UEngine

성공적 디지털 서비스 전환을 위한

MSA-Easy

회사개요

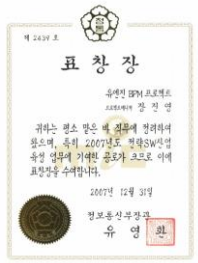
uEngine

유엔진솔루션즈는 10년간의 BPM, SNS, Cloud 솔루션을 개발해오면서 쌓아온 기술력과 제조/금융 분야의 적용 경험으로,
본 서비스를 성공적으로 구축할 수 있는 기술 전문 기업입니다.

업력	• 2007년 03월 20일 설립
주요 고객사	• 한화그룹, 기업은행, 삼성전기, SKT, SKC&C, 현대기아차, 대우건설, 경동나비엔, 생산기술원 등
주요사업분야 및 보유 기술	• 클라우드 컴퓨팅 / SOA 컨설팅 및 시스템, SaaS 지원 시스템 ETRI 공동연구 (2010) • 세계적 오픈소스 커뮤니티(uEnigne.org, Open Cloud Engine)운영 • BPM (업무프로세스관리), BRE(비즈니스룰엔진) • 기업용 소셜 네트워크 기반 협업 시스템 • ALM(어플리케이션 수명주기관리)

인증 및 수상

2012년 SW산업발전
〈국무총리 표창〉



2007년도 전략 소프트웨어
〈산업 육성 공로상〉

sourceforge

2004 Ranked 60th in top Source forge Open Source Project

internet.com

2009 Selected as top 101 Enterprise OSS by internet.com.

보유기술과 서비스

OPEN CLOUD ENGINE

SAMSUNG 삼성전기

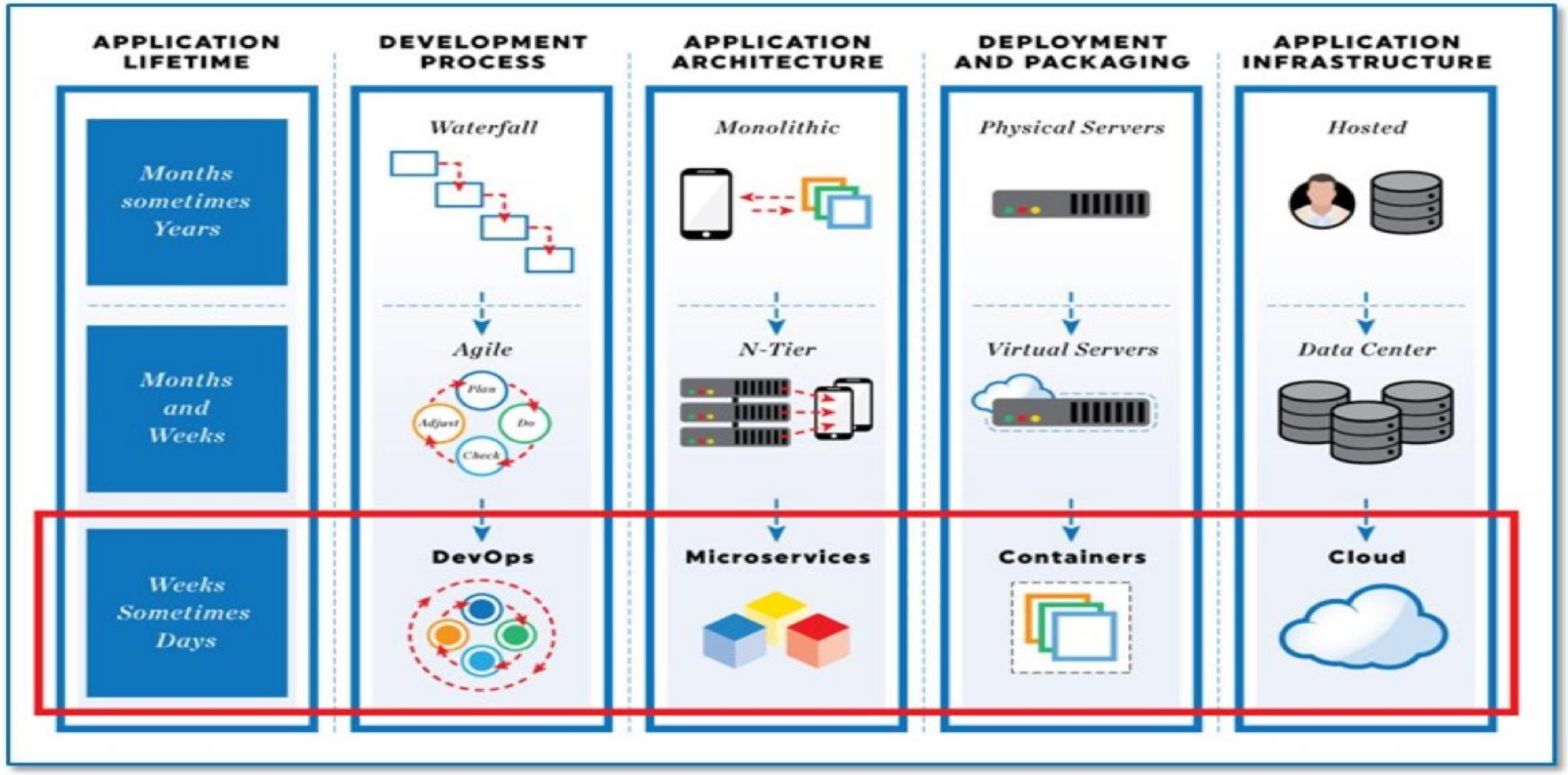
Hanwha

PROCESS COD I

UEngine BPM

LG디스플레이

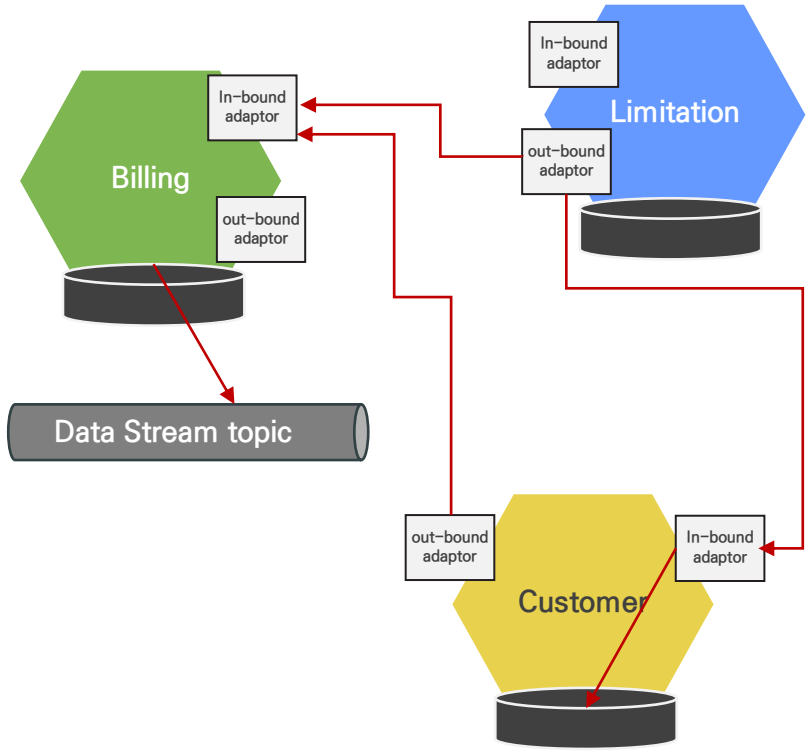
DT 전환에 필요한 것들


APPLICATION LIFETIME
DEVELOPMENT PROCESS
APPLICATION ARCHITECTURE
DEPLOYMENT AND PACKAGING
APPLICATION INFRASTRUCTURE
Months sometimes Years
Months and Weeks
Weeks Sometimes Days
Waterfall
Agile
Plan
Do
Check
Adjust
DevOps
Monolithic
N-Tier
Microservices
Physical Servers
Virtual Servers
Containers
Hosted
Data Center
Cloud

Microservice Architecture

- Updating one service doesn't require changing others

- Ability to upgrade the tech stack (HW/SW/DBMS/NW) Of each service independently



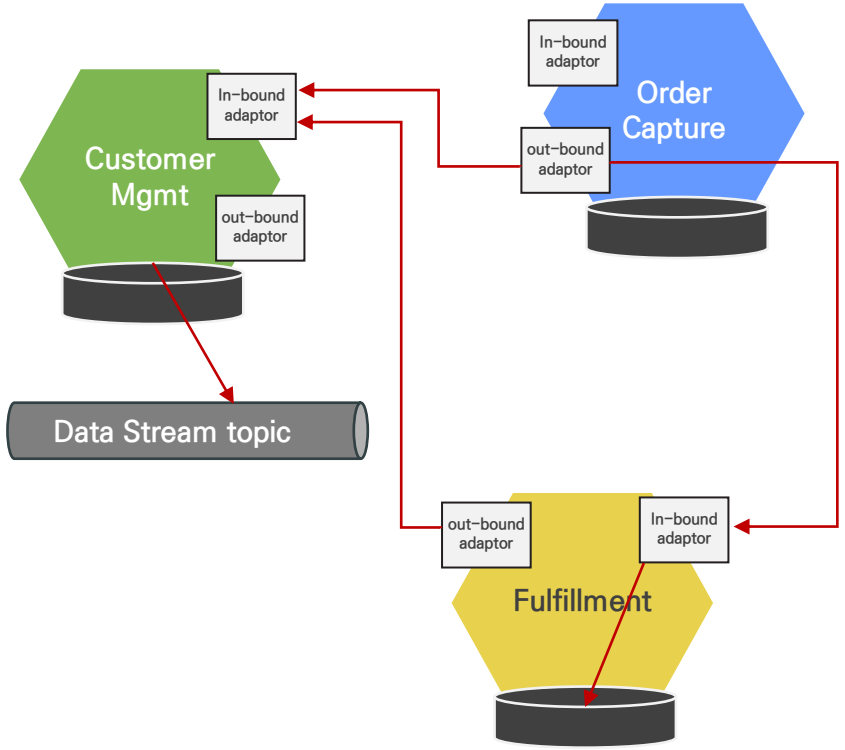
- Good fault isolation

- Smaller, simpler code base

- Options for scaling

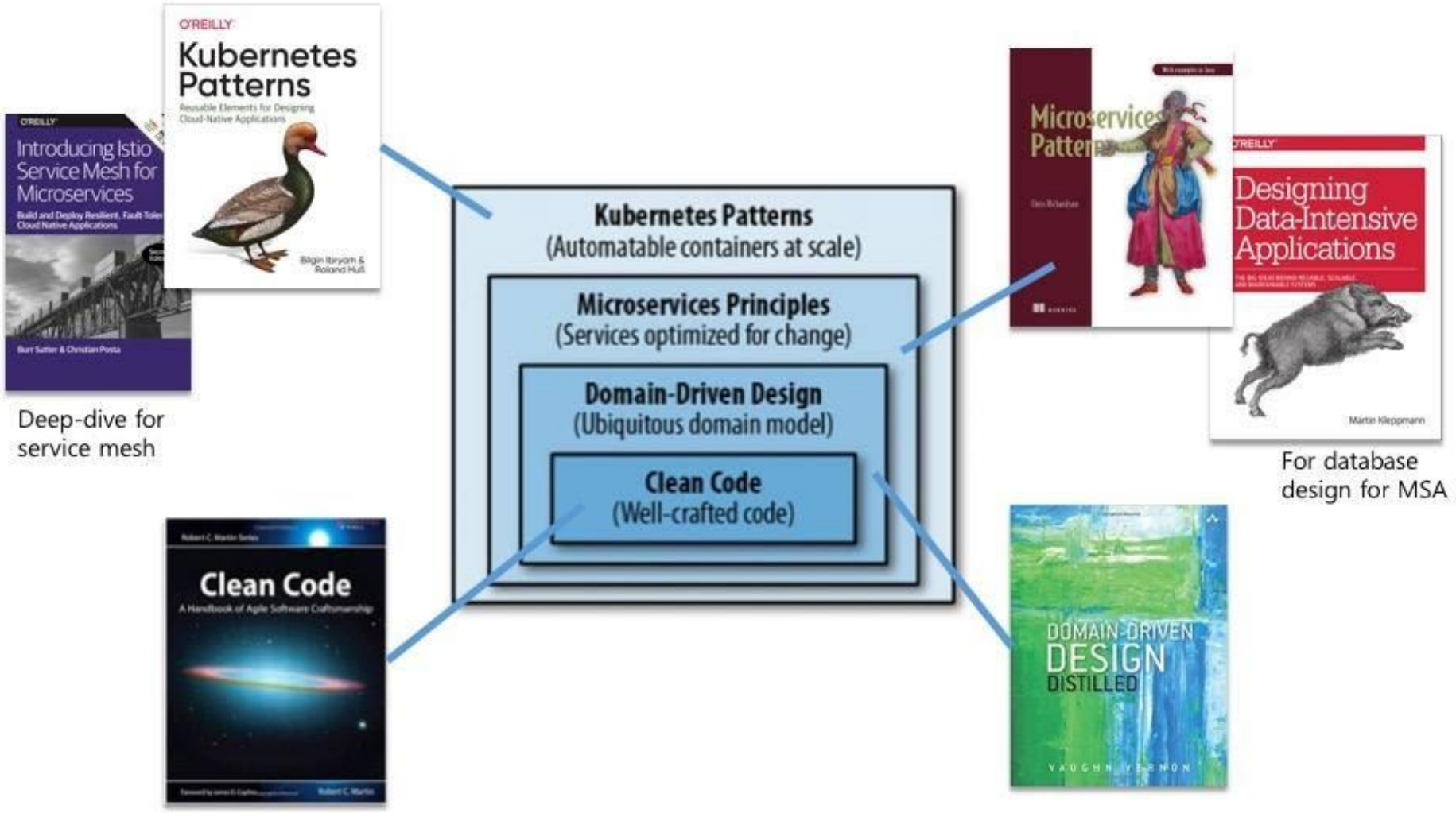
Learning Curves to a Microservice Architecture

Distributed computing adds complexity and slow down initial development



- 개발도구들이 아직 최적화되지 않았다
  Dev tools not optimized for distributed services
- 전체적인 테스트가 복잡해진다
  Testing can be more complicated
- 운영과 디플로이가 복잡해져 쿠버네티스 와 같은 DevOps 환경이 필수적이다
  Deployment and operations are more complex
- 서비스를 어떻게 쪼갤 것인가?
  Where/How to decompose the services?
- 서비스간 연동을 통해서만 구현하는 비용의 상승
  Inter-service communication complicates development
- 분산트랜잭션 (데이터 일관성 등)을 어떻게 보장할 것인가?
  Maintaining consistency with distributed transactions is hard
- 서비스 개수가 많아진 상황의 보안처리를 어떻게 할 것인가?
  What about inter-service security? Identity management?

