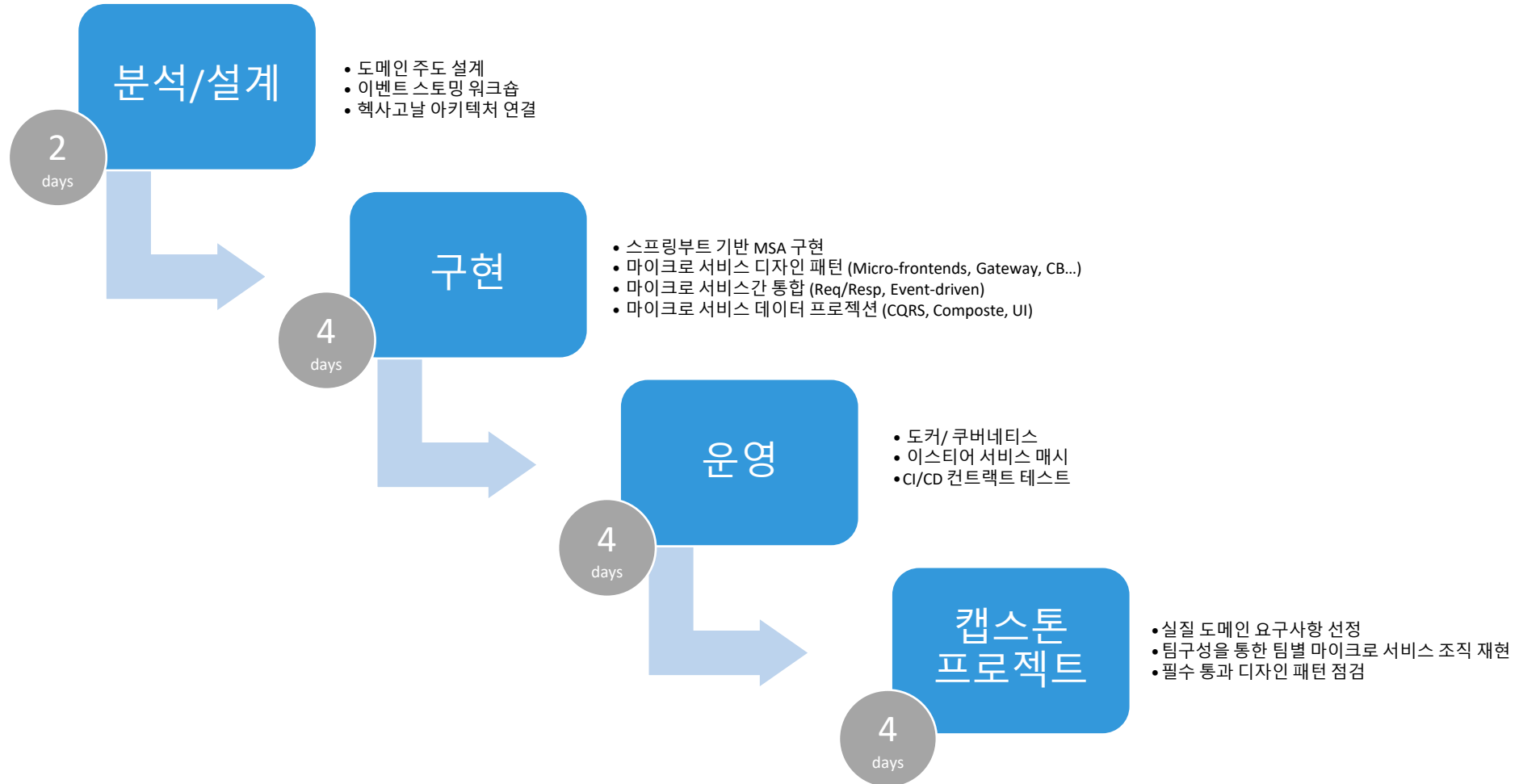


MSA School CNA Development 14Days Course

Full Lecture Itinerary - 14 일 완성 과정



| | | | | |
|------|--|-------------|------|--------------------------------|
| 커리큘럼 | Microservice Modeling | | 교육대상 | Business, Architect, Developer |
| 교육목표 | • MSA 아키텍처의 분석,설계,구현하기 앞서 이벤트 스토밍(Event Storming) 기법을 기반한 실질적 설계, 구현, 운영 기법 학습 | | | |
| 교육기간 | • 2일, 14시간 실습 (40%), 이론 (60%) | 선수지식(수강 요건) | 없음 | |

| 교육내용 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|-------|--------------|----|--|
| | Day 1 | 마이크로서비스 아키텍처 | 7H | <ul style="list-style-type: none">• 마이크로서비스아키텍처(MSA)의 출현 배경 (애자일기업과 MSA)• MSA 개념과 특성 마이크로서비스 아키텍처 문제 및 해결책• MSA의 진화: MSA 새시, 서비스 메시, 그리고 이벤트 드리븐 아키텍처• 최근의 MSA 구현 전략: 외부 (아우터) 아키텍처와 내부 (이너) 아키텍처• MSA 의 구축 라이프사이클: BizDevOps• MSA 실 적용 사례 살펴보기 |
| | Day 2 | 마이크로서비스 설계 | 7H | <ul style="list-style-type: none">• 전략적 설계의 정의• 서브도메인과 바운디드 컨텍스트를 통한 서비스 분해 전략• 컨텍스트 매핑 유형과 연동 프로토콜별 특성 (RPC, REST, Pub/Sub)• 이벤트스토밍 기법을 활용한 바운디드 컨텍스트와 마이크로서비스 식별• 이벤트 드리븐 마이크로서비스 아키텍처의 도출• 기술적 설계의 정의 및 도메인 모델링 이해• 도메인 모델링 (Entity, Value Objects)• 헥사고날 (어니언) 아키텍처를 통한 서비스 구현 전략 |

| | | | | |
|------|--|-------------|------------|----------------------|
| 커리큘럼 | Developing Cloud Native Application | | 교육대상 | Developer, Architect |
| 교육목표 | • Cloud 플랫폼과 Core 서비스를 이해하고 시스템을 구성하고 클라우드 애플리케이션을 개발 · 운영 | | | |
| 교육기간 | • 3일, 14시간 실습 (80%), 이론 (20%) | 선수지식(수강 요건) | 모델링, 쿠버네티스 | |

| 교육내용 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|-------|-----------------------|----|---|
| | Day 3 | 개발 실습 개요 | 1H | <ul style="list-style-type: none">개발 대상: 온라인 커머스 서비스개발 서비스 품질 목표: Resiliency, Scalability, Responsiveness조직의 전환: Horizontal to Vertical, Matrix Organization |
| | | 마이크로서비스의 분석과 설계 | 2H | <ul style="list-style-type: none">DDD와 MSA Design Patterns을 적용한 마이크로서비스 구현 전략, Cloud Native Application 의 특성과 구현원칙들서비스 식별과 분해방법들 – Core Domain 과 Supporting Domain, Bounded Context, Aggregate, Business Capability, Mini Service분해된 서비스의 연동 방법 – Context Mapping, Request/Response or Event-driven MSAOuter Architecture 구성요소 – API GW, 분산큐(Kafka), 서비스 레지스트리, 로드밸런서, 서킷브레이커, 사이드카 패턴 등 |
| | | 단위 마이크로서비스의 구현 | 4H | <ul style="list-style-type: none">DDD 와 이벤트 스토밍의 결과를 hexagon 아키텍처로 매핑하는 방법Spring Boot와 Maven 을 이용한 단위 서비스 포장JPA 를 이용한 도메인 모델과 어댑터 구현 (Entity, Value Objects)Spring Data REST + JPA 를 통한 RESTful API 노출 (HATEOAS) |
| | Day 4 | 프론트엔드를 통한 마이크로 서비스 통합 | 4H | <ul style="list-style-type: none">UI 를 기반한 서비스 통합 전략: Client-side Rendering, SPA, MVVM, HATEOASMVVM Front-end 프레임워크: VueJS, React, Angular2API Gateway 를 통한 진입점 단일화, 전급 관리 및 인증 통합OAuth2 와 JWT 통한 Stateless 한 토큰 기반 접근과 인증Front-end 기반 통합의 한계: 트랜잭션 이슈와 성능 문제 |

