Platform Engineering

BOAZ MLOps 소모임

발표자 소개

김수빈

- BOAZ 20기 엔지니어링 부문
- 백엔드 플랫폼 개발
- 주로 Python과 Go를 사용합니다.

github.com/sudosubin

목차

- Platform Engineering이란?
- DevOps → Platform Engineering
- FinOps → Platform Engineering
- MLOps → Platform Engineering
- 결론

Platform Engineering이란?

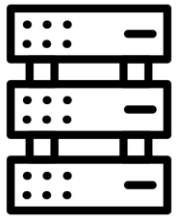
Platform Engineering이란?

- 개발 팀을 위한 도구를 지원하고, 워크플로우를 구축하는 분야
- 서비스 개발(Product Engineering)을 더 잘 할 수 있도록 지원함

아주 먼 옛날

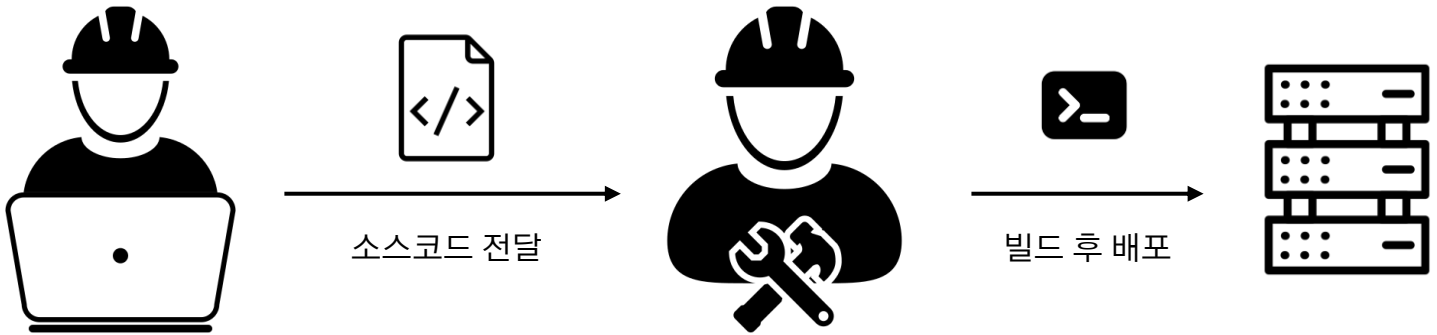
개발 팀

인프라 팀

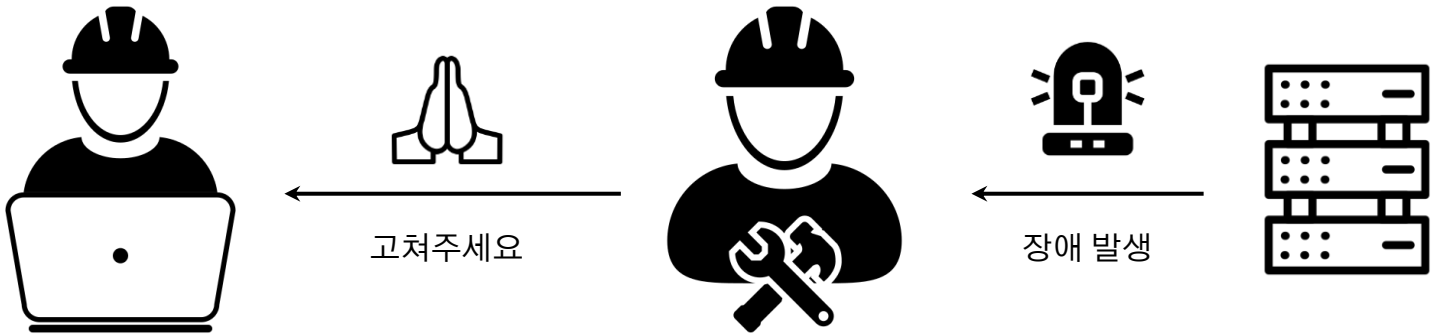


서버

아주 먼 옛날



아주 먼 옛날

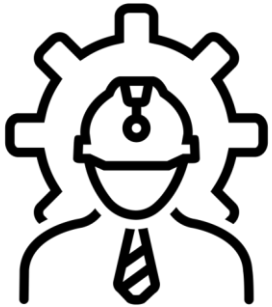


요즘에는

개발 팀

CI/CD 파이프라인
DB 생성
Redis 생성
큐 생성
CDN 설정
로그 주세요
...

보안 취약점
장애 알림
하드웨어 업그레이드
...