Learning Path to Cloud-Native



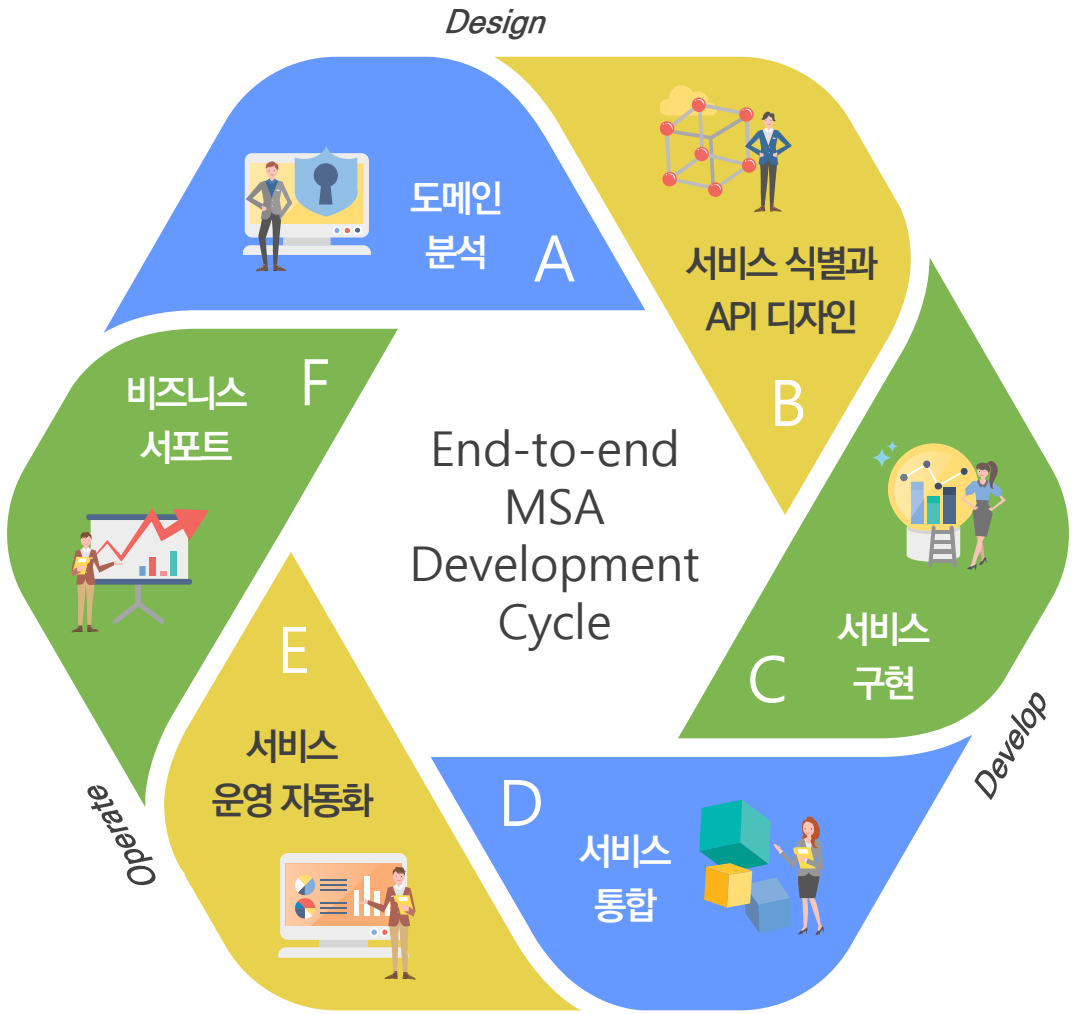
MSA-Easy

UEngine

MSA-Easy 는 DDD 기반 아키텍처
분석에서
운영자동화까지 한번에 지원하는
MSA 운용 플랫폼입니다

http://msaez.io

http://github.com/msa-ez



MSA Easy Value Chain & Tools

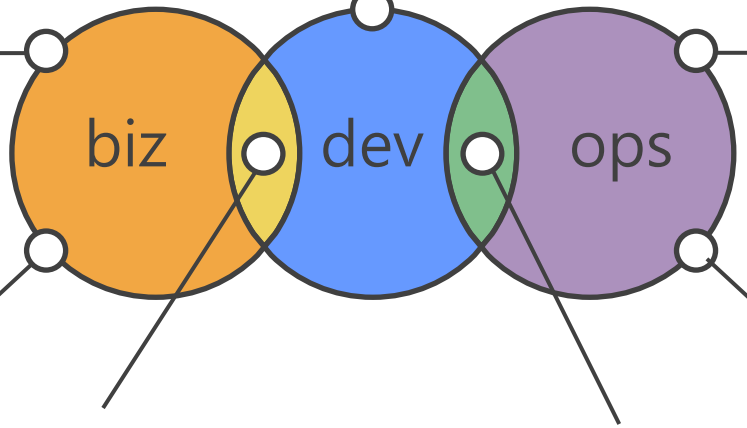
UEngine

Coding / Testing

⑨ Polyglot Clean Code Gen
⑩ Ubiquitous Language Modeling
⑪ Implementation (Cloud IDE)
⑫ UAT Test Gen.

Opportunity Discovery

① Customer Journey Mapping
② Impact Mapping
③ Business Model Canvas



Value Analysis

⑰ SLA/ Speed Monitoring
⑱ User Feedback A/B Testing

Requirement Capturing

④ Big Picture Event Storming (E/S)
⑤ Process Modeling E/S
⑥ Example Mapping
⑦ User Story Mapping

Architecting

⑦ Design Level Event Storming
⑧ Hexagonal Modeling

Deployment

⑬ Deployment Configuration Gen.
⑭ Contract Test Gen.
⑮ Kubernetes Modeling

Delivery

⑯ Kubernetes Operation & CI/CD Integration

성공적 디지털 서비스 전환을 위한 MSA-Easy

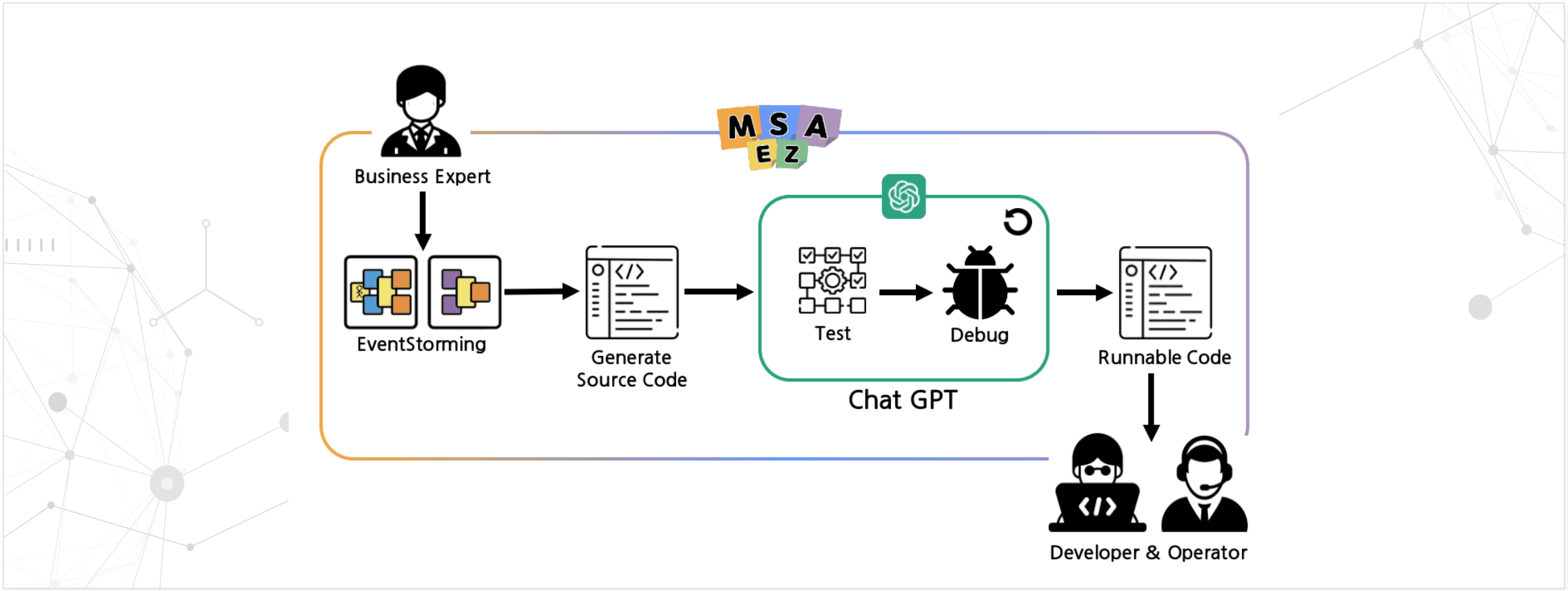
AI 기반 MSA 설계/구현/운영

UEngine

MSA Ez는 LLM을 이용하여 다음을 자동화합니다

- 사용자가 비즈니스 모델을 자연어로 서술하면
- 사용자 저니맵, 비즈니스 모델 캔버스 등의 비즈니스 모델 관련 기획 초안 생성
- 이벤트스토밍 모델 자동생성
- 사용자가 원하는 프로그래밍 언어와 플랫폼에 대한 구현 소스코드 생성

- 구현 소스코드의 컴파일과 반복 자동화된 버그 수정
- 배포 자동화
- K8S에서 발생한 오류 추적 자동 배포 스크립트 수정*

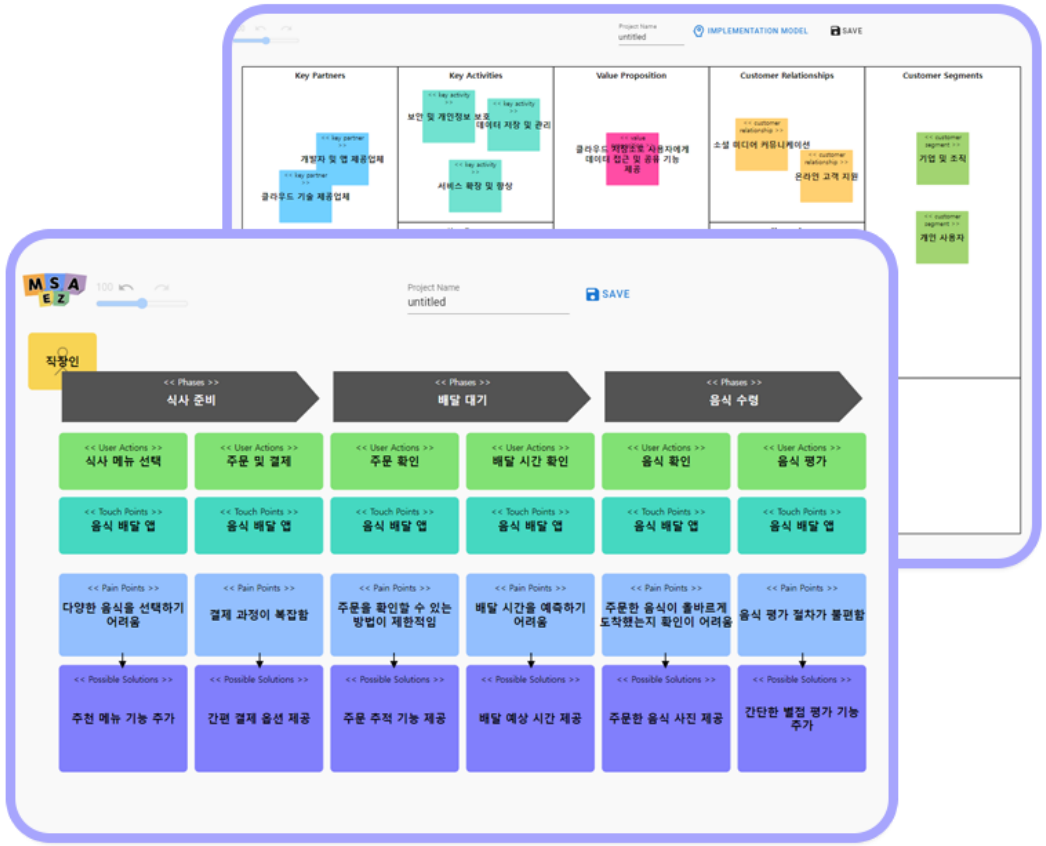


01

Biz Phase

UEngine

Biz - Customer Journey Map



Identifies

- Personas
- Pain-points for each actions and touchpoints by polling the emotions

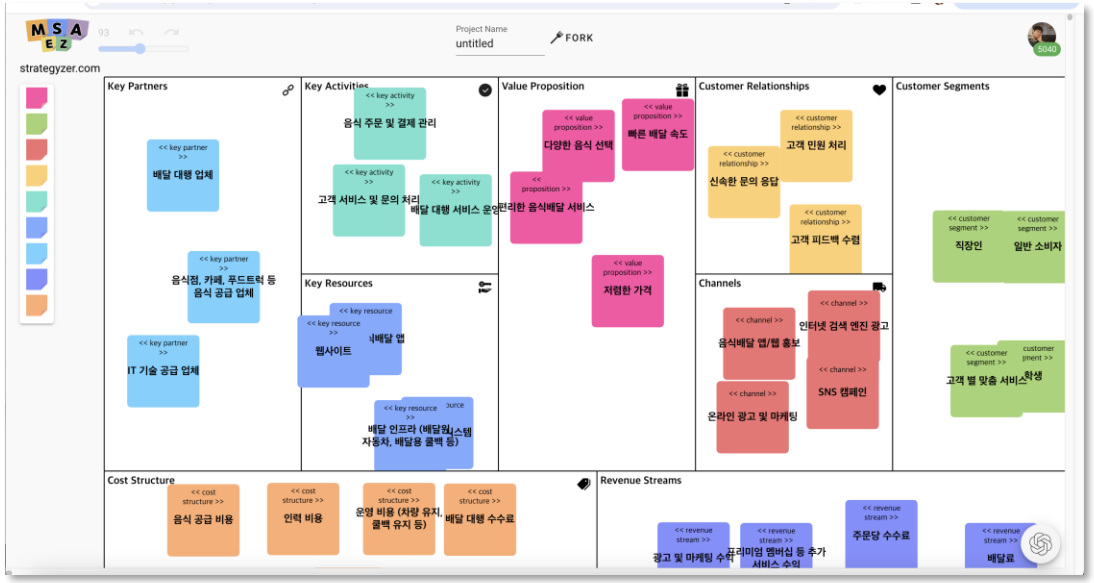
Derives

- Possible Solutions
- Opportunities

Related to

- Business Model (Value Propositions, Customer Segments, Activities)
- Software Design (Events, Commands, Read Models, Aggregates)

Biz - Business Model Canvas



Identifies

- Value Propositions
- Customer Segments
- Customer Relationships / Channels
- Key Activities / Key Resources
- Key Partners
- Cost Structure and Revenue Streams

Derives

- Business Model

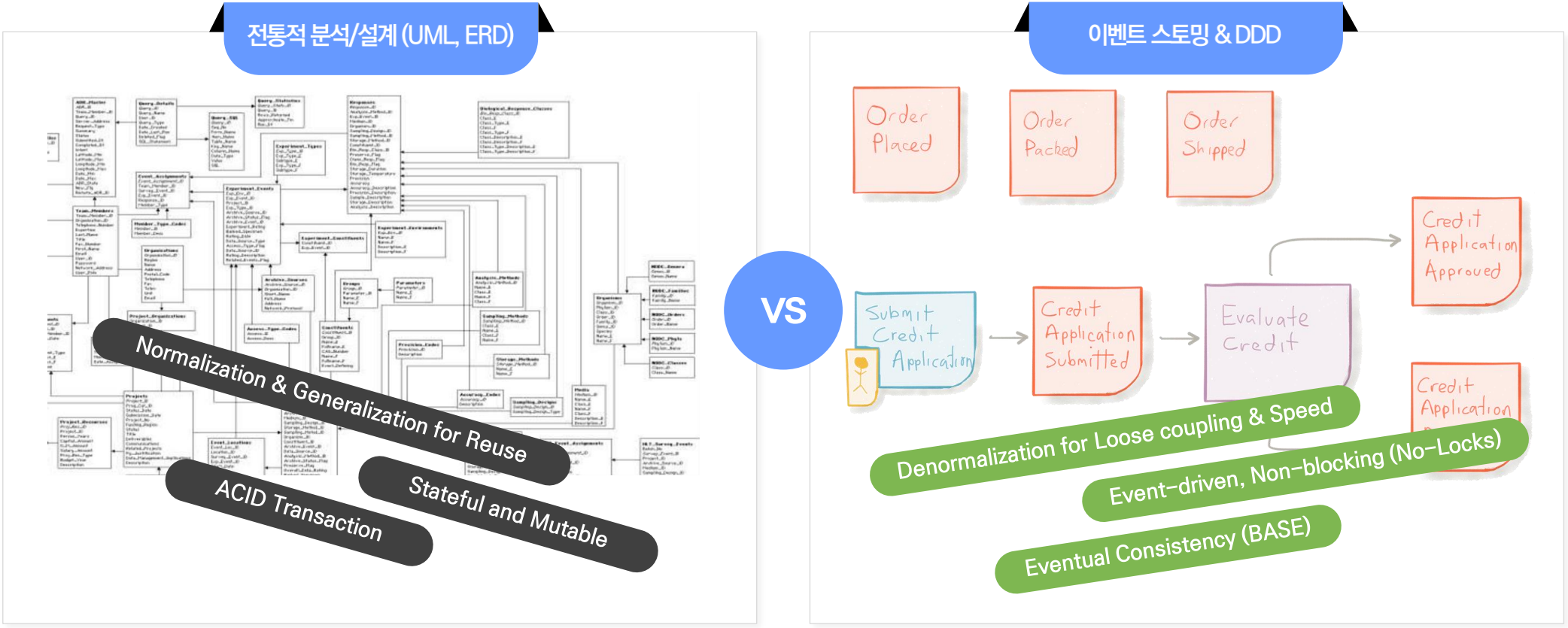
Related to

- ...

