MSA School

전자정부 표준 프레임워크 기반의

클라우드 네이티브 애플리케이션 배포

Part3. 마이크로서비스 배포/운영

UEngine

Tutors

장진영 유엔진솔루션즈 대표이사

MSA School 대표 강사
SAFe 공인 컨설턴트
(전) OMG / SEMAT 국제 표준 도구 분과 위원
現 디지털플랫폼정부 기술자문위원
유엔진BPMS / 오픈 클라우드 엔진 파운더
클라우드 네이티브 (MSA, DDD) 강의
MSA와 SW모델링 Facebook 그룹 운영
(www.facebook.com/groups/cloudswmoding)

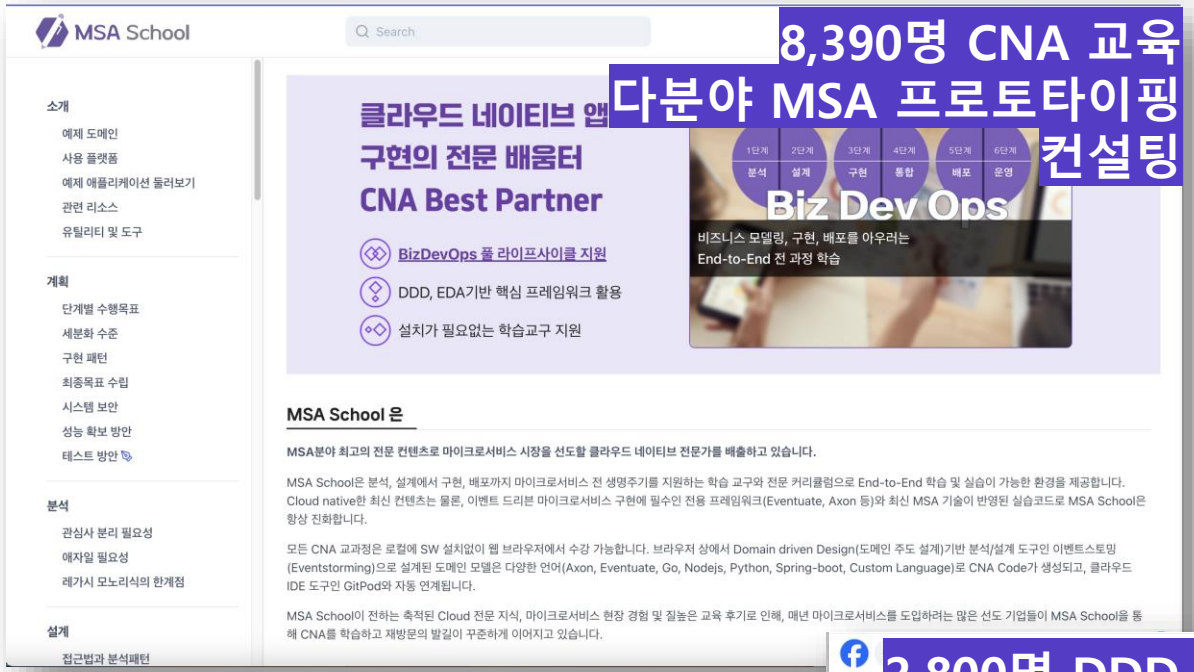
박용주 유엔진솔루션즈 이사

MSA School 대표 강사
• 現 MSA App. Engineering 기업과정 대표강사
• 現 세종사이버대학교 컴퓨터/AI 공학과 겸임교수
• 한국소프트웨어기술진흥협회 전문강사
• '21 : SK MSA App. Engineering 과정
• '21. 06 : KT Microservice 직무전환과정
• '20. 09 : Doosan Microservices 교육
• '19. 09 : KOSTA Microservices 교육
• '19. 02 : LG CNS 이벤트스토밍 교육

윤성열 드림플로우 대표이사

MSA School 대표 강사
• 現 드림플로우 대표이사
• 現 한국소프트웨어기술진흥협회 BAPF 포럼, 교육과정위원회 및 전문강사
• '18. 10 : 원오원글로벌 디지털팀 팀장
• '18. 04 : TTA 사물인터넷 특별기술위원회 사물인터넷 융합서비스 프로젝트그룹 (SPG11) 부의장
• '17. 03 : 가천대학교 겸임교수

About Us

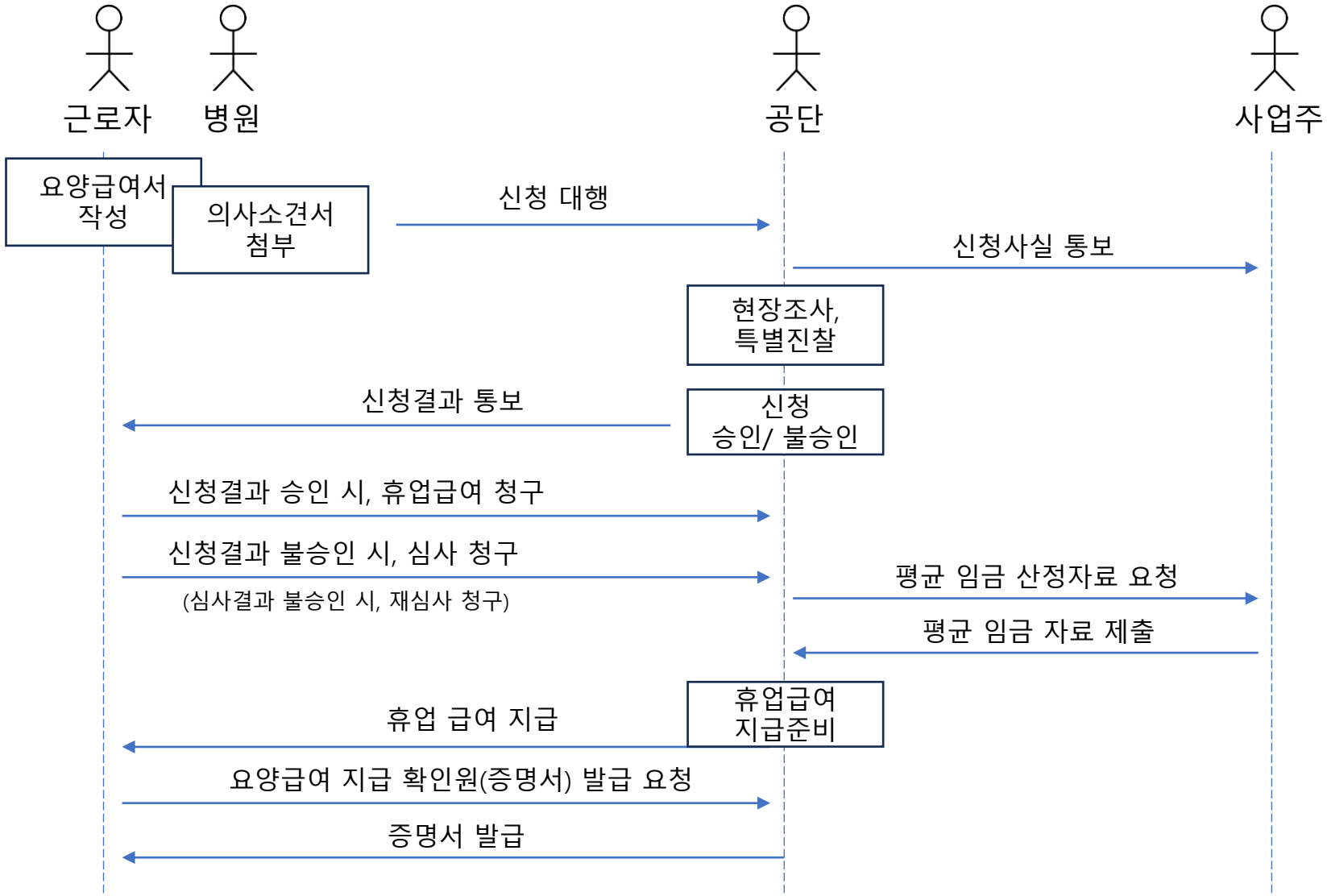


8,390명 CNA 교육
다분야 MSA 프로토타이핑
컨설팅

MSA 분석/구현 전문 도구

2,800명 DDD 페북그룹 운영

웨비나 토픽 : 근로복지공단 '산재신청'



‘산재신청’ 요구사항 도출

기능적 요구사항

- 근로자가 요양급여 신청서를 작성하여 제출한다.
- 병원은 의사의 소견서와 요양 병원 직인을 신청서에 첨부하고, 공단에 대행 신청한다.
- 사업주는 근로복지공단을 통해 산재 신청 사실을 통보 받는다.
- 근로복지공단은 질병의 인과관계 확인을 위해 현장조사와 특별 진찰을 수행한다.
- 인과관계가 확인되면, 근로복지공단은 접수된 신청을 승인한다.
- 승인을 통보 받은 근로자는 휴업 급여를 청구한다.
- 불승인된 경우, 근로자는 심사청구를 요청할 수 있다.
- 근로복지공단은 해당 사업장에 근로자의 평균 임금을 요청한다.
- 근로복지공단은 심사결과와 보상 금액을 근로자에게 통보한다.
- 심사결과 금액에 불복하는 근로자는 재심사를 청구할 수 있다.
- 근로자가 요양급여 지급 확인원(증명서) 발급을 요청한다.

비기능적 요구사항

장애격리

- 요양급여 처리 서비스가 장애 시 라도 요양급여 신청 접수는 문제가 없어야 한다 → Event-driven, Eventual Consistency
- 일부 시스템이 과중 되면 적절히 확장이 이루어져야 한다 → Horizontal Auto Scale

프로세스 준수

- 각 마이크로 서비스를 넘나드는 프로세스는 일관성이 있게 준수되어야 한다. (예: 업무상 재해가 명확한 경우 7일 이내 요양 승인 여부가 결정된다.)

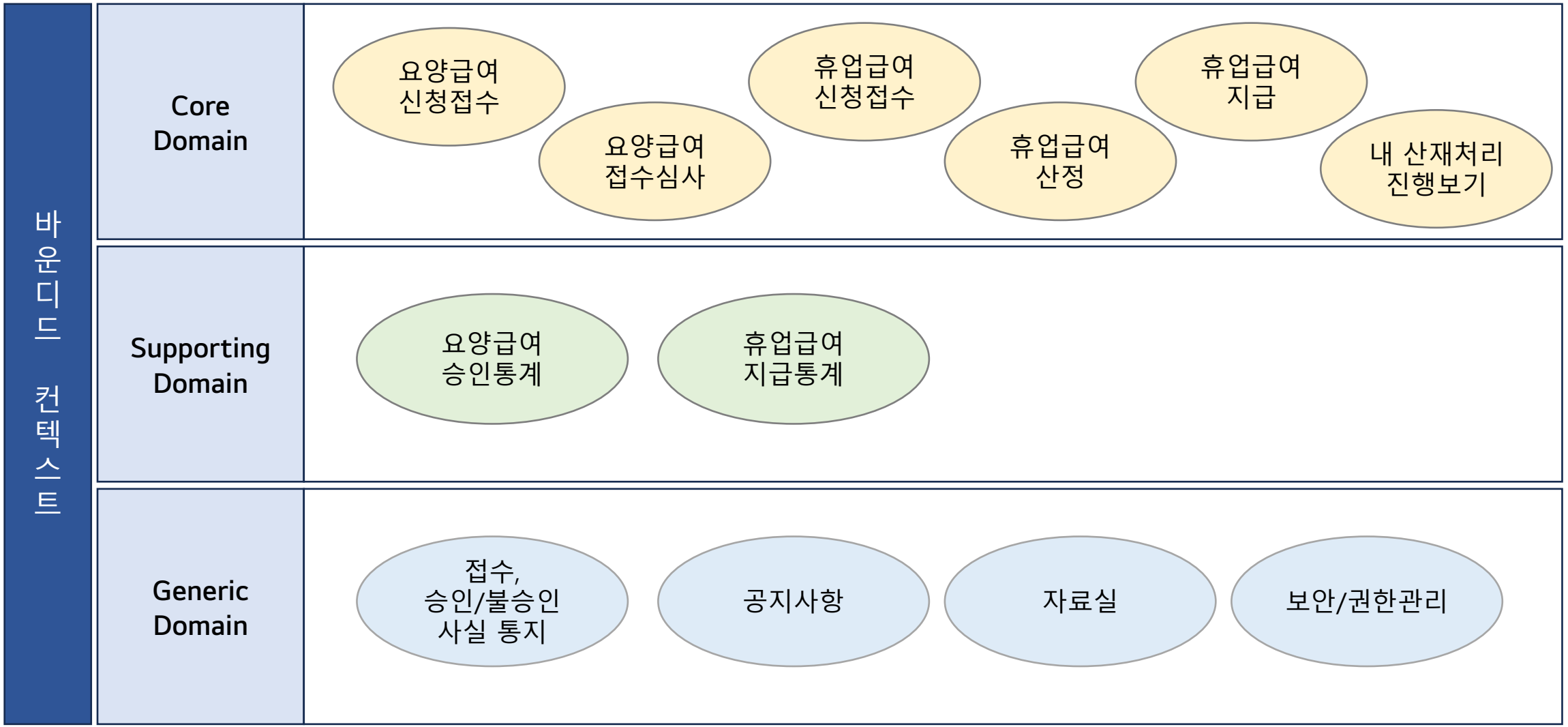
성능

- 근로자 보수총액 신고 기간(매년 3월 말까지) 및 근로자가 자주 민원 진행 상태를 조회하더라도 성능저하가 없어야 한다. → Auto Scale Out, CQRS