운영 팀 (플랫폼 팀)

요즘에는

- 개발 팀의 요청
- 운영 팀 (플랫폼 팀) 요청을 확인함
  - ○ 문제가 있는 요청이라면 이유와 함께 반려
  - ○ 수용 가능한 요청은 처리한 후 결과 안내
- 스크립트를 돌리거나, 직접 콘솔에서 작업하거나, 메신저로 알려주거나

실제로 해보니

- 개발 팀의 확장, 서비스의 확장
- 늘어나는 다양한 요청과 늘어나는 장애
- 하루종일 요청을 처리하고, 장애 대응하고, 운영 하느라 일을 못 함

문제를 해결하기 위해서는

- 자동화가 필요함
  - 자주 들어오는 요청은 플랫폼 팀 없이도 자동으로 처리될 수 있어야 함
- 개발자가 알아서 할 수 있어야 함 (Self Service)
  - 그러면서도 안전하고 문제가 없어야 하고 (보안, 권한, 장애)
  - 개발자가 혼자서도 쉽고 빠르게 할 수 있어야 함

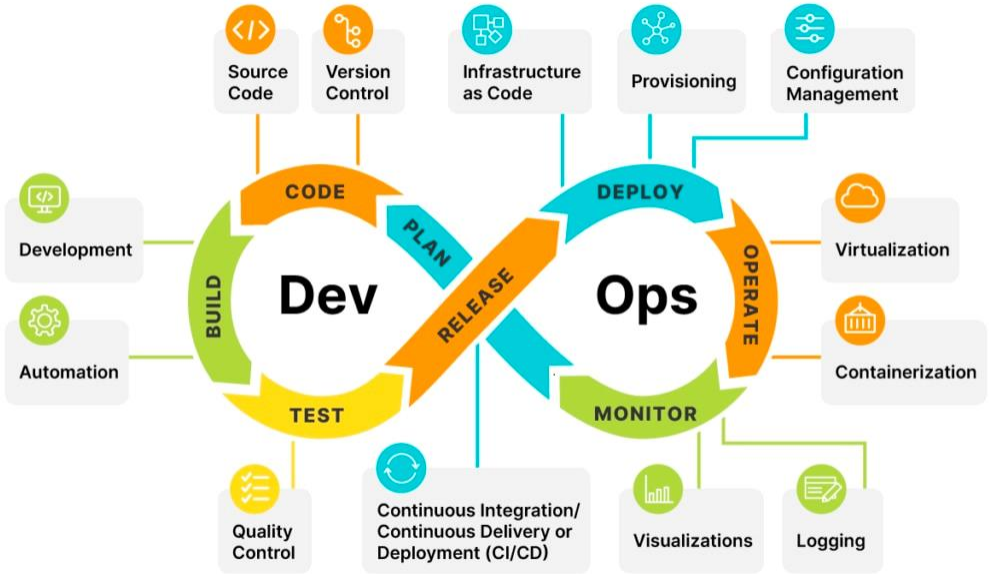
문제를 해결하기 위해서는

- 필요한 것은 바로 IDP (Internal Developer Platform)
- 오늘 발표에서는 `port` 라는 서비스의 데모를 통해 시연해볼게요

DevOps → Platform Engineering

DevOps → Platform Engineering

- 키워드: 빌드, 테스트, 배포, 모니터링, 리소스 생성



https://orangematter.solarwinds.com/2022/03/21/what-is-devops/

빌드

- Container 기반의 서비스 빌드
  - 개발자는 Dockerfile 정도는 스스로 작성하고, 로컬에서 빌드해 볼 수 있어야 함
  - 하지만 못 하는 경우, 플랫폼 개발자가 레포에 들어가 만들어주고, 개발자에게 알려줌
  - 더 나아가 공통 Dockerfile을 만들고, 중앙 관리할 수도 있음

테스트

- Compose 기반의 서비스 테스트
  - docker-compose 를 통해 db, cache 같이 띄워 테스트 실행
  - CI 템플릿을 제공하고, 테스트 결과 (stdout) 를 GitHub PR 댓글로 달아줌
  - 더 나아가 공통 docker-compose 파일 만들고, 중앙 관리할 수도 있음

배포

- Helm 기반의 Kubernetes 배포
  - 각 개발자가 중앙 관리되는 values 저장소에 자신의 서비스 값들을 생성, 수정하고 PR 올림
  - 서비스 이름, 저장소, 포트, Health 경로, CPU와 Memory 리소스, 레플리카 개수, 도메인 등
  - CI/CD 파이프라인은 자동으로 생성, 갱신 (혹은 GitOps)

Services

Search

Filter | Hide | Sort | Group by | + Service

Title	Team	Code Owners	Language	Github URL	Slack Notifications	On Call	JIRA Issues	Monitor Tooling
wish-list	TW Team Wonderwoman	Ibrahim Rotich	Node			Alexander...	4	+1
rating	TS Team Superman	Michael Molina	Python			Nathan O...	1	+1
frontend	TS Team Superman		React			Ibrahim R...	1	+1
authorization	TW Team Wonderwoman	Michael Molina	Python			Francisca ...	4	+1
authentication	TW Team Wonderwoman		Node			Michael M...	2	+1
recommenda...	TS Team Superman		Python			Ryoko De-...	4	+1
fraud-detect...	TS Team Superman		Python			Kristinn P...	1	+1
checkout	TS Team Superman		GO			Anton Sit...	8	+1
load-generat...	TS Team Spiderman		Node			Ibrahim R...	4	+1
pricing	TB Team Batman		GO			Alexander...	22	+1
ads	TB Team Batman		Python			Yuliya Attias	1	+1
payment	TS Team Spiderman		GO			Yu Panya	2	+1
shipping	TB Team Batman		Node			Alexander...	19	+1
order	TB Team Batman	+8	Node			Michael M...	5	+1
currency	TB Team Batman	Yu Panya	GO			Wolfgang ...	22	+1

