



AWS한국사용자모임
#Beginner
소모임

AWSKRUG 비기너 모임

유저 행동 로그 플랫폼 개발기

MSK와 EKS를 활용한 사용자 행동 로그 구축 사례

AWSKRUG 이한섭

CONTENTS

1. 들어가기 앞서

- 발표자 소개
- 유저 행동 로그 플랫폼의 시작
- 데이터기반 의사결정의 중요성
- 예상되는 문제점

2. 아키텍처 개요

- 서비스 운영 및 설계 간 고려사항
- MSK와 EKS를 통합한 아키텍처 설명
- 데이터 수집 아키텍처
- 데이터 가공 아키텍처

3. 결과 및 이점

- 사용자 행동 로그 구축의 결과와 비즈니스 이점
- 사용자 행동 로그의 확장 및 향후 계획

4. 결론 요약



AWS한국사용자모임
#Beginner
소모임

1. 발표자 소개

1. 발표자 소개

Dominic 이한섭

“내가 틀릴 수도 있습니다”

DevOps Engineer @ 백엔드개발본부 DevOps팀

E dominic@millie.town

M 01092640214

(주)밀리의서재

서울특별시 마포구 양화로 45, 16층 (메세나폴리스 세아타워)



독서와
무제한
친해지리

"시스템의 수명은 observability와 함께한다."



AWS한국사용자모임
#Beginner
소모임

1. 유저 행동 로그 플랫폼의 시작

1. 유저 행동 로그 플랫폼의 시작

신규 서비스를 출시하고 난 뒤, 이 서비스가 사용자들에게 어떤 패턴으로 어떻게 이용되고 있을까?



1. 유저 행동 로그 플랫폼의 시작

주관적인 판단 하에 비즈니스 의사결정을 내리기 위해서는 어떤 근거가 필요할까?



1. 유저 행동 로그 플랫폼의 시작

사용자가 불편한 경험을 하기 전에 우리가 먼저 불편함을 이해하고 개선할 순 없을까?



1. 유저 행동 로그 플랫폼의 시작

사용자의 애플리케이션 이용 패턴을 추적하고 분석해야겠다.
아주 상세하게



1. 유저 행동 로그 플랫폼의 시작

구축효과

1. A to Z 회원의 이동 경로 파악 가능
2. 마케팅/이벤트 페이지 내 진입 경로 및 결제 성공까지의 여정을 알 수 있음
3. 특정 배너/섹션을 통한 진입 여부를 판단할 수 있음
4. UI/UX 행동이력 파악 가능
5. 결제 페이지 내 탈출 원인과 해지 방어에 기여할 수 있는 데이터 기반 의사결정 가능
6. 퍼져있는 트래킹/로그 데이터를 통합
7. 피드/나우/이벤트 배너 트래킹 등 로그 데이터를 통합
8. 매번 생기는 프로덕트마다 트래킹/로그 데이터 수집 공수를 들이지 않도록 개선
9. 클라이언트 타임으로 인해 트래킹 ID 별 체류시간을 얻을 수 있음

1. 유저 행동 로그 플랫폼의 시작

구축효과

1. A to Z 회원의 이동 경로 파악 가능
2. 마케팅/이벤트 페이지 내 진입 경로 및 결제 성공까지의 여정을 알 수 있음
3. 특정 배너/섹션을 통한 진입 여부를 판단할 수 있음
4. UI/UX 행동이력 파악 가능
5. 결제 페이지 내 탈출 원인과 해지 방어에 기여할 수 있는 데이터 기반 의사결정 가능
6. 퍼져있는 트래킹/로그 데이터를 통합
7. 피드/나우/이벤트 배너 트래킹 등 로그 데이터를 통합
8. 매번 생기는 프로덕트마다 트래킹/로그 데이터 수집 공수를 들이지 않도록 개선
9. 클라이언트 타임으로 인해 트래킹 ID 별 체류시간을 얻을 수 있음



AWS한국사용자모임
#Beginner
소모임

1. 데이터기반 의사결정의 중요성

1. 데이터기반 의사결정의 중요성



1. 데이터기반 의사결정의 중요성

01

사용자 행동로그를 통한 사용자 이해

02

사용자 행동로그를 통한 데이터 기반 의사결정

03

제품 및 서비스 개선 (개인화된 서비스 제공)

선순환 구조

데이터 기반 의사결정은
지속적인 성장과 혁신을
이끈다.