도메인 제약사항

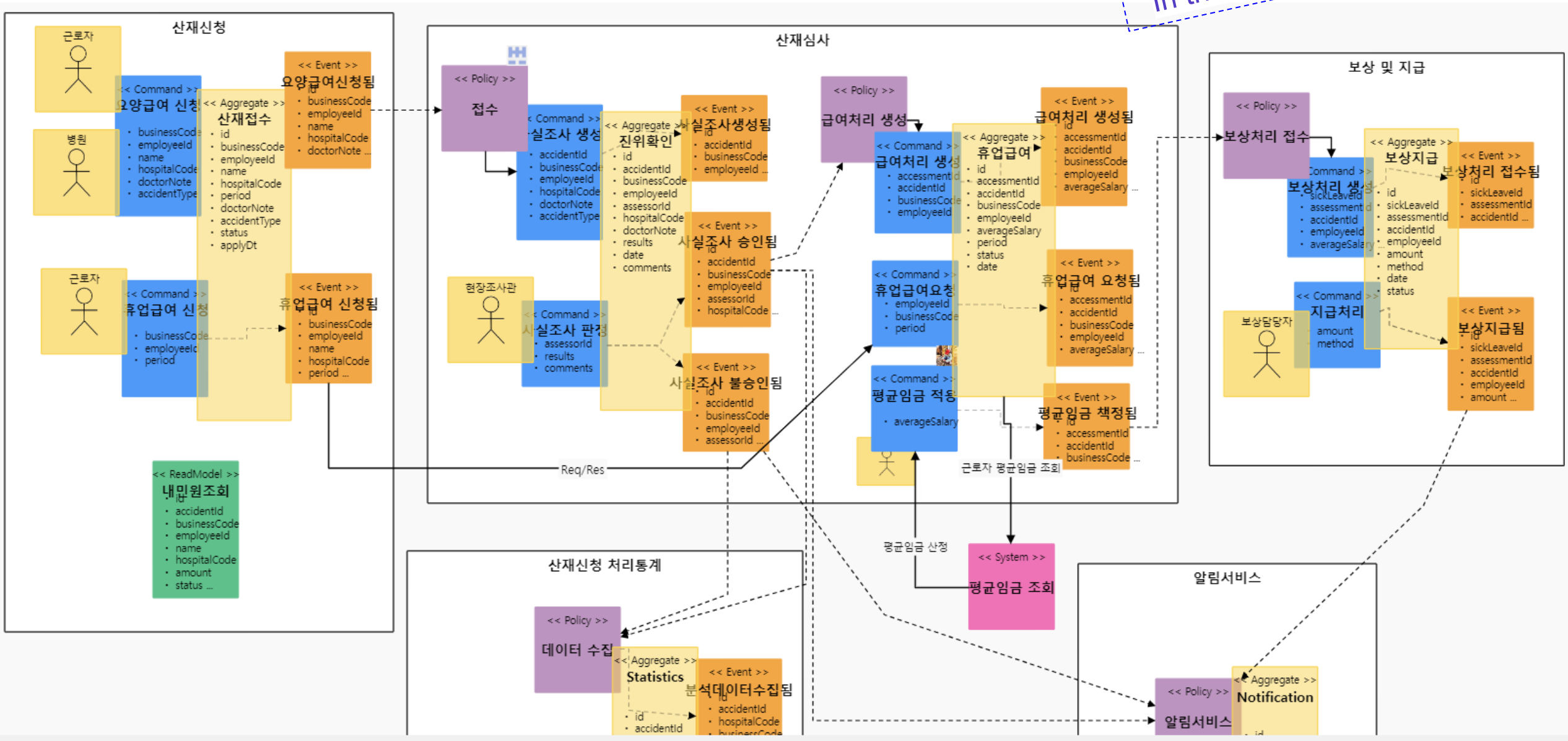
- 산재 신청은 4일 이상 요양이 필요한 질병에 대해 산재신청이 가능하다.
- 휴업 급여는 평균 임금의 70%로 산정한다.

‘산재신청’ 앱 설계 – 멀티도메인 경계 식별



‘산재신청’ 이벤트스토밍 모델

In the past Webinar Part 1.



마이크로서비스 구현 패턴

In the past Webinar Part 2.

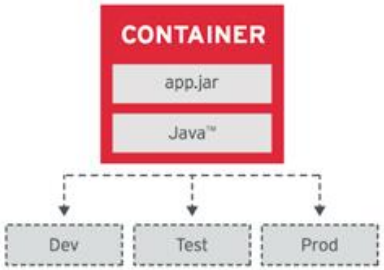
구현 패턴	특 징	비고
서비스 단일 진입을 위한 API 게이트웨이 패턴	• 단일 진입점을 만들어 다른 유형의 클라이언트에게 서로 다른 API조합을 제공할 수도 있고 각 서비스 접근 시 필요한 인증/인가 기능을 한 번에 처리 가능	• Spring Cloud GW • Kubernetes Ingress
장애전파 차단을 위한 서킷 브레이커 패턴	• 하나의 서비스 장애로 다른 서비스가 영향을 받을 수 있는데, 장애가 발생된 서비스를 격리하여 유연하게 처리할 수 있는 방법으로 다양한 구현 단계에서 적용 가능	• Netflix OSS • Service Mesh
SAGA 패턴 (Eventual consistency, Compensation Trx)	• 각 서비스의 로컬 트랜잭션을 순차적으로 처리하는 패턴. 여러 개의 분산된 서비스들을 하나의 트랜잭션을 묶지 않고, 각 로컬 트랜잭션과 보상 트랜잭션을 이용해 비지니스 데이터 정합성 일치	• 오케스트레이션 Saga • 코레오그래피 Saga
CQRS 패턴 (명령 조회 책임 분리, 읽기모델과 쓰기모델 분리)	• 하나의 저장소에 쓰기 모델과 읽기 모델을 분리하는 방식으로 변화시켜 쓰기 서비스와 조회서비스를 분리해 저장소 처리 성능을 향상	• Materialized View
쓰기 최적화 이벤트 소싱(Event Sourcing) 패턴	• 메시지 브로커와 데이터 저장소를 분리하지 않고 하나로 사용해 상태가 아닌 트랜잭션 차체를 저장 하는 전략으로 상태 변경 이벤트를 바로 이벤트 저장소에 저장	• Replay, Snapshot
이벤트 드리븐 마이크로서비스(Event Driven Microservice)	• 발신자가 이벤트를 생성, 발행 (publish)하고 필요로 하는 수신자가 이를 구독((Subscribe)하고, 이벤트를 처리하는 형태의 커뮤니케이션 아키텍처	• EDA, Pub/Sub
서비스 매시(Service Mesh) 패턴	• 통신 네트워크 기능을 비즈니스 로직 기능과 분리하여 네트워크 인프라 계층에서 수행하게 해, 마이크로서비스 개발자가 비즈니스 로직 구현에 포커싱 가능	• Istio, Consul, Linkerd
마이크로 프론트엔드 UI 패턴	• 프론트 엔드가 여러 개의 조각들로 구성되고 각 조각은 여러 개의 마이크로 프론트 엔드의 조합으로 서비스를 제공	• Monolith Frontend • Micro Fronted
마이크로서비스 테스트 패턴 (Unit Test, API Mocking, CDC Test)	• Given-When-Then 테스트 패턴을 활용한 단위 마이크로서비스 테스트, 가상의 실체화된 서비스를 활용한 병렬 개발, 고객 주도 컨트랙 기반 API 정합성 테스트 전략	• Microcks (CNCF) • Spring Cloud Contract
마이크로서비스 인증/인가 패턴	• 인증/인가 처리를 위한 별도의 전담 서비스(또는 SSO 서버)를 두고, API 게이트웨이와 연동해 인증 인가를 처리하고 발급된 토큰을 다운 스트림으로 포워딩	• JWT Token, • Keycloak (CNCF)
중앙화된 로그 집계, 모니터링 및 추적 패턴	• 마이크로서비스 장애 실시간 감지와 서비스 간의 호출 인지 및 실시간 로그 집계를 위한 스택과 각가 서비스 트랜잭션의 추적을 통한 분산 트레이싱 구현	• 모니터링, 로그리게이션 • Zipkin 기반 트레이싱

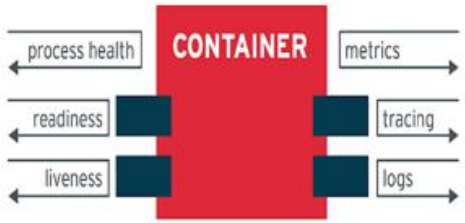
오늘의 주제 – MSA 의 배포와 운영

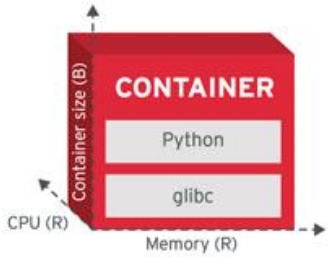
CNA 배포 목표 시나리오

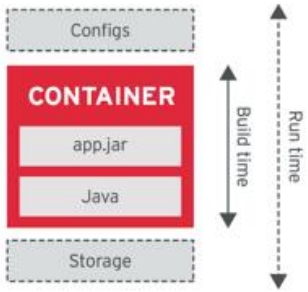
- 서비스 도커라이징(패키징)
- 배포 모델 설계
- 마이크로서비스 배포 자동화
- GitOps 기반 배포 모니터링
- 컨테이너 오케스트레이션
- 서비스 메시의 활용
- 옵저버빌리티와 로그리게이션
- Cloud native SRE 소개

등을 MSA Easy 를 통하여 세부적으로 배포 모델
배포 후 오케스트레이션 하는 과정을 살펴봅니다

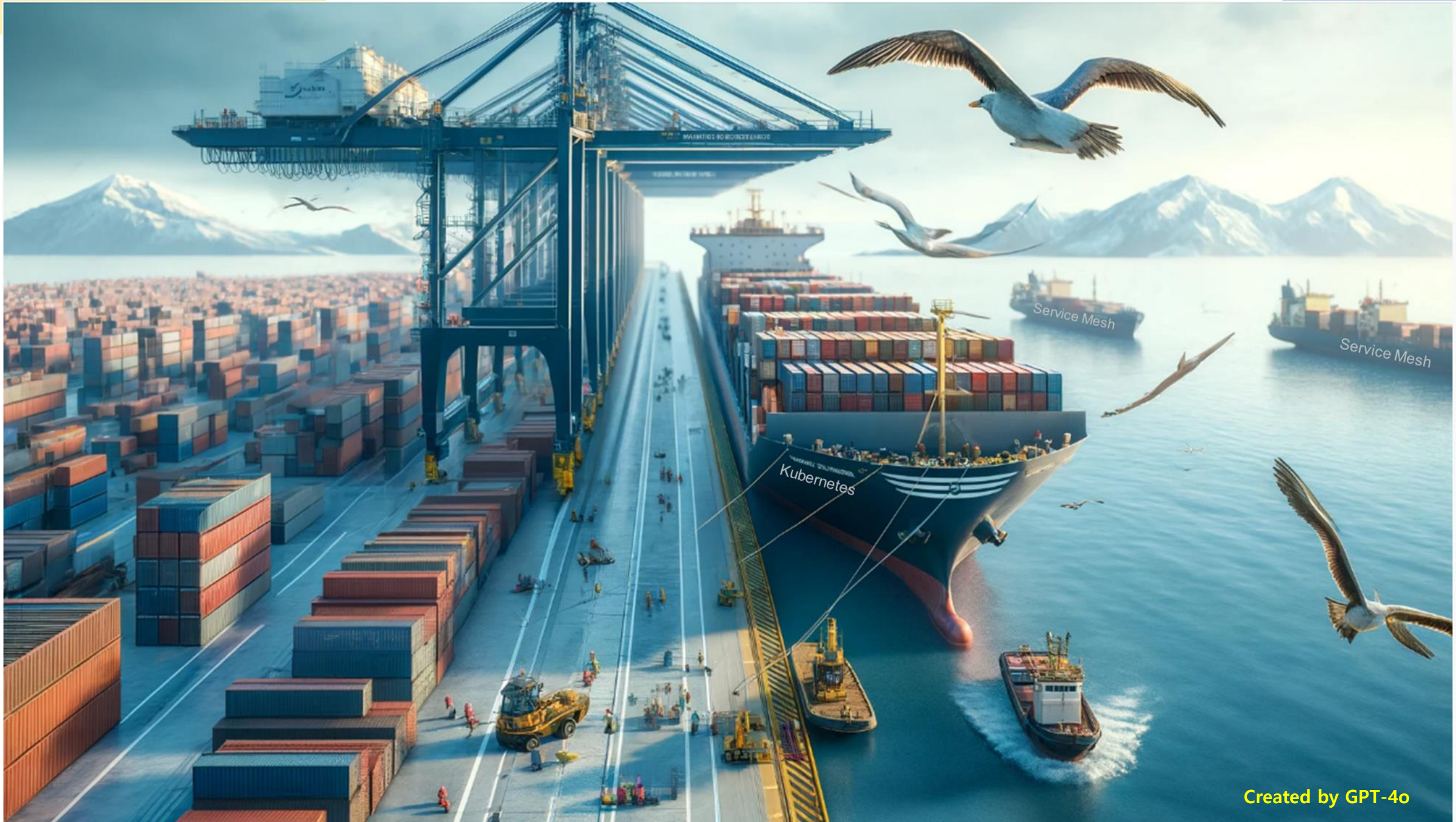






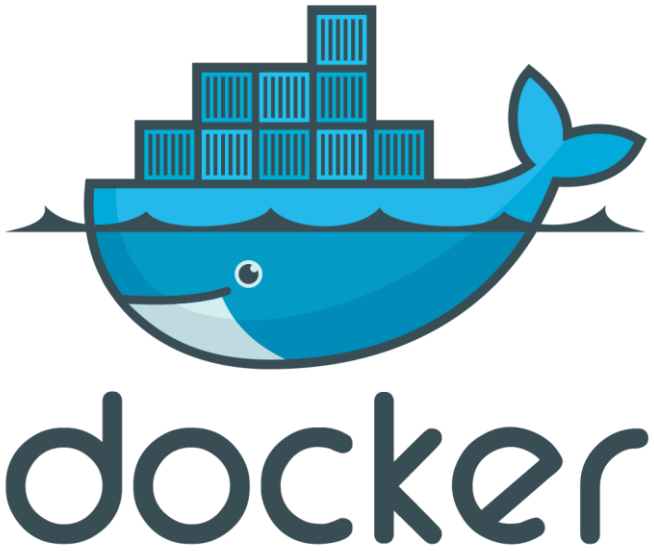


The Principles of Container-based Application Design


Service Mesh
Service Mesh
Kubernetes
Created by GPT-4o

서비스 도커라이징(패키징)

