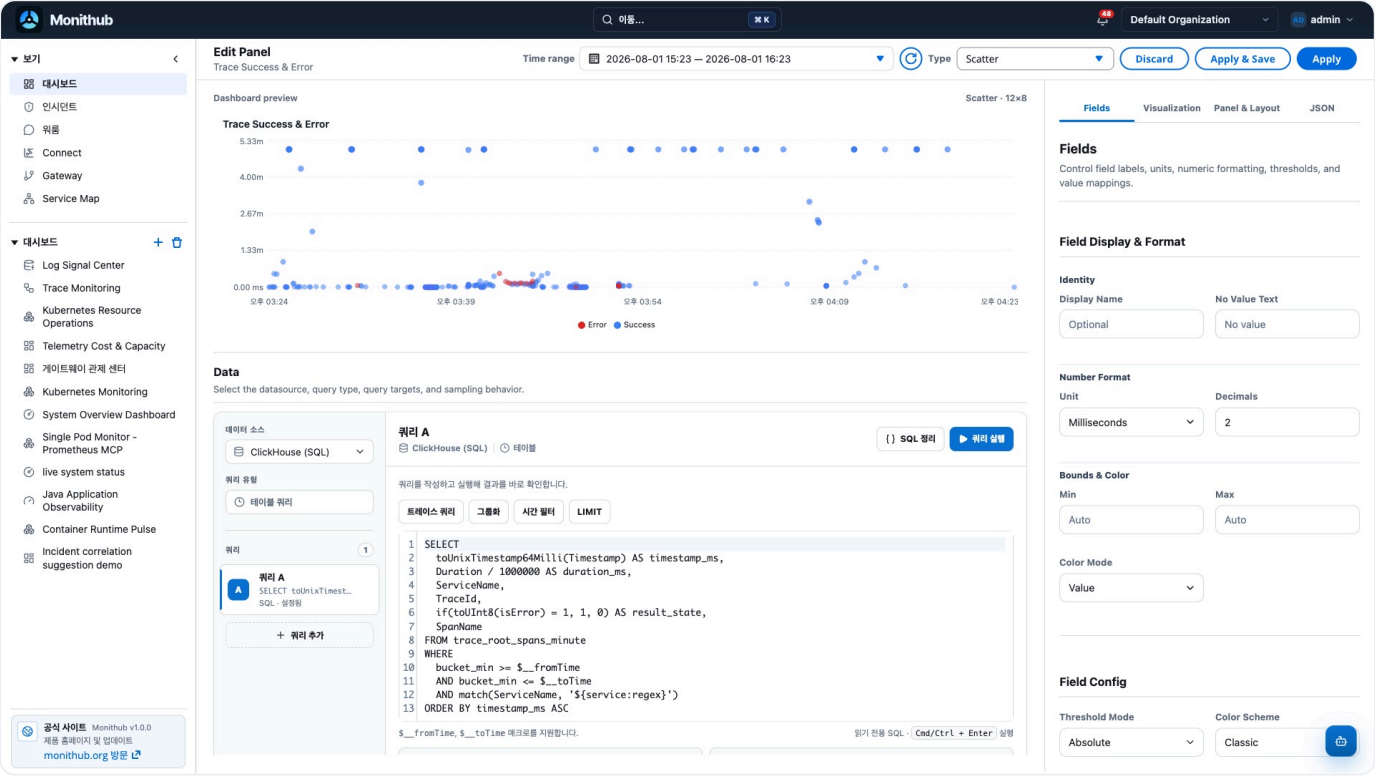
확인 순서 [🔗](#)

Data에서 x/y/value shape를 확인하고, Fields에서 값 단위를 맞춘 뒤, Visualization에서 field mapping과 empty cell 정책을 정합니다.

Scatter 패널

Scatter는 각 row를 하나의 점으로 표시합니다. trace duration, request size, latency outlier처럼 개별 관측치가 중요할 때 적합합니다.

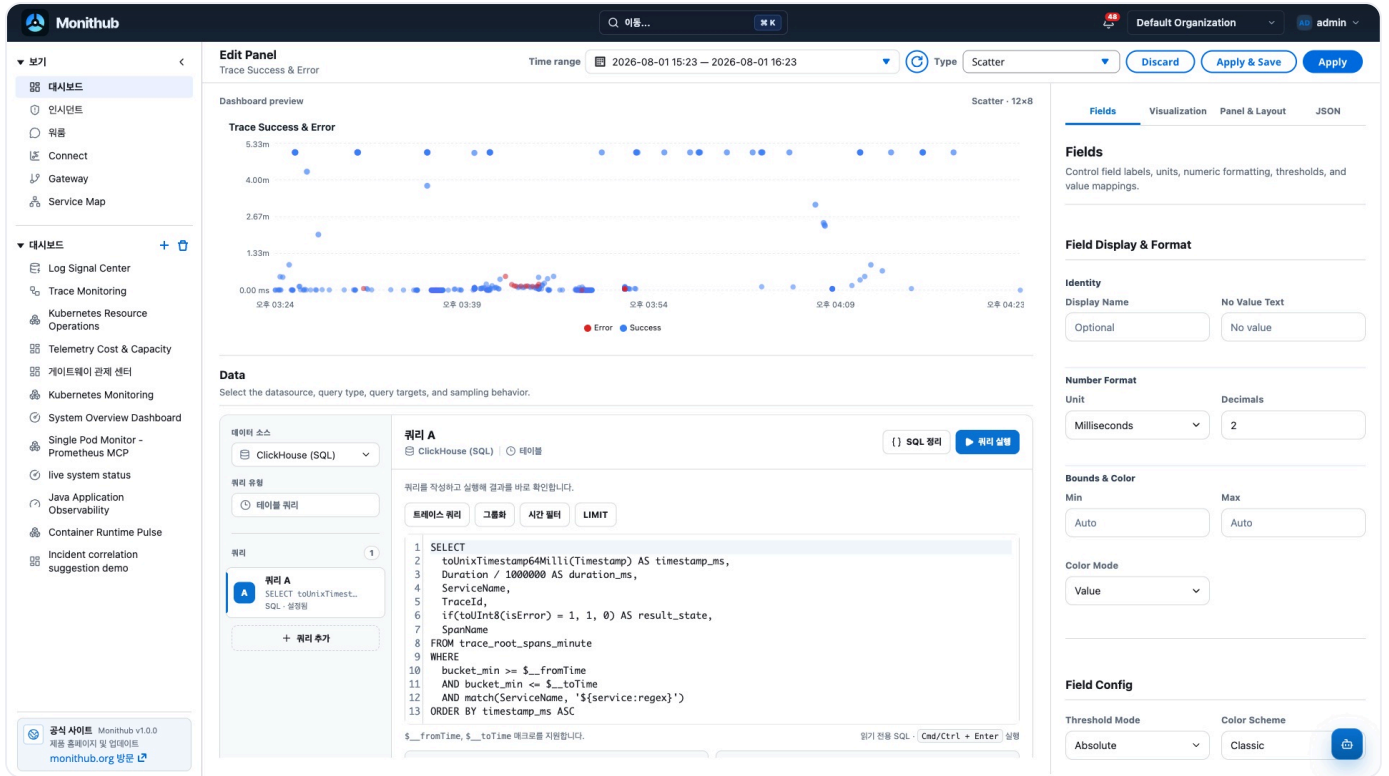
Data



Data에서는 한 row가 한 trace, 요청, 이벤트 같은 하나의 관측치가 되도록 query를 구성합니다.

설정	역할
Datasource	trace duration과 request size 같은 행 단위 데이터는 ClickHouse SQL과 잘 맞습니다.
Query type	ClickHouse는 table query로 동작합니다.
Query Sampling	점이 너무 많을 때 max points와 spike preserve를 조정합니다.
Run Query	x/y field가 있고 y field가 숫자인지 확인합니다.

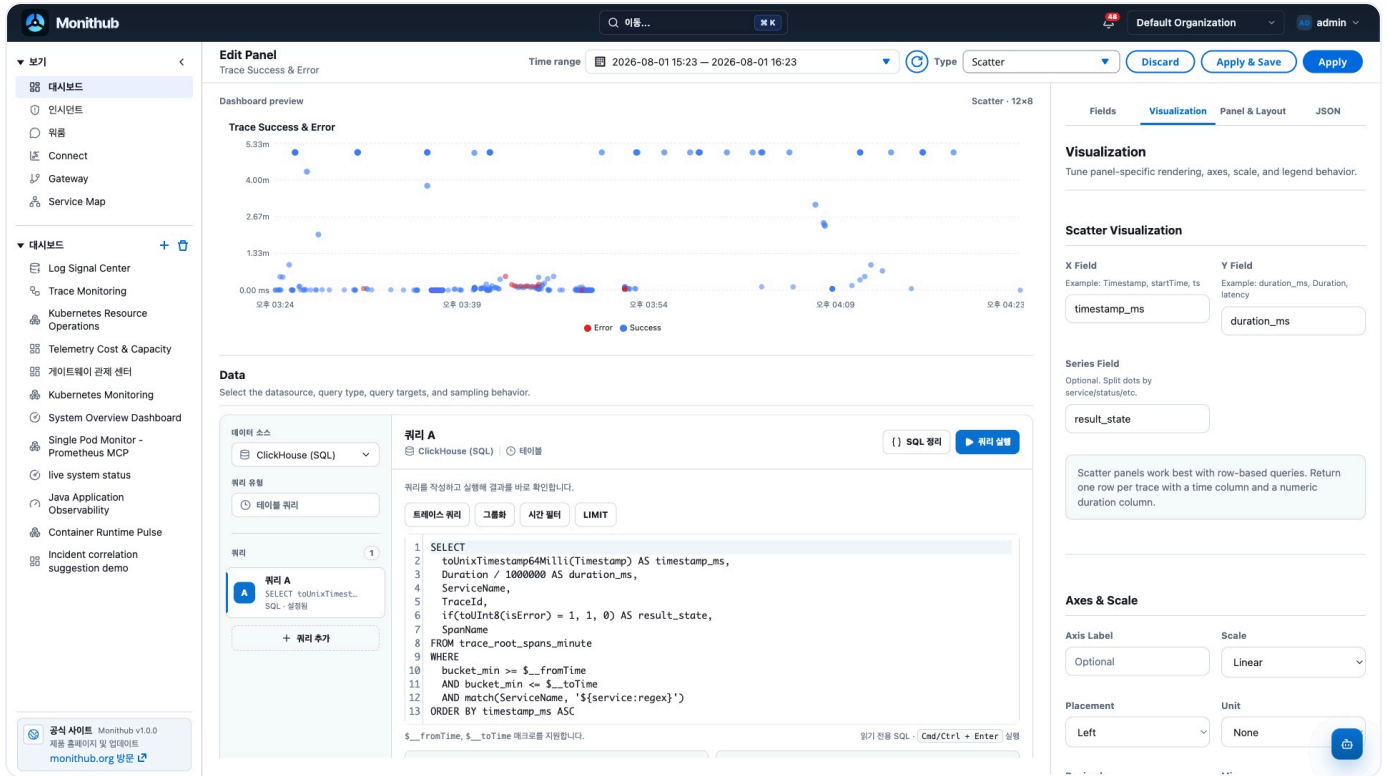
Fields



Fields에서는 y값 tooltip과 축 해석 기준을 맞춥니다. duration이면 ms, size면 bytes 처럼 단위를 먼저 정리합니다.

설정	역할
Unit / Decimals	y값 tooltip과 축 단위의 기본 기준입니다.
Min / Max	y축 범위와 outlier 표시를 조정합니다.
Threshold Steps	점 색상이나 상태 해석 기준으로 쓸 수 있습니다.
Value Mappings	상태값을 tooltip에서 사람이 읽는 문구로 바꿉니다.

Visualization



Visualization에서는 row의 어떤 column을 점의 좌표와 그룹으로 쓸지 정합니다.

설정	역할
X Field	x축 field입니다. 예: <code>Timestamp</code> , <code>startTime</code> , <code>ts</code>
Y Field	y축 numeric field입니다. 예: <code>duration_ms</code> , <code>Duration</code> , <code>latency</code>
Series Field	점 색상/그룹을 나눌 field입니다. 예: <code>ServiceName</code> , <code>status</code>
Axes & Legend	축 label, scale, legend 표시 여부를 조정합니다.

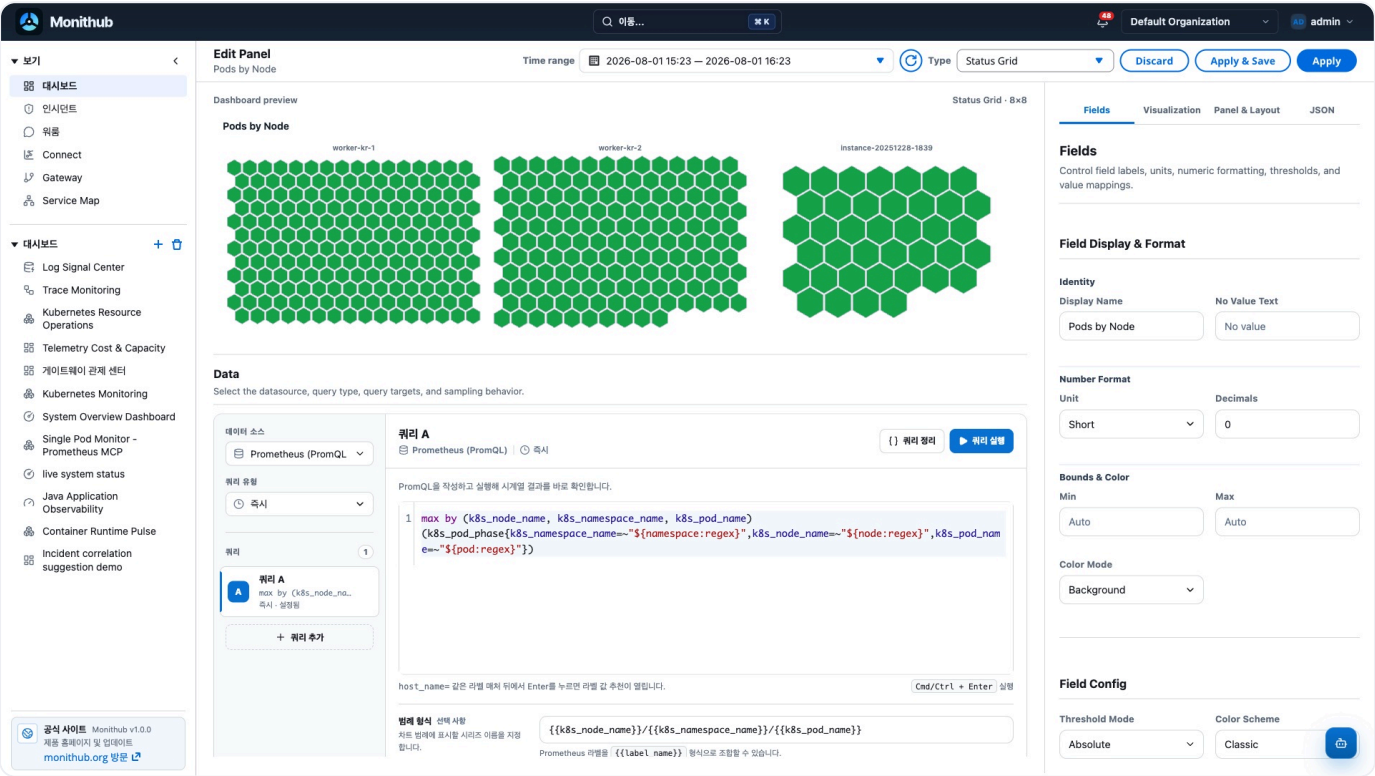
확인 순서 [🔗](#)

Data에서 row grain과 x/y field를 확인하고, Fields에서 y값 단위를 맞춘 뒤, Visualization에서 field mapping을 지정합니다.

Status Grid 패널

Status Grid는 여러 대상의 정상/비정상 상태를 빠르게 스캔하는 패널입니다. 예시는 [Pod CPU Status Grid](#) 편집 화면입니다.

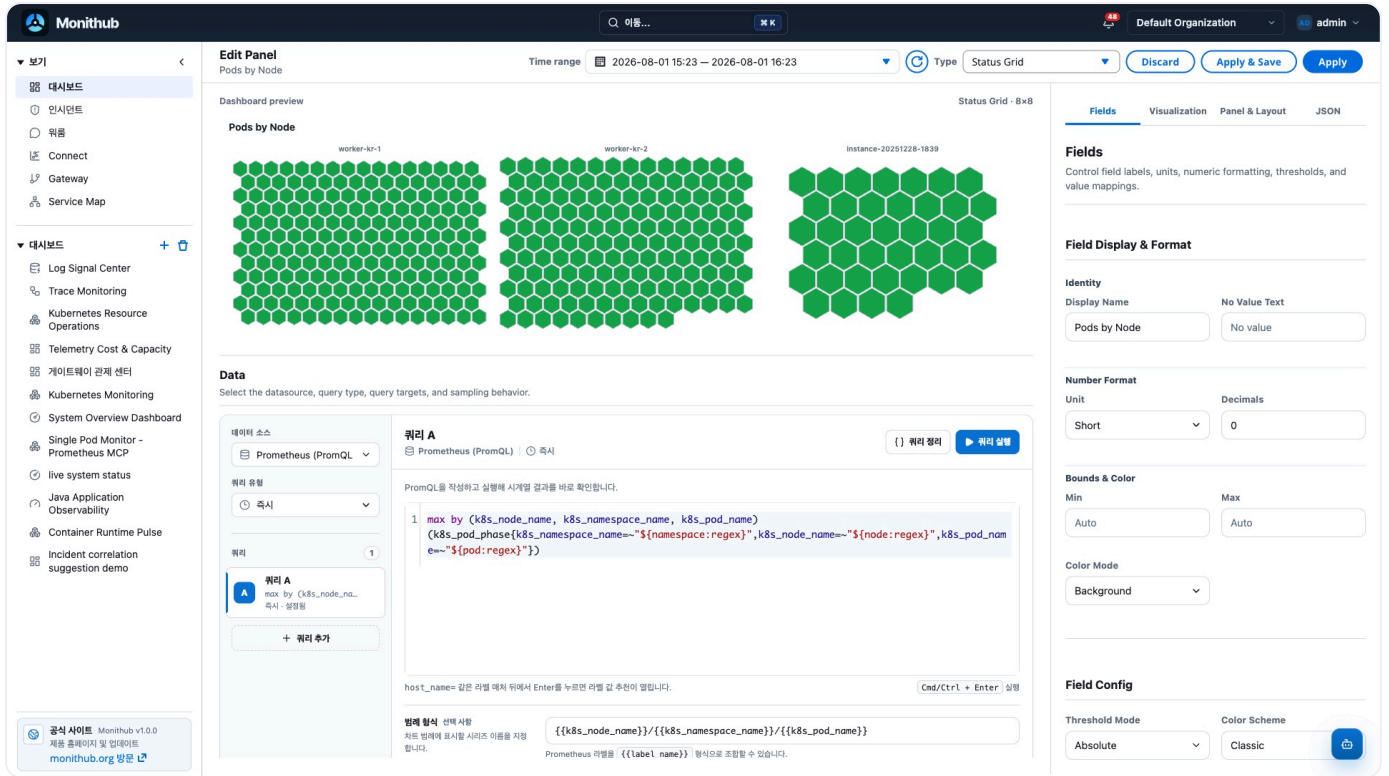
Data



Data에서는 대상 하나가 타일 하나로 나오도록 series label이나 row field를 정리합니다. grouping을 쓸 계획이라면 legend 이름에 [namespace/pod](#), [service/instance](#)처럼 일정한 구분자를 둡니다.

설정	역할
Datasource	상태 metric이 있는 Prometheus나 ClickHouse datasource를 선택합니다.
Query type	현재 상태만 필요하면 Instant , 최근 구간 대표값이 필요하면 Range 를 사용합니다.
Queries	여러 상태 query를 넣을 수 있지만 타일 이름이 겹치지 않게 정리합니다.
Legend Format	group separator로 나눌 이름 규칙을 정합니다.
Run Query	타일 수, label 중복, 비어 있는 series 여부를 확인합니다.

Fields



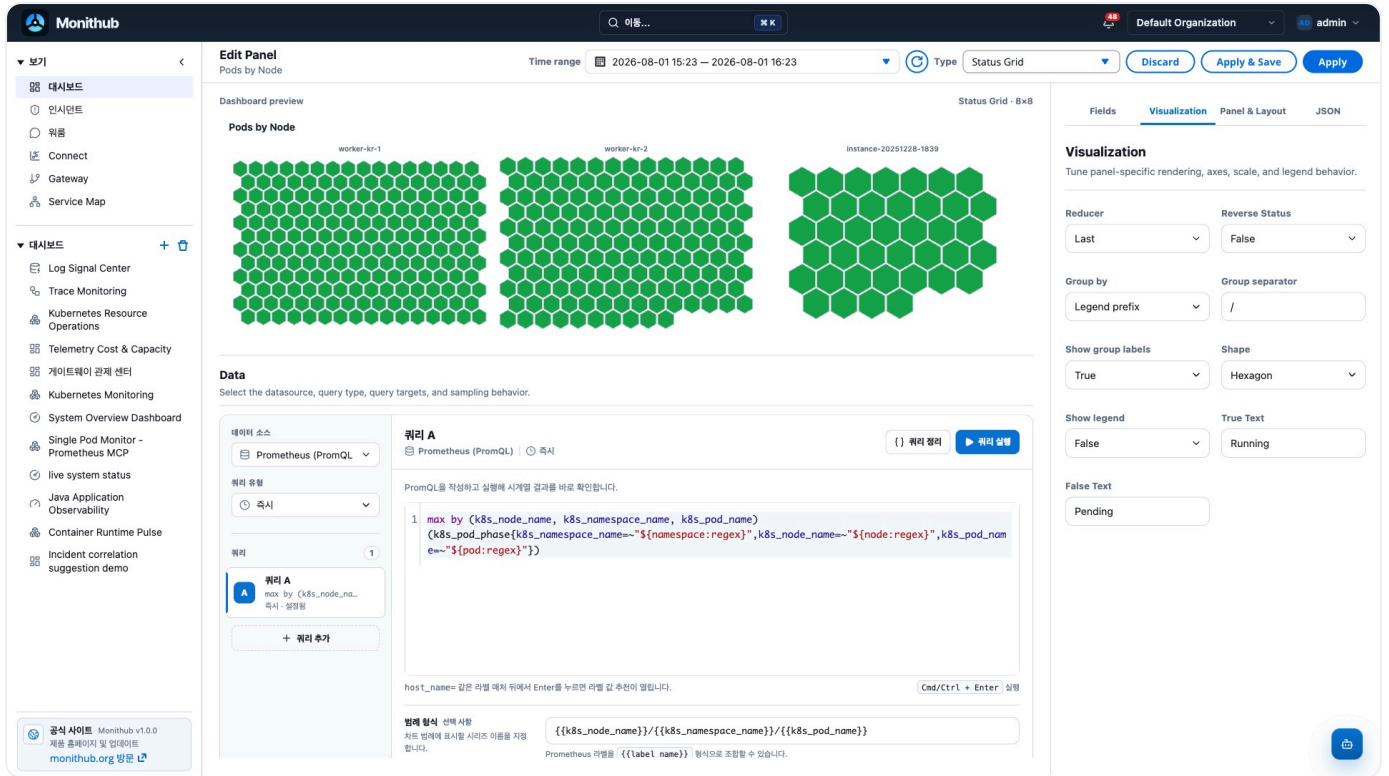
Fields에서는 상태값을 사람이 읽을 수 있는 문구와 색상 기준으로 바꿉니다. Status Grid는 unit보다 threshold와 value mapping 의미가 더 중요합니다.

설정

역할

Display Name / No Value Text	타일 이름과 데이터가 없을 때의 문구를 정합니다.
Unit / Decimals	숫자 상태값을 그대로 보여줄 때의 표시 기준입니다.
Threshold Steps	정상/비정상 판단 기준입니다. 예를 들어 0은 정상, 1 이상은 장애처럼 둡니다.
Value Mappings	0, 1, boolean 값을 Up, Down, Degraded 같은 문구로 바꿉니다.
Color Scheme	threshold 의미와 타일 색상이 맞는지 확인합니다.

Visualization [🔗](#)



Visualization에서는 여러 sample을 하나의 상태로 줄이는 방법, 타일 grouping, 타일 모양과 상태 문구를 정합니다.

설정	역할
Reducer	여러 sample 중 상태 판단에 사용할 대표값입니다.
Reverse Status	1이 정상인지 비정상인지 metric 의미가 반대일 때 뒤집습니다.
Group by	legend prefix를 기준으로 타일을 그룹화합니다.
Group separator	prefix를 나눌 구분자입니다. 기본은 <code>/</code> 입니다.
Show group labels	그룹 이름 표시 여부입니다.
Shape	hexagon 또는 rounded tile 모양을 선택합니다.
Show legend	상태 색상 범례 표시 여부입니다.
True Text / False Text	true/false 상태 문구입니다.

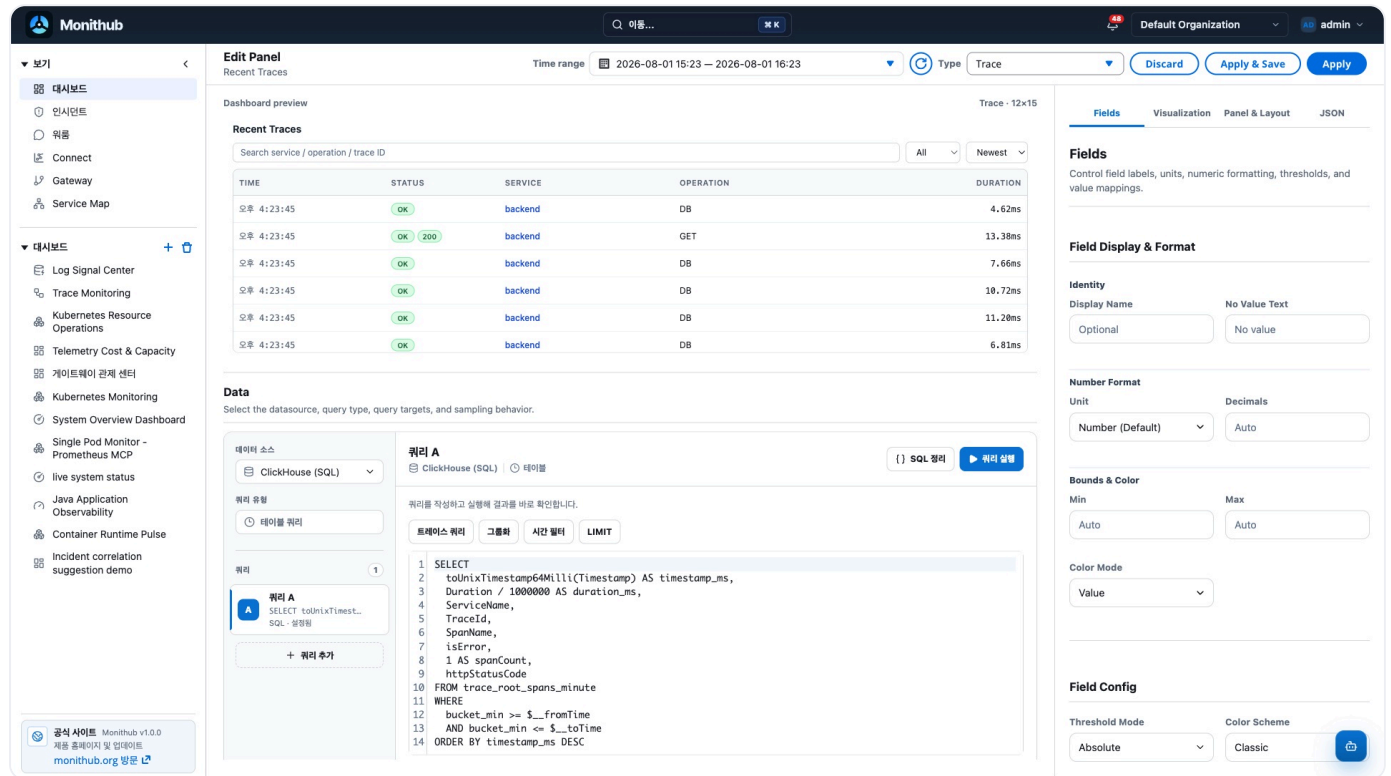
확인 순서 [🔗](#)

Data에서 타일 단위와 legend 구분자를 맞추고, Fields에서 값과 색상 의미를 정한 뒤, Visualization에서 reducer와 reverse status를 함께 확인합니다.

Trace 패널

Trace 패널은 trace 목록과 요약을 보여줍니다. 일반 PromQL 패널보다 ClickHouse trace 데이터와 더 밀접합니다.

Data



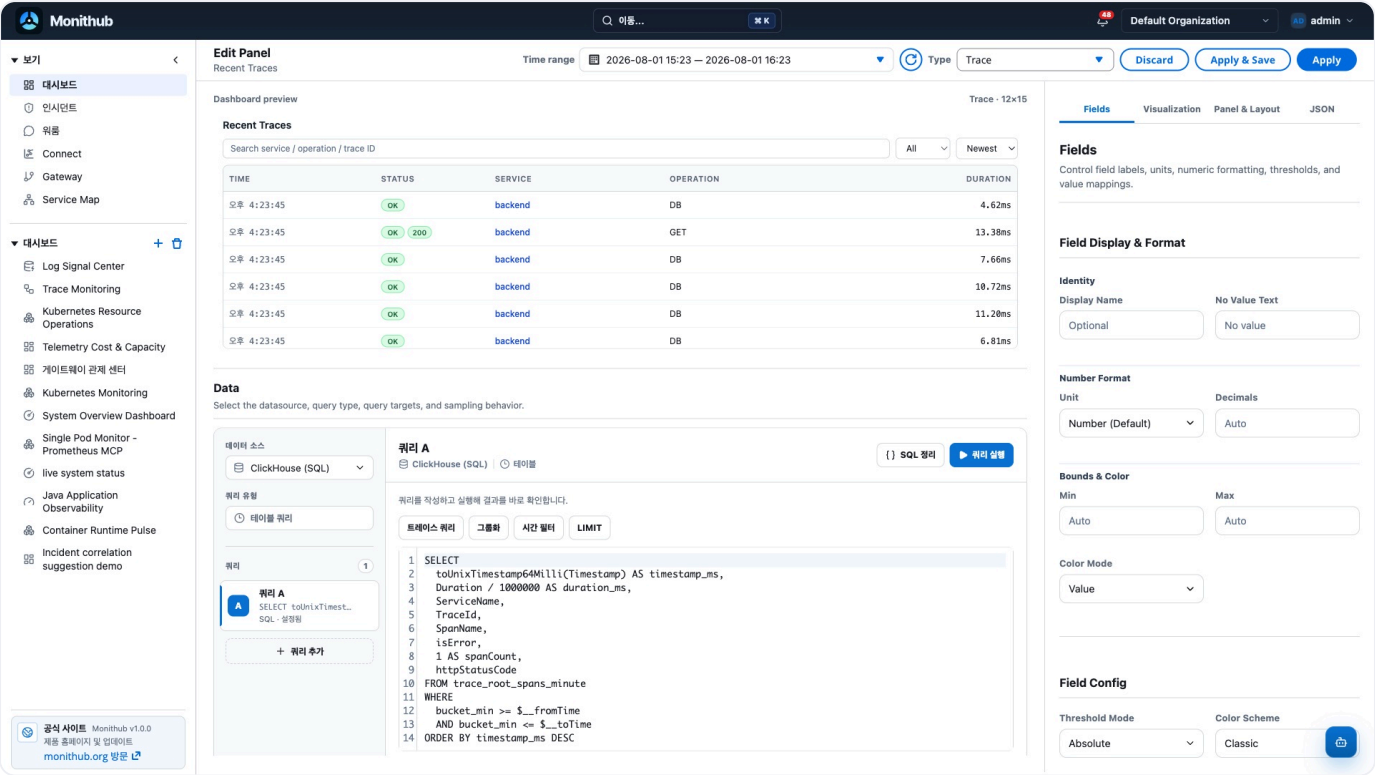
TIME	STATUS	SERVICE	OPERATION	DURATION
오후 4:23:45	OK	backend	DB	4.62ns
오후 4:23:45	OK	200 backend	GET	13.38ns
오후 4:23:45	OK	backend	DB	7.66ns
오후 4:23:45	OK	backend	DB	10.72ns
오후 4:23:45	OK	backend	DB	11.20ns
오후 4:23:45	OK	backend	DB	6.81ns

```
1 SELECT
2   toUnixTimestamp64Milli(timestamp) AS timestamp_ms,
3   Duration / 1000000 AS duration_ms,
4   ServiceName,
5   TraceId,
6   SpanName,
7   isError,
8   1 AS spanCount,
9   httpStatusCode
10  FROM trace_root_spans_minute
11  WHERE
12    bucket_min >= $__fromTime
13    AND bucket_min <= $__toTime
14  ORDER BY timestamp_ms DESC
```

Data에서는 trace 목록을 구성할 column을 안정적으로 반환해야 합니다. `trace_id`, `service`, `operation`, `duration`, `status`, `timestamp` 계열 column이 빠지면 Visualization에서 목록을 제대로 구성하기 어렵습니다.

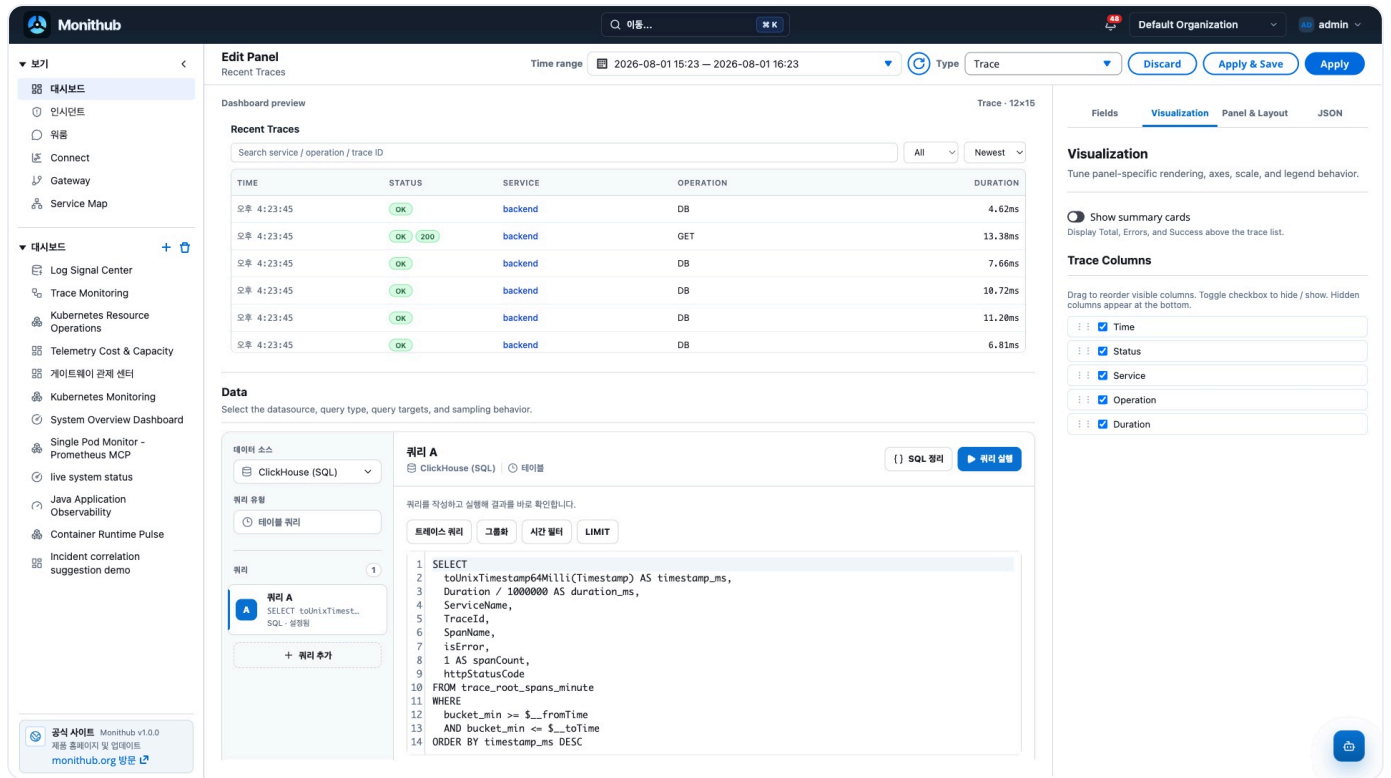
설정 역할

Datasource	보통 <code>ClickHouse (SQL)</code> 을 사용합니다.
Query editor	trace id, service, operation, duration, status, timestamp column을 반환합니다.
Time Filter	<code>\$__fromTime</code> , <code>\$__toTime</code> 매크로로 편집 화면 time range와 맞춥니다.
Trace Query / Group By	trace 목록 또는 service별 집계 template을 빠르게 넣을 때 사용합니다.
Run Query	column 이름과 duration 단위가 Trace 설정과 맞는지 확인합니다.



Fields에서는 trace duration, status code, 빈 값 표시 기준을 정합니다. 목록 column이 많아도 값의 단위와 상태 문구는 여기서 일관되게 맞춥니다.

설정	역할
Display Name / No Value Text	column 이름과 비어 있는 status, operation 문구를 정합니다.
Unit / Decimals	trace duration이 ns/us/ms 중 무엇인지 query와 맞춥니다.
Threshold Steps	duration이나 error count를 상태 색상으로 해석할 때 사용합니다.
Value Mappings	status code를 <code>Error</code> , <code>OK</code> 같은 문구로 바꿉니다.
Color Scheme	status나 duration 색상이 운영 의미와 맞는지 확인합니다.



Visualization에서는 summary card와 trace list column을 정합니다. 장애 triage 화면에서는 status, service, operation, duration처럼 판단 근거가 되는 column을 앞쪽에 둡니다.

설정	역할
Show summary cards	Total, Errors, Success 같은 summary card 표시 여부입니다.
Trace Columns	trace list column 표시 여부와 순서를 정합니다.
Drag to reorder	visible column을 drag해서 순서를 바꿉니다.
Checkbox	column을 숨기거나 다시 표시합니다. 최소 하나의 visible column은 유지합니다.

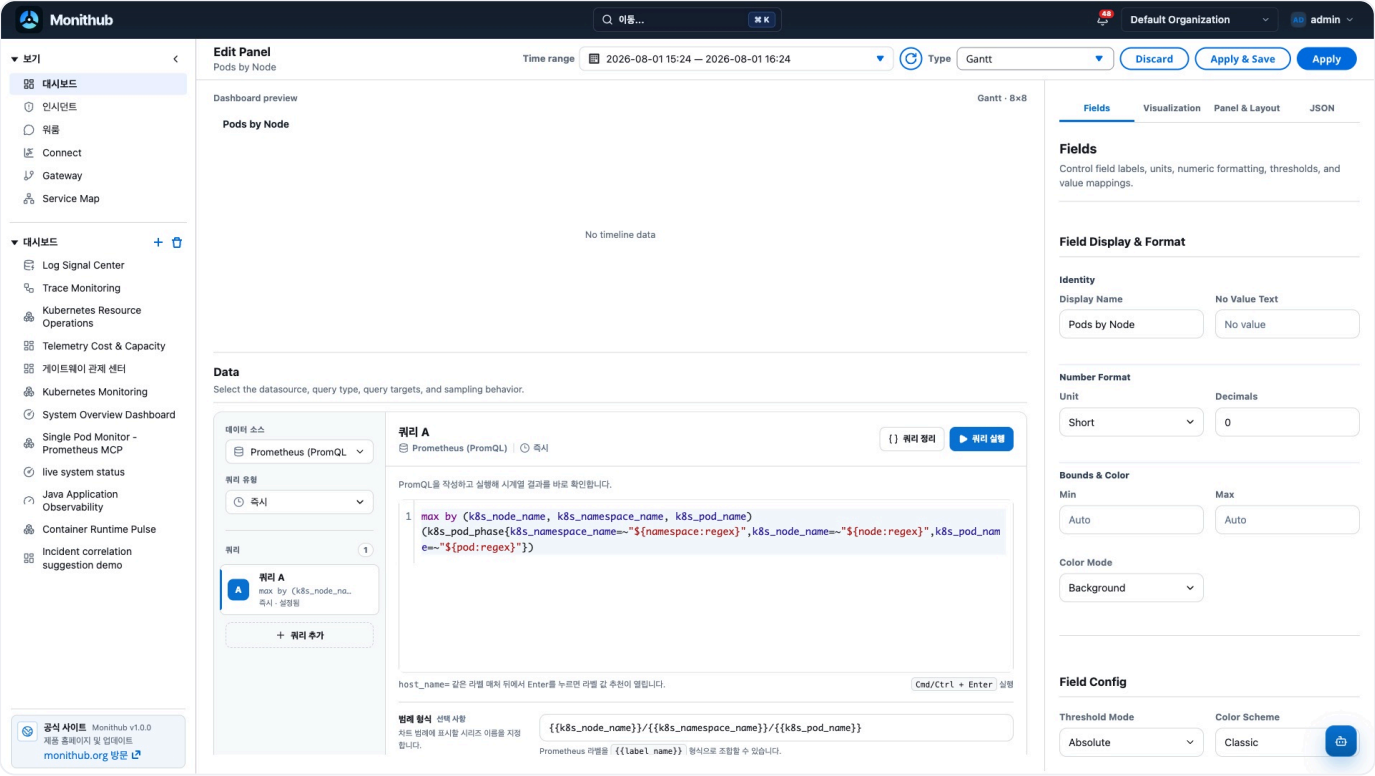
확인 순서 [🔗](#)

Data에서 trace 목록 column을 확인하고, Fields에서 duration 단위와 status 문구를 맞춘 뒤, Visualization에서 summary card와 column 순서를 줄여 읽기 쉽게 만듭니다.

Gantt 패널

Gantt는 시작/종료/기간이 있는 row를 시간 막대로 보여줍니다. span, 배치 흐름, 작업 실행 시간처럼 길이와 순서가 중요한 데이터를 볼 때 사용합니다.

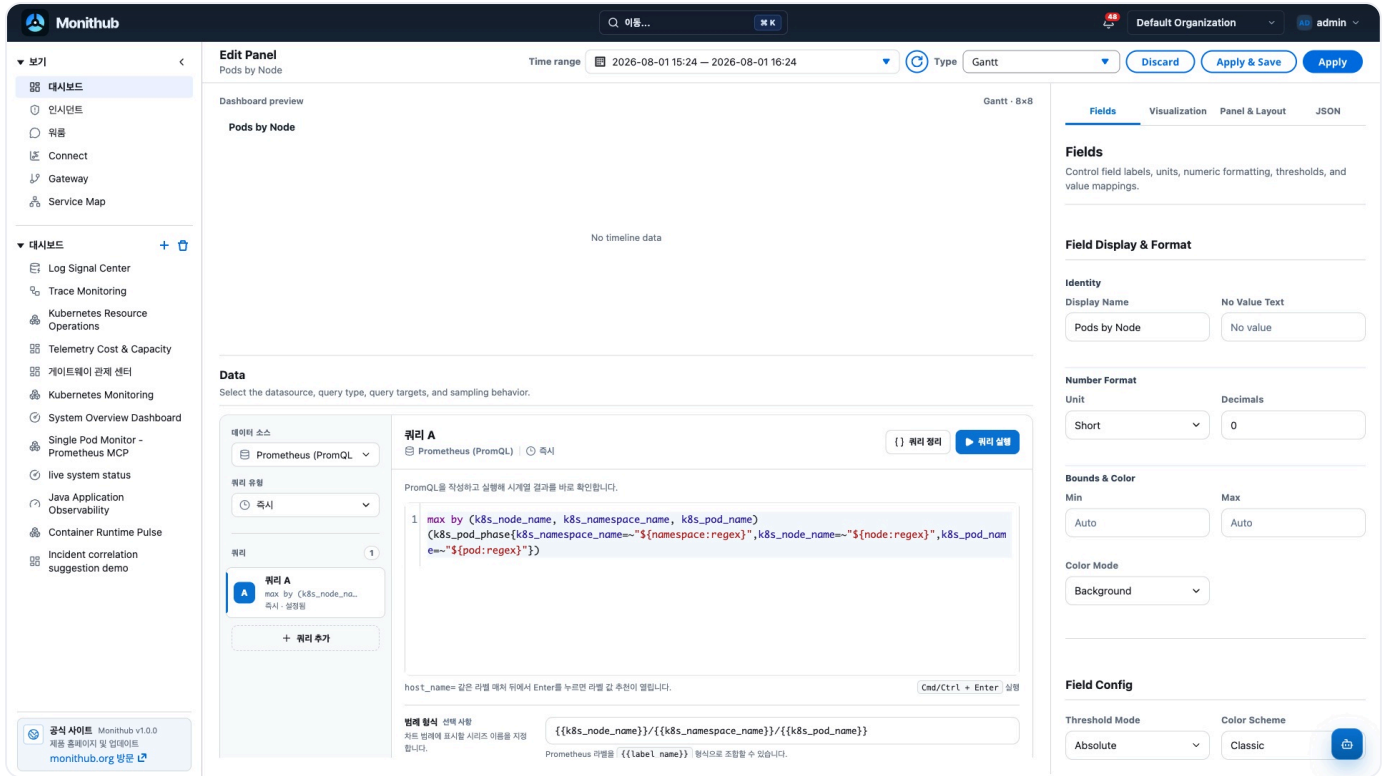
Data



Data에서는 각 row가 하나의 작업, span, 배치 단위가 되도록 query를 구성합니다. start/end 또는 start/duration, label field가 함께 반환되어야 합니다.

설정	역할
Datasource	trace span, 배치 작업, 이벤트 timeline은 ClickHouse SQL과 잘 맞습니다.
Query editor	start, end 또는 start, duration, label field를 반환합니다.
Time Filter	편집 time range와 query 조건을 맞춥니다.
Run Query	시간 단위가 ns/ms/s/unix/ISO 중 무엇인지 확인합니다.
JSON	UI에 보이지 않는 schema mapping이 있으면 함께 확인합니다.

