

클라우드 환경의 이해

개발자가 되기 전에 알았으면 좋았을 것들



안정적인 서비스 운영을 위하여

- 배포 자동화 그 이후의 이야기

- 확장된 서비스 운영의 규모
 - 더 많은 유저, 늘어난 트래픽에 대한 대처가 필요
 - 수많은 인프라 구성 중 문제가 발생한 곳을 빠르게 인식할 수 있어야 함
- 더욱 빨라진 배포 주기
 - 자동화된 배포 파이프라인이 구성됨에 따라 개발 조직은 고객의 요구에 알맞은 기능을 끊임없이, 더욱 자주 배포하게 됨
 - 때문에 더욱 다양한 경우에 대한 검증이 필요
- 배포된 기능에 대한 지표 추적 필요
 - 배포되는 다양한 기능들이, 실제로 잘 동작하는지 비즈니스 관점에서 지표를 추적해야 함

안정적인 서비스 운영을 위하여

- 모니터링이란?

- 사전적 의미로는 어떤 대상을 감시, 관찰한다는 뜻
- IT 환경에서는 시스템과 애플리케이션의 성능, 가용성 등을 확인하고 대비하는 것을 말한다.
- 모니터링 대상에 따라 다양하게 세분화할 수 있다.
 - 시스템 모니터링
 - 시스템에서 사용되고 있는 컴퓨팅이나 메모리, 스토리지와 같은 자원들에 대한 모니터링
 - Elasticsearch (Metricbeats), Prometheus, Zabbix, etc.
 - 네트워크 모니터링
 - 트래픽, 대역폭 사용량 등을 분석하여 네트워크 장애를 예방하고, 침투와 같은 이상 징후를 대비
 - Wireshark, Snort, Cilium, SNPM, etc.
 - 애플리케이션 모니터링
 - 애플리케이션의 성능, 가용성 등을 분석하여 문제를 발견하고 해결하기 위한 모니터링
 - Elastic APM, New Relic, Datadog, etc.
 - 실제 사용자 모니터링
 - 실제 사용자 경험을 분석하여 성능을 최적화하는 모니터링
 - GA(Google Analytics), Datadog RUM, Elastic RUM, etc.
 - etc.

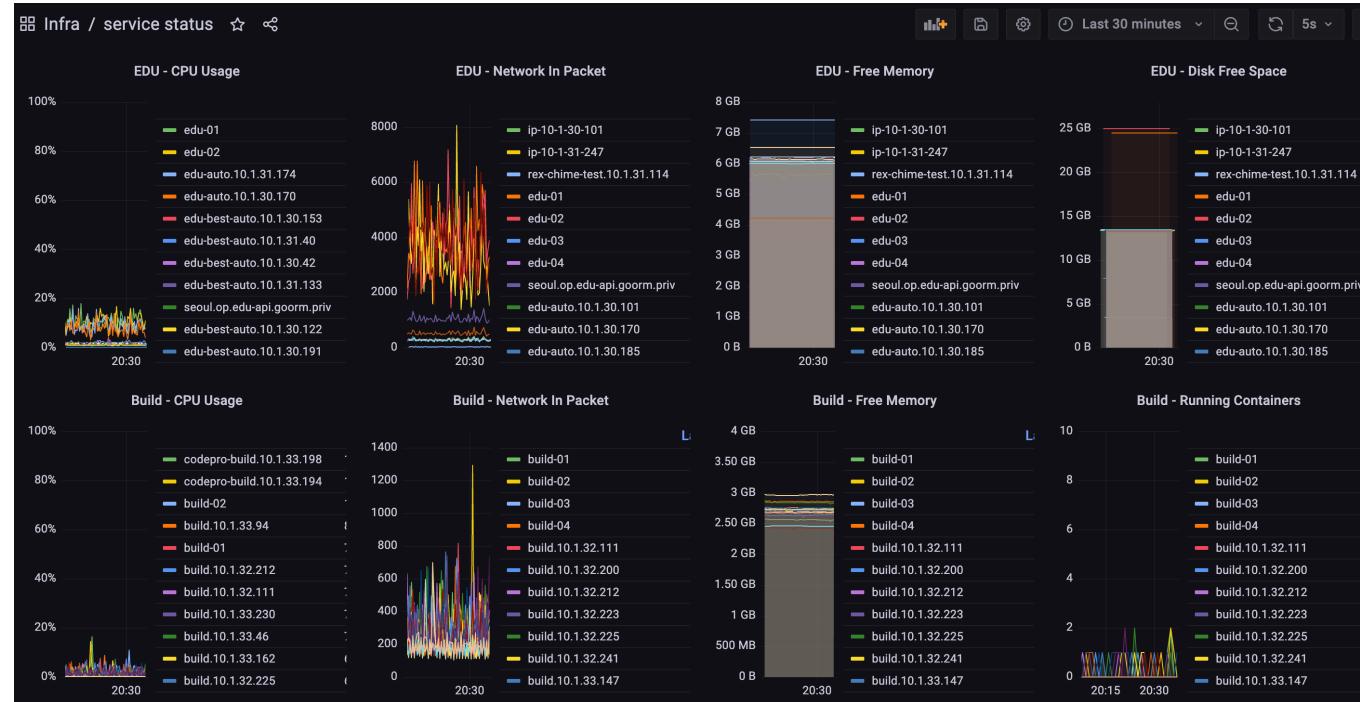
안정적인 서비스 운영을 위하여

- 시스템 모니터링

- 주요 지표
 - CPU
 - 시스템에서 작업 처리를 담당하기에, 사용량이 지속적으로 높은 경우 문제가 발생할 수 있다.
 - CPU Load
 - CPU Load는 CPU에서 실행 중이거나 대기 중인 작업의 수를 보여주는 지표이다.
 - 때문에 CPU Load가 높아진다는 것은 업무가 처리되는 속도가 지연되고 있으며, 작업이 처리되지 않고 있다는 것을 의미한다.
 - Memory
 - 현재 실행 중인 작업들에 대한 정보를 저장하고 있으며, 흔히 접할 수 있는 OOM(Out of Memory)가 발생하면 해당 정보를 손실할 수 있다.
 - Disk
 - 데이터를 저장하는 저장소로 공간 부족으로 인해, 프로세스에서 필요한 임시 파일 등을 생성하지 못하는 경우 장애로 이어질 수 있다.
 - Disk I/O
 - 한 번에 대량의 I/O 작업 요청이 오는 경우, 과부하로 인해 작업이 멈출 수 있으며, 크게는 파일시스템의 손상으로도 이어질 수 있다.