Related Entities

Table | Graph

Running Service | Deployment | Upstream | Downstream | Alert | Misconfiguration

Hide Tabs

Search

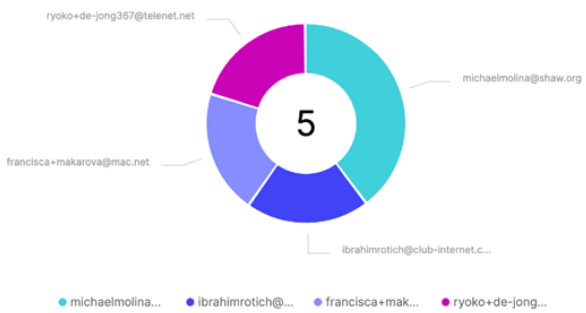
Filter | Hide | Sort | Group by

Title	Deployment Status	Last Update	User	Workflow URL	Image Tag	Commit SHA	Domain
rating-security	Success	5 hours a...	Ibrahim R...		master.dacc88eb-a24c-4f0b-996f-8a3f1b4c17e3...	c2797b5	Subscription
rating-staging	Success	5 hours a...	Michael M...		feature.9ae55a2a-8a68-4ab1-9301-1c6ee0d139e...	7c56803	Subscription
rating-sandbox	Success	5 hours a...	Francisca ...		feature.95e3decf-91b5-4a0d-8467-304562bf870...	054e141	Subscription
rating-QA	Failed	5 hours a...	Michael M...		master.4dc595a8-22ff-45ad-b773-65d807d711b...	bfa01a8	Subscription
rating-product...	In Progress	5 hours a...	Ryoko De-...		master.55f56023-c97a-4a46-8b81-ddaa1580faf5...	e6ebfec	Subscription

5 results

Last Week - Deployment by User

Deployment by User



Number of alerts (high-priorty)

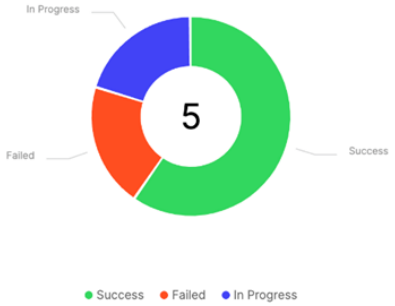
Alerts with the status "Critical" or "Error"