| | | | | |
|------|--|-------------|------------|----------------------|
| 커리큘럼 | Developing Cloud Native Application | | 교육대상 | Developer, Architect |
| 교육목표 | <ul style="list-style-type: none"> Cloud 플랫폼과 Core 서비스를 이해하고 시스템을 구성하고 클라우드 애플리케이션을 개발 · 운영 | | | |
| 교육기간 | <ul style="list-style-type: none"> 3일, 14시간 실습 (80%), 이론 (20%) | 선수지식(수강 요건) | 모델링, 쿠버네티스 | |

| 교육내용 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|---------|---------------------------------------|----|---|
| | Day 5-6 | 인터-마이크로서비스의 통합 1: Request-Response 방식 | 2H | <ul style="list-style-type: none"> Request-Response 방식의 연동 (REST + Sync + Orchestration) 트랜잭션 이슈: 2PC or Shared-Database and schema per service 추적 이슈: 분산 추적을 통한 오류 점검 방법, Istio 예거와 kiali 사용 방법 성능 이슈: 장애전파 상황 그리고 Istio 의 Circuit Breaker, Rate Limiting를 통한 회피 |
| | | 인터-마이크로서비스의 통합 2: Event-driven 방식 | 3H | <ul style="list-style-type: none"> Event-driven 방식의 연동 (Pub/Sub + Async + Choreography) EDA 에서의 분산 큐의 역할과 Kafka 의 장점 Kafka 의 설치와 클라이언트 도구를 통한 Pub/Sub 실습 이벤트의 퍼블리싱 – 어그리게이트 내의 엔티티에 JPA 어노테이션 주입 방법, Spring Cloud Streams 를 통한 이벤트 퍼블리시 이벤트의 서브스크립션 - Spring Cloud Streams 와 Kafka client 를 통한 이벤트의 수신과 처리, Acknowledge 트랜잭션 이슈: Eventual Transaction 와 Saga, Compensation 실습 통한 성능의 장점 이해 : Time-decoupling 과 Non-blocking 속성 |
| | | 데이터 프로젝션 | 2H | <ul style="list-style-type: none"> Join SQL 를 회피하고, 데이터를 취합하여 보여주는 전략 3가지: by UI, by Composite-service, by CQRS Composite-Service 구현방법: 동시 REST 호출 or API GW (GraphQL) 사용 CQRS 구현방법 |
| | | | | |

| | | | | |
|------|---|-------------|------|--------------------------------|
| 커리큘럼 | DevOps: Container Orchestration(Docker & Kubernetes) | | 교육대상 | Architect, Developer, Operator |
| 교육목표 | • Kubernetes와 Docker 의 설치, 애플리케이션 배포, 무정지 재배포, 운영자동화의 실습 및 학습 | | | |
| 교육기간 | • 3일, 21시간 실습 (50%), 이론 (50%) | 선수지식(수강 요건) | 없음 | |

| 교육내용 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|-------|--------------------------------------|----|--|
| | Day 7 | MSA와 DevOps 그리고 Kubernetes | 7H | <ul style="list-style-type: none">마이크로서비스와 DevOps의 적용사례 – 아마존과 넷플릭스컨테이너와 컨테이너 오케스트레이터의 등장컨테이너 기반 애플리케이션 디자인 패턴구글의 Kubernetes를 기반한 MSA와 DevOps 환경의 적용 시나리오 |
| | | Kubernetes 의 구조, 활용 시나리오의 전반적 이해 | | <ul style="list-style-type: none">Kubernetes 등장 배경 (Borg), Features, ArchitectureKubernetes Runtime MechanismKubernetes Object Model 과 Declarative Configuration 개념Kubernetes 의 설치: Configuration, Infrastructure, Localhost vs On-premise vs Cloud Installation, Tools/ResourcesPublic Kubernetes 서비스 (GCP / AWS / Azure) 계정 생성과 접속샘플 MSA 애플리케이션의 디플로이 전과정 미리보기 |
| | Day 8 | Kubernetes 서비스 설치와 운영 | 7H | <ul style="list-style-type: none">Kubernetes 기본 명령과 Kubectl 를 통한 객체 다루기Kubernetes 를 통한 애플리케이션 디플로이, 인터넷으로의 서비스 노출과정Azure VM 서비스에서의 Node 확인과 pod 디플로이 전략 확인Pod, Deployment, ReplicaSet, StatefulSet, DaemonSet 등 객체 세부Self healing, Auto Scaling, Zero-Down time DeploymentLiveness & Readiness 의 설정, 트러블슈팅 방법Volume, PersistenceVolume, PersistenceVolumeClaimYaml, Helm 를 이용한 멀티 티어 애플리케이션 프로비저닝 |
| | Day 9 | 쿠버네티스를 이용한 Micro Services and DevOps | 7H | <ul style="list-style-type: none">MSA Application 서비스 시나리오: 샘플 애플리케이션애플리케이션 패키징을 위한 컨테이너 (도커)의 이해마이크로 서비스 채시 (Chassis): 스프링 클라우드와 넷플릭스 OSS, 그리고 도커를 통한 서비스 패키징Container Registry 를 이용한 도커 레지스트리에 서비스 퍼블리싱애플리케이션 디플로이먼트 디스크립터의 설정, Liveness & Readiness 설정서비스 메시 Istio의 개념과 쿠버네티스와 관계 이해Istio 를 통한 Resilience 향상: 서킷 브레이킹, 레이트 리미팅, 리트라이, 폴이젝션Istio 를 통한 Smart Deploy: 동적 트래픽 라우팅, AB Testing, 카나리 디플로이, 새도우 디플로이Istio 를 통한 모니터링: 프로메테우스를 통한 모니터링 수집, 예거, 키알리를 통한 분산 서비스 추적 프로파일링 |

| | | | | |
|------|---|-------------|-------|--------------------------------|
| 커리큘럼 | DevOps with CI/CD | | 교육대상 | Operator, Developer, Architect |
| 교육목표 | • 퍼블릭 클라우드 환경의 DevOps 환경중 CI/CD 를 기반으로 빌드하고 분석하고 배포(deployment)하는 방법을 학습한다. | | | |
| 교육기간 | • 1일, 7시간 실습 (60%), 이론 (40%) | 선수지식(수강 요건) | 쿠버네티스 | |

| 교육내용 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|--------|-------------------------------------|----|--|
| | Day 10 | CI/CD 프로세스와 Azure 의 대응 서비스 | 3H | <ul style="list-style-type: none">• CI (Continuous Integration) / CD (Continuous Delivery) 개요• CI/CD Pipeline 구성요소 – Source Repository, Builder, Pipeline, Image Registry, Container Orchestrator• Azure 의 대응 서비스 – Azure Repos Git, Azure Pipeline and Azure Container Registry, 그리고 Azure Kubernetes Service |
| | | CI/CD Pipeline with Azure Pipelines | 4H | <ul style="list-style-type: none">• Maven 개념과 라이브러리 디펜던시 (pom.xml) 의 구조 이해• GCB or Azure Pipelines Yaml 파일 구조• 소스코드의 변경을 통한 Git 트리거를 통한 파이프라인 실행• GCB or Azure Pipelines 대시보드 |