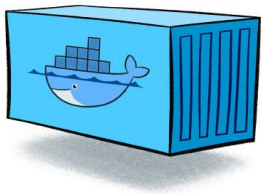
- 구현된 각 마이크로서비스는 각자의 에코시스템을 만들어 독립적으로 운영
- 이를 위한 클라우드 기술로 도커(Docker)와 컨테이너(Container)를 활용해 격리된 환경으로 패키징



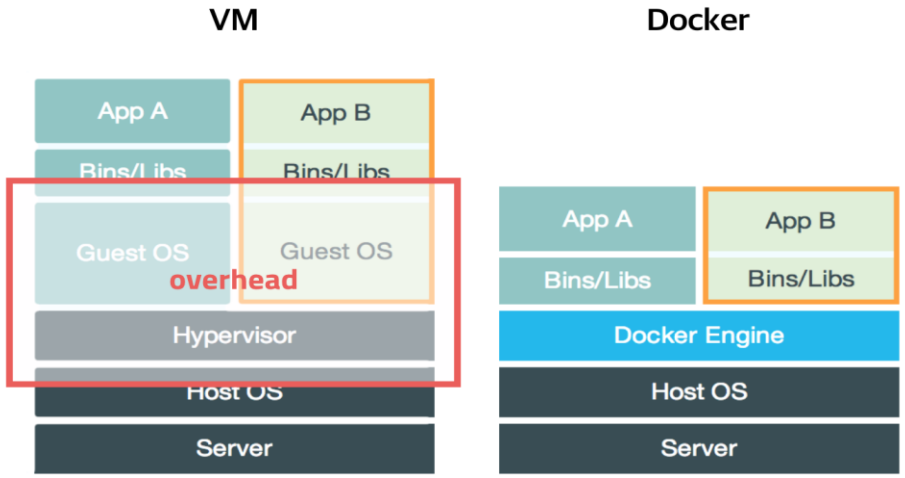
✓ 2013년 3월,
dotCloud 창업자 Solomon Hykes가 Pycon Conference 에서 발표

✓ Go 언어로 작성된 “The future of linux Containers”

✓Container based
- ✓ 프로세스 Isolation(격리) 기술
- ✓ 오픈소스 가상화 플랫폼



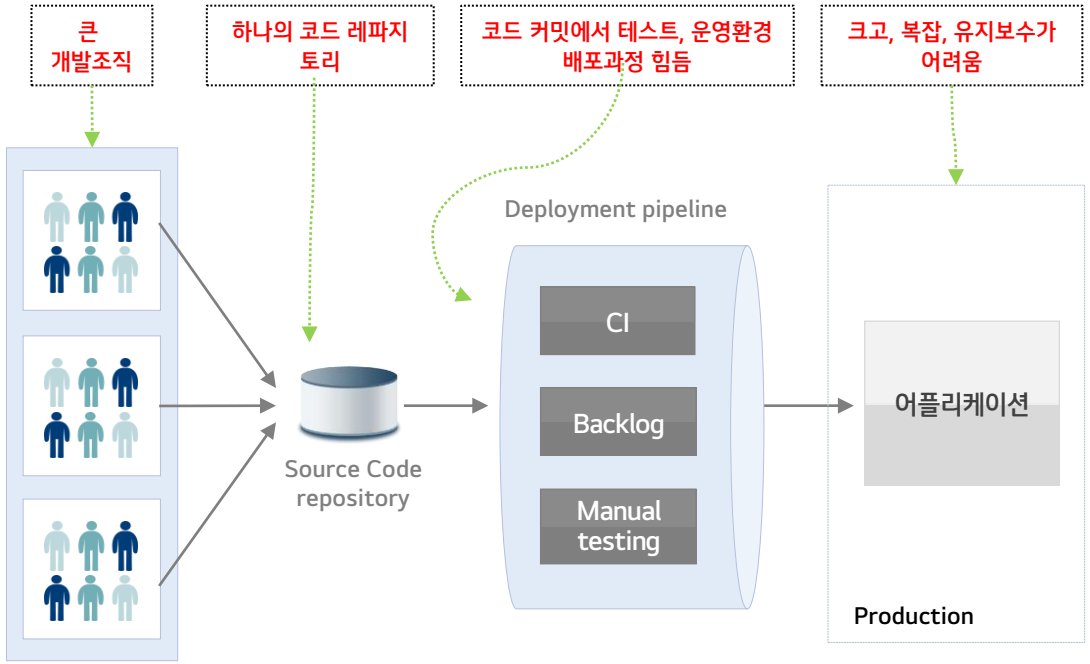
✓가상머신 .vs. Docker



“열차 말고 택시를 타라”

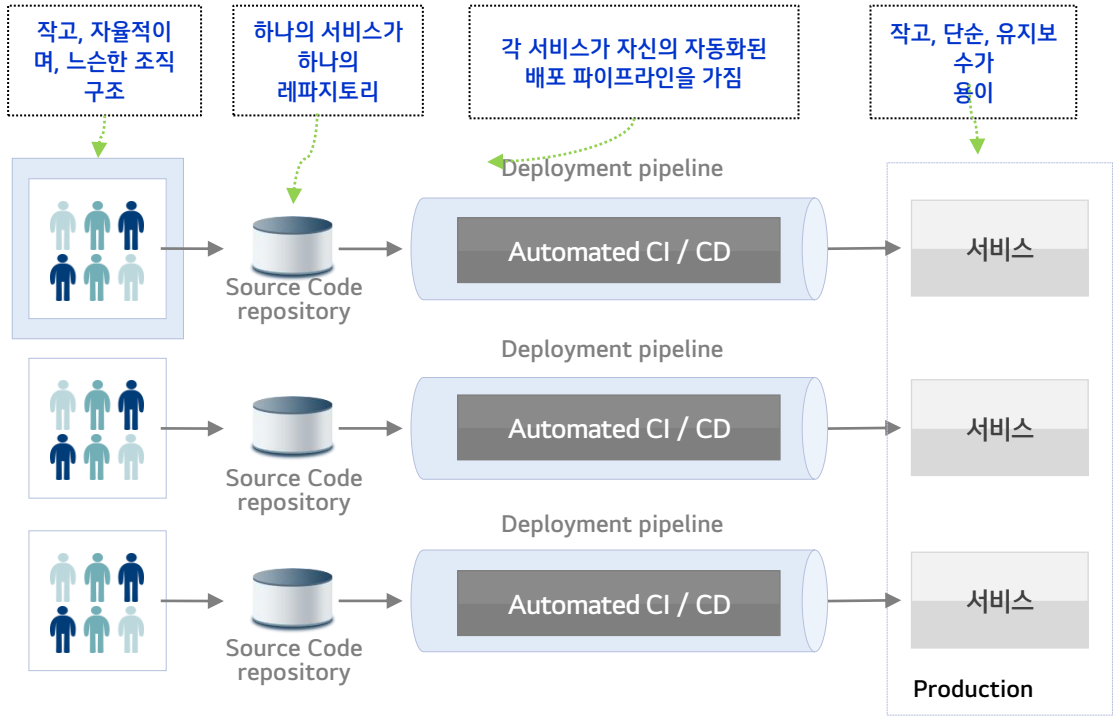
모노리식 개발 및 배포 환경

- 모노리식 환경하에서는 큰 개발팀이 하나의 소스코드 레파지토리에 변경 사항을 커밋하므로 코드간 상호 의존도가 높아 신규 개발 및 변경이 어려움
- 작은 변경에도 전제를 다시 테스트/ 배포하는 구조이므로 통합 스케줄에 맞춘 파이프 라인을 적용하기가 어렵고 Delivery 시간이 과다 소요



마이크로서비스 개발 및 배포 환경

- 작고 분화된 조직에서 서비스를 작은 크기로 나누어 개발하므로 해당 비즈니스 로직에만 집중하게 되어 개발 생산성 향상
- 각 팀만 다른 팀 간섭없이 언제든 새로운 기능의 테스트와 롤 아웃이 가능하고 연관된 마이크로서비스만 테스트를 수행하므로 개발/테스트/배포 일정이 대폭 축소



배포 모델 설계

배포 모델 설계란 배포하기 전, 마이크로서비스 아키텍처를 모델링 하는 것

- 각 서비스의 요구 사항과 시스템 환경에 맞추어 전체 마이크로서비스 운영 토폴로지 구성
- 배포 모델 설계에는 다음과 같은 구성 요소를 포함

모니터링 및 로깅

리소스 관리

CI/CD 통합 및 지속적 배포

보안 및 접근제어

고가용성 및 장애 복구

- 다중 ReplicaSet 및 Deployment를 통해 애플리케이션의 고가용성 보장
- 이상 감지를 통한 셀프 힐링과 무정지 배포의 실현

서비스 리자일런스 적용

- Service Mesh를 사용하여 서비스 간 통신 및 트래픽 관리
- 트래픽 제어, 인증 및 인가, 로깅 및 모니터링 등을 향상

자동 서비스 확장

- HPA를 사용하여 트래픽 부하에 따라 자동으로 Pod 수를 늘리거나 줄이는 설정
- 클러스터 오토 스케일링을 통해 노드의 수를 자동으로 조정

네트워크 관리

- 외부 트래픽을 내부 서비스로 라우팅
- Service 객체를 사용하여 Pod 간의 안정적인 통신 제공

스토리지 클레임 패턴

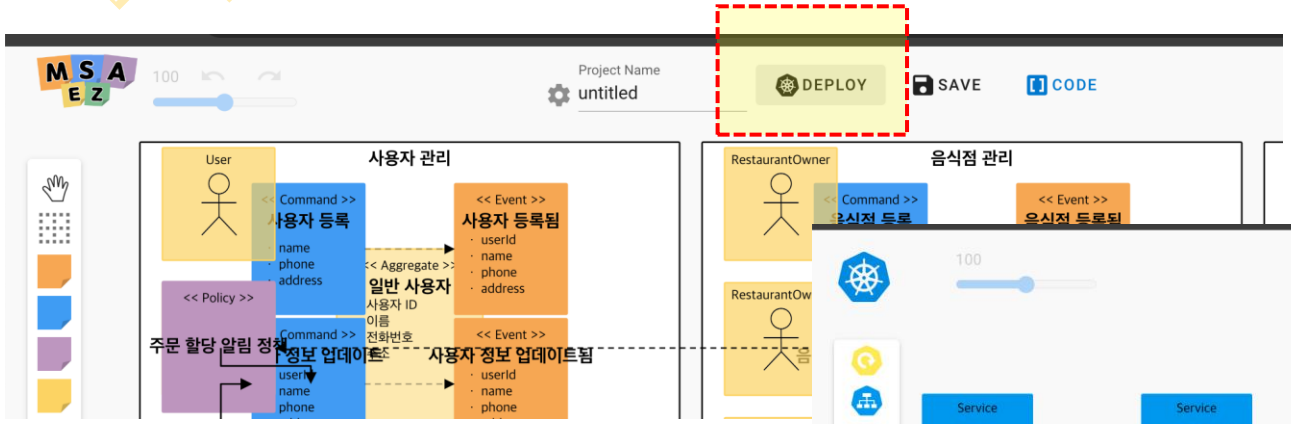
- 클라우드 스토리지 요청 및 바인딩
- 필요한 경우, 동적 프로비저닝을 사용하여 자동 생성 관리

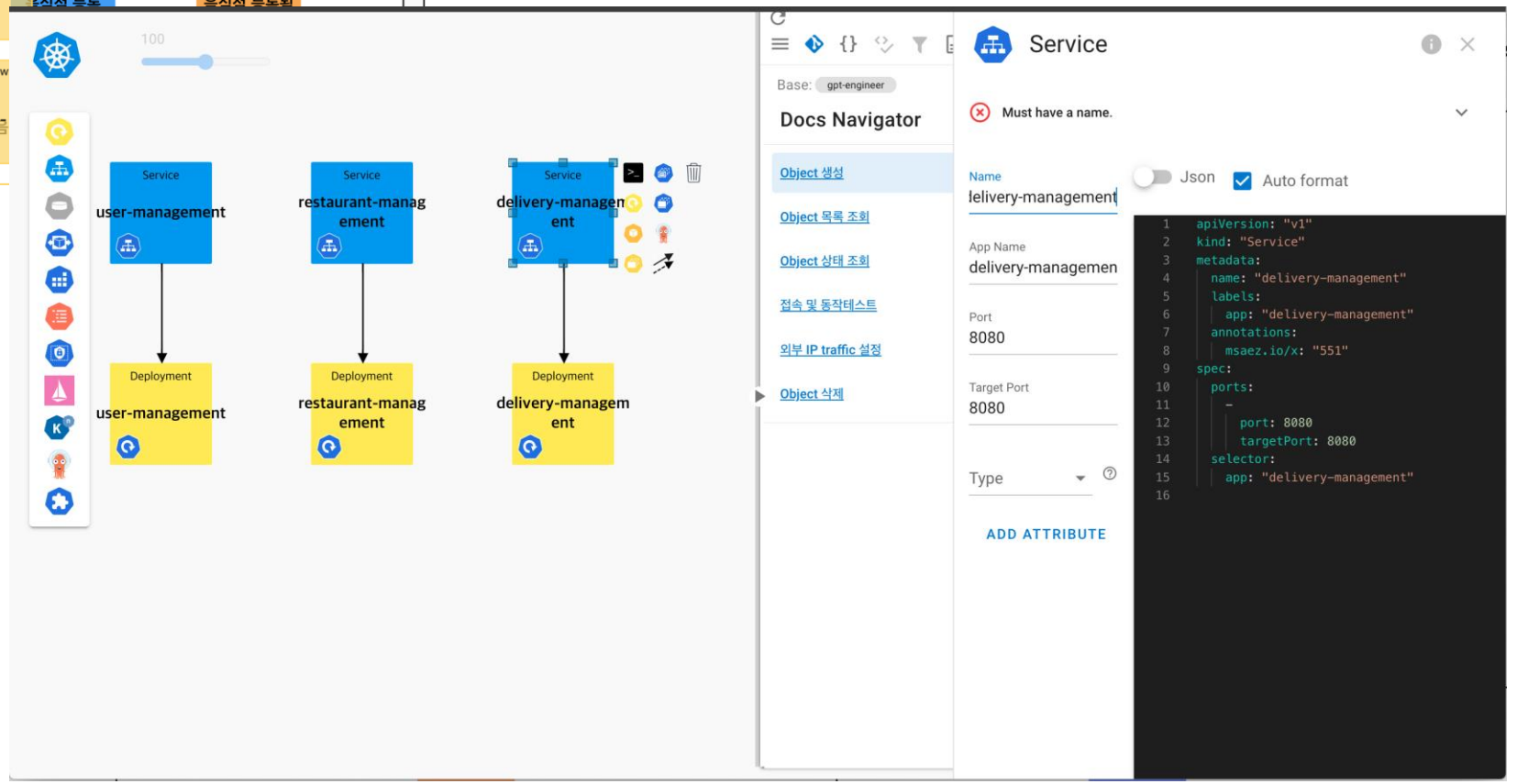
애플리케이션 구성 관리

- ConfigMap 및 Secrets를 사용하여 애플리케이션 구성 데이터 관리
- Environment Variables를 통해 파드 내 구성 데이터 제공