1

Alerts

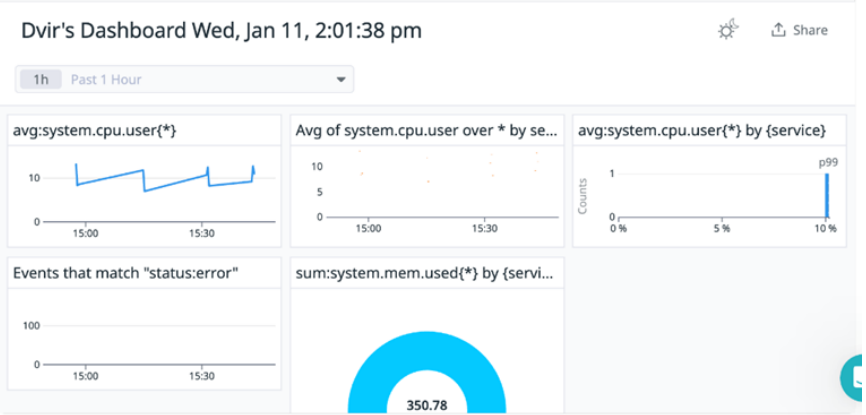
Last Week - Deployment Success Rate

Deployment Success Rate



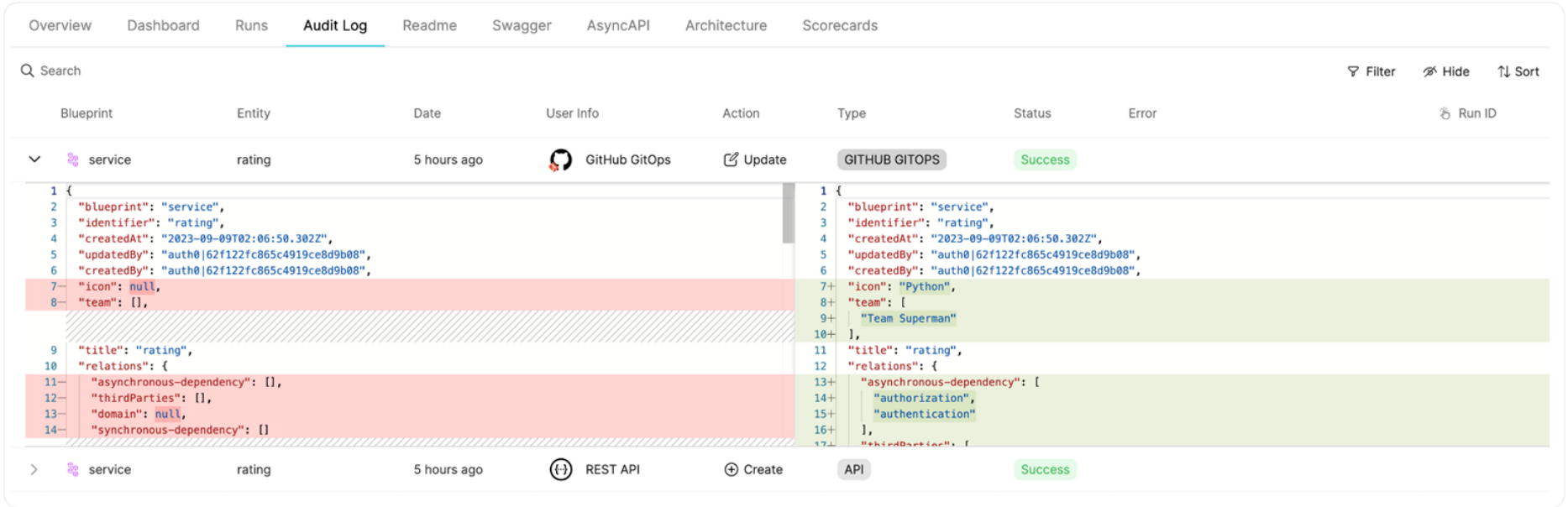
Database monitor

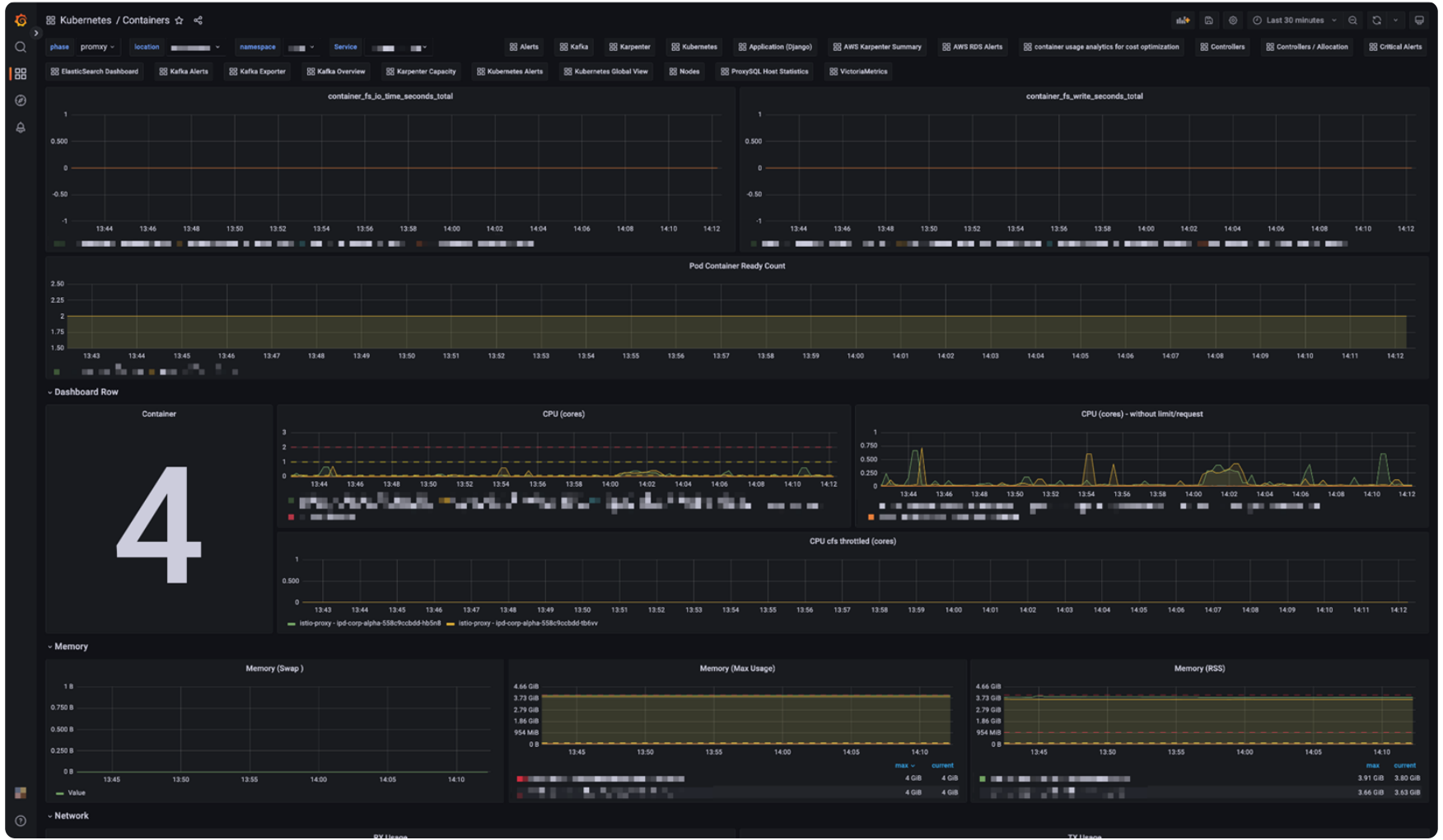
Various metrics about this service database