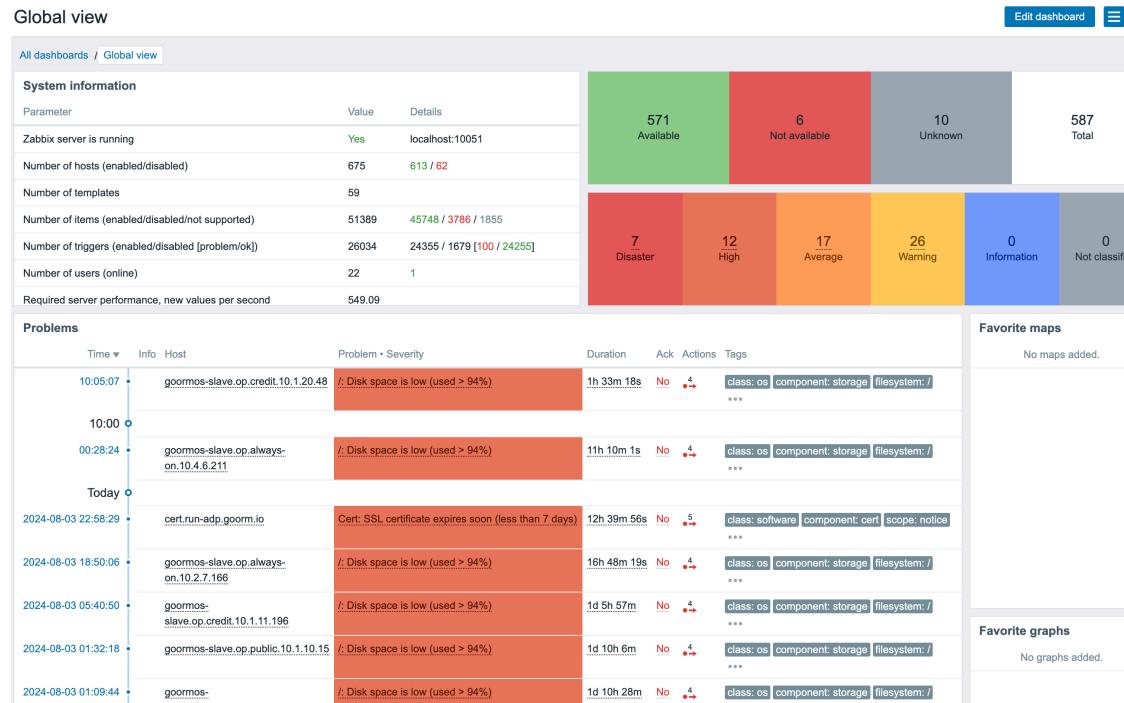
안정적인 서비스 운영을 위하여

- 시스템 모니터링



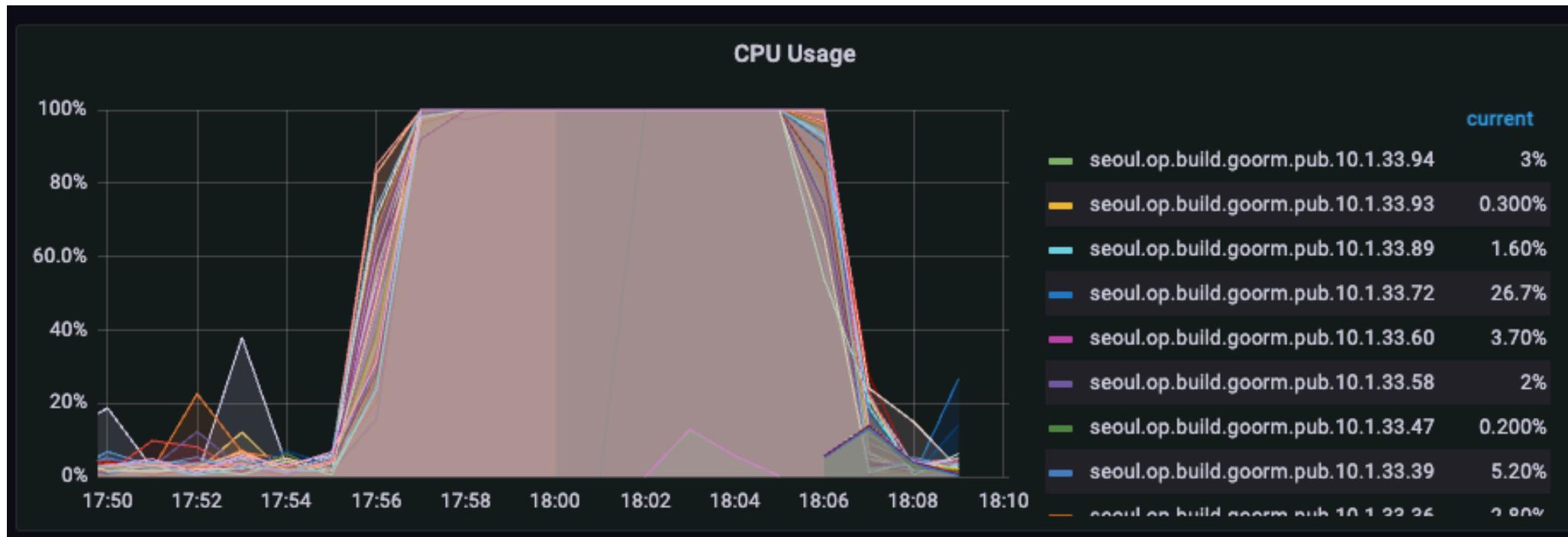
안정적인 서비스 운영을 위하여

- 시스템 모니터링



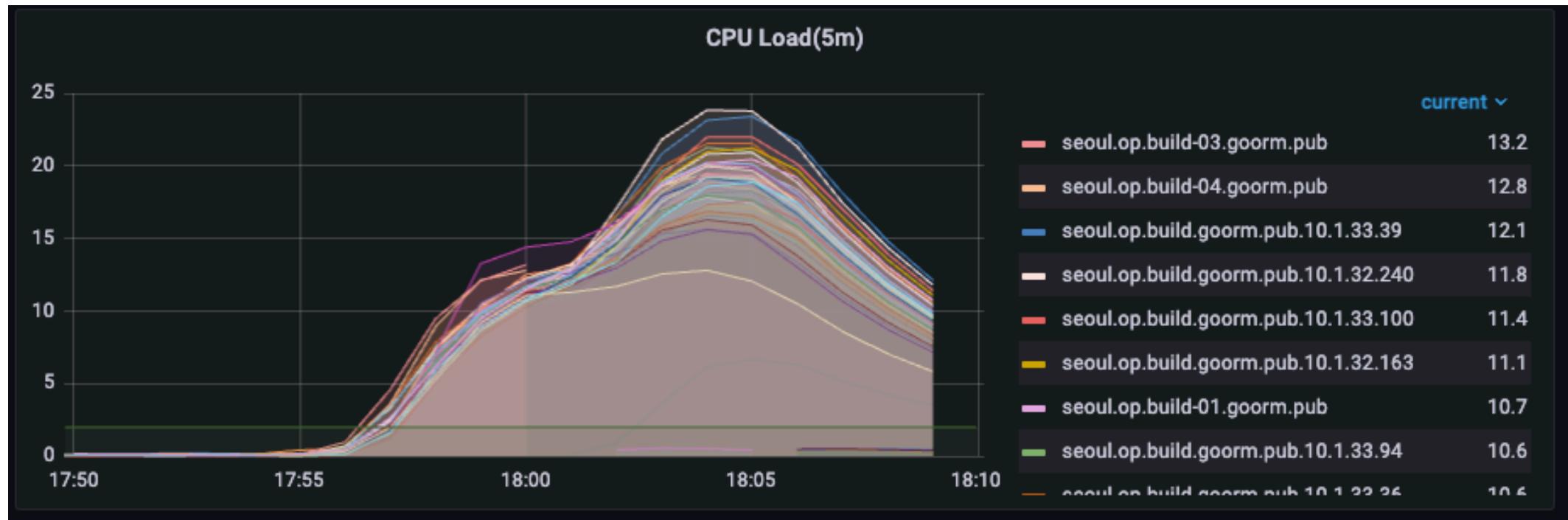
안정적인 서비스 운영을 위하여

- 시스템 모니터링 - CPU



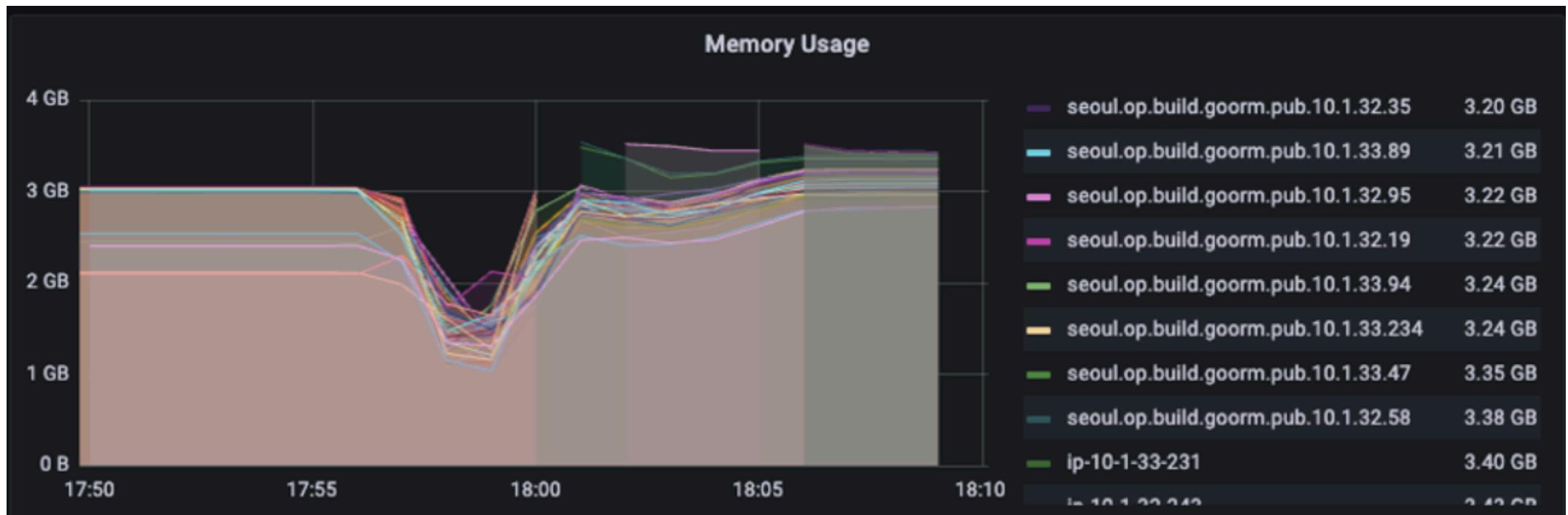
안정적인 서비스 운영을 위하여

- 시스템 모니터링 - CPU Load



안정적인 서비스 운영을 위하여

- 시스템 모니터링 - Memory



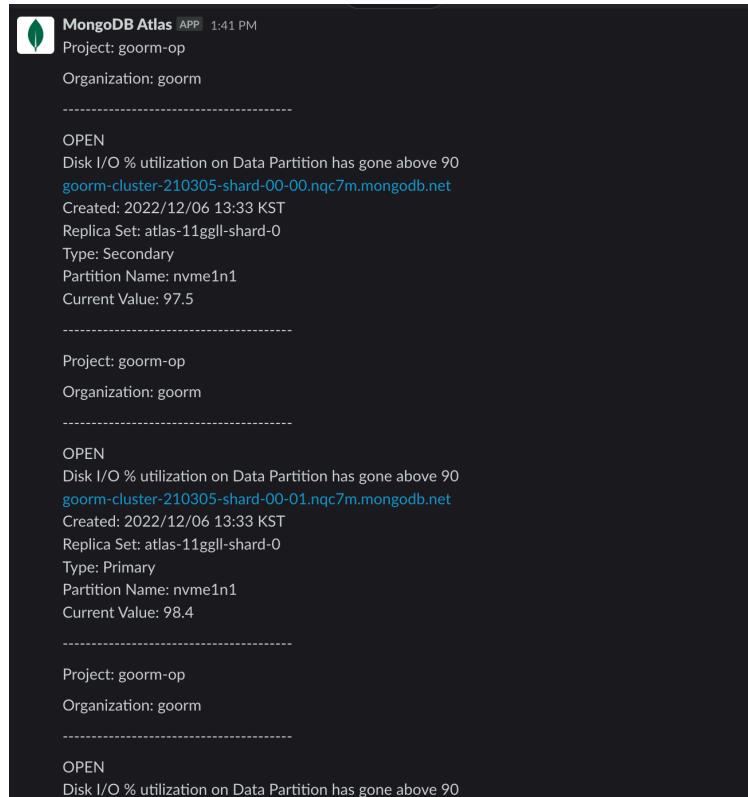
안정적인 서비스 운영을 위하여

- 시스템 모니터링 - Disk

```
Error: ENOSPC: no space left on device, write
Emitted 'error' event on WriteStream instance at:
  at emitErrorNT (node:internal/streams/destroy:157:8)
  at emitErrorCloseNT (node:internal/streams/destroy:122:3)
  at processTicksAndRejections (node:internal/process/task_queues:83:21) {
  errno: -28,
  code: 'ENOSPC',
  syscall: 'write'
}
{"level":"error","message":"Forever detected script exited with code: 1"}
{"level":"error","message":"Script restart attempt #1"}
node:internal/fs/utils:347
  throw err;
^
```

