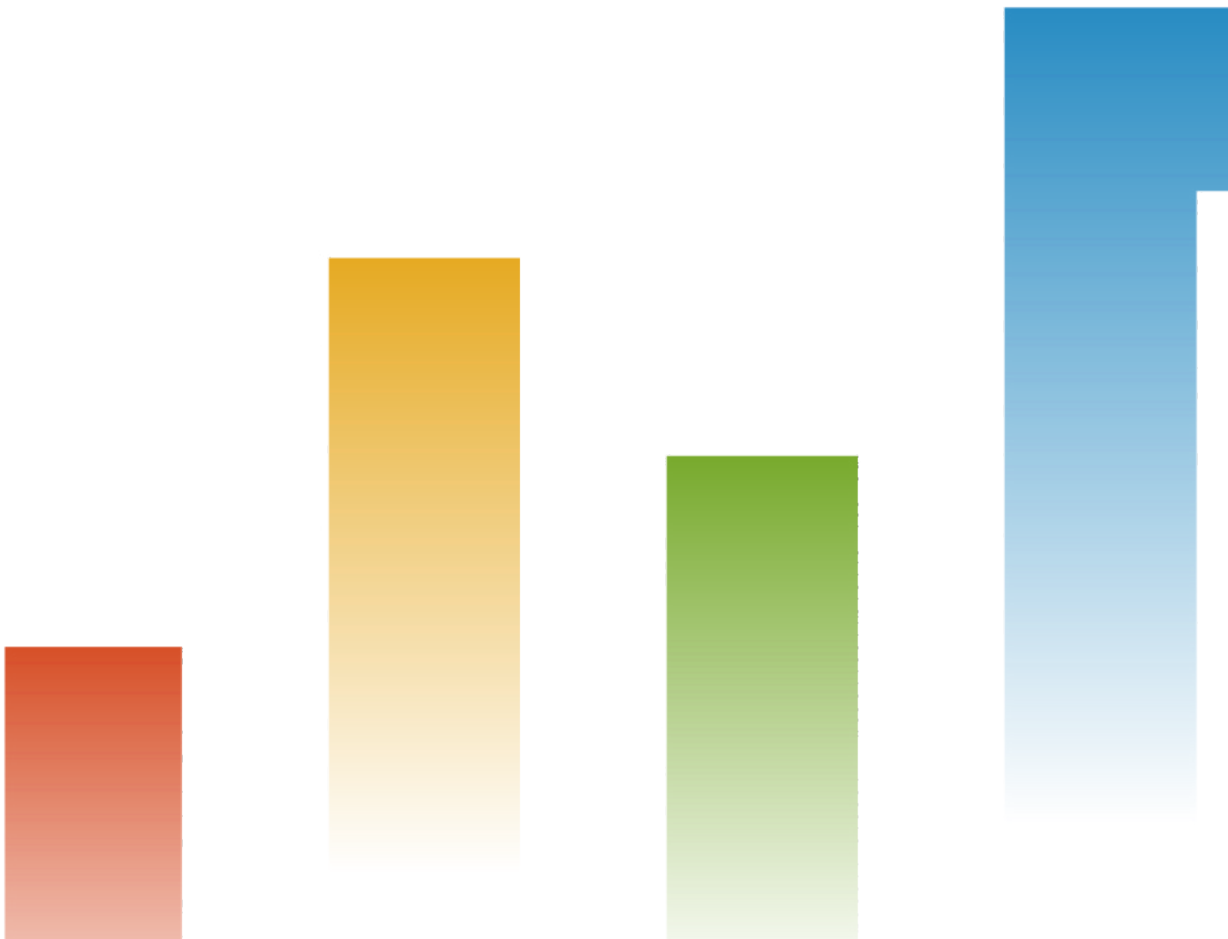
WhaTap

메트릭스 경고 알림 설정

release date. 2026. 09. 10.