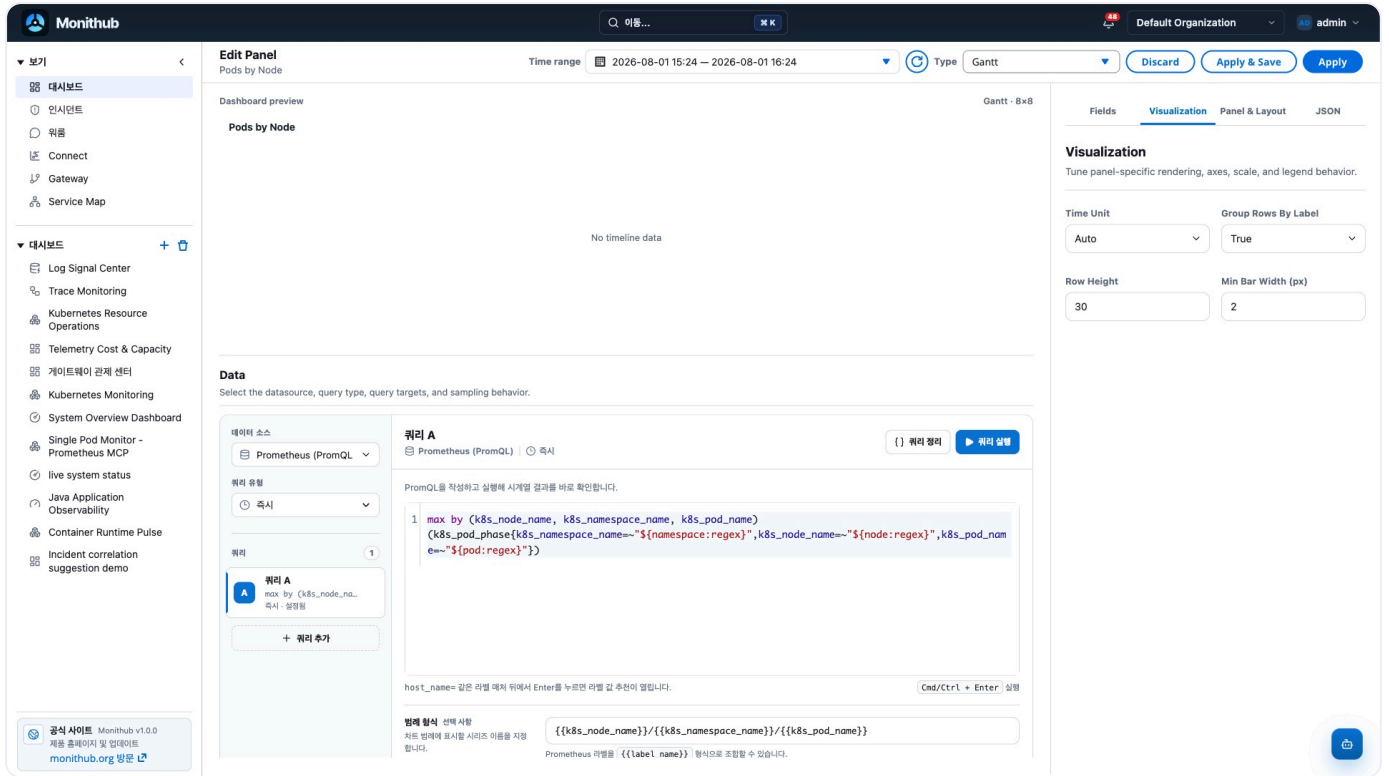
Fields



Fields에서는 duration tooltip과 숫자 표시 기준을 맞춥니다. 긴 작업이나 실패 상태를 색으로 구분하려면 threshold도 함께 정합니다.

설정	역할
Display Name / No Value Text	label이나 end 값이 비어 있을 때 표시할 문구를 정합니다.
Unit / Decimals	duration tooltip과 숫자 표시 기준입니다.
Threshold Steps	긴 작업이나 실패 상태를 색상 기준으로 표시할 때 사용합니다.
Value Mappings	상태 code를 사람이 읽는 문구로 바꿉니다.

Visualization [🔗](#)



Visualization에서는 timestamp 단위와 row grouping, 막대 크기를 정합니다. query column 이름을 바꿨다면 JSON mapping도 함께 확인합니다.

설정

역할

Time Unit	timestamp 단위를 auto, ns, ms, s, unix ms, unix s, ISO date 중 고릅니다.
Group Rows By Label	같은 label의 row를 묶어 표시합니다.
Row Height	각 row의 높이입니다. 너무 낮으면 label이 읽히지 않습니다.
Min Bar Width (px)	아주 짧은 duration도 최소 몇 px로 보이게 할지 정합니다.

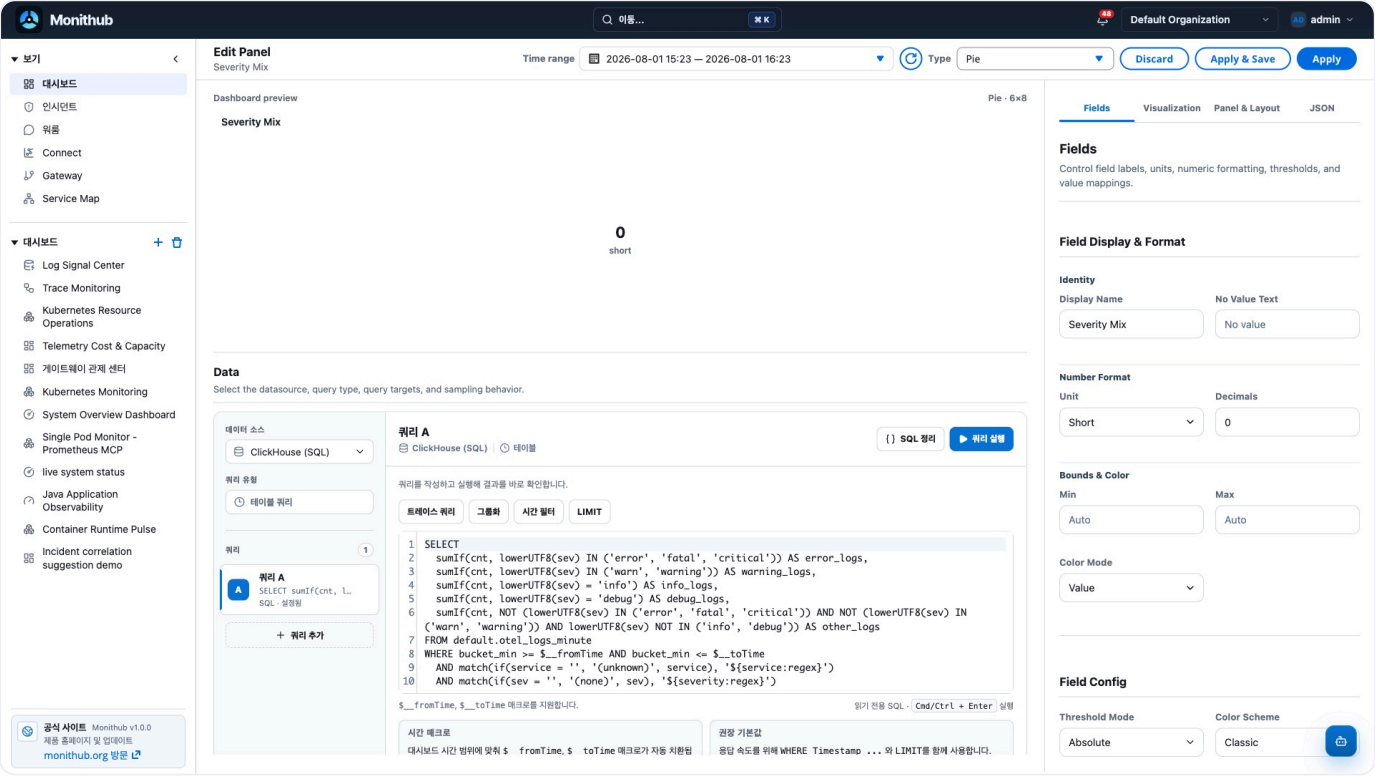
확인 순서 [🔗](#)

Data에서 start/end/duration/label field를 확인하고, Fields에서 duration 단위를 맞춘 뒤, Visualization에서 time unit과 row 높이를 실제 화면 폭에 맞춥니다.

Pie 패널

Pie는 적은 수의 category 비중을 볼 때 사용합니다. category가 너무 많거나 시간 흐름이 중요한 데이터에는 Time Series, Bar, Table이 더 적합합니다.

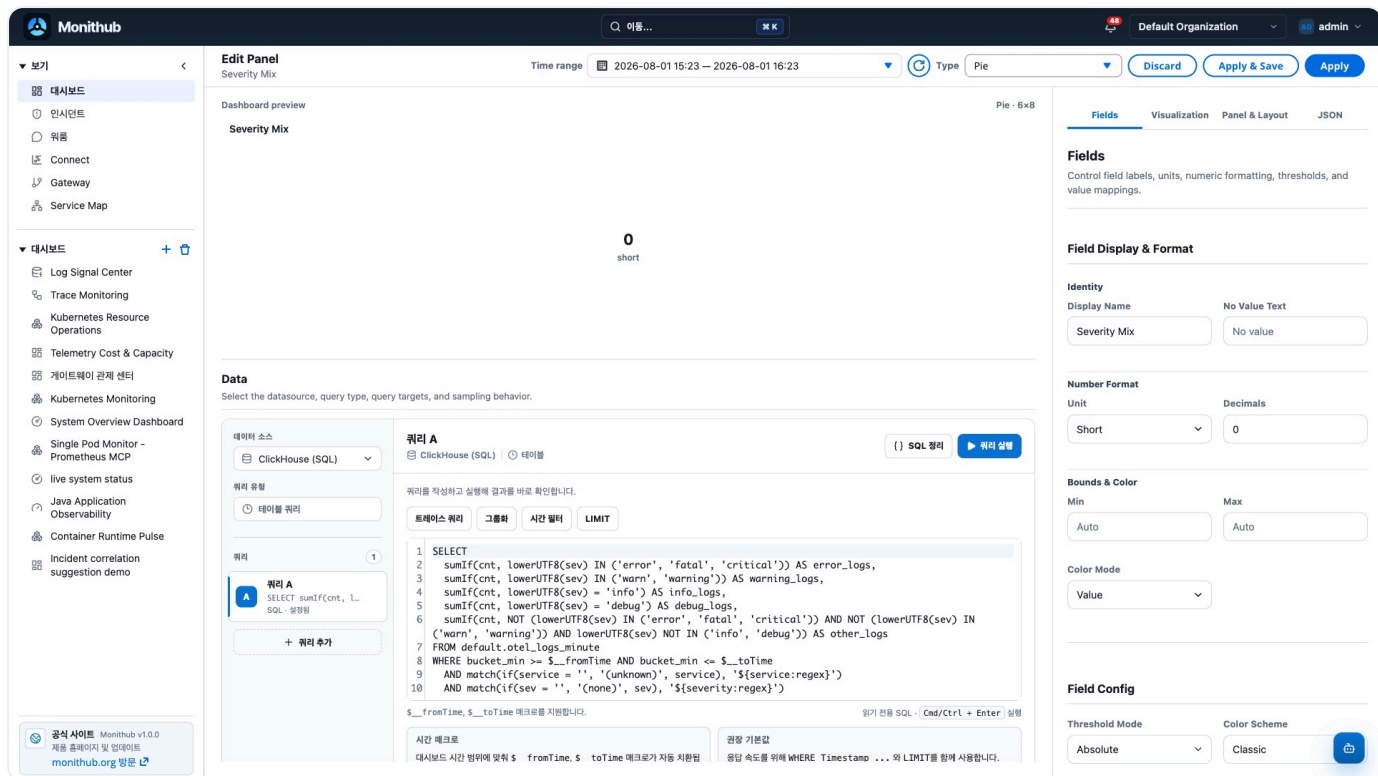
Data



Data에서는 category별 값이 적은 수로 반환되게 query를 줄입니다. 조각 수가 많아지면 pie보다 bar나 table이 더 읽기 쉽습니다.

설정	역할
Datasource	category 집계 가능한 Prometheus나 ClickHouse datasource를 선택합니다.
Query type	category별 현재 비중이면 <code>Instant</code> 나 ClickHouse 집계 query를 사용합니다.
Query editor	<code>GROUP BY service</code> , <code>sum by (status)</code> 처럼 category 기준을 query에서 먼저 줄입니다.
Legend Format	category 이름이 길면 legend가 읽기 어려우므로 짧게 정리합니다.
Run Query	category가 너무 많거나 값이 0에 물리지 않는지 확인합니다.

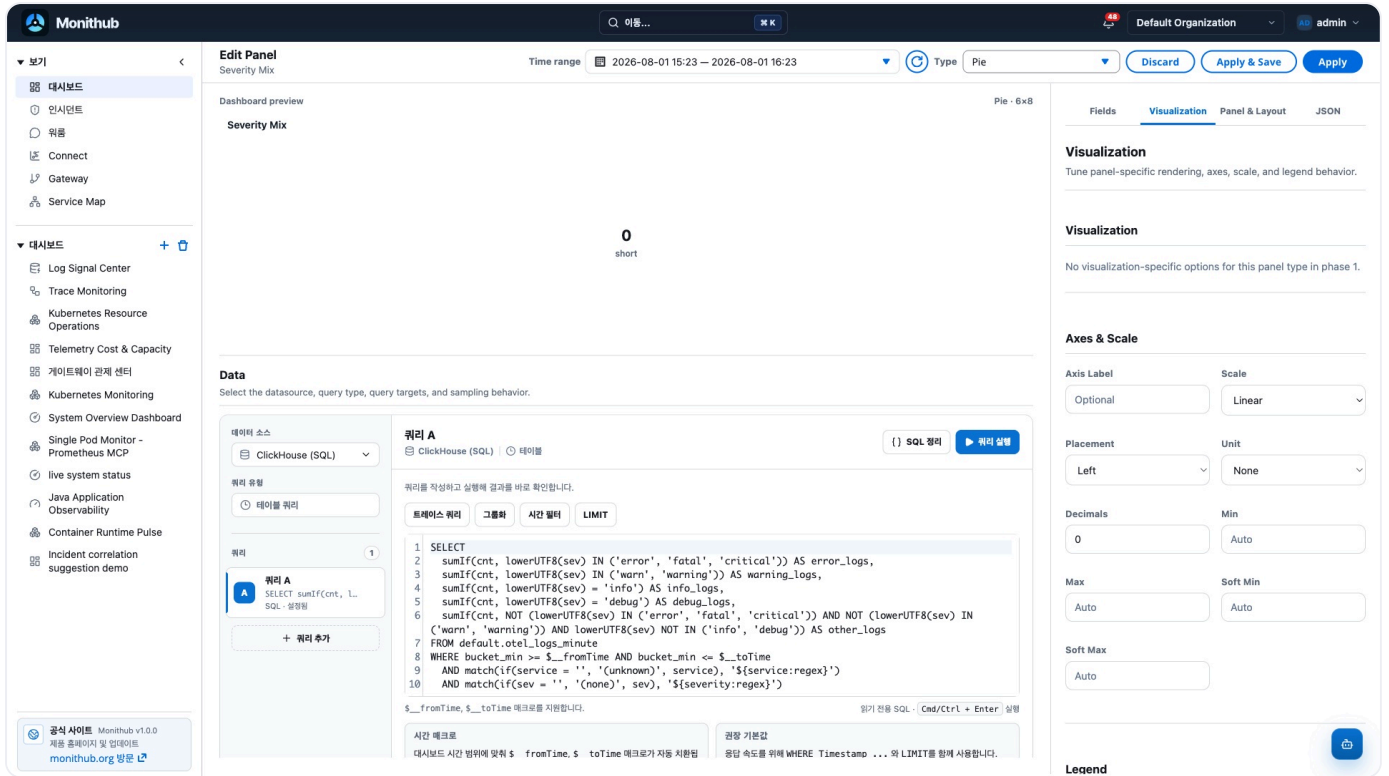
Fields



Fields에서는 조각 tooltip과 legend에 표시되는 값의 이름, 단위, 소수점, 상태 색상을 맞춥니다.

설정	역할
Display Name / No Value Text	값 field 이름과 빈 값 문구를 정합니다.
Unit / Decimals	조각 tooltip과 legend 값을 읽는 기준입니다.
Min / Max	값 범위를 명시해야 하는 경우 사용합니다.
Threshold Steps	pie 자체보다 tooltip/legend 상태 색상 해석에 영향을 줍니다.
Value Mappings	category 값이 숫자 코드일 때 표시 문구를 바꿉니다.

Visualization [🔗](#)



Pie 편집 화면에 **No visualization-specific options for this panel type in phase 1.**가 표시되면 현재 UI에서 pie 전용 옵션을 직접 조정하지 않는 상태입니다. 이때는 Data의 category 수와 Fields의 단위, 그리고 Axes & Legend의 legend 표시를 먼저 조정합니다.

설정	역할
안내 문구	전용 옵션이 없는 상태인지 확인합니다.
Axes & Legend	legend 위치, 표시 여부, category 이름 가독성을 조정합니다.
Fields	값 단위와 소수점이 조각 tooltip에 반영됩니다.
Data	category 수와 비중 계산 기준을 결정합니다.

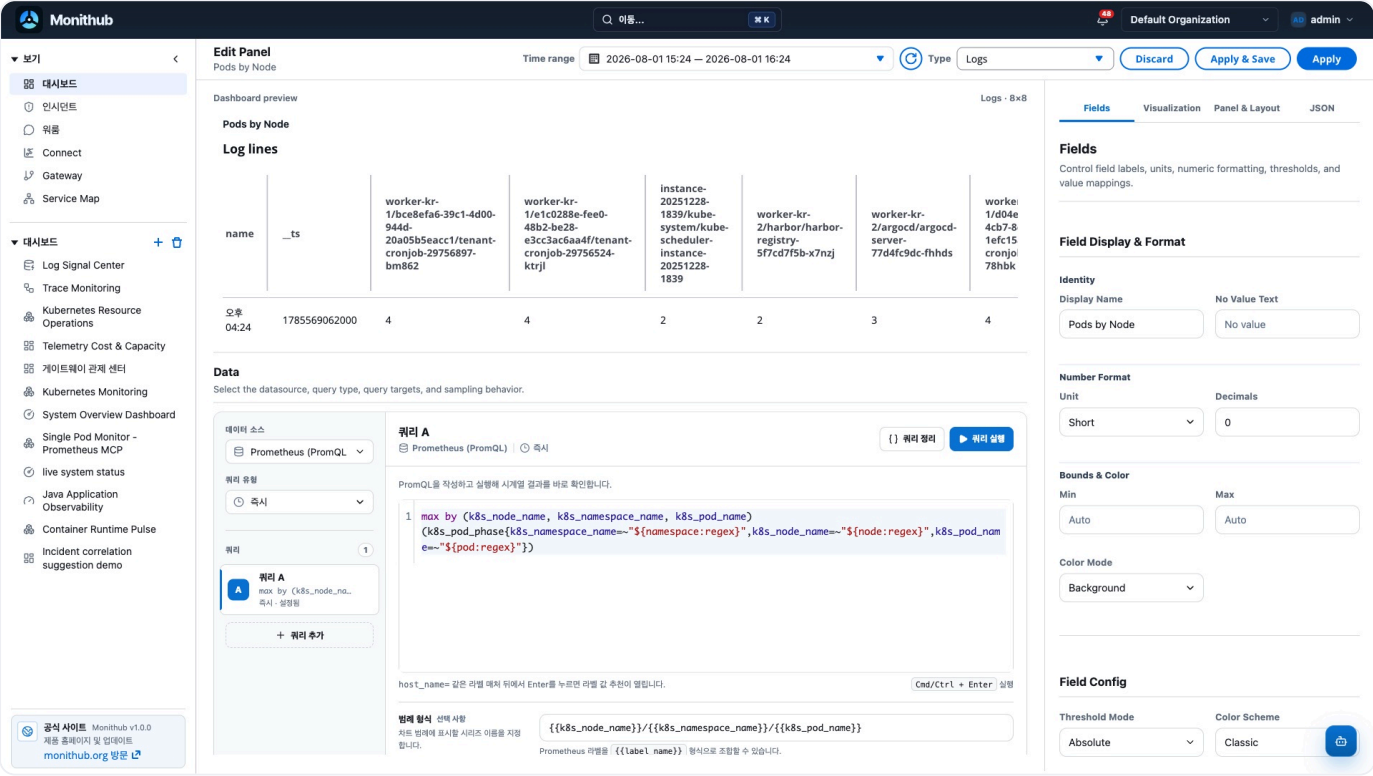
확인 순서 [🔗](#)

Data에서 category 수를 줄이고, Fields에서 unit과 decimals를 맞춘 뒤, Visualization에 전용 옵션이 없으면 Axes & Legend에서 legend를 정리합니다.

Logs 패널

Logs는 로그성 텍스트 결과를 표시할 때 사용합니다. query에서 결과를 좁히고, field formatting과 패널 크기로 읽기 좋은 로그 목록을 만듭니다.

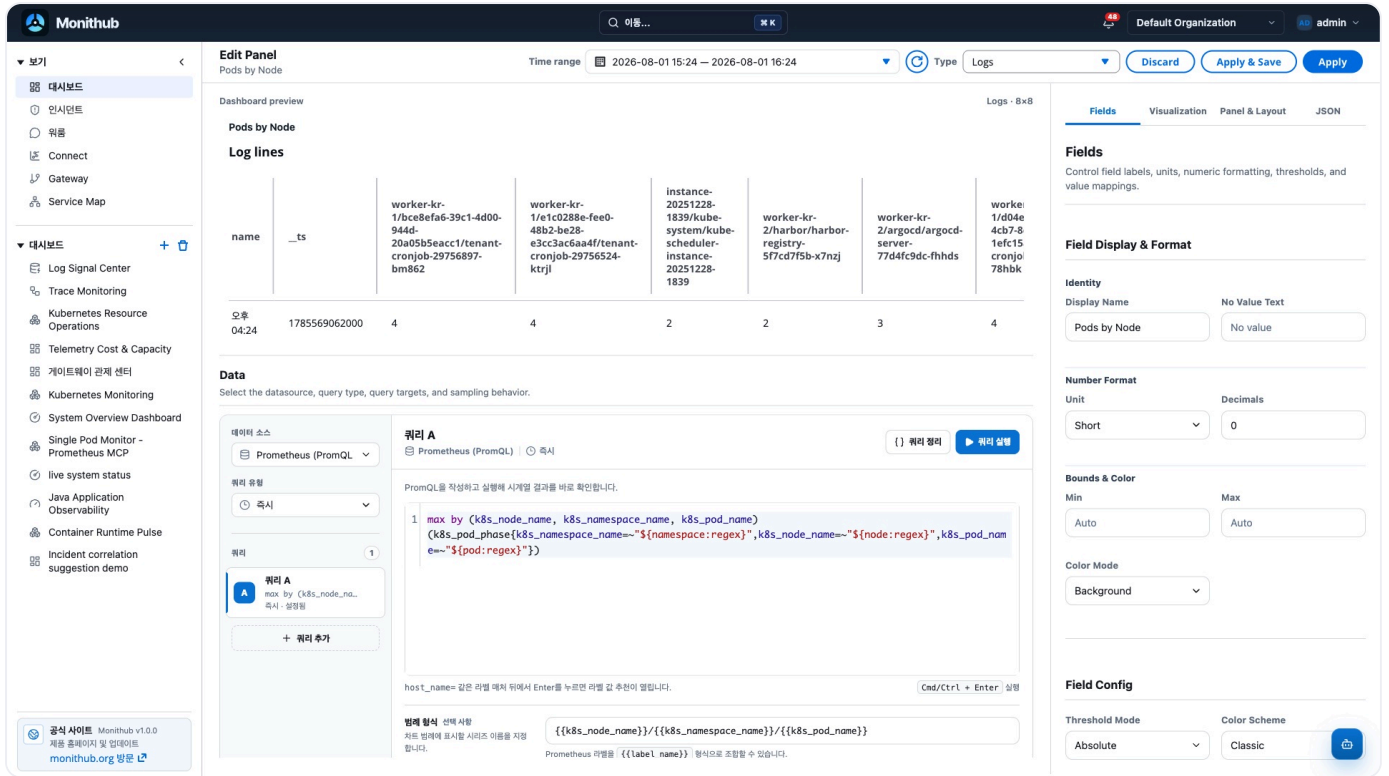
Data



Data에서는 시간 범위와 필터가 로그 row를 충분히 좁히는지 먼저 확인합니다. 로그 패널은 많은 row를 그대로 넣기보다 service, namespace, severity, time range 조건을 query에 넣는 편이 읽기 쉽습니다.

설정	역할
Datasource	로그 저장소가 ClickHouse에 있으면 SQL query를 사용합니다.
Query editor	service, namespace, severity, message, timestamp column을 반환합니다.
Time Filter	ClickHouse SQL에서는 <code>\$_fromTime</code> , <code>\$_toTime</code> 매크로로 편집 화면 time range와 맞춥니다.
LIMIT	운영 화면이 무거워지지 않도록 row 수를 제한합니다.
Run Query	반환 row와 message field가 실제 로그 표시 목적에 맞는지 확인합니다.

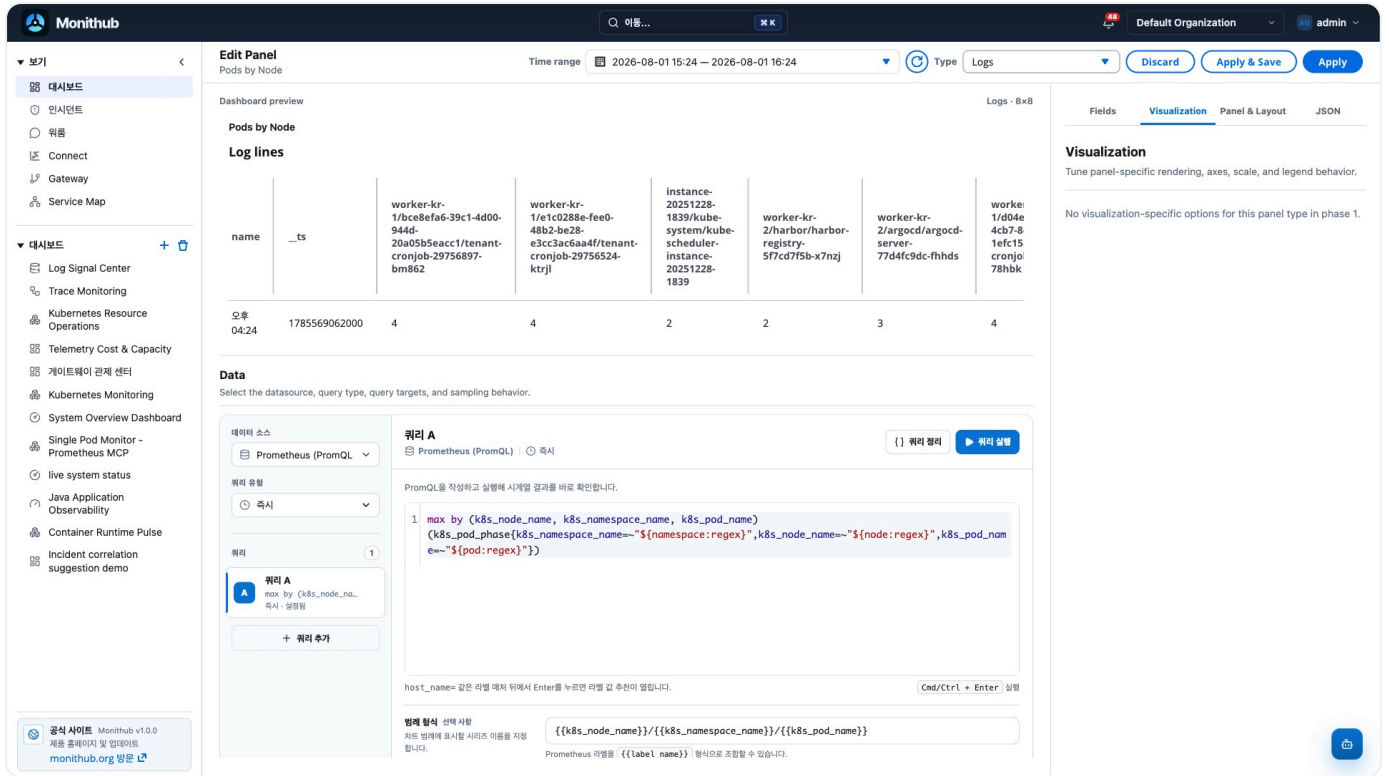
Fields



Fields에서는 로그 row의 field 이름, 빈 값 문구, severity 코드 표시를 정리합니다. 숫자 severity를 색상이나 문구로 바꾸면 목록을 빠르게 훑을 수 있습니다.

설정	역할
Display Name / No Value Text	로그 row의 field 이름과 빈 값 문구를 정합니다.
Unit / Decimals	숫자 field를 함께 표시할 때의 기준입니다.
Threshold Steps	severity 숫자값을 색상 기준으로 읽을 때 사용합니다.
Value Mappings	severity code나 상태값을 사람이 읽는 문구로 바꿉니다.

Visualization [🔗](#)



Logs 편집 화면에 **No visualization-specific options for this panel type in phase 1.**가 표시되면 현재 UI에서 로그 전용 옵션을 직접 조정하지 않는 상태입니다. 이때는 Data에서 row를 줄이고 Fields에서 severity 표시를 맞춘 뒤, 패널 높이와 JSON 설정을 함께 확인합니다.

설정	역할
안내 문구	전용 옵션이 없는 상태인지 확인합니다.
Panel & Layout	로그 줄을 읽을 수 있을 만큼 높이와 폭을 확보합니다.
JSON	UI에 아직 노출되지 않은 로그 표시 옵션을 확인합니다.
Data	row 수와 message column을 안정화합니다.

확인 순서 [🔗](#)

Data에서 시간과 severity 조건을 좁히고, Fields에서 표시 문구를 맞춘 뒤, Visualization에 옵션이 없으면 Panel & Layout과 JSON에서 표시 방식을 확인합니다.

인시던트 및 워룸 개요

인시던트와 워룸은 문제 발견부터 공동 대응, 해결, 사후 정리까지 한 흐름으로 이어 주는 Monithub의 대응 공간입니다.

- **인시던트**에서는 무엇이 발생했고 어디까지 영향을 주는지 확인합니다.
- **워룸**에서는 담당자가 근거와 판단, 조치 결과를 함께 기록합니다.

두 화면을 오가며 상태와 대응 기록을 최신으로 유지하면, 새로 합류한 사람도 같은 맥락에서 업무를 이어갈 수 있습니다.

권장 대응 흐름

1. 인시던트 목록에서 최근에도 이상이 이어지는 항목을 찾습니다.
2. 담당자를 지정하고 **확인 처리**로 대응 시작을 알립니다.
3. 상세 화면에서 주요 대상, 영향 범위, 지속 시간과 연관 신호를 확인합니다.
4. 필요한 신호를 추가하고, 정탐·오탐·무시 여부를 검토합니다.
5. 영향이 있으면 **완화** 상태로 바꾸고 조치와 관찰 기준을 기록합니다.
6. 여러 사람이 함께 대응해야 하면 연결된 워룸에서 역할과 다음 행동을 나눕니다.
7. 문제가 안정되면 해결 조건을 확인한 뒤 **해결** 처리합니다.
8. 워룸 요약과 후속 작업, 필요한 백업을 남기고 채널을 닫습니다.

먼저 구분해야 할 세 가지 동작

동작	쓰는 상황	결과
오탐 근거 제거	특정 이상 신호가 잘못 감지되었고 현재 인시던트의 근거로 쓰지 않을 때	선택한 이상 근거만 현재 인시던트에서 제거합니다.
해결 또는 닫기	실제 대응이 끝났고 기록을 보존해야 할 때	인시던트와 워룸 기록을 남긴 채 수명 주기를 마칩니다.
인시던트 삭제	잘못 만들었거나 테스트용인 인시던트 자체를 없애야 할 때	연결된 근거와 대응 기록을 영구 삭제하고 워룸을 닫습니다.

일반적인 대응 종료에는 삭제가 아니라 **해결**과 **닫기**를 사용합니다.

문서 묶음

문서	확인할 내용
인시던트 확인	목록 필터, 담당자와 상태 관리, 연관 신호 추가, 오탐 근거 제거, 인시던트 삭제
위험 협업	채널 선택, 협업 기록, AI 요약과 브리핑, 명령어, CSV·MPCA 백업
종료와 정리	해결 조건, 채널 닫기, 후속 작업과 기록 보존

처음 사용하는 경우 [인시던트 확인](#) → [위험 협업](#) → [종료와 정리](#) 순서로 읽는 것을 권장합니다.

인시던트 확인

인시던트 화면에서는 대응할 문제를 고르고, 담당자와 상태를 정한 뒤, 판단에 필요한 신호를 한곳에서 검토합니다. 처음부터 원인을 단정하기보다 현재 영향과 시간 흐름을 먼저 확인합니다.

목록에서 대응할 문제 찾기 [🔗](#)

The screenshot shows the Monithub 'Incidents' page. On the left is a sidebar with navigation links: 보기, 대시보드, 인시던트 (selected), 위험, Connect, Gateway, and Service Map. The main area is titled 'Incidents' with the subtitle 'Investigation workspace for correlated anomaly flows.' It includes a search bar 'Search incidents by title, target, or assignee' and filters for 'Anomalies >= 1 and context > 0', 'Status All', and 'Assignee All'. A 'Refresh' button is in the top right. Below the filters is a table of incidents.

Status	Incident	Target	Assignee	Anomalies	이상 발생 구간
open	worker-kr-2 anomaly	worker-kr-2 host	Unassigned	1	26. 8. 14. 오전 12:37:00 ~ 26. 8. 14. 오전 12:39:00
open	worker-kr-1 anomaly	worker-kr-1 host	Unassigned	3	26. 8. 13. 오후 11:41:00 ~ 26. 8. 14. 오전 12:39:00
open	instance-20251228-1839 anomaly	instance-20251228-1839 host	Unassigned	1	26. 8. 13. 오후 11:41:00 ~ 26. 8. 13. 오후 11:43:00
open	worker-kr-2 anomaly	worker-kr-2 host	Unassigned	1	26. 8. 13. 오후 10:40:00 ~ 26. 8. 13. 오후 10:42:00
open	instance-20251228-1839 anomaly	instance-20251228-1839 host	Unassigned	1	26. 8. 13. 오후 10:33:00 ~ 26. 8. 13. 오후 10:36:00
open	worker-kr-1 anomaly	worker-kr-1 host	Unassigned	3	26. 8. 13. 오후 8:53:00 ~ 26. 8. 13. 오후 10:36:00
open	worker-kr-2 anomaly	worker-kr-2 host	Unassigned	1	26. 8. 13. 오후 7:50:00 ~ 26. 8. 13. 오후 8:29:00
open	worker-kr-1 anomaly	worker-kr-1 host	Unassigned	1	26. 8. 13. 오후 7:44:00 ~ 26. 8. 13. 오후 7:47:00
open	worker-kr-1 anomaly	worker-kr-1 host	Unassigned	3	26. 8. 13. 오후 4:50:42 ~ 26. 8. 13. 오후 4:59:00
open	platform/analytics-engine-cfdb9c85-q87qp anomaly	platform/analytics-engine-cfdb9c85-q87qp k8s.pod	Unassigned	1	26. 8. 13. 오후 3:20:00 ~ 26. 8. 13. 오후 3:23:00

At the bottom left of the table area, it says '1-10 of 150'. At the bottom right, there are pagination controls: '< 1 2 ... 15 >'.

Footer text: 공식 사이트 Monithub v1.0.0 제품 홈페이지 및 업데이트 monithub.org 방문

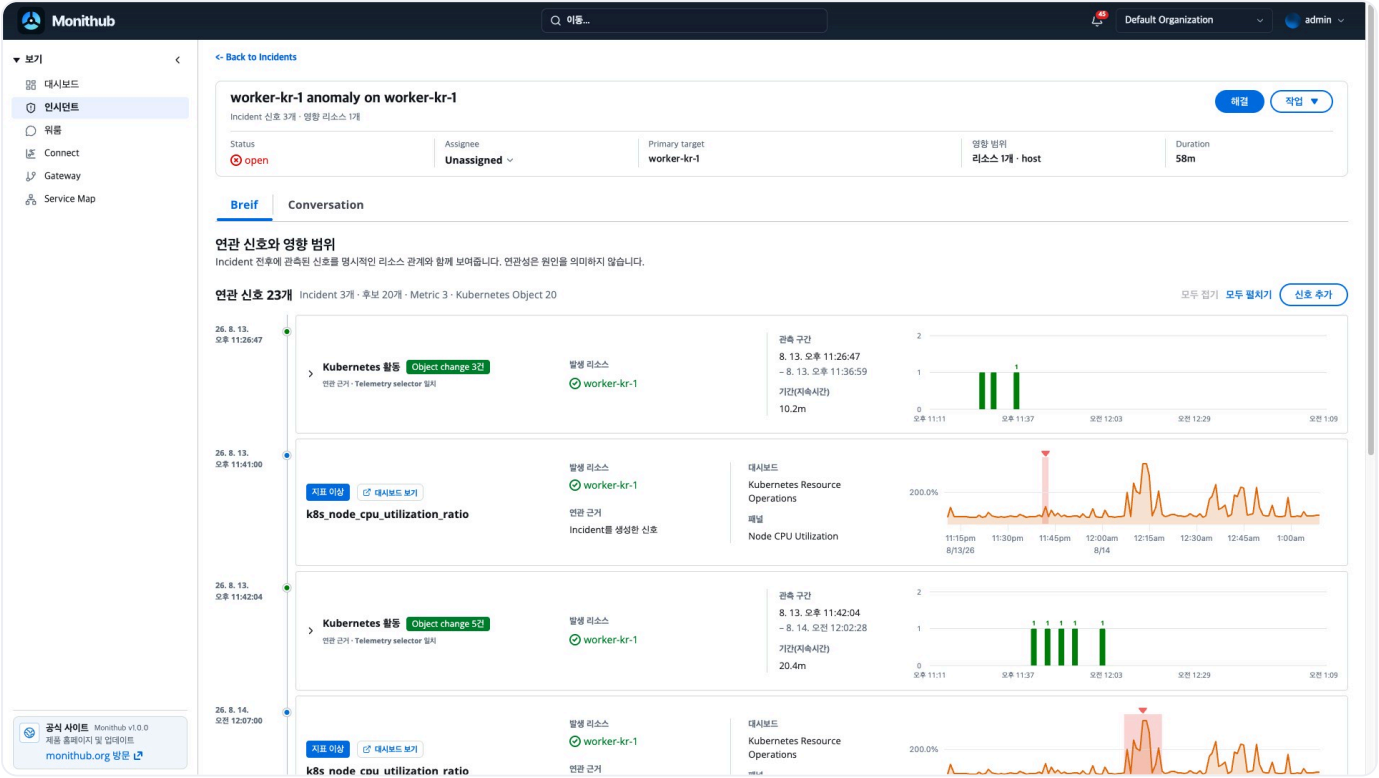
목록 상단의 검색과 필터를 함께 사용하면 대응이 필요한 항목을 빠르게 좁힐 수 있습니다.

기능 사용 방법

검색	인시던트 제목이나 대상 이름을 입력합니다.
이상 신호 수	이상 신호가 있는 인시던트만 보거나, 문맥 신호가 함께 있는 항목을 찾습니다.
상태	열림, 확인, 완화, 해결 등 현재 대응 단계로 좁힙니다.
담당자	내 담당 항목이나 미할당 항목을 찾습니다.
정렬	이상 발생 구간이나 최근 활동을 기준으로 우선순위를 정합니다.
필터 초기화	적용 중인 조건을 한 번에 지웁니다.

각 행에서 담당자를 바로 지정할 수 있습니다. 최근에도 이상이 이어지고 영향 대상이 중요하지만 담당자가 없는 항목을 우선 확인합니다.

상세에서 1차 판단하기

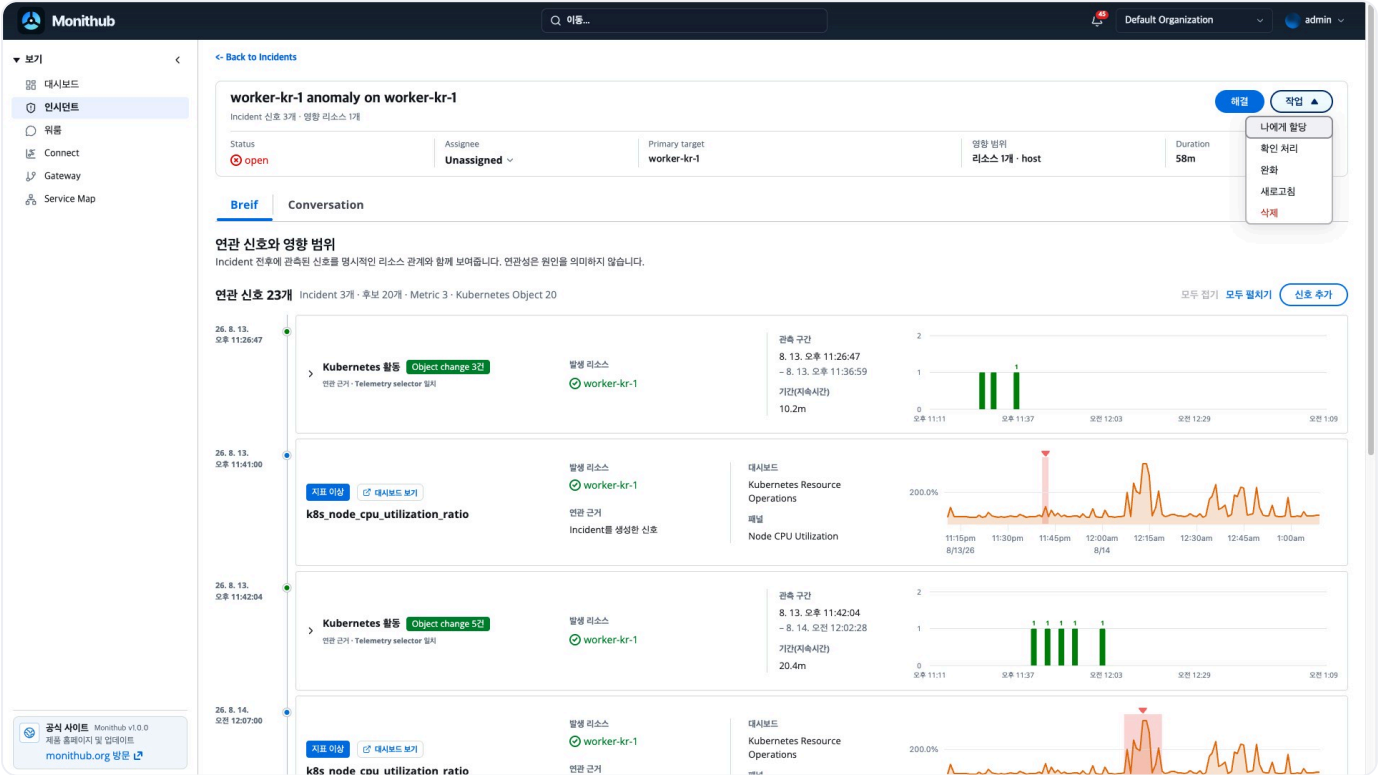


상세 화면을 열면 상단 요약부터 확인합니다.

영역	확인할 내용
상태	아직 열려 있는지, 확인 또는 완화 단계인지 봅니다.
담당자	현재 대응을 이끌 사람이 지정되어 있는지 확인합니다.
주요 대상	문제의 중심이 되는 호스트, 서비스, 워크로드를 확인합니다.
영향 범위	영향을 받는 리소스 수와 유형을 봅니다.
지속 시간	일시적인 흔들림인지 계속 이어지는 문제인지 판단합니다.
Brief	연관 신호와 영향 범위를 시간 순서로 검토합니다.
Conversation	연결된 대응 대화와 기록을 확인합니다.

연관 신호 영역에는 인시던트에 직접 연결된 근거와 후보 신호가 함께 표시됩니다. Kubernetes 활동, 메트릭 등 서로 다른 신호가 같은 시간대와 대상에서 반복되는지 비교합니다. 화면에 표시되는 연관성은 원인 확정을 뜻하지 않으므로, 조치 전에는 실제 영향과 변경 이력을 함께 확인합니다.

담당자와 상태 관리하기

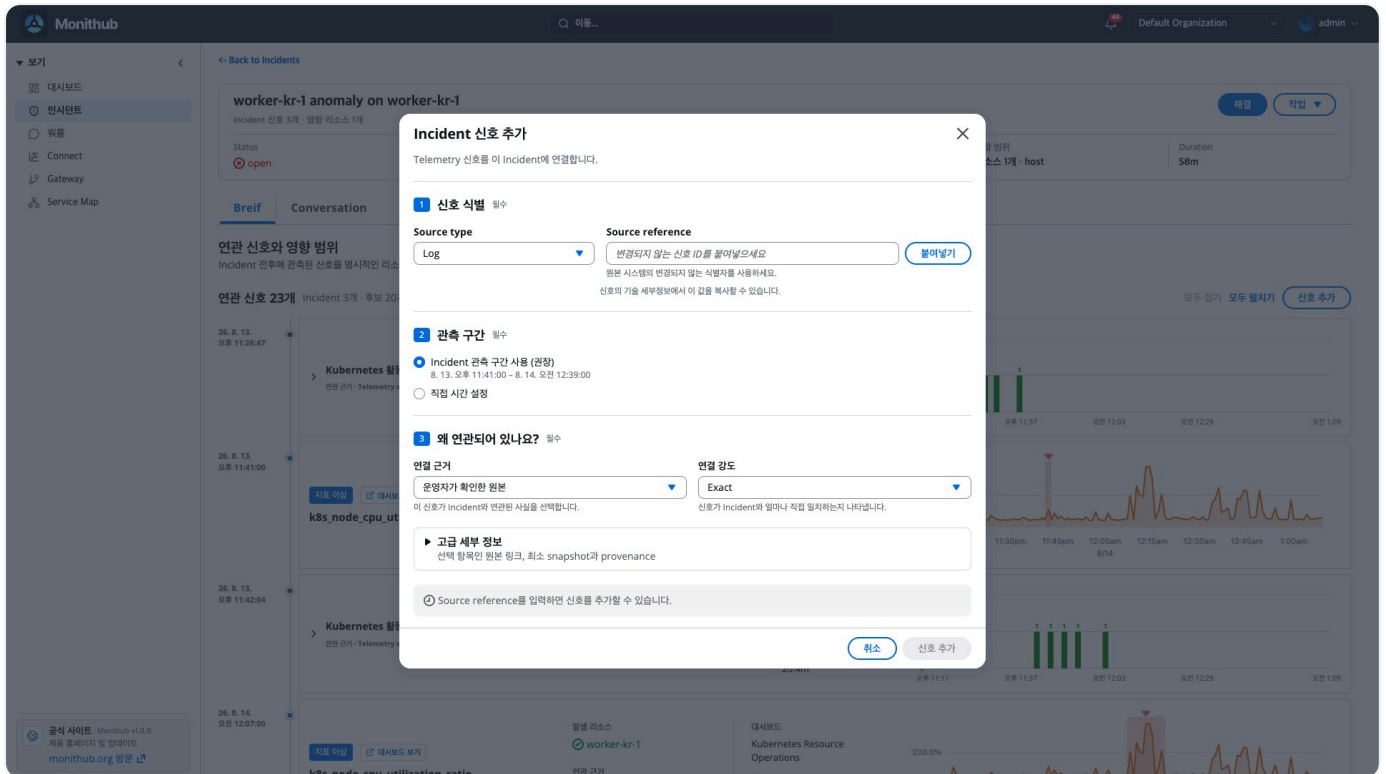


상단의 **작업** 메뉴에서 현재 상태에 맞는 동작을 선택합니다.

동작	쓰는 시점
나에게 할당	내가 대응을 맡을 때
확인 처리	누군가 문제를 확인하기 시작했음을 기록할 때
완화	영향이 계속되지만 임시 조치로 범위를 줄였을 때
새로고침	최신 상태와 연관 신호를 다시 불러올 때
해결	새 이상이 멈추고 서비스가 정상 범위로 돌아왔을 때
삭제	잘못 생성했거나 테스트용인 인시던트 전체를 없앨 때

상태를 바꿀 때는 워크룸에 판단 근거와 조치 시각을 함께 남깁니다. **완화**는 해결과 다릅니다. 임시 조치 뒤에도 관찰 기준을 충족할 때까지 대응을 이어갑니다.

필요한 신호 직접 추가하기



자동으로 묶이지 않은 근거가 있다면 연관 신호 영역의 [신호 추가](#)를 사용합니다.