Lab Time

배포 모델 설계

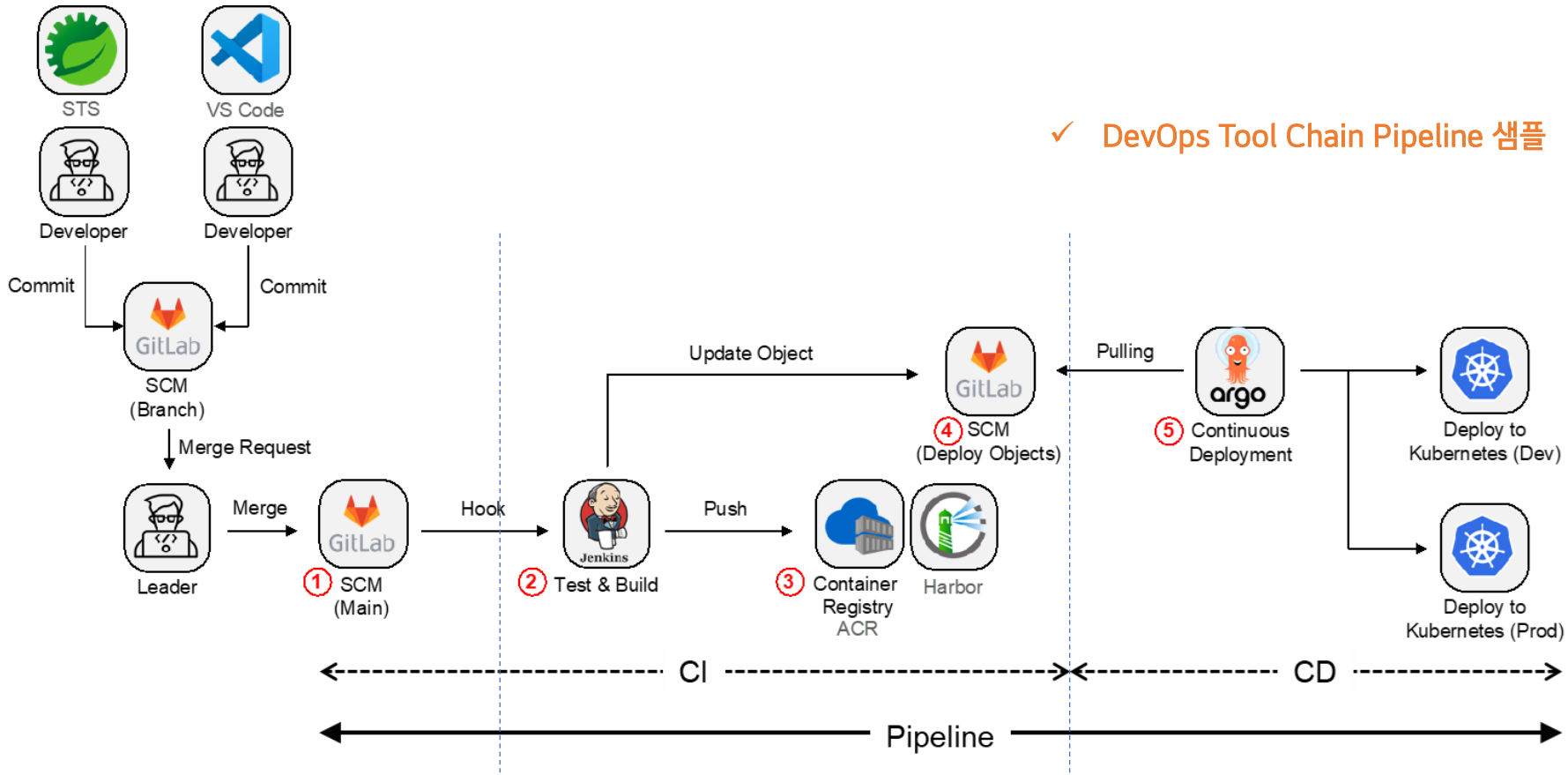




마이크로서비스 배포 자동화

- 이미지 빌드와 런타임 배포가 자동으로 진행되게 함으로써 자동화와 배포 일관성 유지 (Human Error 차단)
- 최신의 서비스를 엔드 유저에게 신속하게 제공하여 사용자 경험을 최적화하고, 빠른 피드백 루프에 따른 장애 조기 발견

✓ DevOps Tool Chain Pipeline 샘플



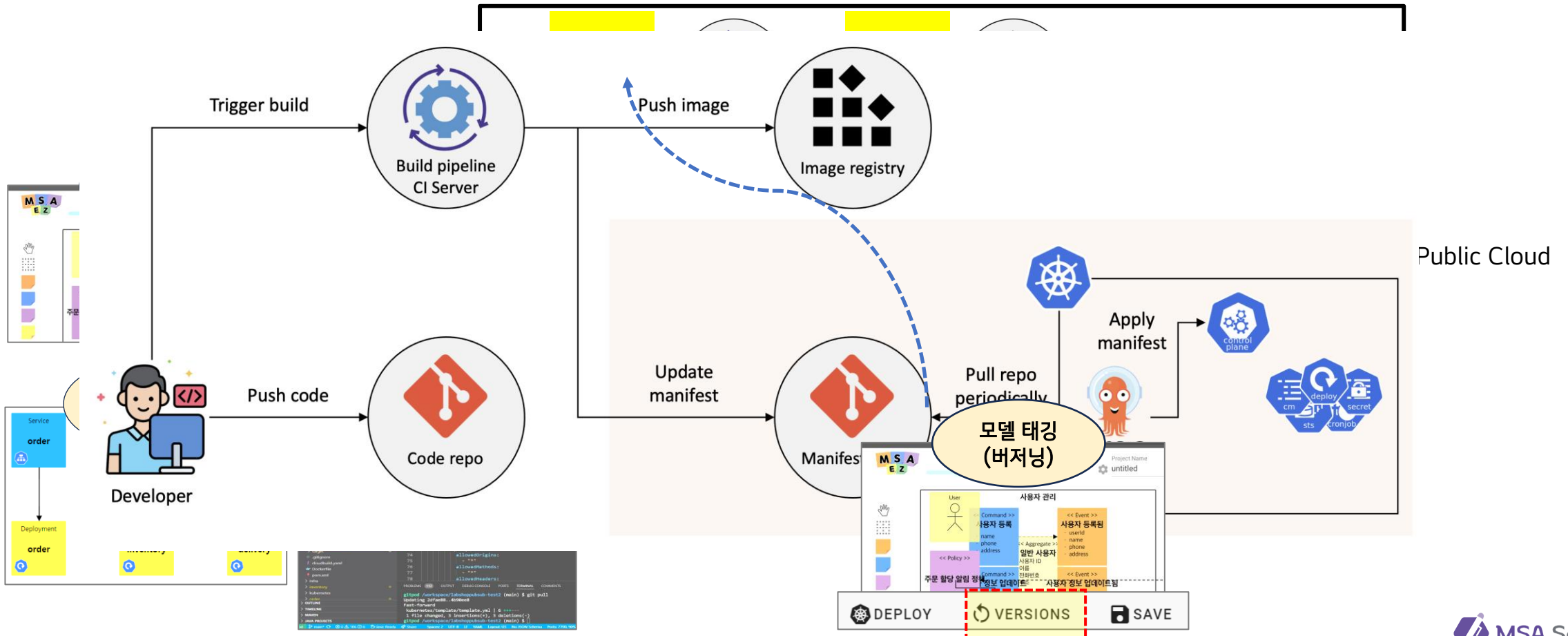
마이크로서비스 배포 자동화 tools

- 클라우드 네이티브 배포 파이프라인은 다양하며, 1) VM 기반 2) 호스팅 기반 3) 온 쿠버네티스 기반으로 분류 가능
- 각 유형별 배포 파이프라인은 특정 환경에 접합하며, 요구사항과 제약조건에 따라 선택 가능

Open Source Tools	CNCF Projects	Cloud Vendors'
Jenkins	argo	Google Cloud Build
TeamCity	flux	Azure Pipelines
Travis CI	TEKTON	AWS CodeBuild
GitHub Actions	JENKINS X	
CI/CD		
VM Install	On-Kubernetes	CSP Hosting