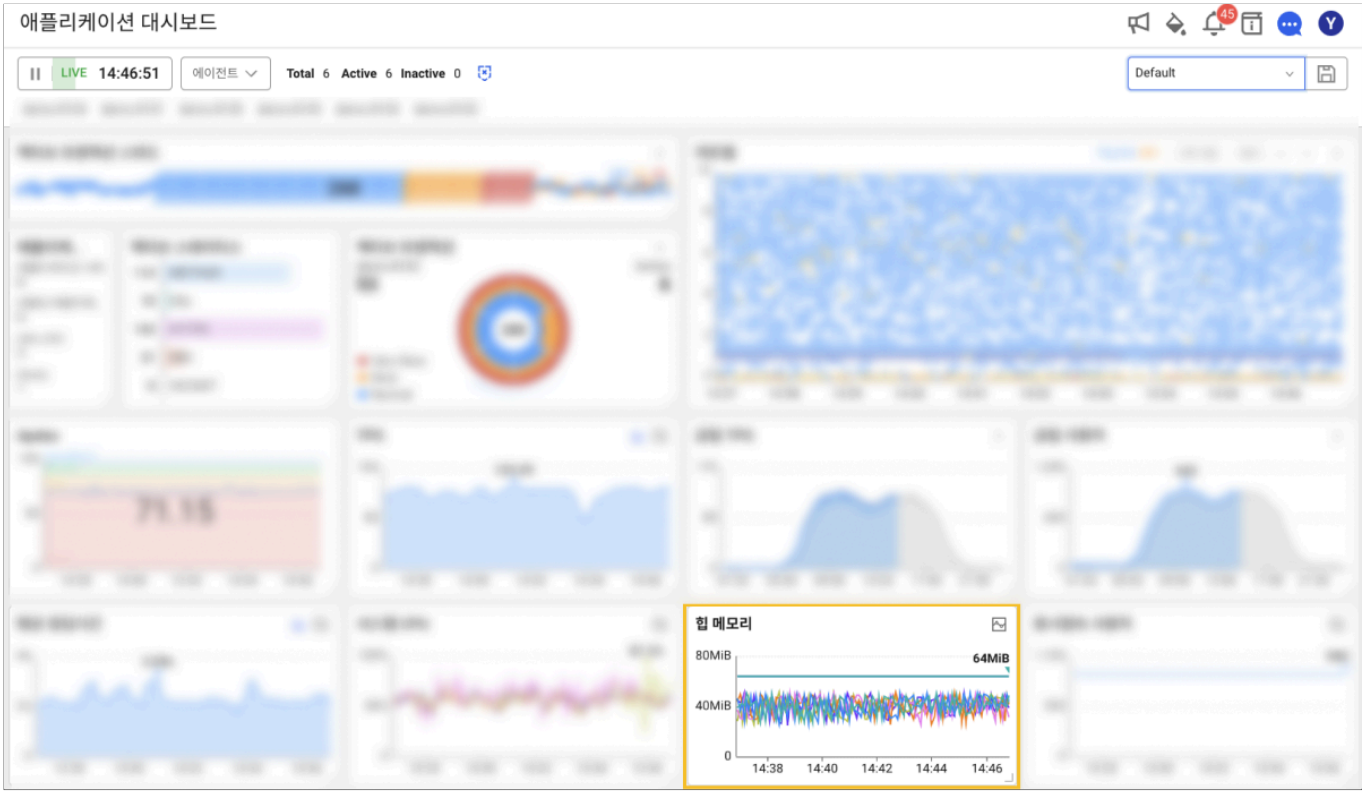
힙 메모리 메트릭스 알림 설정

힙 메모리는 GC·메모리 누수 판단의 핵심 지표지만, 어느 메트릭을 봐야 하고 어떤 임계값으로 알림을 걸어야 하는지 판단하기 어렵습니다. 이 문서는 WhaTap APM의 힙 메모리 관련 지표를 카탈로그화하고, 메트릭스 알림 예시(`heap_used` 기반)와 차트 조회 방법을 안내합니다.

Java 애플리케이션을 WhaTap APM으로 모니터링하는 개발자·운영자를 대상으로 합니다.