안정적인 서비스 운영을 위하여

- 시스템 모니터링 - Disk I/O



MongoDB Atlas APP 1:41 PM
Project: goorm-op
Organization: goorm

OPEN
Disk I/O % utilization on Data Partition has gone above 90
[goorm-cluster-210305-shard-00-00.nqc7m.mongodb.net](#)
Created: 2022/12/06 13:33 KST
Replica Set: atlas-11ggll-shard-0
Type: Secondary
Partition Name: nvme1n1
Current Value: 97.5

Project: goorm-op
Organization: goorm

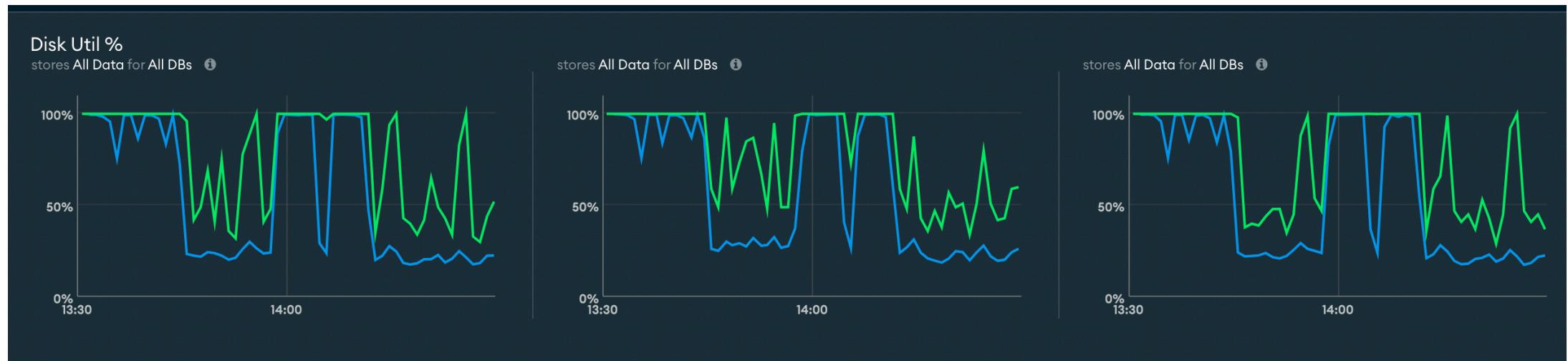
OPEN
Disk I/O % utilization on Data Partition has gone above 90
[goorm-cluster-210305-shard-00-01.nqc7m.mongodb.net](#)
Created: 2022/12/06 13:33 KST
Replica Set: atlas-11ggll-shard-0
Type: Primary
Partition Name: nvme1n1
Current Value: 98.4

Project: goorm-op
Organization: goorm

OPEN
Disk I/O % utilization on Data Partition has gone above 90
[goorm-cluster-210305-shard-00-02.nqc7m.mongodb.net](#)
Created: 2022/12/06 13:33 KST
Replica Set: atlas-11ggll-shard-0
Type: Secondary
Partition Name: nvme1n1
Current Value: 98.4