1. 예상되는 문제점

예상 문제점

- 밀리에서 처음으로 도입하는 소프트웨어가 다수 존재하기에, 러닝커브 발생 예상
 - 데이터팀 주도 하에 러닝커브 완화를 위한 API 설계 가이드를 제공
- 프로덕트가 생길 때 마다 가공/BI 플랫폼 적용이 필요
 - 유동적으로 적재가 가능하도록 파서를 개발하여 리소스를 줄일 수 있음
- 대규모 트래픽이 발생하면, 서버에 부하를 줄 수 있음
 - 키네시스 로그 수집량 * 3 → 일별 1200만 Row, 시간당 22~28만 Row, 초당 57건 예상
 - 인프라팀 살려주세요

1. 예상되는 문제점

예상 문제점

- 밀리에서 처음으로 도입하는 소프트웨어가 다수 존재하기에, 러닝커브 발생 예상
 - 데이터팀 주도 하에 러닝커브 완화를 위한 API 설계 가이드를 제공
- 프로덕트가 생길 때 마다 가공/BI 플랫폼 적용이 필요
 - 유동적으로 적재가 가능하도록 파서를 개발하여 리소스를 줄일 수 있음
- 대규모 트래픽이 발생하면, 서버에 부하를 줄 수 있음
 - 키네시스 로그 수집량 * 3 → 일별 1200만 Row, 시간당 22~28만 Row, 초당 57건 예상
 - 인프라팀 살려주세요

웹 앱에서 이벤트캐치 스크립트를 통해 요청을 전부 user-act-ingress로 보내자.