| | | | | |
|------|---|-------------|------|--------------------------------|
| 커리큘럼 | Capstone Project: Developing on Cloud | | 교육대상 | Operator, Developer, Architect |
| 교육목표 | <ul style="list-style-type: none">클라우드 플랫폼에서 마이크로서비스 아키텍처 기반으로 클라우드 네이티브 애플리케이션을 개발 하는 프로젝트 수행 방식의 교육 | | | |
| 교육기간 | <ul style="list-style-type: none">4일, 28시간 실습 (100%) 및 평가 | 선수지식(수강 요건) | All | |

| 수행방법 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|--------|------------------|----|--|
| | Day 11 | 평가 방법 설명 및 환경 설정 | 3H | <ul style="list-style-type: none">통합 평가 기준과 체크 포인트 설명캡스톤 프로젝트 선정 기준 설명프로젝트 결과 발표 방법 설명팀선정, 팀장선출 |
| | | 팀프로젝트 모델링 | 4H | <ul style="list-style-type: none">팀별 이벤트 스토밍 통한 도메인 모델링강사: 이벤트 스토밍 퍼실러테이션 |

| 수행방법 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|--------|----------|----|--|
| | Day 12 | 팀프로젝트 개발 | 7H | <ul style="list-style-type: none">개발 진행강사:, 질의응답, 문제해결, 가이드 |

| | | | | |
|------|---|-------------|------|--------------------------------|
| 커리큘럼 | Capstone Project: Developing on Cloud | | 교육대상 | Operator, Developer, Architect |
| 교육목표 | <ul style="list-style-type: none">클라우드 플랫폼에서 마이크로서비스 아키텍처 기반으로 클라우드 네이티브 애플리케이션을 개발 하는 프로젝트 수행 방식의 교육 | | | |
| 교육기간 | <ul style="list-style-type: none">4일, 28시간 실습 (100%) 및 평가 | 선수지식(수강 요건) | All | |

| 수행방법 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|--------|----------|----|--|
| | Day 13 | 팀프로젝트 배포 | 4H | <ul style="list-style-type: none">쿠버네티스와 CI/CD 를 이용하여 배포강사: 질의응답, 문제해결, 가이드 |
| | | 팀프로젝트 평가 | 3H | <ul style="list-style-type: none">조별 발표: 개발 내역 체크포인트에 따라강사: 평가 |

| 수행방법 | 기간 | 주제 | 시간 | 주요 학습내용 |
|------|--------|------|----|--|
| | Day 14 | 개인개발 | 3H | <ul style="list-style-type: none">조별 개발 내용에서 추가 마이크로서비스 주제 선정추가 마이크로 서비스에 대한 분석설계 (이벤트스토밍), 개발, 운영의 시나리오로 개발팀별 평가의 동일한 체크포인트 |
| | | 개인평가 | 4H | <ul style="list-style-type: none">개인 발표강사: 평가 후 총괄 개인별 점수 산정 → 보고 |

교육콘텐츠 및 도구

Biz-Dev-Ops 전과정에
요구되는 방법론, 설계패턴,
구현기법을 실질적인
예제를 바탕으로 구성한
온라인 도서를 제공

<http://msaschool.io>

MSA School 소개

예제 도메인

사용 플랫폼

예제 애플리케이션 둘러보기

관련 리소스

SW 및 도구

소개

계획단계

설계/구현/운영단계

참고자료

커뮤니티 및 교육

//Engine

힘난한 MSA 구축 여정의 길라잡이

1단계 분석 2단계 설계 3단계 구현 4단계 통합 5단계 배포 6단계 운영

Biz Dev Ops

계획에서 운영까지 End-to-End 학습을 위한 단계별 효율적인 실전법 제시

- BizDevOps 전 과정 지원
- 실전 MSA 코드 활용
- 눈높이에 맞춘 특화된 교육

MSA 스쿨에 오신 것을 환영합니다.

최근, 마이크로서비스를 활용한 비즈니스 구축 니즈와 이와 관련한 사전 교육 및 컨설팅 요구는 계속적으로 증가하고 있는 추세에 있습니다. 특히, 대기업을 중심으로 모든 신규 프로젝트는 마이크로서비스 아키텍처를 우선 고려할 정도로 IT 시장에서의 MSA는 이미 깊숙히 자리 잡고 있습니다.

MSA School 은 이러한 시장 상황에서 보다 구체적이고 실행 가능한 사례 중심의 구현 전학을 소개하려는 목적으로 설립되었습니다.

교육콘텐츠 및 도구

이벤트 기반
마이크로서비스의 설계를
위한 이벤트스토밍 도구와
구현을 위한 클라우드IDE,
그리고 컨테이너 기반
인프라 설계를 위한
쿠버네티스 모델링 도구를
학습과정에 연계하여 강의
과정을 진행할 수 있는 도구

<http://msaez.io>

The screenshot displays the MSAEZ.io web application interface. At the top, there is a blue header with the MSAEZ logo and a user profile icon. Below the header, a search bar and a 'NEW PROJECT' button are visible. The main content area is divided into sections for 'PUBLIC', 'MINE', 'SHARED', and 'LOCAL' projects. Each section contains a grid of project cards. Each card features a thumbnail image of an event storming diagram, a title, a user profile, and metadata such as 'Created Date' and 'LastModified Date'. A 'JOIN' button is present at the bottom of each card. The interface is designed to showcase and facilitate access to various event storming projects.

| Project Title | User | Created Date | LastModified Date | Action |
|---------------|----------------|----------------------|----------------------|-------------|
| 21년 2차수 1팀 | 용우 (na7149) | 2021년 6월 7일 23시 15분 | 2021년 6월 10일 11시 19분 | JOIN |
| bookkim | 정환 (kjh951203) | 2021년 5월 15일 11시 54분 | 2021년 6월 9일 22시 0분 | JOIN |
| Stock Trading | sanghoon01 | 2021년 6월 9일 14시 3분 | 2021년 6월 9일 15시 4분 | JOIN |
| Stock Trading | jyjang | 2021년 5월 28일 20시 18분 | 2021년 6월 8일 9시 8분 | DELETE EDIT |
| HiFive | sebastiaaa | 2021년 5월 23일 22시 51분 | 2021년 6월 8일 2시 34분 | JOIN |
| theater | neoshim | 2021년 3월 16일 14시 37분 | 2021년 6월 7일 21시 2분 | JOIN |