1. 원본 신호의 [Source type](#)을 선택합니다.
2. 기술 세부정보에서 복사한 변경되지 않는 [Source reference](#)를 입력합니다.
3. 기본값인 인시던트 관측 구간을 사용하거나, 필요한 경우 시간을 직접 지정합니다.
4. 연결 근거와 연결 강도를 선택합니다.
5. 원본 링크나 provenance가 필요하면 [고급 세부 정보](#)에 보완합니다.
6. 내용을 확인한 뒤 [신호 추가](#)를 누릅니다.

단순히 시간대가 비슷하다는 이유만으로 신호를 추가하지 않습니다. 다른 운영자가 다시 검토해도 연결 이유를 이해할 수 있도록 근거와 강도를 실제 상황에 맞게 선택합니다.

정탐·오탐·무시로 근거 분류하기

신호 카드의 선택 버튼을 누르면 해당 이상을 분류할 수 있습니다.

분류 의미

정탐 실제 이상이며 현재 판단에 유효한 근거입니다.

오탐 이상으로 감지됐지만 실제 문제를 나타내지 않는 근거입니다.

무시 이번 판단에서 사용하지 않되, 바로 삭제하지 않고 남겨 둘 근거입니다.

분류는 원인 확정이 아니라 운영자의 검토 결과입니다. 판단 이유를 워룸에 한 줄이라도 남기면 이후 튜닝과 회고에 도움이 됩니다.

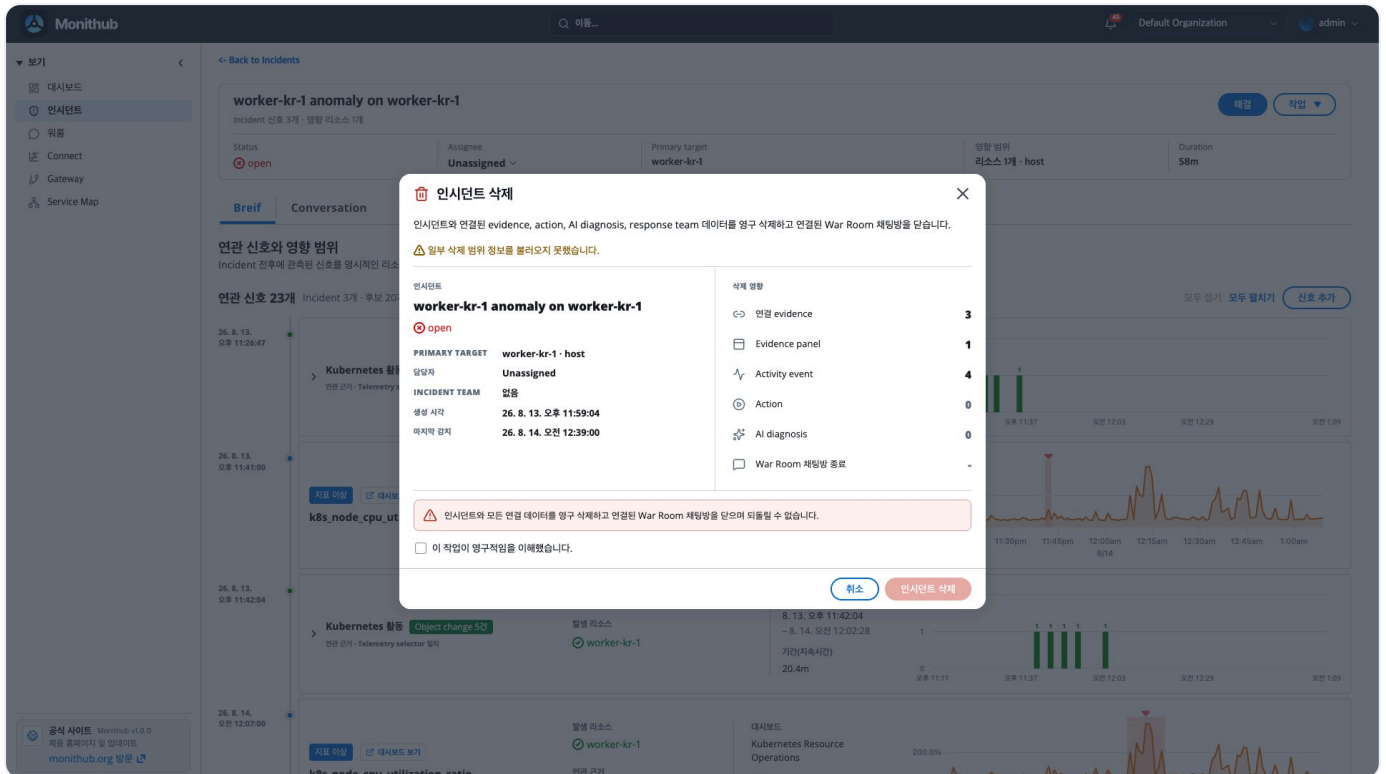
오탐으로 표시한 근거 제거하기 [🔗](#)

오탐으로 표시한 이상 근거를 현재 인시던트에서 제외하려면 다음 순서로 진행합니다.

1. 신호 카드의 선택 버튼을 엽니다.
2. **오탐**을 선택합니다.
3. 휴지통 버튼을 누릅니다.
4. 확인 창의 설명을 읽고 **삭제**를 선택합니다.

이 동작은 **선택한 오탐 근거만 현재 인시던트에서 제거**합니다. 인시던트 자체를 삭제하는 기능이 아닙니다. 판단을 더 확인해야 한다면 **유지**를 선택합니다.

인시던트 전체 삭제하기



작업 → **삭제**는 오탐 근거 제거와 범위가 다릅니다. 인시던트와 연결된 근거, 액션, AI 진단, 대응 팀 기록이 영구 삭제되고 연결된 워룸 채널도 닫힙니다.

삭제가 꼭 필요한 경우에만 다음 순서로 진행합니다.

1. 확인 창에 표시된 연결 데이터 수를 검토합니다.
2. 필요한 워룸 기록을 CSV 또는 MPCA로 먼저 보관합니다.
3. 영구 삭제 안내에 동의하는 체크박스를 선택합니다.
4. **인시던트 영구 삭제**를 실행합니다.

정상적으로 대응을 마친 인시던트는 삭제하지 않습니다. 기록을 보존한 채 **해결** 또는 **닫기** 상태로 수명 주기를 마칩니다.

검토를 워룸 대응으로 잇기

담당자와 상태를 최신으로 맞춘 뒤, 연결된 워룸에 아래 내용을 남깁니다.

- 확인된 사실과 아직 검증되지 않은 가설
- 고객 또는 사용자 영향

- 조치 담당자와 시작 시각
- 조치 전후의 지표 변화
- 해결 판단 조건과 남은 후속 작업

인시던트 상세는 근거를 모으는 곳이고, 워룸은 그 근거로 내린 판단과 행동을 공유하는 곳입니다.

워룸 협업

워룸은 인시던트 대응에 참여한 사람이 사실, 가설, 결정, 조치 결과를 시간 순서로 공유하는 공간입니다. 새로 합류한 사람도 최근 요약과 채널 정보를 통해 현재 상황을 빠르게 따라잡을 수 있습니다.

채널을 찾고 현재 상황 파악하기

Monithub

이동...

Default Organization

admin

보기

대시보드

인시던트

채널

Connect

Gateway

Service Map

인시던트 / 워룸

채널

실시간 업데이트

현재 152

열린 인시던트 150

수동 채널 1

읽기 전용 0

요약 전용 0

채널 검색

시간

채널

참여자

08.13.16:59

worker-kr-1 anomaly

#0b3365...399e

0

08.13.15:30

worker-kr-1 anomaly

#aaf77dbe...0471

0

08.13.15:23

platform/analytics-engine-cfdb9c85-q87qp anomaly

#af0e069f...0e17

0

08.13.14:43

instance-20251228-1839 anomaly

#884b035e...8198

0

08.11.13:04

instance-20251228-1839 anomaly

#74035458...31c4

3

08.10.23:47

worker-kr-1 anomaly

#4919a75c...c523

0

08.09.17:55

worker-kr-1 anomaly

#92c9ed76...bb9d

0

08.10.15:41

instance-20251228-1839 anomaly

#5b644339...ee86

1

08.09.16:35

worker-kr-2 anomaly

#184f5c34b...5411

0

08.09.13:59

worker-kr-2 anomaly

#7ac4e0d7...25cd

0

152개 채널 모두 표시 중

instance-20251228-1839 anomaly

원화 조치 승인합니다. batch-exporter를 일시 중지하고 backlog는 rate limit 적용 후 재처리하겠습니다.

test1 8월 11일 오후 01:00

batch-exporter 중지 완료했습니다. 노드 CPU가 318 → 276으로 내려오는 중이고 load average도 9.8 → 5.1입니다.

Jonghyun 8월 11일 오후 01:00

API p95도 510ms까지 회복했습니다. 오류율 0.2%, 신규 고객 문의는 없습니다. 5분 더 관찰하겠습니다.

admin (L4) 8월 11일 오후 01:00

현재 상황 상태 유지합니다. 재처리의 재개 조건은 CPU 260 이하 유지, p95 500ms 이하입니다.

test1 8월 11일 오후 01:01

CPU 254.8, load average 3.7로 안정화됐습니다. limit 미설정과 backlog burst가 직접 원인으로 보입니다.

Jonghyun 8월 11일 오후 01:01

p95 448ms로 정상 범위 복귀했습니다. 고객 영향 없음으로 정리 가능하고, 재발 방지 항목은 CPU limit과 queue rate limit입니다.

admin (L4) 8월 11일 오후 01:04

/summary

Monithub AI 8월 11일 오후 01:04

War Room summary request accepted. The generated summary will be posted here when ready.

토론 2,656개

대화 요약

가용 인스턴스에서 CPU 지표가 195.4 → 342.1로 급등해 인시던트 발생.

kubelet은 정상, 시스템 파드 재시작 사실 없음. 상위 CPU 프로세스 확인 후 확인.

고객 영향은 큰 범위로 확대되지 않음. API 오류율 0.3% 수준, p95 latency 420ms → 810ms 상승.

CPU 과부하는 batch-exporter가 limit 없이 7.4core를 사용 중이며, 12:45 배포 이후 backlog를 일괄 비움.

영향 범위는 platform-api 두 replica에 한정, timeout-결재-로그인 요청은 정상.

조치로 batch-exporter를 일시 중지하고 backlog에 rate limit 적용 후 재처리 결정.

지정된 조건(CPU ≤ 260 → 5분, p95 ≤ 500 ms)을 만족해 API p95 510ms 회복, 오류율 0.2%.

최종적으로 CPU 254.8, load average 3.7로 안정화, p95 448ms, 고객 영향 없음.

재발 방지 대책으로 CPU limit 적용 및 queue rate limit 추가를 논의.

메시지를 입력하세요. (Shift + Enter로 줄바꿈)

이 채널은 인시던트 채널입니다. 3명의 멤버가 참여 중

채널 정보

유형

공개 범위

목록 공개

상태

연결된 인시던트

74035458...31c4

시작 시각

2026. 08. 11. 오전 12:23

경과 시간

8일

멤버 3

admin (L4)

load

Jonghyun

responder

test1

responder

작업

인시던트 상세 분석

관련 로그 검색 및 분석

워크플로우 이동

채팅 기록 CSV 저장

채팅 백업 저장 (mpca)

채널 닫기

워룸은 세 영역으로 구성됩니다.

영역	확인할 내용
채널 목록	전체, 열린 인시던트, 수동 채널, 읽기 전용, 요약 전용 필터와 검색으로 채널을 찾습니다.
대화	참여 기록, 메시지, 답글, 반응, AI 요약을 시간 순서로 확인합니다.
채널 정보	채널 유형과 상태, 연결된 인시던트, 시작 시각, 경과 시간, 멤버 역할을 봅니다.
작업	인시던트 상세 분석, 관련 로그 분석, 워크플로우 이동, 기록 저장, 채널 닫기를 실행합니다.

먼저 연결된 인시던트와 채널 상태가 맞는지 확인합니다. 채널이 많다면 검색과 필터를 함께 사용하고, 최근 요약이 있으면 전체 대화를 읽기 전에 요약부터 검토합니다.

대응 기록 남기기

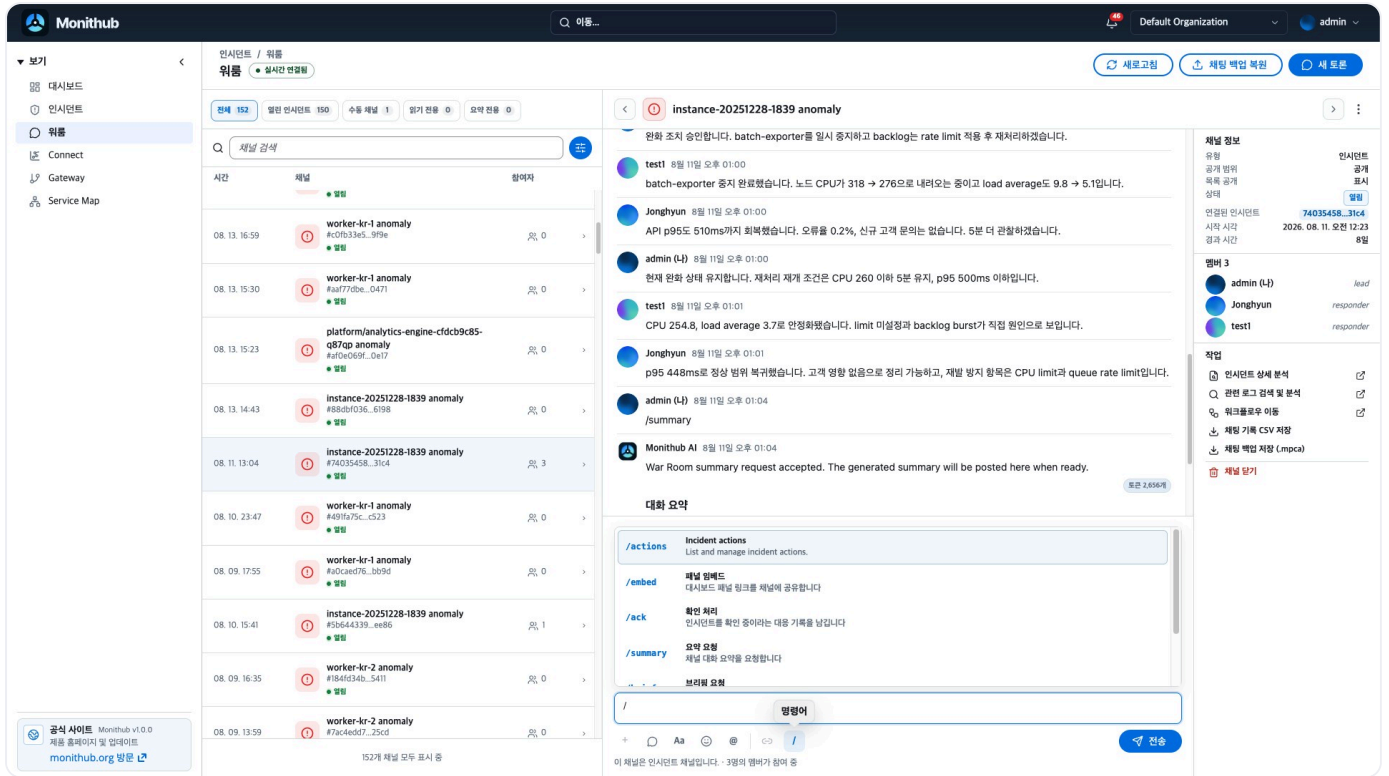
메시지는 짧더라도 사실과 다음 행동이 드러나게 작성합니다.

남길 내용	예시
최초 상황	“12:45 배포 뒤 CPU와 지연이 함께 상승했습니다.”
역할 분담	“저는 최근 배포를, @이름은 고객 영향을 확인해 주세요.”
조치	“13:00에 작업을 일시 중지했습니다.”
관찰 기준	“CPU 260 이하와 p95 500ms 이하를 5분간 확인합니다.”
결과	“13:06 기준 CPU와 p95가 정상 범위로 돌아왔습니다.”
후속 작업	“CPU limit과 queue rate limit 적용을 별도 작업으로 남깁니다.”

메시지 입력창에서는 우선순위, 서식, 이모지, 멘션을 선택할 수 있습니다. 특정 담당자는 @이름, 현재 참여자는 @here, 조직의 활성 멤버 전체는 @everyone으로 부릅니다. 긴급 멘션은 실제 영향이 크고 즉시 행동이 필요할 때만 사용합니다.

각 메시지에는 답글과 반응을 남길 수 있습니다. 별도 주제를 답글로 이어 가면 본문 흐름을 유지하면서 세부 논의를 보존할 수 있습니다.

AI 요약과 명령어 사용하기



메시지 입력창의 **/** 버튼을 누르면 자주 쓰는 작업을 선택할 수 있습니다.

명령어	기능
<code>/actions</code>	인시던트 액션을 조회하고 관리합니다.
<code>/embed</code>	함께 볼 대시보드 채널 링크를 채널에 공유합니다.
<code>/jack</code>	인시던트를 확인 중이라는 대응 기록을 남깁니다.
<code>/summary</code>	지금까지의 채널 대화를 AI가 요약하도록 요청합니다.
<code>/brief</code>	현재 인시던트의 근거를 바탕으로 AI 브리핑을 요청합니다.
<code>/handoff</code>	다음 담당자를 위한 인계 메모를 작성합니다.

`/summary`는 대화의 핵심 사실, 결정, 조치와 남은 일을 따라잡을 때 적합합니다. `/brief`는 인시던트 자체를 리드나 다른 팀에 설명할 때 유용합니다.

AI 결과는 대응을 정리하는 초안입니다. 실제 상태, 영향 범위, 원인과 조치 결과는 담당자가 원본 신호와 대화를 다시 확인하고, 틀리거나 빠진 내용은 바로 보완합니다.

오른쪽 작업 메뉴 활용하기 [🔗](#)

작업	쓰는 시점
인시던트 상세 분석	연결된 인시던트의 근거와 영향 범위를 다시 확인할 때
관련 로그 검색 및 분석	같은 시간대의 로그 단서를 찾아야 할 때
워크플로우 이동	대응 결과를 후속 절차나 작업으로 이어 갈 때
채팅 기록 CSV 저장	회고나 보고서 작성용으로 대화를 검토할 때
채팅 백업 저장 (.mpca)	Monithub에서 복원할 수 있는 형태로 워룸 맥락을 보관할 때
채널 닫기	대응과 기록 정리가 끝나 새 대화를 막을 때

CSV는 사람이 표 형태로 읽고 정리하기 위한 파일입니다. MPCA는 메시지와 답글, 반응, 명령 기록, AI 요약 등 워룸의 맥락을 복원하기 위한 전용 백업입니다. 상단의 [채팅 백업 복원](#)에서 MPCA 파일을 검토하고 가져올 수 있습니다.

MPCA에는 대화 원문과 참여자 정보가 포함될 수 있으므로 외부 공유용 파일처럼 다루지 않습니다. 조직의 보관 정책과 접근 권한을 확인합니다.

채널을 닫기 전 확인하기

1. 해결 판단의 근거와 최종 조치 시각이 남아 있습니다.
2. 조치 전후 상태와 고객 영향 판단이 기록되어 있습니다.
3. 남은 작업마다 담당자가 지정되어 있습니다.
4. 필요하면 [/summary](#) 또는 [/handoff](#)로 마지막 맥락을 정리했습니다.
5. 회고나 복원이 필요하면 CSV 또는 MPCA를 저장했습니다.
6. 연결된 인시던트도 적절한 상태로 변경했습니다.

채널을 닫으면 대화는 읽기 전용이 됩니다. 일반적인 대응 종료에는 채널 닫기를 사용하고, 인시던트 전체 삭제는 잘못 생성된 데이터처럼 기록 자체를 보존할 이유가 없을 때만 사용합니다.

종료와 정리

대응 종료는 상태만 바꾸는 일이 아닙니다. 문제가 안정됐다는 근거를 확인하고, 당시의 판단과 후속 작업을 다음 사람이 이해할 수 있게 남기는 과정입니다.

상태별 의미

상태	의미	다음 행동
열림	새 인시던트가 생성됐지만 담당과 판단이 확정되지 않았습니다.	담당자를 지정하고 영향을 확인합니다.
확인	담당자가 문제를 인지하고 조사 중입니다.	근거와 가설, 역할을 워룸에 남깁니다.
완화	임시 조치로 영향이 줄었지만 관찰이 더 필요합니다.	정상화 기준을 정하고 추이를 확인합니다.
해결	문제 현상이 멈추고 합의한 정상화 기준을 충족했습니다.	후속 작업과 요약を 정리합니다.
닫힘	대응과 기록 정리까지 끝났습니다.	회고와 재발 방지 작업에서 기록을 참조합니다.

조직의 운영 절차에 따라 일부 단계는 생략할 수 있지만, 상태를 바꾼 이유는 워룸에 남기는 것이 좋습니다.

해결 처리 전 확인할 것

1. 같은 이상 신호가 더 이상 늘어나지 않습니다.
2. 영향받던 서비스와 주요 지표가 합의한 정상 범위로 돌아왔습니다.
3. 고객 또는 사용자 영향이 해소됐습니다.
4. 마지막 조치와 관찰 결과가 워룸에 남아 있습니다.
5. 남은 작업과 담당자가 정리됐습니다.
6. 필요한 경우 AI 요약이나 인계 메모를 검토했습니다.
7. 회고 또는 복원이 필요하면 CSV나 MPCA를 저장했습니다.

해결 조건은 “지표가 잠깐 내려갔다”가 아니라, 팀이 정한 수치와 관찰 시간을 충족했는지로 판단합니다.

종료 뒤 남길 기록

- 최초 감지 시각과 실제 영향 시작 시각
- 영향받은 서비스와 사용자 범위

- 주요 판단 근거와 배제한 가설
- 실행한 조치와 담당자
- 조치 전후 지표와 정상화 시각
- 최종 해결 시각
- 늦게 발견했거나 놓친 신호
- 재발 방지 작업과 완료 기준

대화가 길었다면 [/summary](#) 결과를 초안으로 활용하되, 원인과 영향, 후속 작업은 담당자가 원문과 대조해 확정합니다.

삭제와 종료를 혼동하지 않기

상황	사용할 기능
하나의 이상 근거만 잘못 감지됨	해당 신호를 오탐으로 분류한 뒤 현재 인시던트에서 근거 제거
실제 문제 대응이 정상적으로 끝남	인시던트 해결과 워룸 채널 닫기
잘못 생성되거나 테스트용인 인시던트 전체를 없애야 함	인시던트 영구 삭제

영구 삭제는 연결된 근거와 대응 기록을 되돌릴 수 없게 제거합니다. 정상적으로 대응한 기록은 삭제하지 않고 수명 주기를 종료해 보존합니다.

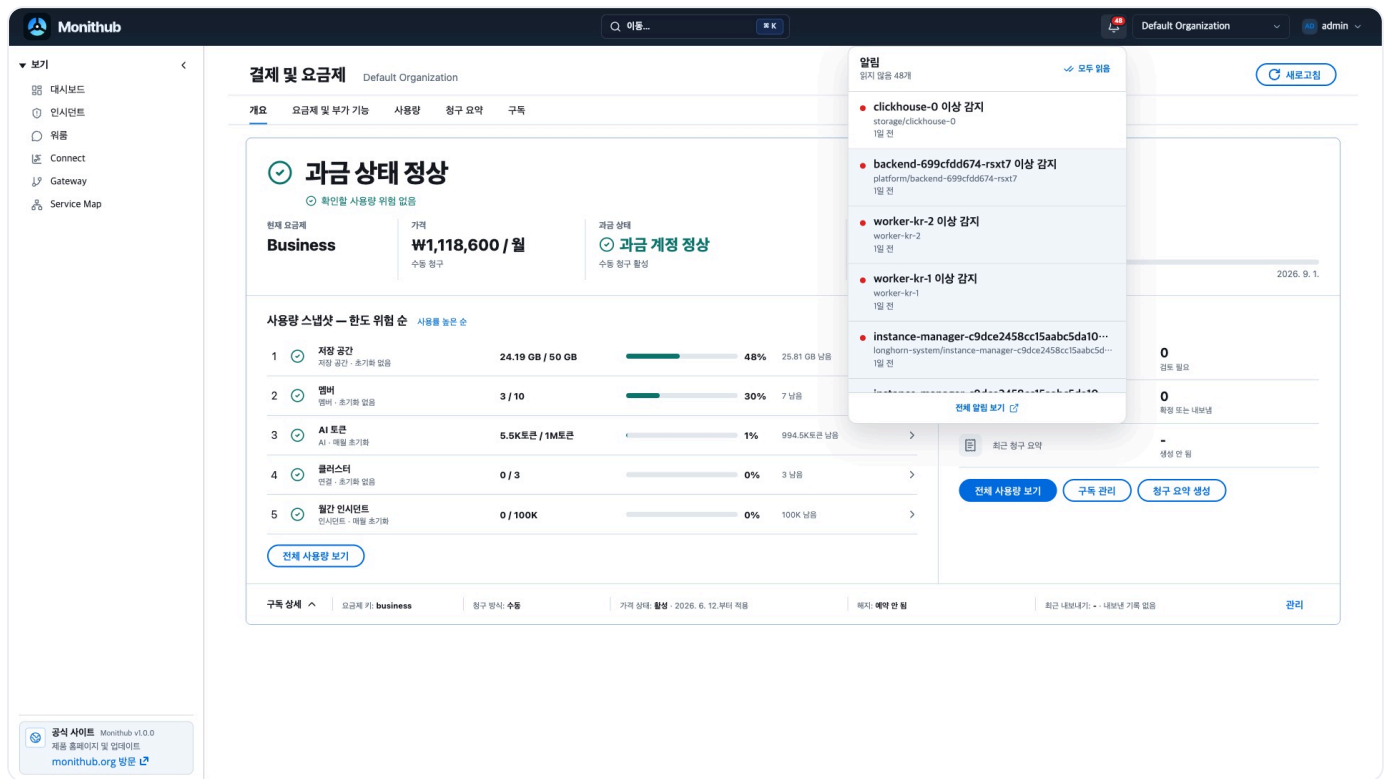
권장 마무리 순서

1. 담당자가 정상화 기준 충족 여부를 확인합니다.
2. 워룸에 마지막 조치, 결과, 남은 작업을 남깁니다.
3. 필요한 요약과 백업을 생성합니다.
4. 인시던트를 [해결](#) 처리합니다.
5. 후속 작업의 담당자와 완료 기준을 확인합니다.
6. 워룸 채널을 닫습니다.

이 흐름을 지키면 현재 대응을 끝내면서도 다음 대응에 필요한 근거를 잃지 않습니다.

알림 확인과 설정

알림에서는 현재 선택한 조직의 인시던트, Connect 수집 상태, 워룸 메시지를 한곳에서 확인할 수 있습니다. 알림을 열기 전에 상단의 조직 이름이 확인하려는 조직과 일치하는지 먼저 봅니다.



최근 알림 빠르게 확인하기

1. 화면 상단에서 확인할 조직을 선택합니다.
2. 오른쪽 위의 종 모양 **알림** 버튼을 누릅니다. 읽지 않은 알림이 있으면 버튼에 개수가 표시됩니다.
3. 펼쳐진 최근 알림에서 제목, 간단한 내용, 도착 시각을 확인합니다.
4. 바로 확인할 알림을 누릅니다. 해당 인시던트, Connect 또는 워룸 화면이 열리며 알림은 읽음으로 바뀝니다.
5. 최근 알림을 모두 정리하려면 **모두 읽음**을 누릅니다.

최근 항목만으로 찾기 어렵다면 아래의 **전체 알림 보기**를 눌러 알림 화면으로 이동합니다. 오른쪽 위 사용자 메뉴에서 **알림**을 선택해도 같은 화면을 열 수 있습니다.

필요한 알림 찾기

알림 화면을 열면 **알림 목록** 탭이 먼저 표시됩니다. 처음에는 최근 30일의 알림을 보여줍니다.

1. 제목이나 내용의 일부를 알고 있다면 **제목 또는 내용 검색**에 키워드를 입력합니다.
2. 아직 확인하지 않은 항목만 보려면 **미읽음만**을 켭니다.
3. 더 자세히 좁히려면 **필터**를 누르고 **읽음 상태**, **유형**, **심각도**를 선택합니다.
4. **기간**에서 최근 24시간, 7일, 30일, 전체 기간 또는 사용자 지정 기간을 선택합니다.
5. 조건을 다시 처음으로 돌리려면 **초기화**를 누릅니다.

유형은 **인시던트**, **워룸**, **Connect** 중에서 고를 수 있습니다. **심각도**는 인시던트 알림을 찾을 때 사용하며, 심각도를 선택하면 인시던트 유형으로 맞춰집니다.

The screenshot shows the Monithub web application. On the left is a sidebar with navigation links: 대시보드, 인시던트, 워룸, Connect, Gateway, and Service Map. The main area is titled '알림' (Alerts) for the 'Default Organization'. It features a search bar with the placeholder '제목 또는 내용 검색', filter buttons, and a dropdown for '미읽음만' (Only unread) and '최근 30일' (Last 30 days). Below this is a table of alerts. One alert is selected, showing its details on the right. The alert is titled 'clickhouse-0 이상 감지' (clickhouse-0 anomaly detected) and is categorized as 'Critical'. The details include a message 'A new Incident was created.', a signal type of '인시던트' (Incident), a namespace of 'storage', a resource of 'clickhouse-0', a target of 'k8s.pod:storage/clickhouse-0', an incident ID of '0874e3f0-6c05-48cb-a771-4955074c15c8', and a source of 'incident'. There is also a 'JSON' button to view the alert's payload.

목록에서 알림을 선택하면 오른쪽에 생성 시간과 신호 정보가 표시됩니다. 화면이 좁을 때는 목록 아래에 상세 내용이 이어집니다.

- 내용만 확인하고 목록에 남아 있으려면 **읽음으로 표시**를 누릅니다.
- 관련 화면에서 이어서 확인하려면 **인시던트 열기**, **Connect 열기** 또는 **War Room 열기**를 누릅니다. 이 때 알림도 읽음으로 바뀝니다.
- 현재 조건에 보이는 항목과 관계없이 읽지 않은 알림을 모두 정리하려면 화면 위쪽의 **모두 읽음**을 누릅니다.

어떤 알림을 받을지 정하기 [🔗](#)

알림 화면에서 **설정** 탭을 선택합니다. **알림 종류**에서는 다음 기준을 확인하거나 변경할 수 있습니다.

알림 종류	언제 받는지
인시던트	새 인시던트가 생기면 조직에 참여 중인 모든 멤버가 받습니다. 이 항목은 항상 켜져 있습니다.
Connect 수집 상태	수집이 지연되거나 중단되었을 때와 정상으로 돌아왔을 때 모든 멤버가 받습니다. 이 항목은 항상 켜져 있습니다.
위룸 메시지 멘션만	위룸에서 나를 멘션한 메시지만 받습니다. 기본 선택값입니다.
위룸 메시지 모든 메시지	내가 참여 중인 위룸에서 다른 사람이 보낸 모든 메시지를 받습니다.
지	
위룸 메시지 끄기	위룸 메시지 알림을 받지 않습니다.

위룸 알림 기준을 누르면 바로 저장되고 **알림 설정을 저장했습니다.**라는 안내가 표시됩니다. 내가 직접 보낸 위룸 메시지는 알림으로 다시 오지 않습니다.

알림을 받을 위치 정하기 [🔗](#)

전송 채널에서 알림을 어디에서 확인할지 정합니다.

위치	동작
Platform 알림 함	위에서 정한 알림을 현재 조직의 알림함에 항상 저장합니다. 끌 수 없는 기본 채널입니다.
Web Push	Monithub 화면을 보고 있지 않을 때 현재 브라우저와 운영체제의 알림으로 알려줍니다. 조직과 브라우저마다 따로 켜집니다.
Telegram	연결한 개인 Telegram 계정으로 같은 알림을 받습니다. 연결 방법은 Telegram 연결과 워를 전송 에서 확인합니다.

Web Push를 사용하려면 [이 브라우저](#)를 켜 뒤 브라우저의 알림 권한 요청에서 허용합니다. 설정 아래에 [현재 조직에서 이 브라우저로 전송 중입니다.](#)가 표시되면 완료입니다.

조직을 바꾼 뒤에도 Web Push가 필요하다면 새로 선택한 조직의 알림 설정에서 다시 켜줍니다.

Monithub를 보고 있을 때 알림 받기

Monithub 화면에 포커스가 있을 때 새 알림이 오면 화면에 알림 배너가 나타납니다. [앱 사용 중 동작](#)의 [알림음](#)을 켜면 배너와 함께 소리가 재생됩니다.

1. [설정](#) 탭의 [앱 사용 중 동작](#)으로 이동합니다.
2. [알림음](#)을 켜줍니다.
3. 오른쪽의 스피커 모양 [알림음 테스트](#) 버튼을 눌러 소리를 확인합니다.
4. [테스트 알림음을 재생했습니다.](#)가 표시되고 소리가 들리면 완료입니다.

2초 안에 여러 알림이 도착하면 배너에는 알림 개수가 함께 표시되고 소리는 한 번만 재생됩니다. 알림함에는 각 알림이 따로 남습니다.

알림이 보이거나 들리지 않을 때

Web Push가 차단된 경우

설정에 [브라우저가 알림 권한을 차단했습니다.](#)가 표시되면 다음 순서로 확인합니다.

1. 브라우저의 사이트 설정에서 Monithub의 알림 권한을 [허용](#)으로 바꿉니다.
2. Monithub 화면을 새로고침합니다.
3. [알림](#) → [설정](#) → [전송 채널](#)로 돌아와 [이 브라우저](#)를 다시 켜줍니다.

이 브라우저에서는 Web Push를 지원하지 않습니다.가 표시되면 Platform 알림함을 사용하거나 Web Push를 지원하는 다른 브라우저에서 다시 설정합니다.

알림음이 들리지 않는 경우 [🔗](#)

1. Monithub의 [알림음](#)이 켜져 있는지 확인합니다.
2. Monithub 화면을 한 번 누른 뒤 [알림음 테스트](#)를 다시 누릅니다.
3. Monithub 탭이 음소거되어 있지 않은지 확인합니다.
4. Monithub 화면이 현재 선택되어 있는지, 기기 음량이 충분한지 확인합니다.

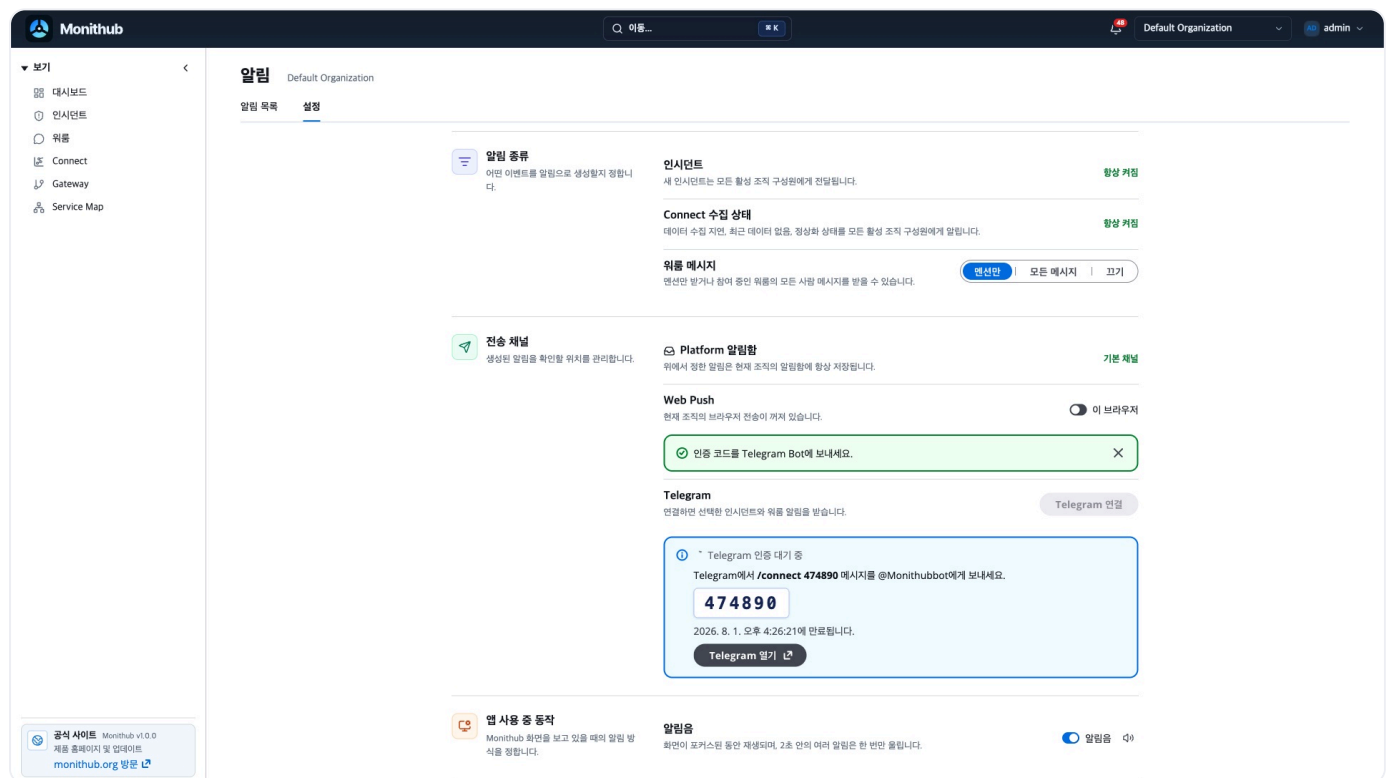
앱 알림음은 Monithub 화면을 보고 있을 때만 재생됩니다. 화면을 보고 있지 않을 때의 소리와 표시 방식은 브라우저와 운영체제의 알림 설정을 따릅니다.

Telegram 연결과 워룸 전송

개인 Telegram 계정을 Monithub에 연결하면 인시던트, Connect 수집 상태, 선택한 워룸 메시지 알림을 Telegram에서 받을 수 있습니다. Telegram Bot과의 개인 대화에서 작성한 텍스트를 접근 가능한 워룸으로 보낼 수도 있습니다.

Telegram 계정 연결하기

1. Monithub 화면 상단에서 연결할 조직을 선택합니다.
2. 오른쪽 위 종 모양 알림 버튼을 누르고 전체 알림 보기를 선택합니다.
3. 설정 탭으로 이동합니다.
4. 전송 채널의 Telegram에서 Telegram 연결을 누릅니다.
5. 화면에 표시된 Bot 이름, 6자리 코드, 만료 시각을 확인합니다.



6. Telegram 열기를 누르거나 Telegram에서 화면에 표시된 Bot을 직접 엽니다.
7. Bot과의 개인 대화에 Monithub 화면에 표시된 /connect 6자리코드 메시지를 그대로 보냅니다.
8. Bot이 연결 완료 메시지를 보냈는지 확인합니다.
9. Monithub로 돌아와 Telegram 계정 이름과 연결됨 상태가 표시되는지 확인합니다.

연결 코드는 10분 동안만 사용할 수 있습니다. 코드가 만료되었거나 연결에 실패했다면 Monithub에서 [Telegram 연결](#)을 다시 눌러 새 코드를 만든 뒤 처음부터 진행합니다. 연결 코드는 다른 사람에게 공유하지 않습니다.

Telegram에서 알림 확인하기 [🔗](#)

연결이 완료되면 현재 조직의 알림 설정에 따라 Telegram으로 알림이 옵니다.

- 새 인시던트와 Connect 수집 상태 알림은 항상 전달됩니다.
- 워룸 메시지는 [멘션만](#), [모든 메시지](#), [끼기](#) 중 선택한 기준을 따릅니다.
- Telegram 알림에 포함된 링크를 누르면 관련 인시던트, Connect 또는 워룸 화면으로 이동할 수 있습니다.

Telegram만을 위한 별도 알림 종류 설정은 없습니다. 받을 워룸 메시지를 바꾸려면 Monithub의 [알림 → 설정 → 알림 종류](#)에서 워룸 기준을 변경합니다. Web Push를 끄더라도 Telegram 연결은 유지됩니다.

Telegram 메시지를 워룸으로 보내기 [🔗](#)

Telegram Bot과의 개인 대화에서 일반 텍스트를 보내면 메시지를 전달할 워룸을 선택할 수 있습니다.

1. 연결한 Monithub Bot과의 개인 대화를 엽니다.
2. 워룸에 남길 내용을 일반 텍스트로 입력해 보냅니다. [/](#)로 시작하는 명령어가 아닌 텍스트를 사용합니다.
3. Bot이 보여주는 워룸 목록에서 메시지를 보낼 곳을 선택합니다. 열려 있고 내가 접근할 수 있는 워룸만 표시됩니다.
4. 목록이 여러 페이지라면 [Next →](#) 또는 [← Previous](#)로 이동해 원하는 워룸을 찾습니다.
5. Bot이 [Sent to War Room](#)과 선택한 워룸 이름을 표시하는지 확인합니다.
6. 필요한 경우 Monithub에서 해당 워룸을 열어 메시지가 등록되었는지 확인합니다.

워룸 선택은 메시지를 보낸 뒤 10분 동안만 유효합니다. 시간이 지나 [This message has expired](#)가 표시되면 같은 내용을 다시 보내고 워룸을 다시 선택합니다. 전송하지 않으려면 [Cancel](#)을 누릅니다.

Telegram에서는 일반 텍스트만 워룸으로 보낼 수 있습니다. 파일이나 사진은 이 방법으로 전송할 수 없습니다.

연결된 조직과 도움말 확인하기

Telegram Bot과의 개인 대화에서 다음 명령어를 사용할 수 있습니다.

명령어	사용하는 때
<code>/start</code>	연결 방법과 사용할 수 있는 기능을 처음부터 확인할 때
<code>/connect 6자리코드</code>	Monithub에서 만든 코드로 개인 계정을 연결할 때
<code>/sync</code>	이 Telegram 계정에 연결된 Monithub 조직을 확인할 때
<code>/help</code>	계정 연결, 조직 확인, 위룸 전송 방법을 다시 볼 때

여러 조직에서 같은 Telegram 계정을 사용하려면 Monithub에서 조직을 하나씩 선택하고 각 조직의 [알림](#) → [설정](#) → [Telegram](#)에서 연결 상태를 확인합니다.

계정을 바꾸거나 연결 해제하기

다른 Telegram 계정으로 바꾸기

1. Monithub에서 바꿀 조직을 선택합니다.
2. [알림](#) → [설정](#) → [전송 채널](#)의 Telegram 영역으로 이동합니다.
3. [재연결](#)을 누릅니다.
4. 새로 표시된 `/connect 6자리코드`를 바꿀 Telegram 계정의 Bot 대화에 보냅니다.
5. Monithub에 새 계정 이름과 [연결됨](#) 상태가 표시되는지 확인합니다.

현재 조직의 연결 끊기

1. 연결을 끊을 조직이 상단에 선택되어 있는지 확인합니다.
2. Telegram 영역에서 [해제](#)를 누릅니다.
3. [Telegram 연결을 해제했습니다.](#)가 표시되고 [Telegram 연결](#) 버튼이 다시 나타나는지 확인합니다.

연결을 해제하면 해당 조직의 Telegram 알림을 더 이상 받지 않고, Telegram에서 해당 조직의 위룸으로 메시지를 보낼 수 없습니다. 다른 조직의 연결은 그대로 유지됩니다.

연결이나 전송이 되지 않을 때

화면에 보이는 상황	다음에 할 일
인증 코드가 만료되었다는 안내	Monithub에서 Telegram 연결 또는 재연결 을 눌러 새 코드를 만든 뒤 다시 보냅니다.
코드가 올바르지 않다는 Bot 안내	/connect 뒤에 Monithub에 표시된 6자리 숫자를 공백 하나로 구분해 다시 보냅니다.
연결된 조직이 없다는 Bot 안내	Monithub에서 현재 조직의 Telegram 연결 상태를 확인하고 새 코드로 연결합니다.
선택할 워룸이 없다는 Bot 안내	Monithub에서 내가 참여할 수 있는 열린 워룸이 있는지 확인하거나 초대를 요청합니다.
워룸 전송 실패 안내	선택한 워룸이 아직 열려 있고 내가 메시지를 쓸 수 있는지 확인한 뒤 다시 보냅니다.
Telegram 알림이 오지 않음	Monithub에서 올바른 조직과 Telegram 계정이 연결되어 있는지, 워룸 알림 기준이 끄기 로 되어 있지 않은지 확인합니다.
문제가 계속되면 Telegram에서 /help 를 보내 현재 사용할 수 있는 명령과 연결 순서를 다시 확인합니다.	

Connect 시작하기

Connect는 서버, 애플리케이션, 데이터베이스, Kubernetes의 관측 데이터를 Monithub에 연결하는 화면입니다. 수집 대상과 운영 환경을 선택하면 설치 명령과 설정 파일이 현재 조직에 맞게 생성됩니다.

시작하기 전에

- 데이터를 저장할 조직과 수집 대상에 접근할 권한이 필요합니다.
- 수집 대상에서 <https://otel.monithub.org>로 HTTPS 통신할 수 있어야 합니다.
- 데이터베이스 비밀번호와 TLS 인증서는 Connect에 입력하지 않습니다. 생성된 안내에 표시된 환경 변수나 경로를 Collector 실행 환경에 직접 설정합니다.
- 생성된 설정과 토큰은 공개 저장소, 메신저, 티켓에 공유하지 않습니다.

첫 연결 만들기

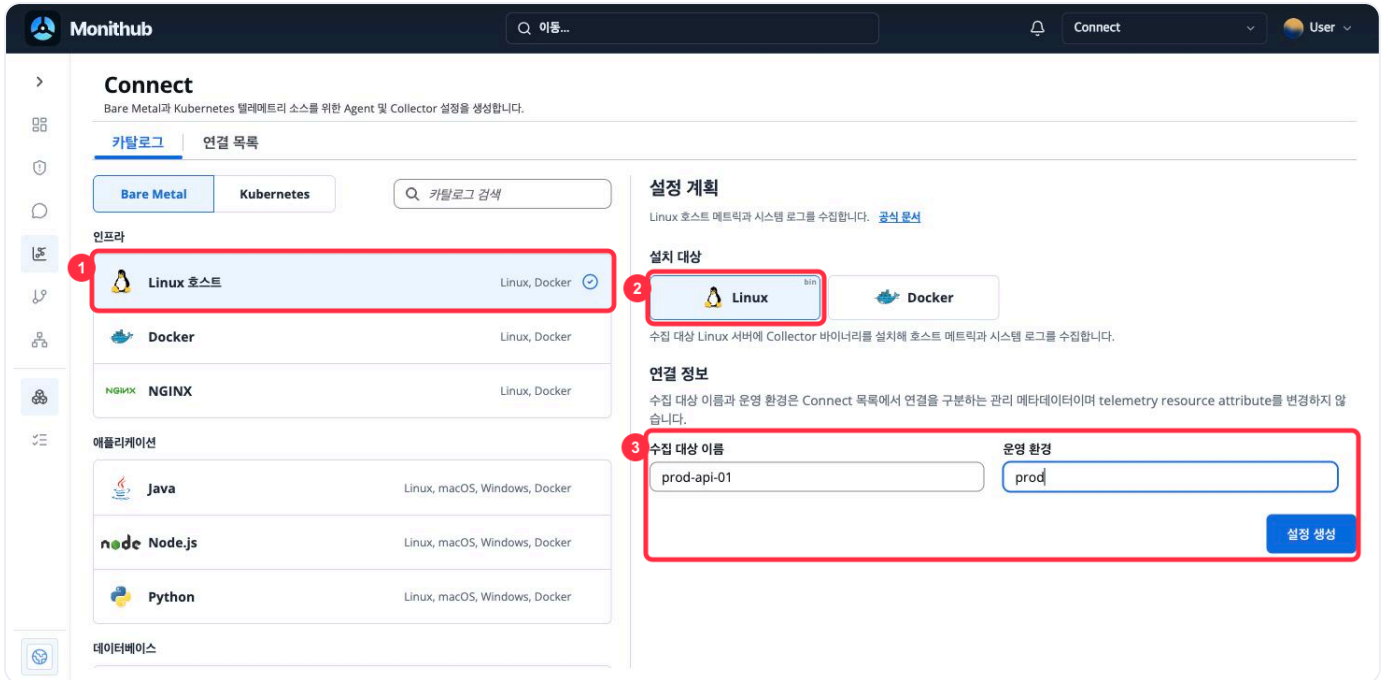
1. Monithub 상단에서 데이터를 저장할 조직을 선택합니다.
2. [Connect](#)에서 연결할 카탈로그와 설치 대상을 선택합니다.
3. [설정 계획](#)에서 설치 방식을 선택합니다.
4. [연결 정보](#)에 수집 대상 이름과 운영 환경을 입력합니다. 애플리케이션은 서비스 이름을, Kubernetes는 클러스터 ID를 입력합니다.
5. 필요한 추가 항목을 선택하고 [설정 생성](#)을 누릅니다.
6. 생성된 블록을 표시된 순서대로 실제 수집 대상에 적용합니다.
7. 애플리케이션 요청, 데이터베이스 쿼리, 컨테이너 실행처럼 수집할 활동을 발생시킵니다.
8. [데이터 수집 확인](#)과 [연결 목록](#)에서 신호별 수집 결과를 확인합니다.