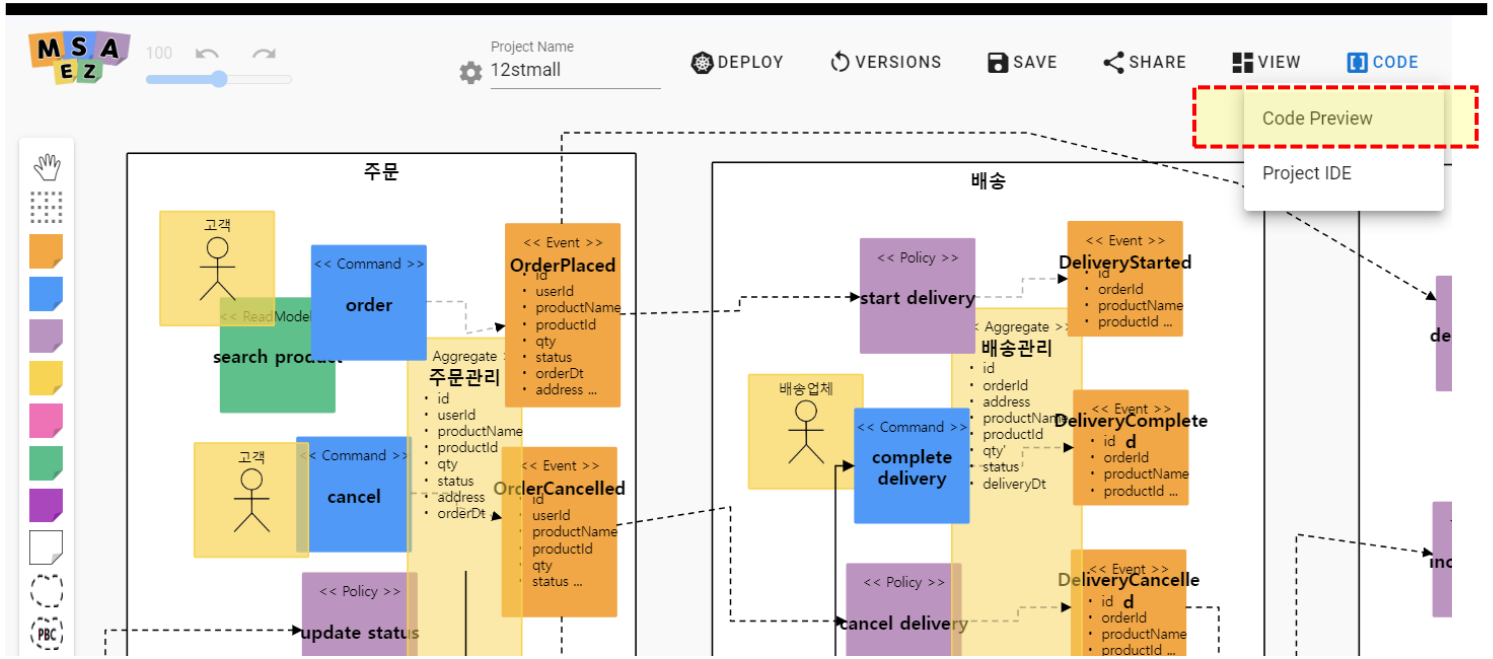
MSA-Ez 배포 자동화

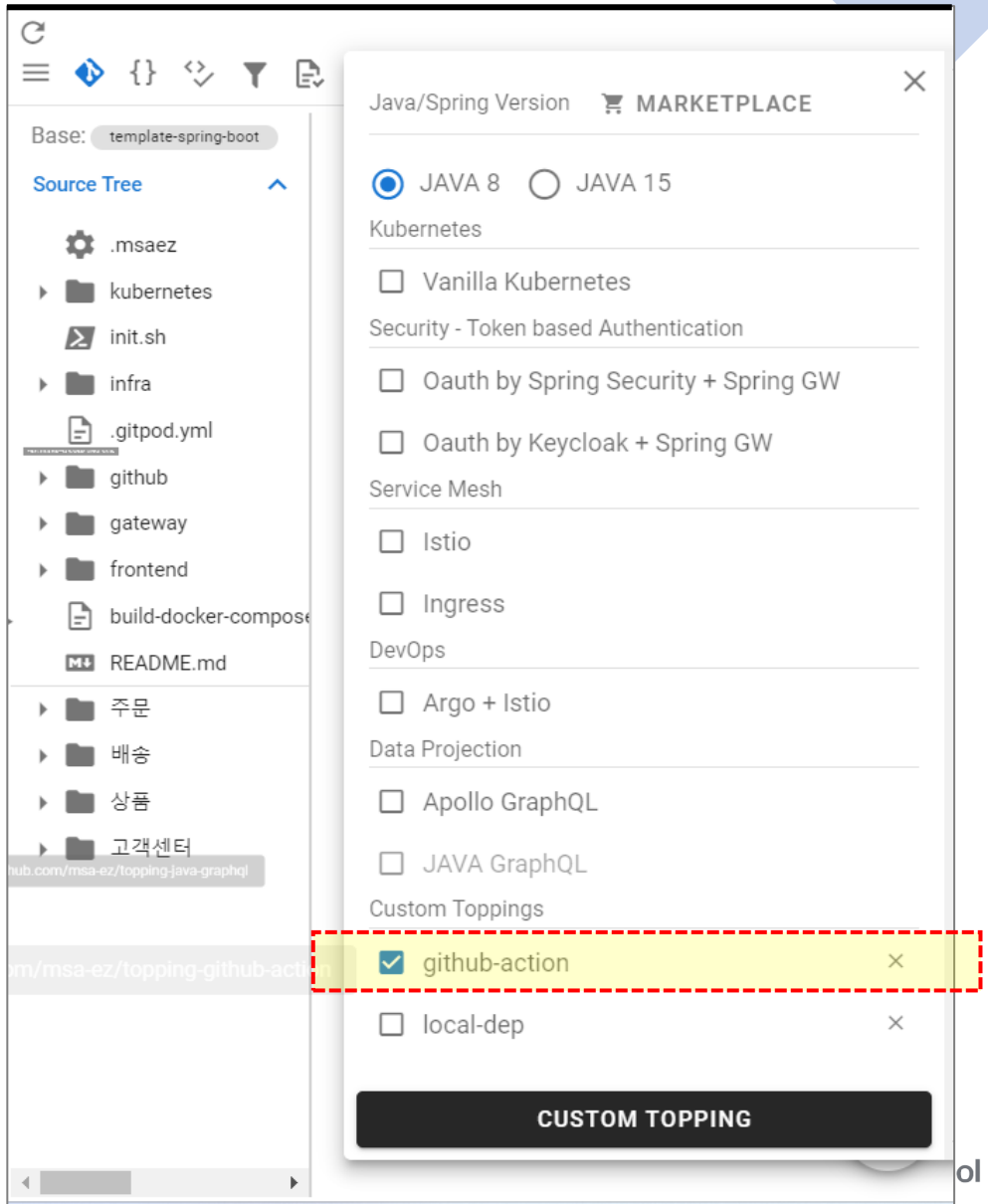
- 모델 태깅 시, Github Actions(GitLab CI)가 트리거 되며, 모델의 모든 이미지를 빌드하고 Git Registry에 Push



Lab Time

마이크로서비스 배포 자동화

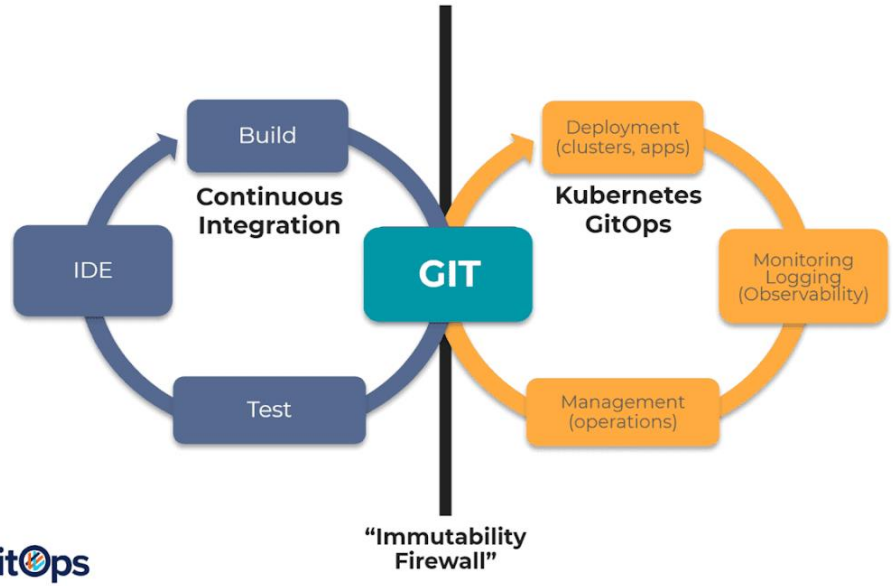




GitOps로 실천하는 DevOps

GitOps : An Operating Model for Cloud Native

- 데이터 관리 원칙 중 하나인 단일 진실 원천(Single Source of Truth, SSOT) 개념으로, 특정 데이터의 정확하고 신뢰할 수 있는 하나의 버전만 시스템 전체에 존재한다는 개념
- 모든 시스템 구성 요소가 동일한 데이터 소스를 참조하므로 데이터의 일관성이 유지되고 정확성과 관리 효율성 증대



Unifying Deployment, Monitoring and Management.

Git as the single source of truth of a system's desired state

ALL intended operations are committed by pull request

ALL diffs between intended and observed state with automatic convergence

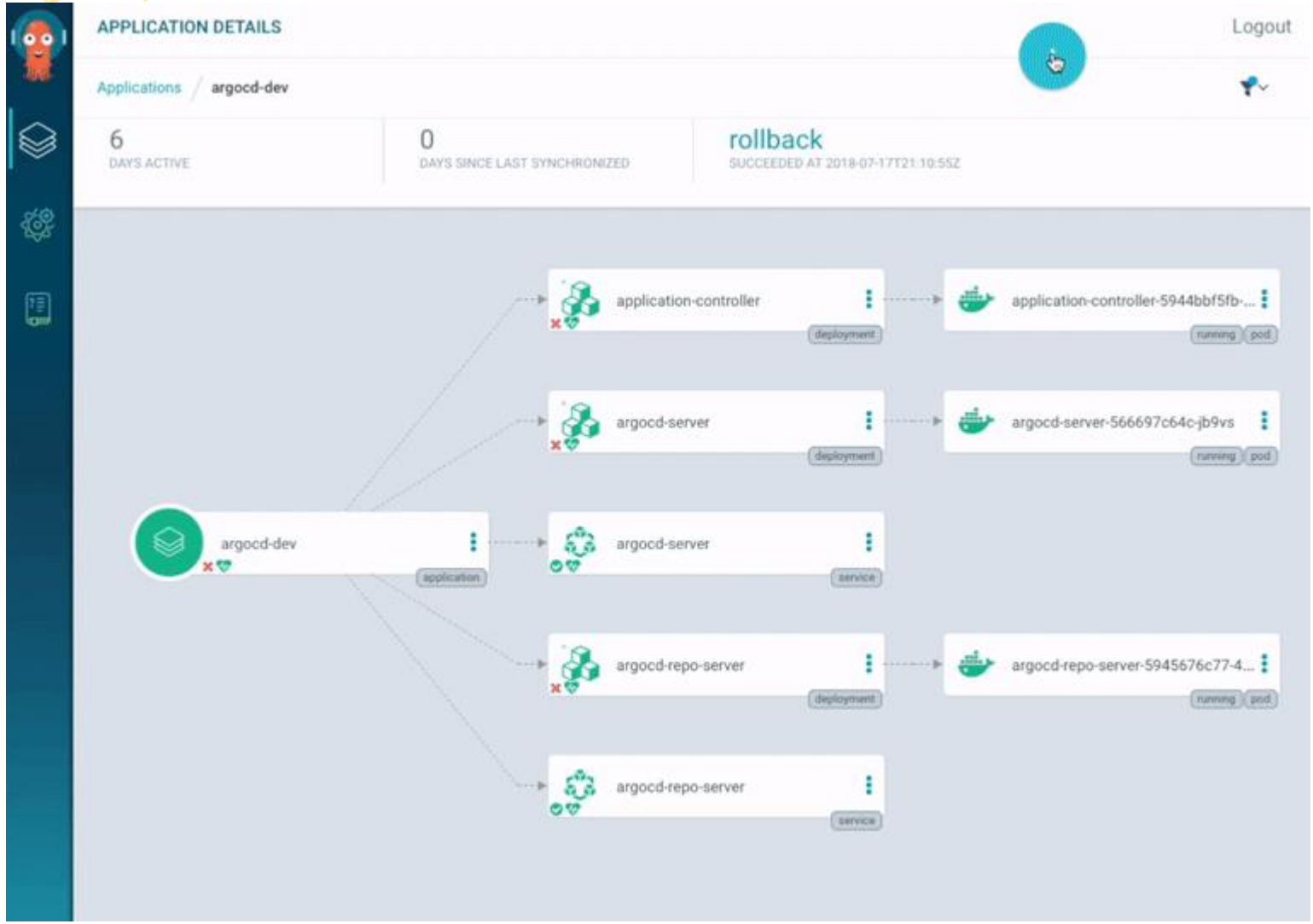
ALL changes are observable, verifiable, and auditable

weaveworks

출처: https://jennifersoft.com/ko/blog/kubernetes/2023-08-30-jennifer-kubernetes-3/

Lab Time

GitOps로 실천하는 DevOps



출처: https://jennifersoft.com/ko/blog/kubernetes/2023-08-30-jennifer-kubernetes-3/

ArgoCD

- 깃옵스 CD 구현에 가장 많이 사용되는 도구

장점

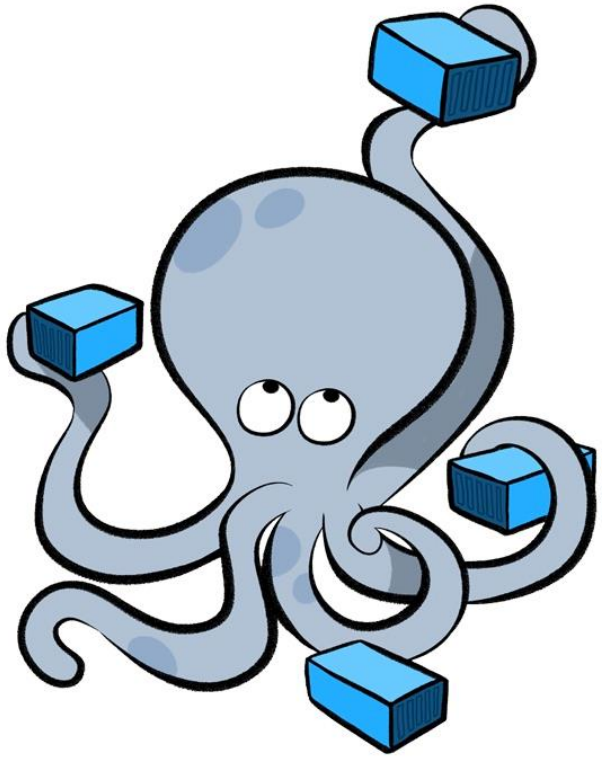
- 사용하기 편리한 UI 제공
- 다른 도구와의 손쉬운 통합
- 풍부한 사용 경험과 실 사례

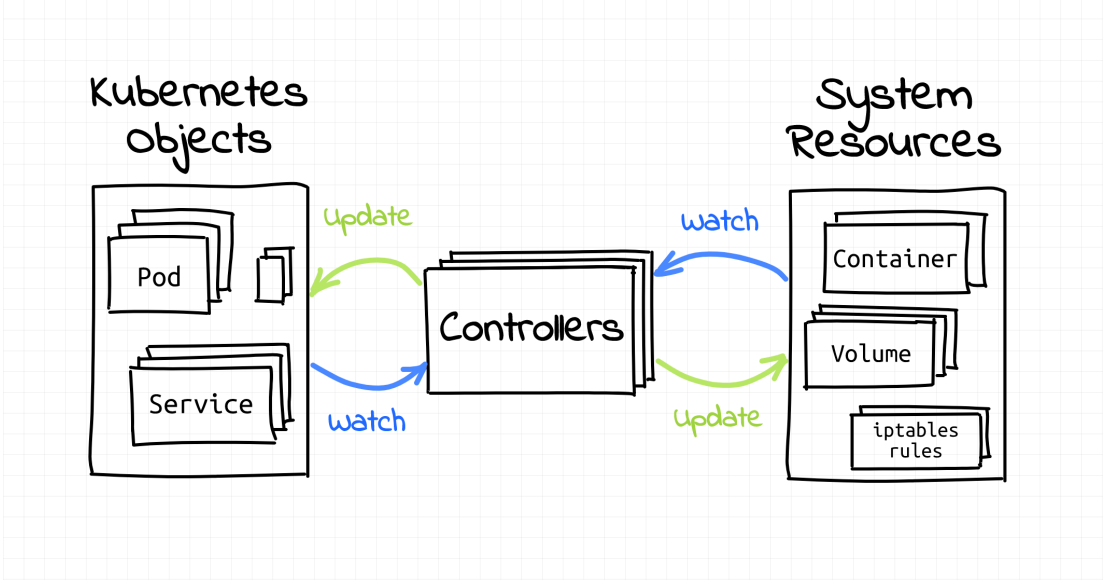
구동방식

- 지정된 Git 저장소와 쿠버네티스 클러스터를 지속적으로 모니터링
- 저장소 변경사항 감지 시, 자동으로 클러스터의 상태를 저장소의 상태와 동기화 (자동/수동 옵션)
- 변경 내역이 추적되므로 문제 발생 시, 이전 상태로의 손쉬운 롤백

컨테이너 오케스트레이션

- 컨테이너 오케스트레이터(De facto, 쿠버네티스) 상에서 컨테이너의 수명 주기를 관리하는 프로세스
- 배포 모델에서 설계된 각 마이크로서비스 운영 요소가 클라우드 네이티브 런타임에서도 정상적으로 유지되기 위한 관리 활동이나 도구