이론강의

Developing Cloud Native MSA Applications

3rd Jan 2020, ver2.0



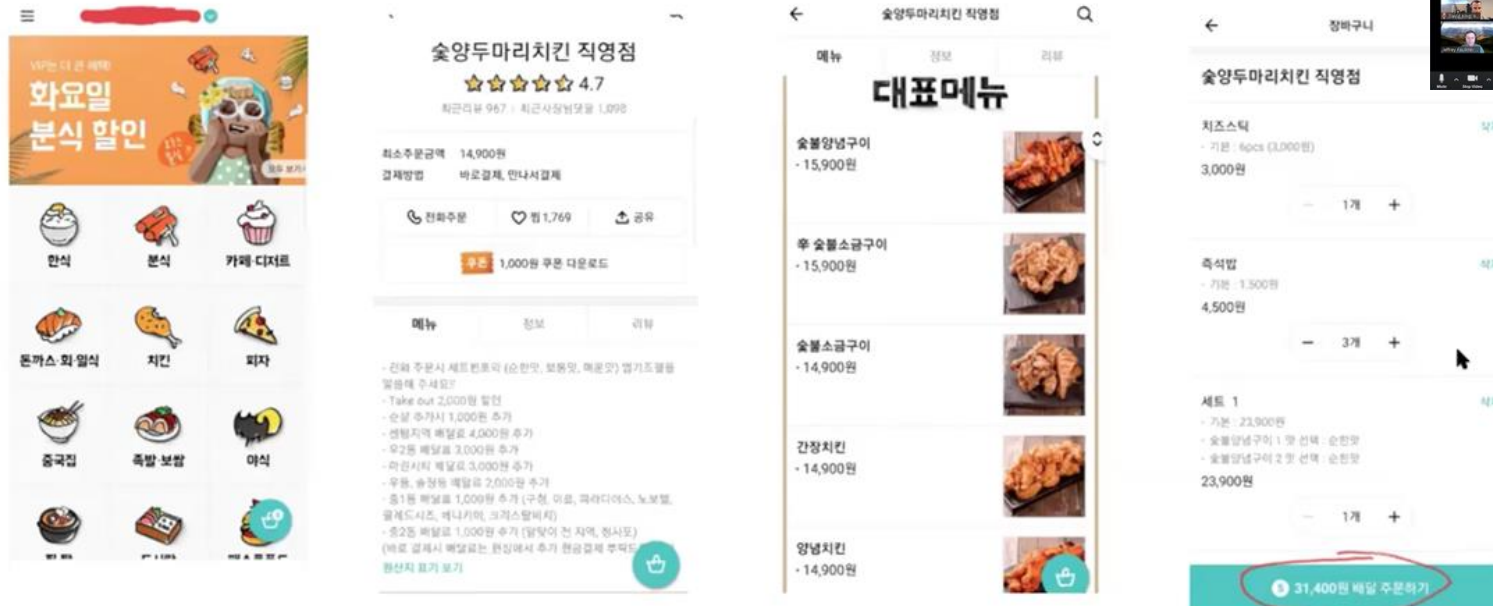
Copyright © 2019. Jinyoung Jang & uEngine-solutions All rights reserved.



예제 기반 강의

Context: Food Delivery App - Order

<https://github.com/msa-ez/example-food-delivery>



퀴즈 - 강의 집중도 유지

zoom.us 회의 보기 편집 창 도움말

EventStorming2Code x 클래스룸 x EventStorming2Code

주소: msaez.io/#/courses/cna/gkn/1/class-room

MSAEZ EventStorming2Code

ENGLISH

[이벤트스토밍] 도메인 이벤트의

Class Room Status

1조: Mercury (0%)

김형성 2/3 오영준 0/3 고동욱 0/3

5조: Jupiter (0%)

중화 1/3 박종화 1/3 이준근 1/3 성대 1/3 조남진 1/3

이론

Quiz

• MSA 구성을 하고자 하는 경우 허용 불가능한 연결선은?

2 번 3%

3 번 3%

4 번 67%

출석 23%

원프리미 3%

Slide 12

Google Slides

7월 8일 (수) 오후 1:52

단계1: 비즈니스 전략 실습

- Business Modeling Canvas

ENGLISH

00:00:00

HOUR
MIN
SEC

제출
클래스룸으로 이동
응답 하기

Instruction

아래 전략을 기반으로 한 비즈니스 모델을 비즈니스 모델 캔버스로 표현하십시오

- 고객의 이탈을 방지하기 위하여 지속적 사용자에게 우월권을 부여함
- 고객이 상품의 개선에 참여함
- 중간상을 없애 중간 비용을 최소화하여 고객에게 저렴한 상품을 직접적으로 제공함

Checkpoints

- 제시된 모든 비즈니스 전술을 담고 있는가 ☐
- 적절한 Perspective 에 전략/전술들을 배치하였는가 ☐

67

IMPLEMENTATION MODEL

SAVE

| | | | | |
|---|---|--|---|---|
| Key Partners << key partner >> Open Innovation | Key Activities << key activity >> Knowledge Management << key activity >> IT Integration << key activity >> Process Standardization << key activity >> Asset Standardization | Value Proposition << value proposition >> BPM Loyalty Programs << value proposition >> BPM OSS Guarantee << value proposition >> Online BPM Self-Service | Customer Relationships << customer relationship >> Co-Branding << customer relationship >> Community and Belonging | Customer Segments << customer segment >> SMB Company << customer segment >> Bank |
| Key Resources << key resource >> Customization << key resource >> Ease of Use << key resource >> Added Functionality | | Channels << channel >> Go Direct << channel >> Cross-selling << channel >> On-Demand | | |
| Cost Structure << cost structure >> SW Development Cost | | Revenue Streams << revenue stream >> Microtransactions - SaaS Model << revenue stream >> User-Defined << revenue stream >> Licensing - On-Premise << revenue stream >> Flexible Pricing | | |

실습도구 - 동료 결과 리뷰

Chrome 파일 수정 보기 방문 기록 북마크 사용자 탭 창 도움말

EventStorming2Code x 클래스룸 x EventStorming2Code x +

주의 요함 | msaez.io/#/courses/cna/gkn/1/class-room

MSAEZ EventStorming2Code

ENGLISH

김광섭 - engkwangseob@gmail.com -

Hash Name: labs-127672363

Checkpoints

- 3개 이상의 이벤트 (오렌지색 스티커)를 도출했는가 ☒
- 도메인에서 사용된 용어 (Ubiquitous Language)를 사용하... ☒
- 과거분사형으로 도출되었는가 ☐

랩실 보기 랩실 삭제

Product Name

Class Room Status

[이벤트스토밍] 도메인 이벤트의 도출

미션 보기 이룬 보기 응답 하기 랩실 가기 랩 목록 돌아가기 상금 종료

1조: Mercury (40%)

- 김광섭 2/3
- 오명준 0/3
- 윤현진 0/3
- 정승연 3/3
- 고동욱 3/3

2조: Venus (0%)

- 하민정 0/3
- 남창우 0/3
- 노재원 0/3
- 김진선 0/3
- 이수영 0/3

3조: Earth (60%)

- 최현 3/3
- 민우 3/3
- 계임 3/3
- 박원석 1/3
- 박윤철 0/3

4조: Mars (20%)

- 이선재 1/3
- 정현일 1/3
- 저성우 1/3
- 최안영 3/3
- 강희진 0/3

5조: Jupiter (20%)

- 종화 1/3
- 이준근 0/3
- 성다혜 3/3
- 기호진 1/3
- 조남진 0/3

6조: Saturn (40%)

- 정 3/3
- 범일 2/3
- 임국형 0/3
- 이선원 0/3
- 한동희 3/3

MSA 분석 - 2단계: 컨텍스트 매핑 실습

Chrome 파일 수정 보기 방문 기록 북마크 사용자 탭 창 도움말

EventStorming2Code x 클래스룸 x EventStorming2Code x 클래스룸

주의 요함 msaez.io/#/courses/cna/gkn/2/jyjang@uengine.org

MSAEZ EventStorming2Code

00:00:00
HOUR MIN SEC

재출 클래스룸으로 이동 응답 하기

Instruction

다음 시나리오에 대하여 Policy 를 부착하고 이름을 부여한 후, 해당하는 이벤트와 연결하시오.

