



성공적 디지털 서비스 전환을 위한

MSA-Easy



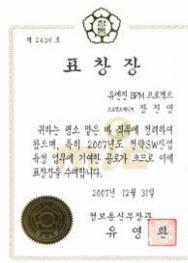


유엔진솔루션즈는 10년간의 BPM, SNS, Cloud 솔루션을 개발해오면서 쌓아온 기술력과 제조/금융 분야의 적용 경험으로, **본 서비스를 성공적으로 구축할 수 있는 기술 전문 기업**입니다.

업력	<ul style="list-style-type: none"> • 2007년 03월 20일 설립
주요 고객사	<ul style="list-style-type: none"> • 한화그룹, 기업은행, 삼성전기, SKT, SKC&C, 현대기아차, 대우건설, 경동나비엔, 생산기술원 등
주요사업분야 및 보유 기술	<ul style="list-style-type: none"> • 클라우드 컴퓨팅 / SOA 컨설팅 및 시스템, SaaS 지원 시스템 ETRI 공동연구 (2010) • 세계적 오픈소스 커뮤니티(uEngine.org, Open Cloud Engine)운영 • BPM (업무프로세스관리), BRE(비즈니스로엔진) • 기업용 소셜 네트워크 기반 협업 시스템 • ALM(어플리케이션 수명주기관리)

인증 및 수상

2012년 SW산업발전
〈국무총리 표창〉



2007년도 전략 소프트웨어
〈산업 육성 공로상〉



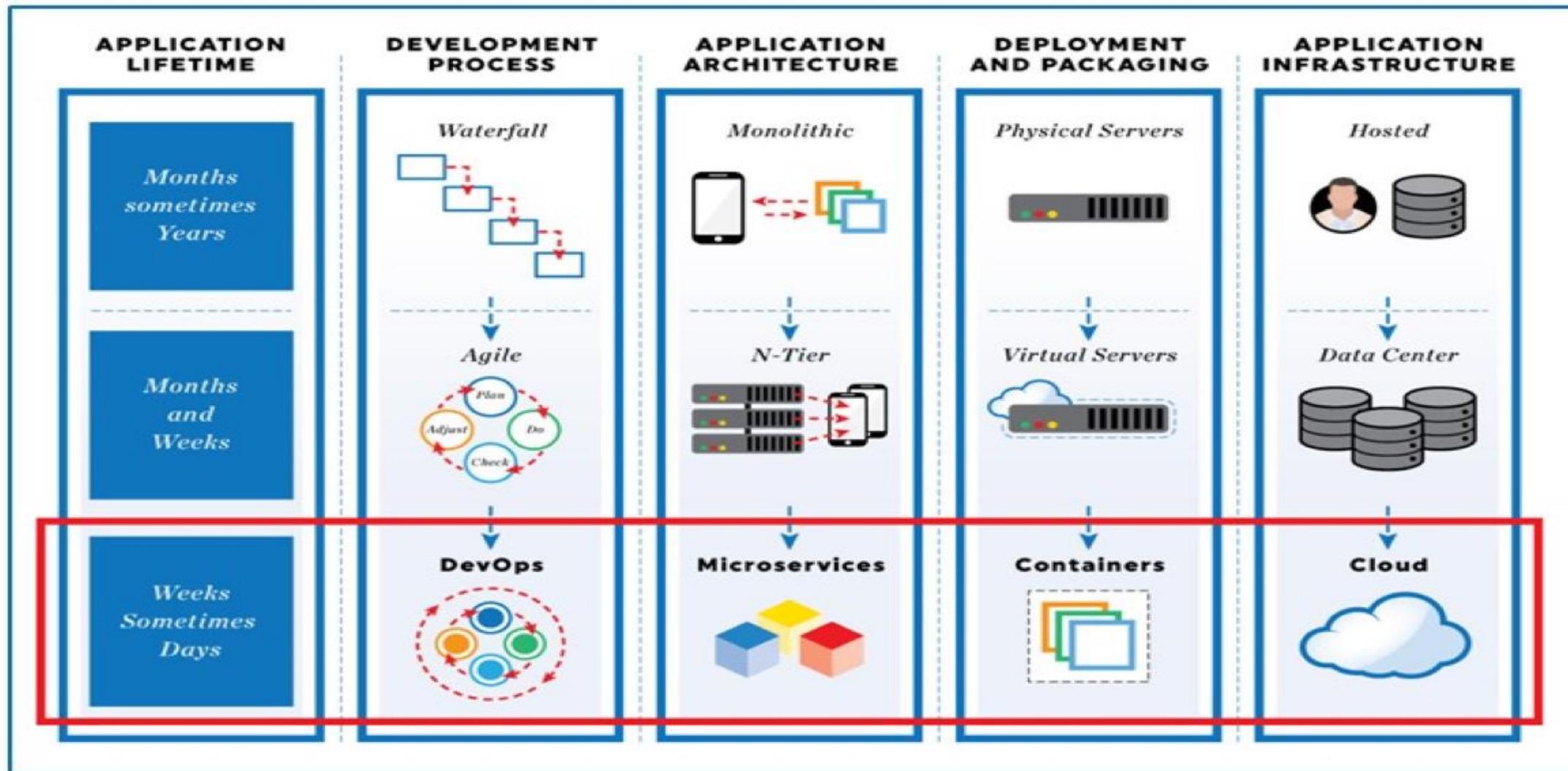
2004 Ranked 60th in top Source
forge Open Source Project