Event Storming : MSA 설계를 쉽게 하는 방법

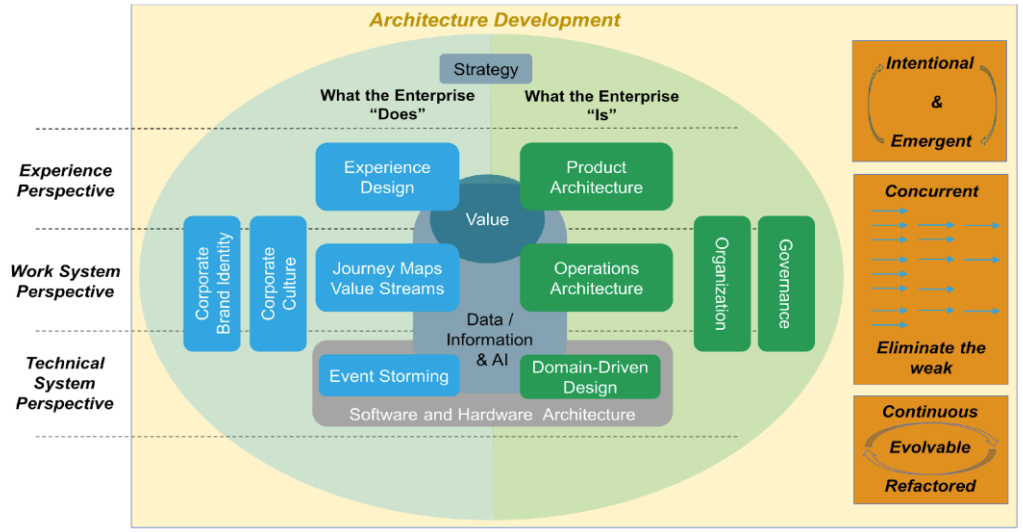
UEngine

- **이벤트스토밍**은 시스템에서 발생하는 **이벤트를 중심 (Event-First) 으로** 분석하는 기법으로 특히 Non-blocking, Event-driven 한 **MSA 기반 시스템을 분석에서 개발까지 필요한 도메인에 대한 탁월하게 빠른 이해를 도모**하는데 유리하다.
- 기존의 유즈케이스나 클래스 다이어그래밍 방식과 다르게 별다른 **사전 훈련된 지식과 도구 없이 진행할 수 있다.**
- 진행과정은 참여자 워크숍 방식의 방법론으로 결과는 **스티키 노트를 벽에 붙힌 것으로 결과가 남으며, 오랜지색 스티키 노트들의 연결로 비즈니스 프로세스가 도출**되며 이들을 이후 BPMN과 UML 등으로 정재하여 전환할 수 있다.



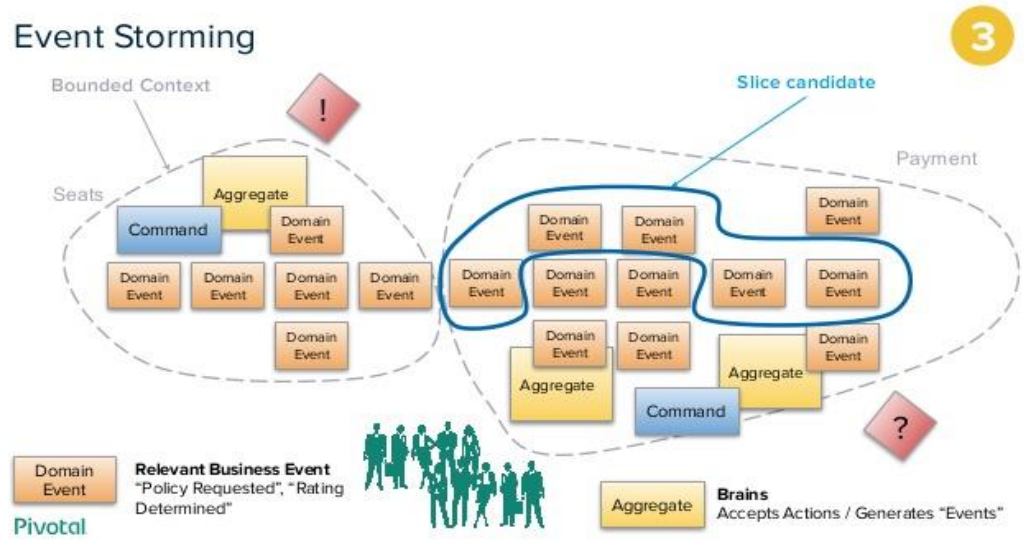
Eventstorming의 위상

The Open Group 의 OAA 의 빌딩블록



https://pubs.opengroup.org/architecture/o-aa-standard/#Introduction

Pivotal 의 AppTX 방법론



https://www.slideshare.net/Pivotal/pivotals-secret-sauce

IBM 의 Garage 방법론… etc

Levels of Eventstorming

BIG PICTURE	EVENTS	HOT SPOTS, SYSTEMS, PEOPLE	CONFLICTS, GOALS, BLOCKERS, BOUNDARIES
PROCESS MODELLING	EVENTS	+ POLICIES, COMMANDS, READ MODELS	VALUE PROPOSITION, POLICIES, PERSONAS, INDIVIDUAL GOALS
SOFTWARE DESIGN	EVENTS	+ AGGREGATES	AGGREGATES, POLICIES, READ MODELS, IDS

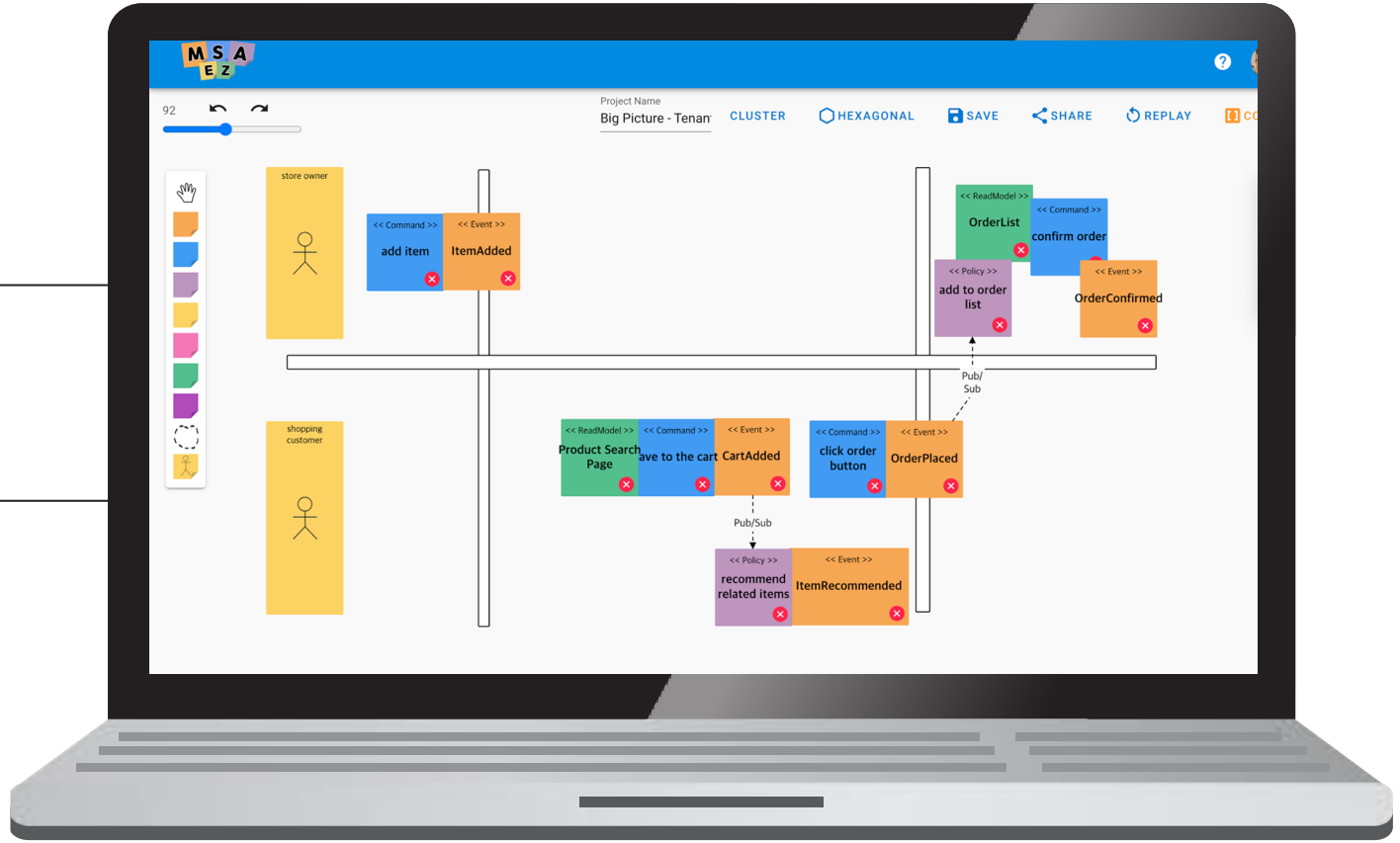
Biz → Big Picture Event Storming

UEngine

Identifies
- Events
- Systems
- People (Actor/Personas)

Derives
- Conflicts, Goals, Blockers
- Boundaries (Bounded Contexts)

Related to
- Process Modeling
- Software Design



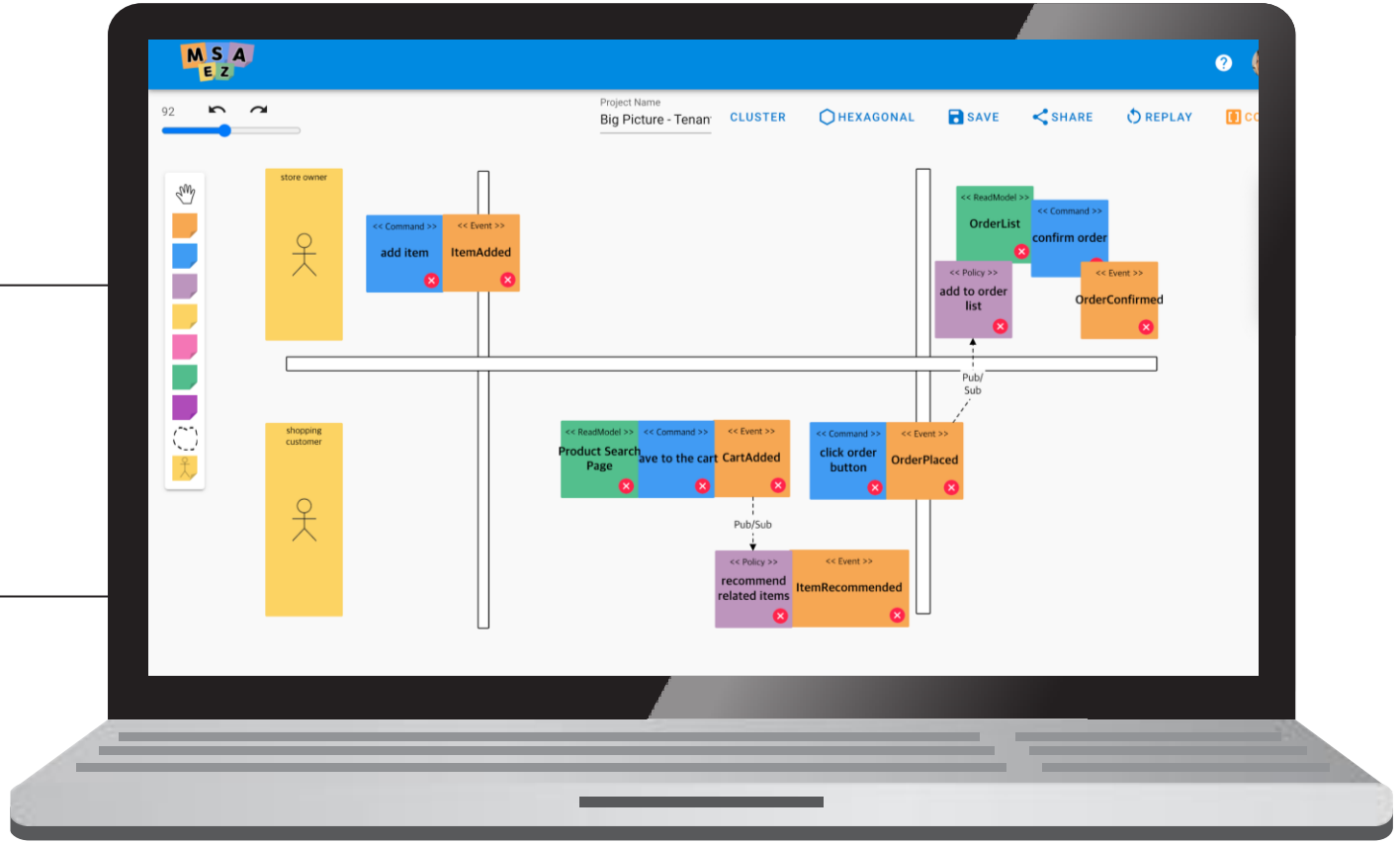
Biz → Process Modeling Eventstorming

UEngine

Identifies
- Policies (Rules)
- Commands (APIs)
- Read Models (UIs)