[설정 생성](#)을 누르면 연결 준비도 함께 시작됩니다. 별도의 시작 버튼을 누를 필요는 없습니다.

화면 따라 연결하기: Linux 호스트 예시

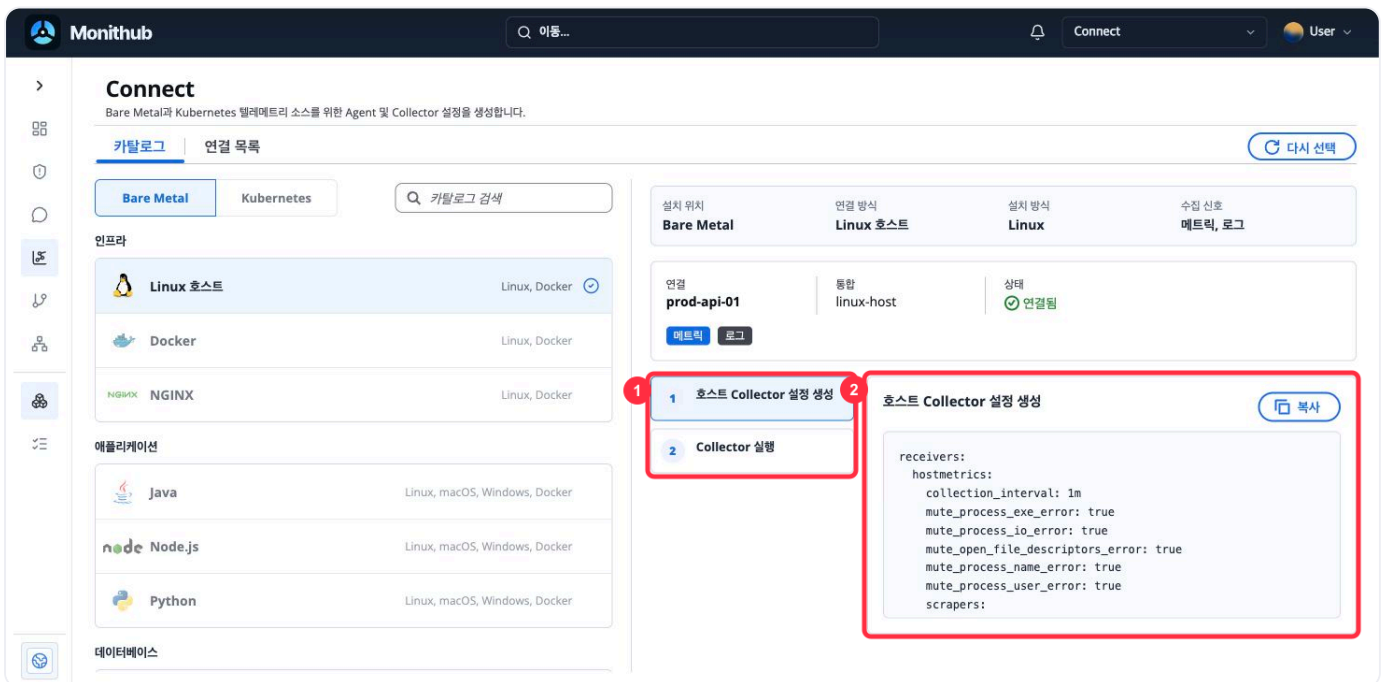
다음은 Linux 서버의 호스트 메트릭과 시스템 로그를 수집하는 예시입니다. 다른 카탈로그도 대상과 설치 환경을 선택하고, 생성된 설정을 적용하는 순서는 같습니다.

1. 대상과 설치 환경 선택하기



1. **Linux 호스트** 카탈로그를 선택합니다.
2. Collector를 실행할 실제 환경인 **Linux**를 선택합니다.
3. 수집 대상 이름과 운영 환경을 입력하고 **설정 생성**을 누릅니다.

2. 생성된 YAML을 파일로 저장하기 [🔗](#)



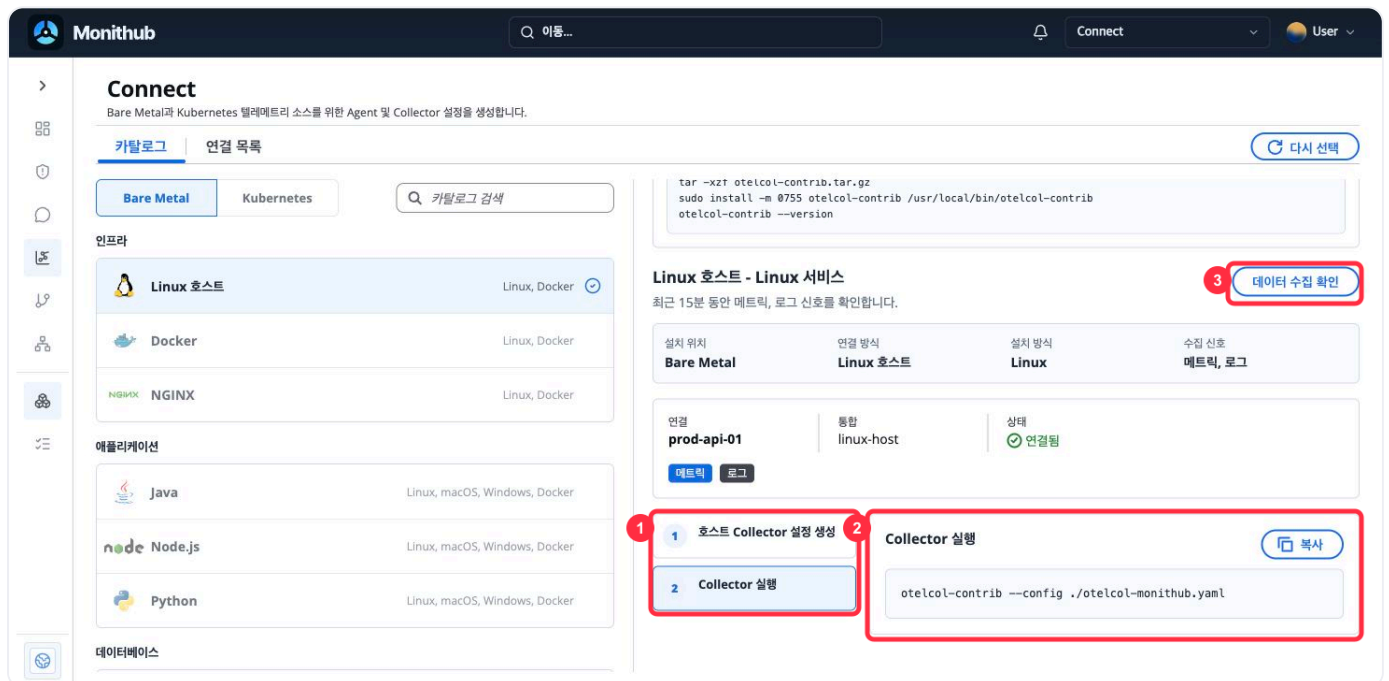
1. **호스트 Collector 설정 생성** 단계를 선택합니다.
2. 오른쪽의 **복사**를 눌러 YAML 전체를 복사합니다.

수집 대상 Linux 서버에서 작업 폴더를 만든 뒤 설정 파일 입력을 시작합니다.

```
mkdir -p ~/monithub-connect/linux-host
cd ~/monithub-connect/linux-host
cat > otelcol-monithub.yaml
```

복사한 YAML 전체를 붙여 넣고 Ctrl+D를 눌러 저장합니다. 문서의 화면에 보이는 일부 YAML만 옮기거나 인증값을 직접 바꾸지 마세요.

3. Collector 실행하고 확인하기 [🔗](#)



1. **Collector 실행** 단계를 선택합니다.
2. 오른쪽의 실행 명령을 복사해 같은 작업 폴더에서 실행합니다.
3. Collector가 실행된 뒤 **데이터 수집 확인**을 누릅니다. 호스트 메트릭은 수집 주기 이후 표시될 수 있습니다.

```
otelcol-contrib --config ./otelcol-monithub.yaml
```

생성된 설정 적용하기 [🔗](#)

Connect에 표시된 전체 내용을 복사해 해당 블록이 안내하는 위치에 적용합니다. 문서의 예시 값을 대신 사용하거나 서로 다른 연결의 설정을 섞지 않습니다.

생성되는 항목	적용 위치
설치 명령	Collector 또는 Agent를 실행할 서버의 터미널
Collector YAML	안내된 파일명의 설정 파일
환경 변수	애플리케이션이나 Collector를 시작하는 셸·배포 환경
실행 명령	실제 애플리케이션 또는 Collector 실행 환경
Dockerfile 조각	애플리케이션 이미지의 Dockerfile
Helm values	연결할 Kubernetes 클러스터의 배포 작업 폴더

Collector 설정은 위 예시처럼 Connect에서 생성된 전체 내용을 파일로 저장한 뒤 실행합니다. 파일명과 명령은 Connect에 생성된 값을 우선합니다.

이미 Collector나 지원되는 애플리케이션 설정을 사용 중이라면 [기존 설정 수정](#)을 선택하고 [기존 설정에 적용](#)을 따릅니다.

어떤 카탈로그를 선택할까요?

연결할 대상	카탈로그
이미 OTLP를 보내는 서비스	커스텀 OTLP
Java, Node.js, Python 서비스	애플리케이션
Linux 서버, Docker Engine, NGINX	인프라
PostgreSQL, MySQL·MariaDB, Redis	데이터베이스
Kubernetes 클러스터	Kubernetes

지원 신호와 환경을 비교하려면 [전체 카탈로그](#)를 확인하세요.

데이터 수집 확인하기

Collector나 애플리케이션이 실행된 것만으로 수집이 확인되지는 않습니다. 대상에서 실제 활동을 발생시킨 뒤 필요한 신호의 최근 데이터가 표시되는지 확인합니다.

- 애플리케이션: HTTP 요청, 배치 작업, 메시지 처리
- Linux·Docker·Kubernetes: 일반적인 시스템 또는 컨테이너 활동
- 데이터베이스: 안전한 조회 쿼리
- NGINX: `stub_status`가 활성화된 서버로 요청
- 커스텀 OTLP: 앞단 송신기에서 실제 OTLP 데이터 전송

상태의 의미와 다시 확인하는 방법은 [연결 및 수집 확인](#)에서 안내합니다.

다음 문서

- [전체 카탈로그](#): 지원 대상, 신호, 설치 환경 비교
- [Custom OTLP](#): 기존 OpenTelemetry 송신기 연결
- [애플리케이션](#): Java, Node.js, Python 연결
- [인프라](#): Linux 호스트, Docker, NGINX 연결
- [데이터베이스](#): PostgreSQL, MySQL·MariaDB, Redis 연결
- [Kubernetes](#): Kubernetes 클러스터 연결

전체 카탈로그

수집하려는 대상과 신호를 기준으로 카탈로그를 선택하세요. 설치 환경은 Collector 또는 Agent를 실제로 실행할 환경을 뜻합니다.

Custom OTLP

카탈로그	수집 범위	설치 환경
커스텀 OTLP	앞단에서 OTLP로 전달한 메트릭, 트레이스, 로그	Linux, macOS, Windows, Docker

이미 계측된 송신기를 연결하는 방법은 [Custom OTLP 연결](#)에서 확인하세요.

애플리케이션

카탈로그	수집 범위	설치 환경
Java 애플리케이션	지원 라이브러리의 트레이스, JVM 메트릭, Logback·Log4j2 로그	Linux, macOS, Windows, Docker
Node.js 애플리케이션	지원 라이브러리의 트레이스, Node·V8 메트릭, 지원 logger 로그	Linux, macOS, Windows, Docker
Python 애플리케이션	지원 라이브러리 텔레메트리, 표준 <code>logging</code> 로그	Linux, macOS, Windows, Docker

설치 흐름은 [애플리케이션 연결](#)에서 확인하세요.

인프라

카탈로그	수집 범위	설치 환경
Linux 호스트	Linux 호스트 메트릭과 시스템 로그	Linux, Docker
Docker	Docker 컨테이너 런타임 메트릭	Linux, Docker
NGINX	<code>stub_status</code> 의 연결·요청 메트릭	Linux, Docker

설치 흐름은 [인프라 연결](#)에서 확인하세요.

데이터베이스

카탈로그	수집 범위	설치 환경
PostgreSQL	PostgreSQL 서버와 데이터베이스 메트릭	Linux, macOS, Windows, Docker
MySQL / MariaDB	MySQL·MariaDB 서버 메트릭	Linux, macOS, Windows, Docker
Redis	Redis INFO 메트릭	Linux, macOS, Windows, Docker

데이터베이스 카탈로그는 로그와 쿼리 본문을 수집하지 않습니다. 연결 방법과 자격 증명 적용 위치는 [데이터베이스 연결](#)에서 확인하세요.

Kubernetes

Kubernetes 카탈로그는 클러스터 환경의 메트릭, 트레이스, 로그 수집을 지원합니다. 적용할 리소스와 권한은 [Kubernetes 연결](#)에서 확인하세요.

공식 문서

지원 범위는 Connect 화면의 안내와 다음 OpenTelemetry 문서를 함께 확인하세요.

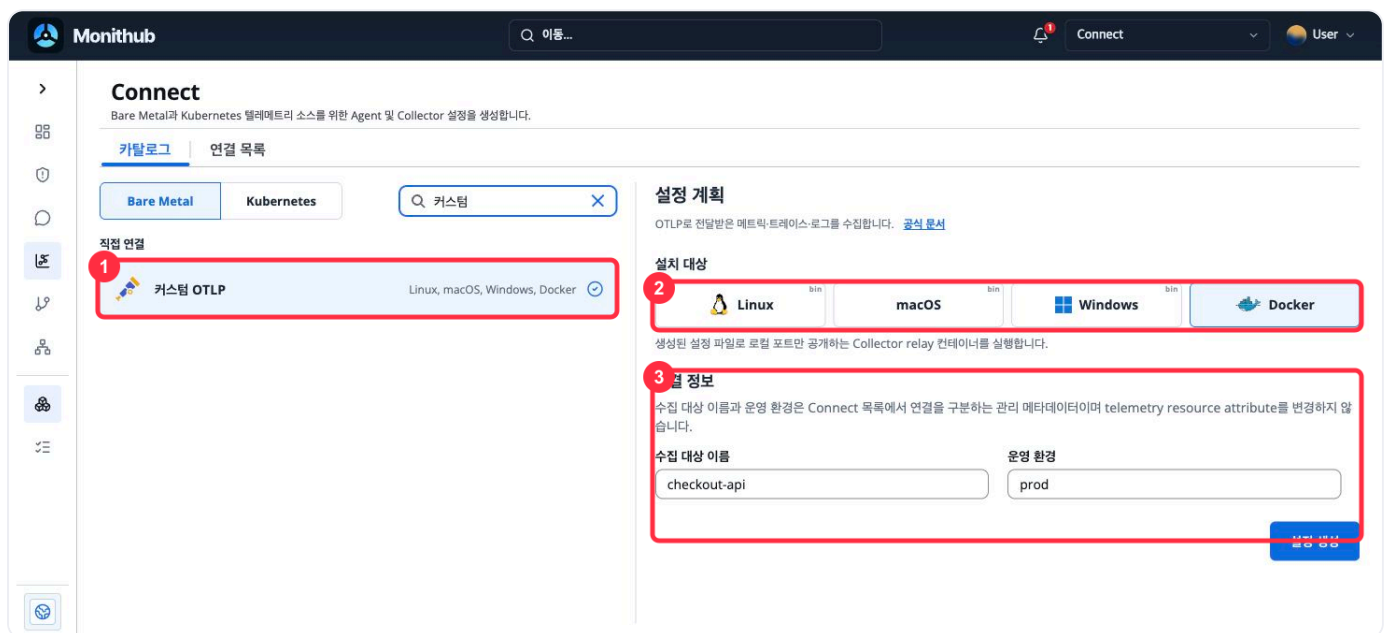
- [OpenTelemetry Collector receiver 목록](#)
- [OpenTelemetry Kubernetes Collector 구성 요소](#)
- [OpenTelemetry Collector 문제 해결](#)

Custom OTLP 연결

Custom OTLP는 특정 애플리케이션이나 인프라를 새로 계측하는 카탈로그가 아닙니다. 이미 OpenTelemetry SDK, Agent 또는 Collector가 보내는 OTLP 데이터를 Monithub로 전달할 때 사용합니다.

수집되는 신호는 앞단 송신기가 실제로 보내는 데이터에 따라 달라집니다. 메트릭, 트레이스와 로그 중 하나만 보내고 있다면 해당 신호만 확인됩니다.

설정 생성 [🔗](#)



1. **커스텀 OTLP**를 선택합니다.
2. OTLP 송신기 가까이에서 Collector를 실행할 환경을 선택합니다.
3. 목록에서 구분할 수집 대상 이름과 운영 환경을 입력하고 **설정 생성**을 누릅니다.

Collector 실행과 송신 주소 적용 [🔗](#)

Connect가 생성한 `otelcol-monithub.yaml` 전체를 저장하고 Collector를 실행한 뒤, 기존 송신기의 OTLP 주소를 해당 Collector로 변경합니다.

```
otelcol-contrib --config ./otelcol-monithub.yaml
export OTEL_EXPORTER_OTLP_ENDPOINT="http://127.0.0.1:4318"
export OTEL_EXPORTER_OTLP_PROTOCOL="http/protobuf"
# 기존 송신기 또는 애플리케이션을 다시 시작합니다.
```

위 주소는 OTLP HTTP 예시입니다. OTLP gRPC 송신기는 Connect가 생성한 [4317](#) 주소를 사용합니다. Docker 방식도 호스트에는 로컬 주소로만 포트를 공개하므로 다른 서버에 그대로 노출하지 마세요.

연결 후 확인 [🔗](#)

앞단 송신기에서 실제 메트릭, 트레이스 또는 로그를 보낸 뒤 [데이터 수집 확인](#)을 다시 누릅니다.

증상	확인할 내용
모든 신호가 없음	송신 endpoint, HTTP·gRPC 프로토콜, 4317 · 4318 포트와 송신기 재시작 여부를 확인합니다.
일부 신호만 없음	앞단 송신기가 해당 신호를 실제로 생성하고 내보내는지 확인합니다.
Collector가 시작되지 않음	Connect에서 생성한 YAML 전체를 저장했는지와 수신 포트 충돌 여부를 확인합니다.

상태의 의미와 재시도 방법은 [연결 및 수집 확인](#)에서 안내합니다.

- [OpenTelemetry OTLP Receiver](#)
- [OpenTelemetry Collector 설정](#)

애플리케이션 연결

애플리케이션 카탈로그는 소스 코드를 직접 수정하지 않고 OpenTelemetry 자동 계측을 시작 과정에 적용합니다. Connect에서 실제 실행 환경을 고르고, 생성된 설치·환경 변수·실행 블록을 같은 애플리케이션에 순서대로 적용하세요.

카탈로그	지원 버전	주로 확인하는 데이터
Java	Java 8 이상	트레이스, JVM 메트릭, 지원 Logback·Log4j2 로그
Node.js	Node.js 22·24, npm	트레이스, Node·V8 메트릭, 지원 logger 로그
Python	Python 3.10 이상	지원 라이브러리 텔레메트리, 표준 <code>logging</code> 로그

자동 계측 범위는 애플리케이션이 사용하는 라이브러리에 따라 달라집니다. 연결 후 HTTP 요청, 배치 작업, 메시지 처리처럼 실제 활동을 발생시키고 먼저 트레이스와 로그를 확인하세요.

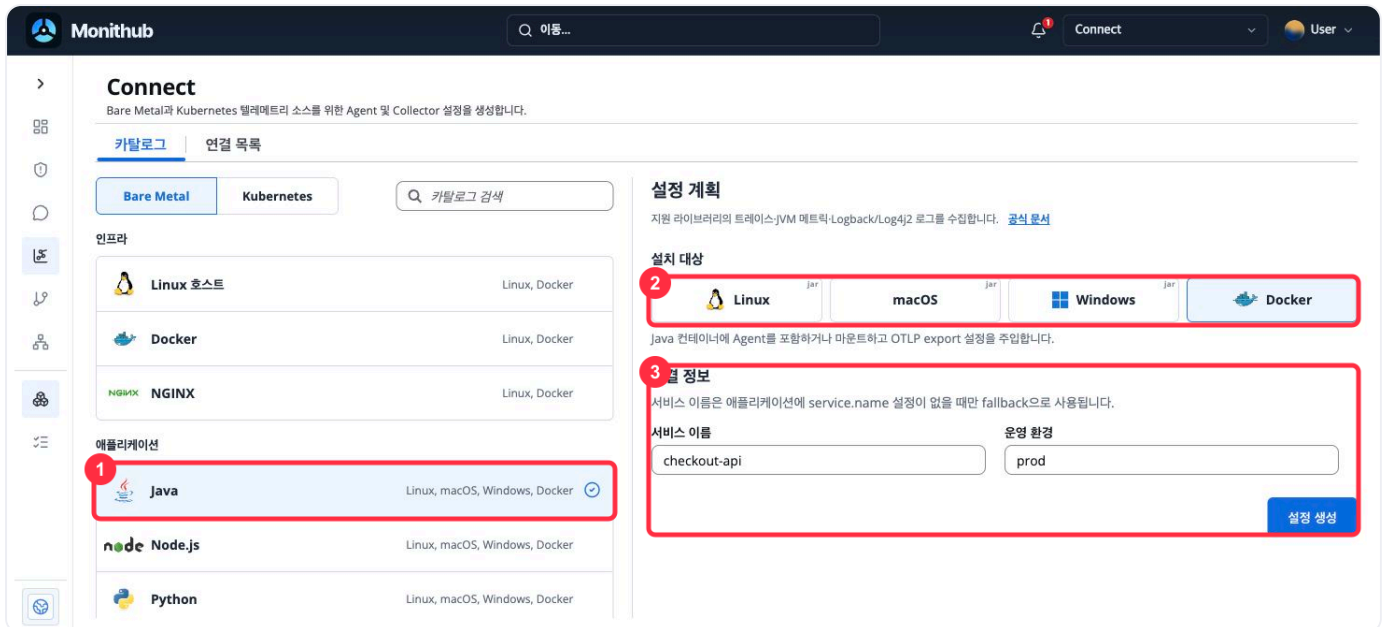
공통 적용 순서

1. 카탈로그와 애플리케이션이 실제 실행되는 환경을 선택합니다.
2. 서비스 이름과 운영 환경을 입력합니다. 애플리케이션에 `service.name`이 이미 있으면 기존 값이 사용됩니다.
3. `설정 생성`을 누르고 표시된 설치 명령을 애플리케이션 디렉터리에서 실행합니다.
4. 생성된 환경 변수 블록 전체를 애플리케이션 실행 환경에 설정합니다.
5. 생성된 방식으로 애플리케이션을 다시 시작하고 실제 요청이나 작업을 발생시킵니다.
6. Connect의 `데이터 수집 확인`에서 신호별 결과를 확인합니다.

문서의 축약 예시를 그대로 복사하지 말고 현재 연결에 표시된 명령과 환경 변수 전체를 사용하세요. 여기에는 실제 조직과 연결에 필요한 값이 포함됩니다.

Java

Java는 OpenTelemetry Java Agent JAR을 JVM 시작 시 연결합니다. 애플리케이션 코드나 빌드 의존성을 바꾸지 않고 지원되는 라이브러리의 요청, 외부 호출, 데이터베이스 호출 등을 계측합니다.



1. **Java**를 선택합니다.
2. JAR을 내려받아 실행할 환경을 선택합니다. Docker는 기존 이미지와 실행 방식을 유지한 채 Agent를 포함하거나 마운트하는 경우에 선택합니다.
3. 서비스 이름과 운영 환경을 입력하고 **설정 생성**을 누릅니다.

생성된 다운로드와 체크섬 확인 명령을 먼저 실행합니다. 이어서 같은 실행 환경에 환경 변수 블록을 적용하고 기존 JAR 이름과 인자를 보존한 채 Agent 옵션을 추가합니다.

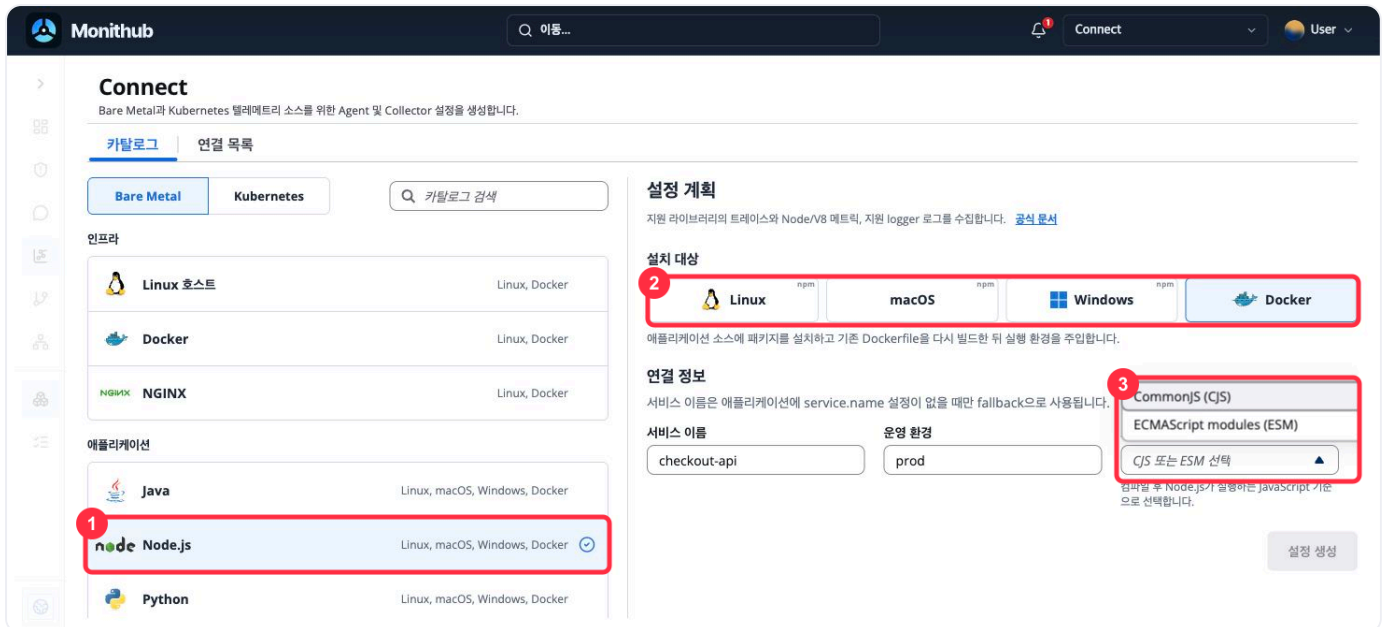
```
# Connect가 생성한 환경 변수 블록을 먼저 적용합니다.
java -javaagent:./opentelemetry-javaagent.jar -jar <your-app>.jar
```

서비스 관리자나 시작 스크립트가 Java를 실행한다면 그 명령에 **-javaagent** 옵션을 추가한 뒤 서비스를 다시 시작하세요. Docker에서는 Connect가 표시한 마운트 경로와 환경 변수를 기존 이미지 실행 명령에 반영합니다.

- [OpenTelemetry Java Agent](#)
- [Java Agent 지원 라이브러리](#)

Node.js [🔗](#)

Node.js는 애플리케이션 소스에 OpenTelemetry 패키지를 npm 의존성으로 설치하고, **NODE_OPTIONS**로 자동 계측을 애플리케이션 코드보다 먼저 불러옵니다.



1. Node.js를 선택합니다.
2. npm 패키지를 설치할 애플리케이션 환경을 선택합니다.
3. Node.js가 실제로 실행하는 빌드 결과를 기준으로 모듈 방식을 선택합니다.
 - CommonJS (CJS): 실행 파일이 `require()`를 사용하거나 CommonJS로 빌드된 경우
 - ECMAScript modules (ESM): 실행 파일이 `.mjs`이거나 실행 시 사용하는 `package.json`에 `"type": "module"`이 있는 경우

TypeScript 소스의 `import` 문법만 보고 고르지 마세요. 컴파일 결과가 CommonJS라면 CJS를 선택합니다.

Connect가 생성한 고정 버전 npm 설치 명령을 애플리케이션 소스 디렉터리에서 실행하고 변경된 `package.json`과 `package-lock.json`을 기존 배포 방식에 반영합니다. 환경 변수와 `NODE_OPTIONS`를 설정한 같은 셸 또는 배포 환경에서 기존 시작 명령을 실행하세요.

```
npm install --save-exact <Connect가 표시한 OpenTelemetry 패키지>
export NODE_OPTIONS="<Connect가 표시한 CJS 또는 ESM 값>"
npm start
```

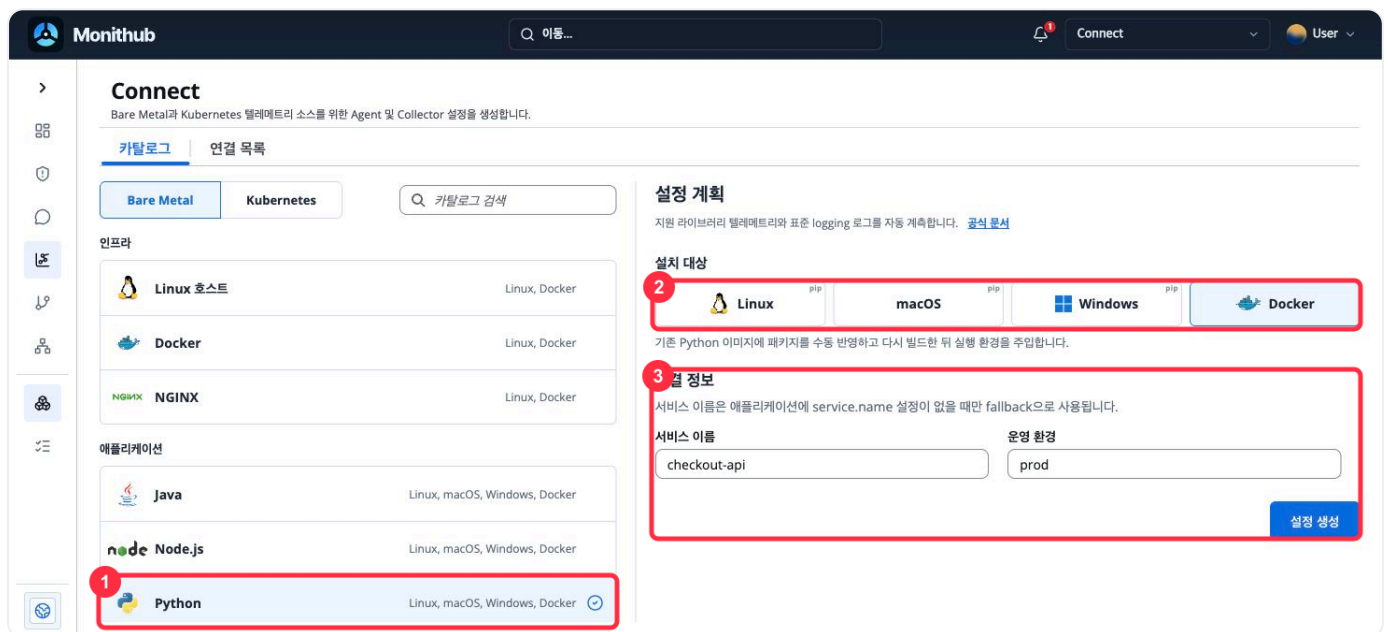
`NODE_OPTIONS`는 Node.js가 시작될 때 공통 옵션을 전달하는 환경 변수입니다. Connect가 생성한 값이 자동 계측을 먼저 불러오므로 애플리케이션과 logger가 계측 가능한 상태로 로드됩니다. 기존 시작 명령이 `npm start`가 아니라면 마지막 줄만 실제 명령으로 바꾸세요.

로그는 애플리케이션이 사용하는 공식 지원 logger에서 수집됩니다. logger를 사용하지 않는 애플리케이션에 Winston, Pino, Bunyan 자체를 새로 설치할 필요는 없습니다.

- [OpenTelemetry JavaScript 제로 코드 계측](#)
- [Node.js 자동 계측 패키지](#)
- [CJS와 ESM 지원 방식](#)

Python

Python은 애플리케이션이 사용하는 Python 환경에 자동 계측 패키지를 설치합니다. 기존 가상환경이 있다면 먼저 활성화하고, 없다면 애플리케이션을 실행하는 현재 Python 환경에서 진행하세요.



1. [Python](#)을 선택합니다.
2. 패키지를 설치하고 애플리케이션을 실행할 동일한 환경을 선택합니다.
3. 서비스 이름과 운영 환경을 입력하고 [설정 생성](#)을 누릅니다.

Connect가 생성한 pip 설치 명령을 실행한 뒤 [opentelemetry-bootstrap](#)으로 현재 환경에 이미 설치된 지원 라이브러리의 계측 패키지를 설치합니다. 환경 변수 블록을 적용하고 기존 실행 명령 앞에 [opentelemetry-instrument](#)를 붙입니다.

```
python -m pip install <Connect가 표시한 OpenTelemetry 패키지>
opentelemetry-bootstrap -a install
opentelemetry-instrument <your-existing-command>
```

`python`, `pip`, `opentelemetry-bootstrap`, 실제 애플리케이션이 서로 다른 가상환경을 가리키지 않는지 확인하세요. Docker는 기존 Dockerfile의 애플리케이션 의존성 설치 뒤에 Connect가 생성한 블록을 추가하고 이미지를 다시 빌드합니다.

- [OpenTelemetry Python 제로 코드 계측](#)
- [OpenTelemetry Python 지원 패키지 목록](#)

연결 후 확인

애플리케이션을 다시 시작한 뒤 실제 요청이나 작업을 여러 번 발생시키고 [데이터 수집 확인](#)을 누릅니다.

- 트레이스가 없으면 새 프로세스의 실행 명령에 Agent 또는 자동 계측 옵션이 포함됐는지 확인합니다.
- Node.js 로그가 없으면 사용하는 logger가 공식 계측 대상인지, 자동 계측이 logger보다 먼저 로드됐는지 확인합니다.
- Python 계측 명령을 찾지 못하면 패키지를 설치한 Python 환경과 실행 환경이 같은지 확인합니다.
- Docker 변경 후에는 이미지를 다시 빌드하고 Connect가 생성한 환경 변수를 새 컨테이너에 전달합니다.
- Node.js와 Python의 메트릭은 사용하는 라이브러리와 실제 활동에 따라 보이지 않을 수 있습니다. 먼저 트레이스와 지원 로그로 연결을 확인합니다.

공통 상태 확인과 재시도 방법은 [연결 및 수집 확인](#)을 참고하세요.

인프라 연결

인프라 카탈로그는 서버, 컨테이너 런타임과 NGINX의 상태를 Collector로 수집합니다. [설치 대상](#)은 수집 대상이 아니라 **Collector를 실제로 실행할 방식**입니다.

카탈로그	수집 데이터	Collector 실행 위치
Linux 호스트	CPU·메모리·디스크·파일시스템·네트워크 메트릭, 시스템 로그	수집할 Linux 서버
Docker	컨테이너 런타임 메트릭	Docker Engine이 실행되는 Linux 호스트
NGINX	stub_status 연결·요청 메트릭	NGINX 상태 주소에 접근할 수 있는 Linux 호스트 또는 컨테이너

Linux 방식은 대상 Linux 호스트에 Collector 바이너리를 설치합니다. Docker 방식은 대상과 가까운 곳에서 전용 Collector 컨테이너를 실행합니다. 생성된 설정 파일과 실행 명령은 선택한 방식을 섞지 말고 같은 열의 안내를 따르세요.

Linux 호스트 [🔗](#)

Linux 호스트는 Collector가 실행되는 서버 자체의 호스트 메트릭과 표준 시스템 로그를 수집합니다. 다른 서버를 대신 수집하는 방식이 아니므로 대상 Linux 서버에서 실행해야 합니다.

Linux 바이너리로 실행 [🔗](#)

1. [Linux 호스트](#)와 [Linux](#)를 선택합니다.
2. 설정 생성 후 대상 서버 CPU에 맞는 x86_64 또는 ARM64 설치 명령을 실행합니다.
3. 생성된 YAML을 [otelcol-monithub.yaml](#)로 저장하고 Collector를 실행합니다.

```
uname -m
otelcol-contrib --config ./otelcol-monithub.yaml
```

Docker로 실행 [🔗](#)

1. [Linux 호스트](#)와 [Docker](#)를 선택합니다.

2. 생성된 YAML을 대상 Linux 서버에 저장합니다.

3. Connect가 생성한 명령으로 호스트 파일시스템을 `/hostfs`에 읽기 전용으로 마운트해 Collector를 실행합니다.

```
docker run --rm --read-only \
-v /:/hostfs:ro \
-v "$PWD/otelcol-monithub.yaml:/etc/otelcol-contrib/config.yaml:ro" \
<Connect가 표시한 Collector 이미지> --config /etc/otelcol-contrib/config.yaml
```

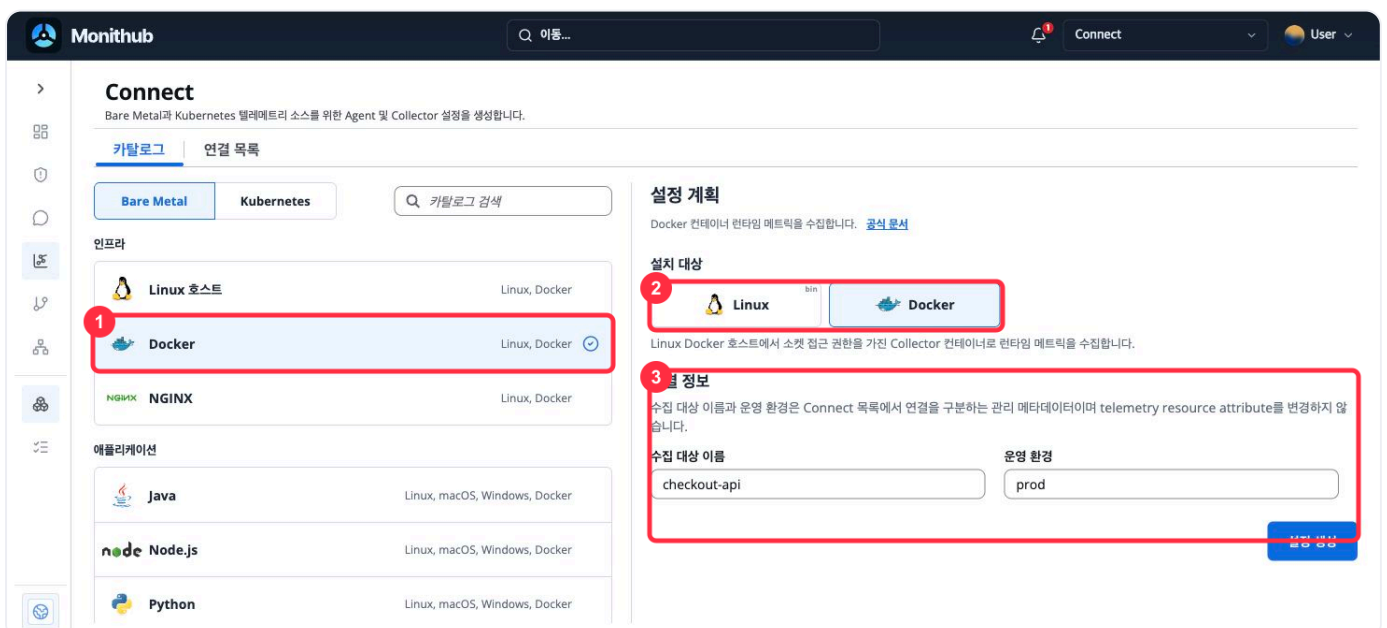
시스템 로그는 Collector가 시작된 이후 새로 기록되는 내용을 읽습니다. 실행했는데 로그가 보이지 않으면 `/var/log/syslog`, `/var/log/messages`, `/var/log/*.log` 중 실제 서버에서 사용하는 파일에 새 로그가 기록되는지와 Collector의 읽기 권한을 확인하세요.

YAML을 복사하고 저장하는 전체 화면 흐름은 [Connect 시작하기의 Linux 호스트 예시](#)를 참고하세요.

- [OpenTelemetry Host Metrics Receiver](#)
- [OpenTelemetry File Log Receiver](#)

Docker

Docker 카탈로그는 Docker Engine에서 실행 중인 컨테이너의 CPU, 메모리, 네트워크, 블록 I/O와 상태 메트릭을 수집합니다. 컨테이너 애플리케이션 로그는 이 카탈로그의 수집 범위가 아닙니다.



The screenshot shows the Monithub Connect web interface. On the left, the '카탈로그' (Catalog) tab is active, displaying a list of infrastructure components under '인프라' (Infrastructure). The 'Docker' component is highlighted with a red box and a red circle with the number '1'. On the right, the '설정 계획' (Setup Plan) section is visible. It shows the '설치 대상' (Installation Target) as 'Linux' and 'Docker'. Below this, the '3. 설정 정보' (3. Configuration Information) section is highlighted with a red box and a red circle with the number '3'. It contains fields for '수집 대상 이름' (Collection Target Name) set to 'checkout-api' and '운영 환경' (Operating Environment) set to 'prod'.

1. [Docker](#)를 선택합니다.
2. Docker Engine이 실행되는 Linux 호스트에서 Collector를 바이너리로 실행할지 컨테이너로 실행할지 선택합니다.
3. 수집 대상 이름과 운영 환경을 입력하고 [설정 생성](#)을 누릅니다.

Linux 바이너리 방식은 Collector 프로세스가 [/var/run/docker.sock](#)에 접근할 수 있어야 합니다. Docker 방식은 Connect가 생성한 명령이 socket을 읽기 전용으로 마운트하고 호스트 socket의 그룹 ID를 Collector에 추가합니다.

```
docker_socket_gid="$(stat -c '%g' /var/run/docker.sock)"
docker run --group-add "$docker_socket_gid" \
-v /var/run/docker.sock:/var/run/docker.sock:ro \
<Connect가 표시한 나머지 실행 옵션>
```

⚠ Docker socket 권한을 넓히지 마세요

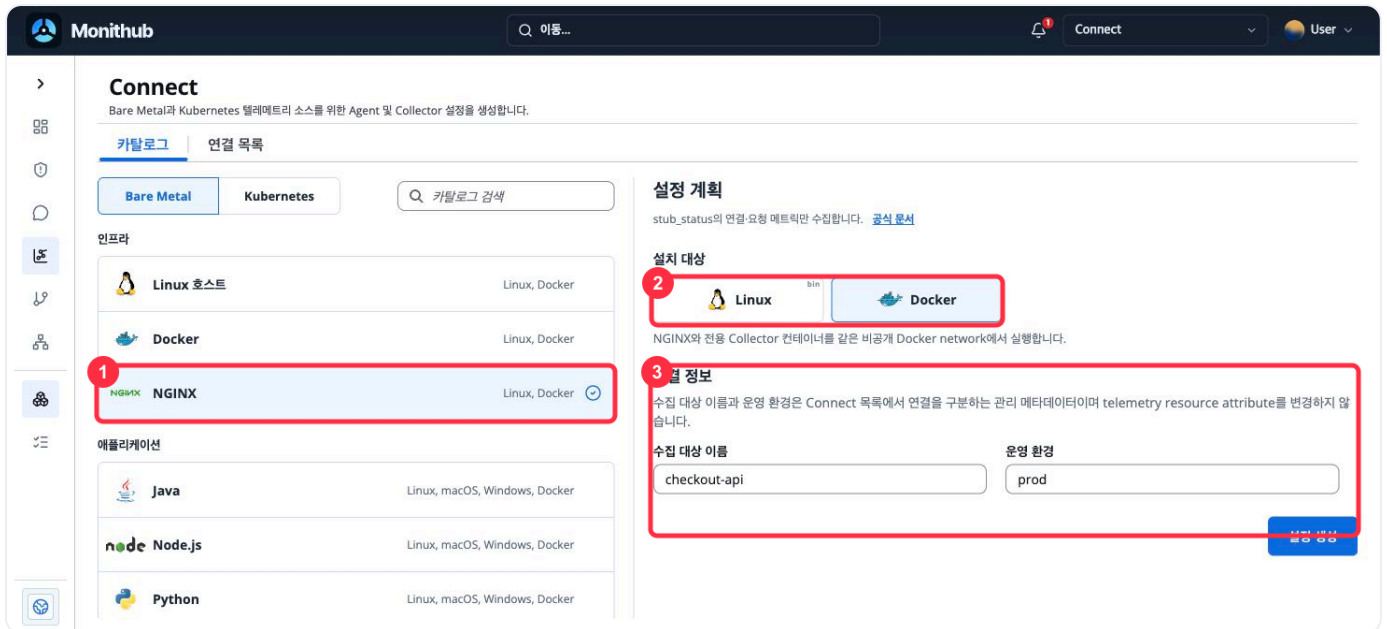
권한 오류를 피하려고 Collector를 [privileged](#) 또는 root로 실행하지 마세요. Connect가 생성한 socket mount와 그룹 추가만 먼저 적용하고, 운영 환경의 별도 보안 정책이 있다면 그 정책을 따릅니다.

Collector 실행 후 테스트 컨테이너를 시작하거나 기존 컨테이너에 부하를 발생시킨 뒤 수집을 확인합니다. 권한 오류가 표시되면 socket의 그룹 ID와 실행 명령의 [--group-add](#) 값이 같은지 확인하세요.

- [OpenTelemetry Docker Stats Receiver](#)

NGINX

NGINX 카탈로그는 NGINX OSS의 [stub_status](#)가 제공하는 연결 수와 누적 요청 수를 수집합니다. access/error 로그, 응답 시간, 상태 코드, upstream 정보는 수집하지 않습니다.