모니터링

- 가시성과 로그를 제공해야 함
  - 내 서비스가 언제 배포되었고, 몇 개 레플리카로 배포되었는지
  - 서비스의 Health 상태는 어떤지
  - 서비스의 요청 로그는 어떤지
  - 이런 것들을 편하게 볼 수 있는 대시보드를 구성할 수도 있음


Overview
Dashboard
Runs
Audit Log
Readme
Swagger
AsyncAPI
Architecture
Scorecards
Search
Filter
Hide
Sort
Blueprint
Entity
Date
User Info
Action
Type
Status
Error
Run ID
service
rating
5 hours ago
GitHub GitOps
Update
GITHUB GITOPS
Success
{
"blueprint": "service",
"identifier": "rating",
"createdAt": "2023-09-09T02:06:50.302Z",
"updatedBy": "auth0|62f122fc865c4919ce8d9b08",
"createdBy": "auth0|62f122fc865c4919ce8d9b08",
"icon": null,
"team": [],
"title": "rating",
"relations": {
"asynchronous-dependency": [],
"thirdParties": [],
"domain": null,
"synchronous-dependency": []
{
"blueprint": "service",
"identifier": "rating",
"createdAt": "2023-09-09T02:06:50.302Z",
"updatedBy": "auth0|62f122fc865c4919ce8d9b08",
"createdBy": "auth0|62f122fc865c4919ce8d9b08",
"icon": "Python",
"team": [
"Team Superman"
],
"title": "rating",
"relations": {
"asynchronous-dependency": [
"authorization",
"authentication"
],
service
rating
5 hours ago
REST API
Create
API
Success

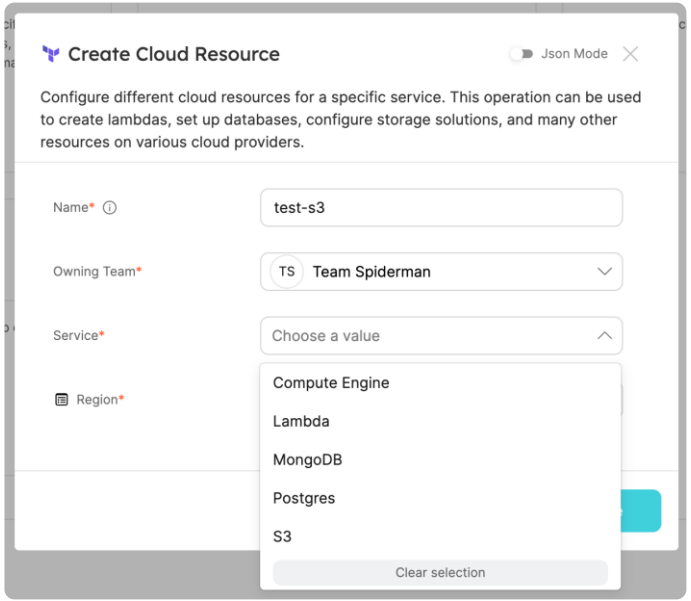

Kubernetes / Containers
Last 30 minutes
phase
promxy
location
namespace
Service
Alerts
Kafka
Karpenter
Kubernetes
Application (Django)
AWS Karpenter Summary
AWS RDS Alerts
container usage analytics for cost optimization
Controllers
Controllers / Allocation
Critical Alerts
ElasticSearch Dashboard
Kafka Alerts
Kafka Exporter
Kafka Overview
Karpenter Capacity
Kubernetes Alerts
Kubernetes Global View
Nodes
ProxySQL Host Statistics
VictoriaMetrics
container_fs_io_time_seconds_total
container_fs_write_seconds_total
Pod Container Ready Count
Dashboard Row
Container
4
CPU (cores)
CPU (cores) - without limit/request
CPU cfs throttled (cores)
istio-proxy - ipd-corp-alpha-558c9ccbdd-hb5n8
istio-proxy - ipd-corp-alpha-558c9ccbdd-tb6vv
Memory
Memory (Swap)
Value
Memory (Max Usage)
Memory (RSS)
max
current
Network

리소스 생성

- DB, Redis, CDN 설정, Kafka 토픽 생성
  - 대시보드에서 확인하고, 필요한 경우 생성 요청을 할 수 있어야 함
  - 일관되고 안전한 구성으로 생성됨 (보안 구성, 스펙)