Derives
- Value Propositions
- Policies
- Personas
- Individual Goals

Related to
- Software Design

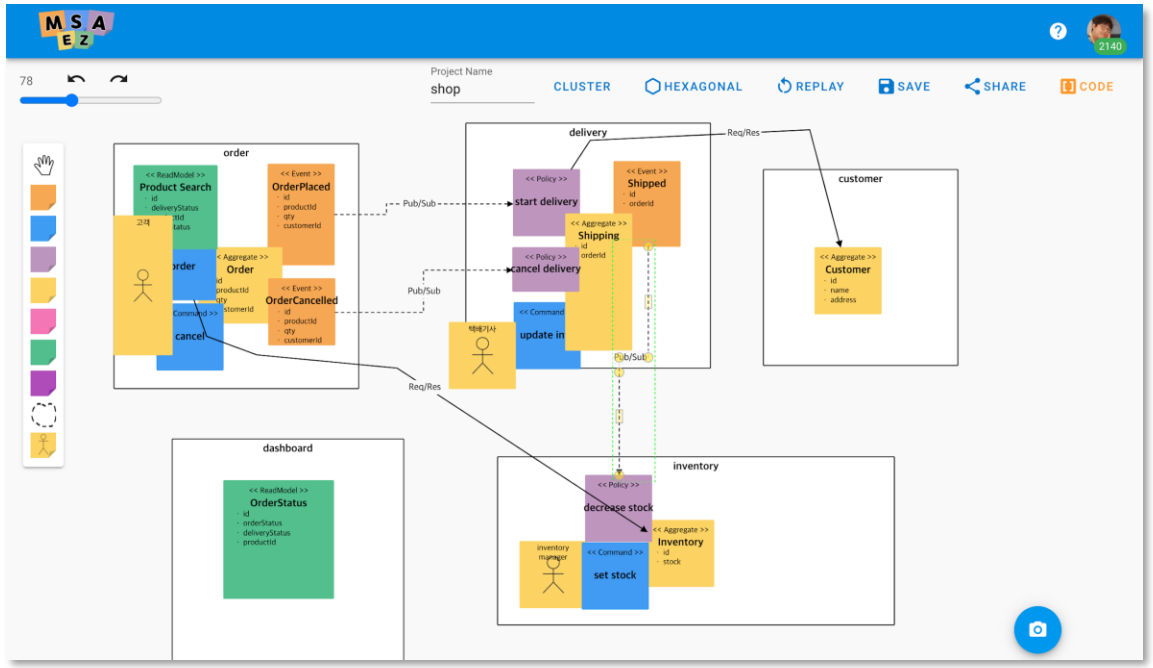


02

Dev Phase

UEngine

Dev - Software Design Eventstorming



Identifies

- Aggregates

Derives

- Aggregates
- Policies Detail
- Read Model Detail

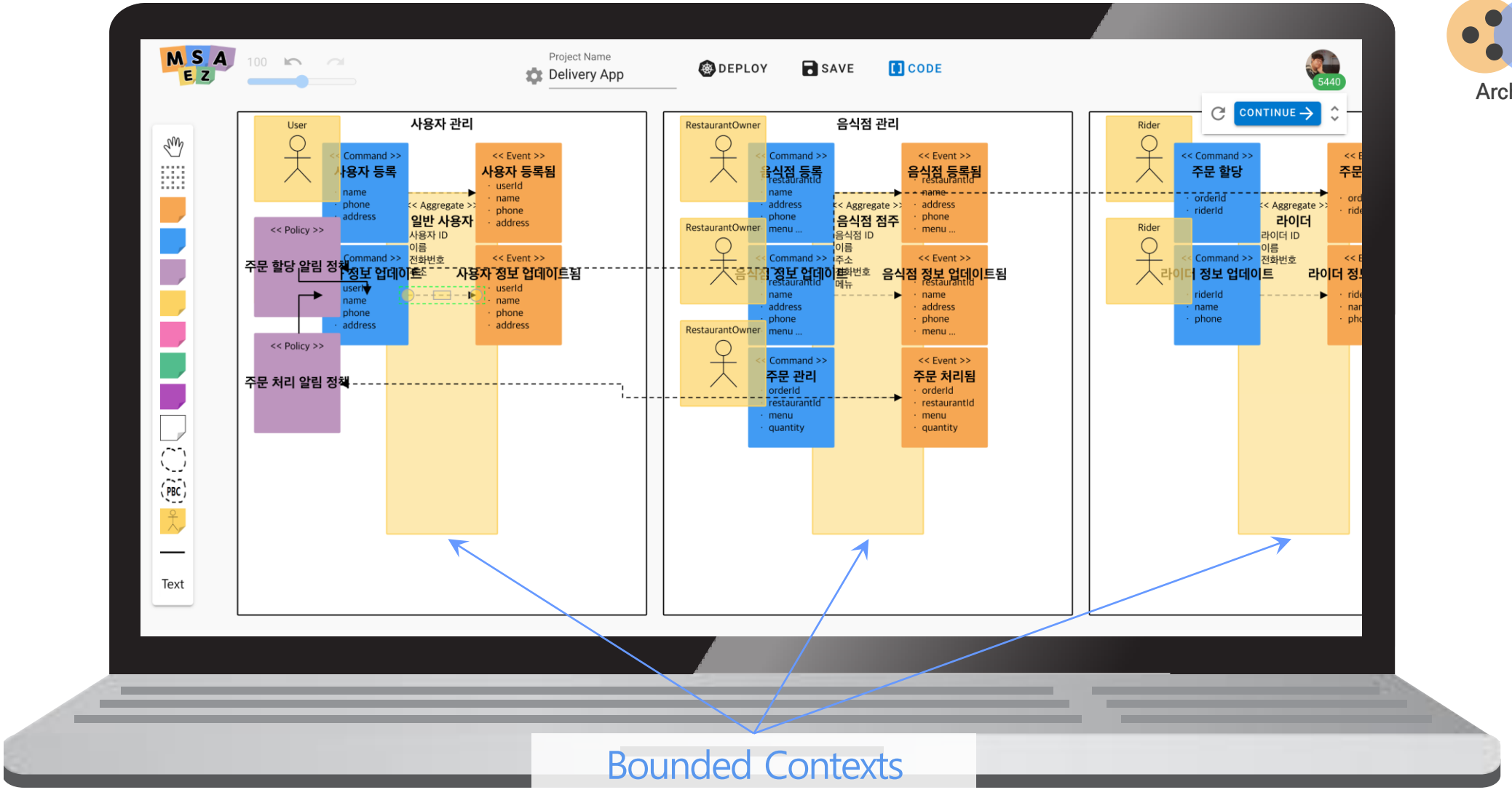
Related to

- Code Generation

이벤트스토밍 모델 생성

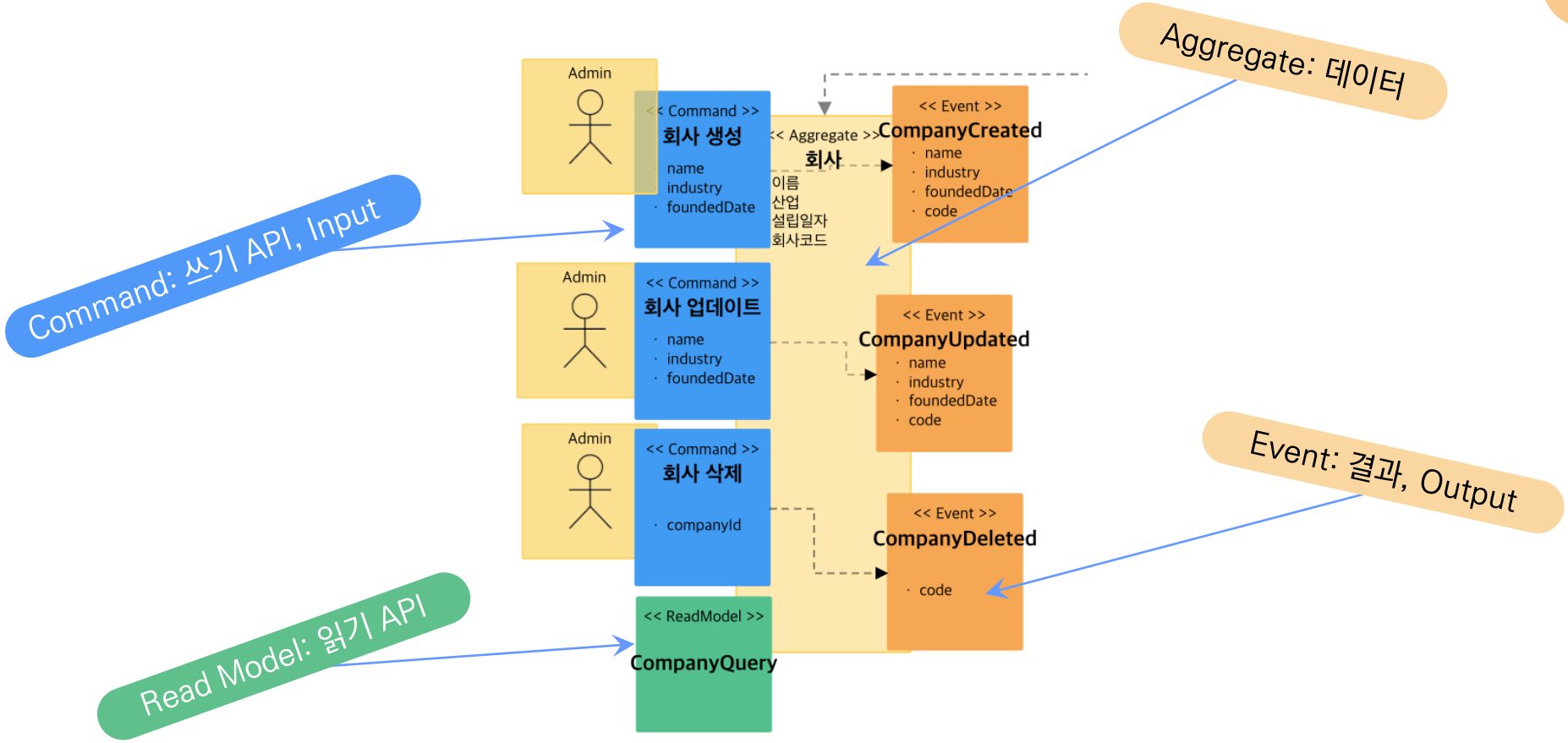
UEngine

Architecting



이벤트스토밍 요소들

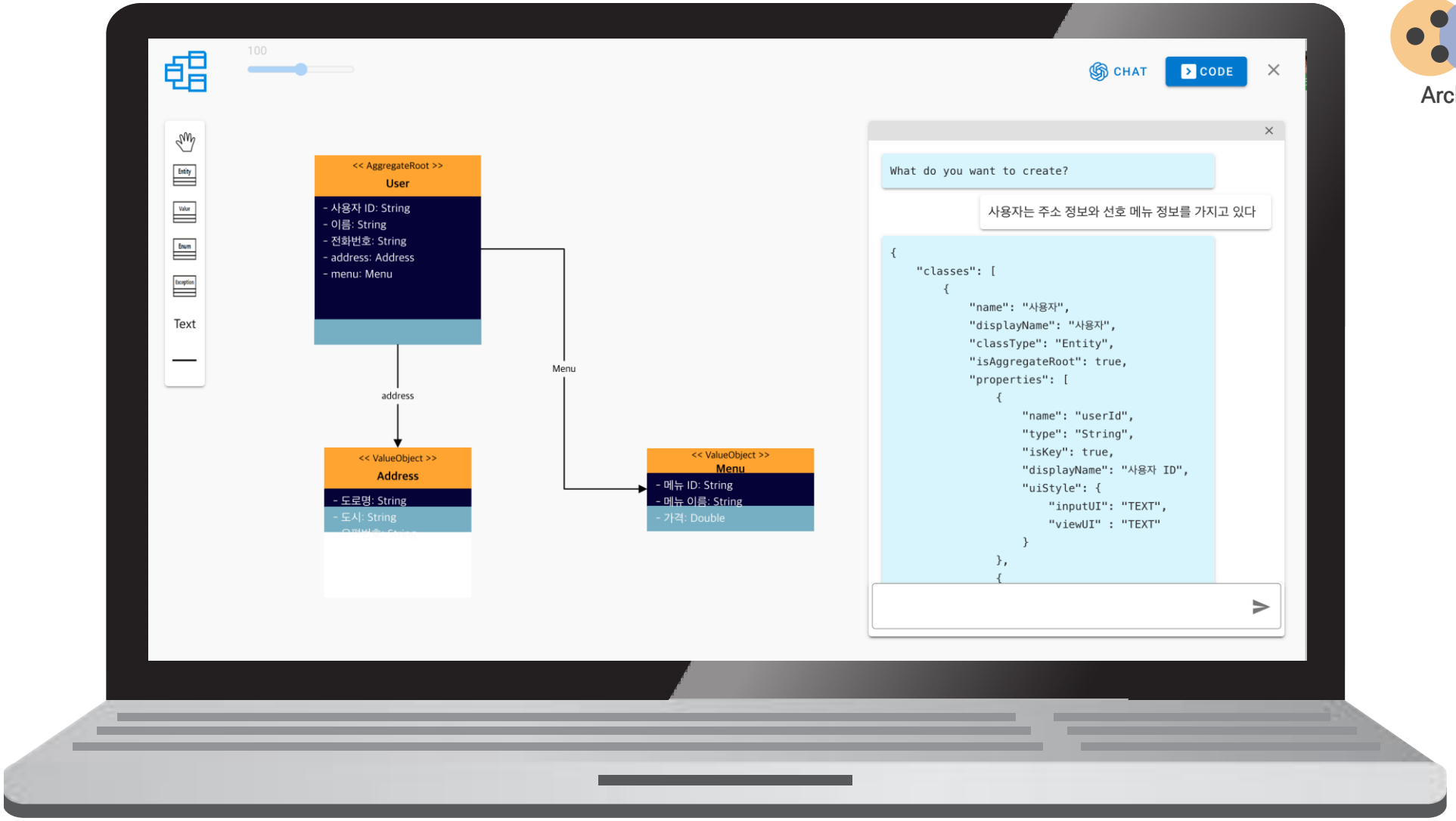
Architecting



도메인 클래스 모델 생성

UEngine

Architecting



Model to Implementation

Business (Domain) Event

Bounded Contexts

Commands

Read Models

Aggregates

Event Driven Architecture

Micro-Services

Deployment Configurations

APIs & Tests

Domain Classes & Test

Example Target

Kafka Events

Spring Boot Project

Kubernetes/Istio Manifests

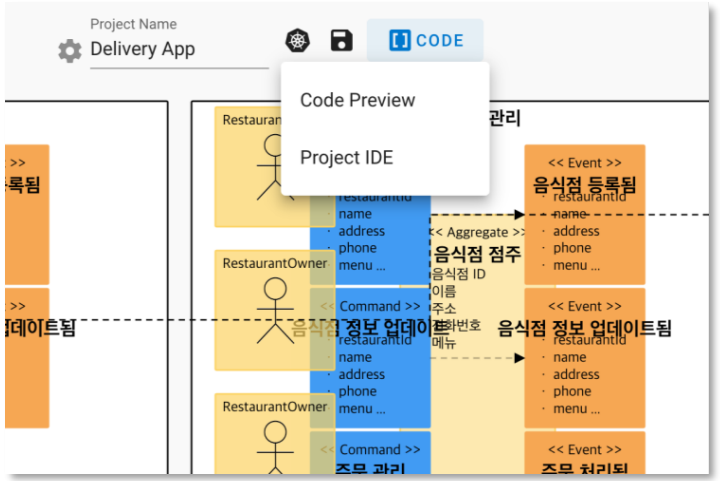
REST Controllers
UI Components
(Buttons/AJAX Calls)
Contract Tests