일별 12,000,000 ROW는 API Gateway로는 처리할 수 없다.
→ 백엔드 개발팀에 협업요청 필요

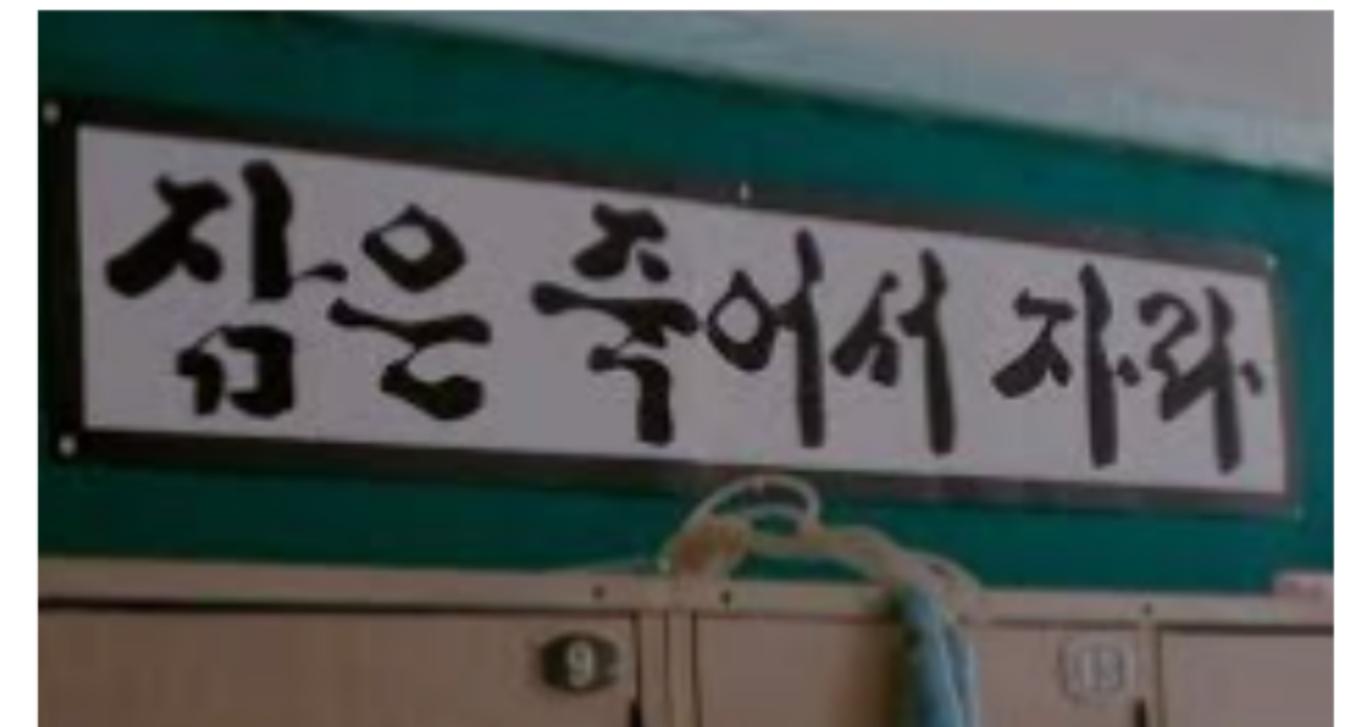
1. 예상되는 문제점

예상 문제점

- 밀리에서 처음으로 도입하는 소프트웨어가 다수 존재하기에, 러닝커브 발생 예상
 - 데이터팀 주도 하에 러닝커브 완화를 위한 API 설계 가이드를 제공
- 프로덕트가 생길 때 마다 가공/BI 플랫폼 적용이 필요
 - 유동적으로 적재가 가능하도록 파서를 개발하여 리소스를 줄일 수 있음
- 대규모 트래픽이 발생하면, 서버에 부하를 줄 수 있음
 - 키네시스 로그 수집량 * 3 → 일별 1200만 Row, 시간당 22~28만 Row, 초당 57건 예상
 - 인프라팀 살려주세요

웹 앱에서 이벤트캐치 스크립트를 통해 요청을 전부 user-act-ingress로 보내자.