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



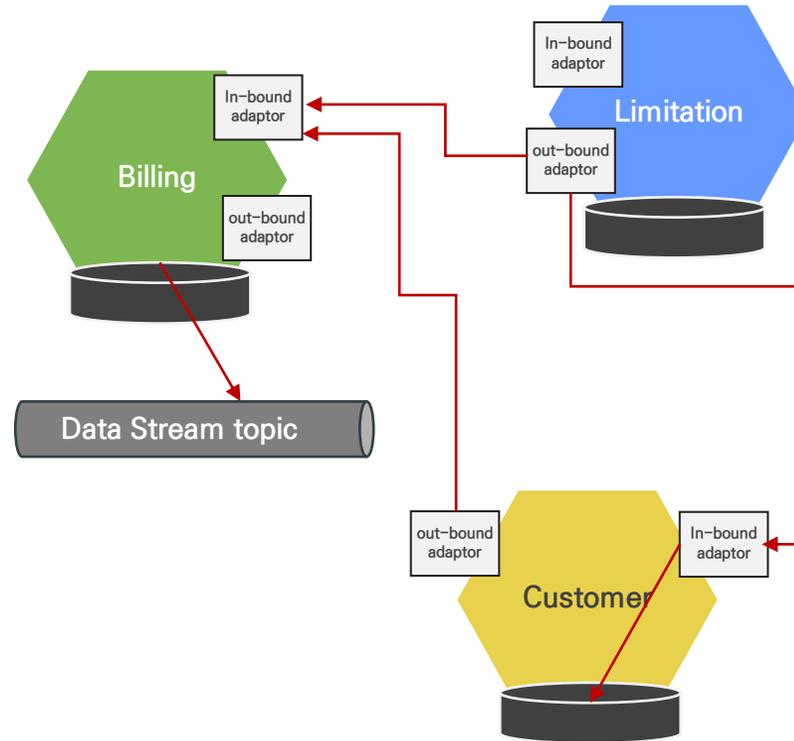
DT 전환에 필요한 것들



- ✓ 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



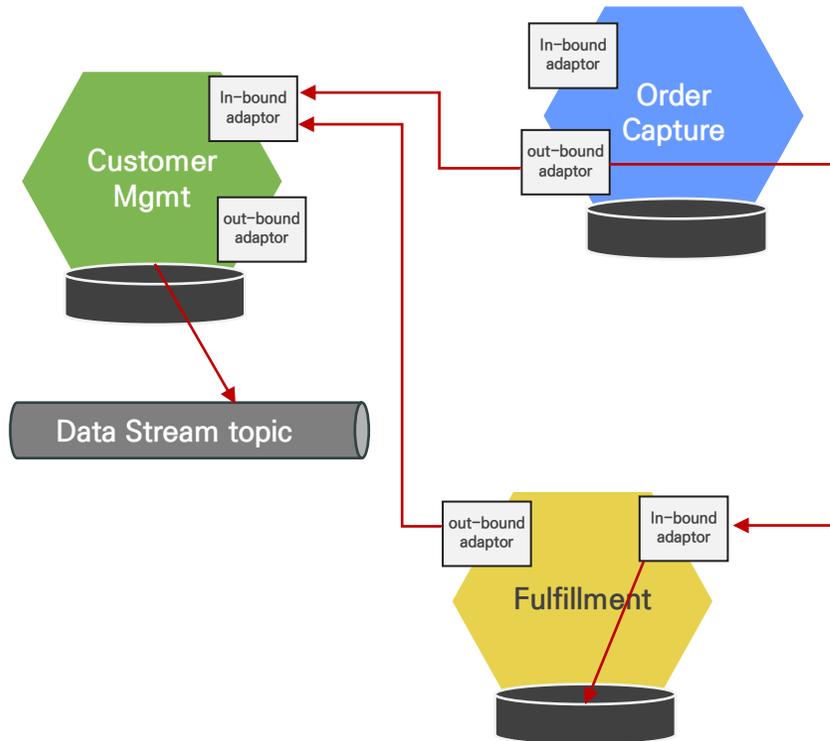
- ✓ Smaller, simpler code base



- ✓ Options for scaling

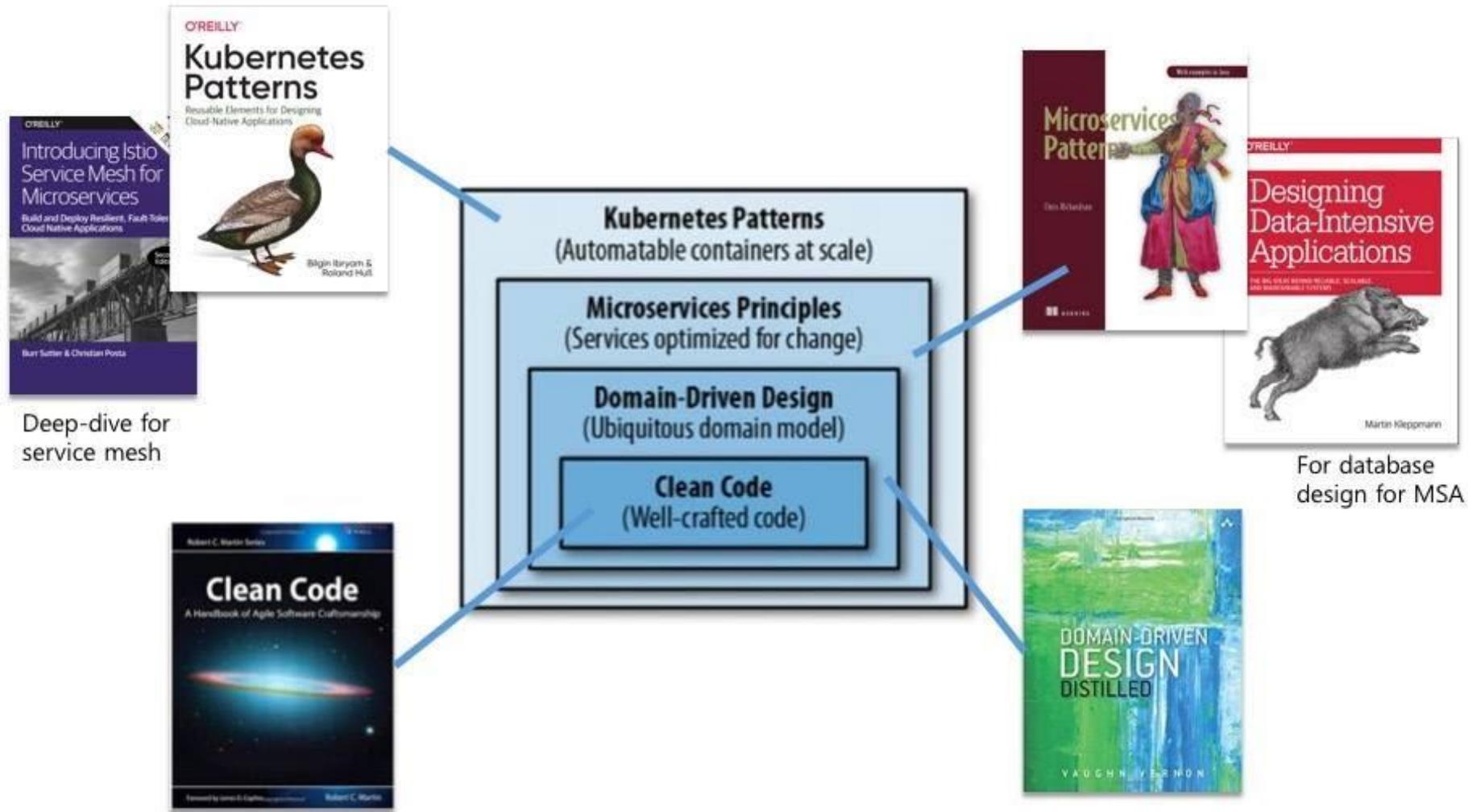
Learning Curves to a Microservice Architecture