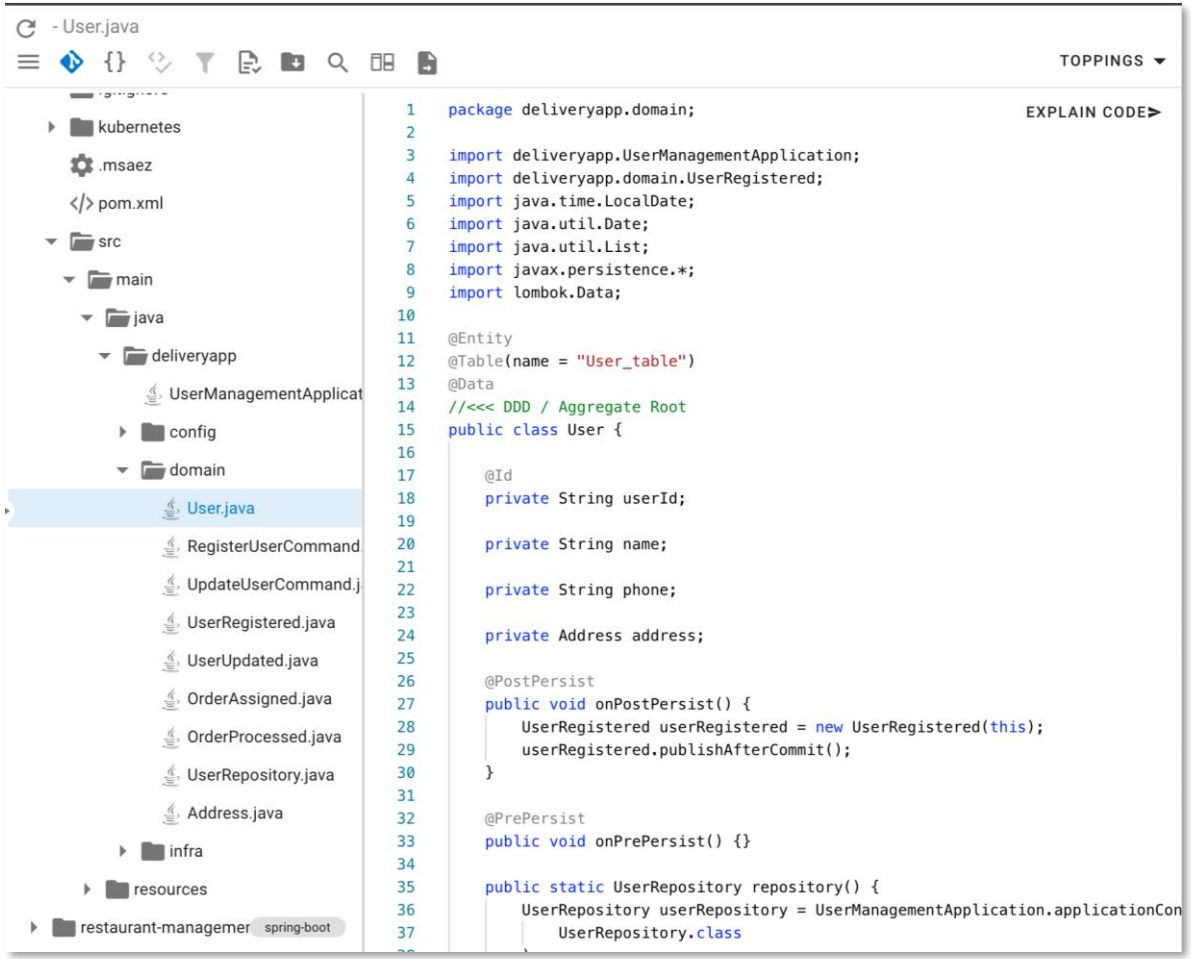
CQRS Code
UI Composition
Graph QL

JPA Entities & Value Objects
VueJS UI
UATs (BDD Cucumber)