1. 상점관리팀은 order 시스템의 주문이 발생할 때마다 (OrderPlaced) 주문 메시지를 점주에게 알린다 (policy 명: notify order).
2. 고객관리팀은 주문상태가 바뀔 때마다 (OrderPlaced, OrderCanceled) 카톡으로 알림을 보낸다 (policy 명: send KakaoTalk).

아래에 명시된 BC 명으로 Bounded Context 를 나누어 각각의 이벤트와 폴리스가 위치할 곳으로 옮겨 담시오

1. 주문: order
2. 상점관리: store
3. 고객관리: customer

Checkpoints

1. 2개의 폴리스 (라일락 스티커)를 도출했는가 ☒
2. 적절한 Event 와 연결되었는가? ☐
3. order(주문시스템)과 store(상점)의 이름으로 된 Bounded ... ☒
4. order(주문시스템)에서 OrderPlaced 가 발생되는가 ☒

Diagram illustrating the Event Storming process:

- order** (Bounded Context):
 - Events: OrderPlaced, OrderPaid, OrderCanceled
- store** (Bounded Context):
 - Events: StoreNotified, DeliveryStarted, DeliveryCanceled
 - Policy: notify order
- customer** (Bounded Context):
 - Policy: send KakaoTalk

Connections (Pub/Sub):

- order OrderPlaced → store StoreNotified
- order OrderCanceled → store DeliveryCanceled
- store StoreNotified → customer send KakaoTalk
- store DeliveryStarted → customer send KakaoTalk
- store DeliveryCanceled → customer send KakaoTalk

Project Name: _____

CODE OPEN SAVE

ENGLISH

클래스룸 - 동료 결과 공유

The screenshot displays the EventStorming2Code Classroom interface. The browser address bar shows the URL `msaez.io/#/courses/cna/gkn/2/class-room`. The interface includes a sidebar on the left with a user profile for 장현일 (hyunil@braincolla.com) and a checklist of five checkpoints. The main area is titled "[이벤트스토밍] 폴리스 도출과 바운디드 컨텍스트의 분리" and shows the status of six groups, each represented by a planet and a progress bar.

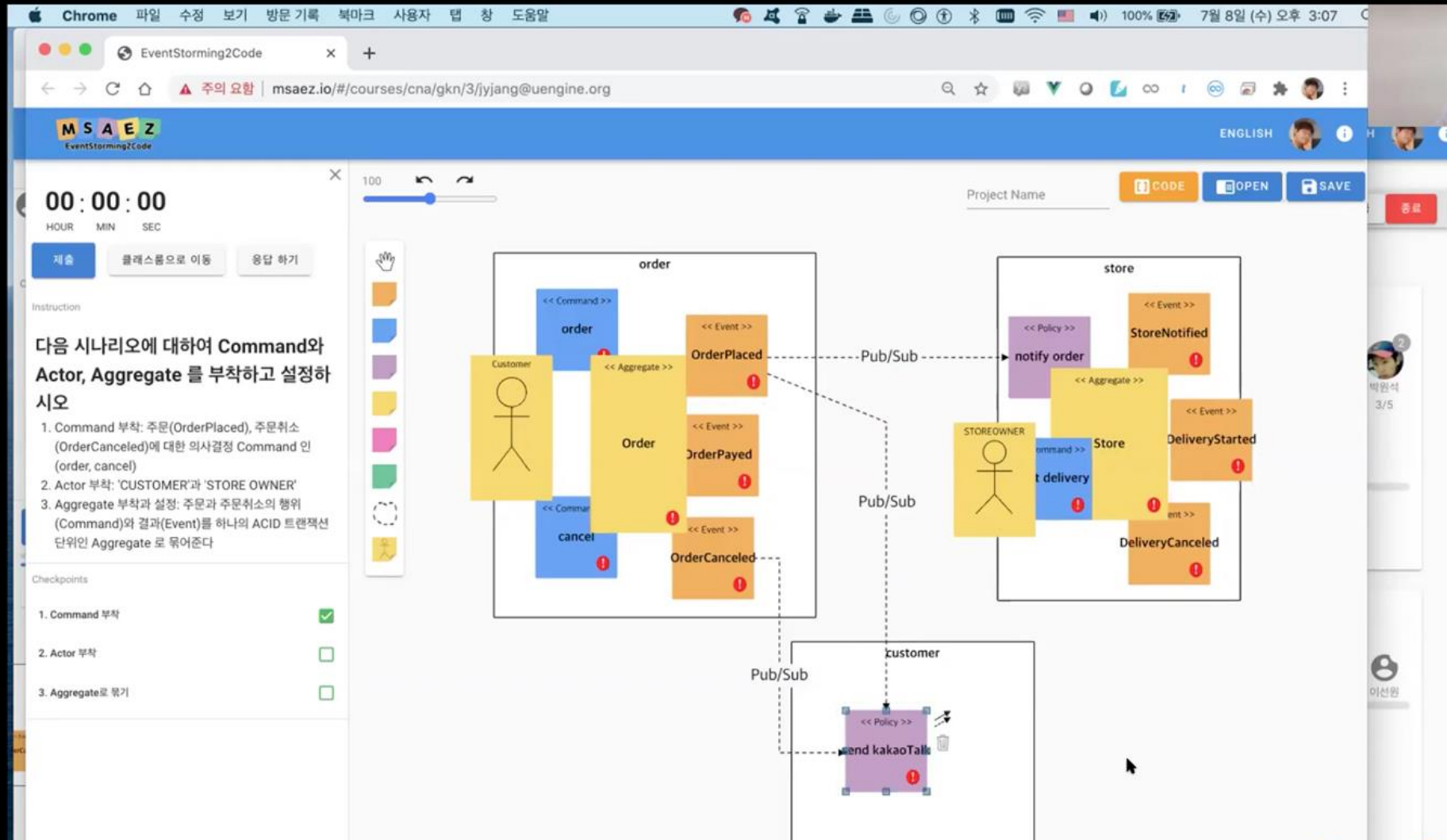
Checkpoints:

1. 2개의 폴리스 (라일락 스타커)를 도출했는가 ☒
2. 적절한 Event 와 연결되었는가? ☐
3. order(주문시스템)과 store(상점)의 이름으로 된 Bounded ... ☒
4. order(주문시스템)에서 OrderPlaced 가 발행되는가 ☒
5. store(상점서비스)에서 'notify order' 가 수신되는가 ☒

Class Room Status:

- 1조: Mercury (0%)**
 - 김광섭 4/5
 - 오명준 0/5
 - 윤현진 0/5
 - 정승연 0/5
 - 고동욱 1/5
- 2조: Venus (0%)**
 - 하민정 1/5
 - 남창우 2/5
 - 노태원 3/5
 - 김진선 1/5
 - 이수영 1/5
- 3조: Earth (0%)**
 - 최현 1/5
 - 민우 2/5
 - 게임 3/5
 - 박원석 1/5
 - 백운철 1/5
- 4조: Mars (0%)**
 - 이찬재 4/5
 - 장현일 4/5
 - 차성우 4/5
 - 최인영 4/5
 - 강희진 4/5
- 5조: Jupiter (0%)**
 - 종화 1/5
 - 이준근 4/5
 - 성다혜 4/5
 - 기호진 4/5
 - 조남진 4/5
- 6조: Saturn (0%)**
 - 징 4/5
 - 병일 1/5
 - 임국형 4/5
 - 이선원 4/5
 - 한동희 4/5