컨테이너 AIOps 도구

- CLI기반 쿠버네티스 클라이언트(Kubectl) 환경에서 AIOps 도구를 활용한 컨테이너 오케스트레이션 support

AI 플러그인	용 도
Kubectl ai	YAML Manifest 자동 생성
Kubectl gpt	커맨드 자동 실행
K8sGPT	Cluster 진단 및 운영 자동화 도구

8080 포트를 사용하는 order 서비스의 복제본 3개를 생성하는 배포 스펙을 만들어줘. 이미지는 apexacme/order:v1이야~

```
✨ Attempting to apply the following manifest:
apiVersion: apps/v1
kind: Deployment
metadata:
  name: order-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: order
  template:
    metadata:
      labels:
        app: order
    spec:
      containers:
      - name: order-container
        image: apexacme/order:v1
        ports:
        - containerPort: 8080
```

https://youtu.be/w6ZC3kZjtMg?t=341

컨테이너 AIOps 도구

- CLI기반 쿠버네티스 클라이언트(Kubectl) 환경에서 AIOps 도구를 활용한 컨테이너 오케스트레이션 support

AI 플러그인	용 도
Kubectl ai	YAML Manifest 자동 생성
Kubectl gpt	커맨드 자동 실행
K8sGPT	Cluster 진단 및 운영 자동화 도구

Order 서비스의 CPU 기반 평균 30% 이상 점유했을 때 10개까지 자동 확장되도록 설정해줘

```
gitpod /workspace/model-for-ops (main) $ kubectl gpt "배포된 order 서비스 명을 찾아
지 자동 확장하도록 설정해 줘. 배포된 order 이름은 order-deployment 야"
 ![WARNING] Please verify the generated commands before executing them on your k8s
nds, as GPT-generated commands may be inaccurate.
[Generated Command]
kubectl autoscale deployment order-deployment --cpu-percent=30 --min=1 --max=10
❄ Do you really want to execute this command? [y/N]: y
horizontalpodautoscaler.autoscaling/order-deployment autoscaled
gitpod /workspace/model-for-ops (main) $
```

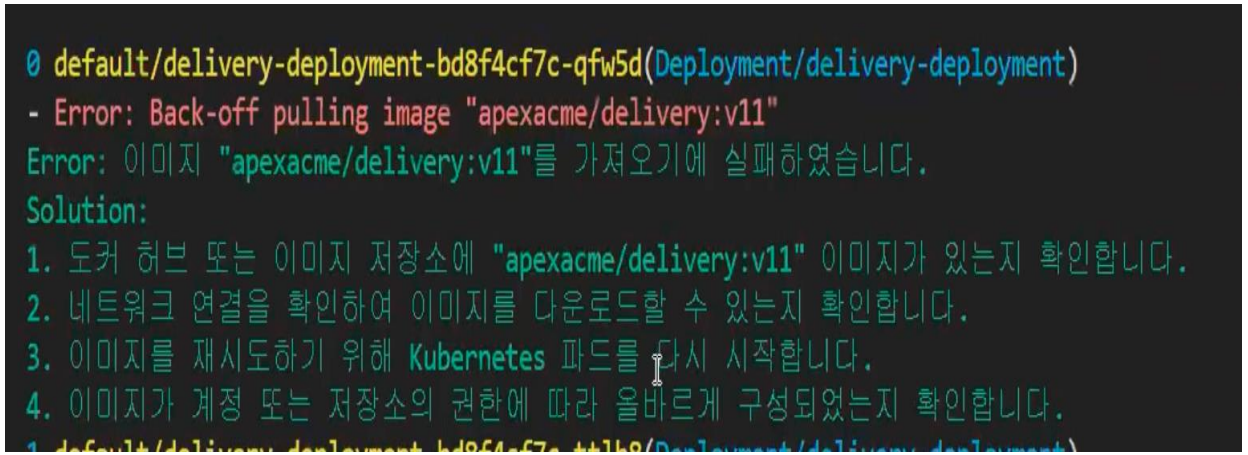
https://youtu.be/w6ZC3kZjtMg?t=835

컨테이너 AIOps 도구

- CLI기반 쿠버네티스 클라이언트(Kubectl) 환경에서 AIOps 도구를 활용한 컨테이너 오케스트레이션 support

AI 플러그인	용 도
Kubectl ai	YAML Manifest 자동 생성
Kubectl gpt	커맨드 자동 실행
K8sGPT	Cluster 진단 및 운영 자동화 도구

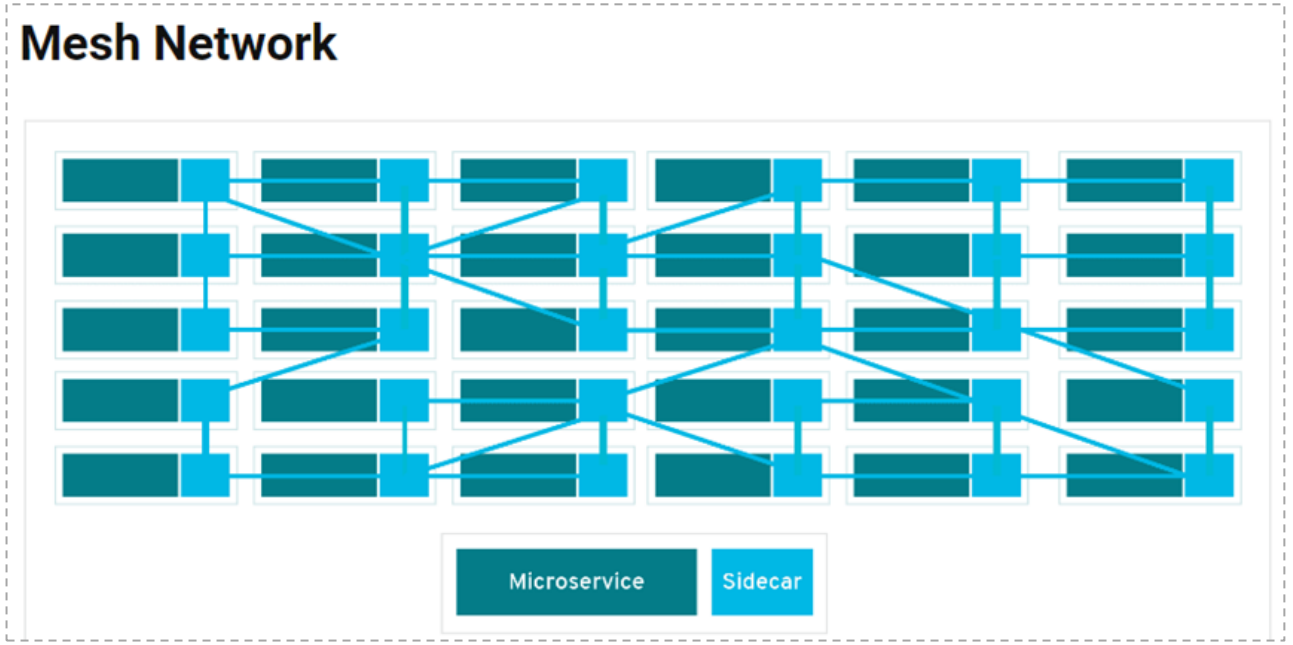
K8sGPT --explain --language 'Korean'



https://youtu.be/w6ZC3kZjtMg?t=1214

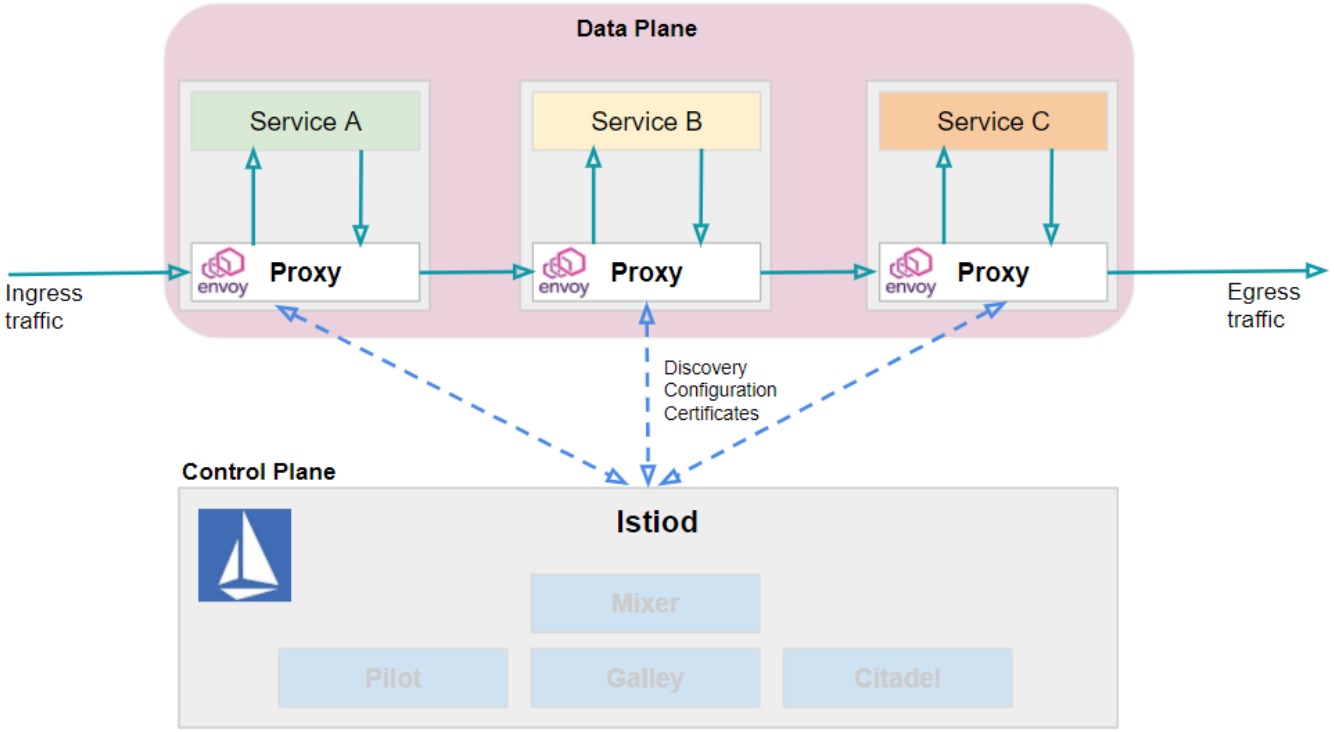
서비스 메시의 활용

- Service Mesh는 마이크로서비스간 통신(네트워크)을 위한 어플리케이션 레벨의 인프라 계층
- 통신 및 네트워크 기능을 비즈니스 로직과 분리한 네트워크 통신 인프라
- 서비스 메시에서의 호출은 마이크로서비스 개별 인프라 계층의 'Sidecar'라고 하는 Proxy를 통해 수행



Sidecar

서비스 메시의 활용



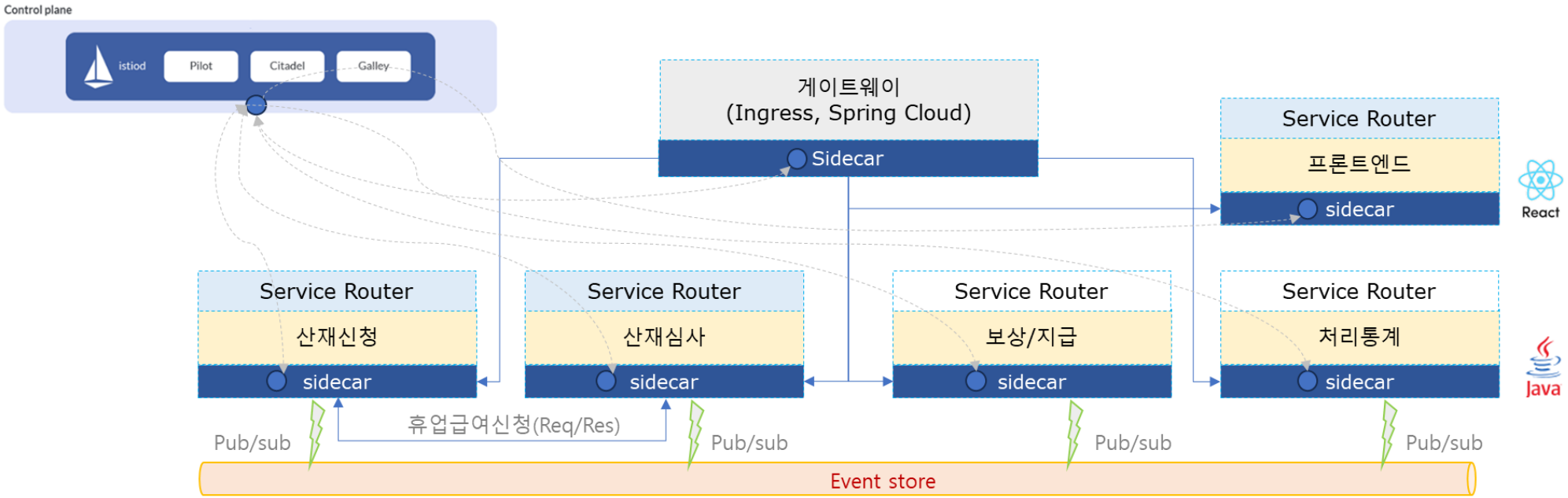
< Istio architecture >

- Data Plane
  - 서비스 옆에 사이드카(Envoy)를 붙여, 서비스로 들어오고 나가는 트래픽을 Envoy를 통해 통제
  - 유입/유출 트래픽 통제 : Ingress/Egress Controller
- Control Plane
  - Pilot : Envoy에 대한 설정 관리 및 서비스 디스커버리 기능 제공
  - Mixer : 액세스 컨트롤 및 다양한 모니터링 지표수집
  - Citadel : 보안관련 기능 담당 모듈로 사용자 인증/ 인가 및 TLS 통신을 위한 인증서 관리
  - Galley : Istio Configuration의 유효성 검사