☑ 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

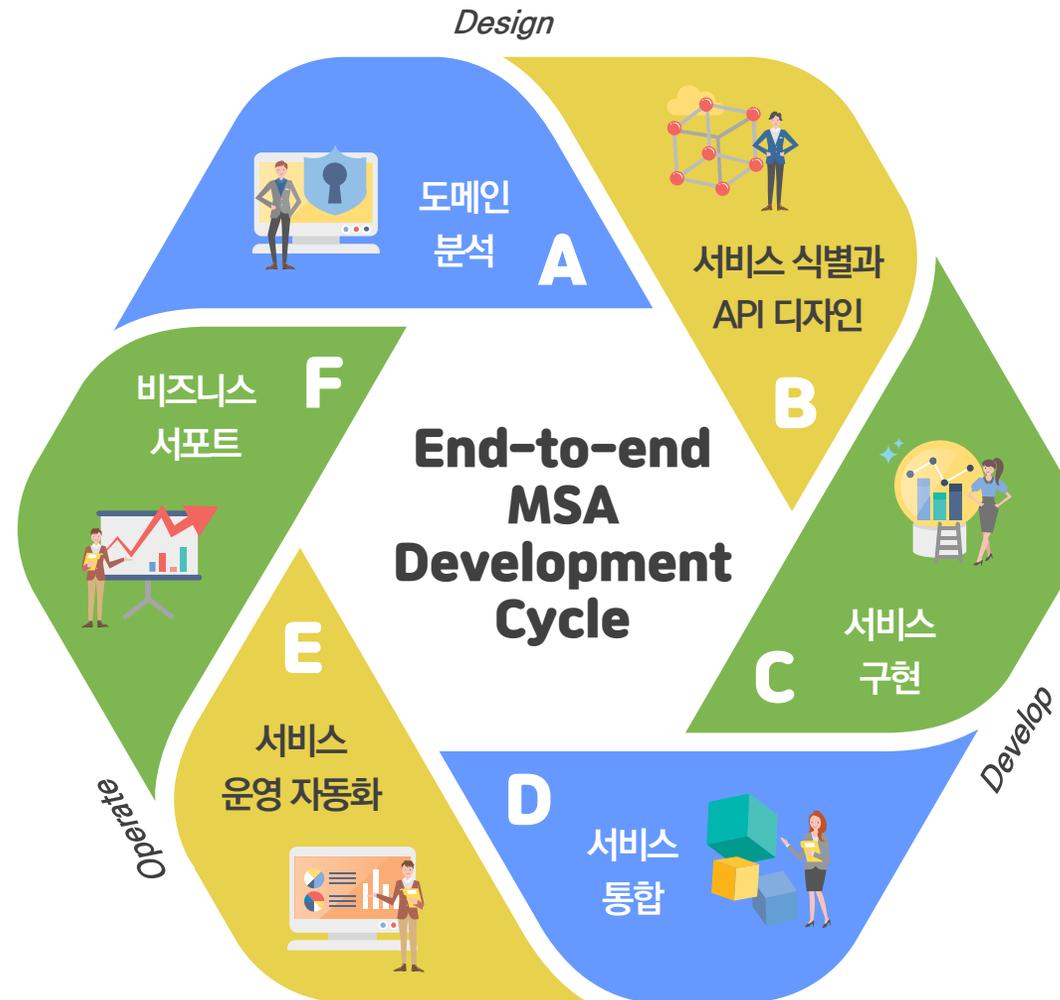




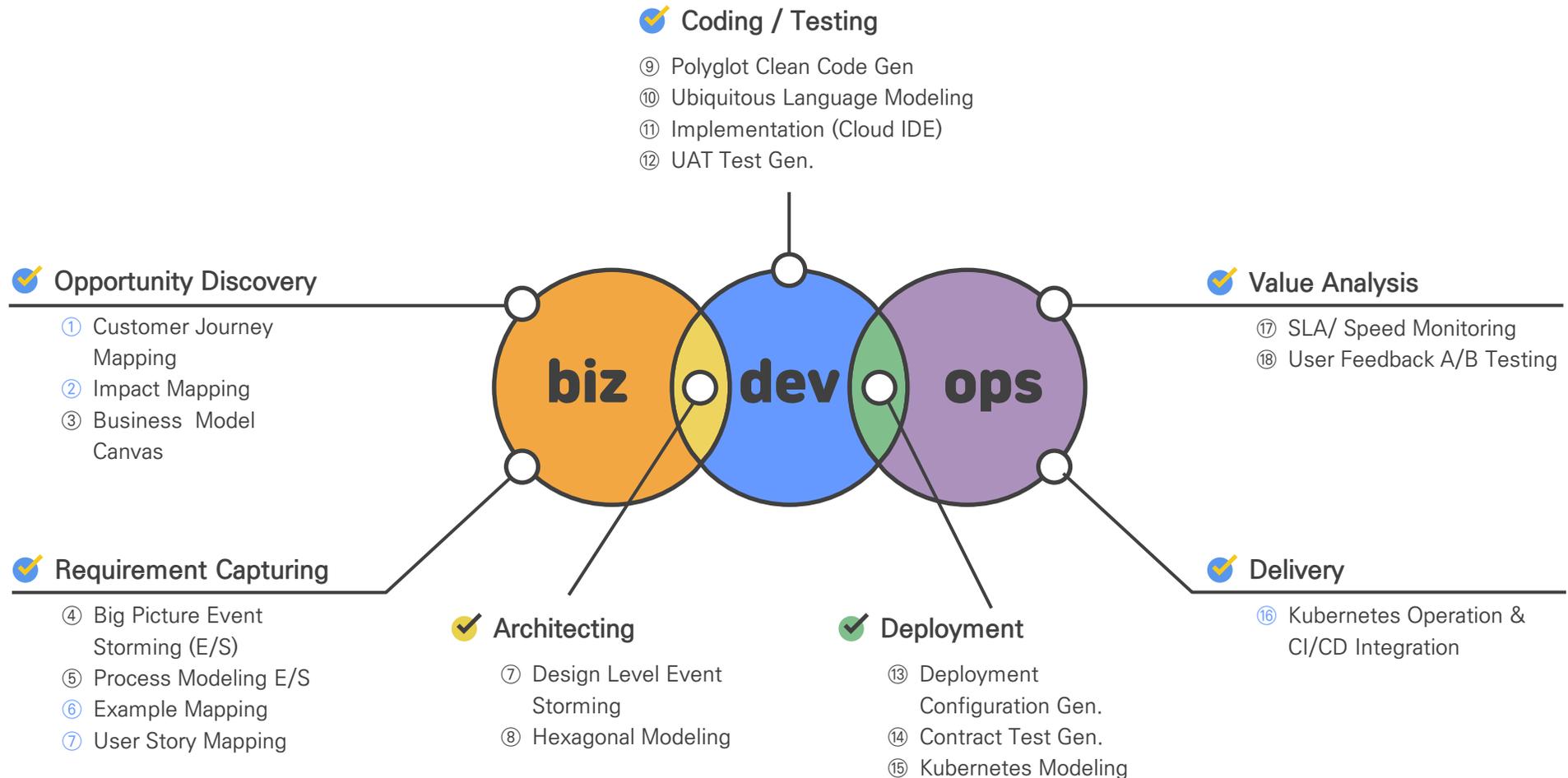
MSA-Easy 는 DDD 기반 아키텍처
분석에서
운영자동화까지 한번에 지원하는
MSA 운용 플랫폼입니다