힙 메모리 메트릭스

힙 메모리(Heap Memory)는 자바 가상 머신(JVM)이 프로그램을 실행하기 위해 할당한 데이터 저장 공간으로 인스턴스와 배열 등 주요 데이터를 저장합니다. 힙 영역은 인스턴스의 라이프 사이클 및 GC와 밀접한 관계를 가지고 있습니다. GC는 불필요한 인스턴스를 힙 영역에서 해제해 힙 영역의 빈 공간을 확보하는 작업입니다.



힙 메모리 관련 지표를 통해 서버별 최대 힙 메모리, 현재 힙 메모리를 확인해 위험 수준을 살펴볼 수 있습니다. 또한 최댓값과 현재 값의 편차를 파악해 적절하게 힙 메모리를 설정할 수 있습니다. 힙 메모리 사용량의 최솟값이 지속적으로 상승하는 경우 메모리

누수 현상을 의심해 볼 수 있습니다.

메트릭스 조회

홈 화면 > 프로젝트 선택 > **사이트맵** > **분석** > **메트릭스 조회**

메트릭스 조회 메뉴에서 확인할 수 있는 힙 메모리 관련 지표를 안내합니다.

Heap Memory

`app_proc_counter` 카테고리 선택 후 확인할 수 있습니다.

- **Heap 커밋** `heap_committed`

  JVM이 사용하도록 커밋된 힙 메모리를 의미합니다.

- **최초 Heap** `heap_init`

  JVM이 메모리 관리를 위해 OS에 요청한 최초 힙 메모리를 의미합니다.

- **Heap 최대 크기** `heap_max`

  JVM이 메모리 관리에 사용할 수 있는 최대 힙 메모리를 의미합니다.

- **Heap 사용량** `heap_used`

  JVM에서 사용 중인 힙 메모리를 의미합니다.

- **Heap 총 사용량** `heap_tot`

  JVM이 할당받은 힙 메모리의 총량을 의미합니다.

Non-heap Memory

`java_memory` 카테고리 선택 후 확인할 수 있습니다.

- **Non-heap 커밋** `nonheap_committed`

  JVM이 사용하도록 커밋된 Non-heap 메모리를 의미합니다.

- **최초 Non-heap** `nonheap_init`

  JVM이 메모리 관리를 위해 OS에 요청한 최초 Non-heap 메모리를 의미합니다.

- **Non-heap 총 사용량** `nonheap_max`

  JVM에서 메모리 관리에 사용할 수 있는 최대 Non-heap 메모리 사용량을 의미합니다.

  > ⓘ Metaspace 및 JIT 코드 캐시 최대 메모리 크기가 정의되지 않은 경우 `-1` 을 반환합니다.

- **Non-heap 사용량** `nonheap_used`

  JVM에서 사용된 Non-heap 메모리를 의미합니다.

메트릭스 차트

홈 화면 > 프로젝트 선택 > **분석** > **메트릭스 차트**

메트릭스 차트 메뉴에서 확인할 수 있는 힙 메모리 관련 지표를 안내합니다.

- **Heap 최대 크기** `heap_max`

  JVM이 메모리 관리에 사용할 수 있는 최대 힙 메모리를 의미합니다.

- **Heap 총 사용량** `heap_tot`

  JVM이 할당받은 힙 메모리의 총량을 의미합니다.

- **Heap 사용량** `heap_use`

  JVM에서 사용 중인 힙 메모리를 의미합니다.

- **Heap Perm 영역 사용량** `heap_perm`

  Perm 영역의 메모리 사용량을 의미합니다.