1. **NGINX**를 선택합니다.
2. NGINX 옆에서 Collector를 Linux 바이너리 또는 Docker 컨테이너로 실행할지 선택합니다.
3. 수집 대상 이름과 운영 환경을 입력하고 **설정 생성**을 누릅니다.

Linux에서 적용 [🔗](#)

Connect가 생성한 `monithub-status.conf` 전체를 기존 NGINX 설정의 include 경로에 저장합니다. 짧은 형태는 다음과 같습니다.

```
server {
    listen 127.0.0.1:18080;
    location = /status { stub_status; }
}
```

다음 순서로 모듈, 전체 설정, 상태 응답을 확인한 뒤 기존 서비스 관리 방식으로 NGINX를 reload합니다.

```
nginx -V 2>&1 | grep http_stub_status_module
nginx -t
curl --fail http://127.0.0.1:18080/status
```

Docker에서 적용 [🔗](#)

Connect가 생성한 `location = /status` 조각을 기존 NGINX server 블록에 포함합니다. NGINX와 Collector를 같은 비공개 Docker network에 두고 상태 포트는 호스트에 공개하지 않습니다.

Collector 설정의 기본 주소는 `http://nginx:80/status`입니다. Compose 서비스 이름이 `nginx`가 아니라면 생성된 YAML의 `hostname`을 실제 서비스 이름으로 바꾸고, 그 network 안에서 상태 응답을 확인하세요.

Connect는 NGINX 설정을 직접 수정하거나 reload하지 않습니다. `nginx -t`가 실패하면 새 조각을 적용하지 말고 기존 설정을 유지하세요.

- [OpenTelemetry NGINX Receiver](#)
- [NGINX stub_status 모듈](#)

연결 후 확인

증상	확인할 내용
Collector가 시작되지 않음	선택한 OS·CPU와 설치 파일이 맞는지, YAML 전체를 저장했는지 확인합니다.
Linux 호스트 로그만 없음	새 로그가 기록되는 파일 경로와 Collector의 파일 읽기 권한을 확인합니다.
Docker 메트릭이 없음	Docker socket mount와 socket 그룹 권한을 확인합니다.
NGINX 메트릭이 없음	<code>stub_status</code> 모듈, <code>nginx -t</code> , 상태 URL 응답과 Docker network hostname을 확인합니다.

대상에서 실제 활동을 발생시킨 뒤 [데이터 수집 확인](#)을 다시 누르세요. 상태의 의미와 재시도 방법은 [연결 및 수집 확인](#)에서 안내합니다.

데이터베이스 연결

데이터베이스 카탈로그는 데이터베이스에 접근할 수 있는 환경에서 전용 Collector를 실행해 운영 메트릭을 읽습니다. 데이터베이스 로그, 쿼리 본문과 결과는 수집하지 않습니다.

카탈로그	수집 데이터	필요한 조회 권한
PostgreSQL	서버와 데이터베이스 상태 메트릭	<code>pg_stat_database</code> 의 <code>SELECT</code>
MySQL / MariaDB	전역 상태와 InnoDB 메트릭	<code>SHOW GLOBAL STATUS</code> 실행
Redis	단일 endpoint의 기본 <code>INFO</code> 메트릭	<code>INFO</code> 실행

가능하면 애플리케이션 계정과 분리된 모니터링 계정을 사용하세요. Connect에는 endpoint, 사용자 이름, 비밀번호, 데이터베이스 이름과 인증서 경로를 입력하지 않습니다. 이 값은 Collector를 실행할 때만 환경 변수나 운영 중인 secret store로 전달합니다.

공통 연결 순서

1. Collector에서 접근 가능한 데이터베이스 endpoint와 모니터링 계정을 준비합니다.
2. 카탈로그와 Collector를 실제로 실행할 환경을 선택합니다.
3. 데이터베이스 전송 보안을 선택하고 수집 대상 이름과 운영 환경을 입력합니다.
4. `설정 생성`을 누르고 표시된 Collector 설정 전체를 `otelcol-monithub.yaml`로 저장합니다.
5. 해당 카탈로그의 런타임 값을 환경 변수로 주입하고 Connect가 표시한 명령으로 Collector를 실행합니다.
6. 한 번 이상 수집될 때까지 기다린 뒤 `데이터 수집 확인`에서 메트릭을 확인합니다.

문서의 축약 예시는 변수 이름과 적용 위치를 설명하기 위한 것입니다. Collector YAML과 설치·실행 명령은 현재 연결에서 생성된 내용을 사용하세요.

전송 보안 선택

방식	준비할 항목	권장 상황
평문 TCP	endpoint와 계정 정보	암호화가 필요 없는 신뢰된 사설망
서버 인증서 검증 TLS	CA 인증서	암호화하고 서버 인증서를 검증할 때
클라이언트 인증서 mTLS	CA, 클라이언트 인증서와 개인 키	서버와 Collector가 서로 인증할 때

평문 TCP는 전송 내용을 암호화하지 않습니다. 인터넷이나 신뢰할 수 없는 네트워크를 통과한다면 TLS 또는 mTLS를 사용하세요. TLS와 mTLS에서는 인증서 검증을 끄지 말고, Collector 프로세스나 컨테이너가 읽을 수 있는 인증서 경로를 지정합니다.

런타임 값 적용 [🔗](#)

Linux와 macOS에서는 값이 들어 있는 파일을 Git에 커밋하지 말고 파일 권한을 제한한 뒤, Collector를 시작하는 셸이나 서비스 관리자에서 불러옵니다.

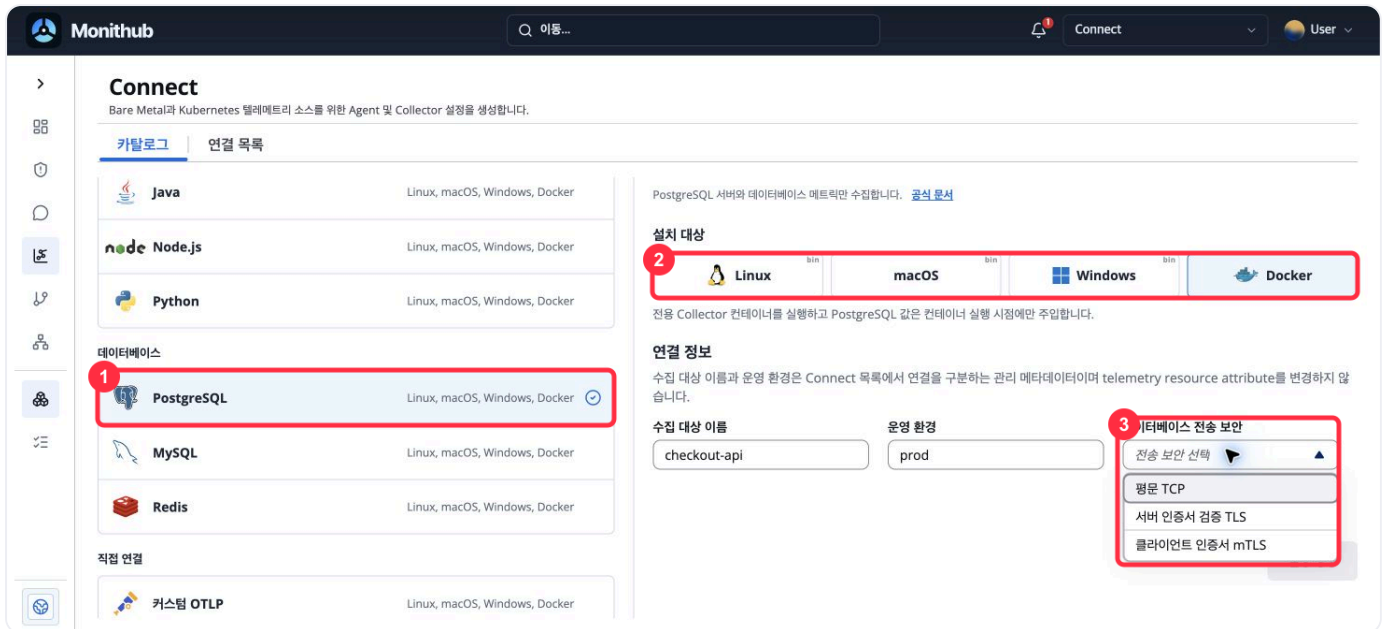
```

chmod 600 ./database-collector.env
set -a
. ./database-collector.env
set +a
otelcol-contrib --config ./otelcol-monithub.yaml

```

Docker를 선택했다면 같은 파일을 Connect가 생성한 `--env-file` 옵션에 사용합니다. YAML과 인증서는 생성된 명령처럼 읽기 전용으로 마운트하세요. Windows에서는 Connect가 표시한 PowerShell 환경 변수 블록과 실행 명령을 같은 세션에서 실행합니다.

PostgreSQL [🔗](#)



1. PostgreSQL을 선택합니다.
2. 데이터베이스에 접근할 수 있는 환경에서 Collector를 바이너리로 실행할지 Docker로 실행할지 선택합니다.
3. 실제 데이터베이스 연결 방식과 일치하는 전송 보안을 선택합니다.

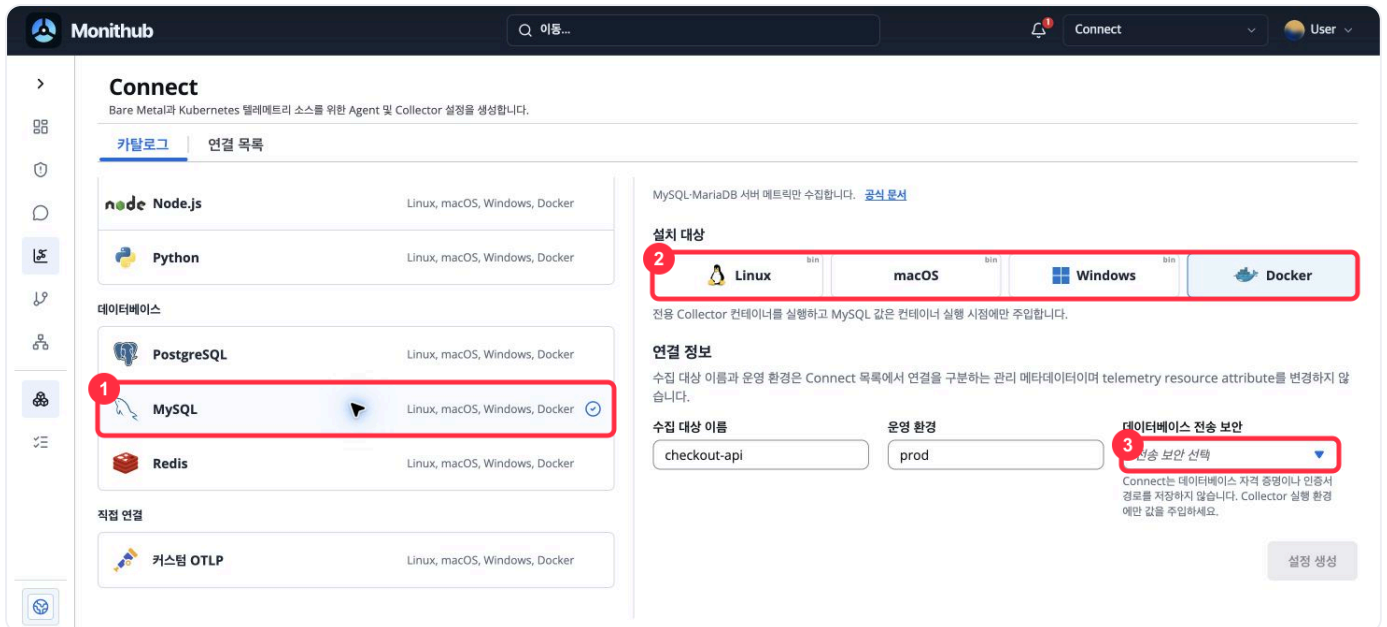
Collector 실행 환경에 다음 값을 주입합니다. endpoint는 `host:port` 형식이며, 데이터베이스는 이 연결에서 수집할 하나를 지정합니다.

```
POSTGRESQL_ENDPOINT=db.internal:5432
POSTGRESQL_USERNAME=<monitoring-user>
POSTGRESQL_PASSWORD=<secret>
POSTGRESQL_DATABASE=<database>
```

모니터링 계정에는 `pg_stat_database`를 읽을 수 있는 `SELECT` 권한이 필요합니다. Connect가 생성하는 설정은 서버와 데이터베이스 메트릭만 수집하며 query sample, top query와 데이터베이스 로그는 활성화하지 않습니다.

- [OpenTelemetry PostgreSQL Receiver](#)
- [PostgreSQL 통계 모니터링](#)

MySQL / MariaDB [🔗](#)



1. [MySQL](#)을 선택합니다. MariaDB도 같은 카탈로그를 사용합니다.
2. 데이터베이스에 접근할 수 있는 Collector 실행 환경을 선택합니다.
3. 실제 서버 설정에 맞는 전송 보안을 선택합니다.

```

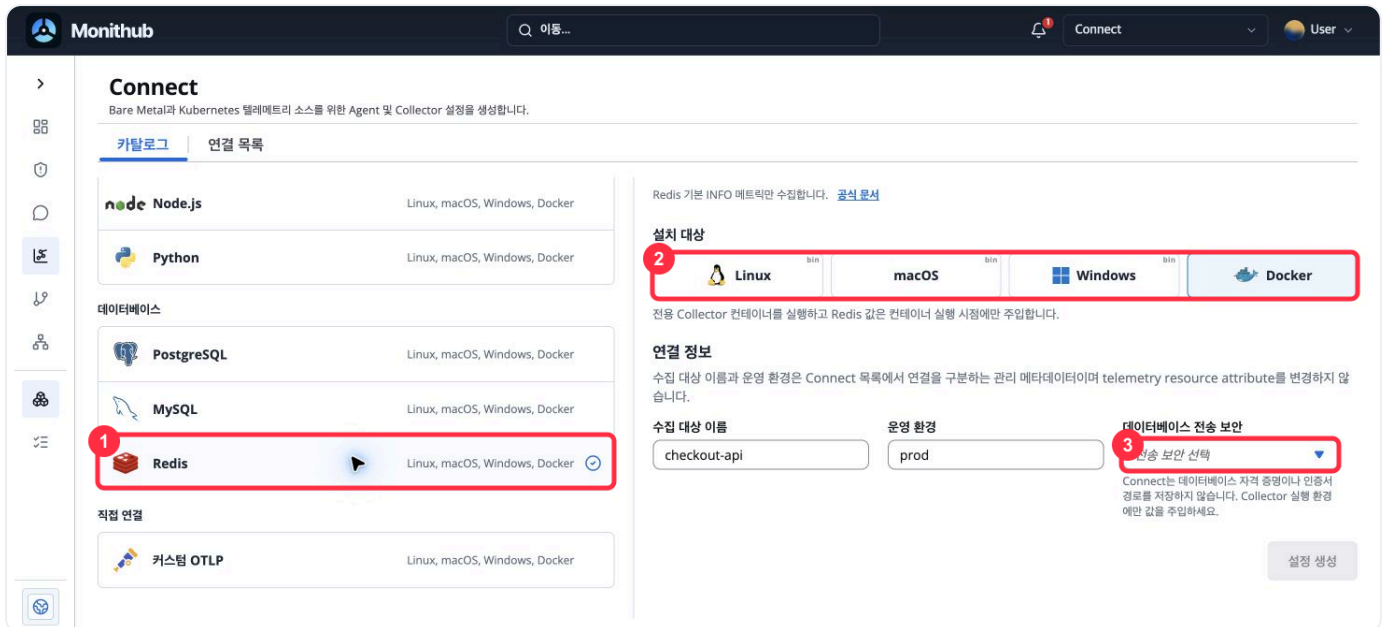
MYSQL_ENDPOINT=db.internal:3306
MYSQL_USERNAME=<monitoring-user>
MYSQL_PASSWORD=<secret>
MYSQL_DATABASE=<database>

```

모니터링 계정은 [SHOW GLOBAL STATUS](#)를 실행할 수 있어야 합니다. 지원 대상은 MySQL 5.7.x, 8.0.x, 8.4.x, 9.x와 MariaDB 10.5.x~10.11.x, 11.x입니다. Connect 설정은 메트릭만 수집하며 query sample, top query와 statement event는 활성화하지 않습니다.

- [OpenTelemetry MySQL Receiver](#)
- [MySQL 서버 상태 변수](#)

Redis [🔗](#)



1. Redis를 선택합니다.
2. Redis endpoint에 접근할 수 있는 Collector 실행 환경을 선택합니다.
3. 실제 Redis 연결 방식에 맞는 전송 보안을 선택합니다.

```
REDIS_ENDPOINT=redis.internal:6379
REDIS_USERNAME=<acl-user>
REDIS_PASSWORD=<secret>
```

사용자 이름은 Redis 6 이상에서 ACL을 사용할 때 필요합니다. 모니터링 계정에는 INFO 실행 권한만 부여하고 애플리케이션 데이터 명령 권한은 주지 않는 것을 권장합니다. 수집되는 값은 서버 상태와 keyspace 요약 메트릭이며 key와 value는 읽지 않습니다.

Redis 연결 하나는 정적 endpoint 하나를 사용합니다. failover 등으로 endpoint가 변경되면 REDIS_ENDPOINT를 바꾸고 Collector를 다시 시작하세요. Sentinel이나 Cluster의 node를 자동으로 찾거나 endpoint를 자동 갱신하지 않습니다.

- [OpenTelemetry Redis Receiver](#)
- [Redis INFO 명령](#)

TLS 인증서 변수 [🔗](#)

전송 보안에서 TLS 또는 mTLS를 선택했다면 같은 카탈로그의 기본 환경 변수에 아래 경로를 추가합니다.

카탈로그	TLS	mTLS에서 추가
PostgreSQL	<code>POSTGRESQL_TLS_CA_FILE</code>	<code>POSTGRESQL_TLS_CERT_FILE</code> , <code>POSTGRESQL_TLS_KEY_FILE</code>
MySQL / MariaDB	<code>MYSQL_TLS_CA_FILE</code>	<code>MYSQL_TLS_CERT_FILE</code> , <code>MYSQL_TLS_KEY_FILE</code>
Redis	<code>REDIS_TLS_CA_FILE</code>	<code>REDIS_TLS_CERT_FILE</code> , <code>REDIS_TLS_KEY_FILE</code>

Docker에서는 호스트 경로가 아니라 컨테이너 안에서 보이는 인증서 경로를 환경 변수에 넣습니다. Connect가 생성한 실행 명령의 읽기 전용 인증서 mount와 경로가 일치하는지 확인하세요.

여러 대상과 관리형 데이터베이스

Connect 연결 하나는 PostgreSQL·MySQL은 endpoint 하나와 데이터베이스 하나, Redis는 정적 endpoint 하나를 기준으로 생성됩니다. 여러 대상을 한 Collector에서 수집하려면 먼저 각각 연결을 확인한 뒤 기존 Collector 설정을 직접 확장할 수 있습니다.

클라우드 관리형 데이터베이스도 공급자가 발급한 endpoint, 모니터링 계정과 CA 인증서를 같은 런타임 변수에 넣습니다. IAM 토큰 발급이나 공급자별 인증 설정은 해당 클라우드의 공식 절차를 따르세요.

연결 후 확인

증상	확인할 내용
연결 시간 초과 또는 거부	Collector 실행 위치에서 endpoint와 port에 접근 가능한지, 방화벽과 사설망 경로가 열려 있는지 확인합니다.
인증 실패	사용자 이름과 비밀번호, PostgreSQL <code>SELECT</code> , MySQL <code>SHOW GLOBAL STATUS</code> , Redis <code>INFO</code> 권한을 확인합니다.
인증서 검증 실패	CA가 서버 인증서를 발급한 CA인지, 인증서의 서버 이름이 endpoint와 일치하는지 확인합니다.
Docker에서 인증서를 찾지 못함	인증서의 읽기 전용 mount와 환경 변수에 적은 컨테이너 내부 경로가 같은지 확인합니다.
Collector는 실행되지만 메트릭이 없음	선택한 데이터베이스 이름과 endpoint를 확인하고 한 번 이상의 수집 주기가 지난 뒤 다시 확인합니다.

Collector 로그에 연결 오류가 없고 메트릭이 수집되면 데이터베이스 연결이 완료된 것입니다. 공통 상태와 재시도 방법은 [연결 및 수집 확인](#)을 참고하세요.

Kubernetes 연결

Kubernetes 연결은 클러스터의 각 노드에 OpenTelemetry Collector를 배치해 노드, Pod, 컨테이너 메트릭을 Monithub로 보냅니다. Connect에서 생성하는 기본 설치 방식은 `otel-host` DaemonSet입니다.

시작하기 전에

다음 조건을 확인합니다.

- 설치할 클러스터를 가리키는 `kubectl` context가 준비되어 있어야 합니다.
- 클러스터에서 `https://otel.monithub.org`로 HTTPS 통신할 수 있어야 합니다.
- Namespace, Secret, ServiceAccount, ClusterRole, ClusterRoleBinding, ConfigMap, DaemonSet을 만들 수 있는 권한이 필요합니다.
- 조직 토큰이 포함된 manifest를 Git 저장소나 공개 위치에 저장하지 않습니다.

권한이 없다면 생성된 manifest를 클러스터 관리자에게 안전한 경로로 전달하고, 필요한 RBAC 범위를 함께 검토합니다.

적용할 터미널에서 현재 context가 대상 클러스터인지 먼저 확인합니다.

```
kubectl config current-context
kubectl get nodes
```

1. Kubernetes 설정 계획 만들기

1. `Connect` → `카탈로그`를 엽니다.
2. 설치 위치에서 `Kubernetes`를 선택합니다.
3. 카탈로그에서 `Kubernetes`를 선택합니다.
4. 설정 계획의 `Kubernetes` 카드를 선택합니다.
5. `서비스 또는 소스 이름`에 클러스터를 구분할 이름을 입력합니다. 예: `prod-cluster-a`.
6. `환경 이름`에 `prod`, `staging`과 같은 환경을 입력합니다.
7. `연결 준비`를 누릅니다.

같은 클러스터를 여러 이름으로 중복 연결하지 않도록 기존 `연결 목록`을 먼저 확인합니다.

2. 생성된 manifest 검토하기

설정 계획은 `monithub-kubernetes-otel-host.yaml` manifest를 생성합니다. 적용 전에 다음 리소스를 검토합니다.

리소스	역할
Namespace	Monithub 수집 리소스를 분리합니다.
Secret	조직 토큰과 조직·사용자·Connect 식별자를 보관합니다.
ServiceAccount와 RBAC	노드, Pod, kubelet 통계를 읽을 권한을 부여합니다.
ConfigMap	Collector receiver, processor, exporter 설정을 보관합니다.
DaemonSet	각 노드에서 <code>otel-host</code> Collector를 실행합니다.

기본 RBAC는 수집에 필요한 조회 권한을 대상으로 합니다. 조직 보안 정책에 따라 ClusterRole과 host 접근을 검토한 뒤 적용합니다.

3. 클러스터에 적용하기

설정 계획의 manifest를 파일로 저장한 뒤, 현재 context가 대상 클러스터인지 다시 확인하고 화면에 표시된 명령을 실행합니다.

```
kubectl config current-context
kubectl get nodes
kubectl apply -f ./monithub-kubernetes-otel-host.yaml
kubectl -n monithub rollout status daemonset/monithub-otel-host
kubectl -n monithub logs daemonset/monithub-otel-host --tail=50
```

추가로 Pod 배치 상태를 확인하려면 다음 명령을 사용합니다.

```
kubectl -n monithub get pods -o wide
```

모든 대상 노드의 Pod가 **Running**이고 DaemonSet rollout이 완료되어야 합니다. 일부 노드에 Pod가 없다면 node selector, taint, 리소스 부족, 이미지 pull, 네트워크 정책을 확인합니다.

4. 수집 범위 확인하기

기본 DaemonSet은 다음 데이터를 자동 수집합니다.

- kubelet 통계 기반의 노드, Pod, 컨테이너 메트릭
- 각 노드의 CPU, memory, disk, filesystem, network, process 메트릭

manifest에는 OTLP receiver도 포함되어 있지만, 이 기본 설치만으로 워크로드용 고정 Service나 애플리케이션 전송 경로가 만들어지지 않습니다. 따라서 애플리케이션 트레이스·로그를 자동 수집한다고 가정하면 안 됩니다. 워크로드 계측과 Collector gateway를 함께 관리하려면 [Monithub K8S Operator 설치](#) 절차를 사용합니다.

5. Connect에서 연결 확인하기

1. DaemonSet rollout과 로그를 확인한 뒤 Connect 화면으로 돌아갑니다.
2. **연결 시작**을 누릅니다.
3. 클러스터에서 실제 워크로드 트래픽을 발생시킵니다.
4. **데이터 수집 확인**에서 신호별 결과를 확인합니다.
5. **연결 목록**에서 연결 상태, 데이터 수집 상태, 마지막 수집 시각을 확인합니다.

첫 수집 결과가 나타나기까지 Collector 전송 주기와 워크로드 트래픽에 따라 시간이 걸릴 수 있습니다. **연결됨**인데 데이터가 없다면 Collector Pod 로그에서 인증 오류와 exporter 오류를 먼저 확인합니다.

Operator 설치와의 차이

이 문서는 Connect 화면이 생성하는 기본 **otel-host** DaemonSet 설치 절차입니다. Monithub K8S Operator를 사용해 수집 구성과 Flow Agent를 관리하려면 [Monithub K8S Operator 설치](#) 가이드를 따릅니다.

두 방식을 같은 클러스터에 중복 적용하기 전에 수집 범위와 Connect ID가 겹치지 않는지 확인합니다.

연결을 삭제할 때

Connect의 연결 삭제는 Monithub에 저장된 연결과 설정 계획만 삭제합니다. 고객 클러스터에 이미 적용한 Kubernetes 리소스는 자동으로 제거되지 않습니다.

Caution

생성 manifest에는 `monithub` Namespace와 `monithub-otel-host`라는 고정 이름의 ClusterRole·ClusterRoleBinding이 포함됩니다. 다른 Monithub 수집 구성이나 리소스가 같은 Namespace를 사용 중이면 `kubectl delete -f`를 실행하지 마세요. Namespace 삭제로 해당 Namespace의 다른 리소스도 함께 삭제될 수 있습니다.

먼저 이 Connect가 만든 리소스만 있는 전용 Namespace인지 확인합니다. Secret 값은 출력하지 않습니다.

● ● ●

```
kubectl -n monithub get daemonset,configmap,serviceaccount,secret  
kubectl get clusterrole,clusterrolebinding monithub-otel-host
```

출력된 리소스가 모두 이 Connect를 위해 만든 것이고 `monithub` Namespace가 전용인 경우에만 다음 명령으로 제거합니다.

● ● ●

```
kubectl delete -f ./monithub-kubernetes-otel-host.yaml
```

기존 설정에 적용

기존 설정 수정은 새 Collector나 Java Agent를 추가하지 않고 현재 실행 중인 구성에 Connect 설정만 병합하는 방식입니다. Monithub는 사용자 환경을 자동으로 수정하지 않으므로 백업, 병합, 검증과 재시작은 기존 배포 절차로 진행합니다.

지원 범위 [🔗](#)

카탈로그	기존 설정에 추가하는 내용	먼저 확인할 것
Custom OTLP	Connect 전용 receiver와 메트릭·로그·트레이스 pipeline	전용 OTLP 포트와 기존 포트가 충돌하지 않는지 확인
Linux 호스트	hostmetrics·filelog receiver와 pipeline	Collector가 같은 호스트에서 로그 파일을 읽을 수 있는지 확인
Docker	docker_stats receiver와 메트릭 pipeline	Collector가 Docker socket에 접근할 수 있는지 확인
Java 애플리케이션	기존 Java Agent의 OTLP 환경 변수	기존 Agent와 애플리케이션 시작 명령을 유지할 수 있는지 확인

Node.js, Python, NGINX, 데이터베이스와 Kubernetes 카탈로그는 이 방식을 제공하지 않습니다. 해당 카탈로그 문서의 새 설정 적용 절차를 사용하세요.

1. 현재 구성 백업 [🔗](#)

Collector 설정 파일, 서비스 정의, Compose 파일 또는 애플리케이션 배포 설정을 먼저 백업합니다. 실행 중인 Collector가 **otelcol-contrib**인지도 확인하세요.

```
otelcol-contrib --version
cp /etc/otelcol-contrib/config.yaml /etc/otelcol-contrib/config.yaml.backup
```

경로와 백업 방식은 실제 배포 환경에 맞게 변경합니다. 다른 호스트나 Docker Engine의 데이터는 기존 Collector가 자동으로 읽을 수 없으므로 수집 대상과 같은 환경의 구성을 수정해야 합니다.

2. Connect 설정 복사 [🔗](#)

1. 지원 카탈로그와 실행 환경을 선택하고 **설정 생성**을 누릅니다.
2. 설정 결과의 **설정 적용 방식**에서 **기존 설정 수정**을 선택합니다.
3. 표시된 Connect ID와 설정 조각 전체를 복사합니다.
4. 카탈로그별 접근 조건 경고를 확인합니다.

⚠ Caution

생성된 설정에는 인증 정보가 포함됩니다. Git 저장소, 이슈나 메신저에 올리지 말고 기존 secret 관리 방식으로 보관하세요.

3. 기존 구성에 병합 [🔗](#)

Collector YAML은 **receivers**, **processors**, **exporters**, **service**를 파일 아래에 다시 만들지 말고 기존 최상위 키 아래에 생성된 하위 항목을 추가합니다.

```
receivers:
  existing_receiver: { ... }
  generated_receiver: { ... }
service:
  pipelines:
    generated_pipeline: { ... }
```

- 기존 receiver, processor, exporter와 pipeline을 삭제하거나 교체하지 않습니다.
- 생성된 컴포넌트 이름과 Connect ID를 바꾸거나 다른 연결과 공유하지 않습니다.
- Custom OTLP 포트를 변경한다면 송신 애플리케이션의 endpoint도 같은 값으로 변경합니다.
- 컨테이너 Collector는 화면에서 안내한 host root 또는 Docker socket mount를 기존 배포 정의에 추가합니다.

Java 애플리케이션은 Collector YAML 대신 표시된 OTLP 환경 변수를 기존 서비스, Compose 또는 배포 설정에 병합합니다. 기존 **service.name**과 다른 resource attribute는 유지하세요.

4. 검증하고 반영 [🔗](#)

실제 운영에서 사용하는 Collector로 병합한 전체 파일을 검증합니다.

```
otelcol-contrib validate --config /etc/otelcol-contrib/config.yaml
```

검증이 성공하면 현재 관리 방식으로 Collector 또는 Java 애플리케이션을 재시작하거나 다시 배포합니다.

실행 방식	반영 후 확인
systemd	서비스 상태와 journal 로그
Docker Compose	다시 생성된 컨테이너 상태와 Compose 로그
직접 실행	Collector 표준 출력의 설정·export 오류
Java 서비스	Java Agent 시작과 OTLP export 오류

검증이나 재시작에 실패하면 백업한 설정으로 복구한 뒤 오류를 확인하세요.

5. 데이터 수집 확인 [🔗](#)

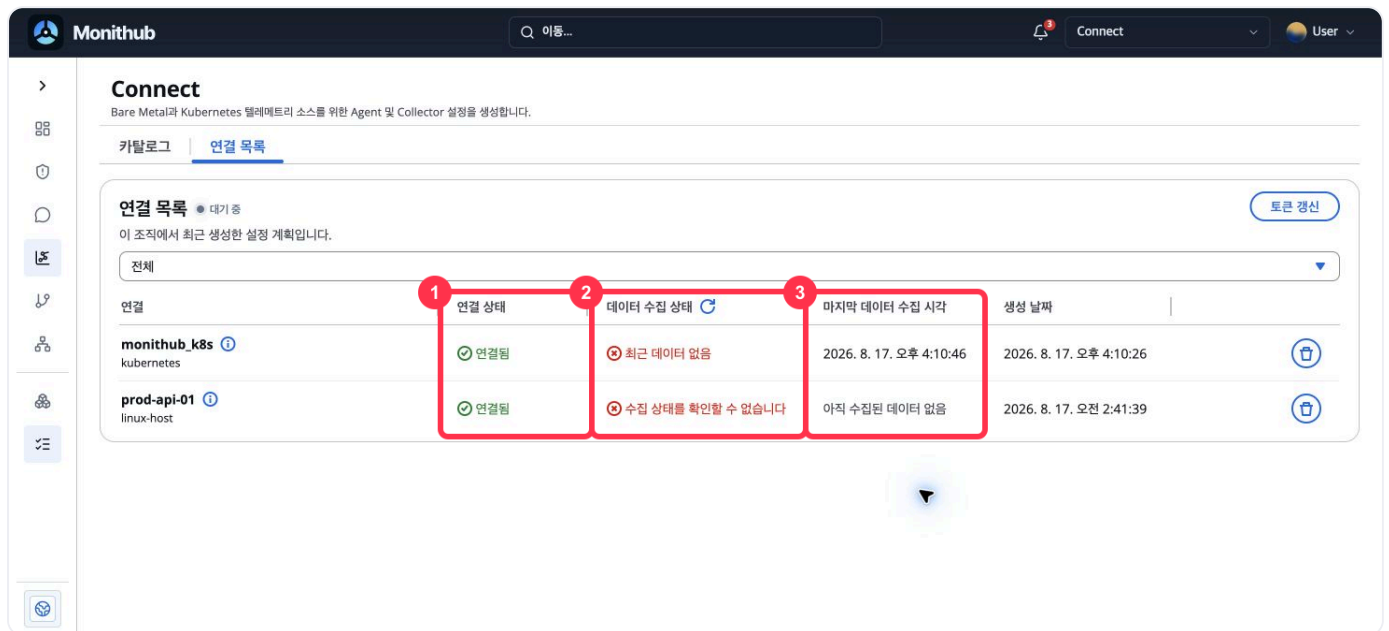
1. Collector 또는 애플리케이션 로그에 설정, 권한, 포트 충돌과 인증 오류가 없는지 확인합니다.
2. Custom OTLP와 Java는 실제 요청을 발생시키고, Linux와 Docker는 한 번 이상의 수집 주기를 기다립니다.
3. Connect의 [데이터 수집 확인](#)에서 예상 신호가 들어오는지 확인합니다.
4. [연결 목록](#)에서 마지막 수집 시각을 확인합니다.

증상	확인할 내용
mapping key ... already defined	최상위 YAML 키를 중복으로 추가하지 않았는지 확인합니다.
unknown type	실제 실행 파일이 otelcol-contrib 이고 생성된 receiver를 지원하는 버전인지 확인합니다.
address already in use	Custom OTLP receiver와 애플리케이션 endpoint의 포트를 함께 변경합니다.
데이터가 없음	파일·host root·Docker socket 권한과 pipeline의 Connect ID를 확인합니다.
401 또는 403	Connect에서 최신 설정을 다시 생성해 인증 정보를 교체합니다.

공통 상태와 재확인 순서는 [연결 및 수집 확인](#)을 참고하세요.

연결 및 수집 확인

[Connect](#) → [연결 목록](#)에서는 현재 조직에서 만든 연결을 한곳에서 관리합니다. 연결 이름, 카탈로그, 연결 상태, 데이터 수집 상태, 마지막 수집 시각, 생성 시각을 확인할 수 있습니다.



1. [연결 상태](#)에서 설정 생성 후 연결 구성이 완료됐는지 확인합니다.
2. [데이터 수집 상태](#)에서 최근 텔레메트리 유입 여부를 확인합니다.
3. [마지막 데이터 수집 시각](#)에서 마지막으로 데이터가 들어온 시간을 확인합니다.

연결 상태와 수집 상태 구분하기 [🔗](#)

두 상태는 서로 다른 내용을 나타냅니다.

항목	의미
연결 상태	설정 생성 후 연결 구성이 완료됐는지 나타냅니다.
데이터 수집 상태	Monithub에 최근 텔레메트리가 들어오고 있는지 나타냅니다.

따라서 [연결됨](#)이 표시돼도 수집 대상에서 데이터가 발생하지 않거나 Collector·Agent가 중단되면 [수집 지연](#) 또는 [최근 데이터 없음](#)이 표시될 수 있습니다.

연결 상태 확인하기 [🔗](#)

상태	의미	할 일
연결됨	연결 설정이 활성화되었습니다.	데이터 수집 상태와 마지막 수집 시각을 확인합니다.
대기 중·연결 중	연결 구성이 진행 중입니다.	완료될 때까지 기다립니다.
연결 실패	연결 과정이 완료되지 않았습니다.	표시된 안내를 확인한 뒤 재시도 를 누릅니다.
연결 시간 만료	정해진 시간 안에 연결을 완료하지 못했습니다.	수집 환경을 확인하고 재시도 를 누릅니다.
연결 해제	현재 연결이 활성 상태가 아닙니다.	다시 사용할 경우 다시 연결 을 누릅니다.

화면을 다시 열거나 브라우저 탭으로 돌아오면 최신 상태를 다시 불러옵니다. 진행 상태가 오래 바뀌지 않으면 목록을 새로고침한 뒤 Collector·Agent의 실행 상태를 확인합니다.

데이터 수집 상태 확인하기

상태	의미	할 일
수집 정상	최근 텔레메트리가 확인되었습니다.	대시보드, 로그, 트레이스에서 데이터를 확인합니다.
수집 확인 중	첫 데이터 또는 최신 상태를 확인하고 있습니다.	수집 대상에서 실제 작업을 발생시키고 잠시 기다립니다.
수집 지연	마지막 수집 후 3분이 지났습니다.	Collector·Agent, 네트워크, 수집 대상 트래픽을 확인합니다.
최근 데이터 없음	마지막 수집 후 5분이 지났거나 아직 데이터가 없습니다.	실행 설정과 인증 오류를 확인합니다.
수집 상태 확인 불가	현재 상태 조회를 완료하지 못했습니다.	잠시 후 새로고침하고 같은 문제가 반복되는지 확인합니다.

연결 이름 옆의 상태 상세 버튼을 누르면 Connect ID, 상태 확인 시작 시각, 상태 변경 시각, 마지막 수집 시각을 볼 수 있습니다. 표 제목의 새로고침 버튼은 현재 조직의 수집 상태를 다시 조회합니다.

데이터 수집 확인 사용하기

설정 계획 화면의 **데이터 수집 확인**은 최근 메트릭, 로그, 트레이스를 신호별로 확인합니다.

- **수집 확인**: 해당 신호의 최근 데이터가 있습니다.

- **아직 없음**: 조회는 성공했지만 해당 신호 데이터가 아직 없습니다.
- **확인하지 못함**: 현재 해당 신호를 조회할 수 없습니다. 잠시 후 다시 시도합니다.

카탈로그가 모든 신호를 자동으로 만드는 것은 아닙니다. 예를 들어 커스텀 OTLP는 앞단 송신기가 보낸 신호만 확인되고, Java 애플리케이션은 실제 애플리케이션 요청과 지원 logger 사용 여부에 따라 결과가 달라집니다.

다시 선택, 재시도와 삭제

설정 생성 후 카탈로그나 실행 환경을 바꾸려면 화면 위쪽의 **다시 선택**을 누릅니다.

- 수집된 데이터가 있는 연결: **연결 유지하고 새로 추가**, **연결 삭제하고 다시 선택**, **닫기** 중에서 선택합니다.
- 수집된 데이터가 없는 연결: **연결 삭제하고 다시 선택**하거나 모달을 닫습니다.
- 연결을 삭제해도 이미 수집된 과거 데이터는 삭제되지 않습니다.
- **재시도**는 실패하거나 만료된 연결의 활성화를 다시 요청합니다. 먼저 수집 환경의 설정과 네트워크를 확인합니다.
- **다시 연결**은 해제된 연결을 다시 활성화합니다.
- 삭제는 선택한 연결과 Monithub에 저장된 설정 계획을 제거하며 되돌릴 수 없습니다.

연결을 삭제해도 고객 환경에 적용한 수집 구성은 자동으로 중단되지 않습니다. 더 이상 전송하지 않을 경우 해당 카탈로그의 적용 가이드에 따라 수집 구성을 중단합니다.

조직 토큰 갱신

토큰 갱신은 현재 조직의 모든 Connect가 사용하는 조직 토큰을 새 값으로 교체합니다. 한 연결만 변경하는 기능이 아닙니다.

1. 갱신 확인창에서 영향을 받는 모든 Collector와 Agent를 다시 설정할 수 있는지 확인합니다.
2. **토큰 갱신**을 누릅니다.
3. 한 번만 표시되는 새 토큰을 즉시 복사해 안전한 비밀 저장소에 보관합니다.
4. 현재 조직의 모든 Collector, Agent, Operator 설정에서 기존 Bearer token을 교체합니다.
5. 각 수집기를 재시작하거나 배포 설정을 다시 적용합니다.

6. 연결 목록과 [데이터 수집 확인](#)에서 복구 여부를 확인합니다.

기존 토큰은 갱신 즉시 사용할 수 없게 됩니다. 일부 수집기만 교체하면 나머지 연결에서 403 응답과 수집 중단이 발생합니다.

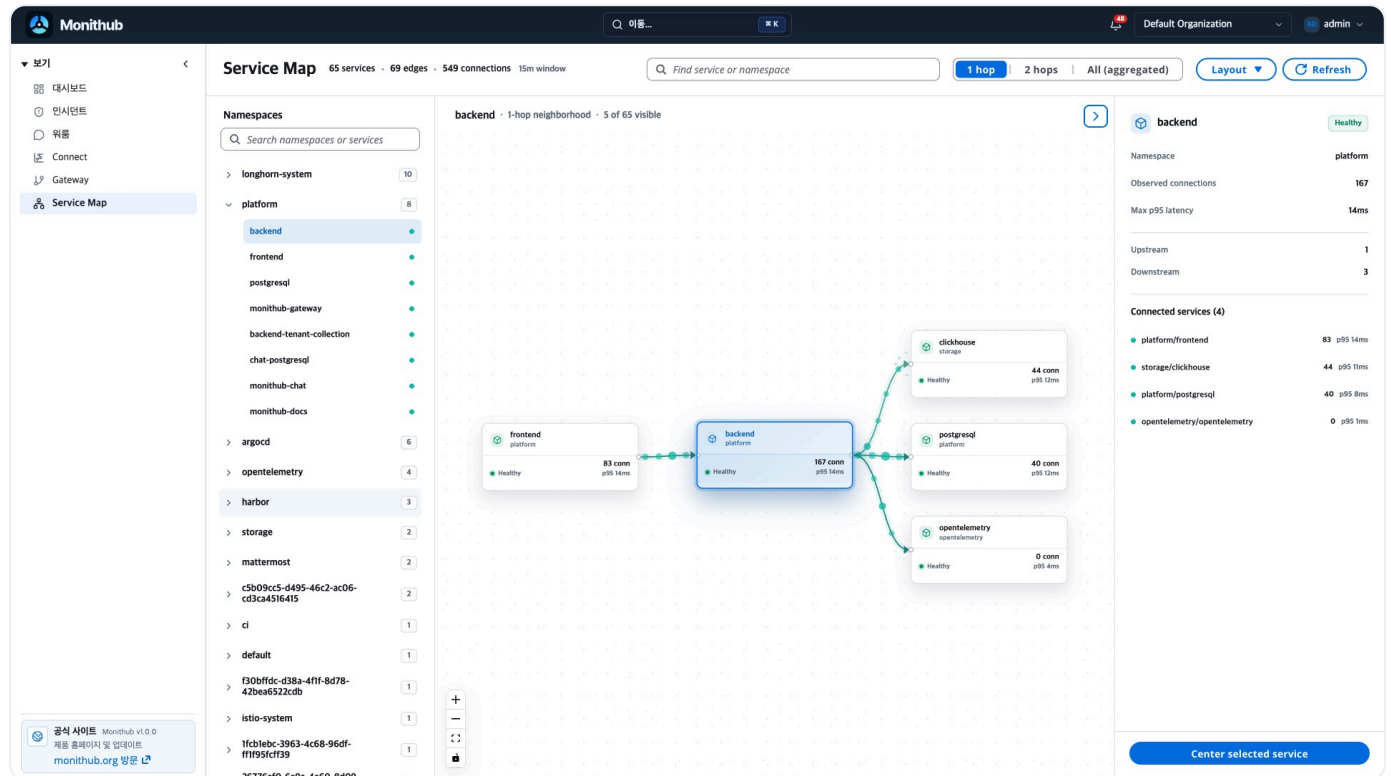
다음 단계 [🔗](#)

수집이 정상이라면 [대시보드](#)에서 메트릭을 확인하고, [서비스 맵](#)에서 서비스 간 흐름을 확인합니다. 연결된 데이터를 정기 분석하려면 [Sentinel 주기 분석](#)을 확인합니다.

서비스 맵

서비스 맵은 최근 15분 동안 관찰된 서비스 연결을 토폴로지로 보여 줍니다. 문제가 있는 서비스를 중심으로 upstream과 downstream을 좁혀 보며 지연이나 실패가 전파되는 경로를 확인할 수 있습니다.

서비스 맵 열기 [🔗](#)



왼쪽 메뉴에서 **Service Map**을 선택합니다. 처음에는 한 서비스를 중심으로 직접 연결된 서비스가 표시됩니다.

영역	설명
상단 요약	전체 서비스와 edge 수, 관찰된 연결 수, 조회 구간을 보여 줍니다.
검색	서비스 또는 네임스페이스 이름으로 대상을 찾습니다.
Namespaces	네임스페이스별 서비스를 펼쳐 보고 분석할 서비스를 선택합니다.
토폴로지 범위	선택한 서비스를 기준으로 1 hop , 2 hops , All (aggregated) 범위를 전환합니다.
그래프	서비스 상태와 연결 방향, 연결 수, p95 지연 시간을 보여 줍니다.
상세 패널	선택한 서비스의 상태, 네임스페이스, 연결 수, 최대 p95 지연 시간, upstream·downstream 수를 보여 줍니다.
Layout	그래프를 다시 배치하거나 배치 방식과 간격을 조정합니다.
Refresh	최신 서비스 맵 데이터를 즉시 다시 불러옵니다.

연결 범위 좁혀 보기 [🔗](#)

1. 검색창이나 **Namespaces** 목록에서 분석할 서비스를 선택합니다.
2. [1 hop](#)에서 직접 연결된 upstream과 downstream을 확인합니다.
3. 원인을 더 넓게 추적해야 하면 [2 hops](#)로 전환합니다.
4. 전체 토폴로지를 비교해야 할 때만 [All \(aggregated\)](#)을 사용합니다.
5. 그래프의 노드 또는 오른쪽 **Connected services** 항목을 선택해 분석 대상을 이동합니다.

노드의 [Healthy](#), [Slow](#), [Failed](#) 상태와 p95 지연 시간을 함께 확인합니다. 연결이 많아 그래프가 복잡하면 [1 hop](#)부터 시작하고, **Layout**의 재배치 또는 **Fit view**를 사용해 화면을 정리합니다.

문제 구간 분석하기 [🔗](#)

1. [Slow](#) 또는 [Failed](#) 상태인 서비스를 선택합니다.
2. 상세 패널에서 최대 p95 지연 시간과 연결 수를 확인합니다.
3. [1 hop](#)에서 직접 연결된 upstream과 downstream의 상태를 비교합니다.
4. 원인 후보가 보이지 않으면 [2 hops](#)로 범위를 넓힙니다.
5. 관련 인시던트가 있다면 인시던트 화면의 메트릭·로그·Trace·Kubernetes 이벤트를 함께 확인합니다.