<http://msaez.io>

<http://github.com/msa-ez>



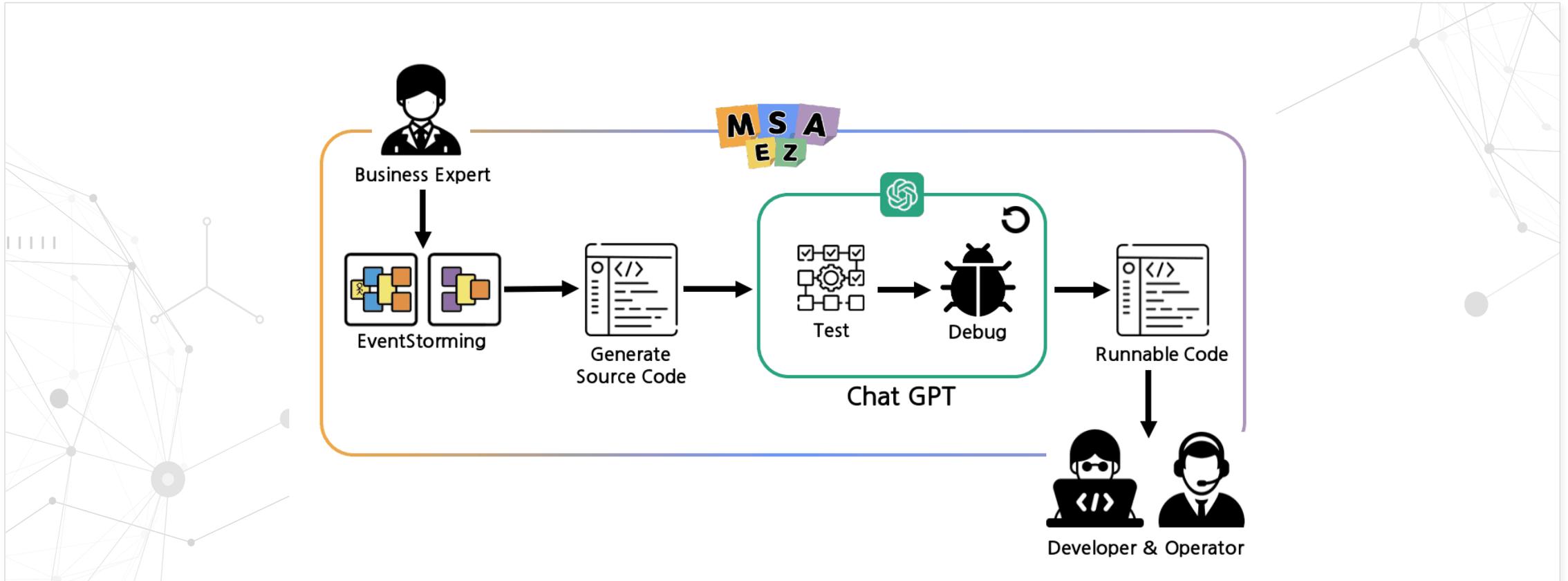
MSA Easy Value Chain & Tools



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*



M S A
E Z

01
—

Biz Phase

 **WEngine**



Identifies

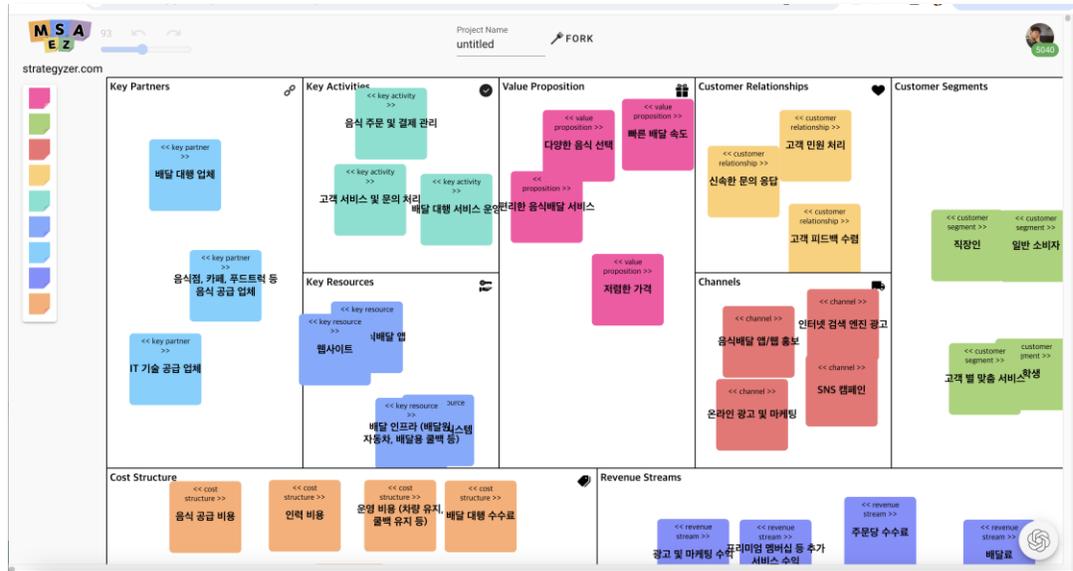
- Personas
- Pain-points for each actions and touchpoints by polling the emotions