출처 : https://azderica.github.io/00-architecture-istio/

서비스 메시의 활용

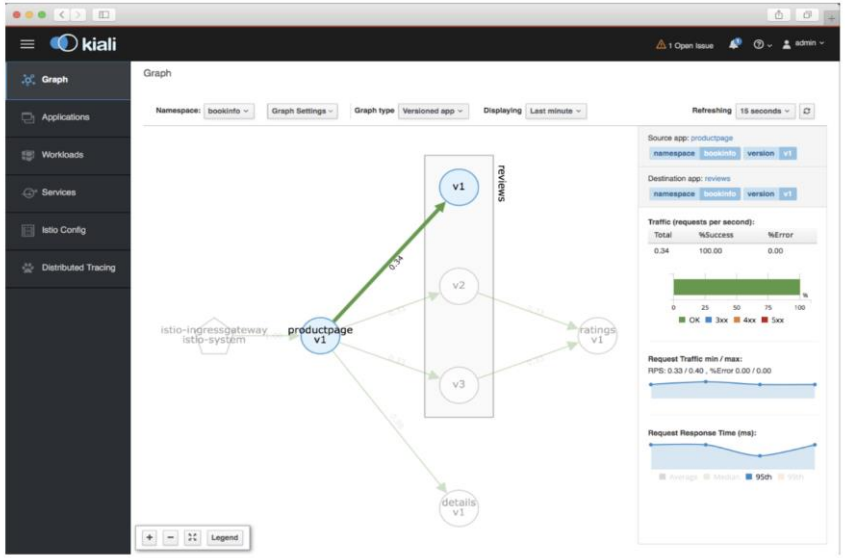
- 단일진입점 게이트웨이 Sidecar가 주제 도메인(산업재해 신청) 서비스를 구성하는 다운 스트림의 Sidecar를 호출
- 각 서비스(데이터 플레인)의 Sidecar는 컨트롤 플레인으로부터 Policy를 전달받거나, 수행 Metric 정보를 컨트롤 플레인에 제공



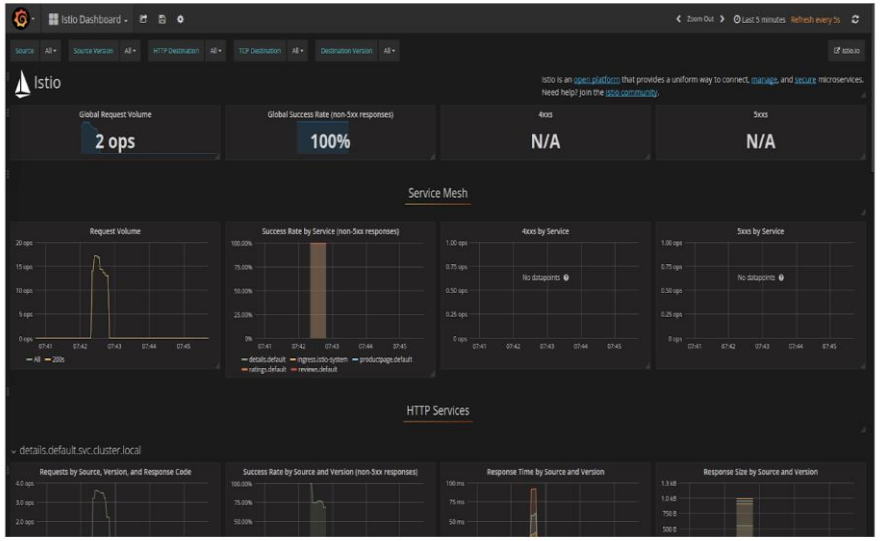
< Istio enabled architecture >

옵저버빌리티와 로그리게이션

- Distributed Tracing and Measure
- 서비스간 호출 내용 기록
- Istio 설치 시, Kiali, Jaeger, Loki, Grafana 가 Built-in 되어 각각 Monitoring, Tracing 및 로그리게이션 지원

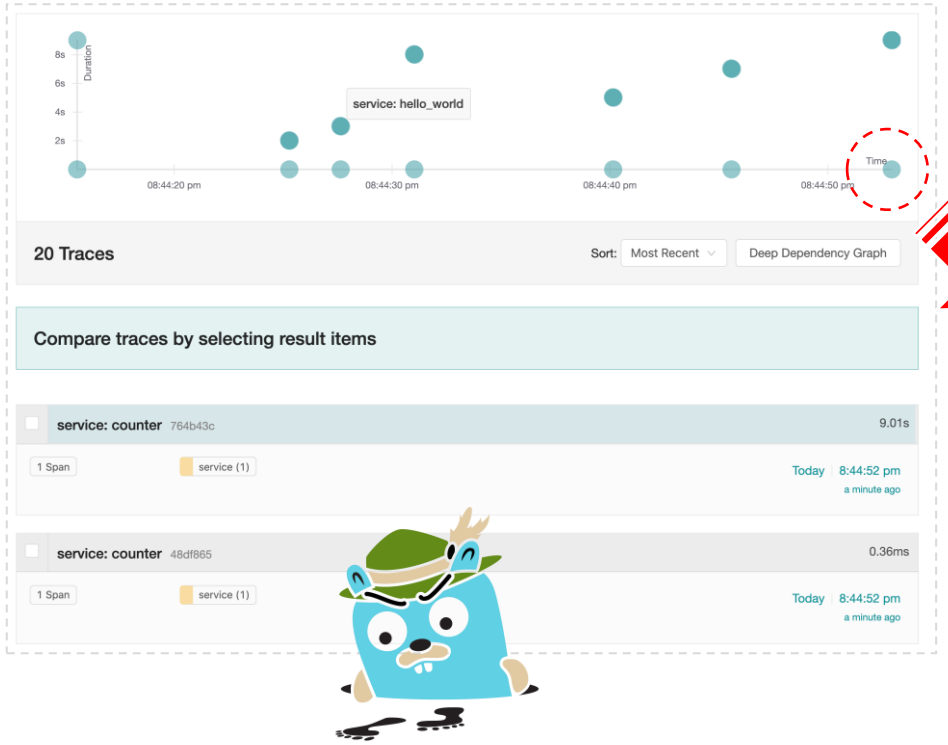


< Topology View >



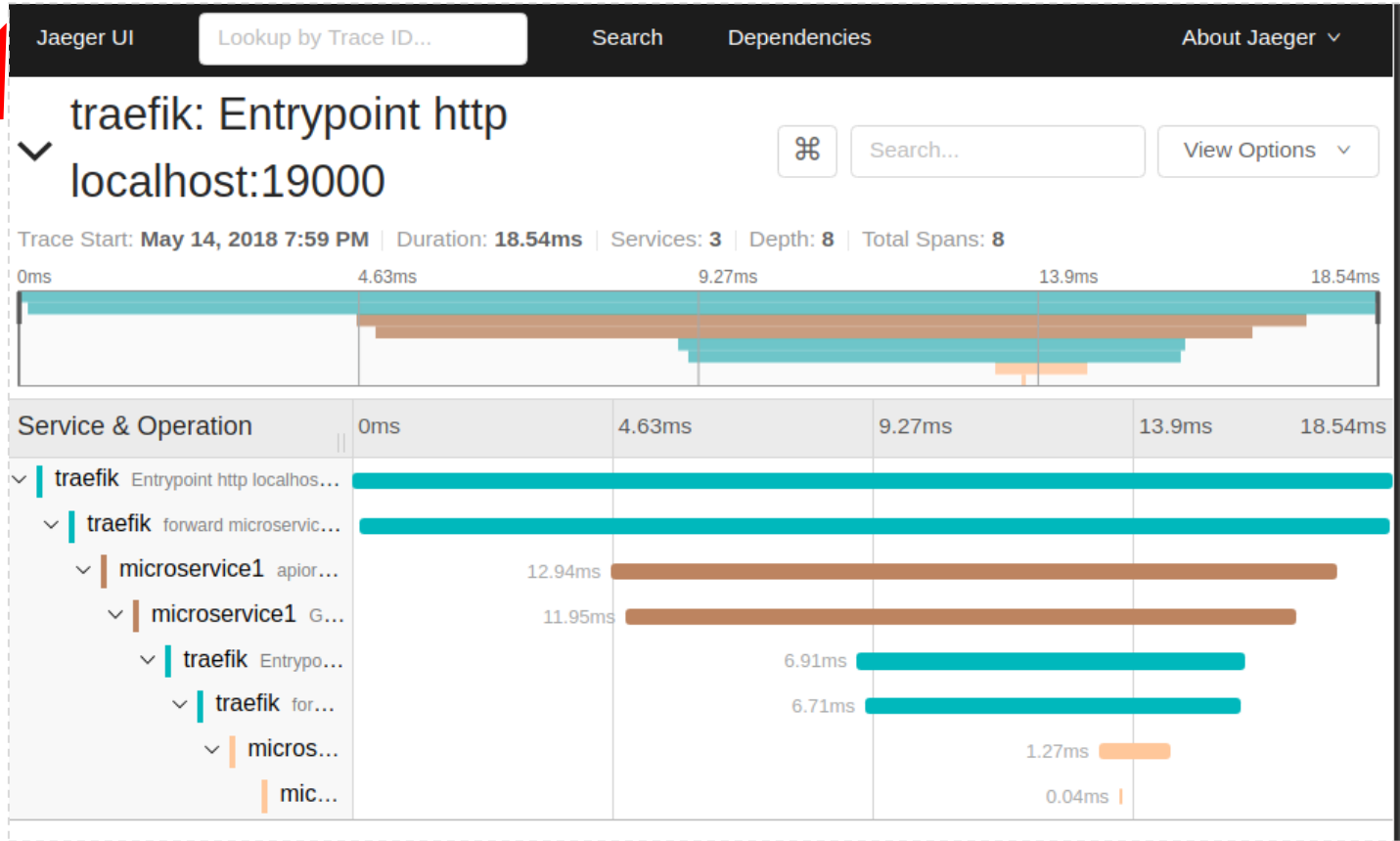
< Realtime Monitoring >

옵저버빌리티와 로그리게이션



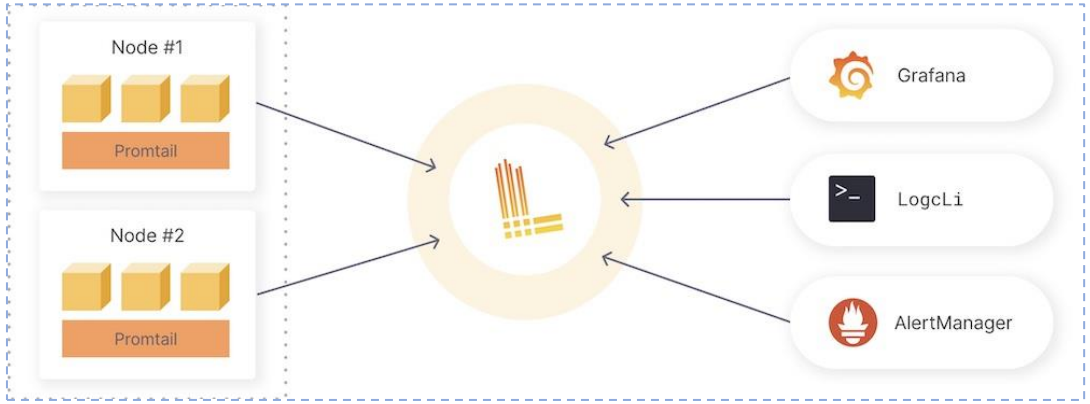
Zipkin, or Jaeger Stack

Tracing between Inner Architectures

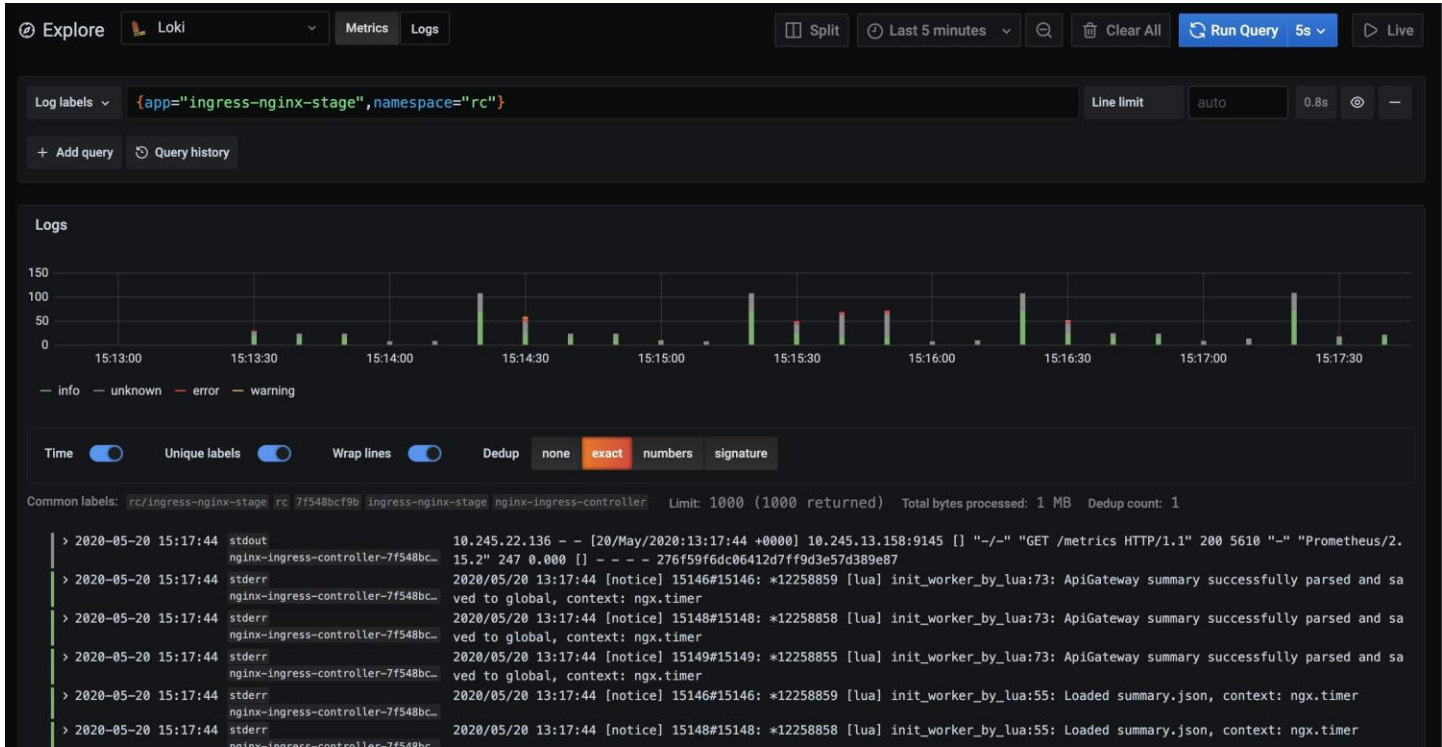


옵저버빌리티와 로그리게이션

- 일반적인 로그 수집 스택인 ELK(Elasticsearch, Logstash, Kibana), EF(Fluent-bit)K 와 유사
- 더 간단하고 클러스터 증가에 따른 관리 비용 효율화 차원에서 등장
- 로그 수집기인 Promtail과 Grafana를 통한 로그 시각화를 제공하는 새로운 중앙 로깅 시스템 (PLG)