리소스 생성

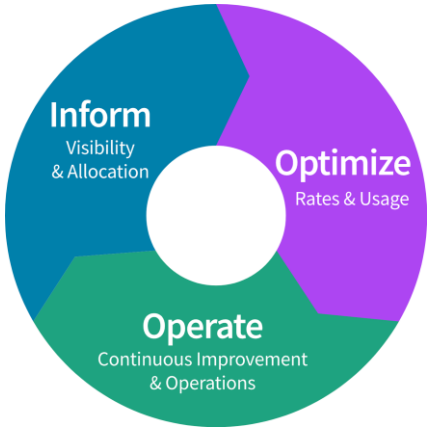
- DB, Redis
  - CPU, Memory, Disk 사이즈
- Kafka 토픽
  - 토픽 이름 컨벤션
  - 파티션 수
  - 복제 수



FinOps → Platform Engineering

FinOps → Platform Engineering

- 키워드: 비용 시각화, 비용 최적화



https://www.finops.org/framework/phases/

비용 시각화

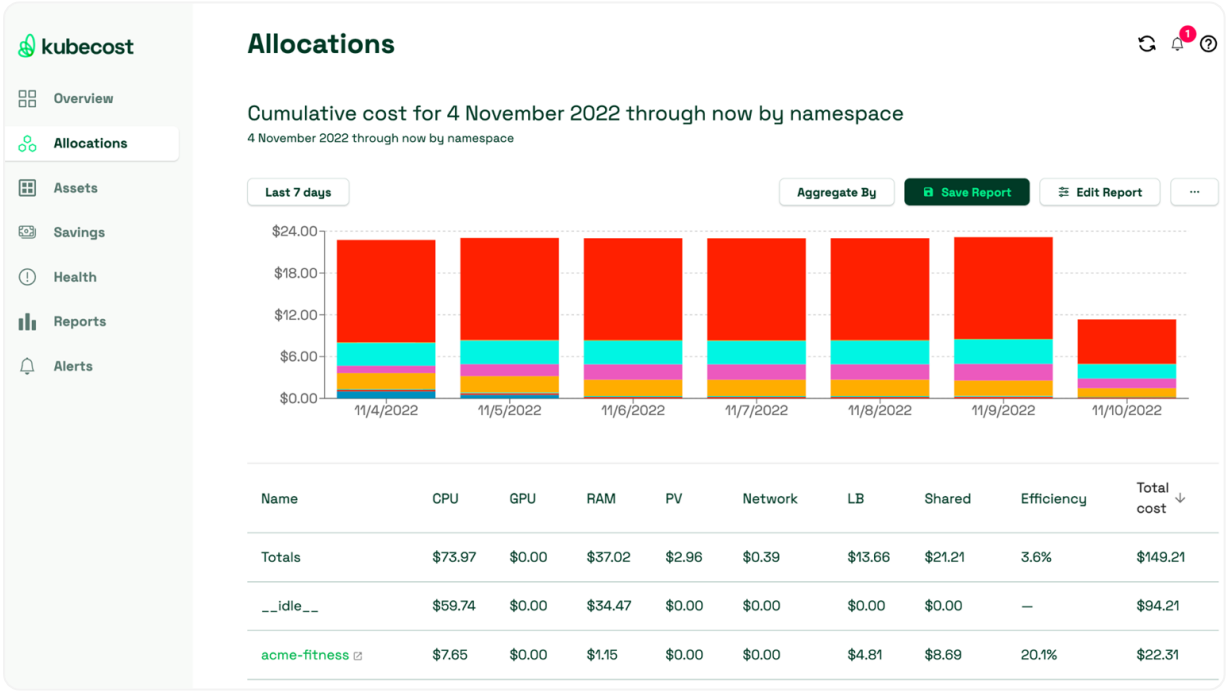
- 개발자들은 내 서비스가 쓰는 돈만 궁금함
  - 전체 EC2 비용, RDS 비용에는 관심 없음
- AWS Cost Explorer는 한계가 있다
  - 모든 리소스에 태그를 달기 어렵고,
  - 여러 마이크로 서비스가 하나의 리소스를 공유해 사용하는 경우도 있다