Trace는 별도 메뉴나 독립 화면으로 제공하지 않습니다. 조직별 대시보드의 Trace 패널 또는 인시던트의 telemetry context에서 확인합니다.

서비스 맵이 보이지 않을 때

- 현재 조직에 `dashboard.read`와 `datasource.read` 권한이 있는지 확인합니다.
- **Settings > ClickHouse**에서 데이터소스가 활성화되어 있고 연결 테스트가 성공하는지 확인합니다.
- 데이터가 비어 있으면 최근 15분에 서비스 간 호출이 있었는지와 Collector 또는 agent의 export 상태를 확인합니다.
- 로딩에 실패하면 ClickHouse 연결과 Backend 응답을 차례로 점검합니다.

Monithub Sentinel

Monithub Sentinel은 수집된 데이터의 흐름을 분석해 살펴봐야 할 이상 징후를 찾도록 돕는 분석 엔진입니다. Sentinel 화면에서는 분석 모델을 생성하고, 탐지 결과에 라벨을 남겨 판단을 관리하며, 주기 분석과 최근 분석 결과를 확인할 수 있습니다. 필요할 때는 재학습과 튜닝으로 데이터 흐름에 맞게 분석 기준을 조정합니다.

Sentinel에서 하는 일 [🔗](#)

화면 영역	확인할 내용
모델 버전	모델 생성으로 만든 후보와 현재 활성 버전의 상태를 확인합니다.
모델 상세 정보	선택한 모델의 품질, 소스 커버리지 등 상세 정보를 확인합니다.
라벨	탐지 결과에 남긴 정탐, 오탐, 미탐, 무시됨 판단을 검토합니다.
주기 분석	현재 분석 주기와 다음 분석 예정 시각을 확인하거나 변경합니다.
최근 분석	최근에 실행한 분석의 시각, 대상 범위와 상태를 확인합니다.

화면을 보는 순서

1. **모델 버전**에서 현재 활성 버전과 새 후보, 학습 중인 항목이 있는지 확인합니다.
2. 확인할 모델을 선택해 **모델 상세 정보**에서 품질과 소스 커버리지 등 상세 정보를 봅니다.
3. **라벨**을 열어 탐지 결과에 남긴 판단과 재학습 반영 상태를 검토합니다.
4. **주기 분석**에서 다음 분석 예정 시각을, **최근 분석**에서 최근 실행 결과를 확인합니다.

화면마다 확인하는 목적이 다릅니다

모델 버전과 상세 정보는 운영 기준을 고르는 데 사용합니다. 라벨은 탐지 결과에 대한 판단을 남기는 기록이고, 주기 분석과 최근 분석은 설정한 기준으로 분석이 정상 실행되고 있는지 확인하는 영역입니다.

시작하기 전에

Sentinel을 열기 전에 상단에서 현재 조직이 맞는지 확인합니다. 모델, 학습 실행, 라벨은 조직별로 구분되어 보입니다.

처음 사용하는 경우에는 모델 버전이 없다는 안내와 **첫 학습** 버튼이 표시됩니다. 모델을 이미 사용하고 있다면 현재 활성 버전, 새 후보, 최근 학습 상태부터 확인합니다.

기본 사용 흐름

1. 모델 버전이 없다면 **실습 가이드**를 따라 첫 학습부터 후보 활성화까지 진행합니다.

2. [모델 관리](#)에서 현재 활성 버전과 후보의 상태, 품질, 소스 커버리지를 확인합니다.
3. 새 후보가 필요하면 [학습과 튜닝](#)에서 재학습과 설정 조정을 검토합니다.
4. [주기 분석](#)에서 분석 간격과 다음 분석 예정 시각을 확인합니다.
5. [최근 분석](#)에서 최근 실행 결과를 확인합니다.
6. 탐지 결과를 검토한 뒤 [라벨](#)에 판단을 남기고, 반복되는 문제는 [확인](#)과 [문제 해결](#)에서 점검합니다.

주기 분석과 최근 분석

활성 모델과 데이터 수집 연결이 준비되면 Sentinel은 정해진 간격으로 분석을 실행합니다. [주기 분석](#)에서는 현재 분석 주기와 다음 분석 예정 시각을 확인할 수 있습니다.

분석 주기를 바꿀 수 있는 권한이 있다면 원하는 간격을 고른 뒤 [변경 내용 확인](#)을 누릅니다. 표시되는 다음 분석 예정 시각을 확인하고 [변경 저장](#)으로 적용합니다. 다른 사용자가 같은 설정을 바꾼 경우에는 최신 설정을 다시 확인한 뒤 저장합니다.

[최근 분석](#)에서는 분석 시각, 분석 대상 기간, 상태를 확인합니다. 실행이 실패했거나 조치가 필요하다는 안내가 보이면 Connect 연결과 분석에 사용할 대시보드 설정을 먼저 확인합니다.

Connect 연결이 없거나 활성 모델이 없으면 주기 분석이 시작되지 않습니다. 화면 안내에 따라 Connect를 연결하거나 모델 버전을 활성화한 뒤 상태를 다시 확인합니다.

다음에 볼 문서

문서	확인할 내용
실습 가이드	첫 학습, 후보 확인, 활성화까지의 버튼 흐름
모델 관리	버전 상태, 상세 정보, 후보 활성화와 삭제
학습과 튜닝	학습·재학습의 의미, 설정 조정, 실행 상태
라벨	라벨의 의미, 원본 화면 확인과 삭제 방법
확인 과 문제 해결	활성 버전이 없거나 학습이 실패할 때의 점검 순서

결과를 믿기 전에 확인할 것

Sentinel은 운영 판단을 돕는 기준입니다. 후보를 활성화하기 전에는 품질과 소스 커버리지를 확인하고, 실제 대시보드의 시간대와 함께 결과를 검토합니다. 결과가 기대와 다르다면 바로 여러 번 실행하기보다 현재 실행 상태와 데이터 연결 상태를 먼저 확인합니다.

모델 관리

모델 관리에서는 Sentinel이 사용할 모델 버전과 현재 상태를 확인합니다. 새 후보가 운영에 사용할 만큼 준비됐는지, 어떤 버전이 현재 활성 상태인지 비교하고 결정하는 화면입니다.

빨간 테두리로 표시된 모델 버전 목록과 모델 상세 정보를 함께 확인합니다. 목록에서 확인할 버전을 선택하면 오른쪽에서 해당 버전의 상세 정보를 볼 수 있습니다.

❗ 후보를 바로 활성화하지 마세요

새 후보를 선택한 뒤에는 먼저 품질과 소스 커버리지를 확인합니다. 현재 활성 버전과 탐지 결과도 함께 비교한 뒤 운영 반영 여부를 결정하세요.

모델 목록 읽기 [🔗](#)

모델 목록에서는 새로 만든 후보, 현재 사용 중인 활성 버전, 학습 중인 버전을 비교합니다. 모델을 선택하면 오른쪽의 **모델 상세 정보**가 선택한 버전 기준으로 바뀝니다.

항목	확인할 내용
모델 이름	목록에 표시된 모델 버전 이름입니다.
상태	학습 대기 중 , 학습 중 , 후보 , 활성화 중 현재 단계가 표시됩니다.
품질	후보가 운영 기준으로 검토 가능한지 판단하는 정보입니다. 학습 중인 항목은 결과가 표시되지 않을 수 있습니다.
모델 생성 시간	언제 만든 버전인지 확인합니다. 비슷한 후보가 여러 개면 생성 시간을 기준으로 비교합니다.
행 오른쪽 버튼	후보 행의 체크(✓) 버튼은 모델 버전 승격 에, X 버튼은 더 이상 보관하지 않을 버전 삭제에 사용합니다.

모델 상세 정보 읽기

목록에서 모델을 선택하면 오른쪽 [모델 상세 정보](#)에서 선택한 버전의 정보를 확인할 수 있습니다.

항목	확인할 내용
모델 이름	지금 선택해 검토하는 모델 버전이 맞는지 확인합니다.
소스 커버리지	학습에 반영된 데이터가 충분한지 살펴봅니다. 낮게 표시되면 바로 활성화하지 않습니다.
다음 학습 시간	자동 재학습이 설정된 경우 다음 학습 예정 시각을 확인합니다.
최근 활성화 시간	현재 또는 이전에 운영 기준으로 사용한 이력을 확인합니다.
학습 당시 탐지 성향	이 버전을 만들 때 선택한 탐지 성향을 확인합니다.

소스 커버리지가 낮거나 품질 확인이 충분하지 않다면, 해당 후보를 바로 활성화하지 않고 데이터와 설정을 먼저 다시 확인합니다.

모델 개수와 활성 상태 관리하기

모델 버전은 최대 3개까지 만들 수 있습니다. 이미 3개가 있다면 새 모델을 만들기 전에 더 이상 사용하지 않는 후보나 이전 버전을 정리합니다.

첫 학습이 완료된 뒤에는 Sentinel이 분석에 사용할 활성 모델을 항상 하나 이상 유지해야 합니다. 모델이 하나만 남아 있고 그 모델이 활성 상태라면 삭제할 수 없습니다. 새 버전으로 바꾸려면 먼저 새 후보를 만들

고 검토해 활성화한 뒤, 이전 버전을 정리합니다.

상태별로 할 일

상태	의미	다음 행동
학습 대기 중 또는 학습 중	새 버전을 만드는 중입니다.	같은 목적의 학습을 다시 만들지 말고 최종 상태가 바뀔 때까지 기다립니다.
후보	새 학습이 끝나 검토를 기다리는 버전입니다.	품질과 소스 커버리지를 확인하고 현재 활성 버전과 비교합니다.
활성화	현재 Sentinel이 분석 기준으로 사용하는 버전입니다.	라벨과 실제 대시보드를 함께 보면서 탐지 결과가 기대와 맞는지 확인합니다.

후보를 활성화하기

후보를 활성화하면 현재 운영 기준이 바뀝니다. 다음 순서로 진행합니다.

1. 목록에서 후보 버전을 선택합니다.
2. 오른쪽 모델 상세 정보에서 품질과 소스 커버리지를 확인합니다.
3. 후보 행 오른쪽의 체크(✓) 버튼인 [모델 버전 승격](#)을 누릅니다.
4. 목록에서 해당 버전의 상태가 [활성화](#)로 바뀌었는지 확인합니다.

실제 화면에서 후보를 확인하고 승격하는 전체 순서는 [실습 가이드](#)를 참고하세요.

활성화한 뒤에는 평소와 다른 결과가 늘어나지 않는지 대시보드와 라벨을 함께 살펴봅니다. 새 후보가 필요하면 [학습과 튜닝](#)에서 재학습을 시작합니다.

모델 버전 삭제

더 이상 보관할 필요가 없는 버전은 해당 행 오른쪽의 [X](#) 버튼을 눌러 삭제할 수 있습니다. 삭제한 버전은 복구할 수 없으므로, 현재 활성 버전인지와 나중에 비교할 필요가 없는지 확인한 뒤 진행합니다.

1. 삭제할 후보 또는 이전 버전이 맞는지 모델 이름과 상태를 확인합니다.
2. 현재 활성 버전이거나 비교에 필요한 후보가 아닌지 확인합니다.
3. 해당 행 오른쪽의 [X](#) 버튼을 누릅니다.

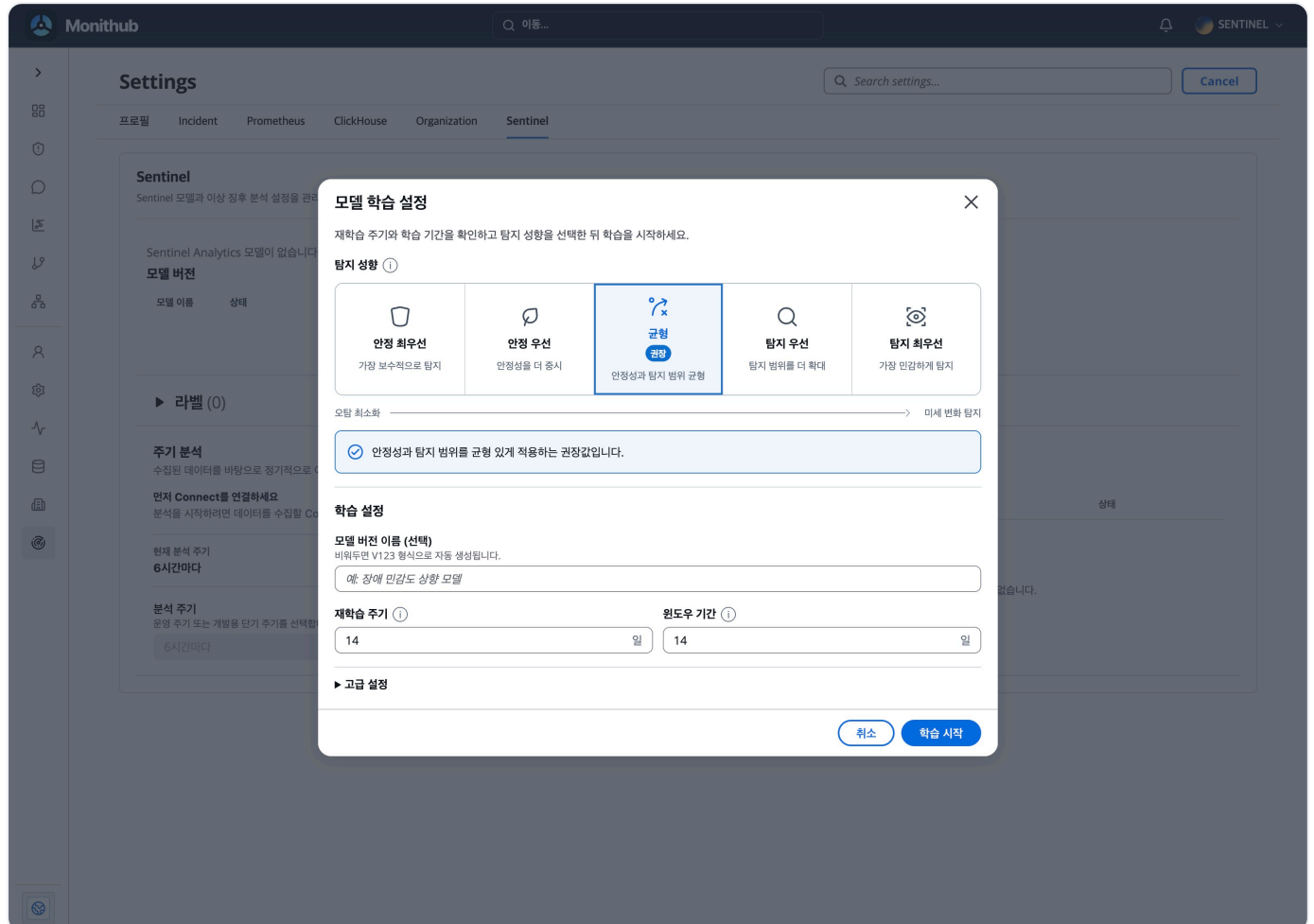
삭제한 모델 버전은 복구할 수 없습니다

삭제하기 전에 현재 활성 상태인지, 새 후보와 비교할 필요가 없는지 확인하세요. 첫 학습 이후에는 Sentinel이 사용할 활성 모델을 하나 이상 유지해야 합니다.

첫 학습 이후에는 활성 모델을 하나 이상 유지해야 합니다. 모델이 하나만 남아 있고 그 모델이 활성 상태이면  버튼으로 삭제할 수 없습니다. 삭제 권한이 없는 경우에도 버튼을 사용할 수 없습니다.

학습과 튜닝

학습은 Sentinel이 사용할 운영 기준을 만들고, 재학습은 이미 만든 모델에 최신 데이터와 설정을 반영해 새 후보 버전을 만드는 작업입니다. 튜닝은 Sentinel이 데이터 변화를 얼마나 민감하게 살필지 정하는 설정입니다. 실제 첫 학습부터 후보 활성화까지의 진행 흐름은 [실습 가이드](#)에서 안내합니다.



학습과 재학습의 차이 [🔗](#)

첫 학습은 모델 버전이 없는 상태에서 첫 후보를 만드는 작업입니다. 재학습은 기존 모델을 유지한 채 새 후보를 비교하고 싶을 때 사용합니다. 실제 화면에서 버튼을 누르는 순서는 [실습 가이드](#)를 참고하세요.

기존 모델은 유지한 채 새 후보를 비교하고 싶다면 **새 버전 학습**으로 재학습을 시작합니다. 아래와 같은 경우에 재학습을 검토합니다.

- 최근 배포나 운영 변화로 데이터 흐름이 달라졌습니다.
- 정상 구간이 자주 이상으로 표시되거나, 실제 이상을 자주 놓칩니다.
- 기존 후보의 품질이나 소스 커버리지를 다시 확인해야 합니다.

새 버전 학습의 버튼 순서와 후보 확인 방법은 [실습 가이드](#)에서 확인합니다.

같은 실행을 여러 번 반복하기 전에 현재 실행 상태와 화면 메시지를 먼저 확인합니다. 모델, 버전, 실행 상태를 다시 확인할 때는 [새로고침](#)을 사용합니다.

실행 상태 확인하기

상태	확인할 내용
학습 대기 중	설정을 다시 확인하고, 계속 진행할지 판단합니다.
학습 중	완료될 때까지 상태 변화를 확인합니다.
완료	새 후보가 표시되면 모델 관리에서 품질과 소스 커버리지를 확인합니다.
실패	화면 메시지와 입력값, 데이터 상태를 확인합니다.

[재학습 중단](#) 버튼이 보이는 실행은 더 이상 진행할 필요가 없을 때 중단할 수 있습니다. 중단하면 진행 중인 학습 결과는 저장되지 않고, 기존 활성 모델은 유지됩니다. 중단한 뒤에는 목록의 최종 상태를 확인하고, 새 실행이 필요하면 값을 점검한 뒤 다시 만듭니다.

탐지 성향 고르기

탐지 성향은 세부 수치를 직접 바꾸기 전에 선택할 수 있는 기본 설정입니다. 처음에는 [균형](#)을 권장하며, 운영 상황에 맞게 한 단계씩 조정합니다.

탐지 성향	사용하는 때
안정 최우선	확실한 이상만 우선 확인하고 싶을 때
안정 우선	오탐을 줄이는 방향으로 조정하고 싶을 때
균형	안정성과 탐지 범위를 고르게 적용하고 싶을 때
탐지 우선	작은 변화도 더 넓게 확인하고 싶을 때
탐지 최우선	오탐 가능성을 감수하고 미세한 변화까지 살피고 싶을 때

고급 설정 조정하기

고급 설정에서는 학습에 사용할 기간과 탐지 기준을 직접 조정할 수 있습니다. 기본 탐지 성향으로 결과를 먼저 확인한 뒤, 반복되는 오탐·미탐이나 학습 범위 문제가 있을 때 사용합니다.

❗ 고급 설정을 바꾸기 전에

첫 학습에서는 기본값을 유지하는 것을 권장합니다. 여러 값을 함께 바꾸면 어떤 변경이 결과에 영향을 주었는지 알기 어려우므로, 라벨과 원본 대시보드를 확인한 뒤 한 번에 한 항목씩 조정하세요.

학습 설정

모델 버전 이름 (선택)

비워두면 V123 형식으로 자동 생성됩니다.

예: 장애 민감도 상향 모델

재학습 주기 ⓘ

14

일

윈도우 기간 ⓘ

14

일

▼ 고급 설정

Sentinel Model 튜닝 수치를 직접 설정합니다.

오염 비율 ⓘ

0.01

트리 개수 ⓘ

150

설정	무엇을 조정하나요	이런 경우 먼저 검토하세요
재학습 주기	새 후보를 만드는 간격입니다. 짧게 하면 최근 변화가 더 빨리 반영될 수 있지만, 검토할 후보도 더 자주 생깁니다.	최근 배포나 운영 변화가 자주 발생해 기준을 더 자주 갱신해야 할 때
윈도우 기간	학습에 포함할 과거 데이터의 범위입니다. 짧게 하면 최근 흐름에 집중하고, 길게 하면 더 다양한 과거 흐름을 함께 반영합니다.	최근 변화가 반영되지 않거나, 짧은 기간의 변동에 결과가 크게 흔들릴 때
오염 비율	이상으로 분류할 구간의 비율을 정하는 기준입니다. 값을 높이면 이상으로 표시되는 범위가 넓어질 수 있고, 낮추면 더 보수적으로 표시됩니다.	정상 구간이 자주 이상으로 보이거나 실제 이상을 반복해서 놓칠 때
트리 개수	학습 결과를 더 안정적으로 만들기 위해 사용하는 반복 횟수입니다. 값을 높이면 결과가 더 안정적일 수 있지만 학습 시간이 길어질 수 있습니다. 화면에서 허용하는 범위는 50~300입니다.	학습 결과가 일관되지 않거나 학습 시간이 오래 걸릴 때

고급 설정 조정 순서

- 1. 먼저 [탐지 성향](#)을 조정해도 해결되는지 확인합니다.
- 2. [라벨](#)과 원본 대시보드에서 오탐·미탐이 반복되는 시간대와 패턴을 확인합니다.
- 3. 고급 설정에서는 한 항목만 조정합니다. 최근 변화 반영이 문제라면 재학습 주기 또는 윈도우 기간을, 탐지 범위가 문제라면 오염 비율을 먼저 검토합니다.
- 4. [학습 시작](#)으로 새 후보를 만든 뒤 완료 상태를 확인합니다.
- 5. [모델 관리](#)에서 새 후보의 품질과 소스 커버리지를 비교하고, 운영에 반영할지 결정합니다.

결과에 따라 조정하기

관찰한 결과	먼저 조정할 항목
정상 구간이 자주 이상으로 표시됩니다.	먼저 탐지 성향을 안정 우선 쪽으로 조정합니다.
실제 이상을 자주 놓칩니다.	먼저 탐지 성향을 탐지 우선 쪽으로 조정합니다.
최근 변화가 충분히 반영되지 않습니다.	윈도우 기간과 재학습 주기를 확인합니다.
학습이 오래 걸립니다.	고급 설정의 트리 개수 또는 윈도우 기간을 검토합니다.

새 실행이 끝나면 [모델 관리](#)에서 후보를 검토합니다. 실행이 계속 실패하면 [확인](#)과 [문제 해결](#)의 순서대로 점검합니다.

주기 분석

주기 분석은 Connect로 수집된 데이터를 Sentinel이 정해진 간격으로 분석하는 기능입니다. [설정 → Sentinel](#)에서 현재 주기, 다음 분석 예정 시각과 최근 실행 결과를 함께 확인할 수 있습니다.

분석 시각	분석 대상 범위	상태
2026. 8. 17. 오전 11:34:02	2026. 8. 17. 오전 5:34:00 ~ 2026. 8. 17. 오전 11:34:00	진행 중
2026. 8. 17. 오전 5:34:02	2026. 8. 16. 오후 11:34:00 ~ 2026. 8. 17. 오전 5:34:00	진행 중
2026. 8. 16. 오후 11:34:02	2026. 8. 16. 오후 5:34:00 ~ 2026. 8. 16. 오후 11:34:00	진행 중
2026. 8. 16. 오후 5:34:03	2026. 8. 16. 오전 11:34:00 ~ 2026. 8. 16. 오후 5:34:00	진행 중
2026. 8. 16. 오전 11:34:02	2026. 8. 16. 오전 5:34:00 ~ 2026. 8. 16. 오전 11:34:00	진행 중

- [주기 분석](#)에서 현재 주기와 다음 분석 예정 시각을 확인하거나 변경합니다.
- [최근 분석](#)에서 실행 시각, 분석 대상 범위와 상태를 확인합니다.

사용 조건 [🔗](#)

- 현재 조직에 연결된 Connect가 하나 이상 있어야 합니다.
- Sentinel에서 사용할 모델 버전이 활성화되어 있어야 합니다.
- 설정 조회 또는 변경 권한이 필요합니다.

화면 안내

필요한 작업

먼저 Connect를 연결하세요	Connect에서 수집 설정을 만들고 데이터가 들어오는지 확인합니다.
분석이 일시 중지되었습니다	연결이 다시 활성화되면 다음 분석이 자동으로 예약됩니다.
먼저 Sentinel 모델을 활성화하세요	학습을 완료하고 검토한 모델 버전을 활성화합니다.
변경 권한이 없습니다	조직 관리자에게 Sentinel 설정 변경 권한을 요청합니다.

분석 주기 변경 [🔗](#)

1. 상단에서 대상 조직을 선택합니다.
2. [설정](#) → [Sentinel](#)을 엽니다.
3. [분석 주기](#)에서 원하는 간격을 선택합니다.
4. [변경 내용 확인](#)을 누릅니다.
5. 확인창에서 변경 후 주기와 다음 분석 예정 시각을 확인합니다.
6. [변경 저장](#)을 누르고 다음 예정 시각이 갱신됐는지 확인합니다.

다른 사용자가 같은 조직의 설정을 먼저 바꿨다는 안내가 나오면 화면을 새로고침한 뒤 최신 값으로 다시 저장하세요.

최근 분석 확인 [🔗](#)

항목	의미
분석 시각	주기 분석이 시작된 시각
분석 대상 범위	이번 실행이 확인한 데이터의 시작·종료 시각
상태	진행 중, 완료 또는 실패 여부

최근 분석이 완료되면 해당 시간대의 대시보드와 Sentinel 결과를 함께 확인합니다. 실패가 표시되면 데이터 수집 상태와 활성 모델을 먼저 확인하세요.

예정 시각이 지나도 실행되지 않을 때 [🔗](#)

1. [연결 및 수집 확인](#)에서 하나 이상의 연결이 [연결됨](#)이고 최근 데이터가 있는지 확인합니다.
2. Sentinel 모델 목록에서 사용할 버전이 활성 상태인지 확인합니다.
3. 다음 분석 예정 시각이 표시되는지 확인합니다.
4. 예정 시각이 지난 뒤 [최근 분석](#)을 다시 확인합니다.

주기 분석은 데이터 수집기를 대신하지 않습니다. Connect에서 데이터가 들어오지 않으면 주기를 설정해도 새 분석 결과가 만들어지지 않습니다.

라벨

라벨은 Sentinel이 표시한 구간을 검토한 뒤 남기는 판단 기록입니다. 라벨만 보고 결론을 내리기보다, 원본 대시보드의 시간대와 메트릭을 함께 확인해 다음 조치를 정합니다.

▼ 라벨 (39)							
수정	라벨	대상 유형	패널	메트릭	재학습 반영	열기	Actions
2026. 8. 16. 오후 7:29:24	정탐	이상	ts-node-cpu	k8s_node_cpu_utilization_ratio	미반영	열기	×
2026. 8. 16. 오후 7:29:18	정탐	이상	ts-pod-network	k8s_pod_network_io_bytes_total	미반영	열기	×
2026. 8. 16. 오후 7:29:15	정탐	이상	ts-pod-network	k8s_pod_network_io_bytes_total	미반영	열기	×
2026. 8. 16. 오후 7:29:12	정탐	이상	ts-pod-cpu-top	k8s_pod_cpu_utilization_ratio	미반영	열기	×
2026. 8. 16. 오후 7:29:07	정탐	이상	ts-node-cpu	k8s_node_cpu_utilization_ratio	미반영	열기	×
2026. 8. 16. 오후 7:28:39	미탐	구간	ts-node-cpu	worker-kr-1	미반영	열기	×
2026. 8. 16. 오후 7:28:15	정탐	이상	ts-pod-cpu-top	k8s_pod_cpu_utilization_ratio	미반영	열기	×
2026. 8. 15. 오후 2:36:15	오탐	이상	km-ts-node-cpu	k8s_node_cpu_utilization_ratio	V15 반영됨	열기	×
2026. 8. 15. 오후 2:36:13	오탐	이상	ts-node-cpu	k8s_node_cpu_utilization_ratio	V15 반영됨	열기	×
2026. 8. 15. 오후 2:35:58	오탐	이상	ts-node-cpu	k8s_node_cpu_utilization_ratio	V15 반영됨	열기	×

❗ 라벨은 판단을 남기는 기록입니다

라벨을 남겼다고 현재 활성 모델이 바로 바뀌지는 않습니다. 같은 판단이 반복되는지 확인하고, 필요할 때 새 후보 학습을 만들어 결과를 비교하세요.

라벨의 의미 [🔗](#)

라벨	의미	다음 행동
정탐	실제로 확인이 필요한 이상 신호입니다.	원인과 영향 범위를 확인하고 필요한 대응을 진행합니다.
오탐	이상으로 표시됐지만 실제 이상이 아닌 신호입니다.	같은 상황이 반복되는지 확인하고 튜닝 조정이 필요한지 검토합니다.
미탐	이상으로 봐야 했지만 Sentinel이 표시하지 못한 구간입니다.	해당 시간대와 데이터를 다시 확인하고 재학습 또는 튜닝을 검토합니다.
무시됨	현재 판단에서 제외하기로 한 신호입니다.	제외한 이유를 확인하고 필요할 때 다시 검토합니다.

라벨 목록 읽기 [🔗](#)

라벨 목록에서는 아래 정보를 함께 확인합니다.

항목	확인할 내용
수정	판단이 마지막으로 바뀐 시각을 확인합니다.
라벨	정탐, 오탐, 미탐, 무시됨 중 어떤 판단인지 확인합니다.
대상 유형	한 번의 이상 신호인 이상 인지, 일정 시간 범위인 구간 인지 구분합니다.
패널	어느 대시보드 패널에서 확인한 내용인지 봅니다.
메트릭	어떤 지표를 보고 판단했는지 확인합니다.
재학습 반영	V15 반영됨 처럼 라벨이 어느 모델 학습에 반영됐는지, 미반영 으로 남아 있는지 확인합니다.
열기	라벨과 연결된 원본 화면으로 이동합니다.
Actions	해당 라벨을 삭제할 때 X 버튼을 누릅니다.

라벨이 많으면 목록 오른쪽 위의 페이지 번호와 화살표를 눌러 이전 기록도 확인할 수 있습니다. 라벨을 찾을 때는 먼저 패널과 메트릭을 기준으로 같은 판단이 반복되는지 살펴보세요.

라벨 확인 순서

1. 라벨 목록에서 **수정** 시각, **패널**, **메트릭**을 확인해 같은 상황의 기록을 찾습니다.
2. **열기**를 눌러 원본 화면에서 라벨 시점과 실제 변화 시점이 맞는지 확인합니다.
3. 한 번의 신호인지, 일정 시간 동안 이어진 구간인지 **대상 유형**을 확인합니다.
4. 정탐·오탐·미탐·무시됨 중 현재 상황에 맞는 판단인지 검토합니다.
5. 같은 오탐 또는 미탐이 반복되면 **학습과 튜닝**에서 새 후보 학습을 검토합니다.

원본 화면에서 확인하기

라벨 목록의 **열기**를 누르면 판단과 연결된 원본 화면으로 이동합니다. 원본 화면에서는 다음을 함께 확인합니다.

1. 라벨을 남긴 시점과 실제 변화 시점이 맞는지 확인합니다.
2. 같은 시간대의 패널과 메트릭이 어떤 흐름을 보였는지 확인합니다.
3. 관련 인시던트, 배포, 운영 변경 사항이 있었는지 확인합니다.
4. 판단을 유지할지, 조정할지 결정합니다.

재학습 반영 상태 이해하기

재학습 반영 열은 라벨이 어떤 모델 학습에 사용됐는지 보여줍니다.

표시	의미	다음 행동
V15 반영 됨	라벨이 표시된 버전의 학습에 반영된 상태입니다.	해당 후보의 품질과 소스 커버리지를 확인합니다.
미반영	라벨 기록은 남아 있지만, 아직 새 후보 학습에 반영되지 않은 상태입니다.	같은 판단이 반복되는지 확인하고 필요하면 새 버전 학습을 검토합니다.

라벨이 **미반영**이라고 해서 잘못된 기록인 것은 아닙니다. 라벨을 충분히 검토한 뒤, 재학습으로 새 후보를 만들 필요가 있을 때만 다음 단계를 진행합니다.

라벨 삭제하기

더 이상 사용할 필요가 없는 라벨은 아래 순서로 삭제합니다.

- 삭제할 라벨이 있는 행의 **Actions** 열을 확인합니다.
- X 버튼을 누릅니다.
- 이상 라벨 삭제** 확인 창에서 라벨, 패널, 메트릭이 맞는지 다시 확인합니다.
- 삭제할 라벨이 맞으면 **이상 라벨 삭제**를 누릅니다. 삭제하지 않으려면 **취소**를 누릅니다.

판단을 바꾸려면 삭제 전에 원본 화면을 다시 확인하세요

삭제한 라벨은 복구할 수 없고 이후 재학습에 반영할 수 없습니다. 단순히 판단이 확실하지 않다는 이유만으로 바로 삭제하기보다, 원본 대시보드와 같은 시간대의 변화를 다시 확인하세요.

삭제하면 라벨 목록에서 제거되며, 이후 재학습에 반영할 수 없고 복구할 수 없습니다. X 버튼이 비활성화되어 있으면 현재 계정에 라벨을 삭제할 권한이 없는 상태입니다.

반복되는 라벨을 활용하는 방법

반복되는 판단	먼저 확인할 내용	다음 단계
오탐	같은 시간대에 정상적인 변화가 반복되는지	탐지 성향과 고급 설정 조정을 검토합니다.
미탐	놓친 구간의 시간대와 메트릭 변화	새 후보 학습과 탐지 범위 조정을 검토합니다.
정탐 판단이 엇갈림	원본 화면의 변화와 운영 상황	판단 근거를 다시 확인하고 필요하면 라벨을 재검토합니다.
무시됨 이 반복됨	제외한 이유가 지금도 유효한지	필요하면 원본 화면을 다시 확인해 판단을 검토합니다.

라벨이 보이지 않거나 **열기**로 원본 화면이 열리지 않으면 **새로고침**한 뒤 현재 조직과 시간 범위를 확인합니다. 라벨 목록이 비어 있으면 아직 남겨진 판단 기록이 없는 상태입니다.

라벨은 결과를 더 잘 이해하기 위한 기록입니다. 중요한 운영 판단은 항상 실제 대시보드와 현재 상황을 함께 확인한 뒤 내립니다.

확인하고 문제 해결

Sentinel 결과가 기대와 다르거나 학습이 진행되지 않을 때는, 멈춘 단계를 먼저 찾는 것이 중요합니다. 아래 순서대로 확인하면 불필요한 재실행을 줄이고 다음 조치를 정하기 쉽습니다.

❗ 먼저 현재 상태를 확인하세요

같은 학습이나 새 버전 학습을 반복해서 시작하기 전에 모델 목록, 학습 상태, 화면에 표시된 안내를 먼저 확인하세요. 현재 상태에 맞는 조치를 선택하면 불필요한 후보가 늘어나는 일을 줄일 수 있습니다.

상태별 빠른 안내

화면에서 보이는 상태	먼저 확인할 것	다음 조치
모델 버전이 없고 첫 학습 이 보임	현재 조직과 데이터 수집 연결	실습 가이드 를 따라 첫 후보를 만듭니다.
학습 대기 중 또는 학습 중	현재 실행의 상태와 시작 시각	같은 학습을 다시 만들지 말고 완료 또는 실패 상태를 확인합니다.
후보가 표시됨	품질과 소스 커버리지	모델 관리 에서 운영에 반영할지 검토합니다.
활성 모델이 없음	후보의 상태와 활성화 가능 여부	검토한 후보를 활성화한 뒤 주기 분석 상태를 다시 확인합니다.
Connect 연결 또는 활성 모델이 필요하다는 안내	데이터 수집 연결과 활성 버전	필요한 연결 또는 활성 버전을 준비한 뒤 새로고침 으로 상태를 확인합니다.
최근 분석이 비어 있거나 진행 중	분석 예정 시각과 최근 분석 상태	분석 주기와 연결 상태를 확인하고 다음 실행 결과를 기다립니다.

1분 점검 순서

1. 현재 조직이 맞는지 확인합니다.
2. 활성 버전이 있는지 확인합니다.
3. 최근 후보의 품질과 소스 커버리지를 확인합니다.
4. 학습 실행 상태와 화면 메시지를 확인합니다.
5. 주기 분석의 연결 상태와 최근 분석 결과를 확인합니다.

6. 아래 증상별 안내에 따라 한 가지씩 조치한 뒤 상태를 다시 확인합니다.

새 모델을 만들 수 없을 때

모델 버전은 최대 3개까지 만들 수 있습니다. 이미 3개가 있다면 새 모델을 만들기 전에 [모델 관리](#)에서 더 이상 사용하지 않는 후보나 이전 버전을 정리합니다.

첫 학습 후에는 활성 모델을 하나 이상 유지해야 합니다. 모델이 하나만 남아 있고 활성 상태라면 해당 행의 **X** 버튼으로 삭제할 수 없습니다. 새 모델이 필요하면 새 후보를 만들고 검토해 활성화한 뒤, 이전 버전을 정리합니다.

학습 설정 화면에서 입력값 안내가 보이면 빈 항목이나 입력 범위를 먼저 확인합니다. 값을 고친 뒤 **학습 시작**을 다시 누르고, 새 실행이 목록에 표시되는지 확인합니다.

활성 버전이 없을 때

활성 버전이 없으면 Sentinel 결과를 현재 운영 기준으로 사용하기 어렵습니다.

1. 모델 버전 목록에 후보가 있는지 확인합니다.
2. 후보가 있으면 품질과 소스 커버리지를 확인합니다.
3. 조건이 맞는 후보를 활성화합니다.
4. 후보가 없으면 **첫 학습** 또는 **새 버전 학습**으로 새 후보를 만듭니다.
5. 실행이 끝난 뒤 후보와 활성 상태를 다시 확인합니다.

소스 커버리지가 낮을 때

소스 커버리지는 학습에 반영된 데이터가 충분한지 판단하는 정보입니다. 낮게 표시되면 결과를 바로 활성화하기보다 아래 항목을 점검합니다.

- 대시보드에서 필요한 시간대의 데이터가 정상적으로 보이는지 확인합니다.
- 사용하는 데이터 소스가 연결된 상태인지 확인합니다.
- 대시보드 변수나 선택한 대상이 비어 있지 않은지 확인합니다.
- 윈도우 기간이 너무 짧지 않은지 확인합니다.

확인한 뒤 새 실행을 만들고, 새 후보의 소스 커버리지를 다시 봅니다.

학습이 실패할 때

실패가 표시되면 먼저 화면의 메시지와 발생 시각을 확인합니다.

1. 튜닝 값에 빠진 항목이나 형식 오류가 없는지 확인합니다.
2. 학습에 사용할 기간의 데이터가 대시보드에 보이는지 확인합니다.
3. 현재 실행이 끝난 상태인지 확인합니다.
4. 값을 조정했으면 입력값을 다시 확인한 뒤 새 버전 학습을 시작합니다.
5. 같은 실패가 반복되면 메시지, 발생 시각, 확인한 조치를 함께 기록해 담당자에게 전달합니다.

상태가 바로 바뀌지 않을 때

값을 바꾸거나 실행을 만든 직후에는 화면 상태가 갱신되기까지 시간이 걸릴 수 있습니다.

- [새로고침](#)으로 모델, 버전, 실행 상태를 다시 확인합니다.
- 실행이 [학습 대기 중](#) 또는 [학습 중](#)이면 같은 작업을 반복해서 만들지 않고 완료 상태를 기다립니다.
- 일시적으로 실행할 수 없다는 안내가 보이면 잠시 뒤 다시 확인합니다.
- 계속 같은 안내가 보이면 화면 메시지와 발생 시각을 기록해 담당자에게 전달합니다.

진행 중인 학습은 반복해서 만들지 마세요

[학습 대기 중](#) 또는 [학습 중](#) 상태에서 같은 설정으로 새 실행을 여러 번 만들면 후보가 늘어나 비교가 어려워질 수 있습니다. 먼저 현재 실행의 최종 상태를 확인한 뒤 필요한 경우에만 새 학습을 시작하세요.

결과가 기대와 다를 때

정상 구간이 자주 이상으로 보이거나 실제 이상을 놓친다면, 바로 활성 버전을 바꾸기보다 라벨과 원본 대시보드를 먼저 확인합니다.

1. [라벨](#)에서 정탐, 오탐, 미탐 기록을 확인합니다.
2. 원본 화면에서 시간대와 메트릭 변화를 다시 봅니다.
3. 반복되는 패턴이 있다면 [학습과 튜닝](#)에서 한두 설정씩 조정합니다.
4. 새 후보가 만들어지면 품질과 소스 커버리지를 확인한 뒤 활성화 여부를 결정합니다.

주기 분석이 시작되지 않을 때

주기 분석에서 안내를 먼저 확인합니다.

- Connect 연결이 필요하다는 안내가 보이면 데이터 수집 연결을 먼저 준비합니다.
- 활성 모델이 필요하다는 안내가 보이면 후보 버전을 검토한 뒤 활성화합니다.
- 분석이 일시 중지됐다는 안내가 보이면 Connect 연결 상태를 확인한 뒤 다시 실행합니다.
- 최근 분석이 실패했거나 조치가 필요하다고 표시되면 Connect 연결과 분석에 사용할 대시보드 설정을 확인합니다.

주기 설정을 변경하는 버튼이 보이지 않거나 비활성화돼 있다면, 현재 권한으로는 설정을 변경할 수 없는 상태일 수 있습니다. 설정을 확인할 수 있는 담당자에게 도움을 요청합니다.

최근 분석 결과가 기대와 다를 때

최근 분석 목록에서 분석 시각, 분석 대상 범위, 상태를 확인합니다. 결과를 바로 판단하기보다 같은 시간대의 원본 대시보드와 [라벨](#)을 함께 확인하세요.

- 정상 구간이 반복해서 이상으로 보이면 탐지 성향을 [안정 우선](#) 쪽으로 조정할지 검토합니다.
- 실제 이상을 반복해서 놓치면 탐지 성향을 [탐지 우선](#) 쪽으로 조정할지 검토합니다.
- 같은 문제가 계속되면 [학습과 튜닝](#)에서 한 항목씩 조정해 새 후보를 만들고 비교합니다.

도움을 요청할 때 남길 정보

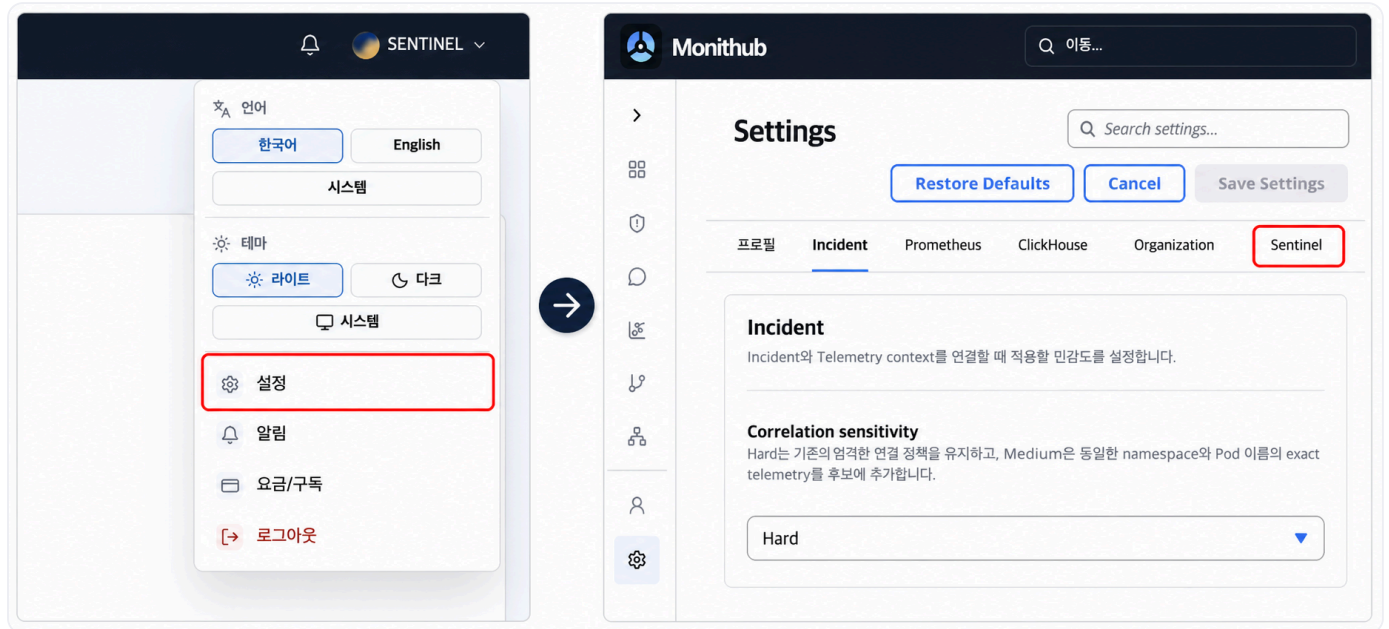
문제가 계속되면 아래 정보를 함께 전달하면 확인이 빨라집니다.

- 현재 보고 있는 조직과 화면
- 모델 또는 실행의 상태
- 화면에 표시된 메시지와 발생 시각
- 이미 확인하거나 시도한 조치

비밀번호, 토큰, 개인 정보처럼 민감한 값은 전달하지 않습니다.

실습 가이드

이 페이지는 모델 버전이 없는 상태에서 첫 학습을 시작하고, 만든 후보를 확인한 뒤 운영에 반영하는 순서를 화면의 버튼 이름에 맞춰 안내합니다. 설정값의 의미나 모델 상태가 궁금할 때는 각 단계의 링크를 열어 자세히 확인할 수 있습니다.

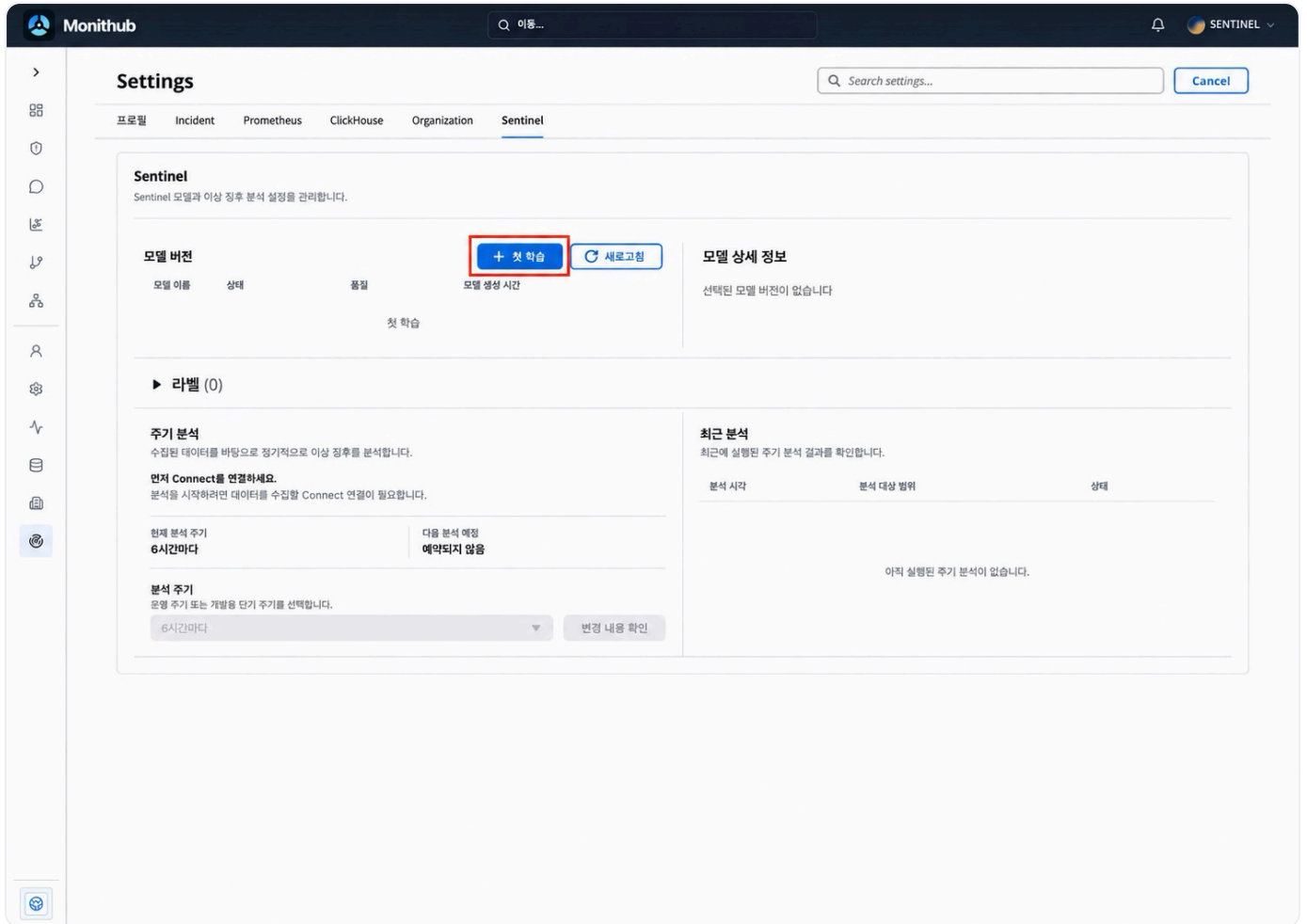


시작 전 확인하기 [🔗](#)

1. Sentinel 화면에서 현재 조직이 맞는지 확인합니다.
2. 모델 버전 영역에 모델 버전이 없는지 확인합니다.
3. 첫 학습 버튼이 보이지 않거나 사용할 수 없다면 [확인하고 문제 해결](#)에서 먼저 상태와 권한을 점검합니다.