일별 12,000,000 ROW는 API Gateway로는 처리할 수 없다. (1400\$ 이상)
→ 백엔드 개발팀에 협업요청 필요

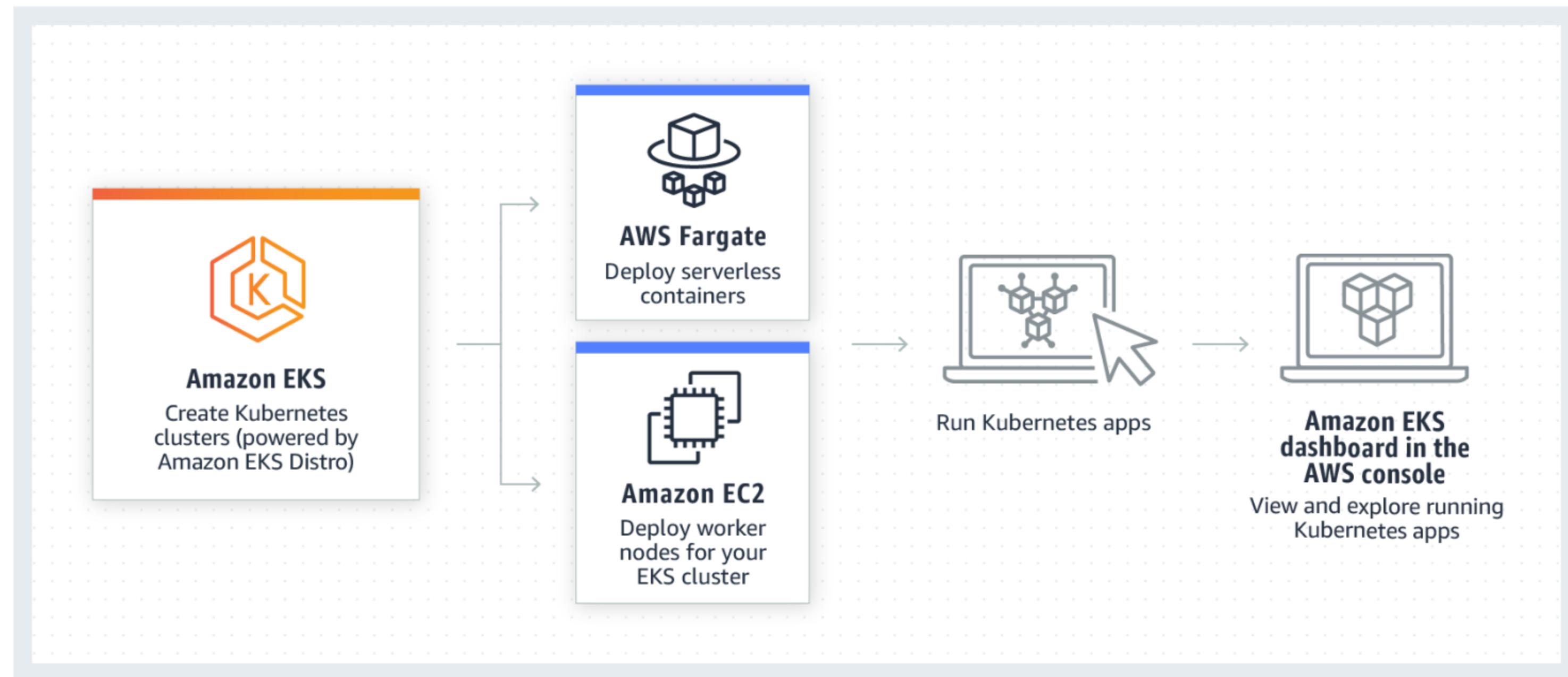




AWS한국사용자모임
#Beginner
소모임

2. 아키텍처 개요

2. 아키텍처 개요에 앞서 FARGATE란...



2. 아키텍처 개요에 앞서 FARGATE란...

Pods가 Fargate에 예약되면 Pod 사양 내의 vCPU 및 메모리 예약에 따라 Pod에 프로비저닝할 CPU 및 메모리 양이 결정됩니다. Init 컨테이너의 최대 요청은 Init 요청 vCPU 및 메모리 요구 사항을 결정하는 데 사용됩니다. Fargate는 필요한 Kubernetes 구성 요소(kubelet, kube-proxy, containerd)에 대한 각 Pod의 메모리 예약에 256MB를 추가합니다.

- Fargate Spec

vCPU 값	메모리 값
.25 vCPU	0.5GB, 1GB, 2GB
.5 vCPU	1GB, 2GB, 3GB, 4GB
1 vCPU	2GB, 3GB, 4GB, 5GB, 6GB, 7GB, 8GB
2 vCPU	4~16GB(1GB 증분)
4 vCPU	8~30GB(1GB 증분)
8 vCPU	16~60GB(4GB 단위)
16 vCPU	32~120GB(8GB 단위)

- Fargate Storage

기본적으로 파드당 20GB를 제공 (fargate비용 내 포함)
2023년 8월 1일부로 파드 당 175GB까지 증설 가능 (별도비용)

ex) 50GB가 필요한 두개의 컨테이너가 있는 파드를 생성하면
요청량 + 각 컨테이너 스토리지의 10%를 예약 + 시스템 구성용 5GiB
100GB가 아닌, 115GB를 할당

- 그 외 특이사항

Host기반의 모든 기능이 불가 (daemonset, HostPort, HostNetwork)
Privileged Container 불가, GPU불가, public fargate 불가

2. 아키텍처 개요

서비스 운영 및 설계 간 고려사항

DATA팀

- 수집 데이터 정의
- DATA 수집, 가공 아키텍쳐 설계
- 이벤트 및 사용자 속성 정의
- 데이터 정합성 검토

DevOps팀

- 필요 리소스(서비스) 산정 및 비용 검토
- 적절한 아키텍처 설계 및 구축 (EKS? EC2? ECS?)
- 부하테스트
- 배포 파이프라인 및 모니터링 시스템 추가

BE팀

- Stream Producer API 개발

FE팀

- 웹/앱 이벤트 캐치 스크립트
- 이벤트 Class 정의/정리 및 부여
- 세션 별 Tracking ID 발급 로직 개발
- Buffer Flush Trigger (세션만료) 스크립트