Derives

- Possible Solutions
- Opportunities

Related to

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



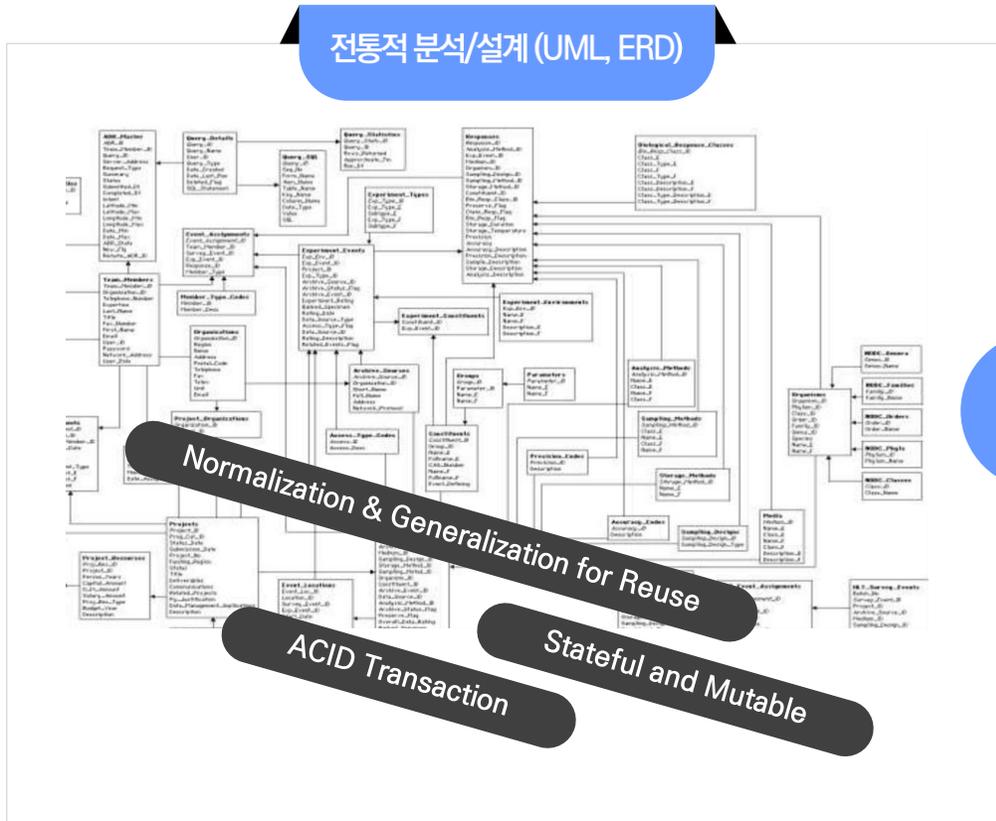
- Identifies
 - Value Propositions
 - Customer Segments
 - Customer Relationships / Channels
 - Key Activities / Key Resources
 - Key Partners
 - Cost Structure and Revenue Streams