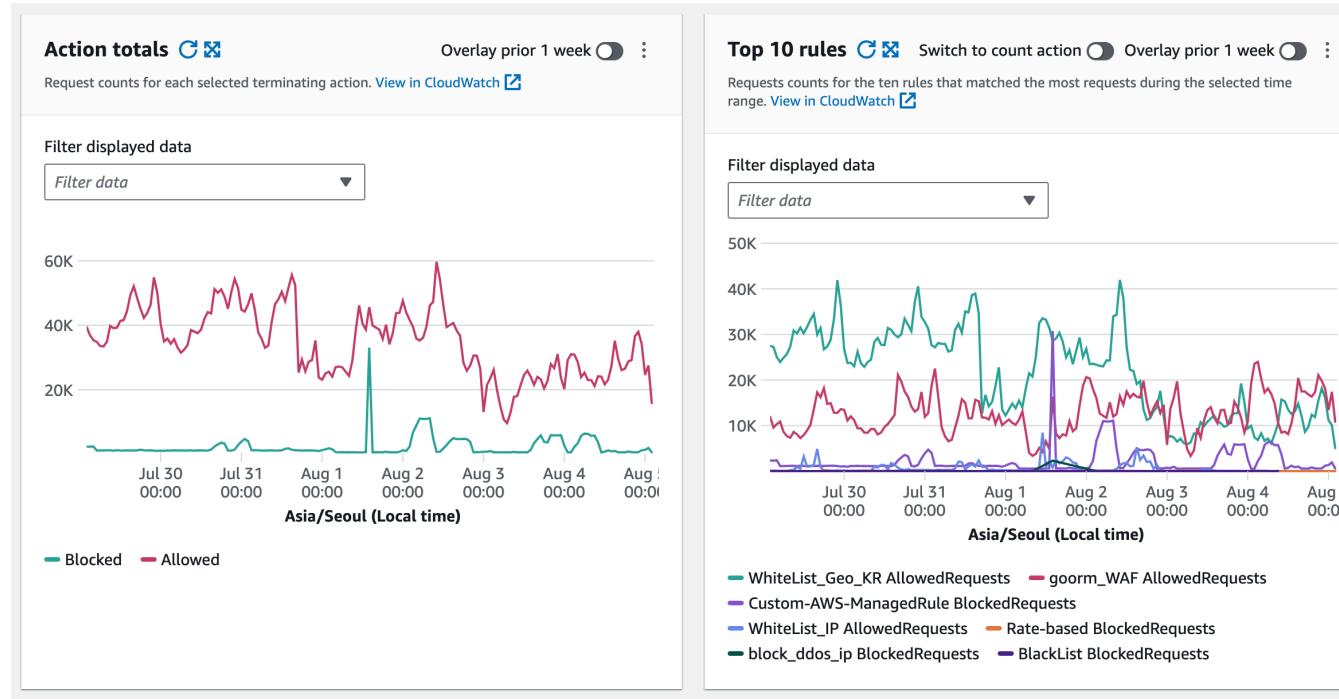
안정적인 서비스 운영을 위하여

- 시스템 모니터링 - Disk I/O



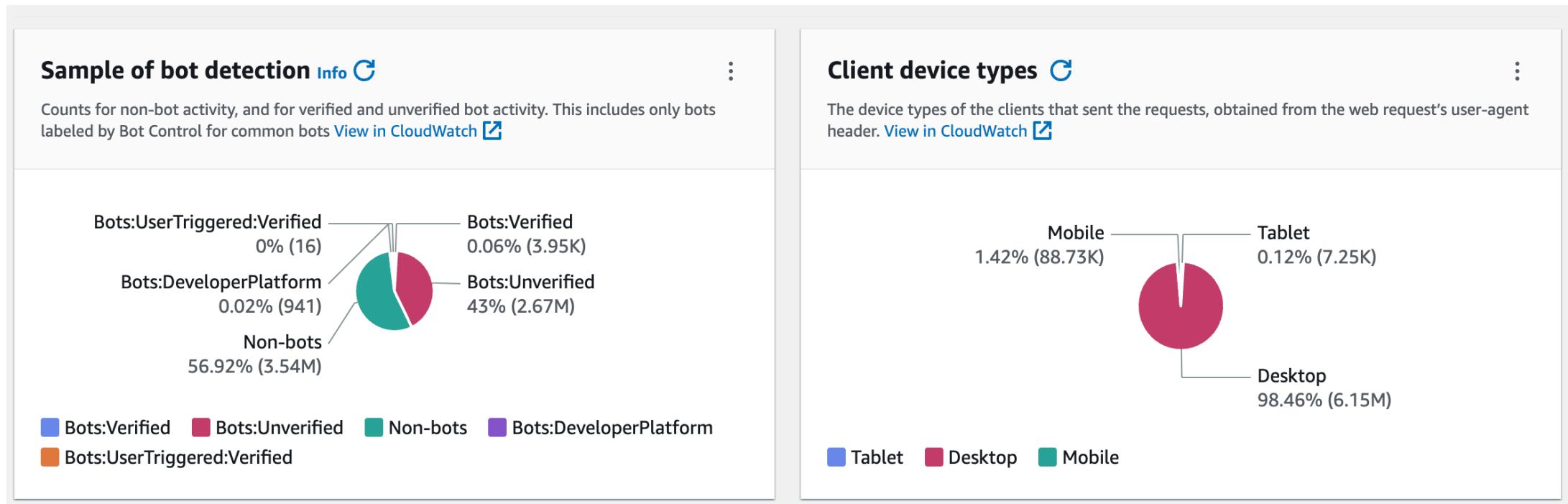
안정적인 서비스 운영을 위하여

- 네트워크 모니터링



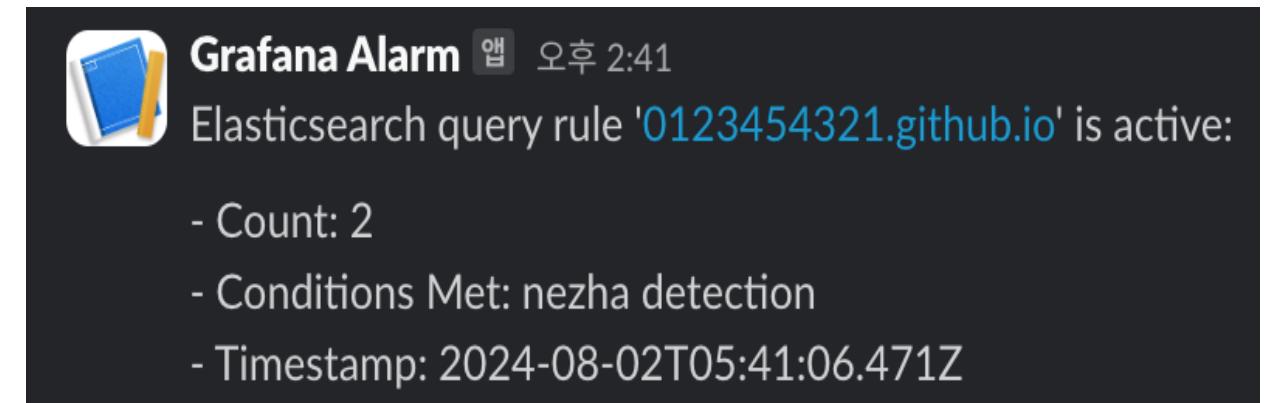
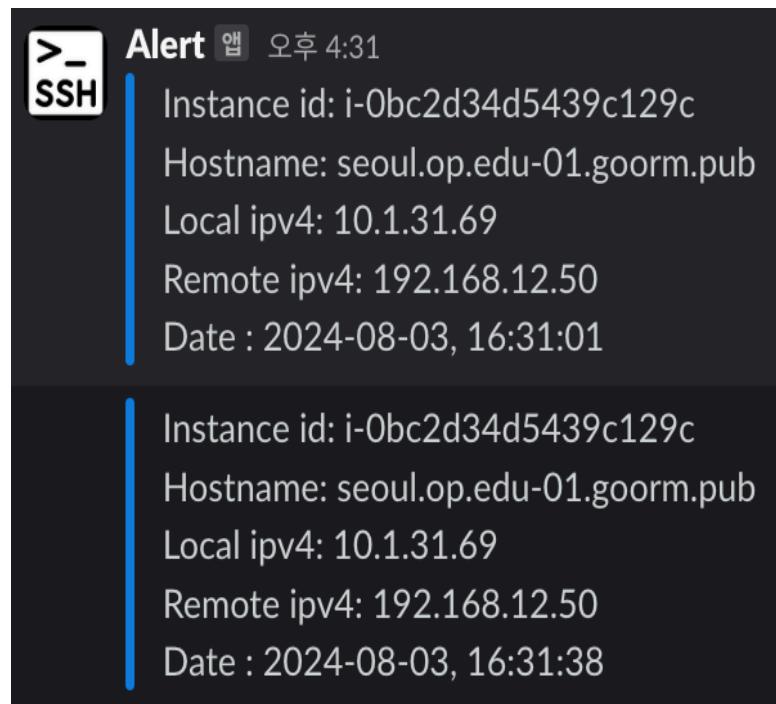
안정적인 서비스 운영을 위하여