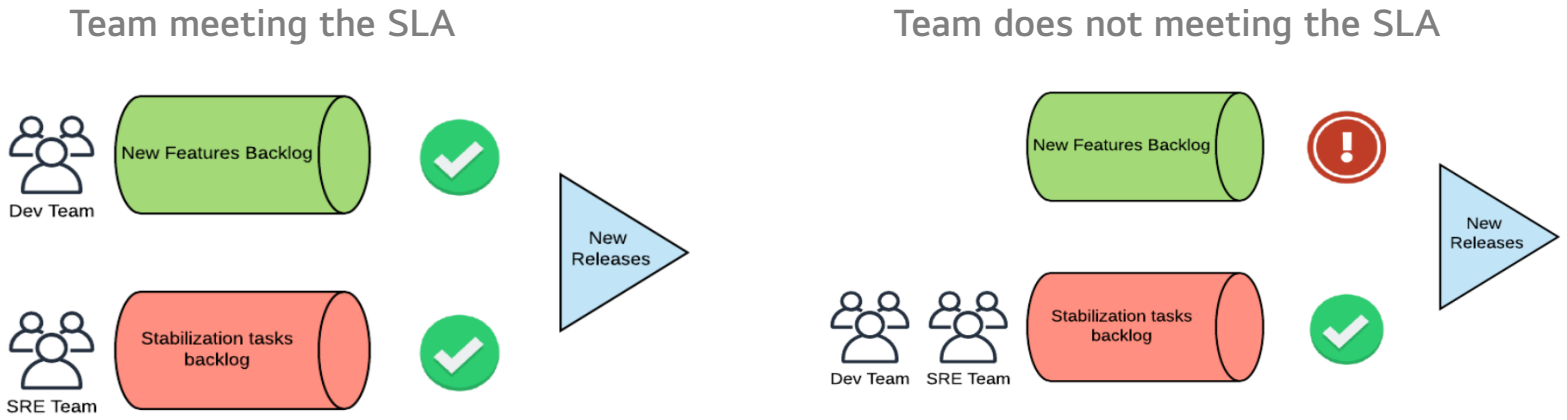
< PLG architecture >



Cloud Native SRE 활용

- SRE ?
  Site Reliability Engineering, 서비스 운영에 대한 소프트웨어 엔지니어링 접근 방식

- 서비스 수준 지표(SLI) 및 서비스 수준 목표(SLO)를 사용해 서비스 수준 계약(SLA), 즉 마이크로서비스에 요구되는 신뢰성을 정의함으로써 새로운 기능의 계속적인 출시 여부를 결정
- 서비스가 SLA 수준을 준수하면, 개발 팀은 언제든지 새 버전 출시 가능, 하지만 SLA 수준에 못 미치는 경우 SLA가 충족될 때까지 신규 버전 출시 불가 → 사이트 신뢰성 향상에 총력



SRE 측정 기준

- 첫번째, 가용성에 대한 명확한 정의 (Service-Level Indicator, SLI)
- 두번째, 가용성 목표 정의 (Service-Level Objective, SLO)
- 세번째, 장애 발생에 대한 계획 (Service Level Agreement, SLA)

SLIs | Metrics that you use to define the SLO targets

SLOs | Targets you set for the overall health of your services

SLAs | Promises you make about your service's health

SRE 측정 기준

- SLI (Service-Level Indicator) : 서비스 수준 측정을 위한 정량적 지표
  - 예시 : 응답시간(Request Latency), 에러율(Error Rate), 처리량(Throughput), 가용성 (Availability), etc
  - 모니터링 시스템을 통해 수집 주기, 수집범위를 정해 이를 지표화하고 시각화

- SLO (Service-Level Objective) : 지표에 대한 목표 값
  - SLO = SLI + 목표 값(Goal)
  - 예시 : “매주, 99% of REST API호출의 응답 시간은 100ms 이하여야 한다.” 에서 "매주,99% REST API 호출 응답시간" 이 SLI가 되고 응답시간 100ms가 목표 값이 되어 위의 전체 문장이 SLO가 됨
  - 단순하면서 완벽한 값을 사용하지 말고 가급적 적은 수의 SLO만 정의

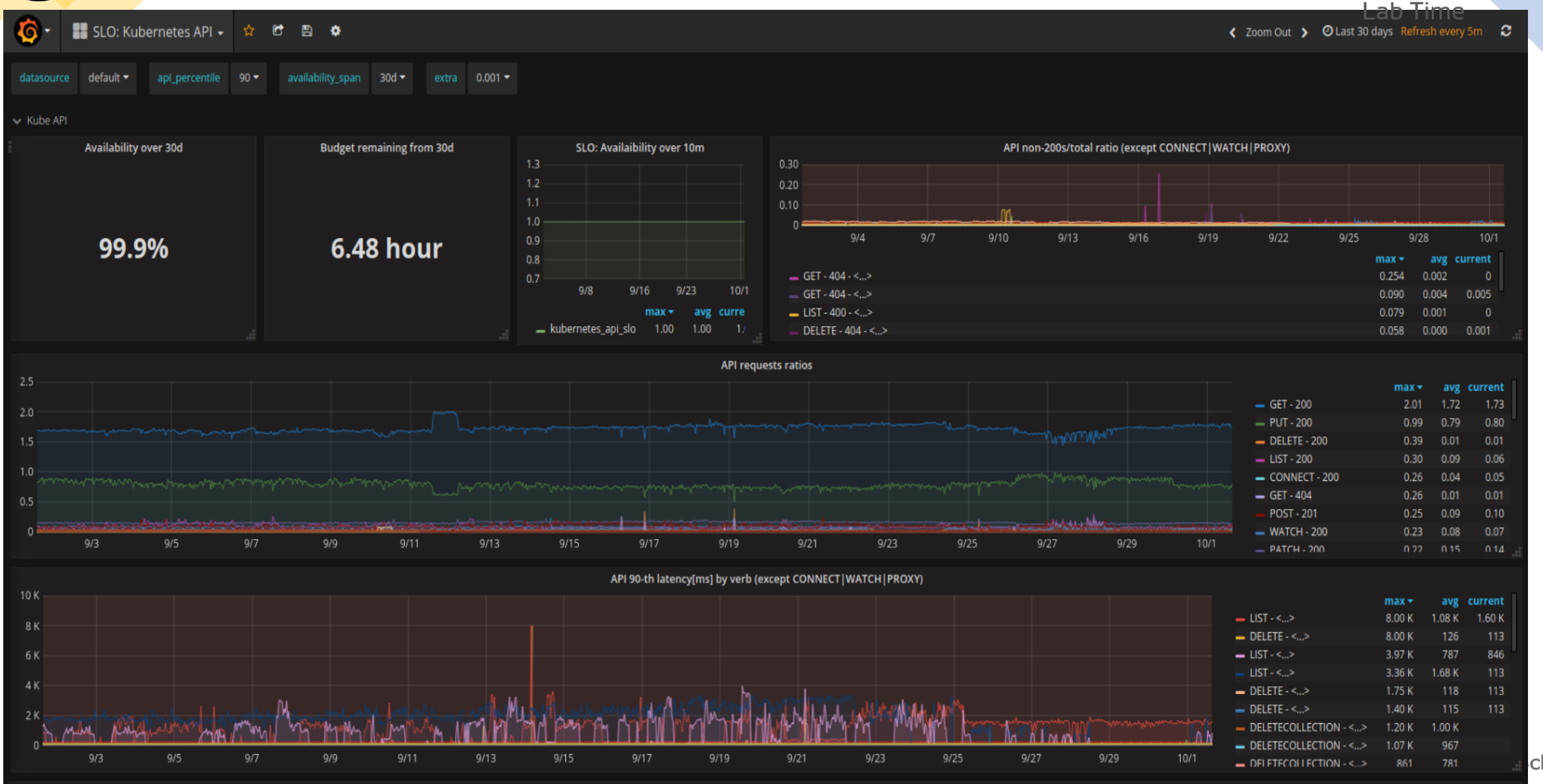
Error Budget ?

- 가용성에 따라서 허용되는 장애 시간
- Error budget = [100% - availability target]
  - 예를 들어, 한달에 SLO가 99.999%를 목표치로 설정했다면, 한달간 SLO는 0.001%의 다운 타임을 허용하게 되고, 이 0.001%가 Error budget이 된다.

- Error budget 활용
  - Error budget 은 계획된 다운 타임이나, 장애 등에 의한 계획되지 않은 다운 타임에도 차감
  - 만약 Error budget이 없다면 새로운 기능에 대한 업데이트를 중지하고, 시스템 가용성을 높이는 자동화나 프로세스 개선 등의 작업을 수행

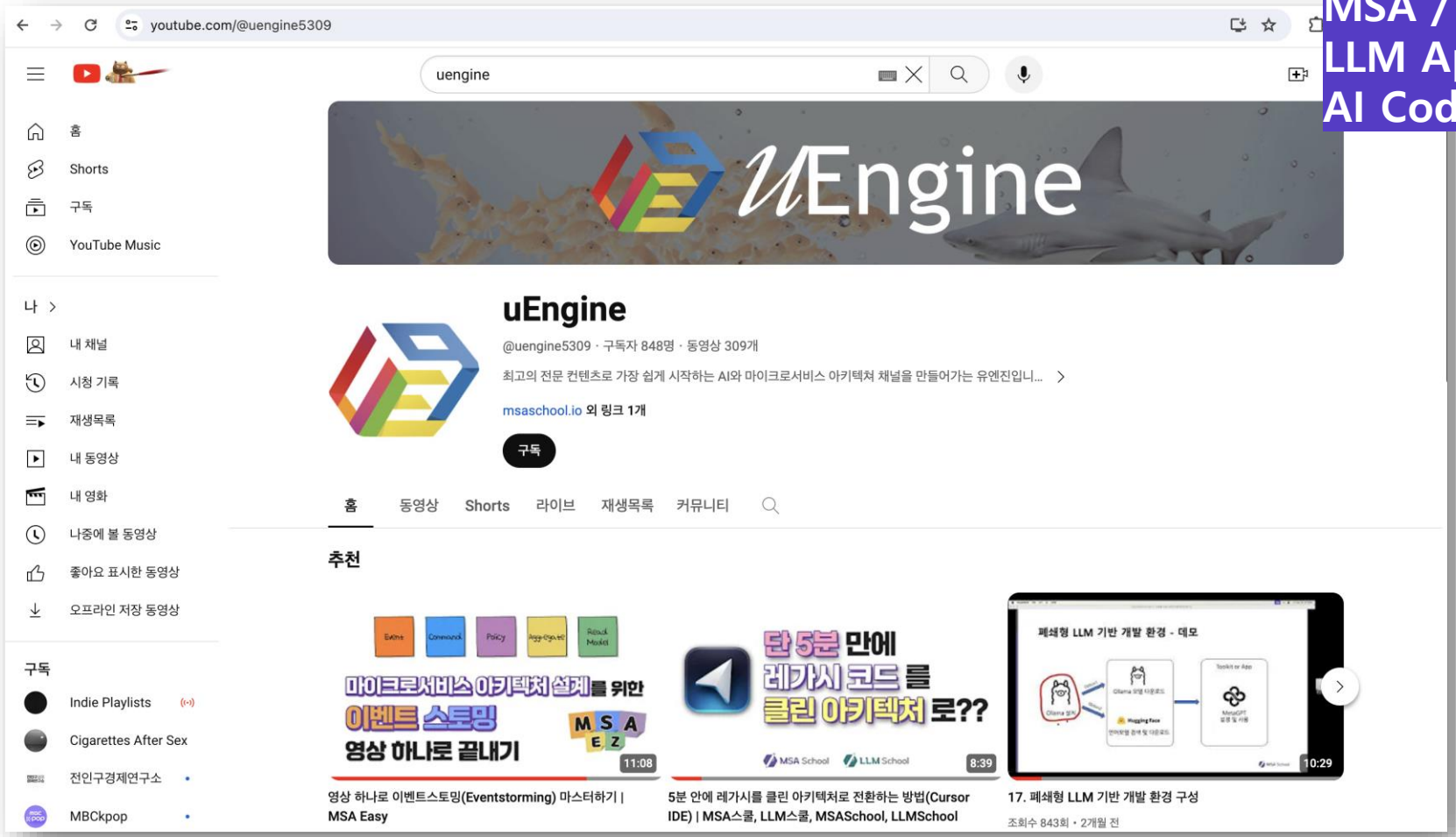
– Google –

SRE Monitoring



유엔진 콘텐츠를 계속 시청하시려면 유튜브를 구독하세요

주제:
MSA / CNA / DDD
LLM Application 개발방법
AI Coding / Legacy Modernization



MSA School