- Derives
 - Business Model

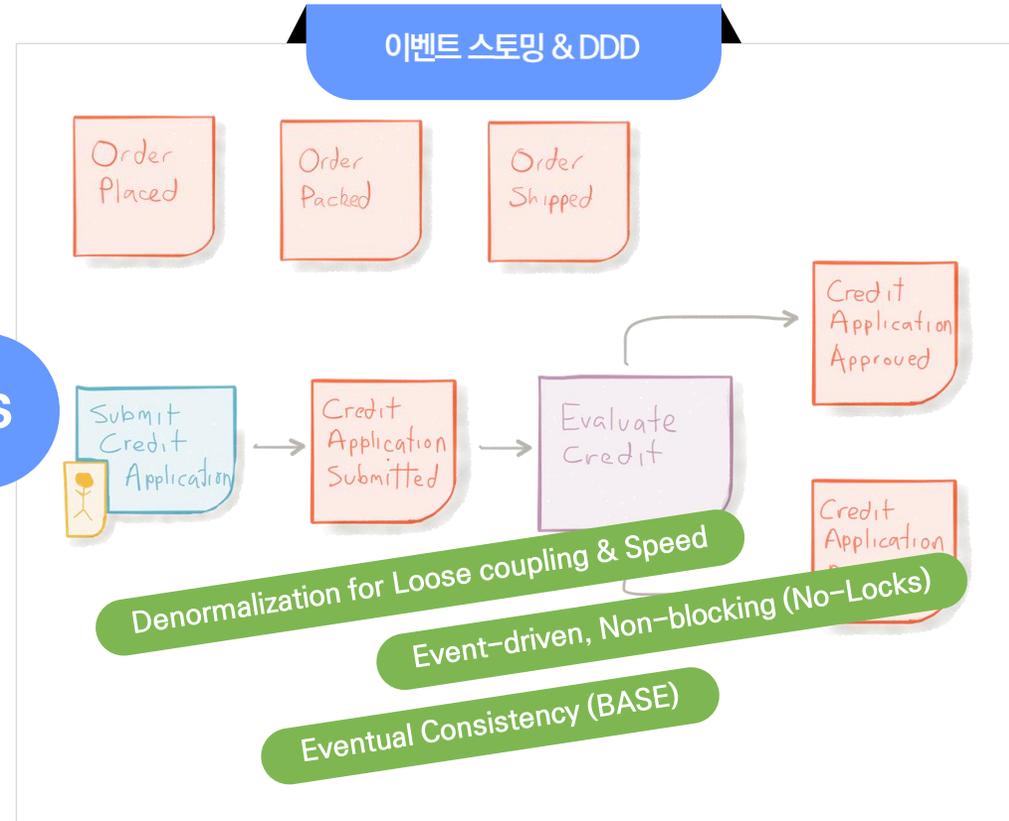
- Related to
 - ...

Event Storming : MSA 설계를 쉽게 하는 방법

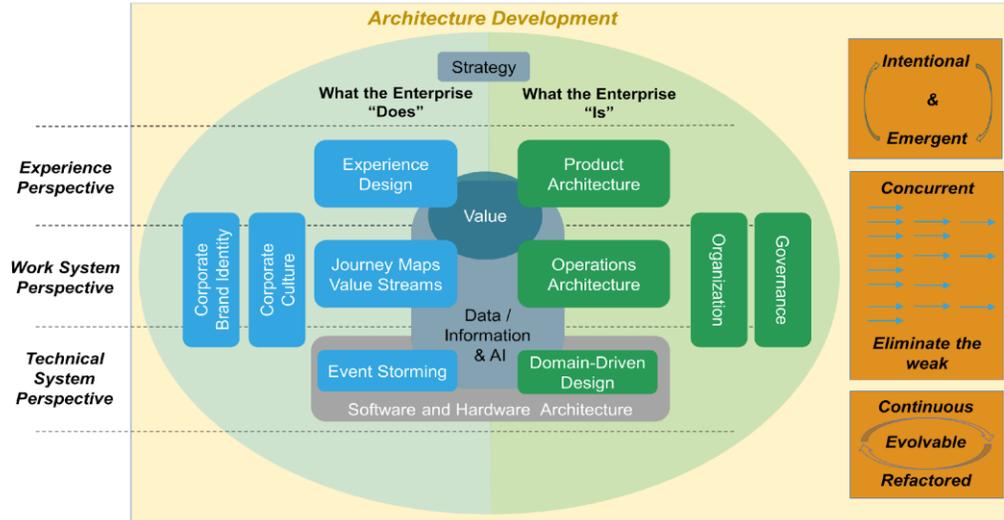
- 이벤트스토밍은 시스템에서 발생하는 이벤트를 중심 (Event-First) 으로 분석하는 기법으로 특히 Non-blocking, Event-driven 한 MSA 기반 시스템을 분석에서 개발까지 필요한 도메인에 대한 탁월하게 빠른 이해를 도모하는데 유리하다.
- 기존의 유즈케이스나 클래스 다이어그램 방식과 다르게 별다른 사전 훈련된 지식과 도구 없이 진행할 수 있다.
- 진행과정은 참여자 워크숍 방식의 방법론으로 결과는 스티키 노트를 벽에 붙힌 것으로 결과가 남으며, 오렌지색 스티키 노트들의 연결로 비즈니스 프로세스가 도출되며 이들을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