- 네트워크 모니터링



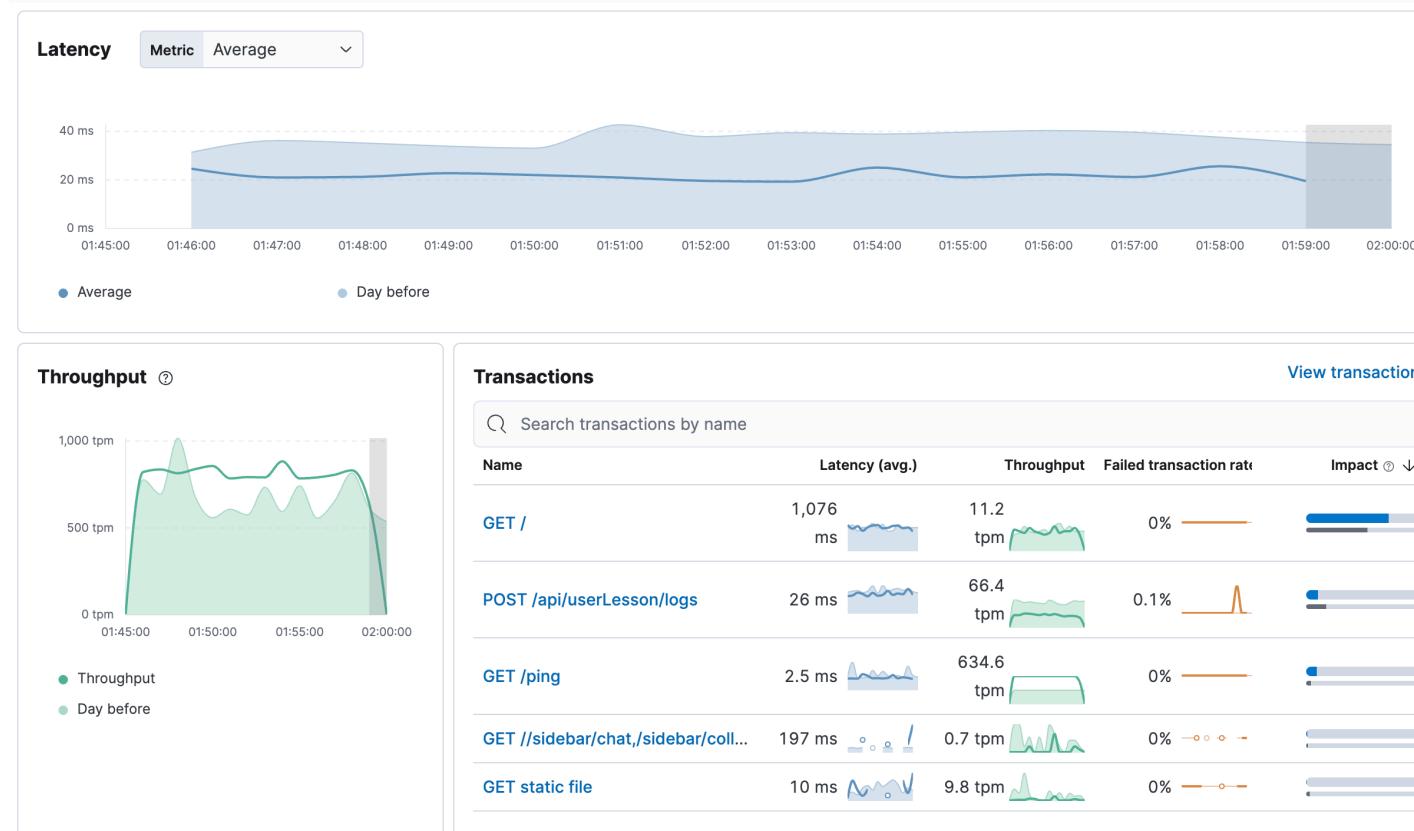
안정적인 서비스 운영을 위하여

- 네트워크 모니터링



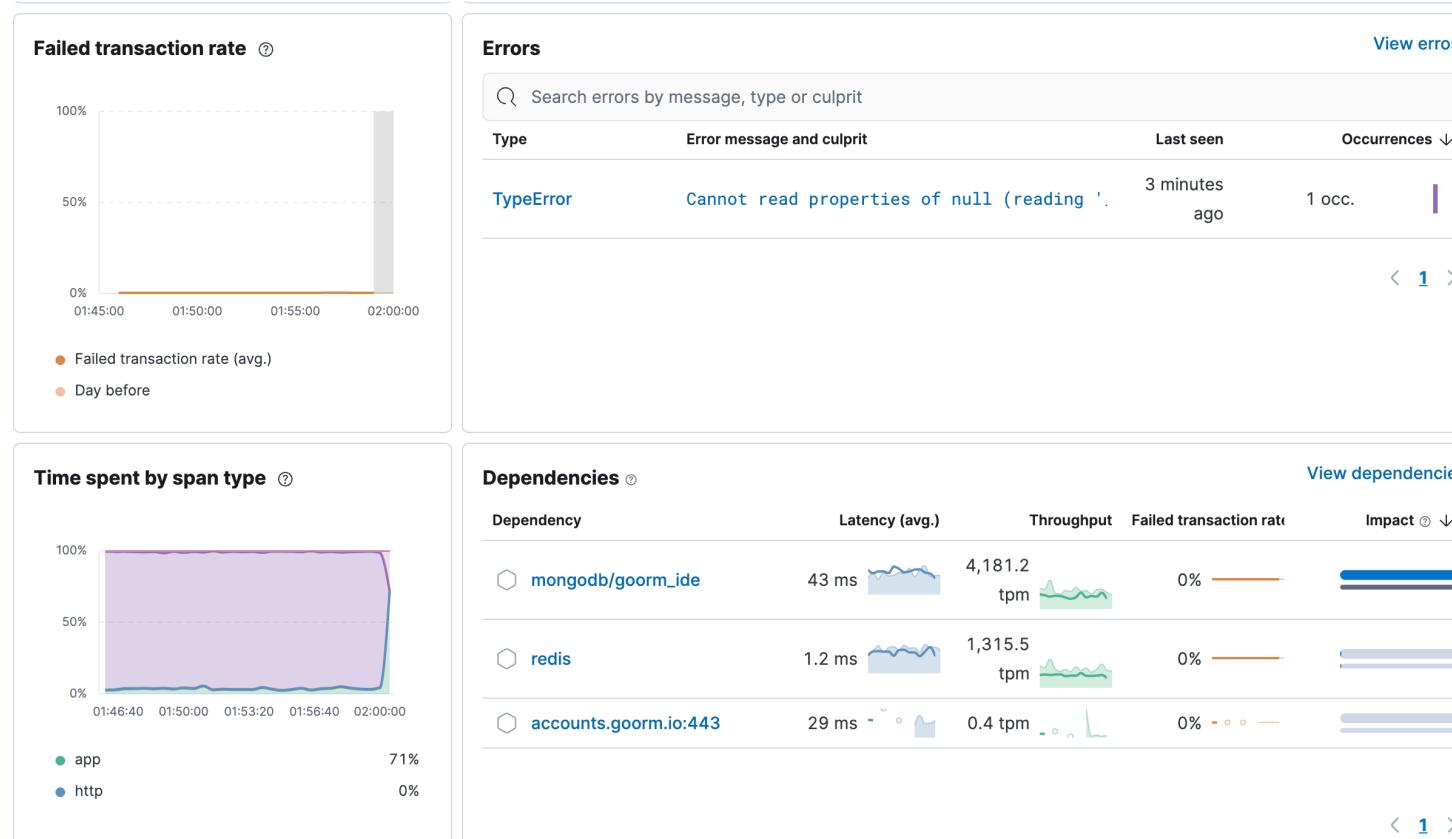
안정적인 서비스 운영을 위하여

- 애플리케이션 모니터링



안정적인 서비스 운영을 위하여

- 애플리케이션 모니터링



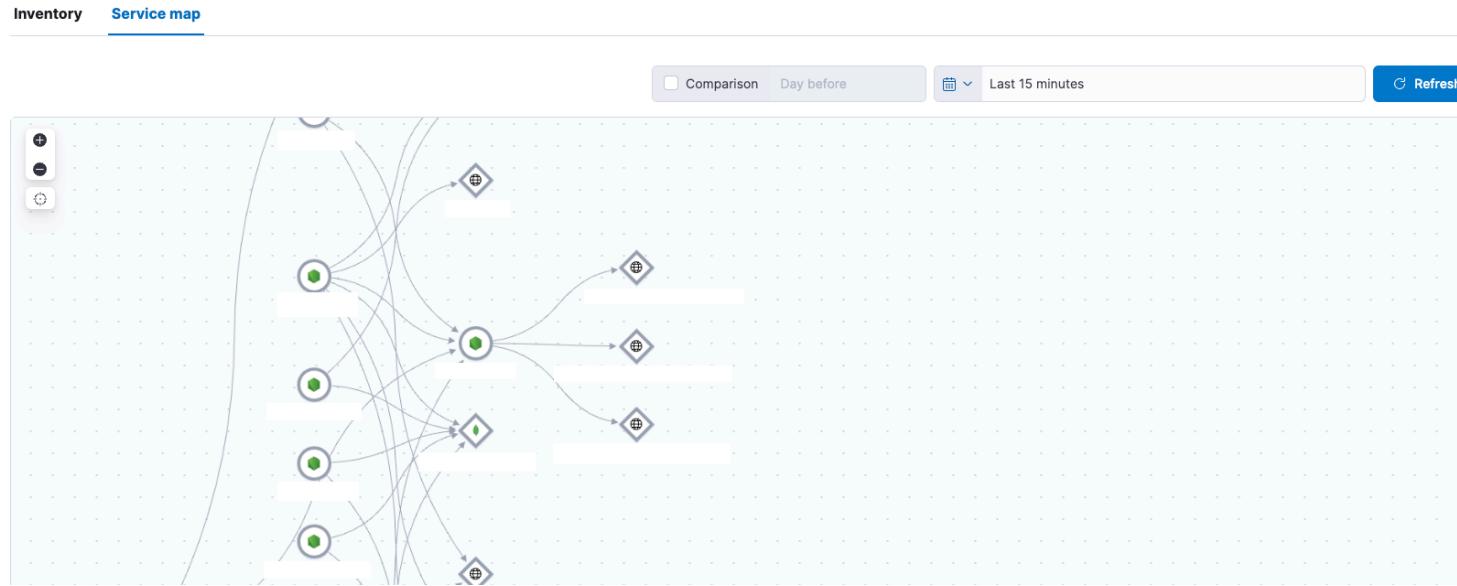
안정적인 서비스 운영을 위하여

- 애플리케이션 모니터링

Top 59 instances							
Node name	Latency (avg.)	Throughput ↓	Failed transaction rate	CPU usage (avg.)	Memory usage (avg.)		
localhost	38 ms	332.9 tpm	0.0%	0.9%	37%	...	▼
54bfaf2113f9	172 ms	7.9 tpm	0%	2.6%	26%	...	▼
cdf24c0352e0	196 ms	7.9 tpm	0%	3.0%	25%	...	▼
ip-10-1-30-122.ap-northeast-2.compute.internal	2.4 ms	7.5 tpm	0%	0.5%	25%	...	▼
ip-10-1-30-148.ap-northeast-2.compute.internal	2.2 ms	7.5 tpm	0%	0.5%	25%	...	▼
< 1 2 3 4 5 ... 12 >							