2. 아키텍처 개요

서비스 운영 및 설계 간 고려사항 - 필요 리소스 산정 및 검토



- EKS 클러스터를 n대 운영중
 - 환경별 Data, Common, Airflow, Opeartion, api 등등
- EKS의 worknode는 총 3개의 속성으로 운영중
 - Nodegroupless, Karpenter, EC2
- 효율적인 자원의 재분배를 위한 작업을 항상 진행중
- 레거시 → 차세대 아키텍처로 진화 중

2. 아키텍처 개요

서비스 운영 및 설계 간 고려사항 - 필요 리소스 산정 및 검토



- EKS 클러스터를 n대 운영중
 - 환경별 Data, Common, Airflow, Opeartion, api 등등
- EKS의 worknode는 총 3개의 속성으로 운영중
 - Nodegroupless, Karpenter, EC2
- 효율적인 자원의 재분배를 위한 작업을 항상 진행중
- 레거시 → 차세대 아키텍처로 진화 중

이미 줄일만큼 줄여봤다.

FinOps를 한다는 것은 인프라를 축소시키는 것이 아닌, 효율적인 자원의 재분배



2. 아키텍처 개요

서비스 운영 및 설계 간 고려사항 - 필요 리소스 산정 및 검토

stateless하기에 파드의 라이프사이클에 큰 제약이 없다.