- **Heap Pending Finalization 개수** `heap_pending_finalization`

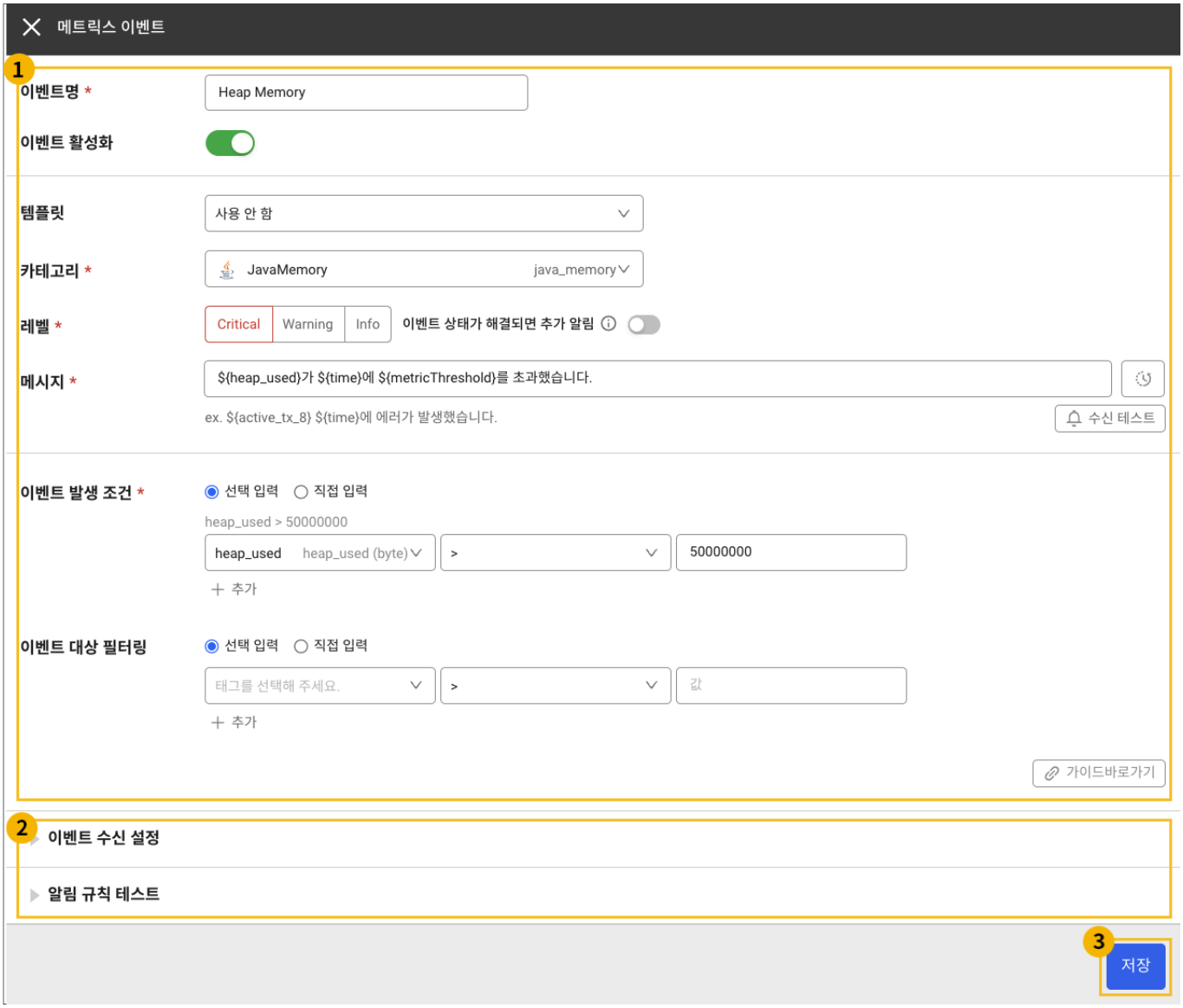
  대기 중인 Finalization을 가지고 있는 Object의 대략적인 개수를 의미합니다.

> ⓘ 애플리케이션 지표에 대한 내용은 다음 문서를 참조하세요.