MSA 분석 - 3단계 (Actor, Command)



단계별 실습

Chrome 파일 수정 보기 방문 기록 북마크 사용자 탭 창 도움말

EventStorming2Code x 라운지 x EventStorming2Code x +

msaez.io/#/courses/cna/gkn

MSAEZ EventStorming2Code

ENGLISH

Command와 Actor, Aggregate를 부착하고 설정하시오
10분 이내 완료

입장

4 [이벤트스토밍-구현] 코드의 생성
...
10분 이내 완료

입장

5 완료

[구현] 마이크로서비스의 실행
생성된 Cloud Native Application 서비스를 실행하시오.
10분 이내 완료

입장

6 [구현] Correlation 폴리시의 구현
주문 서비스의 도메인 이벤트에 따른 상점서비스의 비즈니스를 구현하시오.
10분 이내 완료

입장

7 [구현] 폴리시(이벤트리스너)의 구현

MSA 구현 - 스프링부트 개발도구

(클라우드 IDE - Visual Studio 와 같은 UI)

Chrome 파일 수정 보기 방문 기록 북마크 사용자 탭 창 도움말

EventStorming2Code x 려운지 x EventStorming2Code x +

← → ↺ ⌂ ⓘ 주의 요함 | msaez.io/#/courses/cna/gkn/5/jinyoungj@gmail.com

MSAEZ EventStorming2Code

00:00:00
HOUR MIN SEC

제출 클래스룸으로 이동 응답 하기

Instruction

생성된 마이크로 서비스들의 기동

- order, store, customer 마이크로 서비스를 각각 8081, 8082, 8083 포트로 실행한다.
- order 서비스의 Application.java 파일로 이동한다.
- 14행과 15행 사이의 'Run'을 클릭 후, 5초 정도 지나면 서비스가 터미널 창에서 실행된다.
- 새로운 터미널 창에서 netstat -lntp 명령어로 실행중인 서비스 포트를 확인한다.
- store, customer 서비스에 대해서도 동일한 작업을 수행한다.
- 기동된 order 서비스를 호출하여 주문 1건을 요청한다.
http localhost:8081/orders productId=1001 storeId=101

Checkpoints

1. 포트별 마이크로서비스 실행(8081) ☐
2. 포트별 마이크로서비스 실행(8082) ☐
3. 포트별 마이크로서비스 실행(8083) ☐
4. 주문 처리가 됨 ☐

File Edit Selection View Go Labs Debug Terminal Help

EXPLORER

- PROJECT
 - fooddelivery_lab
 - customer
 - .settings
 - kubernetes
 - src
 - target
 - classpath
 - project
 - azure-pipelines.yml
 - cloudbuild.yml
 - customer.iml
 - Dockerfile
 - pom.xml
 - gateway
 - order
 - store

JAVA DEPENDENCIES

- boot-camp-gateway
- customer
- order
- store

Application.java x

```
1 package fooddeliveryrevival;
2 import fooddeliveryrevival.config.kafka.KafkaProcessor;
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.context.ApplicationContext;
6 import org.springframework.cloud.stream.annotation.EnableBinding;
7 import org.springframework.cloud.openfeign.EnableFeignClients;
8
9
10 @SpringBootApplication
11 @EnableBinding(KafkaProcessor.class)
12 @EnableFeignClients
13 public class Application {
14     protected static ApplicationContext applicationContext;
15
16     @Run|Debug
17     public static void main(String[] args) {
18         applicationContext = SpringApplication.run(Application.class, args);
19     }
20 }
```

Do you want to exclude the Theia Java Example Java project settings files (.classpath, .project, .settings, .factorypath) from the file explorer?

MSA 구현 - Lab 결과 제출

The screenshot displays a web browser window with the URL `msaez.io/#/courses/cna/gkn/5/jinyoungj@gmail.com`. A notification box in the center reads "msaez.io 내용: 제출하였습니다" (msaez.io content: Submitted) with a "확인" (Check) button. The browser's address bar shows "주의 요함" (Caution) and "msaez.io/#/courses/cna/gkn/5/jinyoungj@gmail.com". The browser's top bar includes "Chrome", "파일 수정 보기 방문 기록 북마크 사용자 탭 창 도움말", and "100% 7월 8일 (수) 오후 3:51".

Below the browser window, an IDE (Visual Studio Code) is open, showing a project structure on the left and code in the main editor. The project structure includes:

- PROJECT
 - fooddelivery_lab
 - customer
 - .settings
 - kubernetes
 - src
 - main
 - java
 - fooddeliveryrevival
 - config
 - AbstractEvent.java
 - Application.java
 - OrderCanceled.java
 - PolicyHandler.java
 - resources
 - target
 - .classpath
 - .project
 - azure-pipelines.yml
 - cloudbuild.yml

The main editor shows the code for `Application.java` in the `customer` directory. The code includes annotations for Spring Boot, Kafka, and Feign, and a `public class Application` definition. The code is as follows:

```
8
9
10 @SpringBootApplication
11 @EnableBinding(KafkaProcessor.class)
12 @EnableFeignClients
13 public class Application {
```

The IDE also shows a Java Process Console with the following output:

```
root@labs--1029590031:/home/project# http localhost:8081/orders productId=1001 storeId=101
HTTP/1.1 201
Content-type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2020 06:49:41 GMT
Location: http://localhost:8081/orders/5
Transfer-Encoding: chunked

{
  "_links": {
    "order": {
      "href": "http://localhost:8081/orders/5"
    },
    "self": {
      "href": "http://localhost:8081/orders/5"
    }
  },
  "productId": "1001",
  "status": null,
  "storeId": "101"
}
```

MSA 구현 - 자동 채점

MSA E Z
EventStorming2Code

00:00:00
HOUR MIN SEC

재출 클러스터로 이동 응답 하기

Instruction

생성된 마이크로 서비스들의 기동

- order, store, customer 마이크로 서비스를 각각 8081, 8082, 8083 포트로 실행한다.
- order 서비스의 Application.java 파일로 이동한다.
- 14행과 15행 사이의 'Run'을 클릭 후, 5초 정도 지나면 서비스가 터미널 창에서 실행된다.
- 새로운 터미널 창에서 netstat -lntp 명령어로 실행중인 서비스 포트를 확인한다.
- store, customer 서비스에 대해서도 동일한 작업을 수행한다.
- 기동된 order 서비스를 호출하여 주문 1건을 요청한다.
http localhost:8081/orders productId=1001 storeId=101

Checkpoints

1. 포트별 마이크로서비스 실행(8081) ☒
2. 포트별 마이크로서비스 실행(8082) ☒
3. 포트별 마이크로서비스 실행(8083) ☒
4. 주문 처리가 됨 ☒

File Edit Selection View Go Labs Debug Terminal Help

EXPLORER

- PROJECT
- fooddelivery_lab
- customer
- settings
- kubernetes
- src
- main
- java
- fooddeliveryrevival
- config
- AbstractEvent.java
- Application.java
- OrderCanceled.java
- PolicyHandler.java
- resources
- target
- classpath
- project
- azure-pipelines.yml
- cloudbuild.yml
- customer.yml
- Dockerfile
- pom.xml
- gateway
- order
- store

JAVA DEPENDENCIES

- boot-camp-gateway
- customer
- order
- store

Application.java

```
12 @EnableFeignClients
13 public class Application {
14     protected static ApplicationContext applicationContext;
15     @Run|Debug
16     public static void main(String[] args) {
17         applicationContext = SpringApplication.run(Application.class, args);
18     }
19 }
```