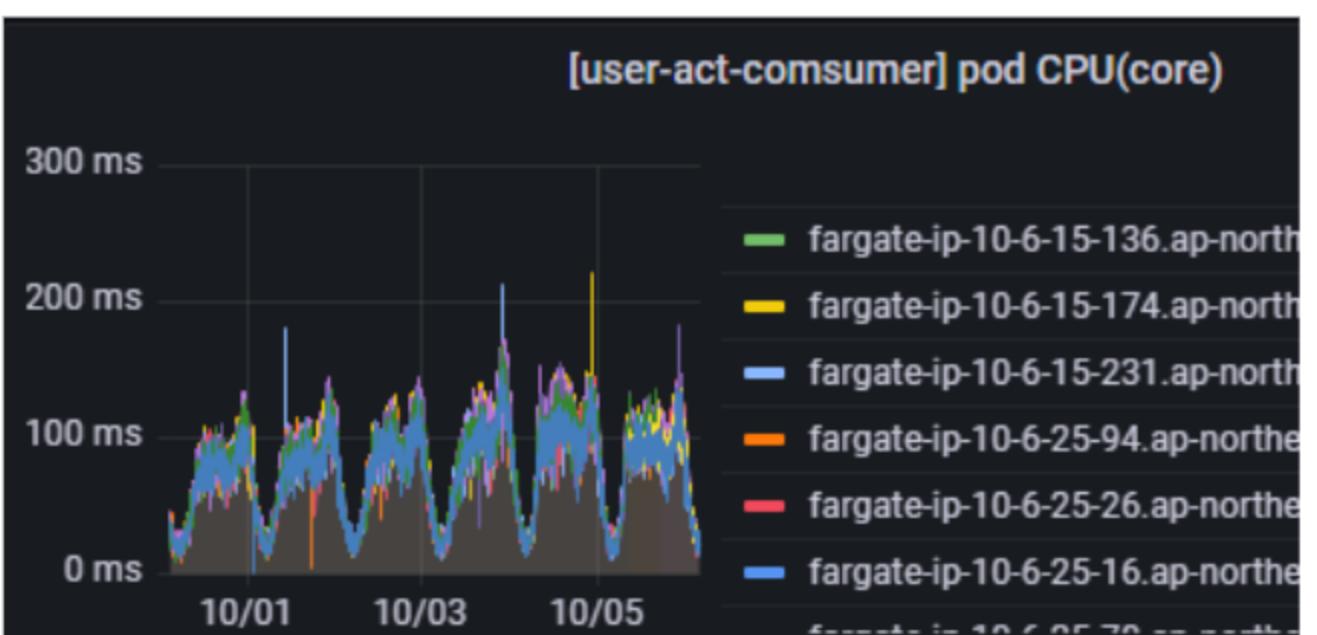
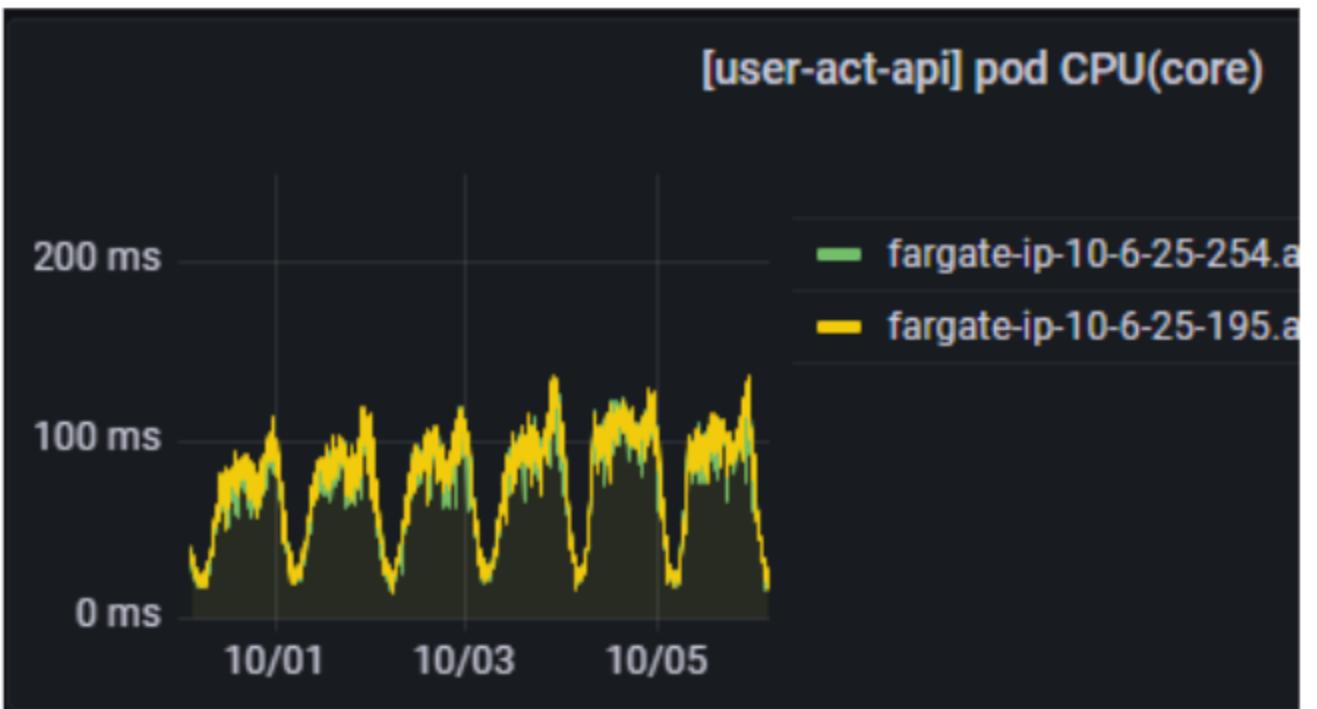
파드가 순식간에 프로비저닝되어야하는 기민성을 가지고 있지 않다.

- Karpenter패턴과 fargate를 적절히 활용

리소스가 합리적으로 소모되고 있는가 -> 작은 pod들에 사용하기엔 스펙의 유연함이 부족하다.