힙 메모리 메트릭스 알림 설정

홈 화면 > 프로젝트 선택 > **경고 알림** > **메트릭스** 탭 선택

힙 메모리 관련 경고 알림을 사용하기 위해서는 메트릭스 경고 알림을 설정해야 합니다. 메트릭스 경고 알림은 기본 알림보다 구체적이고 복잡한 알림 설정이 가능합니다. 자세한 내용은 다음 문서를 참조하세요. 다음의 예시는 힙 메모리 사용량(`heap_used`)을 기준으로 메트릭스 알림을 설정하는 방법을 안내합니다.



1. **경고 알림** 하위의 **이벤트 설정** 메뉴에서 **메트릭스** 탭을 선택하세요.
2. 화면 상단 오른쪽에서 **+ 이벤트 추가** 버튼을 클릭하세요.
3. **메트릭스 이벤트** 창의 1 영역에서 **이벤트명**, **카테고리**, **레벨**, **메시지**, **이벤트 발생 조건** 등을 설정하세요.
   - **이벤트명**을 입력하세요.

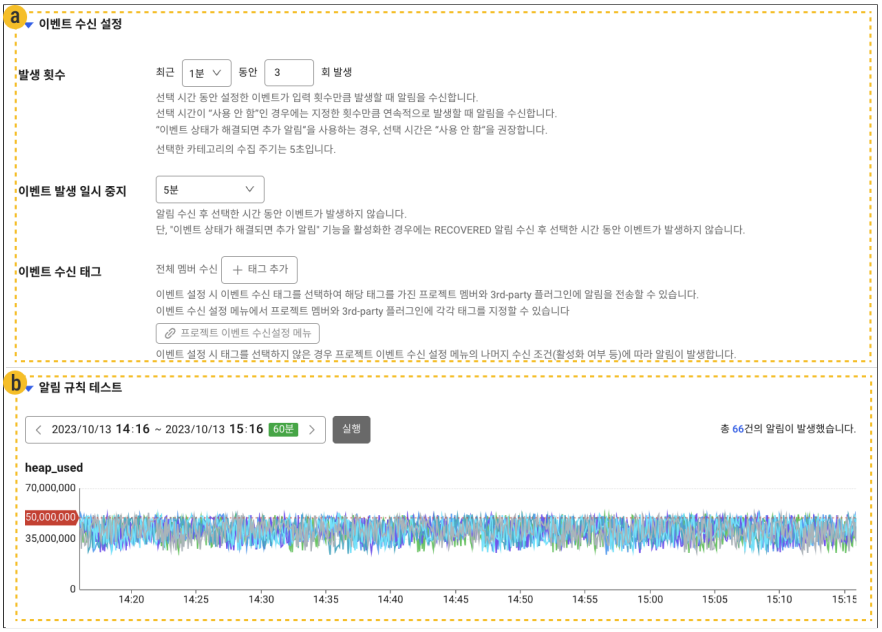
     Heap Memory

   - **레벨**은 이벤트 발생 시 경고 수준을 의미합니다. 레벨을 설정하세요. 기본 설정은 Critical 수준입니다.
   - **카테고리** 입력창에서 **JavaMemory**(java_memory)를 선택하세요.
   - 이벤트 **메시지**를 입력하세요. 예시 메시지의 경우 힙 메모리 사용량이 설정한 임계값 초과 시 해당 이벤트 발생 시간과 함께 발송됩니다.

     ${heap_used} 가 ${time} 에 ${metricThreshold} 를 초과했습니다.

   - 필드, 연산자, 임계값을 입력해 **이벤트 발생 조건**을 설정하세요. 힙 사용량이 50,000,000 byte 초과 시 알림을 발생시키는 조건 예시는 다음과 같습니다.

     필드 heap_used , 연산자 > , 임계값 50000000
4. 2 영역에서 a 이벤트 수신 관련 세부 요건을 설정하고 해당 이벤트 알림 규칙을 b 테스트해 보세요.



- ⓐ **이벤트 수신 설정**
  - **발생 횟수**는 최근 `n` 의 기간 동안 해당 이벤트가 `n'` 회 발생 시 알림을 수신하도록 합니다.