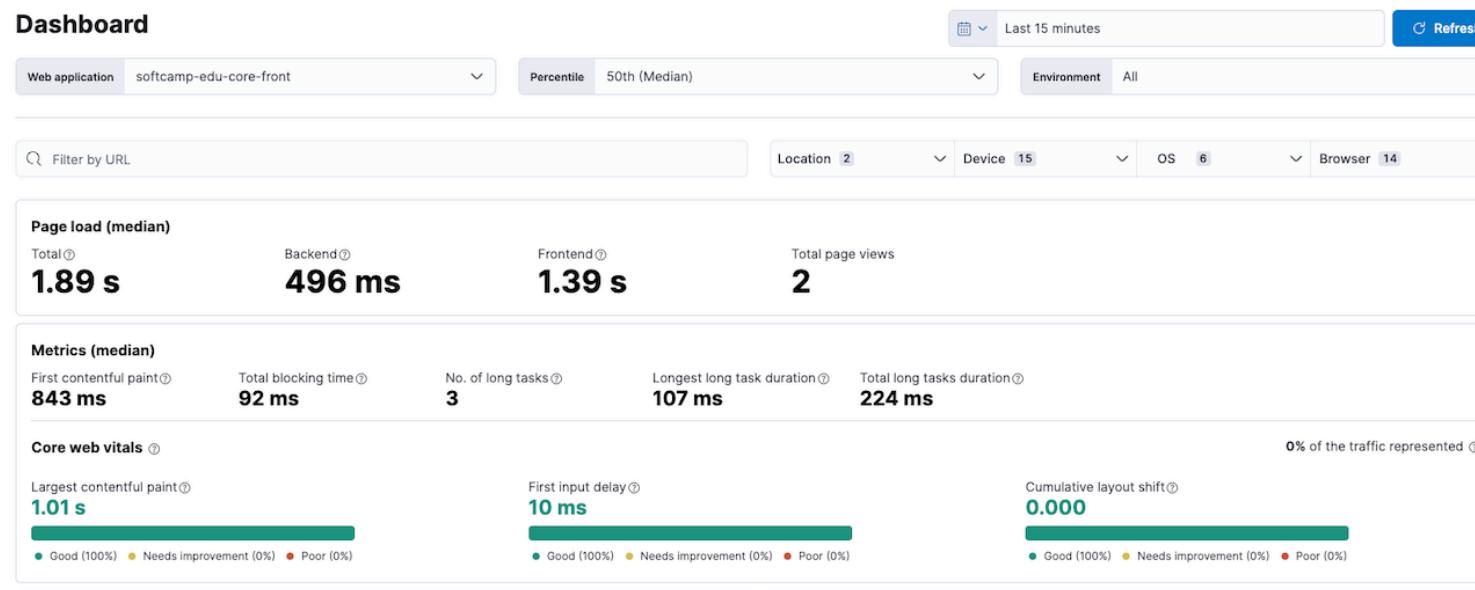
안정적인 서비스 운영을 위하여

- 애플리케이션 모니터링



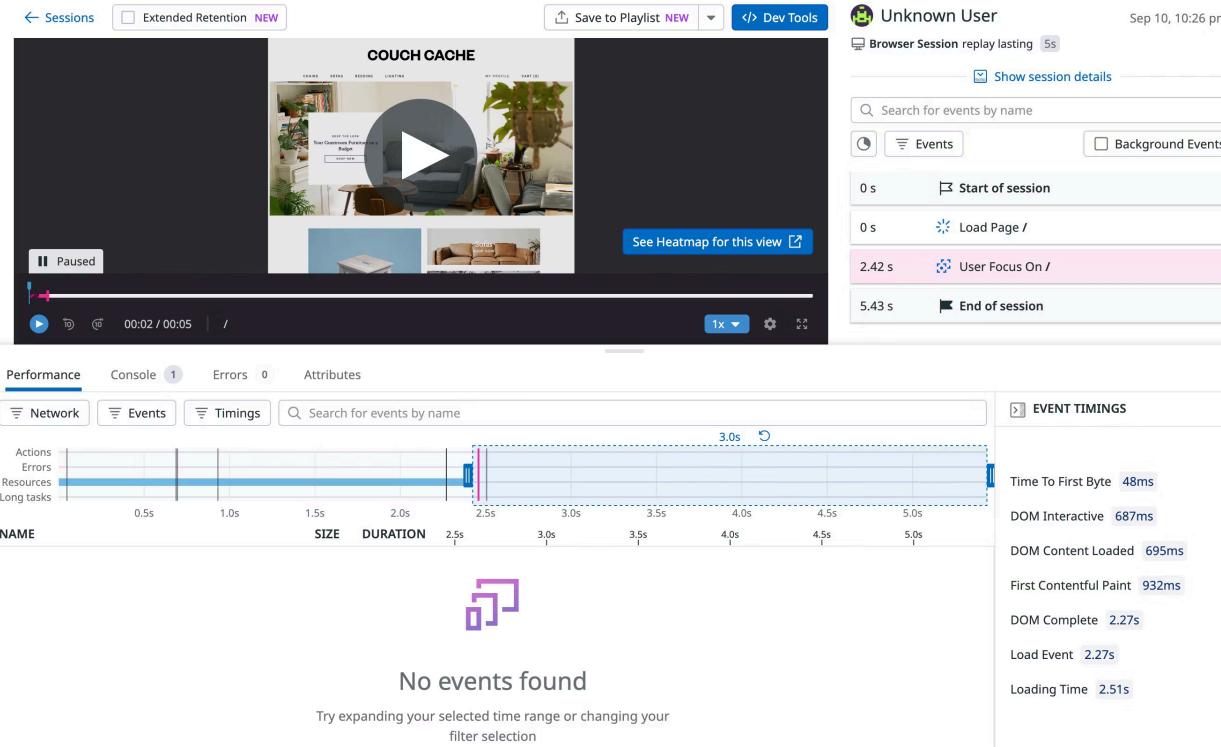
안정적인 서비스 운영을 위하여

- 실제 사용자 모니터링



안정적인 서비스 운영을 위하여

- 실제 사용자 모니터링



안정적인 서비스 운영을 위하여

- 모니터링을 통해 얻을 수 있는 것

- 최적화
 - CPU, 메모리, 디스크 사용량과 같은 지표를 통해 제품이 운영되기 위한 시스템 환경이 적절하게 운영되고 있는지 확인하고 증설/감축에 대한 판단을 할 수 있다.
 - 이를 통해 더욱 질 좋은 서비스를 제공할 수 있을 뿐만 아니라, 자원을 효율적으로 운영하여 비용 절감의 효과도 가져올 수 있다.
 - 문제 발견 및 장애 방지
 - 기존 운영 환경에서 보여지는 것과 다른 이상 징후를 탐지하여 조기에 대응함으로써 장애를 막을 수 있다.
 - 장애가 발생하더라도 즉각적으로 알림을 받아 빠르게 대응 및 복구할 수 있다.
 - 고객 경험 개선
 - 모니터링을 통해 시스템의 가용성이 유지되도록하여 안정적인 서비스를 제공한다.
 - 실제 사용자가 겪고 있는 상황을 모니터링하고, 병목이 생기거나 에러가 많이 발생하는 지점을 빠르게 찾아 수정하여 고객의 사용자 경험을 개선할 수 있다.

안정적인 서비스 운영을 위하여

- **로그 시스템의 필요성**

- 모니터링 시스템만으로 모든 것을 관측할 수 없다
 - 애플리케이션이 개발되면서 발생하는 수많은 문제를 단순히 지표 수치만으로 원인을 파악하여 해결하는 것은 불가능
 - 지표를 통해 문제의 전후를 알더라도, 정확한 원인을 유발한 것이 어떤 것인지 알 수 없다.
 - 모니터링 수치는 결국 결과 이기 때문에, 근본적인 문제 해결을 위해서는 더욱 자세한 정보가 필요다.

- 내부 요구 사항의 다양화

- 새롭게 배포된 기능의 사용성, 사용자 경험에 대한 분석이 필요하다.
 - 개발된 기능의 문제인지, 사용자의 사용 환경에서 실행할 수 없는 기능인지 모니터링만으로는 알기 어렵다.
 - 데이터 기반의 의사결정을 통해 비즈니스에 인사이트를 줄 수 있다.

안정적인 서비스 운영을 위하여

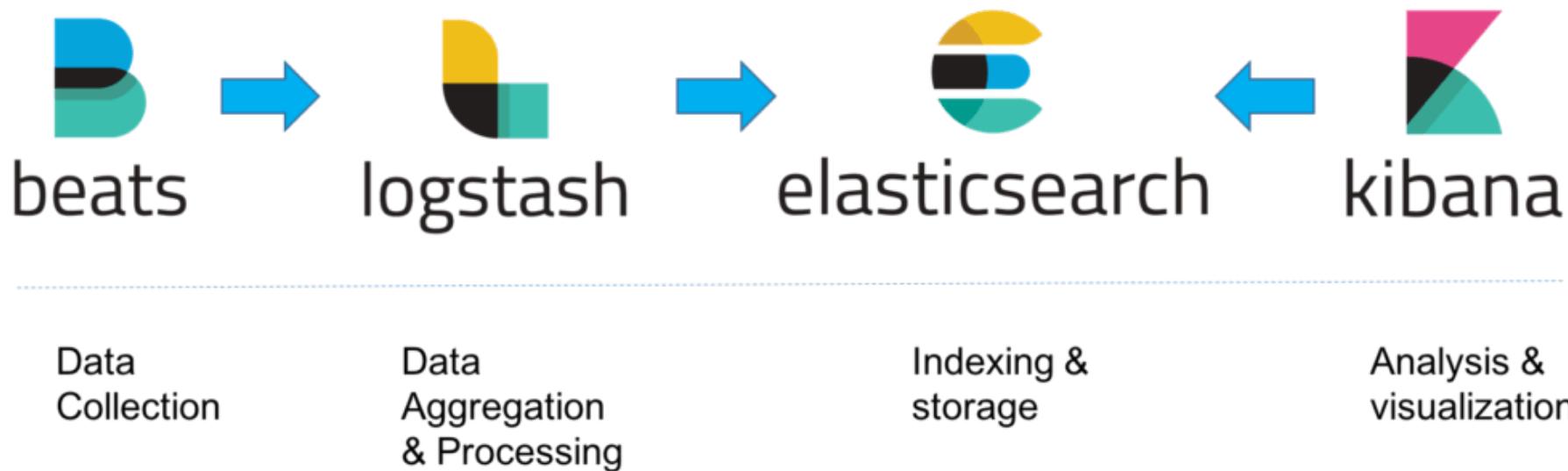
- **로그의 종류**

- 시스템 로그 (System Log)
 - 시스템의 시작 및 종료, 여러 메세지 등 시스템 운영체제에서 발생하는 이벤트를 기록한다.
 - /var/log/syslog, /var/log/kern.log, Windows Event Viewer 등
- 애플리케이션 로그 (Application Log)
 - 애플리케이션에서 발생하는 이벤트를 기록한다. 기록하고자 하는 로그를 정의하여 기록할 수 있다.
 - 웹 서버 로그(Nginx, Apache), 기타 애플리케이션에서 직접 찍는 로그
- 감사 로그 (Audit Log)
 - 시스템에 접속하는 사용자 및 사용자의 활동을 기록한다.
 - 보안이 중요 시 되는 금융 시스템의 접속 기록 로그, 인프라 관리 차원에서 리소스 접속 기록 로그 등.
- etc.

안정적인 서비스 운영을 위하여

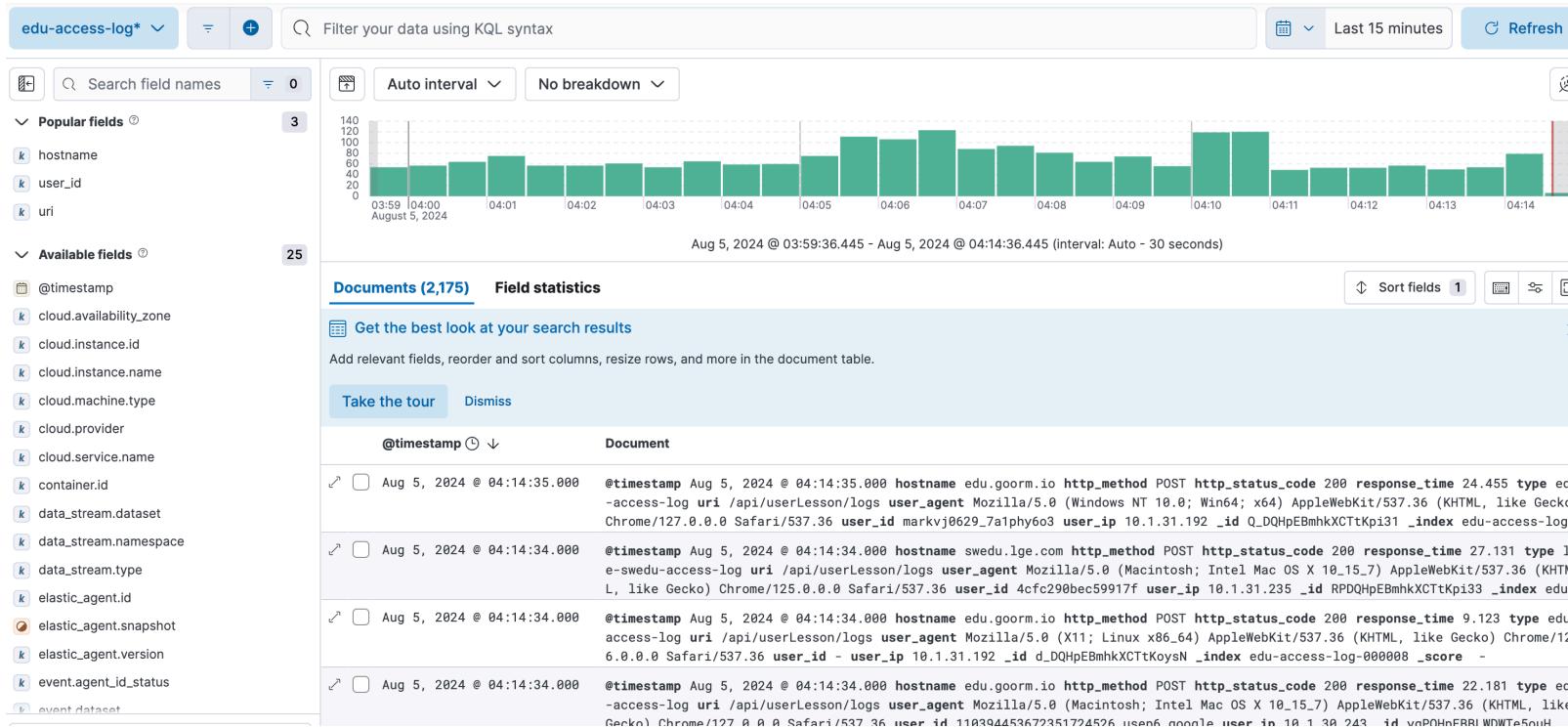
- **ELK Stack**

- ELK(E lasticsearch + L ogstash + K ibana)에 Beats를 추가해서 ELK Stack이라고 불린다.