1. 첫 학습 시작하기 [🔗](#)

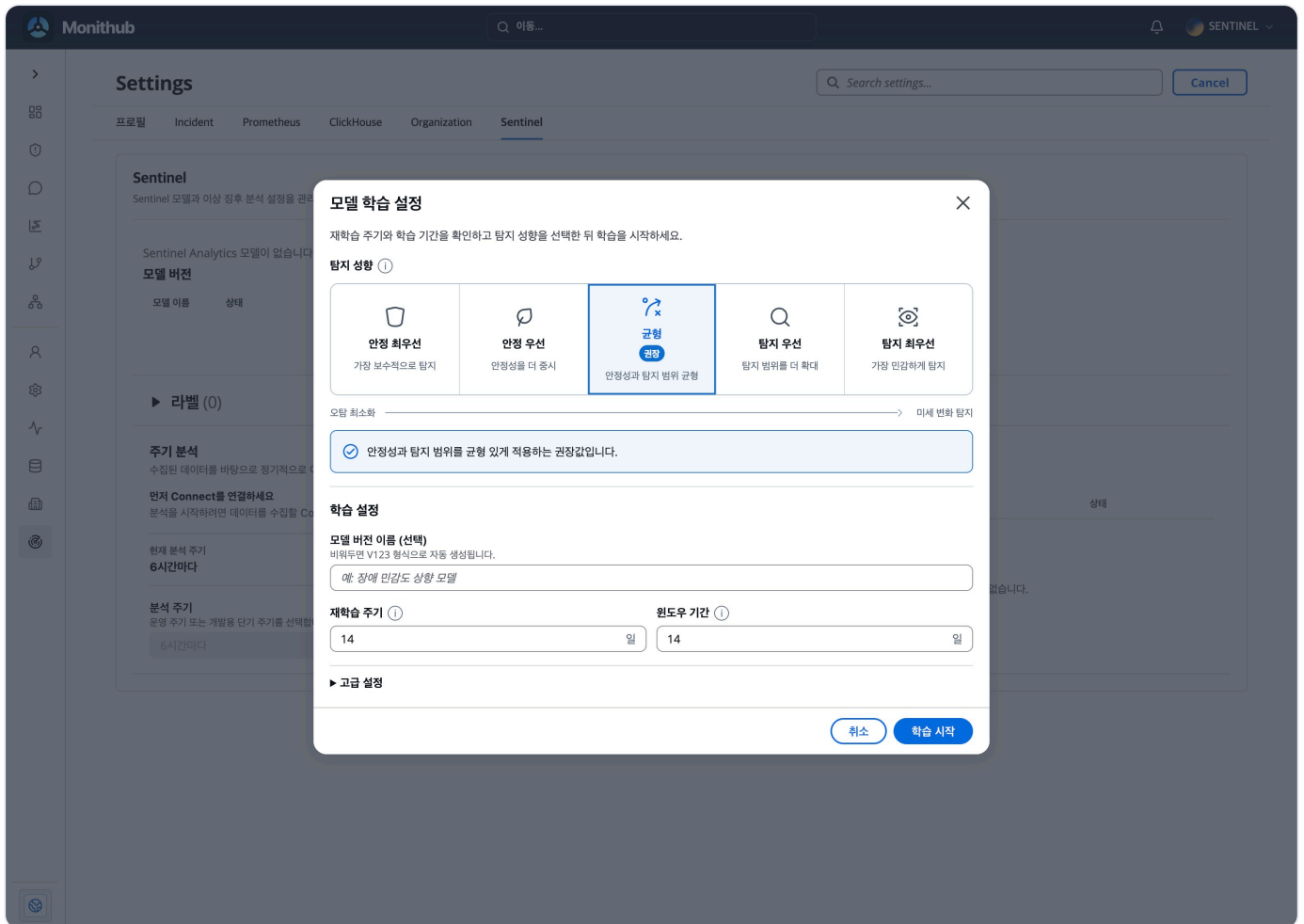
모델 버전 영역의 [첫 학습](#)을 누릅니다.



학습 설정 창이 열리면 다음 단계로 진행합니다.

2. 학습 설정 확인하고 시작하기 [🔗](#)

학습 설정 창에서 탐지 성향과 학습 기간을 확인한 뒤 화면 오른쪽 아래의 **학습 시작**을 누릅니다.



1. 처음 학습한다면 탐지 성향은 **균형**을 선택하는 것을 권장합니다.
2. 필요하면 모델 버전 이름을 입력합니다. 비워 두면 버전 이름이 자동으로 만들어집니다.
3. 재학습 주기와 윈도우 기간을 확인합니다.
4. 설정을 마쳤다면 화면 오른쪽 아래의 **학습 시작**을 누릅니다.

탐지 성향, 재학습 주기, 윈도우 기간과 고급 설정의 의미는 [학습과 튜닝](#)에서 확인할 수 있습니다.

2-1. 고급 설정 조정하기 [🔗](#)

고급 설정이 필요한 경우에만 학습 설정 하단의 **고급 설정**을 눌러 펼칩니다. 처음 학습할 때는 기본값을 유지하고, 라벨과 탐지 결과를 검토한 뒤 조정하는 것을 권장합니다.

학습 설정

모델 버전 이름 (선택)

비워두면 V123 형식으로 자동 생성됩니다.

예: 장애 민감도 상향 모델

재학습 주기 ⓘ

14

일

윈도우 기간 ⓘ

14

일

▼ 고급 설정

Sentinel Model 튜닝 수치를 직접 설정합니다.

오염 비율 ⓘ

0.01

트리 개수 ⓘ

150

1. [고급 설정](#)을 눌러 조정할 항목을 표시합니다.
2. 오탐이나 미탐이 반복되는 경우에만 [오염 비율](#)을 검토합니다. 이 값은 이상으로 판단할 구간의 범위에 영향을 줍니다.
3. 학습 결과의 안정성이나 실행 시간이 기대와 다를 때 [트리 개수](#)를 검토합니다.
4. 한 번에 여러 값을 바꾸지 말고 한두 항목만 조정한 뒤, 아래의 [학습 시작](#)을 눌러 새 후보를 만듭니다.

설정값별 영향과 조정 방향은 [학습과 튜닝](#)에서 자세히 확인할 수 있습니다.

3. 학습 완료 기다리기 [🔗](#)

학습을 시작하면 모델 목록에서 실행 상태를 확인합니다.

- [학습 대기 중](#) 또는 [학습 중](#)이면 같은 실행을 다시 만들지 않고 완료될 때까지 기다립니다.
- [완료](#)가 표시되면 새 후보가 만들어진 것입니다.
- [실패](#)가 표시되면 화면 메시지와 입력값을 확인한 뒤 [확인](#)과 [문제 해결](#)을 참고합니다.

Settings

프로필 Incident Prometheus ClickHouse Organization **Sentinel**

Search settings... Cancel

Sentinel
Sentinel 모델과 이상 징후 분석 설정을 관리합니다.

모델 버전 + 새 버전 학습 ↻ 새로고침

모델 이름	상태	품질	모델 생성 시간	
V16	학습 중	-	2026. 8. 16. 오후 8:25:01	X
V15	후보	주의	2026. 8. 15. 오후 2:41:41	X
V14	활성화	주의	2026. 8. 11. 오후 9:59:36	X
V13	후보	주의	2026. 8. 11. 오후 9:25:09	X

모델 상세 정보

모델 이름
V13

Sentinel Schema
0.1.0

소스 커버리지
78.4% (69/88)

최근 활성화 시간
2026. 8. 11. 오후 9:25:24

다음 학습 시간
2026. 8. 30. 오후 8:25:01

학습 당시 탐지 상황
균형

라벨 (39)

주기 분석
수집된 데이터를 바탕으로 정기적으로 이상 징후를 분석합니다. 분석이 예약되어 있습니다

현재 분석 주기
6시간마다

다음 분석 예정
2026. 8. 16. 오후 11:34:00

분석 주기
분석 주기 또는 개발을 단기 주기를 선택합니다.

6시간마다

변경 내용 확인

최근 분석
최근에 실행된 주기 분석 결과를 확인합니다.

분석 시간	분석 대상 범위	상태
2026. 8. 16. 오후 5:34:03	2026. 8. 16. 오전 11:34:00 ~ 2026. 8. 16. 오후 5:34:00	진행 중
2026. 8. 16. 오전 11:34:02	2026. 8. 16. 오전 5:34:00 ~ 2026. 8. 16. 오전 11:34:00	진행 중
2026. 8. 16. 오전 5:34:02	2026. 8. 15. 오후 11:34:00 ~ 2026. 8. 16. 오전 5:34:00	진행 중
2026. 8. 15. 오후 11:34:03	2026. 8. 15. 오후 5:34:00 ~ 2026. 8. 15. 오후 11:34:00	진행 중
2026. 8. 15. 오후 5:34:02	2026. 8. 15. 오전 11:34:00 ~ 2026. 8. 15. 오후 5:34:00	진행 중

< 1 2 >

4. 후보 확인하기 [🔗](#)

학습이 완료되면 [모델 관리](#)를 열어 새 후보를 선택합니다. 모델 상세 정보에서 아래 항목을 확인합니다.

- 품질
- 소스 커버리지
- 현재 활성 버전과의 차이

각 항목의 의미와 상태별 판단 방법은 [모델 관리](#)에서 자세히 확인할 수 있습니다.

Monithub

이름...

99%

Default Organization

SENTINEL

Settings

Search settings...

Cancel

프로필

Incident

Prometheus

ClickHouse

Organization

Sentinel

Sentinel

Sentinel 모델과 이상 징후 분석 설정을 관리합니다.

모델 버전

+ 새 버전 학습

새로고침

모델 이름	상태	품질	모델 생성 시간
V16	후보	주의	2026. 8. 16. 오후 8:28...
V15	후보	주의	2026. 8. 15. 오후 2:41...
V14	활성화	주의	2026. 8. 11. 오후 9:59...

모델 상세 정보

모델 이름

V16

Sentinel Schema

0.1.0

소스 커버리지

76.1% (67/88)

다음 학습 시간

2026. 8. 30. 오후 8:28:43

최근 활성화 시간

-

학습 당시 탐지 성향

균형

라벨 (39)

주기 분석

분석이 예약되어 있습니다

수집된 데이터를 바탕으로 정기적으로 이상 징후를 분석합니다.

현재 분석 주기

6시간마다

다음 분석 예정

2026. 8. 16. 오후 11:34:00

분석 주기

운영 주기 또는 개별용 단기 주기를 선택합니다.

6시간마다

변경 내용 확인

최근 분석

최근에 실행된 주기 분석 결과를 확인합니다.

분석 시작	분석 대상 범위	상태
2026. 8. 16. 오후 5:34:03	2026. 8. 16. 오전 11:34:00 ~ 2026. 8. 16. 오후 5:34:00	진행 중
2026. 8. 16. 오전 11:34:02	2026. 8. 16. 오전 5:34:00 ~ 2026. 8. 16. 오전 11:34:00	진행 중
2026. 8. 16. 오전 5:34:02	2026. 8. 15. 오후 11:34:00 ~ 2026. 8. 16. 오전 5:34:00	진행 중
2026. 8. 15. 오후 11:34:03	2026. 8. 15. 오후 5:34:00 ~ 2026. 8. 15. 오후 11:34:00	진행 중
2026. 8. 15. 오후 5:34:02	2026. 8. 15. 오전 11:34:00 ~ 2026. 8. 15. 오후 5:34:00	진행 중

5. 운영에 반영하기

후보를 운영에 사용하기로 결정했다면 다음 순서로 진행합니다.

1. 모델 관리에서 후보 버전을 선택합니다.
2. 후보 행 오른쪽의 체크(✓) 버튼인 **모델 버전 승격**을 누릅니다.
3. 목록에서 해당 버전의 상태가 **활성화**로 바뀌었는지 확인합니다.

Settings

프로필 Incident Prometheus ClickHouse Organization **Sentinel**

Sentinel
Sentinel 모델과 이상 징후 분석 설정을 관리합니다.

모델 버전 [+ 새 버전 학습](#) [새로고침](#)

모델 이름	상태	품질	모델 생성 시간	
V16	후보	주의	2026. 8. 16. 오후 8:28...	🔄 ✕
V15	후보	주의	2026. 8. 15. 오후 2:41...	🔄 ✕
V14	활성화	주의	2026. 8. 11. 오후 9:59...	🔄 ✕

모델 상세 정보

모델 이름: V16

Sentinel Schema: 0.1.0

소스 커버리지: 76.1% (67/88)

다음 학습 시간: 2026. 8. 30. 오후 8:28:43

최근 활성화 시간: -

학습 당시 탐지 성향: 균형

라벨 (39)

주기 분석 🟢 분석이 예약되어 있습니다

수집된 데이터를 바탕으로 정기적으로 이상 징후를 분석합니다.

현재 분석 주기: 6시간마다

다음 분석 예정: 2026. 8. 16. 오후 11:34:00

분석 주기: 운영 주기 또는 개별용 단기 주기를 선택합니다.

6시간마다 [변경 내용 확인](#)

최근 분석 최근에 실행된 주기 분석 결과를 확인합니다.

분석 시작	분석 대상 범위	상태
2026. 8. 16. 오후 5:34:03	2026. 8. 16. 오전 11:34:00 ~ 2026. 8. 16. 오후 5:34:00	🕒 진행 중
2026. 8. 16. 오전 11:34:02	2026. 8. 16. 오전 5:34:00 ~ 2026. 8. 16. 오전 11:34:00	🕒 진행 중
2026. 8. 16. 오전 5:34:02	2026. 8. 15. 오후 11:34:00 ~ 2026. 8. 16. 오전 5:34:00	🕒 진행 중
2026. 8. 15. 오후 11:34:03	2026. 8. 15. 오후 5:34:00 ~ 2026. 8. 15. 오후 11:34:00	🕒 진행 중
2026. 8. 15. 오후 5:34:02	2026. 8. 15. 오전 11:34:00 ~ 2026. 8. 15. 오후 5:34:00	🕒 진행 중

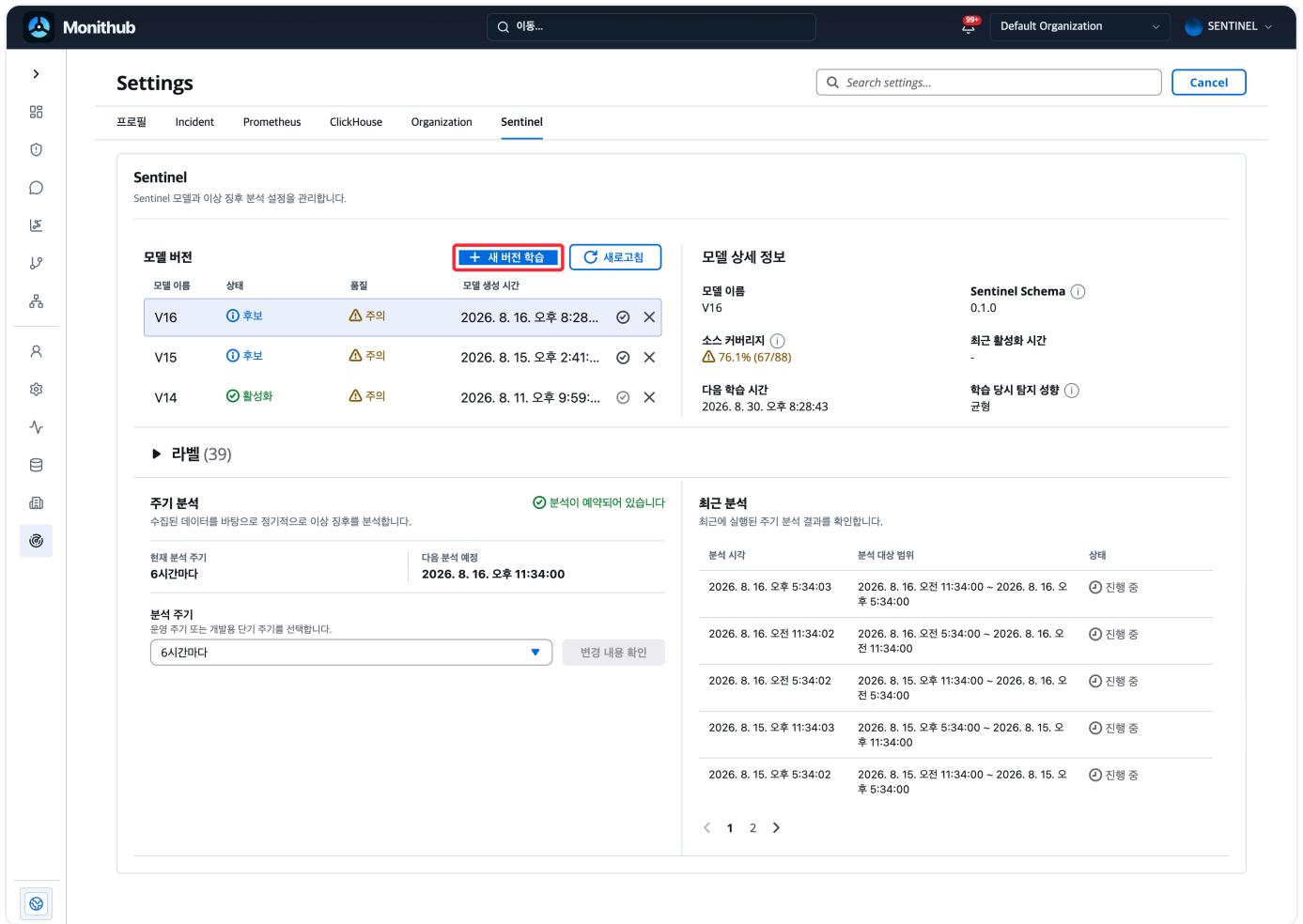
< 1 2 >

활성화한 뒤에는 [라벨](#)과 실제 대시보드를 함께 보면서 탐지 결과가 기대와 맞는지 확인합니다. 이후 새 후보가 필요할 때는 [학습과 튜닝](#)에서 재학습과 설정 조정을 검토합니다.

6. 새 버전 학습이 필요할 때 [🔗](#)

운영 환경이 바뀌었거나 오탐·미탐이 반복된다면 새 후보를 만들어 비교할 수 있습니다.

1. 모델 버전 영역의 [새 버전 학습](#)을 누릅니다.
2. 모델 버전 이름과 학습 설정을 확인한 뒤 [학습 시작](#)을 누릅니다.
3. 학습이 완료되면 위의 4~5단계와 같이 후보를 확인하고 운영 반영 여부를 결정합니다.



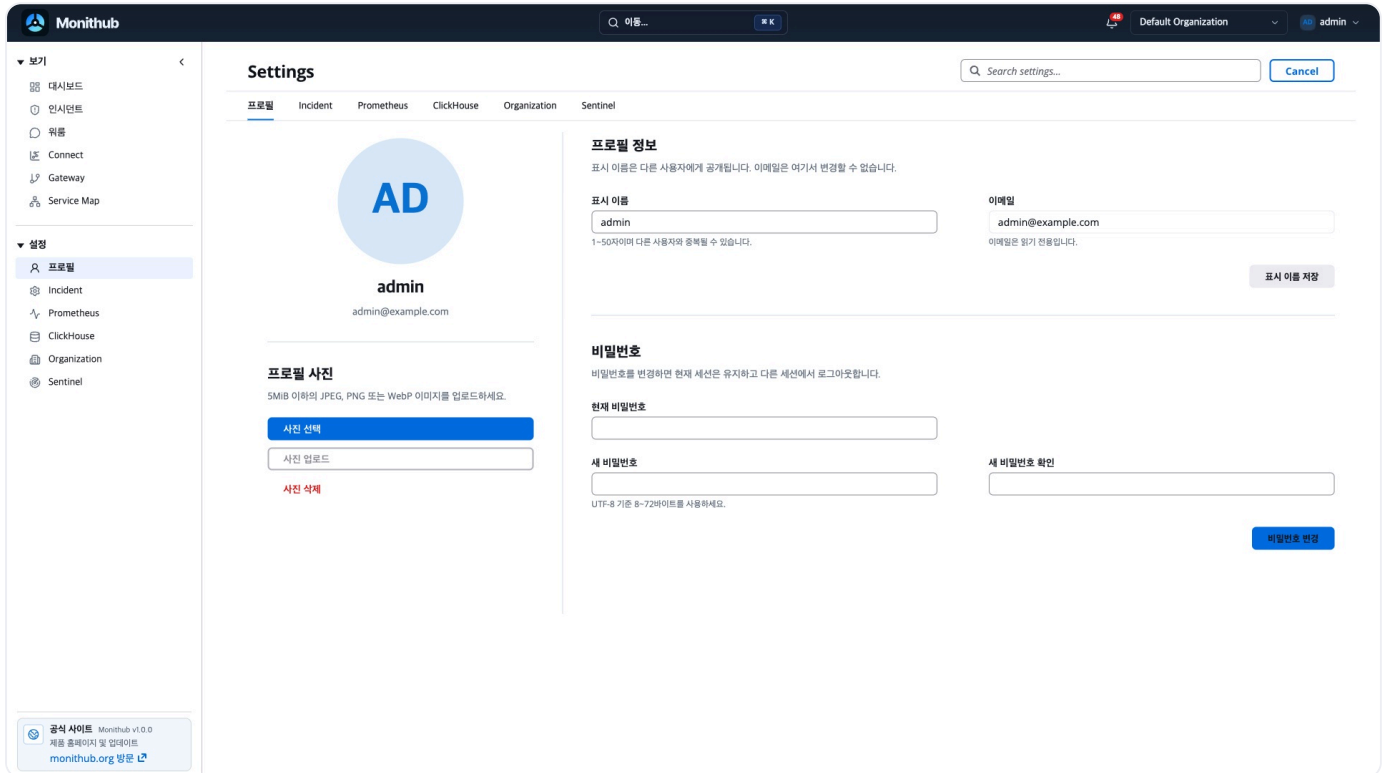
설정값을 어떤 방향으로 조정할지 고민된다면 [학습과 튜닝](#)을, 실행이 실패하면 [확인](#)과 [문제 해결](#)을 참고하세요.

설정

설정에서는 내 프로필과 인시던트, 데이터소스, 조직, Sentinel 설정을 관리합니다.

Profile [🔗](#)

오른쪽 상단의 사용자 이름을 누른 뒤 **프로필**을 선택합니다. 아직 참여한 조직이 없어도 내 프로필은 관리할 수 있습니다.



표시 이름 바꾸기 [🔗](#)

1. **프로필 정보**에서 다른 사용자에게 보여 줄 이름을 입력합니다.
2. 이름은 앞뒤 공백을 제외하고 1~50자로 작성합니다. 다른 사용자와 같은 이름도 사용할 수 있습니다.
3. **표시 이름 저장**을 누릅니다.
4. 변경 완료 안내가 나타나고 오른쪽 상단의 사용자 이름이 바뀌었는지 확인합니다.

이메일은 로그인 계정을 확인하는 값으로, Profile에서는 변경할 수 없습니다.

프로필 사진 바꾸기 [🔗](#)

1. **사진 선택**을 누르고 JPEG, PNG 또는 WebP 이미지를 고릅니다.

2. 선택한 사진의 미리보기와 파일 이름을 확인합니다. 파일 크기는 5MiB 이하여야 합니다.
3. 다른 사진을 고르려면 **선택 취소** 후 다시 선택합니다.
4. **사진 업로드**를 누르고 변경 완료 안내를 확인합니다.

현재 사진을 없애려면 **사진 삭제**를 누릅니다. 사진을 삭제했거나 사진을 불러오지 못한 경우에는 표시 이름의 이니셜이 대신 나타납니다.

비밀번호 바꾸기 [🔗](#)

1. **현재 비밀번호**를 입력합니다.
2. **새 비밀번호**와 **새 비밀번호 확인**에 같은 값을 입력합니다. 새 비밀번호는 UTF-8 기준 8~72바이트이며 현재 비밀번호와 달라야 합니다.
3. **비밀번호 변경**을 누릅니다.
4. 변경 완료 안내에서 로그아웃된 다른 세션 수를 확인합니다.

비밀번호를 변경해도 지금 사용 중인 화면은 로그인 상태를 유지합니다. 다른 브라우저와 기기에서는 로그아웃되므로 새 비밀번호로 다시 로그인해야 합니다. 변경되지 않으면 현재 비밀번호와 새 비밀번호 확인값을 다시 확인합니다.

Settings [🔗](#)

The screenshot displays the Monithub Settings interface. The 'Organization' tab is selected, showing a table of existing organizations. The table has columns for 'Organization', 'Slug', and 'Status'. Two organizations are listed: 'My org' with slug 'my-org' and 'Default Organization' with slug 'default', both marked as 'active'. A '+ Create Organization' button is located in the top right corner of the table area. The left sidebar contains navigation links for various settings categories.

Organization	Slug	Status	
My org	my-org	active	Manage →
Default Organization	default	active	Manage →

오른쪽 상단의 사용자 이름을 누른 뒤 **설정**을 선택합니다. Settings 상단의 검색으로 설정 항목을 찾고, **Cancel**로 설정 화면을 닫습니다. 각 섹션의 변경 사항은 해당 영역에 표시되는 저장 또는 테스트 동작으로 반영합니다.

설정 섹션

설정 검색창으로 항목을 찾거나 탭을 좌우로 이동해 섹션을 선택합니다.

섹션	용도
Profile	표시 이름, 프로필 사진, 비밀번호를 관리합니다.
Incident	인시던트 동작과 표시 기준을 관리합니다.
Prometheus	Prometheus 계열 메트릭 연동 값을 설정합니다.
ClickHouse	로그와 trace 저장소 접근 값을 설정합니다.
Organization	조직과 멤버 관련 설정을 관리합니다.
Sentinel	이상탐지 모델과 튜닝 설정을 관리합니다. 자세한 흐름은 Monithub Sentinel 문서를 봅니다.

Profile 탭의 사진·표시 이름과 비밀번호는 각 영역의 버튼으로 따로 저장합니다. 다른 설정에서 연결 주소를 바꾸기 전에는 현재 조직과 환경 표시가 맞는지 확인합니다.

요금과 구독

요금과 구독에서는 선택한 조직의 현재 요금제와 사용량을 확인하고, 수동 청구 요약과 구독 상태를 관리합니다. 오른쪽 상단의 사용자 이름을 누른 뒤 **요금/구독**을 선택하거나 전체 메뉴의 관리 영역에서 진입합니다.

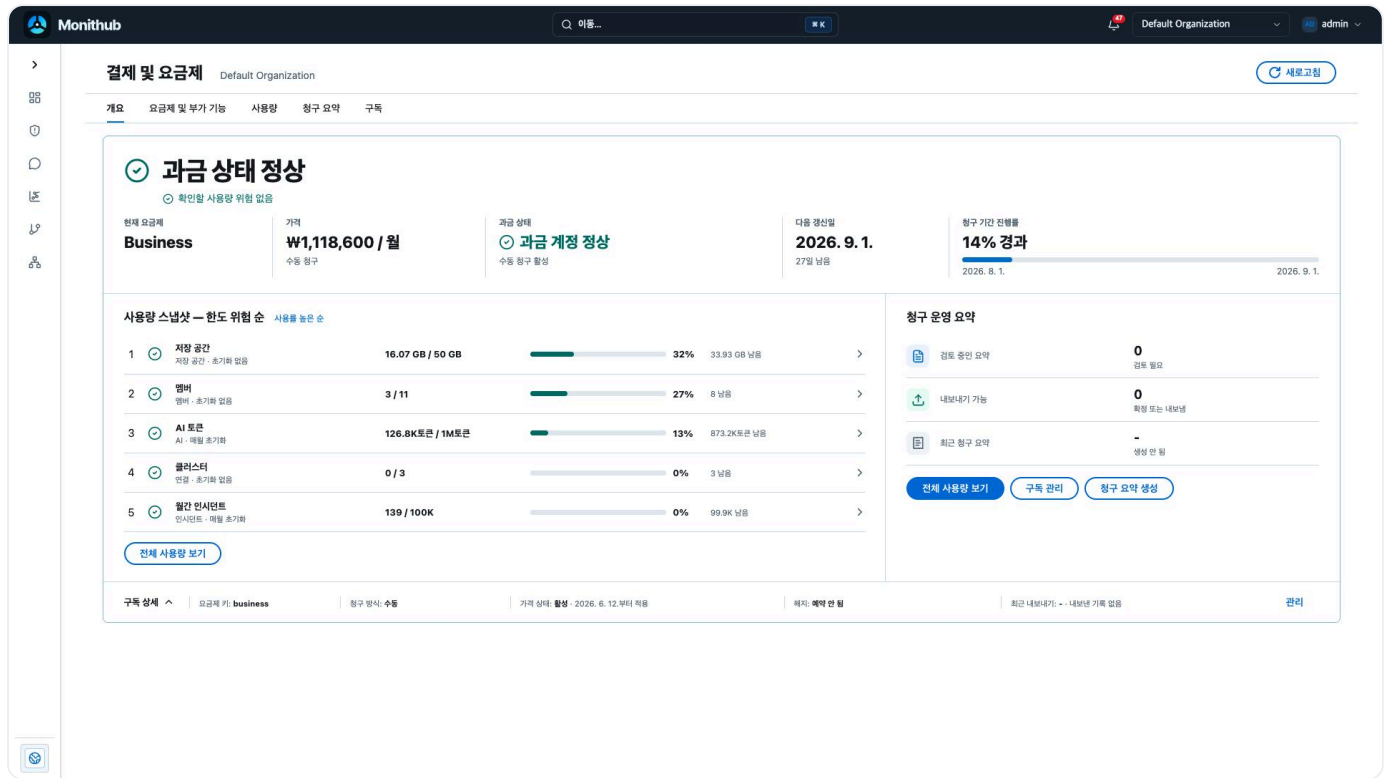
시작하기 전에

- 화면 상단에서 확인하려는 조직을 먼저 선택합니다. 조직을 바꾸면 요금제, 사용량, 청구 요약과 구독 정보도 함께 바뀝니다.
- 조회 권한이 있는 사용자는 모든 탭을 볼 수 있습니다. 요금제 변경, 부가 기능 저장, 청구 요약 처리와 구독 제어에는 `billing.write` 권한이 필요합니다.
- 쓰기 권한이 없으면 화면 상단에 읽기 전용 안내가 나타나고 변경 버튼이 비활성화됩니다.
- 현재 수동 청구 구독만 화면에서 변경할 수 있습니다. 결제 플랫폼 관리 구독은 PoC에서 변경 작업이 비활성화됩니다.
- 화면 값은 마지막으로 불러온 시점의 정보입니다. 작업 전 **새로고침**으로 최신 상태를 확인합니다.

탭 한눈에 보기

탭	확인하거나 수행하는 작업
개요	과금 상태, 현재 요금제, 갱신일, 사용량 위험과 청구 운영 상태를 한 화면에서 확인합니다.
요금제 및 부가 기능	선택 가능한 요금제와 가격을 비교하고, 수동 구독의 요금제 또는 부가 기능 수량을 변경합니다.
사용량	항목별 현재 사용량, 포함 한도, 잔여량과 저장 공간 구성 요소를 확인합니다.
청구 요약	청구 기간 요약을 생성하고 검토·확정한 뒤 CSV 또는 JSON으로 내보냅니다.
구독	현재 구독 상태와 기간을 확인하고 즉시 해지, 기간 종료 시 해지 또는 일시 중지를 실행합니다.

개요



개요는 과금과 사용량 상태를 빠르게 점검하는 첫 화면입니다.

영역 설명

과금 상태	활성 구독 여부와 한도 초과 항목을 종합해 정상 또는 확인 필요 상태를 보여 줍니다.
현재 요금제 · 가격	사용 중인 요금제, 표시 가격, 청구 주기와 수동 또는 결제 플랫폼 관리 여부를 보여 줍니다.
다음 갱신일	현재 청구 기간이 끝나는 날짜와 남은 일수를 보여 줍니다.
청구 기간 진행률	현재 기간의 시작일과 종료일, 경과 비율을 보여 줍니다.
사용량 스냅샷	저장 공간, 멤버, AI 토큰, 클러스터, 월간 인시던트 등을 사용률이 높은 순서로 보여 줍니다. 항목을 선택하면 사용량 탭으로 이동합니다.
청구 운영 요약	검토 중인 요약, 내보내기 가능한 요약과 최근 청구 요약 상태를 보여 줍니다.
구독 상세	요금제 키, 청구 방식, 가격 상태, 해지 예약과 최근 내보내기 정보를 보여 줍니다.

개요 확인 순서 [🔗](#)

1. 상단의 과금 상태가 **정상**인지, 확인할 사용량 위험이 있는지 봅니다.
2. 현재 요금제와 가격, 다음 갱신일이 예상과 같은지 확인합니다.

3. 사용량 스냅샷에서 사용률이 높은 항목과 잔여량을 확인합니다.
4. 검토 중인 청구 요약이 있으면 **청구 요약** 탭으로 이동합니다.
5. 한도가 부족하면 **전체 사용량 보기** 또는 **요금제 및 부가 기능**에서 상세를 확인합니다.

요금제 및 부가 기능 [🔗](#)

요금제 및 부가 기능 Business · 수동 청구

현재 요금제: **Business** (business)

현재 가격: **₩1,118,600 / 월** (월성)

사용량 적합도: **한도 내** (저장 공간, 멤버, 플러그인, 월간 인스턴트, AI 토큰)

다음 갱신일: **2026. 9. 1.** (27일 남음)

요금제 목록 (활성 가격을 선택해 수동 구독을 변경합니다. 요금제 4개)

요금제	가격	청구 주기	선택 상태	작업
Free Forever 세 가지의 제한 없이 무제한 요금제 free_forever	₩0 / 월 월성	월 KRW	선택 가능 활성 가격	선택
Team PoC 팀 요금제 team	₩418,600 / 월 월성	월 KRW	선택 가능 활성 가격	선택
Business (현재) 유료 부가 기능을 지원하는 비즈니스 요금제 business	₩1,118,600 / 월 월성	월 KRW	선택됨 활성 가격	현재
Enterprise 맞춤 제어 요금제 enterprise	₩0 / 수동 월성	수동 KRW	선택 가능 활성 가격	선택

요금제 부가 기능 (필요한 만큼 추가해 요금제 한도를 늘릴 수 있습니다. 3개 사용 가능)

기능	단가	수량	작업
멤버 멤버 한도를 늘립니다.	₩50,000 계약당 / 월	1	사용 중 저장
로그 보관 기간 데이터 보관 기간을 늘립니다.	가격 미정 일당 / 월	0	선택 안 함 저장
저장 공간 저장 공간을 늘립니다.	가격 미정 GB당 / 월	0	선택 안 함 저장

상단 요약은 현재 요금제, 현재 가격, 주요 사용량의 한도 적합도와 다음 갱신일을 보여 줍니다.

요금제 변경 [🔗](#)

1. **요금제 목록**에서 요금제 설명, 가격, 청구 주기, 통화와 선택 상태를 비교합니다.
2. 현재 요금제는 **현재**로 표시되며 다시 선택할 수 없습니다.
3. 다른 활성 가격 행의 **선택**을 누릅니다.
4. 확인창에서 대상 요금제 이름, 요금제 키와 가격을 확인한 뒤 **요금제 변경**을 누릅니다.
5. 변경 후 현재 요금제, 부가 기능, 사용량과 청구 요약이 다시 불러와졌는지 확인합니다.

요금제 변경은 수동 청구 구독이 활성 상태이고 **billing.write** 권한이 있을 때만 사용할 수 있습니다. 표시 가격이 **가격 미정** 또는 맞춤 가격이면 실제 청구 기준을 조직의 계약 정보와 함께 확인합니다.

부가 기능 변경

부가 기능은 현재 요금제와 청구 주기에 사용할 수 있는 항목만 표시됩니다.

항목	의미
기능	늘어나는 한도와 용도를 보여 줍니다.
단가	계정, 일 또는 GB 같은 단위와 월별 가격을 보여 줍니다.
수량	현재 적용할 수량입니다. 0을 저장하면 선택을 해제합니다.
작업	현재 사용 여부를 확인하고 입력한 수량을 저장합니다.

수량은 0 이상의 숫자로 입력합니다. 저장이 끝나면 사용량 한도와 청구 요약도 다시 계산되므로 두 탭에서 변경 결과를 확인합니다. 가격이 아직 정해지지 않은 부가 기능은 계약과 청구 기준이 확정되기 전까지 임의로 사용하지 않습니다.

사용량

Monithub

이동...

Default Organization

admin

결제 및 요금제

개요요금제 및 부가 기능**사용량**청구 요약구독

사용량

2026년 8월 · 최근 계산: 2026. 8. 5. 오후 6:57

저장 공간

16.07 GB / 50 GB

33.93 GB 남음 (67.9%)

멤버

3 / 11

8 남음 (72.7%)

클러스터

0 / 3

3 남음 (100%)

활간 인스턴트

139 / 100K

99.9K 남음 (99.9%)

AI 토큰

126.8K토큰 / 1M토큰

873.2K토큰 남음 (87.3%)

요금제 한도

기능

현재 사용량

포함 한도

잔여량

저장 공간	16.07 GB	50 GB	33.93 GB 남음 (67.9%)
저장 공간 상세			
메트릭	1.52 GB · 경신 2026. 8. 5. 오후 6:45		
로그	0.46 GB · 경신 2026. 8. 5. 오후 6:45		
트레이스	14.05 GB · 경신 2026. 8. 5. 오후 6:45		
워크플로	0.05 MB · 경신 2026. 8. 5. 오전 1:43		
인스턴트	0.03 GB · 경신 2026. 8. 5. 오후 6:45		
채팅	2.93 MB · 경신 2026. 8. 5. 오전 1:43		
멤버	3	11	8 남음 (72.7%)
클러스터	0	3	3 남음 (100%)
활간 인스턴트	139	100K	99.9K 남음 (99.9%)
AI 토큰	126.8K토큰	1M토큰	873.2K토큰 남음 (87.3%)
데이터 보관 기간	해당 없음	30일	소모되지 않음

한도는 최근 사용 가능한 데이터를 기준으로 계산하므로 보고된 사용량과 조금 다를 수 있습니다.

사용량 탭은 현재 월의 주요 할당량과 요금제 한도를 보여 줍니다. 상단의 **최근 계산** 시각으로 데이터 기준 시점을 확인합니다.

할당량 요약

저장 공간, 멤버, 클러스터, 월간 인시던트와 AI 토큰의 현재값, 포함 한도와 잔여 비율을 카드로 비교합니다. 한도를 초과한 항목은 별도 경고로 표시됩니다. AI 토큰처럼 표시 전용인 한도는 초과 표시가 곧 기능 차단을 뜻하지 않을 수 있습니다.

요금제 한도 표

열	설명
기능	사용량 또는 정책 항목의 사용자용 이름입니다.
현재 사용량	마지막 계산 시점에 집계된 값입니다.
포함 한도	현재 요금제와 부가 기능, 조직별 조정을 합친 한도입니다.
잔여량	한도에서 현재 사용량과 예약량을 제외한 값과 비율입니다.

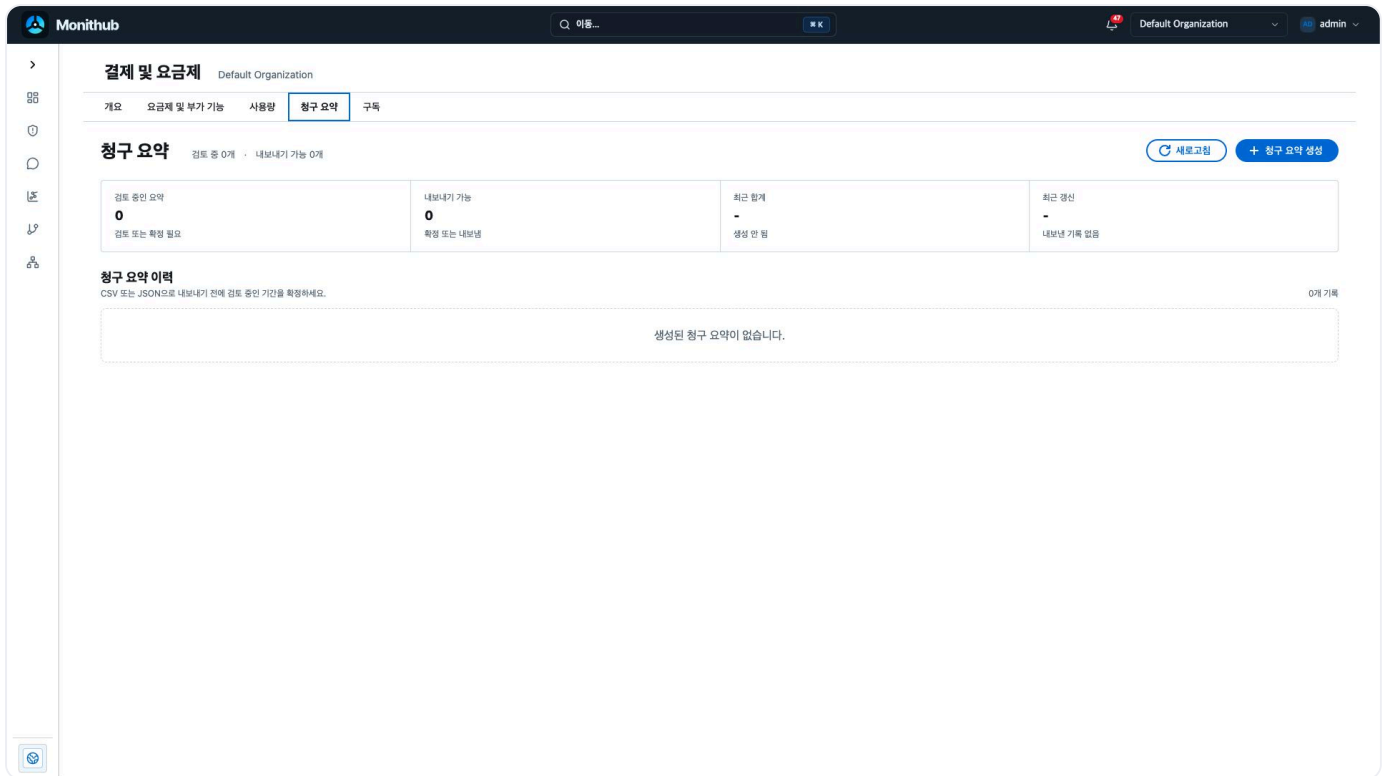
- **기능 검색**으로 필요한 항목만 찾을 수 있습니다.
- 상태 필터에서 전체, 초과, 한도 내 또는 무제한 항목을 좁혀 볼 수 있습니다.
- 1,000 이상의 개수와 토큰 값은 K, M, B 단위로 축약될 수 있습니다.
- **데이터 보관 기간**은 소모되는 사용량이 아니라 적용 정책이므로 현재 사용량은 **해당 없음**, 잔여량은 **소모되지 않음**으로 표시됩니다.

저장 공간 상세

저장 공간 행의 **상세 보기**를 누르면 메트릭, 로그, 트레이스, 워룸, 인시던트와 채팅이 차지하는 공간을 나누어 볼 수 있습니다. 각 값 옆의 갱신 시각은 구성 요소별 마지막 측정 시점입니다. 전체 저장 공간과 구성 요소의 시점이 다를 수 있으므로 장애 조사나 청구 확정 전에는 **새로고침** 후 기준 시각을 함께 확인합니다.

표시 값은 최근 사용 가능한 데이터를 기준으로 계산하므로 다른 화면의 즉시 집계값과 조금 다를 수 있습니다.

청구 요약



청구 요약은 수동 청구를 준비하기 위한 기간별 스냅샷입니다. 상단에서 검토 중인 요약 수, 내보내기 가능한 요약 수, 최근 합계와 최근 갱신 시각을 확인합니다.