    최근 `1분` 동안 해당 이벤트 `3` 회 발생 시 알림이 발송됩니다.

  - **이벤트 발생 일시 중지**는 해당 알림 발생 후 설정한 시간 동안 동일한 알림이 발생하지 않도록 합니다.
- ⓑ **알림 규칙 테스트** 옵션은 선택한 기간 동안 해당 이벤트 알림 설정 시 몇 번의 경고 알림이 발생했는지 확인할 수 있습니다.

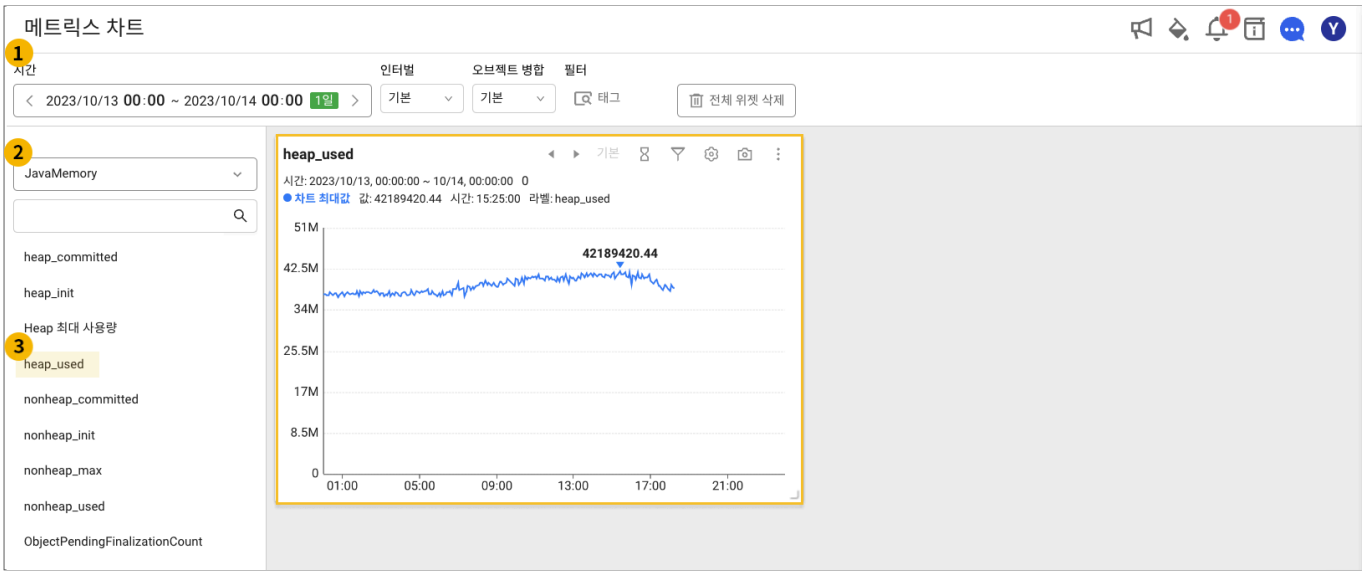
  지난 `60분` 동안 `1분` 내 연달아 `3` 회 이상 힙 메모리 사용량(`heap_used`)이 `50,000,000` byte를 초과하는 이벤트 발생 시 발송한 알림 건수는 총 **66**건입니다.

5. **메트릭스 이벤트** 창 하단의 ③ **저장** 버튼을 클릭해 알림 설정을 완료하세요.