VS



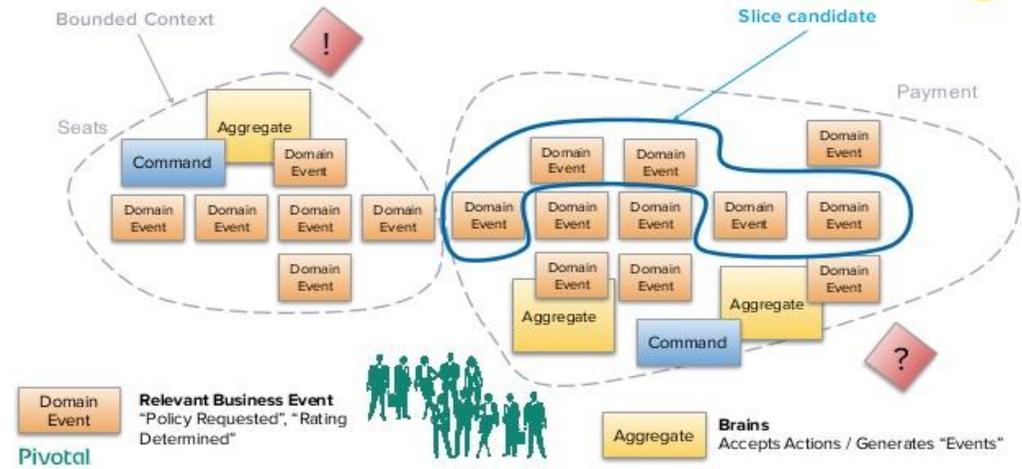
The Open Group의 OAA의 빌딩블록



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

Pivotal의 AppTX 방법론

Event Storming



<https://www.slideshare.net/Pivotal/pivotal-secret-sauce>

IBM의 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

Identifies

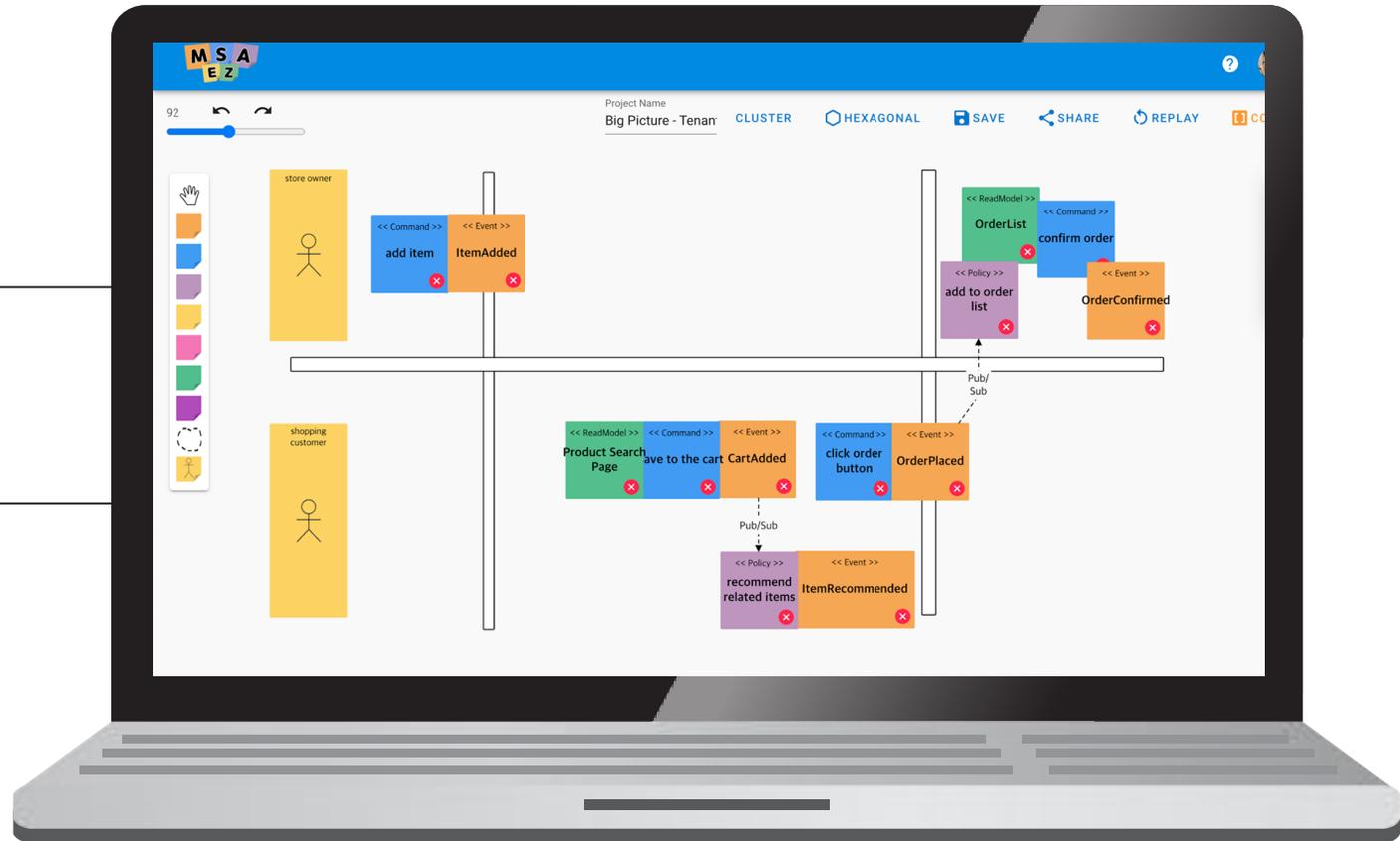
- Events
- Systems
- People (Actor/Personas)