코드의 생성

Coding

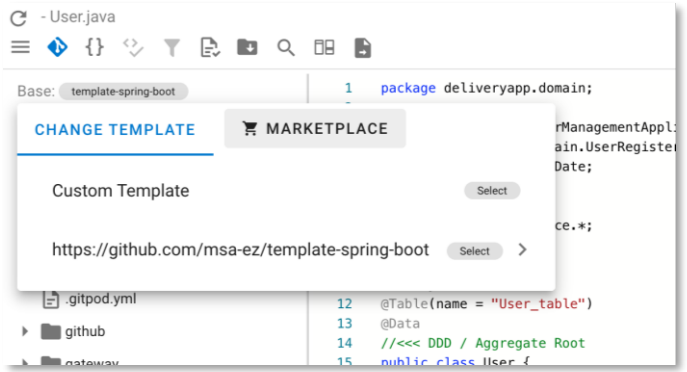


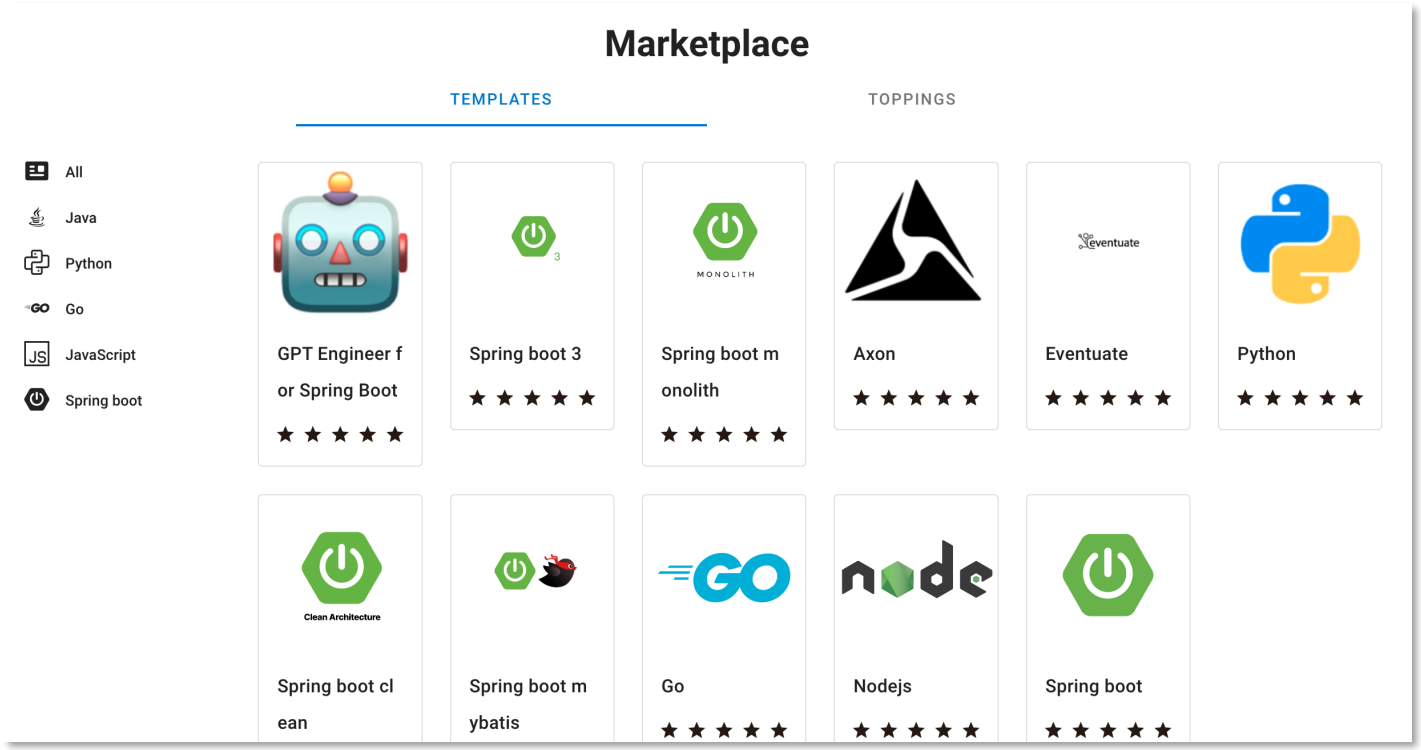


마켓플레이스 - 템플릿의 선택

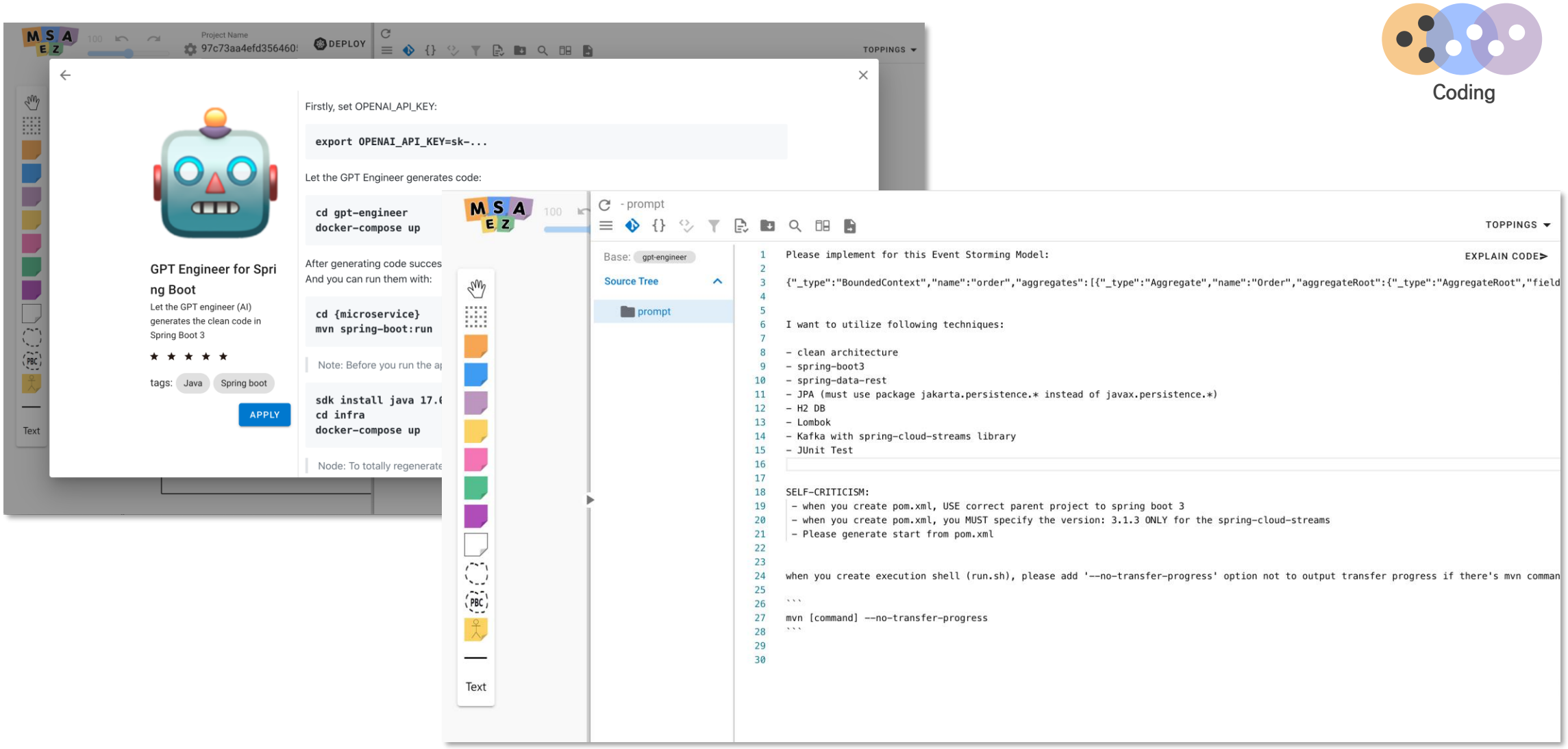
UEngine

Coding





GPT-Engineer 템플릿 선택



Coding

GPT-Engineer 실행하여 코드 생성

UEngine

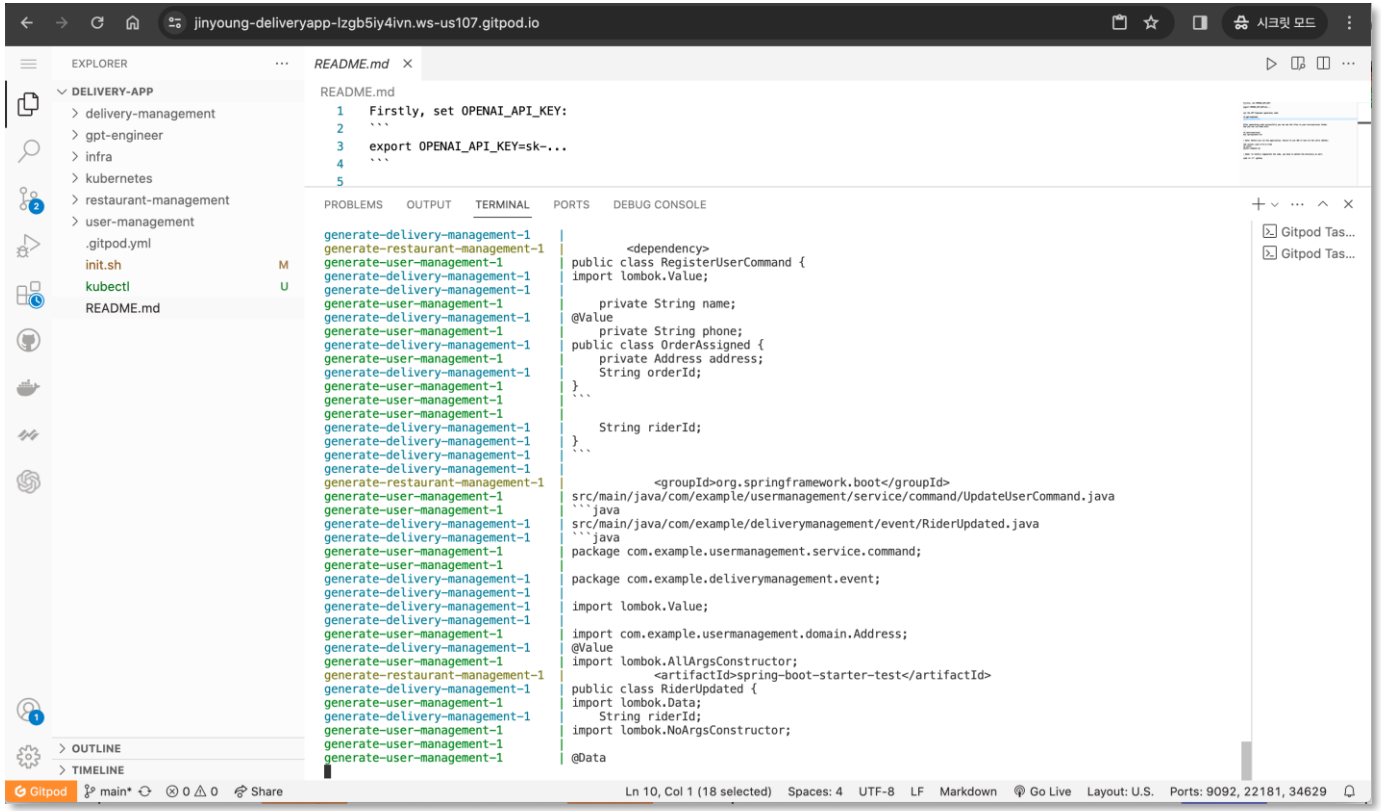
Coding

docker-compose.yml ×

gpt-engineer > docker-compose.yml > { } services > { } generate-user-mana

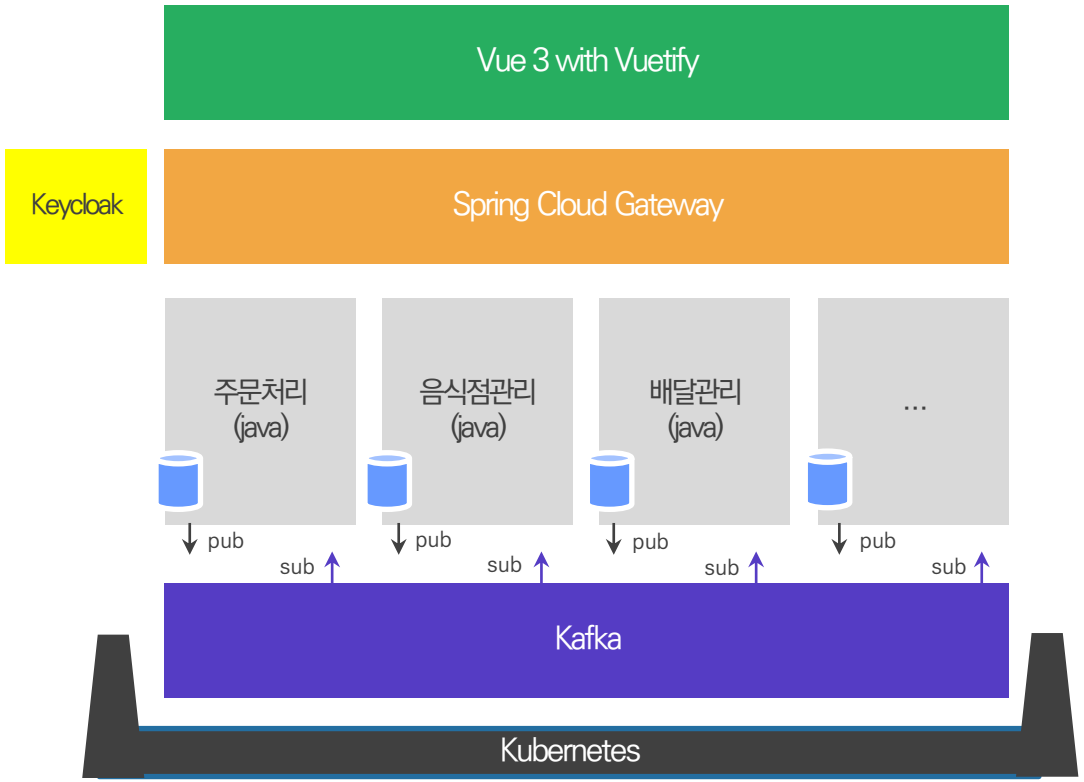
docker-compose.yml - The Compose specification establishes a standard for the defini

```yaml
version: "3"

services:

  generate-user-management:
    tty: true
    stdin_open: true
    environment:
      - OPENAI_API_KEY=${OPENAI_API_KEY}
    image: ghcr.io/jinyoung/gpt-engineer:v5
#    command: ["-sh"]
    volumes:
      - ../user-management:/project
  generate-restaurant-management:
    tty: true
    stdin_open: true
    environment:
      - OPENAI_API_KEY=${OPENAI_API_KEY}
    image: ghcr.io/jinyoung/gpt-engineer:v5
#    command: ["-sh"]
    volumes:
      - ../restaurant-management:/project
  generate-delivery-management:
    tty: true
    stdin_open: true
    environment:
      - OPENAI_API_KEY=${OPENAI_API_KEY}
    image: ghcr.io/jinyoung/gpt-engineer:v5
#    command: ["-sh"]
```

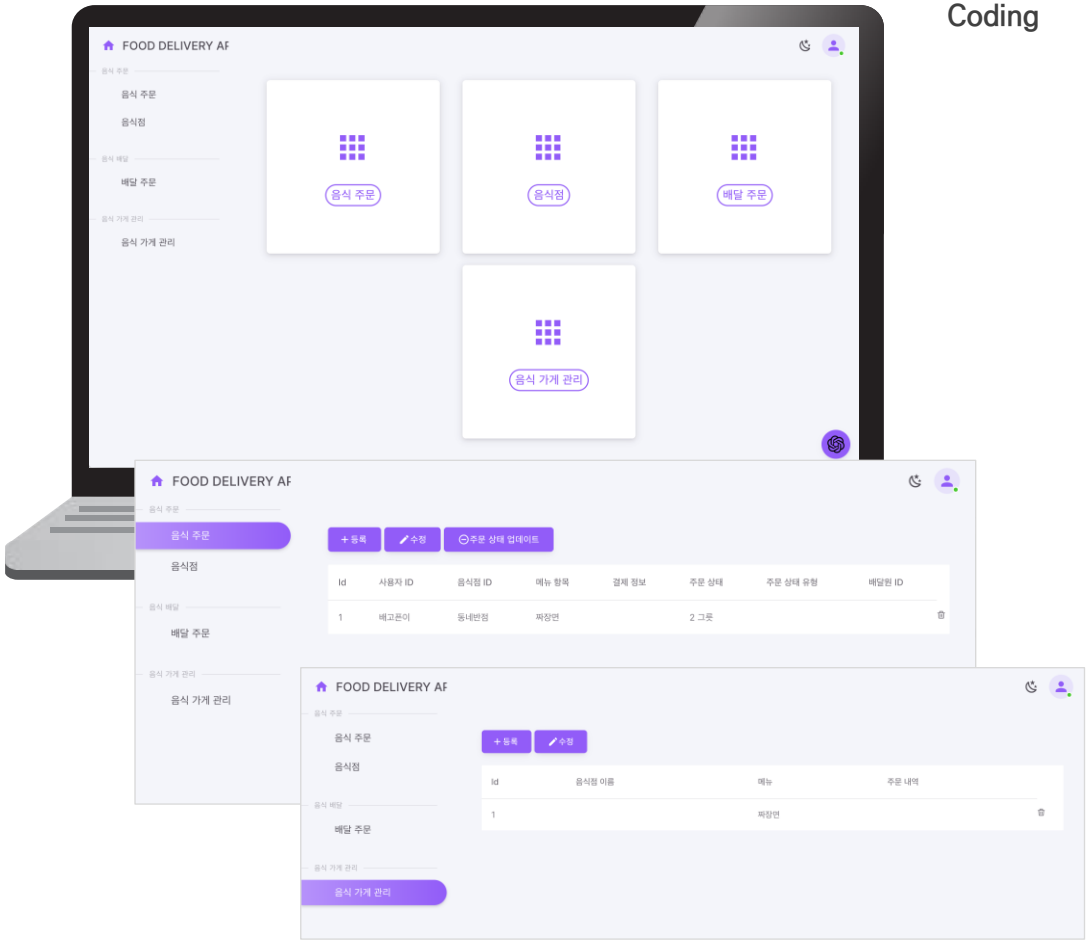


각 마이크로서비스별
에이전트가 병렬로 코드를 생성

완성된 애플리케이션



Coding

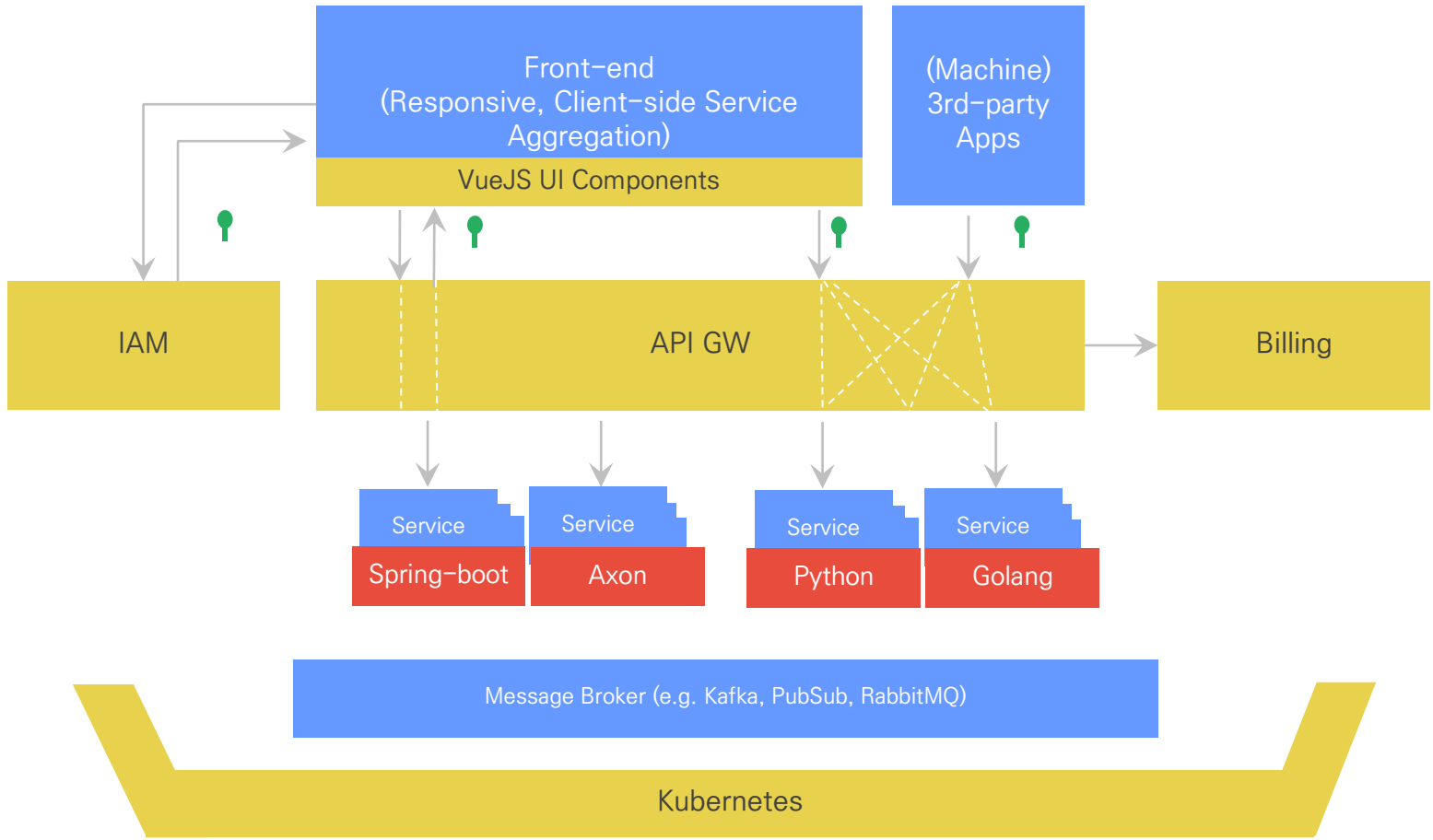


Generated Inner/Outer Architecture

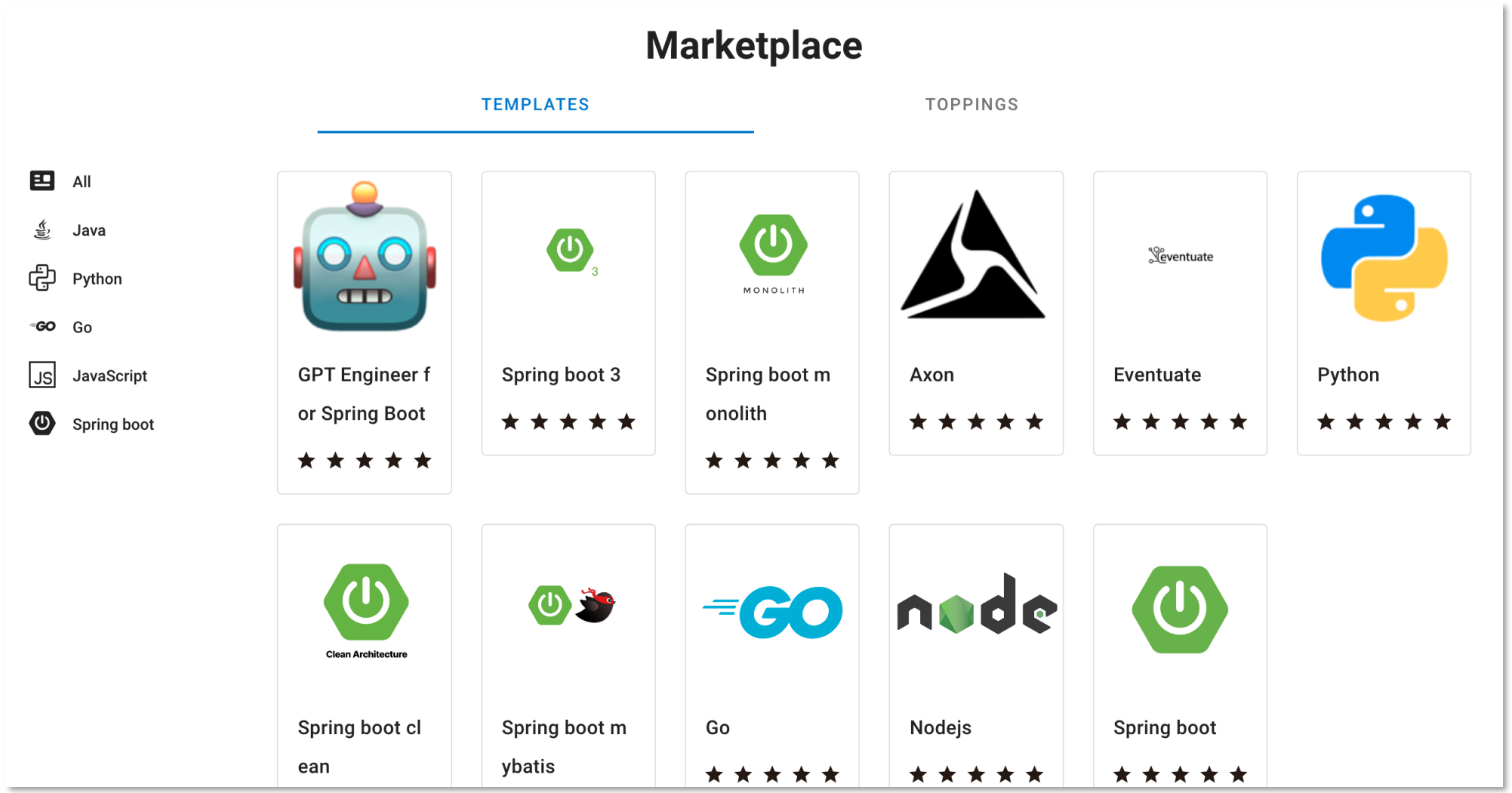
UEngine

Applied MSA Design Patterns

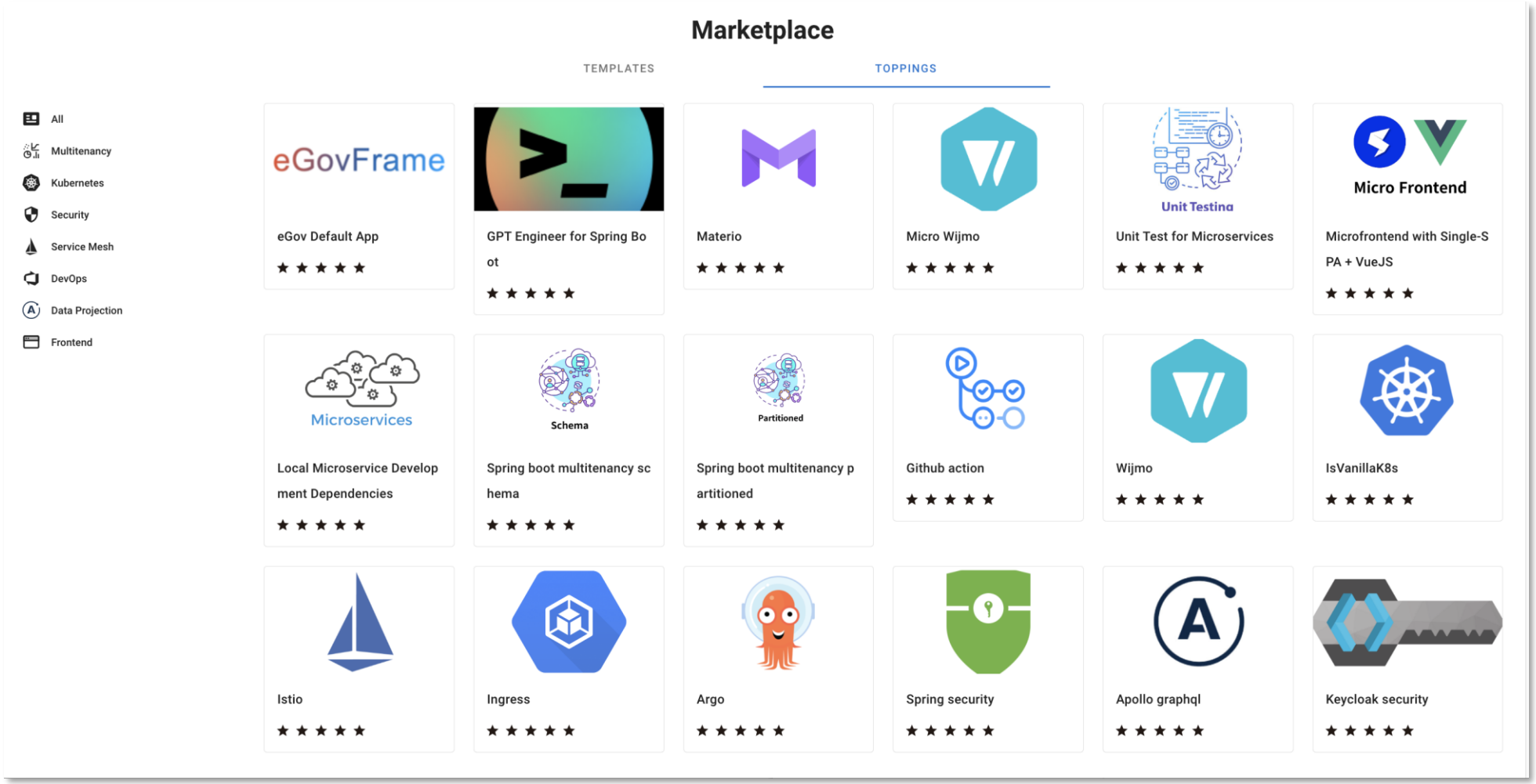
- Containerization
- Database per Service
- CQRS
- Event Sourcing / Sagas
- Contract Testing
- Externalized Configuration
- API Gateway
- Client-side discovery
- Circuit Breaker
- Access Token
- Log Aggregation
- Health Checking
- Distributed Tracing
- Client-side UI Composition