힙 메모리 사용량 확인

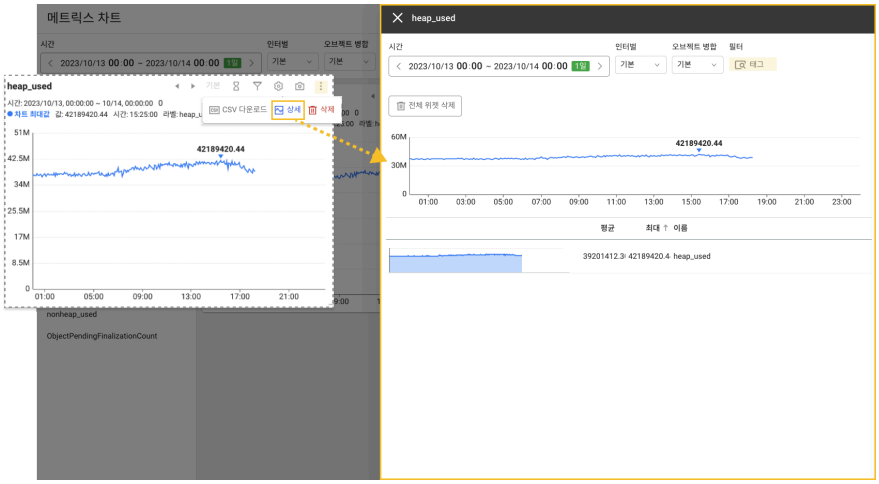
홈 화면 > 프로젝트 선택 > **분석** > **메트릭스 차트**

선택한 기간 동안의 힙 메모리 사용량(`heap_used`)을 상세하게 확인하고 싶다면 **메트릭스 차트**로 이동하세요.

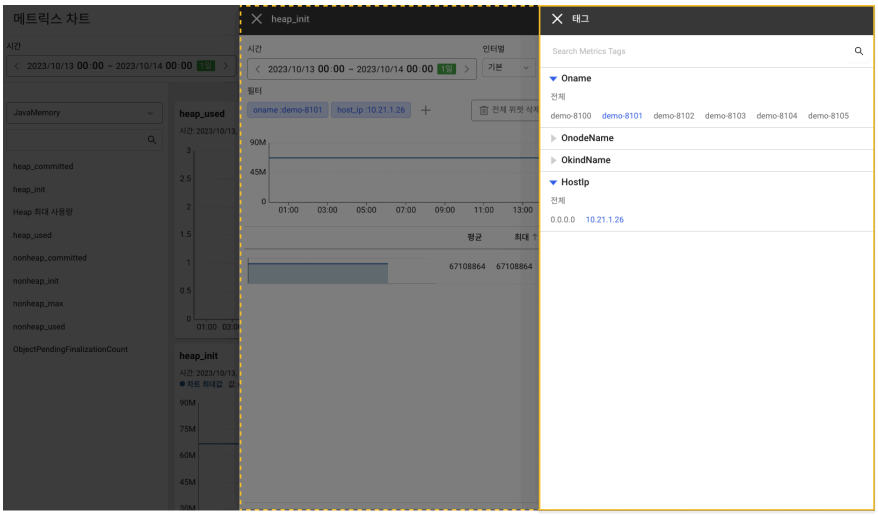


1. 상단의 ① 시간 선택자를 통해 조회를 원하는 기간을 선택하세요.

2. 왼쪽 지표 목록에서 2 **카테고리**는 **JavaMemory**(java_memory)를 선택하세요.
3. 3 **JavaMemory** 카테고리 하위에서 heap_used 를 선택하세요. 차트 영역에서 예시처럼 힙 메모리 사용량 차트를 조회할 수 있습니다.
4. heap_used 위젯 상단 오른쪽의 **더보기** 아이콘을 클릭 후 옵션 메뉴에서 **상세** 아이콘 선택 시 모니터링 대상의 지표 추이를 개별로 확인할 수 있습니다.



5. **태그** 필터 선택 시 다음과 같이 에이전트, 에이전트 그룹, 에이전트 호스트 등을 선택해 개별 차트를 조회할 수 있습니다.



참고

- Java SE 22
- Java Memory Usage
- 애플리케이션 지표

부하·성능 테스트 결과 가시화

애플리케이션 개발이 완료 단계에 가까워지면 **목표 성능을 실제로 낼 수 있는지**를 부하·성능 테스트로 검증합니다. 대부분의 테스트 도구(JMeter·LoadRunner 등)는 **시나리오별 집계 통계**를 제공하지만, "어느 트랜잭션의 어느 구간이 병목인가"를 서비스 관점에서 파악하긴 어렵습니다. WhaTap Monitoring을 함께 활용하면 테스트 결과를 **서비스·트랜잭션 관점으로 실시간 가시화**하고, 병목 지점·에러 원인을 정확히 추적할 수 있습니다.