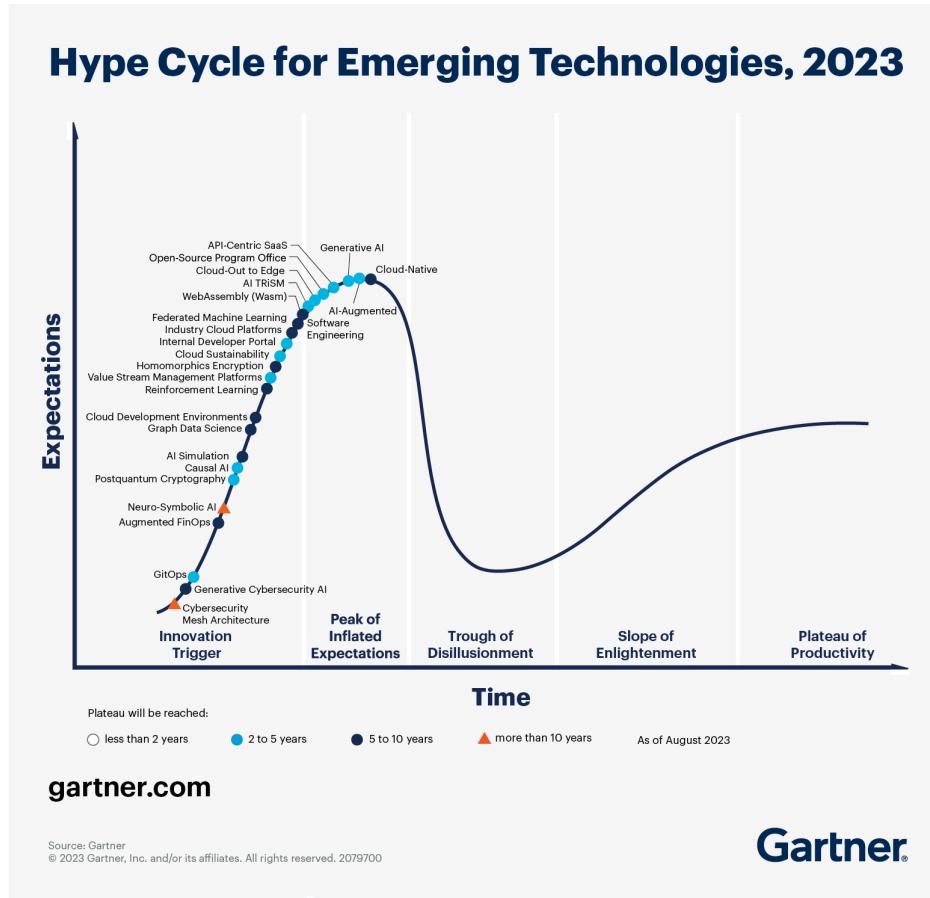
안정적인 서비스 운영을 위하여

- ELK Stack



새로운 기술을 대하는 자세

- 새로운 기술은 날이 갈수록 더 빨리, 더 많이 나온다.



새로운 기술을 대하는 자세

- 우리는 끊임없이 배워나가야 합니다..



새로운 기술을 대하는 자세

- IaC(Infrastructure as Code), 그 이전의 삶
 - 대규모 코딩테스트
 - 수 백대에 달하는 VM, LB 등의 자원을 수동으로 프로비저닝
 - 각 VM의 환경설정과 애플리케이션 설정 확인을 위해 SSH로 일일이 붙어서 확인
 - 부하테스트를 진행할 경우, 부하 명령어를 실행하기 위해 SSH 연결을 유지한 채로 대기
 - 코딩테스트 후, 로그 업로드를 위해서도 마찬가지로 SSH로 붙어서 직접 업로드
 - 구축 사업
 - 고객사의 요청에 따라서 다른 인프라 환경에 구축 업무를 진행하는 경우가 왕왕 있었는데, 이를 매번 수동으로 프로비저닝
 - 애플리케이션에서 사용하는 환경변수에 문제가 발생할 경우 디버깅을 각 연결 지점마다 찾아서 일일이 확인해야함

새로운 기술을 대하는 자세

- IaC(Infrastructure as Code) 도입 이후
 - 다양한 인프라 환경이 사전에 정의된 코드를 통해 프로비저닝됨
 - 투입되는 인력은 줄어들고, 훨씬 빠르게 인프라 구성을 진행하며, 휴먼 에러가 발생할 여지를 최소화하여 에러 발생 빈도도 크게 줄었다.
 - 이렇게 확보된 리소스를 다른 업무를 하기 위해 투입할 수 있다.
 - 대량의 인프라의 환경점검도 손쉽게 진행
 - Ansible을 통해 프로비저닝된 인프라에 손쉽게 접근하여 작업을 진행

새로운 기술을 대하는 자세

- IaC(Infrastructure as Code) 장점만 있을까?
 - 한 번의 잘못된 코드 작성이 큰 문제를 일으킬 수 있다.
 - 손쉽게 프로비저닝할 수 있기에 자칫 비용 증가로 이어질 수 있다.
 - 비정상적인 동작으로 서비스에 장애를 초래할 수 있다.
 - IaC 코드 관리의 문제
 - 지속적인 유지보수를 진행하지 않으면 기술 부채로 이어질 수 있다.
 - 러닝커브가 높기에 새롭게 합류하는 구성원이 이를 익히고 사용하는데 오랜 시간이 걸릴 수 있다.
 - 인프라 환경이 복잡할수록 이를 표현하는 코드 또한 복잡해지기에 이를 관리할 때, 실수나 버그가 발생할 확률이 커진다.
- IaC 도구의 의존성
 - 특정 도구 혹은 특정 CSP에 의존성이 생길 수 있다.

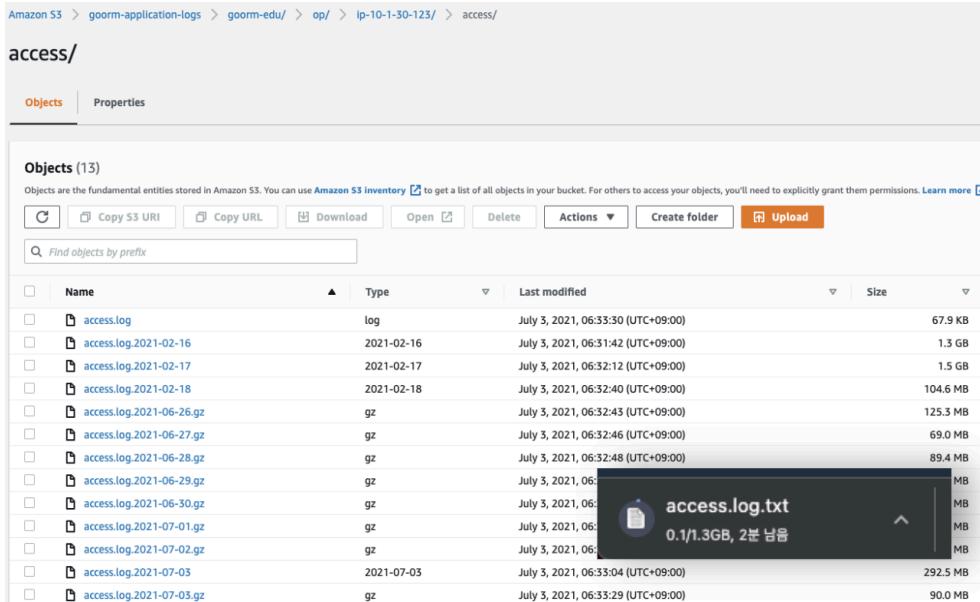
새로운 기술을 대하는 자세

- IaC(Infrastructure as Code) 장점만 있을까?