비용 시각화

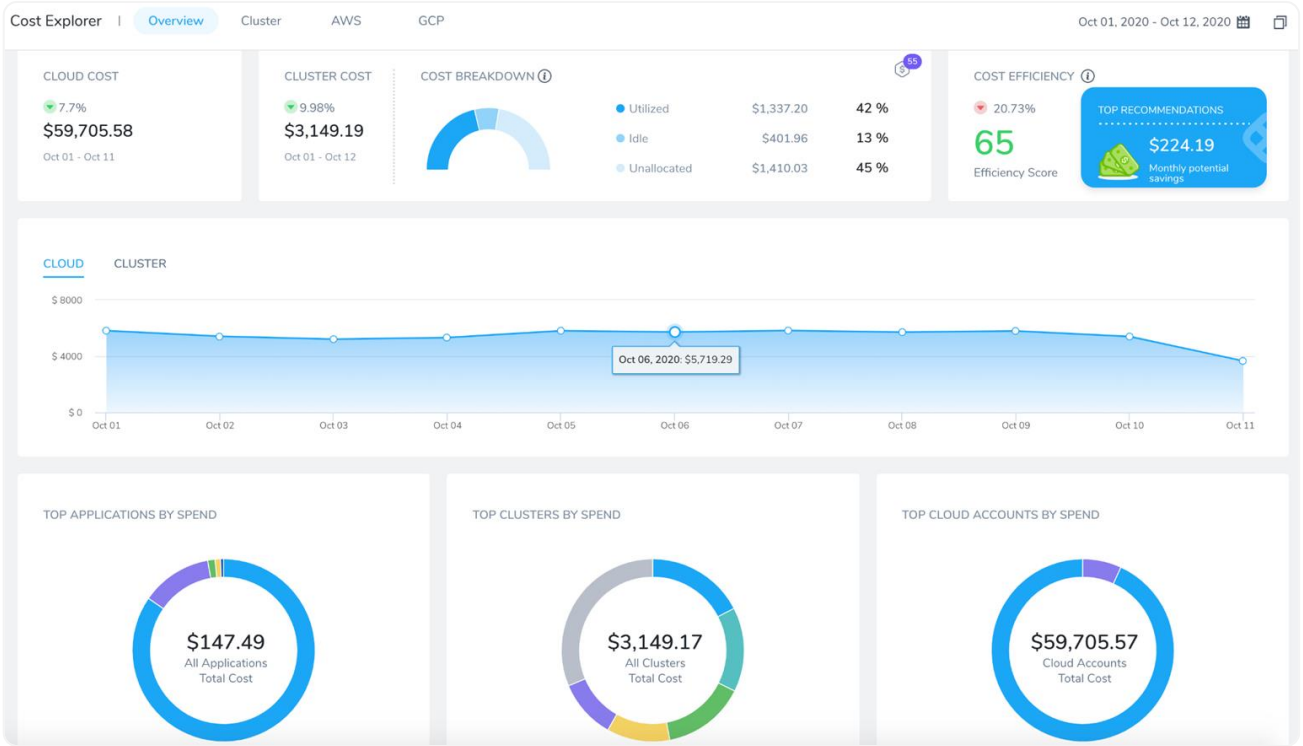
- AWS 리소스에 최대한 서비스 별 태그를 달아두고,
- RDS, ElastiCache, MSK(Kafka) 등 공용 리소스들은
  1. 사용하는 서비스들이 n빵 하거나
  2. 테이블 용량, Key 개수, 파티션 용량 등으로 분할 계산

비용 시각화

- 비용 알림
  - 비용이 스파이크를 치는 경우
  - 특정 월에 크게 늘거나 줄어든 경우



https://blog.kubecost.com/blog/kubecost-version-1-98-0-new-feature-highlights/



https://developer.harness.io/docs/first-gen/cloud-cost-management/concepts-ccm/a-cost-explorer-walkthrough/

비용 최적화

- 비용 알림
  - 비용이 급격히 늘어나는 경우 알림을 받고, 바로 대응할 수 있음
- 비용 비교
  - 내 서비스가 다른 서비스보다 더 많이 돈을 쓴다면,
    알아서 줄이게 됨

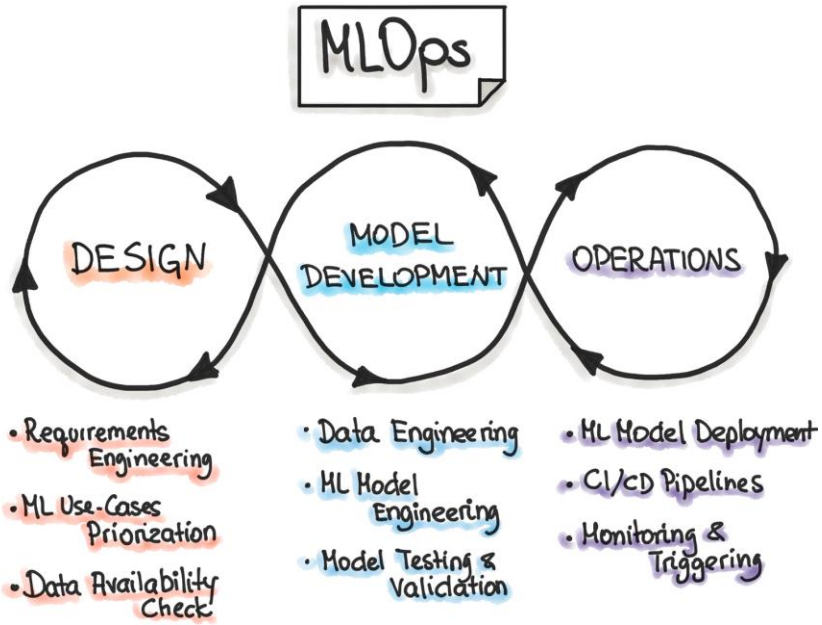
비용 최적화

- 그 외에는 성능 튜닝이 필요함
  - CPU, Memory를 적게 사용하고 API 응답 속도를 높여서 Replica를 줄이는 방법
  - 쿼리 튜닝을 통해 DB 부하를 줄이는 방법
  - 불필요한 로직을 개선하고 리팩토링 하는 방법
- 플랫폼 팀, SRE 팀이 도와줄 수도 있고, 따로 시간을 할당해 개선할 수도 있다.