부하·성능 테스트의 4가지 유형

표 | 부하·성능 테스트 4가지 유형

유형	목적
부하 테스트 (Load)	예상 최대 트래픽 상황에서의 성능·안정성 확인
스트레스 테스트 (Stress)	한계점·복구 동작·파손 임계 확인
스파이크 테스트 (Spike)	급격한 트래픽 증감에서의 반응 확인
내구성 테스트 (Endurance)	장시간 지속 부하에서 누수·누적 이슈 확인

핵심 측정 지표

- **응답 시간 (Response Time)** — 평균·p95·p99
- **처리량 (TPS, Transactions per Second)**
- **에러율 (Error Rate)**

도구 리포트만으론 **"왜 이 시나리오의 평균 응답 시간이 튀었는가"** 를 찾기 어렵습니다. 개별 트랜잭션 레벨의 추적이 필요합니다.

WhaTap과 함께 쓰는 방식

① 테스트 도구에서 부하 실행

JMeter, LoadRunner, k6, Locust 등 팀이 쓰는 도구로 시나리오를 실행합니다. **테스트 도구는 부하 생성과 집계 통계를 담당합니다.**

② WhaTap에서 서비스 관점 가시화

테스트 대상 애플리케이션에 WhaTap 에이전트가 설치되어 있으면, 테스트 동안 발생한 트랜잭션이 **실시간으로 수집**됩니다.

사용 메뉴:

- **애플리케이션 대시보드** — TPS·응답 시간·에러율 추이를 서비스 전체 관점에서
- **히트맵 트랜잭션** — 시간대별 트랜잭션 응답 분포, 이상 점 뭉침 즉시 식별
- **트랜잭션 트레이스** — 느린 트랜잭션 1건의 호출 스택, SQL·HTTP·외부 호출 시간
- **DB 연결 상태·Slow SQL** — DB 측 병목 교차 확인
- **힙 메모리·액티브 스택** — GC·메모리·스레드 병목 확인

③ 원인 파악 → 테스트 재실행 → 비교

개선 전후 **히트맵 패턴**, **트랜잭션 트레이스**, **대시보드 수치**를 같은 기간 범위로 비교해 **튜닝 효과를 정량적으로** 확인합니다.

활용 시나리오

릴리즈 전 부하 테스트

1. 테스트 도구로 예상 피크 트래픽을 재현하세요.
2. WhaTap 대시보드에서 **TPS·응답·에러율을 실시간으로 추적**하세요.
3. 에러 스파이크나 응답 급증 시 트랜잭션 트레이스에서 원인을 바로 분석하세요.

4. 결과를 릴리즈 검증 시나리오의 베이스라인 기록으로 활용하세요.

성능 회귀 검출

1. 이전 버전 대비 **같은 부하 조건**에서 테스트를 실행하세요.
2. 주요 지표(TPS, p95, 에러율) 변화를 정량적으로 비교하세요.
3. 회귀 발견 시 트랜잭션 트레이스에서 느려진 구간을 식별하세요.

병목 식별·튜닝 루프

- 히트맵에서 이상 점 뭉침을 찾고, 트레이스로 원인을 파악한 뒤, 코드·SQL·인덱스를 수정하고 재테스트로 비교합니다.
- 개선 사이클을 직감이 아닌 **측정 기반**으로 수행합니다.

다음 단계

- **히트맵·트레이스 심화** → 히트맵 트랜잭션 살펴보기
- **DB 측 병목 확인** → DB 연결 지연과 커넥션 풀, 데이터베이스 실시간 감시
- **메모리·GC 분석** → 힙 메모리 경고 알림
- **테스트 결과를 릴리즈 판단 기준으로** → 릴리즈 검증 시나리오