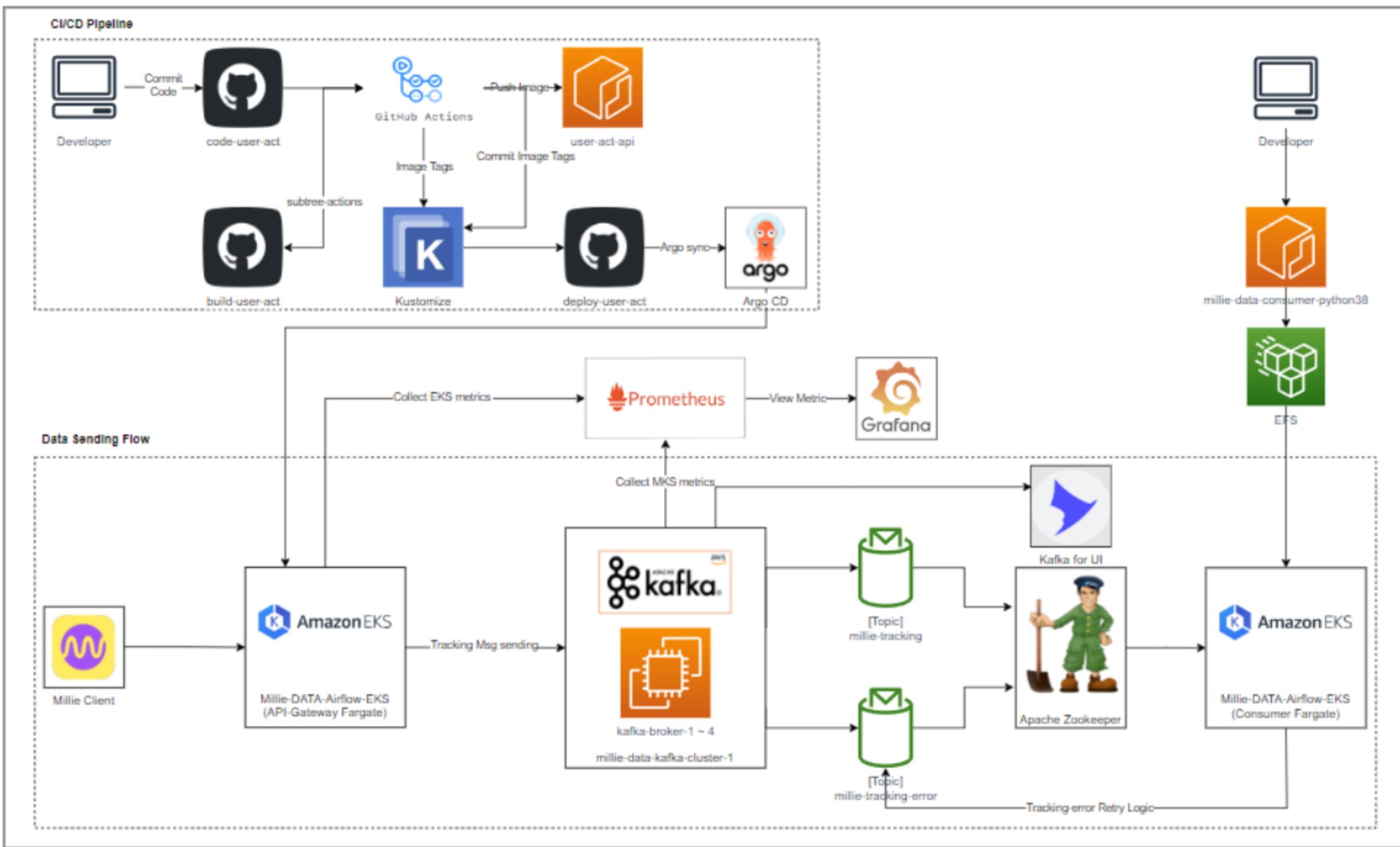
부하 테스트 (Pod CPU Utilization per 1m)

구분	min	max
User-act-ingress	2500 (Request per min)	20000 (Request per min)
Producer API(HPA 2~4)	40ms	150ms
Consumer API (P:C 1대1비율)	40ms	140ms