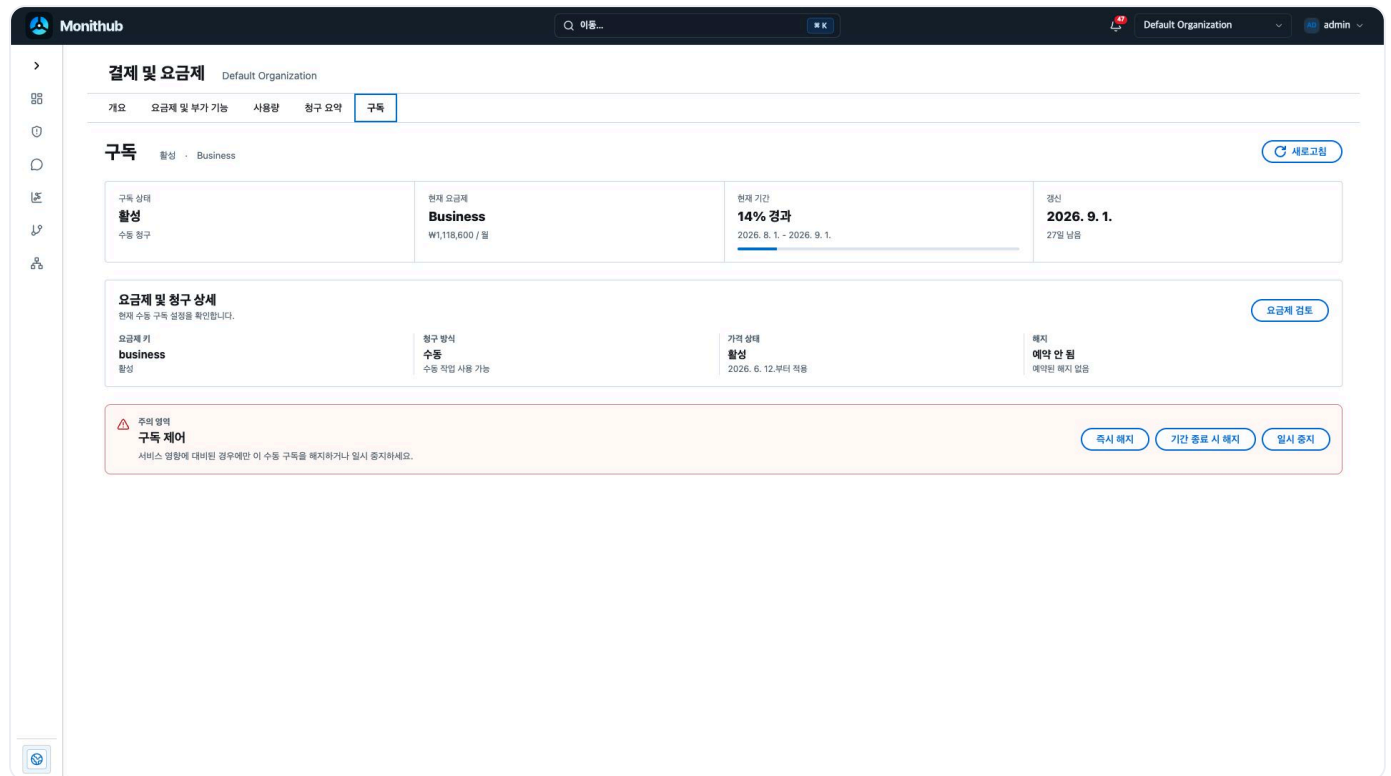
처리 순서 [🔗](#)

1. **청구 요약 생성**을 눌러 현재 구독 기간의 요약을 만듭니다.
2. **검토 중** 상태에서 기간, 기본 금액, 사용량 금액과 통화를 확인합니다.
3. 내용이 맞으면 **확정**을 눌러 더 이상 자동 갱신되지 않는 청구 기준으로 만듭니다.
4. **확정됨** 또는 **내보냄** 상태에서 CSV 또는 JSON을 선택해 내보냅니다.
5. 내려받은 파일의 기간, 통화와 합계를 실제 청구 자료와 대조합니다.

상태	의미와 가능한 작업
검토 중	생성 또는 갱신된 요약입니다. 내용을 확인한 뒤 확정할 수 있습니다.
확정됨	검토가 끝난 요약입니다. CSV 또는 JSON으로 내보낼 수 있습니다.
내보냄	한 번 이상 내보낸 요약입니다. 마지막 형식을 확인하고 다시 내보낼 수 있습니다.
무효	청구 기준에서 제외된 요약입니다. 내보내기 전에 상태를 다시 확인합니다.

검토 중인 요약은 바로 내보낼 수 없습니다. 청구 요약 생성, 확정과 내보내기에는 **billing.write** 권한이 필요합니다. 화면에 기록이 없으면 아직 생성된 청구 요약이 없는 상태입니다.

구독



구독 탭은 현재 구독의 상태와 서비스 영향이 있는 제어 작업을 한곳에 모읍니다.

영역	확인할 내용
구독 개요	활성, 일시 중지 또는 해지 같은 상태와 수동 또는 결제 플랫폼 관리 여부를 확인합니다.
현재 요금제	요금제 이름, 가격과 청구 주기를 확인합니다.
현재 기간	시작일, 종료일과 경과 비율을 확인합니다.
갱신	다음 기간 시작일과 남은 일수를 확인합니다.
요금제 및 청구 상세	요금제 키, 청구 방식, 가격 상태, 적용일과 해지 예약을 확인합니다.

구독 제어

구독 제어는 수동 청구 구독과 [billing.write](#) 권한이 있는 사용자에게만 활성화됩니다.

- **즉시 해지:** 구독을 바로 해지 상태로 전환합니다. 서비스 영향과 필요한 데이터 보관 조치를 먼저 확인합니다.
- **기간 종료 시 해지:** 현재 기간 종료 시점으로 해지를 예약합니다. 이미 예약된 경우 같은 작업을 다시 실행할 수 없습니다.

- **일시 중지:** 구독을 일시 중지 상태로 전환합니다. 기능 변경 작업이 제한될 수 있으므로 운영 담당자와 영향 범위를 확인합니다.

해지되면 추가 해지나 일시 중지 작업을 실행할 수 없고, 일시 중지 상태에서는 일시 중지 버튼이 비활성화됩니다. 이 작업들은 서비스에 영향을 줄 수 있으므로 화면의 현재 조직, 구독 상태와 청구 기간을 다시 확인한 뒤 실행합니다.

정기 확인 흐름

1. **개요**에서 과금 상태와 사용량 위험을 확인합니다.
2. **사용량**에서 한도에 가까운 항목과 최근 계산 시각을 확인합니다.
3. 필요한 경우 **요금제 및 부가 기능**에서 한도를 조정합니다.
4. 청구 시점에 **청구 요약**을 생성하고 검토·확정·내보내기 순서로 처리합니다.
5. **구독**에서 다음 갱신일과 해지 예약 여부를 마지막으로 확인합니다.

문서의 화면 값과 날짜는 예시입니다. 실제 판단에는 현재 선택한 조직의 최신 화면과 계약 정보를 사용합니다.

Monithub K8S Operator 설치

Monithub K8S Operator는 고객 Kubernetes 클러스터 안에서 Monithub 수집 컴포넌트를 설치하고 관리합니다. 기본 설치에는 Monithub K8S Operator, OpenTelemetry Collector gateway, Flow Agent, [MonithubAgent](#) 리소스를 함께 구성해 서비스 맵과 네트워크 흐름 데이터를 Monithub로 보냅니다.

이 가이드는 처음 설치하는 사용자가 준비해야 할 값, 실행해야 할 명령, 설치 후 확인할 화면을 순서대로 설명합니다.

 Note

현재 고객용 기본 설치 경로는 Helm OCI chart입니다. Kustomize 배포 파일은 Helm을 쓰기 어려운 환경의 보조 경로로만 사용합니다.

1. 값 준비

Monithub Connect에서 cluster ID, connect ID, token, OTLP endpoint, registry 인증 정보를 확인합니다.

2. 클러스터 설치

Secret과 values 파일을 만든 뒤 Helm chart로 Monithub K8S Operator와 기본 Agent를 설치합니다.

3. 데이터 확인

Rollout, Agent condition, Monithub 서비스 맵을 순서대로 확인해 수집 상태를 검증합니다.

설치 전 준비

항목	필요 내용
Kubernetes 권한	CRD, ClusterRole, ServiceAccount, Deployment, DaemonSet을 생성할 수 있는 권한
로컬 도구	kubectl , Helm 3.8 이상, 설치 대상 클러스터 context
네트워크	클러스터에서 registry.monithub.org 와 Monithub OTLP endpoint로 나가는 egress
Monithub 값	clusterId , connect.id , connect.orgId , connect.tenantId , Connect token
Registry 인증	Monithub가 발급한 registry pull username과 token

⚠ Caution

Connect token과 registry pull token은 문서, Git 저장소, 이슈, 메신저에 그대로 남기지 않습니다. 설치 명령에는 placeholder를 사용하고 실제 값은 Secret으로만 주입합니다.

1. Monithub Connect 값 확인 [🔗](#)

Monithub에서 Kubernetes 연결을 만들면 설치에 필요한 값이 발급됩니다. 설치 전에 아래 값을 한곳에 모아 둡니다.

값	설명	예시
<code>clusterId</code>	Monithub에서 클러스터를 식별하는 이름입니다. 설치 후 바꾸면 기존 데이터와 새 데이터가 분리될 수 있습니다.	<code>customer-prod-a</code>
<code>connect.id</code>	Connect 설정과 telemetry를 연결하는 ID입니다.	<code>customer-connect-a</code>
<code>connect.orgId</code>	Monithub 조직 ID입니다.	<code>org_customer_a</code>
<code>connect.tenantId</code>	tenant routing에 사용할 ID입니다. 별도 값이 없으면 보통 <code>orgId</code> 와 같은 값을 사용합니다.	<code>org_customer_a</code>
<code>endpoint.otlpHttp</code>	Collector가 데이터를 보낼 OTLP HTTP endpoint 입니다.	<code>https://otel.monithub.org</code>
Connect token	Collector export 인증에 사용하는 token입니다.	<code><MONITHUB_CONNECT_TOKEN></code>
Registry pull 정보	Monithub K8S Operator와 Flow Agent 이미지를 pull하기 위한 registry 인증 정보입니다.	<code><MONITHUB_REGISTRY_PULL_USERNAME></code>

2. 클러스터 context 확인 [🔗](#)

설치 명령을 실행하기 전에 현재 `kubectl`이 올바른 클러스터를 바라보는지 확인합니다.

● ● ●

```
kubectl config current-context
kubectl get nodes
```

운영 클러스터와 개발 클러스터를 함께 다룬다면 context 이름, namespace, `clusterId`가 서로 맞는지 한번 더 확인합니다.

3. Namespace와 Secret 생성 [🔗](#)

Monithub K8S Operator는 기본적으로 `monithub-system` namespace에 설치합니다. 민감한 값은 Helm values 파일에 직접 쓰지 않고 Kubernetes Secret으로 분리합니다.

```
kubectl create namespace monithub-system --dry-run=client -o yaml | kubectl apply -f -

kubectl -n monithub-system create secret generic monithub-auth \
  --from-literal=token='<MONITHUB_CONNECT_TOKEN>' \
  --dry-run=client -o yaml | kubectl apply -f -

kubectl -n monithub-system create secret docker-registry registry-credentials \
  --docker-server=registry.monithub.org \
  --docker-username='<MONITHUB_REGISTRY_PULL_USERNAME>' \
  --docker-password='<MONITHUB_REGISTRY_PULL_TOKEN>' \
  --dry-run=client -o yaml | kubectl apply -f -
```

생성 후 Secret 이름만 확인합니다. Secret 값을 출력할 필요는 없습니다.

```
kubectl -n monithub-system get secret monithub-auth registry-credentials
```

4. values 파일 작성 [🔗](#)

로컬에 `monithub-values.yaml` 파일을 만들고 Monithub에서 발급받은 값으로 채웁니다.

clusterId: `customer-prod-a`

connect:

id: `customer-connect-a`

orgId: `org_customer_a`

tenantId: `org_customer_a`

endpoint:

otlpHttp: `https://otel.monithub.org`

authSecret:

name: `monithub-auth`

key: `token`

imagePullSecrets:

- name: `registry-credentials`

telemetryPreset: `service-map-lite`

telemetry preset 선택 [🔗](#)

preset	수집 범위	권장 상황
<code>service-map-lite</code>	Monithub K8S Operator, Collector gateway, Flow Agent 중심의 서비스 맵 기본 수집	첫 설치, 비용과 리소스를 보수적으로 시작할 때
<code>kubernetes-basic</code>	<code>service-map-lite</code> 에 host metrics, kubelet stats, k8s cluster 수집을 추가	클러스터 상태와 워크로드 리소스 신호까지 함께 보고 싶을 때

처음 설치할 때는 `service-map-lite`로 시작한 뒤, 데이터가 정상 유입되는 것을 확인하고 필요할 때 `kubernetes-basic`으로 확장하는 방식을 권장합니다.

5. Helm registry 로그인 [🔗](#)

Monithub chart는 OCI registry로 배포됩니다.

```
helm registry login registry.monithub.org
```

로그인에는 Monithub에서 발급받은 registry 계정을 사용합니다. 클러스터 내부 이미지 pull은 앞에서 만든 `registry-credentials` Secret이 담당합니다.

6. Monithub K8S Operator 설치 [🔗](#)

아래 명령은 `v0.1.2` chart를 기준으로 Monithub K8S Operator를 설치하거나 기존 설치를 업데이트합니다.

```
helm upgrade --install monithub-operator
oci://registry.monithub.org/monithub/charts/monithub-operator \
-n monithub-system \
--version 0.1.2 \
-f monithub-values.yaml
```

설치가 시작되면 Helm release와 Kubernetes 리소스를 확인합니다.

```
helm -n monithub-system status monithub-operator
kubectl -n monithub-system get deploy,ds,svc,cm
```

7. 설치 상태 확인 [🔗](#)

먼저 Monithub K8S Operator와 Collector, Flow Agent가 정상적으로 rollout되는지 확인합니다.

```
kubectl -n monithub-system rollout status deploy/monithub-operator
kubectl -n monithub-system rollout status deploy/monithub-collector
kubectl -n monithub-system rollout status daemonset/monithub-flow-agent
```

그다음 `MonithubAgent` 상태를 확인합니다.

```
kubectl -n monithub-system get monithubagent monithub
```

```
kubectl -n monithub-system get monithubagent monithub \
-o jsonpath='{range .status.conditions[*]}.{type}{"\t"}{.status}{"\t"}{.reason}{"\n"}{end}'
```

주요 condition은 아래처럼 읽습니다.

condition	정상 상태	의미
CollectorAvailable	True	Monithub Collector gateway가 실행 중입니다.
AgentAvailable	True	Flow Agent DaemonSet이 사용 가능한 노드에 배포되었습니다.
EBPFSupported	True 또는 Unknown	eBPF 기반 네트워크 수집 상태입니다. 설치 직후에는 metrics scrape 전까지 Unknown일 수 있습니다.
DataExporting	True 또는 Unknown	Monithub로 데이터 export가 관찰되었거나, 트래픽을 기다리는 상태입니다.
TelemetryGuardrails	True	고비용 telemetry 옵션이 보호 정책에 맞게 적용되었습니다.

Tip

`DataExporting`이 `AwaitingNetworkTraffic` 상태라면 설치된 것은 맞지만 관찰할 네트워크 트래픽이 아직 적을 수 있습니다. 클러스터 안에서 실제 서비스 요청을 몇 분 정도 발생시킨 뒤 다시 확인합니다.

8. Monithub 화면에서 확인

클러스터 리소스가 Ready 상태가 되면 Monithub에서 아래 순서로 확인합니다.

1. Connect 화면에서 해당 Kubernetes 연결이 활성 상태인지 확인합니다.
2. Service Map에서 `clusterId`에 해당하는 노드와 서비스가 나타나는지 확인합니다.
3. 대시보드에서 최근 메트릭, 로그, 트레이스 또는 network edge 데이터가 들어오는지 확인합니다.
4. 데이터가 보이지 않으면 설치한 `clusterId`, `connect.id`, `orgId`, OTLP endpoint가 values 파일과 Connect 화면에서 같은지 비교합니다.

첫 데이터가 보이기까지는 클러스터 트래픽과 수집 주기에 따라 몇 분이 걸릴 수 있습니다.

자주 바꾸는 설정 [🔗](#)

namespace 수집 범위 제한 [🔗](#)

특정 namespace만 네트워크 topology에 포함하려면 `topology.network.includeNamespaces`를 사용합니다.

```
topology:
  network:
    includeNamespaces:
      - default
      - payments
```

특정 namespace를 제외하려면 `excludeNamespaces`를 사용합니다. 기본값은 `kube-system`, `monithub-system`을 제외합니다.

```
topology:
  network:
    excludeNamespaces:
      - kube-system
      - monithub-system
      - monitoring
```

Kubernetes 기본 telemetry 켜기 [🔗](#)

노드, kubelet, cluster 상태를 함께 보고 싶다면 preset을 바꿉니다.

```
telemetryPreset: kubernetes-basic
```

적용은 같은 Helm upgrade 명령을 다시 실행하면 됩니다.

```
helm upgrade monithub-operator oci://registry.monithub.org/monithub/charts/monithub-operator \
-n monithub-system \
--version 0.1.2 \
-f monithub-values.yaml
```

application trace 수집 켜기 [🔗](#)

애플리케이션 HTTP 또는 gRPC trace를 eBPF 기반으로 수집하려면 application topology를 명시적으로 켭니다.

```
topology:
  application:
    enabled: true
    provider: obi
    openPorts: 80,443,3000,5000,8000-8999
    traceSampleRate: 0.01
```

처음부터 모든 옵션을 켜기보다, 서비스 맵 기본 수집이 안정적으로 동작하는 것을 확인한 뒤 단계적으로 확장하는 편이 운영 리스크가 낮습니다.

업데이트와 롤백 [🔗](#)

values 파일을 수정하거나 새 chart 버전을 적용할 때는 같은 release 이름으로 upgrade합니다.

```
helm upgrade monithub-operator oci://registry.monithub.org/monithub/charts/monithub-operator \
-n monithub-system \
--version 0.1.2 \
-f monithub-values.yaml
```

이전 release로 되돌릴 때는 history를 확인한 뒤 원하는 revision으로 rollback합니다.

```
helm -n monithub-system history monithub-operator
helm -n monithub-system rollback monithub-operator 1
```

CRD schema가 바뀌는 릴리즈는 Helm upgrade 전에 CRD를 먼저 적용해야 합니다.



```
helm show crds oci://registry.monithub.org/monithub/charts/monithub-operator \
--version 0.1.2 | kubectl apply --server-side -f -
```

제거 [🔗](#)

Monithub K8S Operator를 제거할 때는 Helm release를 먼저 삭제합니다.



```
helm -n monithub-system uninstall monithub-operator
```

더 이상 같은 클러스터에서 Monithub K8S Operator를 사용하지 않는다면 Secret과 namespace도 정리할 수 있습니다.



```
kubectl -n monithub-system delete secret monithub-auth registry-credentials
kubectl delete namespace monithub-system
```

CRD는 다른 `MonithubAgent` 또는 `MonithubInstrumentationPolicy` 리소스가 없는지 확인한 뒤 삭제합니다.



```
kubectl get monithubagent --all-namespaces
kubectl get monithubinstrumentationpolicy --all-namespaces
kubectl delete crd monithubagents.monithub.io monithubinstrumentationpolicies.monithub.io
```

Troubleshooting [🔗](#)

증상	확인할 것	조치
<code>ImagePullBackOff</code>	<code>registry-credentials</code> Secret 이름과 registry 계정	Secret을 다시 만들고 Helm values의 <code>imagePullSecrets</code> 를 확인합니다.
<code>MonithubAgent</code> 가 생성되지 않음	Helm values의 <code>monithubAgent.create</code> 값, CRD 설치 여부	<code>helm status</code> , <code>kubectl get crd monithubagents.monithub.io</code> 를 확인합니다.
<code>InvalidSpec</code> condition	<code>clusterId</code> , <code>endpoint.otlpHttp</code> 또는 <code>endpoint.otlpGrpc</code> 값	values 파일의 필수 값을 채운 뒤 Helm upgrade를 다시 실행합니다.
Collector는 Ready지만 데이터가 없음	Connect token, OTLP endpoint, egress 정책	<code>monithub-auth</code> Secret과 방화벽, proxy, endpoint 오차를 확인합니다.
Flow Agent DaemonSet이 일부 노드에서 Ready가 아님	노드 Ready 상태, 커널/eBPF 지원, taint	<code>kubectl get nodes</code> , Agent pod event, <code>EBPFSupported</code> condition을 확인합니다.
<code>DataExporting</code> 이 <code>Unknown</code>	실제 네트워크 트래픽 또는 metrics scrape 대기	서비스 요청을 발생시키고 몇 분 뒤 <code>MonithubAgent</code> condition을 다시 확인합니다.
Kubernetes telemetry collector가 없음	<code>telemetryPreset</code> 값	기본값 <code>service-map-lite</code> 에서는 host/cluster collector가 생성되지 않습니다.

진단 로그는 아래 순서로 확인합니다.

```

kubectl -n monithub-system logs deploy/monithub-operator --tail=100
kubectl -n monithub-system logs deploy/monithub-collector --tail=100
kubectl -n monithub-system logs daemonset/monithub-flow-agent --tail=100
kubectl -n monithub-system describe monithubagent monithub

```

다음 단계 [🔗](#)

- [Metrics, Logs 및 Traces 필터링](#)에서 불필요한 telemetry를 backend 전송 전에 제외하는 방법을 확인합니다.
- [Connect](#)에서 연결 생성 화면을 확인합니다.
- [서비스 맵](#)에서 서비스 간 연결 상태를 확인합니다.
- [대시보드](#)에서 운영 지표를 해석하는 방법을 확인합니다.

Metrics, Logs 및 Traces 필터링

Telemetry 필터링은 반복적으로 발생하지만 분석에는 거의 사용하지 않는 metric, log, span을 Monithub로 보내기 전에 제외하는 기능입니다. Kubernetes 클러스터 안의 Monithub Collector gateway에서 필터를 적용하므로, 조건에 맞는 데이터는 중앙 Collector와 저장소까지 전송되지 않습니다.

대표적으로 다음 데이터를 줄일 때 사용합니다.

- 상태 확인용 endpoint에서 반복되는 정상 access log
- 개발 환경이나 테스트 workload에서 생성되는 불필요한 log
- 사용하지 않는 runtime metric 전체
- 하나의 metric 안에서 분석 가치가 낮은 특정 datapoint
- health check처럼 짧고 반복적인 정상 span

Caution

필터에 걸린 데이터는 Monithub 저장소에 기록되지 않으며 나중에 복구할 수 없습니다. 처음에는 범위가 좁은 규칙 하나로 시작하고, 제외 대상과 비교용 데이터를 함께 확인한 뒤 범위를 넓히세요.

동작 방식

필터는 Monithub Collector gateway가 Kubernetes와 Monithub 식별 속성을 보강한 다음, batch와 exporter를 실행하기 전에 적용됩니다.

애플리케이션 또는 수집 Agent
→ Monithub Collector gateway
→ Kubernetes/Monithub 속성 보강
→ Metrics/Logs/Traces 필터
→ 민감 속성 제거와 batch
→ Monithub backend

사용자는 OpenTelemetry Transformation Language(OTTL)를 직접 작성하지 않습니다.

[MonithubAgent](#)에 필드, 연산자, 값의 타입을 지정하면 Monithub K8S Operator가 Collector 설정으로 변환합니다.

필터는 **제외(drop) 전용** 방식입니다. 규칙에 일치하는 데이터만 제외하고, 일치하지 않는 데이터는 그대로 통과시킵니다. 규칙을 설정하지 않거나 `rules: []`로 두면 모든 데이터가 통과합니다.

시작하기 전에 [🔗](#)

다음 조건을 확인합니다.

- Metrics, Logs 및 Traces 필터링 API가 포함된 Monithub K8S Operator chart를 사용해야 합니다.
- Monithub K8S Operator는 gateway 모드의 Collector를 사용해야 합니다.
- 기존 Helm values 파일을 수정할 수 있어야 합니다.
- CRD schema가 chart 버전과 같아야 합니다.

Helm은 upgrade할 때 `crds/` 디렉터리의 기존 CRD를 자동으로 갱신하지 않습니다. 필터링 기능이 포함된 버전으로 처음 업데이트한다면 Monithub K8S Operator를 upgrade하기 전에 CRD부터 적용합니다.

```
helm show crds oci://registry.monithub.org/monithub/charts/monithub-operator \
--version <CHART_VERSION> | kubectl apply --server-side -f -
```

Note

`<CHART_VERSION>`에는 Monithub가 안내한 필터링 지원 버전을 입력합니다. 기존 설치의 chart 버전과 새 CRD 버전이 다르면 `unknown field` 또는 admission validation 오류가 발생할 수 있습니다.

권장 설정 예제 [🔗](#)

기존 `monithub-values.yaml`에 필요한 데이터 종류별 필터 규칙을 추가합니다. 다음 예제는 Metrics, Logs, Traces를 함께 설정하지만, 사용하지 않을 데이터 종류는 생략해도 됩니다.

telemetry:

metrics:

filtering:

rules:

go.runtime.* metric 전체를 제외합니다.

- name: drop-go-runtime-metrics

scope: metric

match:

all:

- field: metricName

operator: startsWith

value:

string: go.runtime.

container.cpu.usage의 idle datapoint만 제외합니다.

- name: drop-idle-container-points

scope: datapoint

match:

all:

- field: metricName

operator: equals

value:

string: container.cpu.usage

- field: datapointAttribute

key: container.state

operator: equals

value:

string: idle

logs:

filtering:

rules:

WARN 미만이면서 health/ready endpoint에서 발생한 log를 제외합니다.

- name: drop-health-access-logs

match:

all:

- field: severityNumber

operator: lessThan

value:

integer: 13

```
- field: logAttribute
  key: http.route
  operator: matches
  value:
    string: ^/(health|ready)$
```

body에 반복적인 probe 문구가 포함된 log를 제외합니다.

```
- name: drop-probe-body-noise
  match:
    all:
      - field: body
        operator: contains
        value:
          string: probe completed
```

traces:

spanFiltering:

rules:

server span 중 health/ready endpoint 호출만 제외합니다.

```
- name: drop-health-check-spans
  match:
    all:
      - field: spanKind
        operator: equals
        value:
          string: server
      - field: spanAttribute
        key: http.route
        operator: matches
        value:
          string: ^/(health|ready)$
```

예제의 metric 이름, route, attribute key는 실제 계측 데이터에 맞게 바뀌어야 합니다. 먼저 Monithub에서 제외할 데이터 한 건을 열어 정확한 이름, 값, 타입을 확인하세요.

설정을 적용합니다.

```
helm upgrade monithub-operator oci://registry.monithub.org/monithub/charts/monithub-
operator \
  -n monithub-system \
  --version <CHART_VERSION> \
  -f monithub-values.yaml
```

Tip

Helm이 관리하는 `MonithubAgent`를 `kubectl edit`로 직접 수정하면 다음 Helm upgrade에서 값이 되돌아갈 수 있습니다. 운영 설정은 values 파일을 기준으로 관리하세요.

규칙을 읽는 방법

한 규칙 안의 조건은 모두 만족해야 합니다

`match.all`에 조건이 여러 개 있으면 모든 조건이 참일 때만 데이터가 제외됩니다.

다음 규칙은 `http.route=/health`인 log를 모두 버리는 것이 아닙니다. severity가 WARN 미만이라는 조건까지 함께 만족해야 합니다.

```
- name: drop-health-access-logs
match:
  all:
    - field: severityNumber
      operator: lessThan
      value:
        integer: 13
    - field: logAttribute
      key: http.route
      operator: equals
      value:
        string: /health
```

log	결과	이유
INFO, <code>/health</code>	제외	두 조건을 모두 만족합니다.
ERROR, <code>/health</code>	통과	severity 조건을 만족하지 않습니다.
INFO, <code>/orders</code>	통과	route 조건을 만족하지 않습니다.

여러 규칙은 하나만 일치해도 제외됩니다 [🔗](#)

`rules` 아래의 규칙은 서로 OR 관계입니다. `drop-health-access-logs` 또는 `drop-probe-body-noise` 중 하나에만 일치해도 해당 log는 제외됩니다.

타입이 다르면 안전하게 통과합니다 [🔗](#)

attribute 값의 실제 타입과 규칙의 `value` 타입이 다르면 그 조건은 일치하지 않는 것으로 처리합니다. 예를 들어 실제 `http.status_code`가 integer인데 규칙을 `string: "200"`으로 작성했다면 해당 데이터는 제외되지 않습니다.

이 동작은 잘못된 타입 지정으로 예상보다 많은 데이터가 사라지는 것을 막기 위한 안전 통과(fail-open) 정책입니다. 필터가 예상대로 적용되지 않으면 먼저 실제 attribute의 이름과 타입을 확인하세요.

Metric과 datapoint의 차이 [🔗](#)

하나의 metric에는 attribute 조합이 서로 다른 여러 datapoint가 들어갈 수 있습니다.

```
container.cpu.usage
├── container.name=api, state=active
├── container.name=worker, state=active
└── container.name=batch, state=idle
```

scope: metric [🔗](#)

metric 이름이나 타입이 일치하면 metric 전체와 그 안의 모든 datapoint를 제거합니다.


```
- name: drop-runtime-metric
scope: metric
match:
  all:
    - field: metricName
      operator: equals
      value:
        string: process.runtime.jvm.memory.usage
```

scope: datapoint [🔗](#)

metric은 유지하면서 조건에 일치하는 datapoint만 제거합니다.

```
- name: drop-idle-points
scope: datapoint
match:
  all:
    - field: datapointAttribute
      key: state
      operator: equals
      value:
        string: idle
```

모든 datapoint가 제거되면 값이 없는 빈 metric도 함께 제거됩니다. Metric scope 규칙은 datapoint scope 규칙보다 먼저 평가됩니다.

Trace 필터링에서 알아둘 점 [🔗](#)

Trace는 하나의 요청이 여러 서비스를 거치는 전체 흐름이고, span은 그 안의 개별 처리 단계입니다. Trace 필터는 trace 전체가 아니라 조건에 일치하는 **개별 span**을 제거합니다. OBI/Beyla, OpenTelemetry 자동 계측, 외부 SDK가 gateway로 보낸 span에 같은 규칙이 적용됩니다.

다음 예제는 `backend` 서비스의 이름이 `DB`인 span만 제거합니다.

```

telemetry:
  traces:
    spanFiltering:
      rules:
        - name: drop-backend-db-spans
          match:
            all:
              - field: spanName
                operator: equals
                value:
                  string: DB
              - field: resourceAttribute
                key: service.name
                operator: equals
                value:
                  string: backend

```

span	결과	이유
service.name=backend, 이름 DB	제외	두 조건을 모두 만족합니다.
service.name=frontend, 이름 DB	통과	service 조건을 만족하지 않습니다.
service.name=backend, 이름 HTTP GET	통과	span name 조건을 만족하지 않습니다.

실행 시간으로 필터링할 때는 서비스와 span 이름을 함께 지정해 범위를 좁히는 편이 안전합니다. 다음 규칙은 `orders-worker` 서비스에서 발생한 정상 `cache.poll` span 중 5ms보다 짧은 것만 제외합니다.

- name: `drop-fast-cache-poll-spans`

match:

all:

- field: `resourceAttribute`

key: `service.name`

operator: `equals`

value:

string: `orders-worker`

- field: `spanName`

operator: `equals`

value:

string: `cache.poll`

- field: `statusCode`

operator: `notEquals`

value:

string: `error`

- field: `duration`

operator: `lessThan`

value:

duration: `5ms`

Caution

Parent span만 제거하면 child span은 남아 trace가 중간에서 끊긴 것처럼 보일 수 있습니다.

Client span과 server span 중 한쪽만 제거하면 Service Map의 호출 관계, 지연 시간, 오류 집계에도 영향을 줄 수 있습니다. Trace 규칙은 반복되는 health check처럼 영향 범위가 명확한 span부터 적용하세요.

Span 필터링과 sampling은 다릅니다

기능	판단 단위	사용하는 때
Head sampling	trace ID 비율	수집 시작 지점에서 전체 수집량을 일정 비율로 줄일 때
Span 필터링	개별 span 내용	이름, 상태, 시간, attribute가 특정 조건인 span만 제외할 때
Tail sampling	완성된 trace	오류 또는 고지연 trace 전체를 선택할 때. 현재 필터 기능에는 포함되지 않습니다.

Head sampling은 span이 gateway에 도착하기 전에 실행될 수 있습니다. 필터 규칙에서 오류 span을 제외하지 않았더라도 앞선 수집 단계에서 이미 선택되지 않은 오류 trace를 복구할 수는 없습니다.

일부 단순한 `url.path equals` 규칙은 지원되는 OBI/Beyla 환경에서 수집 지점에도 자동 적용되어 node와 gateway 사이의 트래픽을 줄일 수 있습니다. 이 최적화가 적용되지 않더라도 gateway Collector가 모든 Trace 규칙을 계속 실행합니다.

지원하는 필드 [🔗](#)

Metrics [🔗](#)

scope	field	설명	key 필요 여부
metric	metricName	OpenTelemetry metric 이름	필요 없음
metric	metricType	gauge, sum, histogram, exponentialHistogram, summary	필요 없음
metric	scopeName	metric을 만든 instrumentation scope 이름	필요 없음
metric	resourceAttribute	서비스, Pod, 환경 등 resource attribute	필요
datapoint	metricName	datapoint가 속한 metric 이름	필요 없음
datapoint	scopeName	instrumentation scope 이름	필요 없음
datapoint	datapointAttribute	datapoint 자체의 attribute	필요
datapoint	resourceAttribute	datapoint가 속한 resource attribute	필요

`metricType`은 `equals`와 `notEquals`만 지원합니다. 숫자로 된 datapoint 값 자체를 비교하는 기능은 지원하지 않습니다.

Logs [🔗](#)

field	설명	key 필요 여부
severityNumber	OpenTelemetry severity number	필요 없음
severityText	INFO, WARN 같은 severity 문자열	필요 없음
body	log message body	필요 없음
scopeName	로그를 만든 instrumentation scope 이름	필요 없음
logAttribute	log record에 포함된 attribute	필요
resourceAttribute	서비스, Pod, 환경 등 resource attribute	필요

body는 실제 값이 문자열일 때만 비교합니다. 구조화된 map, 배열, 숫자 body는 문자열 규칙과 일치하지 않으므로 통과합니다.

Severity number 참고 [🔗](#)

범위	수준
0	UNSPECIFIED
1~4	TRACE
5~8	DEBUG
9~12	INFO
13~16	WARN
17~20	ERROR
21~24	FATAL

예를 들어 lessThan: 13에 해당하는 설정은 TRACE, DEBUG, INFO를 대상으로 하며 WARN 이상은 통과 시킵니다. severityNumber 비교 값은 0부터 24까지만 사용할 수 있습니다.

Traces [🔗](#)

field	설명	값과 연산자	key 필요 여부
<code>spanName</code>	span 이름	string 비교와 pattern	필요 없음
<code>spanKind</code>	span 역할	<code>unspecified</code> , <code>internal</code> , <code>server</code> , <code>client</code> , <code>producer</code> , <code>consumer</code> 의 <code>equals</code> , <code>notEquals</code>	필요 없음
<code>statusCode</code>	OpenTelemetry span 상태	<code>unset</code> , <code>ok</code> , <code>error</code> 의 <code>equals</code> , <code>notEquals</code>	필요 없음
<code>duration</code>	span 종료 시각과 시작 시각의 차이	양수 duration의 숫자 비교	필요 없음
<code>scopeName</code>	instrumentation scope 이름	string 비교와 pattern	필요 없음
<code>spanAttribute</code>	span 자체의 attribute	typed 비교	필요
<code>resourceAttribute</code>	서비스, Pod, 환경 등 resource attribute	typed 비교	필요

`statusCode: unset`은 성공을 뜻하지 않습니다. 오류 span을 보존하려는 규칙이라면 실제 계측 라이브러리가 status를 어떻게 기록하는지 먼저 확인하세요.

지원하는 연산자 [🔗](#)

operator	의미	사용할 수 있는 값
<code>exists</code>	attribute가 존재함	<code>value</code> 를 작성하지 않음
<code>notExists</code>	attribute가 존재하지 않음	<code>value</code> 를 작성하지 않음
<code>equals</code>	값이 같음	string, integer, double, boolean
<code>notEquals</code>	값이 다름	string, integer, double, boolean
<code>contains</code>	문자열에 일부 문자가 포함됨	string
<code>startsWith</code>	문자열이 지정한 값으로 시작함	string
<code>endsWith</code>	문자열이 지정한 값으로 끝남	string
<code>matches</code>	정규식에 일치함	string
<code>greaterThan</code>	지정한 숫자나 시간보다 큼	integer, double, duration
<code>greaterThanOrEqual</code>	지정한 숫자나 시간보다 크거나 같음	integer, double, duration
<code>lessThan</code>	지정한 숫자나 시간보다 작음	integer, double, duration
<code>lessThanOrEqual</code>	지정한 숫자나 시간보다 작거나 같음	integer, double, duration

`exists`와 `notExists`는 `logAttribute`, `datapointAttribute`, `spanAttribute`, `resourceAttribute`처럼 attribute를 확인하는 필드에서만 사용할 수 있습니다.

Duration 비교는 Trace의 `duration` 필드에서만 사용할 수 있으며 `equals`와 `notEquals`는 지원하지 않습니다.

`matches`에는 RE2 호환 정규식을 사용합니다. 잘못된 정규식은 적용 전에 거부됩니다. 복잡하고 넓은 정규식보다 정확한 prefix나 `equals`를 우선하면 의도하지 않은 데이터 제외를 줄일 수 있습니다.

값 타입 작성법

`value`에는 비교 대상에 맞는 타입을 정확히 하나만 작성합니다. 기본 타입은 string, integer, double, boolean이며, Trace의 `duration` 필드에는 duration을 사용합니다.

```
value:
  string: production
```

```
value:  
integer: 13
```

```
value:  
double: 0.5
```

```
value:  
boolean: true
```

```
value:  
duration: 250ms
```

attribute의 타입과 비교 값의 타입은 같아야 합니다. 특히 `integer: 1`과 `double: 1.0`은 서로 다른 타입으로 평가됩니다.

Duration은 `5ms`, `250ms`, `2s`, `1m`처럼 Kubernetes duration 형식으로 작성하며 `0`보다 커야 합니다.

Caution

필터 규칙은 Kubernetes 리소스에 평문으로 저장됩니다. Token, 비밀번호, 개인 식별 정보 같은 민감한 값을 정확한 일치 조건으로 넣지 마세요. 해당 값은 `MonithubAgent` 읽기 권한이 있는 사용자에게 보일 수 있습니다.

규칙 제한 [🔗](#)

항목	제한
Metrics 규칙	최대 20개
Logs 규칙	최대 20개
Traces 규칙	최대 20개
한 규칙의 <code>match.all</code> 조건	1개 이상, 최대 10개
규칙 이름	최대 63자, 소문자 영문·숫자·하이픈 사용
attribute key	최대 256자
string 값	최대 512자
정규식	최대 256자, RE2 문법
Trace duration	<code>0</code> 보다 큰 값

규칙 이름은 같은 데이터 종류 안에서 중복될 수 없습니다. 다음처럼 짧고 목적이 드러나는 이름을 사용합니다.

```
drop-health-access-logs
drop-health-check-spans
drop-idle-container-points
drop-runtime-metrics
```

적용 상태 확인 [🔗](#)

Helm upgrade 후 Monithub K8S Operator와 Collector rollout을 확인합니다.

```
kubectl -n monithub-system rollout status deploy/monithub-operator
kubectl -n monithub-system rollout status deploy/monithub-collector
```

필터 상태를 확인합니다.

```
kubectl -n monithub-system get monithubagent monithub -o yaml
```

정상적으로 적용되면 다음 값을 확인할 수 있습니다.

```
status:
  spanFiltering:
    desiredMode: dropMatching
    appliedMode: dropMatching
    configuredRuleCount: 1
    appliedRuleCount: 1
    desiredRevisionHash: <REVISION_HASH>
    appliedRevisionHash: <REVISION_HASH>
  metricFiltering:
    desiredMode: dropMatching
    appliedMode: dropMatching
    configuredRuleCount: 2
    appliedRuleCount: 2
    desiredRevisionHash: <REVISION_HASH>
    appliedRevisionHash: <REVISION_HASH>
  logFiltering:
    desiredMode: dropMatching
    appliedMode: dropMatching
    configuredRuleCount: 2
    appliedRuleCount: 2
    desiredRevisionHash: <REVISION_HASH>
    appliedRevisionHash: <REVISION_HASH>
```

주요 condition은 다음과 같이 읽습니다.

condition/reason	의미
<code>SpanFilteringApplied=True, RulesApplied</code>	Traces 규칙이 적용되었습니다.
<code>MetricFilteringApplied=True, RulesApplied</code>	Metrics 규칙이 적용되었습니다.
<code>LogFilteringApplied=True, RulesApplied</code>	Logs 규칙이 적용되었습니다.
<code>PassAll</code>	해당 데이터 종류에 규칙이 없어 모두 통과합니다.
<code>RulesAppliedWithWarnings</code>	규칙은 적용됐지만 범위가 넓거나 수집 지점 최적화가 제한될 수 있습니다. 경고 내용을 확인합니다.
<code>RolloutPending</code>	새 Collector revision이 준비되는 중입니다.
<code>InvalidSpanFilterPolicy</code>	Traces 규칙의 필드, 연산자 또는 값이 잘못되었습니다.
<code>InvalidMetricFilterPolicy</code>	Metrics 규칙의 필드, 연산자 또는 값이 잘못되었습니다.
<code>InvalidLogFilterPolicy</code>	Logs 규칙의 필드, 연산자 또는 값이 잘못되었습니다.
<code>CollectorConfigRejected</code>	새 Collector revision이 실행되지 않아 거부되었습니다.
<code>RolledBackToLastKnownGood</code>	이전에 정상 동작하던 Collector revision으로 복구되었습니다.

`desiredRevisionHash`와 `appliedRevisionHash`가 같고 적용 condition이 `True`이면 현재 values의 규칙이 실행 중입니다.

Trace, Metrics, Logs 설정은 하나의 Collector revision으로 함께 적용됩니다. 어느 한 데이터 종류의 규칙이 잘못되면 새 revision 전체를 적용하지 않으며, 기존에 정상 동작하던 설정을 유지합니다.

안전하게 검증하는 방법

운영 환경에 넓은 규칙을 바로 적용하지 말고 다음 순서로 검증합니다.

1. 개발 또는 staging 클러스터에서 metric 이름, log route, span 이름처럼 범위가 좁은 대상 하나로 규칙을 만듭니다.
2. 조건에 일치해 사라져야 하는 제외 대상 데이터를 발생시킵니다.
3. 조건 하나가 다른 비교용 데이터도 함께 발생시킵니다.

4. 사용한 데이터 종류의 `SpanFilteringApplied`, `MetricFilteringApplied`, `LogFilteringApplied` condition이 `True`인지 확인합니다.
5. Monithub의 최근 시간 범위에서 제외 대상은 없고 비교용 데이터는 남아 있는지 확인합니다.
6. 충분한 시간 동안 대시보드, 알림, 인시던트 분석에 필요한 데이터가 유지되는지 확인한 뒤 범위를 넓힙니다.

필터를 적용하기 전에 사용량이 높은 metric, log, span을 먼저 파악하면 비용 절감 효과와 데이터 손실 위험을 함께 판단하기 쉽습니다.

필터 제거하기 [🔗](#)

필터링을 중단하려면 각 데이터 종류의 규칙을 빈 배열로 바꾸고 Helm upgrade를 다시 실행합니다.

```
telemetry:
  traces:
    spanFiltering:
      rules: []
  metrics:
    filtering:
      rules: []
  logs:
    filtering:
      rules: []
```

모두 통과시키는 새 Collector revision이 Ready가 되면 이후에 들어오는 데이터부터 다시 저장됩니다. 필터가 활성화된 동안 이미 제외된 데이터는 복구되지 않습니다.

문제 해결 [🔗](#)

증상	원인 후보	확인하고 조치
<code>unknown field "filtering"</code> 또는 <code>"spanFiltering"</code>	CRD가 이전 버전임	chart의 CRD를 먼저 server-side apply한 뒤 Helm upgrade를 다시 실행합니다.
<code>InvalidSpanFilterPolicy</code>	Trace field, duration, span kind 또는 status 값 오류	Traces 지원 필드 표와 Monithub K8S Operator의 condition을 확인합니다.
<code>InvalidMetricFilterPolicy</code>	scope에 맞지 않는 field, 잘못된 타입 또는 연산자	Metrics 지원 필드 표와 실제 attribute 타입을 확인합니다.
<code>InvalidLogFilterPolicy</code>	severity 범위, 정규식, key/value 계약 오류	Monithub K8S Operator의 condition과 log를 확인하고 규칙을 좁혀 다시 적용합니다.
<code>RolloutPending</code> 이 오래 지속됨	Collector Pod가 Ready가 되지 않음	Collector Pod event, image pull, resource 부족 상태를 확인합니다.
<code>RolledBackToLastKnownGood</code>	새 설정 또는 custom Collector image가 실행되지 않음	현재 규칙과 Collector image의 filter processor 지원 여부를 확인합니다.
적용 상태는 정상인데 데이터가 계속 보임	기존 저장 데이터, 다른 수집 경로, field/key/type 불일치	최근 시간 범위로 조회하고 실제 telemetry의 attribute 이름과 타입을 확인합니다.
예상보다 많은 데이터가 사라짐	조건이 너무 넓은	규칙을 빈 배열로 되돌린 뒤 정확한 metric 이름, service 또는 route 조건을 추가합니다.
Trace가 중간에서 끊겨 보임	부모 span만 필터링됨	부모·자식 관계를 확인하고 전체 호출 구간에 공통으로 존재하는 좁은 조건을 사용합니다.
<code>RulesAppliedWithWarnings</code> 가 표시됨	넓은 Trace 규칙 또는 제한적인 수집 지점 최적화	경고 메시지를 확인합니다. 규칙은 gateway에 적용되므로 데이터 결과를 검증한 뒤 필요하면 범위를 좁힙니다.
문자열 조건이 숫자 attribute에 적용되지 않음	타입 불일치로 fail-open됨	실제 타입에 맞춰 <code>integer</code> , <code>double</code> , <code>boolean</code> 중 하나를 사용합니다.

진단할 때는 다음 로그와 리소스를 순서대로 확인합니다.

```

kubectrl -n monithub-system describe monithubagent monithub
kubectrl -n monithub-system logs deploy/monithub-operator --tail=200
kubectrl -n monithub-system logs deploy/monithub-collector --tail=200
kubectrl -n monithub-system describe deploy/monithub-collector

```

자주 묻는 질문 [🔗](#)

기존에 저장된 데이터도 삭제되나요? [🔗](#)

아니요. 필터는 적용 이후 gateway에 들어오는 데이터에만 작동합니다. 이미 Monithub에 저장된 데이터는 삭제하지 않습니다.

특정 데이터만 통과시키는 keep 규칙을 만들 수 있나요? [🔗](#)

현재는 제외 규칙만 지원합니다. 허용 목록(allowlist)이 필요하다면 여러 개의 `notEquals` 규칙을 조합하기보다 수집 범위를 먼저 조정하거나 Monithub 지원팀과 정책을 검토하세요.

필터를 바꾸면 Collector가 재시작되나요? [🔗](#)

네. Monithub K8S Operator는 전체 Collector 설정의 새 revision을 만들고 순차적으로 교체합니다. 새 revision이 Ready가 된 경우에만 적용 완료로 기록하며, 실패하면 이전 정상 revision으로 되돌립니다.

Trace 규칙에 일치하면 trace 전체가 제외되나요? [🔗](#)

아니요. 조건에 일치하는 개별 span만 제외됩니다. 부모 span이나 호출 관계의 한쪽만 제외하면 trace와 Service Map이 끊겨 보일 수 있으므로, 적용 전에 부모·자식 관계를 함께 확인하세요.

Span 필터링이 sampling을 대신하나요? [🔗](#)

아니요. Sampling은 trace 전체의 수집량을 비율이나 완성된 결과로 결정하고, span 필터링은 gateway에 도착한 개별 span의 이름, 속성, 상태, 시간을 기준으로 제외합니다. 두 기능은 목적과 판단 시점이 다릅니다.

실제 제외 건수를 MonithubAgent 상태에서 볼 수 있나요? [🔗](#)

현재 `MonithubAgent` status는 적용된 규칙 수와 revision을 보여주지만 누적 제외 건수는 제공하지 않습니다. Collector 내부 metrics를 별도로 수집하는 환경에서는 `otelcol_processor_filter_spans.filtered`, `otelcol_processor_filter_datapoints.filtered`, `otelcol_processor_filter_logs.filtered`를 사용할 수 있습니다.

다음 단계 [🔗](#)

- [Monithub K8S Operator 설치](#)에서 values 관리와 업데이트 절차를 확인합니다.
- [Connect와 Gateway](#)에서 데이터 연결 상태를 확인합니다.
- [대시보드](#)에서 필터 적용 전후의 데이터 범위를 비교합니다.