템플릿/토핑 마켓플레이스



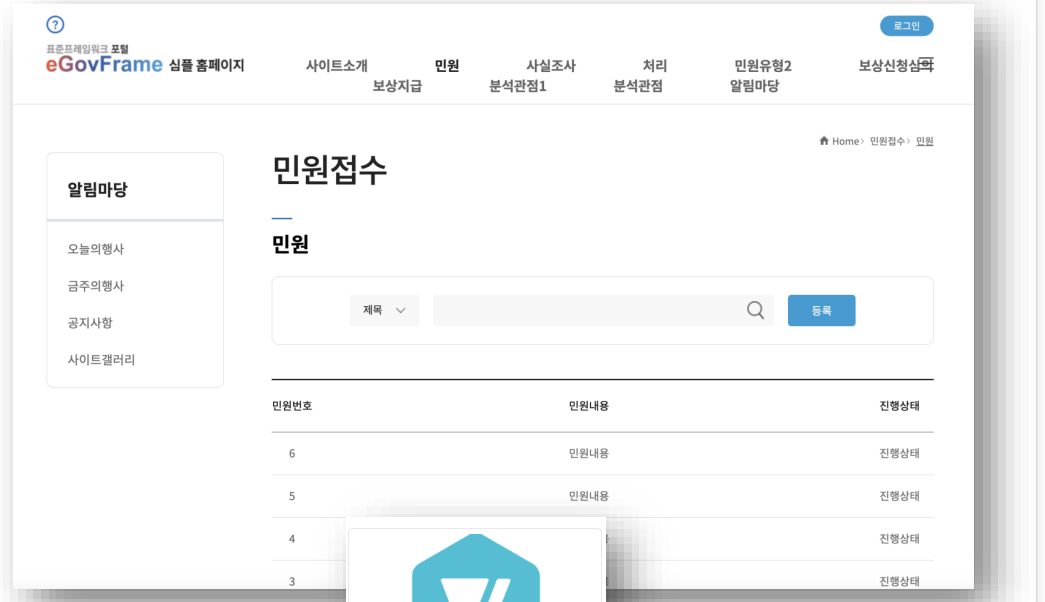
토핑 추가


Marketplace
TEMPLATES
TOPPINGS
All
Multitenancy
Kubernetes
Security
Service Mesh
DevOps
Data Projection
Frontend
eGovFrame
eGov Default App
GPT Engineer for Spring Boot
Materio
Micro Wijmo
Unit Testing
Unit Test for Microservices
Micro Frontend
Microfrontend with Single-SPA + VueJS
Microservices
Local Microservice Development Dependencies
Schema
Spring boot multitenancy schema
Partitioned
Spring boot multitenancy partitioned
Github action
Wijmo
IsVanillaK8s
Istio
Ingress
Argo
Spring security
Apollo graphql
Keycloak security

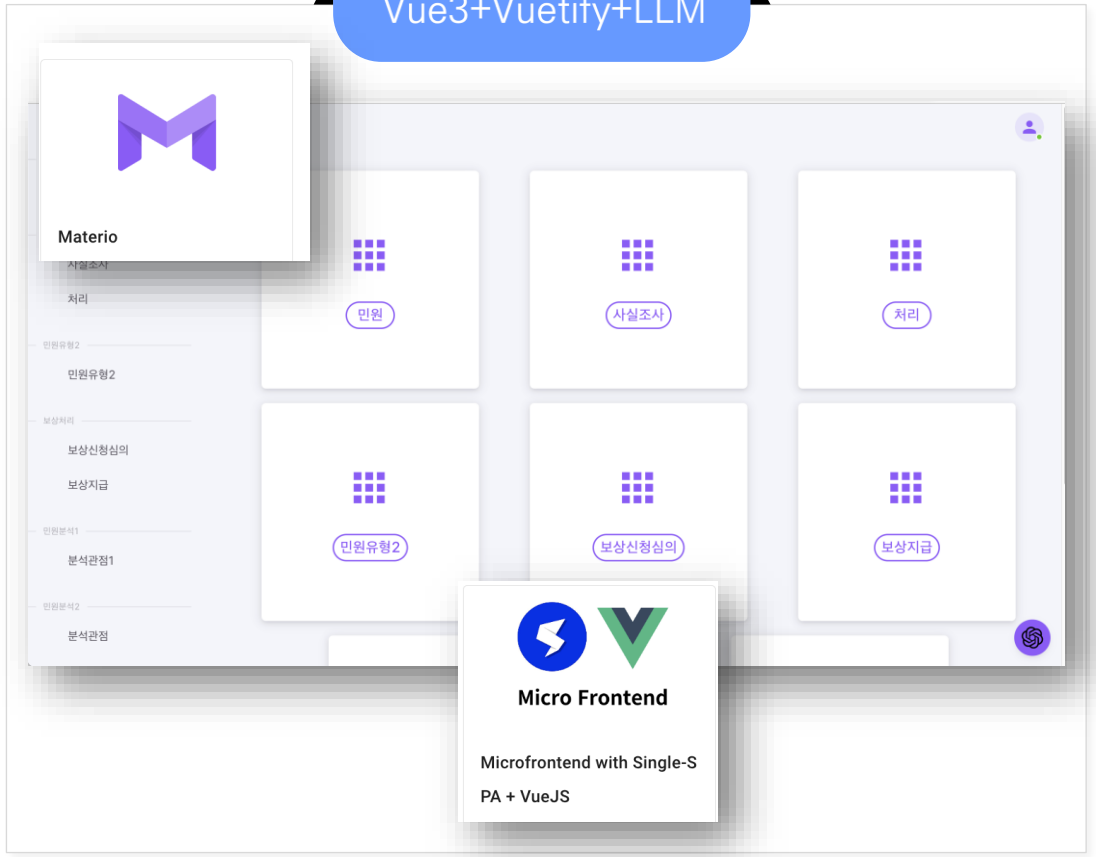
토핑 : 다양한 프론트엔드 지원

UEngine

React



Vue3+Vuetify+LLM



토핑 : DevOps 도구들

UEngine

테스팅

Local Microservice Development Dependencies

★ ★ ★ ★ ★

API 문서화

Unit Test for Microservices

★ ★ ★ ★ ★

쿠버네티스

IsVanillaK8s

★ ★ ★ ★ ★

Ingress

★ ★ ★ ★ ★

서비스 매시

Istio

★ ★ ★ ★ ★

깃 옵스

Argo

★ ★ ★ ★ ★

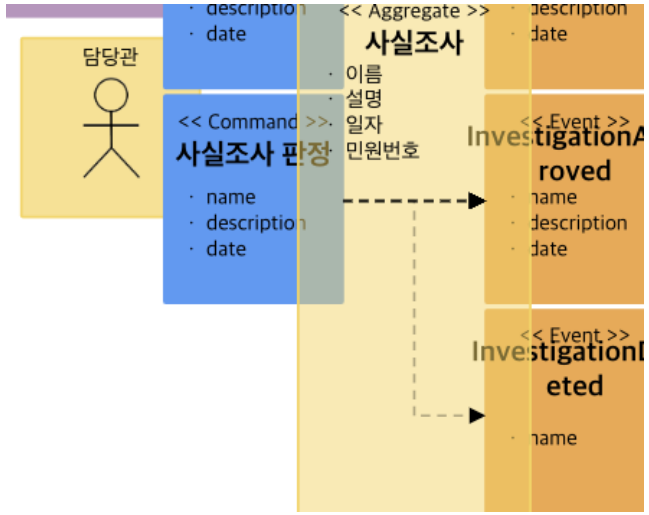
Github action

★ ★ ★ ★ ★

토핑 : 보안 관련

UEngine

Architecting



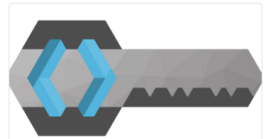
☑ 각 Command에 설정한 액터 → 권한

☑ 스프링 시큐리티

Spring security

★ ★ ★ ★ ★

☑ 키클락



Keycloak security

★ ★ ★ ★ ★

☑ Frontend와 Backend의 보안설정 (@AllowedScopes)으로 코드 생성

토핑 : eGov 표준 프레임워크

MSA Ez는 전자정부 표준 프레임워크의 호환성 확인 가이드라인을 준수합니다.

전자정부 표준프레임워크

eGovFrame

Open Source Open Platform

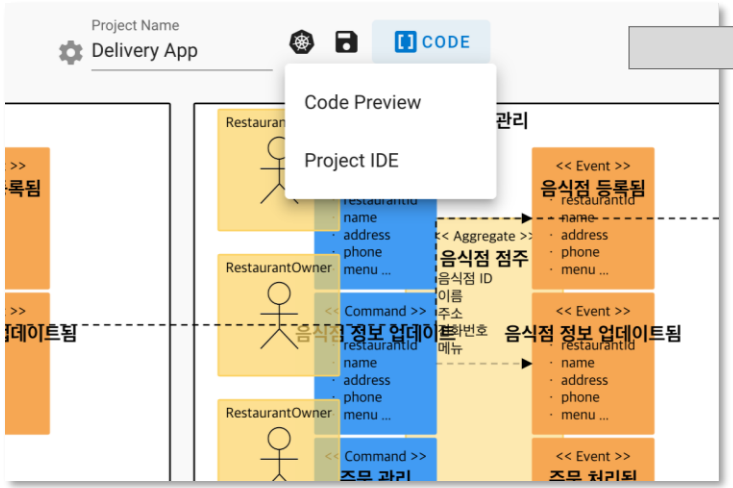
- 전자정부 실행환경 라이브러리를 인위적인 변경없이 그대로 적용합니다.
  호환성 확인 기반 S/W 최소 Maven Dependency 준수 (org.egovframe.rte.xx)
- 설정 파일들은 특정 위치에 존재합니다.
  프로젝트 폴더 내에 xml 파일 중, beans 혹은 sqlMap 엘리먼트 존재 시,
- Controller 클래스들은 전자정부 표준 아키텍처를 준수합니다.
- 클래스간 결합도를 낮추기 위해 서비스 아키텍처는 EgovAbstractServiceImpl 클래스를 확장하며, 인터페이스를 선언하고 이를 구현해 사용합니다.

토핑 : eGov 표준 프레임워크

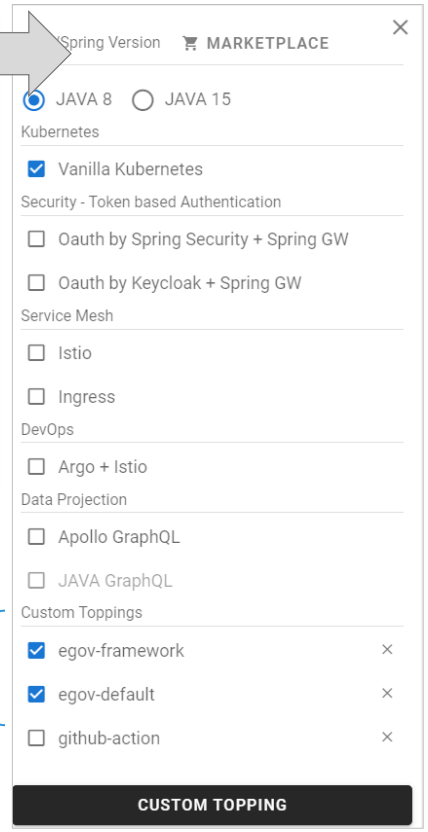
UEngine

Architecting

도메인 모델



표준 프레임 토핑 적용



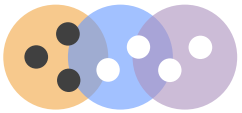
eGov Framework Toppings

코드 자동 생성

- OrderServiceImpl.java

```java
package stmall.service.impl;

import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;
import java.util.stream.StreamSupport;
import org.egovframe.rte.fdl.cmmn.EgovAbstractServiceImpl;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import stmall.domain.CancelCommand;
import stmall.domain.Order;
import stmall.domain.OrderCommand;
import stmall.domain.OrderRepository;
import stmall.service.OrderService;

@Service("orderService")
@Transactional
public class OrderServiceImpl extends EgovAbstractServiceImpl implements OrderService {

    private static final Logger LOGGER = LoggerFactory.getLogger(
        OrderServiceImpl.class
    );

    @Autowired
    OrderRepository orderRepository;

    @Override
    public List<Order> getAllOrders() throws Exception {
        // Get all orders
        return StreamSupport
            .stream(orderRepository.findAll().spliterator(), false)
            .collect(Collectors.toList());
    }
```