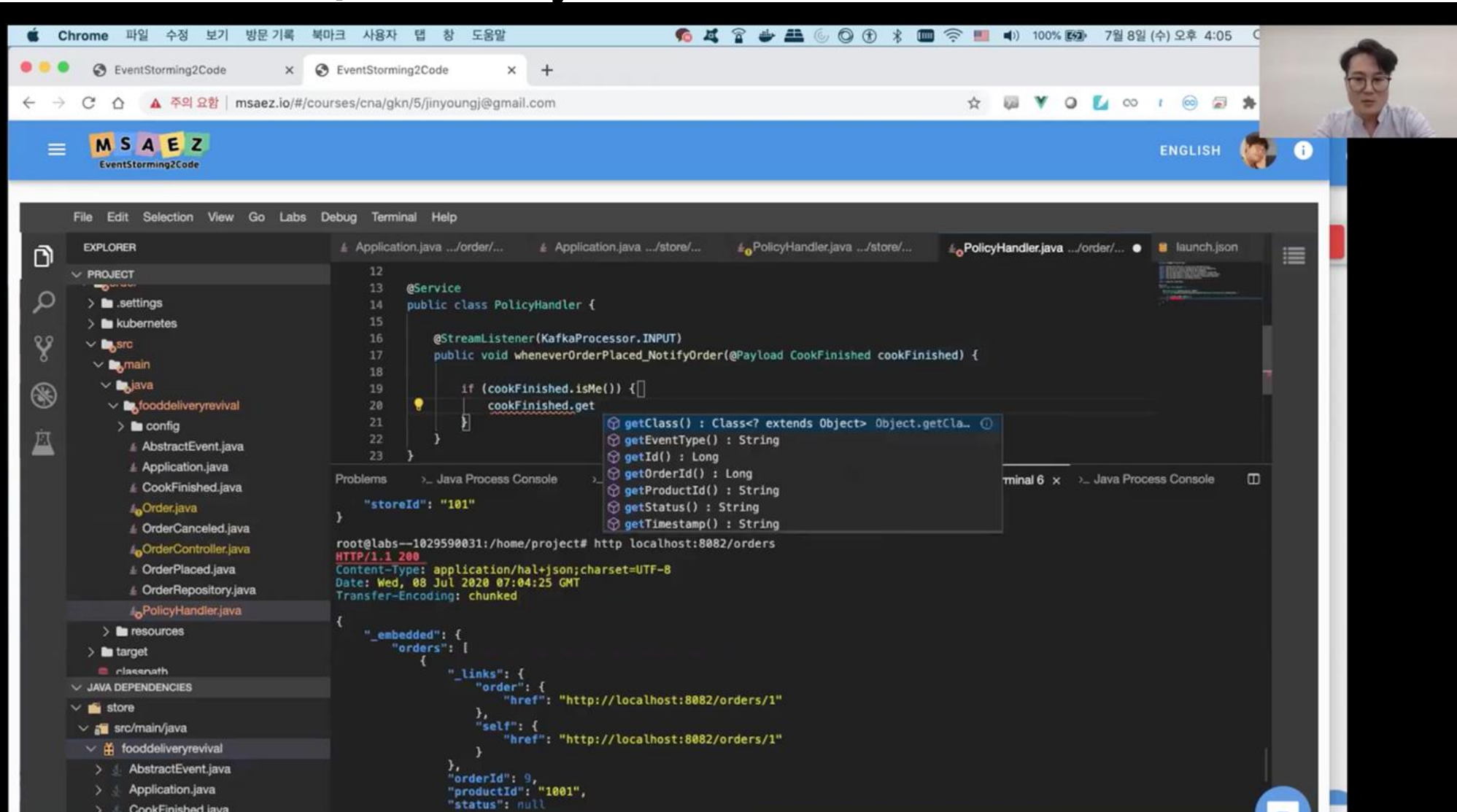
test 8083 : HTTP/1.1 200
HTTP/1.1 200
Content-Type: application/hal+json;charset=UTF-8
Date: Wed, 08 Jul 2020 06:40:37 GMT
Transfer-Encoding: chunked

```
{
  "_links": {
    "profile": {
      "href": "http://localhost:8083/profile"
    }
  }
}
```

test 8081 to save order item
HTTP/1.1 201
Content-Type: application/json;charset=UTF-8
Date: Wed, 08 Jul 2020 06:40:37 GMT
Location: http://localhost:8081/orders/4
Transfer-Encoding: chunked

```
{
  "_links": {
    "order": {
      "href": "http://localhost:8081/orders/4"
    },
    "self": {
      "href": "http://localhost:8081/orders/4"
    }
  },
  "productId": "1001",
  "status": null,
  "storeId": "101"
}
```

MSA 구현 - 코드 어시스트 (VS Code 와 동일)

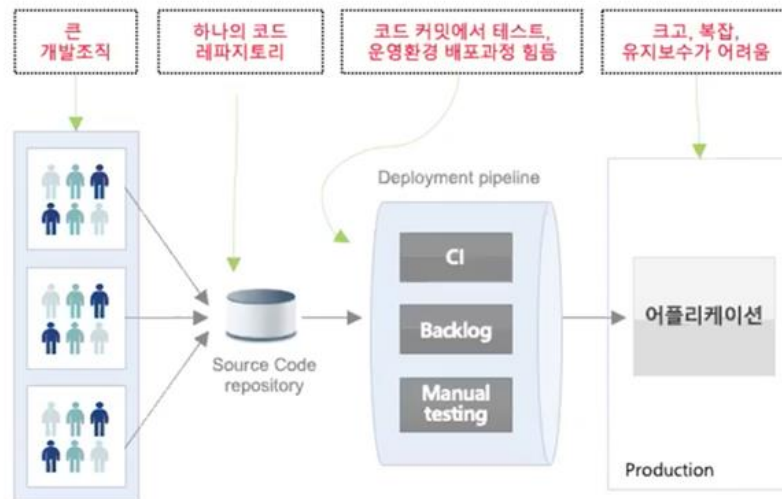


MSA 운영 - 이론

Process Change : 열차말고 택시를 타라!

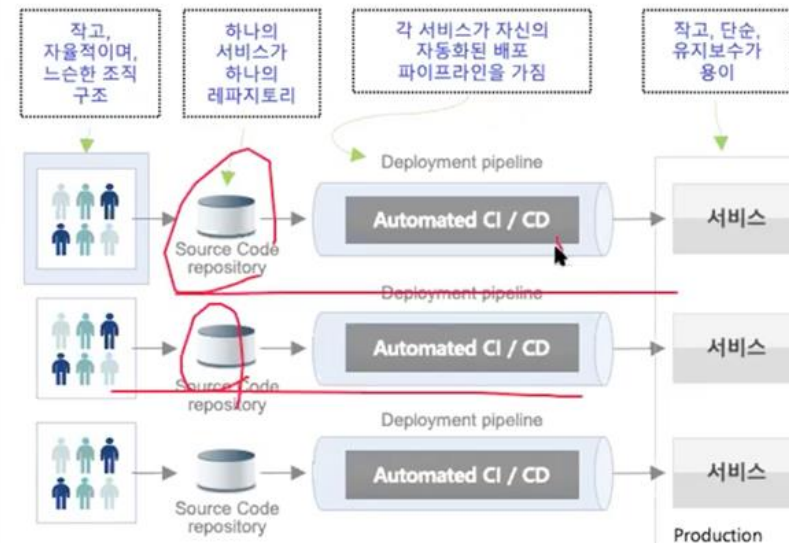
모노리식 개발 및 운영환경

- 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 파이프라인을 적용하기가 어렵고 Delivery 시간이 과다 소요