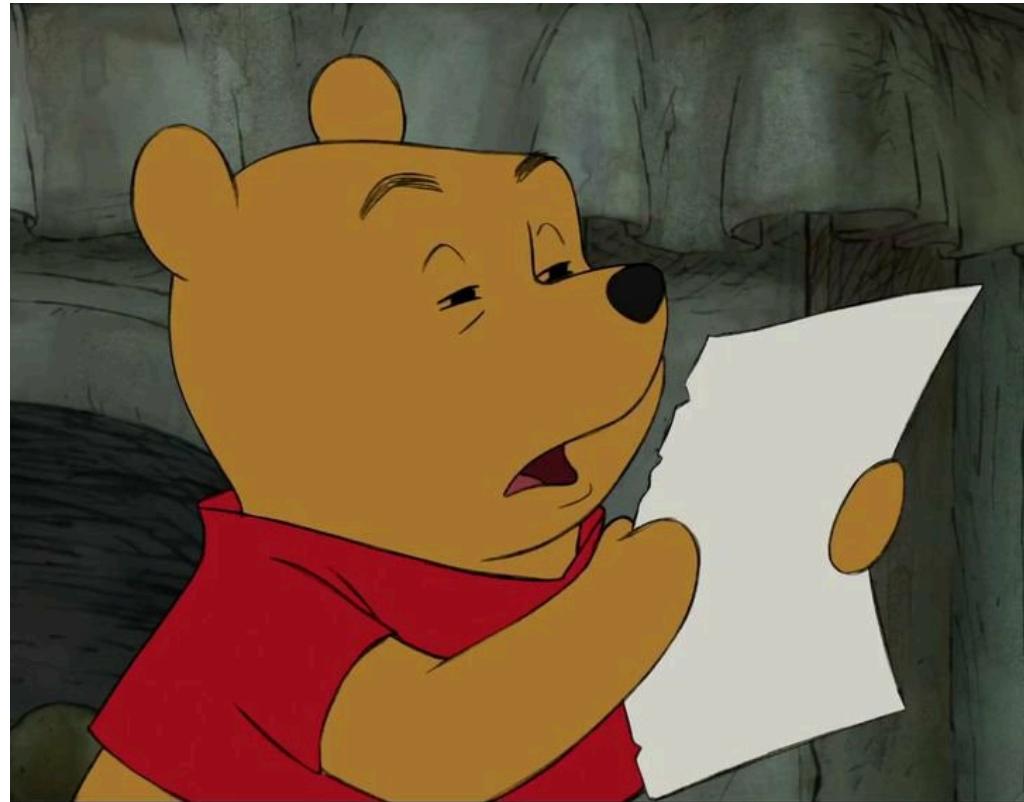


새로운 기술을 대하는 자세

- 로그 시스템 개선
 - 필요할 때 다운받아서 파일을 직접 열어 봐야했다.

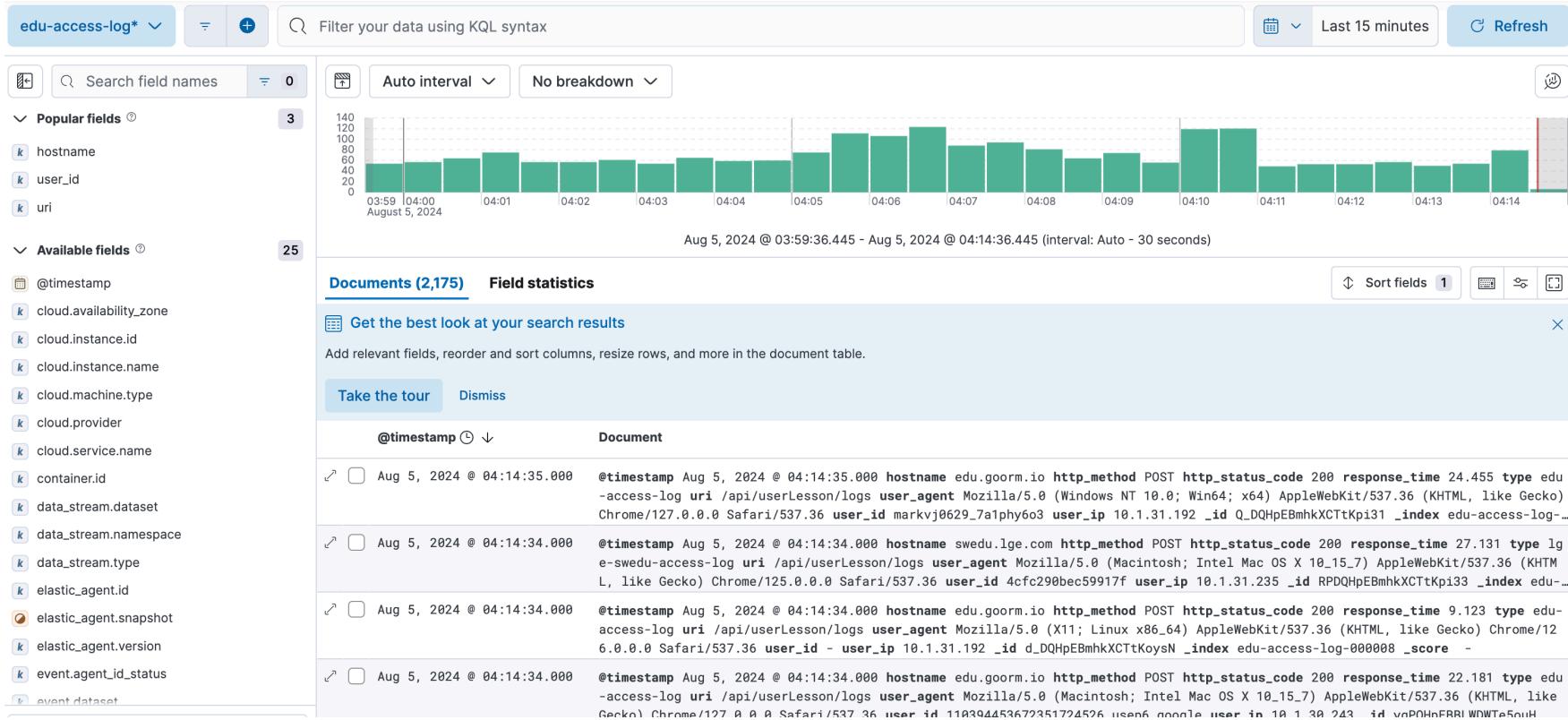


새로운 기술을 대하는 자세



새로운 기술을 대하는 자세

- 로그 시스템 개선



새로운 기술을 대하는 자세

- AI

- 대세 기술로 자리 잡았으며 다양한 서비스들이 존재한다.
 - ex) chatGPT, claude, gemini, sora
- 이미 다양한 영역에서 사람이 하는 것보다도 훨씬 뛰어난 결과물과 효율을 보여준다.



새로운 기술을 대하는 자세

- 어차피 따라잡을 수 없다면, 효율적으로 사용하자
 - 역사적으로 새로운 도구의 발견 시기에는 항상 혼란스러웠다.
 - 산업혁명으로 인한 기계화
 - 디지털 혁명
 - 인공지능과 자동화의 발전

새로운 기술을 대하는 자세

- 사내 LLM - Claude

Node.js 앱에서 MongoDB 첫 연결시에는 연결이 안되는데, 단순히 앱을 재시작하면 연결이 되는데 어떤 문제가 있는걸까?

이런 문제가 발생하는 데에는 여러 가지 원인이 있을 수 있습니다. 주요 원인과 해결 방법을 살펴보겠습니다.

1. 비동기 연결 문제:

MongoDB 연결은 비동기적으로 이루어집니다. 첫 연결 시 연결이 완료되기 전에 쿼리를 실행하려고 하면 오류가 발생할 수 있습니다.

해결방법: 연결이 완료된 후에 앱을 시작하도록 코드를 구성합니다.

```
mongoose.connect(uri).then(() => {
  console.log('Connected to MongoDB');
  app.listen(3000, () => console.log('Server started'));
}).catch(err => console.error('Failed to connect to MongoDB', err));
```

2. 네트워크 지연:

첫 연결 시 네트워크 지연으로 인해 연결 시간 초과가 발생할 수 있습니다.

해결방법: 연결 옵션에서 타임아웃 값을 늘립니다.

```
mongoose.connect(uri, { serverSelectionTimeoutMS: 5000 });
```

새로운 기술을 대하는 자세

- 사내 LLM - Elasticsearch

The screenshot shows a web-based interface titled "EDU-Core APM 서비스 개선 제안". It displays two suggested queries from an "Elastic Assistant".

1. 트랜잭션 성능 분석:

```
FROM traces-apm*
| WHERE processor.event == "transaction"
| STATS avg_duration = AVG(transaction.duration.us), max_duration =
MAX(transaction.duration.us), count = COUNT(*) BY service.name, transaction.name
| SORT avg_duration DESC
| LIMIT 10
```

[Display results](#) [Visualize this query](#)

이 쿼리는 서비스별, 트랜잭션별 평균 및 최대 지속 시간을 보여줍니다. 결과를 확인한 후 가장 느린 트랜잭션들을 최적화하는 것이 첫 번째 개선점이 될 수 있습니다.

2. 오류 분석:

```
FROM logs-apm*
| WHERE processor.event == "error"
| STATS error_count = COUNT(*) BY service.name, error.grouping_name
| SORT error_count DESC
| LIMIT 10
```

[Display results](#) [Visualize this query](#)

Send a message to the Assistant

새로운 기술을 대하는 자세

- 새로운 것과 옛 것의 충돌
 - 레거시 시스템은 존재할 수 밖에 없다.
 - 적절한 유지보수가 이뤄지지 않은 것은 잘못일 수 있으나, 레거시 시스템 자체는 잘못이 없다.
 - 작성 당시에는 합당한 근거가 있었다. (물론 아닐 수 있습니다.)
 - 그러니 너무 미워하지 말아주세요..
 - 새로운 기술이 항상 정답은 아니다.
 - 서비스의 종류에 따라선, 안정성이 더욱이 중요시 되는 시스템이 있을 수 있다. 이 때, 검증이 덜 된 기술을 선불리 사용할 수 있을까?
 - 이식성이 떨어질 수 있다. 어떤 플랫폼에선 가능하고, 또 다른 플랫폼에서는 불가능한 기능을 개발하게 된다면?
 - 온고지신
 - 레거시의 점진적 개선과 이와 함께 신기술 도입이 어우러져야 한다.

마치며

- 부트캠프를 진행하면서 얻은 인연을 소중히 해주세요.
- 강의를 통해 배운 것을 내스킬로 가져가주세요.
- 부트캠프에서 한 걸음 만 더 나아가주세요.

마치며