토핑 : eGov 표준 프레임워크



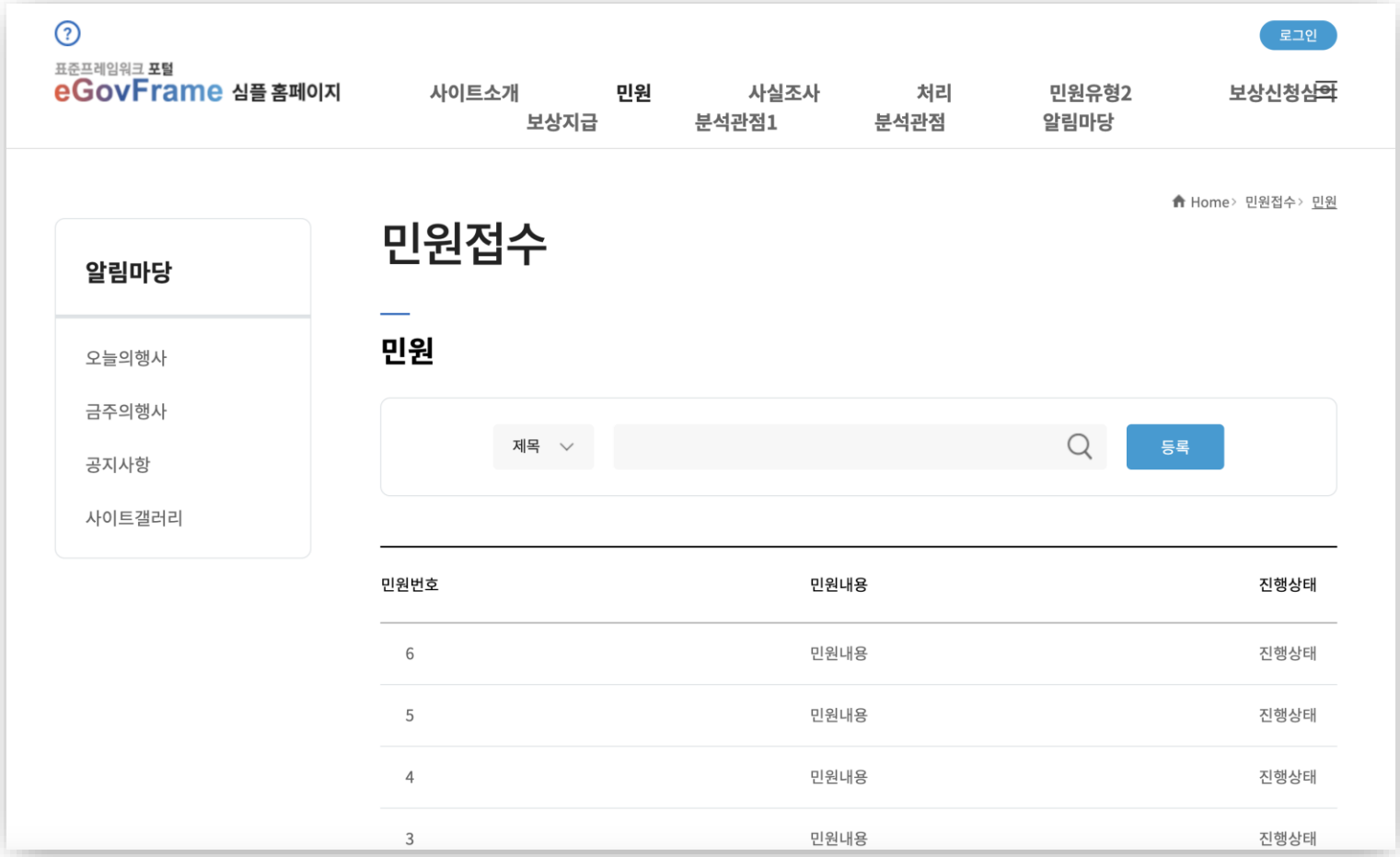
eGov 토핑	설 명	비 고
Egov-framework	▪ 이벤트스토밍으로 설계된 도메인 모델에 대해 표준 프레임을 지원하는 Spring boot 기반의 클라우드 네이티브 코드를 자동 생성한다.	
Egov-default	▪ 표준 프레임워크에서 제공하는 FrontEnd와 BackEnd를 자동 생성되는 코드에 추가한다. ▪ FrontEnd에는 eGov 심플 프론트 엔드에 이벤트스토밍으로 설계된 도메인 모델 코드가 자동으로 추가되어 UI상에 출력된다. ▪ BackEnd에는 eGov 심플 백엔드를 참고용으로 Dockerfile을 포함해 추가한다.	https://github.com/eGovFramework/

토핑 : eGov 표준 프레임워크

UEngine

Architecting

프론트 엔드 생성 예시



Templates : Java Implementations

Template Name	Language	Framework Used	Architecture Style	Recommended For
Spring-boot-clean	Java	Spring-boot Spring Data REST JPA	Clean Architecture (separation of domain class and infra)	Core Domain
Spring-boot-mybatis	Java	Spring-boot My-batis	Transaction Script (SQL) + Active Record (ORM)	Subdomains (Supporting or generic domain)
Quarkus	Java	Quarkus Spring-boot JPA REST-easy	Clean Architecture	Requires High Performance
Axon	Java	Axon Framework	Clean Archictecture (Ref.: …)	Core Domain With Event Sourcing

Templates : Polygolt Implementations

Template Name	Language	Framework Used	Architecture Style	Recommended For
Go	Go	GORM Echo Resty	Clean Architecture (Ref.:https://github.com/bxcodec/go-clean-arch)	Requires high performance and low-footprint of container
Python	Python	Flask SqlAlchemy Request	Clean Architecture (Ref.: URL)	Machine Learning related Automation related
Node	NodeJS	Express Sequelize	Clean Architecture (Ref.:https://betterprogramming.pub/node-clean-architecture-deep-dive-ab68e523554b)	All

Templates : User Interface Implementations

Template Name	Language	Framework Used	Architecture Style	Recommended For
VueJS	JavaScript VueJS	VueJS	Clean Architecture (Ref.: URL)	Low learning curve
React	JavaScript React	React	Clean Architecture (Ref.: URL)	Variety of component market
Angular2	JavaScript Angular2	Angular2	Clean Architecture (Ref.: URL)	Variety of component market

Toppings : Data Projection

Template Name	Language	Framework Used	Functionalities	Recommended For
Apollo GraphQL	JavaScript	Apollo	• GraphQL fetch • RestAPI composition	If you have many Javascript developers
Java GraphQL	Java	Spring Boot GraphQL	• GraphQL fetch • RestAPI composition	If you have many Java Developers

Toppings : Security

Topping Name	Platform	Functionalities	Recommended For
Keycloak + Spring Gateway	• Spring Gateway • Auth provider: Keycloak	OIDC / OAuth2 authn/authzn	If most of your microservices are written in Java and Spring.
Spring Security + Spring Gateway	• Spring Gateway • Auth provider: Spring Security (In house)	OIDC / OAuth2 authn/authzn	If most of your microservices are written in Java and Spring.
Dex + Open LDAP + Istio	• Auth provider: Dex + Open LDAP • Controlled by Envoy Filter	OIDC / OAuth2 authn/authzn	If your microservices are written in diverse languages – polyglot.

Toppings : Service Mesh & DevOps

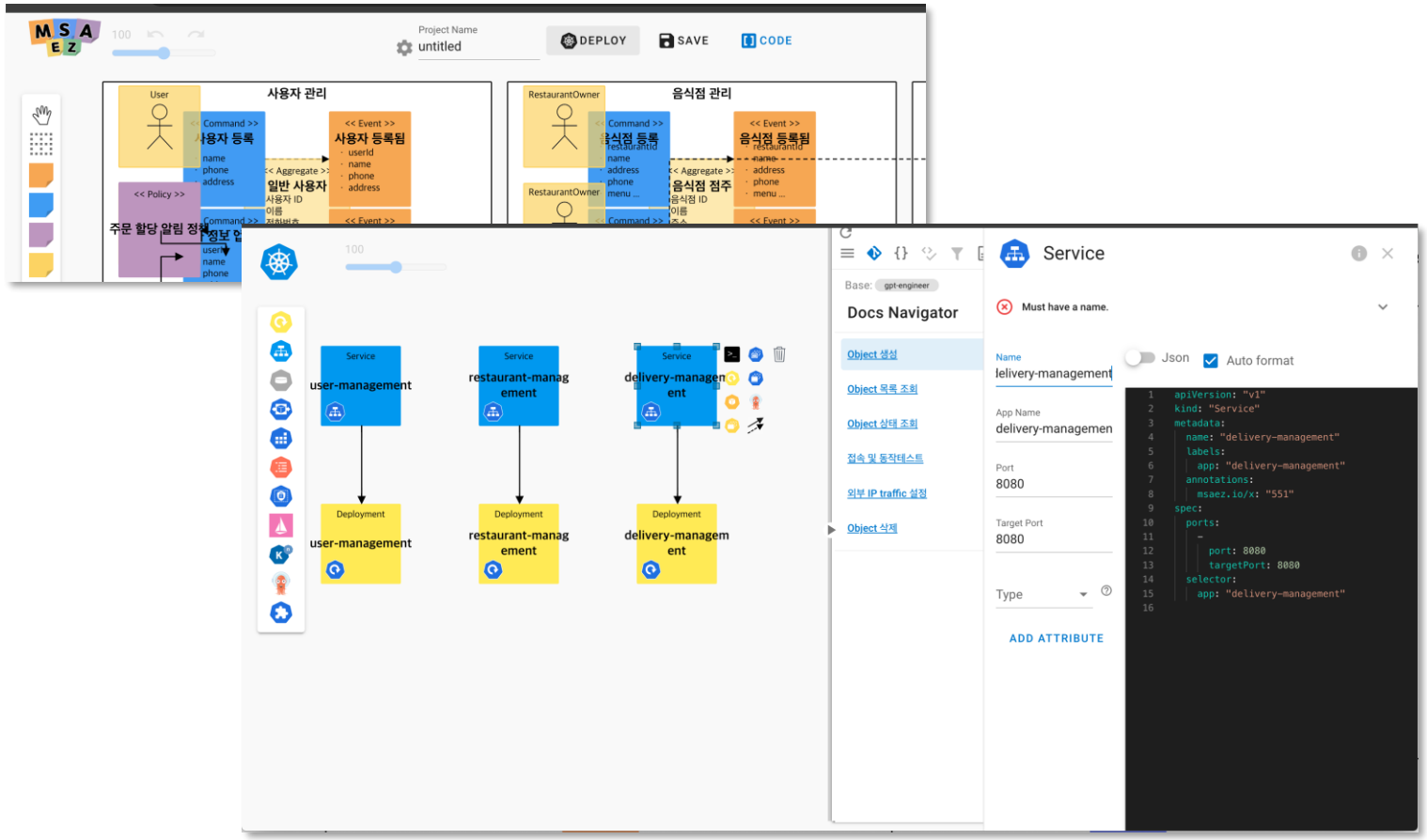
Topping Name	Platform	Functionalities	Recommended For
Ingress	Kubernetes	• Traffic Management	Simple Configuration
Istio	Kubernetes	• Traffic Management • Circuit Breaking / Rate Limiting • Access Control • Distributed Monitoring	Advanced Resiliency, Dynamic Deployment, Monitoring
Argo CD + Rollout	Kubernetes	• Canary Deployment • GitOps	Advanced Deployment

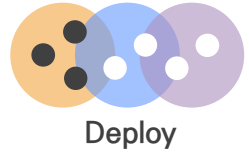
03

Ops Phase

UEngine

Kubernetes 모델링 선택





K8S 상의 클라우드 네이티브 아키텍처를 표현하는 매니페스트 작성과 운영 전 단계를 모델링과 GUI로 관리

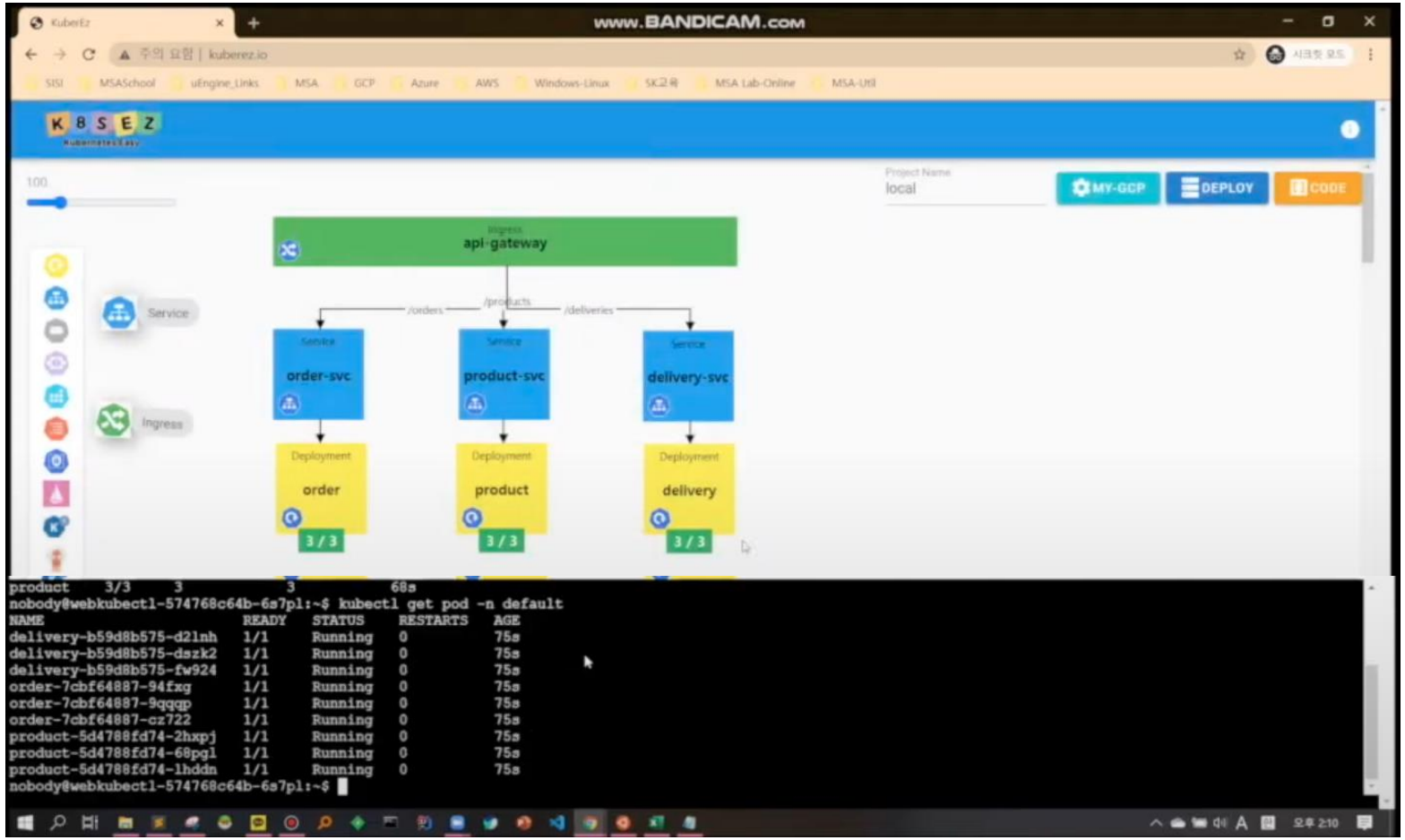
- 아키텍처 문서 = 디플로이 현환
- 직관적인 디플로이 UI
- 모니터링 시각화
- 운영 개선 힌트

– 시각적인 CNA 설계 및 운영

Supported Kubernetes Objects

UEngine

Deployment Monitoring



지속적 통합과 배포를 위한 설정을 해놓으면 주기적으로 최신의 소스 형상에서 개발기를 업데이트하고 운영기로 배포

- 모델링된 배포 모델에 기반한 배포 과정 모니터링
- 웹 터미널 기반(k9s) CLI 도구 통합
- UI 액션에 대응되는 kubectl 명령 미러링을 통한 명령 학습
- Istio 등의 객체를 통한 카나리 디플로이, 섀도우 디플로이 모델의 설정과 Scheduling을 통한 디플로이
- ArgoCD와의 연동을 통한 통합 모니터링

04

Values

AS-IS vs TO-BE

- 개발자 개인기에 의존한 클라우드 네이티브 구현
- 마이크로 서비스 아키텍처, 컨테이너 등 다루어야 할 기술 요소가 너무 많아

빠른 클라우드 전환

품질 높은 클라우드 아키텍처와
비즈니스 모델로의
안정적이고 빠른 전환

자동화된 운영

높은 업타임,
끊김 없는 서비스,
직원의 행복지수 향상,
고객 만족도 향상