MLOps → Platform Engineering

MLOps → Platform Engineering

- 키워드: 모델 제공 및 운영



https://ml-ops.org/content/mlops-principles

Feature Store

- Feature Store
  - ○ 모델 개발과 학습, 운영에 필요한 데이터 (=Feature)
  - ○ 이러한 데이터를 저장하고, 관리하고, 필요한 모델에 주입해 학습할 수 있는 플랫폼
- 이걸 잘 운영하고 개선하기만 해도 성공적

모델 제공 및 운영

- API를 통해 실시간으로 데이터를 Feature Store에 적재하고,
- 각 서비스는 Python 코드를 통해 실시간으로 / 배치로 Feature Store의 데이터를 사용해 학습하거나 예측함
- ML Pipeline에서 Feature Store를 접근해 Pre-Trained 된 모델을 배포하기도 함

결론

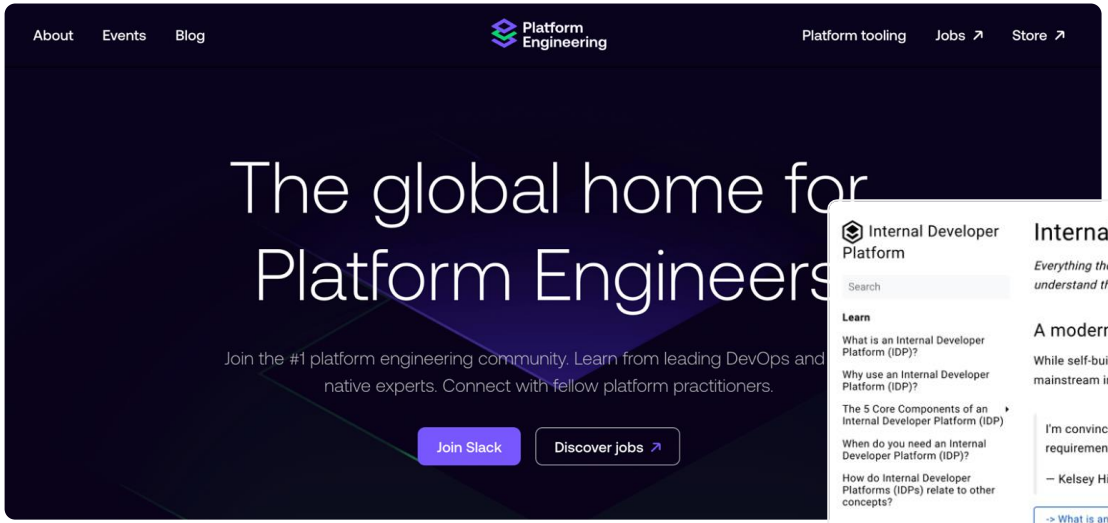
Platform Engineering을 통해

- 무엇이든 운영이 필요한 것들은 Self Service 화
- 개발자는 스스로 서비스를 개발하고, 배포하고, 운영할 수 있게 되었다
  - 비용도 측정하고, DB, Cache도 생성
- 플랫폼 팀은 행복해졌다 

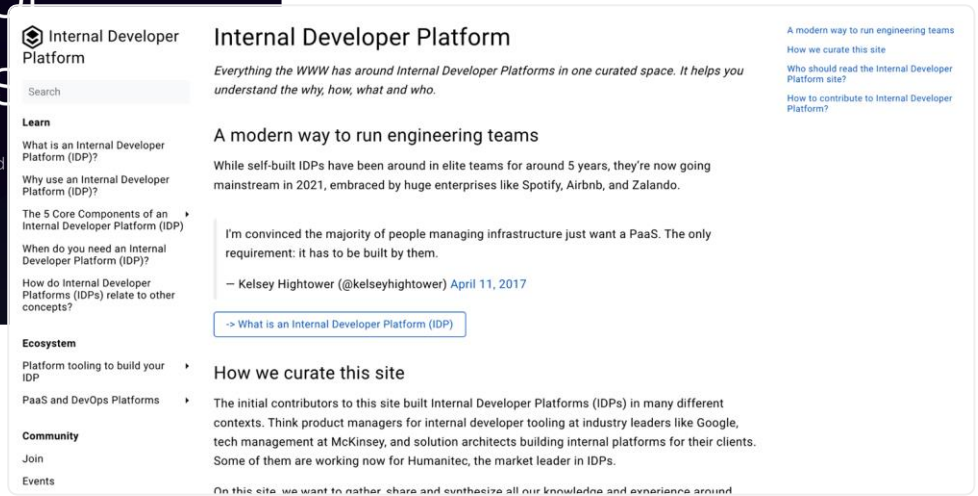
중요한 것은

- DX, DevEX (Developer Experience)
  - 개발자가 쉽고 편하게 사용할 수 있는 IDP를 개발해야 한다.
- IDP도 제품
  - 일반적인 제품을 다루듯 피드백을 거쳐가며 개선해야 한다.

Platform Engineering 커뮤니티



https://platformengineering.org/



https://internaldeveloperplatform.org/