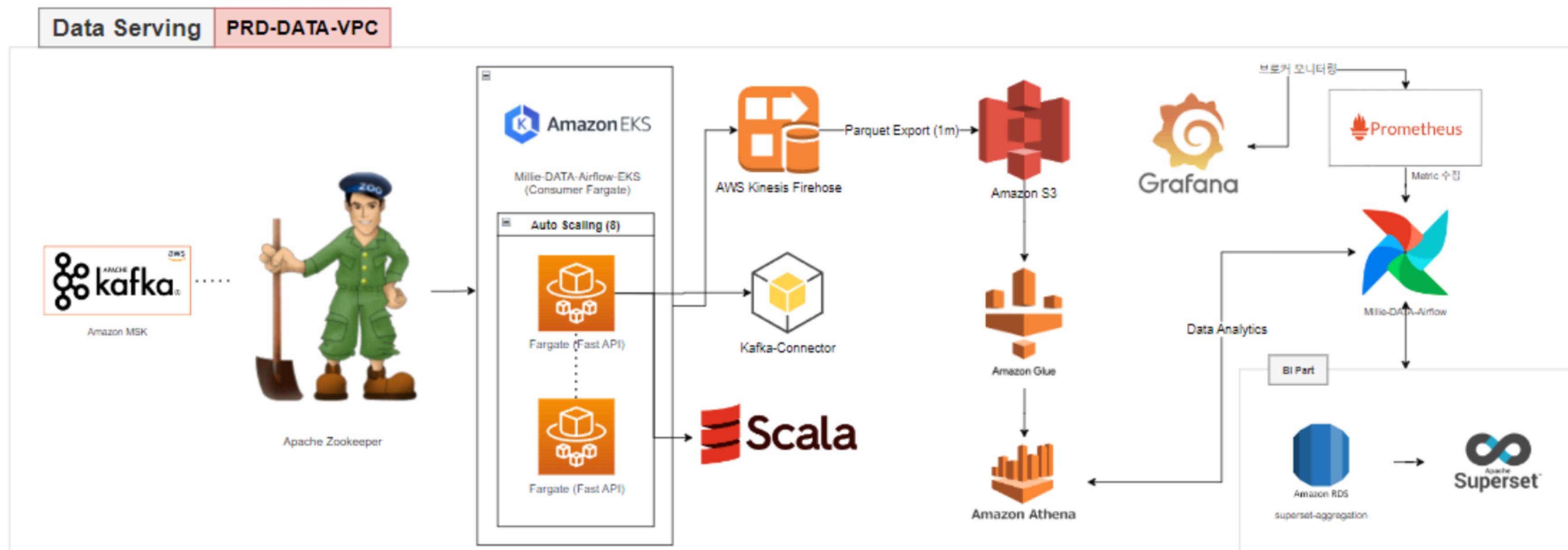


2. 아키텍처 개요 - 데이터 수집

Sending Event PRD-VPC



2. 아키텍처 개요 - 데이터 가공





AWS한국사용자모임
#Beginner
소모임

03 결과 및 이점

3. 결과 및 이점

1. 의사결정의 기반을 잡다

고객의 서비스 사용패턴, 머문시간, 이용 도서, 접근 경로 등을 분석하여 데이터 기반의 의사결정을 실천하고 있습니다.
명확한 성과와 근거를 가지고 동료들이 업무를 진행할 수 있게 되었습니다.

== 더 많은 CRM 발굴과 그로인한 고객만족도 향상효과를 기대해봅니다.



3. 결과 및 이점

2. 동료들의 궁금증

??? : 서비스인데 노드가 왜 생기나요?

A : 개발자가 서버를 관리할 필요가 없는 서비스라는 뜻입니다. 진짜 서버가 없는게 아님!

??? : 종종 컨슈머 리밸런싱이 일어나서 통합로그 플랫폼이 불안정해요!

A : max.partition.fetch.bytes 값을 줄이고 세션 제한 시간을 조정했습니다.

단일 poll()이 반환하는 데이터의 양이 많으면 소비자가 데이터를 처리하는데,

시간이 오래 걸릴 수 있기에 max.partition.fetch.bytes 값을 줄이는 대신 잣은 poll()을 하는 걸로 변경했습니다.



결론



참고사항

키워드 및 레퍼런스 정리

- Fargate 관련

<https://github.com/firecracker-microvm/firecracker>

<https://aws.amazon.com/ko/blogs/korea/firecracker-lightweight-virtualization-for-serverless-computing/>

<https://aws.amazon.com/ko/blogs/containers/under-the-hood-fargate-data-plane/>

- kafka broker관련

<https://dattell.com/data-architecture-blog/optimizing-kafka-brokers/>

THANK YOU!