마이크로서비스 개발 및 운영환경

- 작고 분화된 조직에서 서비스를 작은 크기로 나누어 개발하므로 해당 비즈니스 로직에만 집중하게 되어 개발 생산성 향상
- 연관된 마이크로서비스만 테스트를 수행하므로 개발/테스트/배포 일정이 대폭 축소



실습 - 쿠버네티스 오브젝트 설계

The screenshot shows a web browser window with the URL `msaez.io/#/courses/cna/gkn/11/jyjang@uengine.org`. The page is titled "MSAEZ EventStorming2Code" and features a timer at "00:00:00". A sidebar on the left contains icons for various Kubernetes resources. The main content area is divided into three sections:

- Instruction**:
 - 쿠버네티스 디플로이 설계**
 1. 각 서비스별 (BC별) Deployment 를 작성한다 (order, store, customer)
 2. 작성한 Deployment 로 연결될 Service 들을 작성
 3. Ingress 를 이용하여 진입점을 통일한다. Path-based Routing 로 각각의 마이크로 서비스들이 연결되도록 설정한다. 각 path 는 '/orders', '/orders', '/customers' 임
 - 확장주제**
 1. 데이터베이스로 Mongo DB 를 이용하기 위해 mongo 용 Deployment 를 만든다.
 2. Service 로 'mongo' 를 연결한다.
 3. Persistence Volume 와 PersistenceVolume 을 선언하여 mongo 에 mount 시킨다.
- Checkpoints**:
 - 1. 각 서비스별 (BC별) Deployment 를 작성 ☐
 - 2. 각 Deployment 들로 Service 가 연결됨 ☐
 - 3. Ingress 를 이용하여 각 Service 들이 모두 연결됨 ☐
 - 4. 모든 Deployment 의 container port 는 8080임 ☐
- Deployment**:

```
1 ---
2 apiVersion: "apps/v1"
3 kind: "Deployment"
4 metadata:
5   name: "store"
6   labels:
7     app: "store"
8   spec:
9     selector:
10       matchLabels:
11         app: "store"
12       replicas: 4
13     template:
14       metadata:
15         labels:
16           app: "store"
17       spec:
18         containers:
19           -
20             name: "store"
21             image: "jyjang/store"
22             ports:
23               -
24                 containerPort: 80
```

Configuration details on the right:

 - Name: store
 - Image: jyjang/store
 - Replicas: 4
 - Target Port: 80

실습 - 쿠버네티스 오브젝트 설계

MSAEZ
EventStorming2Code

00:00:00
HOUR MIN SEC

재출 클래스룸으로 이동 응답 하기

Instruction

쿠버네티스 디플로이 설계

1. 각 서비스별 (BC별) Deployment 를 작성한다 (order, store, customer)
2. 작성한 Deployment 로 연결될 Service 들을 작성
3. Ingress 를 이용하여 진입점을 통일한다. Path-based Routing 로 각각의 마이크로 서비스들이 연결되도록 설정한다. 각 path 는 '/orders', '/orders', '/customers' 임

확장주제

1. 데이터베이스로 Mongo DB 를 이용하기 위해 mongo 용 Deployment 를 만든다.
2. Service 로 'mongo' 를 연결한다.
3. Persistence Volume 와 PersistenceVolume 을 선언하여 mongo 에 mount 시킨다.

Checkpoints

1. 각 서비스별 (BC별) Deployment 를 작성

2. 각 Deployment 들로 Service 가 연결됨

3. Ingress 를 이용하여 각 Service 들이 모두 연결됨

4. 모든 Deployment 의 container port 는 8080임

5. 모든 Service 의 target port 는 8080 이고 port 도 8080 임

100

Service
order

Deployment
order

Service
store

Deployment
store

Service
customer

Deployment
customer

DEPLOY

CODE

실습 - 쿠버네티스 디플로이 다이어그래밍

Chrome 파일 수정 보기 방문 기록 북마크 사용자 탭 창 도움말

클래스룸 EventStorming2Code

주의 요함 msaez.io/#/courses/cna/gkn/11/jyjang@uengine.org

MSAEZ EventStorming2Code

00:00:00
HOUR MIN SEC

100

DEPLOY CODE

제출 클래스룸으로 이동 응답 하기

Instruction

쿠버네티스 디플로이 설계

1. 각 서비스별 (BC별) Deployment 를 작성한다 (order, store, customer)
2. 작성한 Deployment 로 연결될 Service 들을 작성
3. Ingress 를 이용하여 진입점을 통일한다. Path-based Routing 로 각각의 마이크로 서비스들이 연결되도록 설정한다. 각 path 는 '/orders', '/orders', '/customers' 임

확장주제

1. 데이터베이스로 Mongo DB 를 이용하기 위해 mongo 용 Deployment 를 만든다.
2. Service 로 'mongo' 를 연결한다.
3. Persistence Volume 와 PersistenceVolume 을 선언 하여 mongo 에 mount 시킨다.

Checkpoints

1. 각 서비스별 (BC별) Deployment 를 작성 ☒
2. 각 Deployment 들로 Service 가 연결됨 ☒
3. Ingress 를 이용하여 각 Service 들이 모두 연결됨 ☒

```
graph TD; Ingress[Ingress  
my-ingress] --> ServiceOrder[Service  
order]; Ingress --> ServiceStore[Service  
store]; Ingress --> ServiceCustomer[Service  
customer]; ServiceOrder --> DeploymentOrder[Deployment  
order]; ServiceStore --> DeploymentStore[Deployment  
store]; ServiceCustomer --> DeploymentCustomer[Deployment  
customer];
```

MSA 운영 - 쿠버네티스 CLI 실습

- Web-based Shell

The screenshot displays a web-based Kubernetes lab interface. The top navigation bar includes the 'MSAEZ EventStorming2Code' logo, a language selector set to 'ENGLISH', and a user profile icon. The browser's address bar shows the URL 'localhost:8081/#/courses/analysis/rose-systems/12/jyjang@uengine.org'.

Instructions:

쿠버네티스 배포

1. 앞서 생성한 도커이미지 명을 기반으로 쿠버네티스에서 서비스를 배포한다. `kubectl create -f kubernetes/deployment.yml`
2. Kafka가 없으면 기동시 오류가 나는 것을 확인한다.
3. 쿠버네티스 내에 kafka 서버를 Helm 을 통해서 배포한다.
4. 좀있다 기존 deployment 가 제대로 동작함을 `kubectl logs` 를 통해서 확인한다.

Checkpoints:

- 1. Deployment 3개 배포 여부 ☒
- 2. Service 3개 배포 여부 ☒
- 3. Order 서비스 접속 성공 ☐

Terminal Output:

```
##### test Deployments : order, store, customer
++ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
customer  1/1     1             1           14s
order     1/1     1             1           14s
store     1/1     1             1           14s
++ set +x

##### test Services : order, store, customer
++ kubectl get services
NAME      TYPE       CLUSTER-IP   EXTERNAL-IP   PORT(S)    AGE
customer  ClusterIP  10.0.14.233   <none>         8080/TCP    14h
order     ClusterIP  10.0.1.14    <none>         8080/TCP    14h
store     ClusterIP  10.0.5.110   <none>         8080/TCP    14h
++ set +x

##### test OrderServiceIsOn : HTTP/1.1 201
```