Derives

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

Related to

- Process Modeling
- Software Design



Identifies

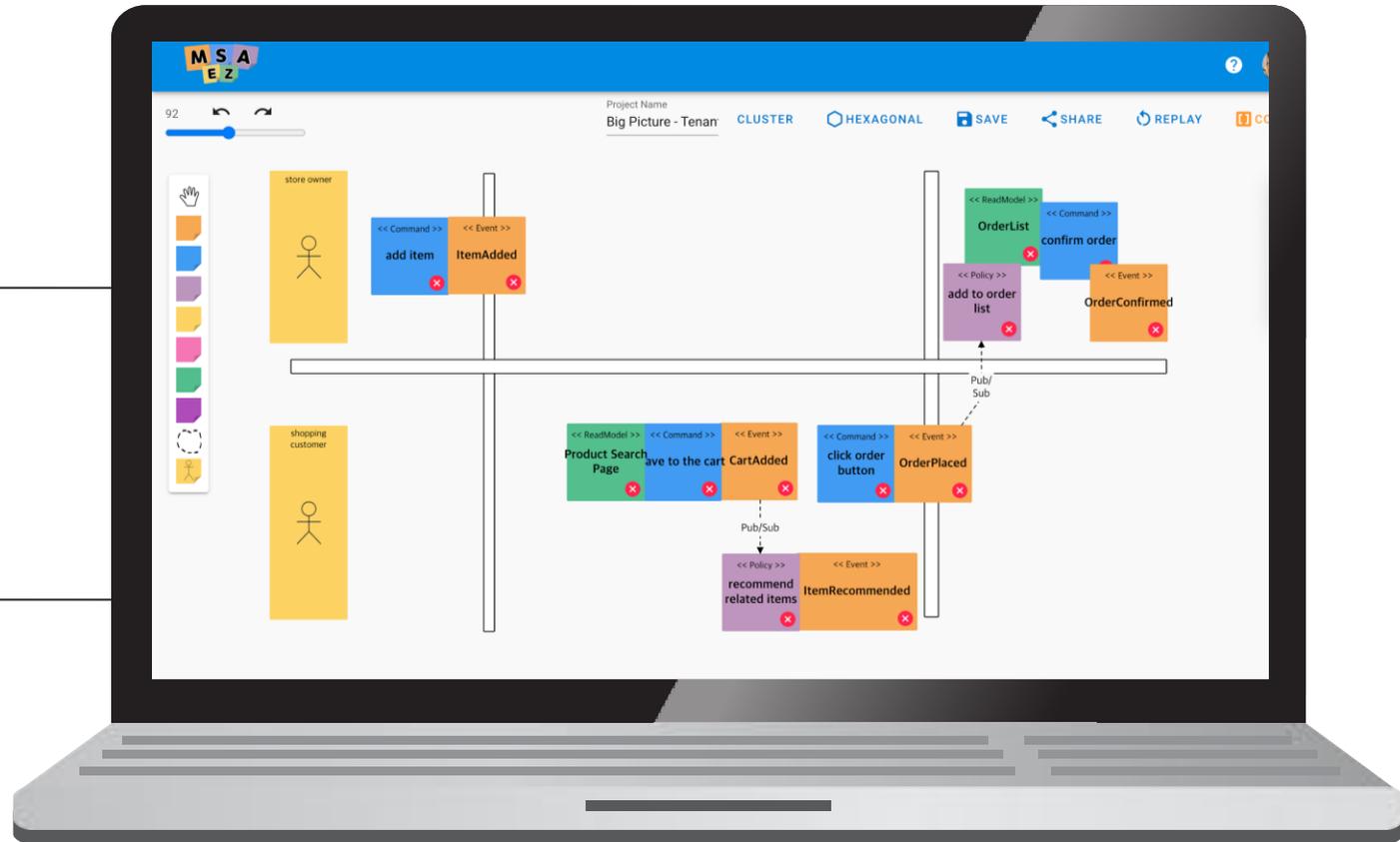
- Policies (Rules)
- Commands (APIs)
- Read Models (UIs)

Derives

- Value Propositions
- Policies
- Personas
- Individual Goals

Related to

- Software Design

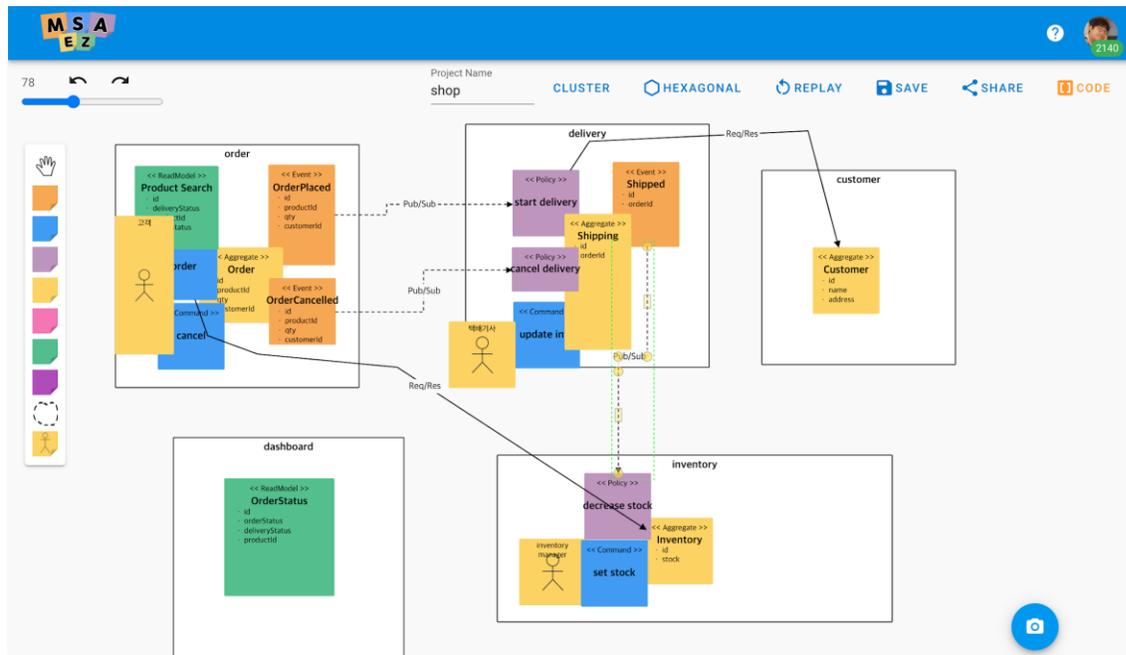


M S A
E Z

02
—

Dev Phase

 UEngine



Identifies

- Aggregates

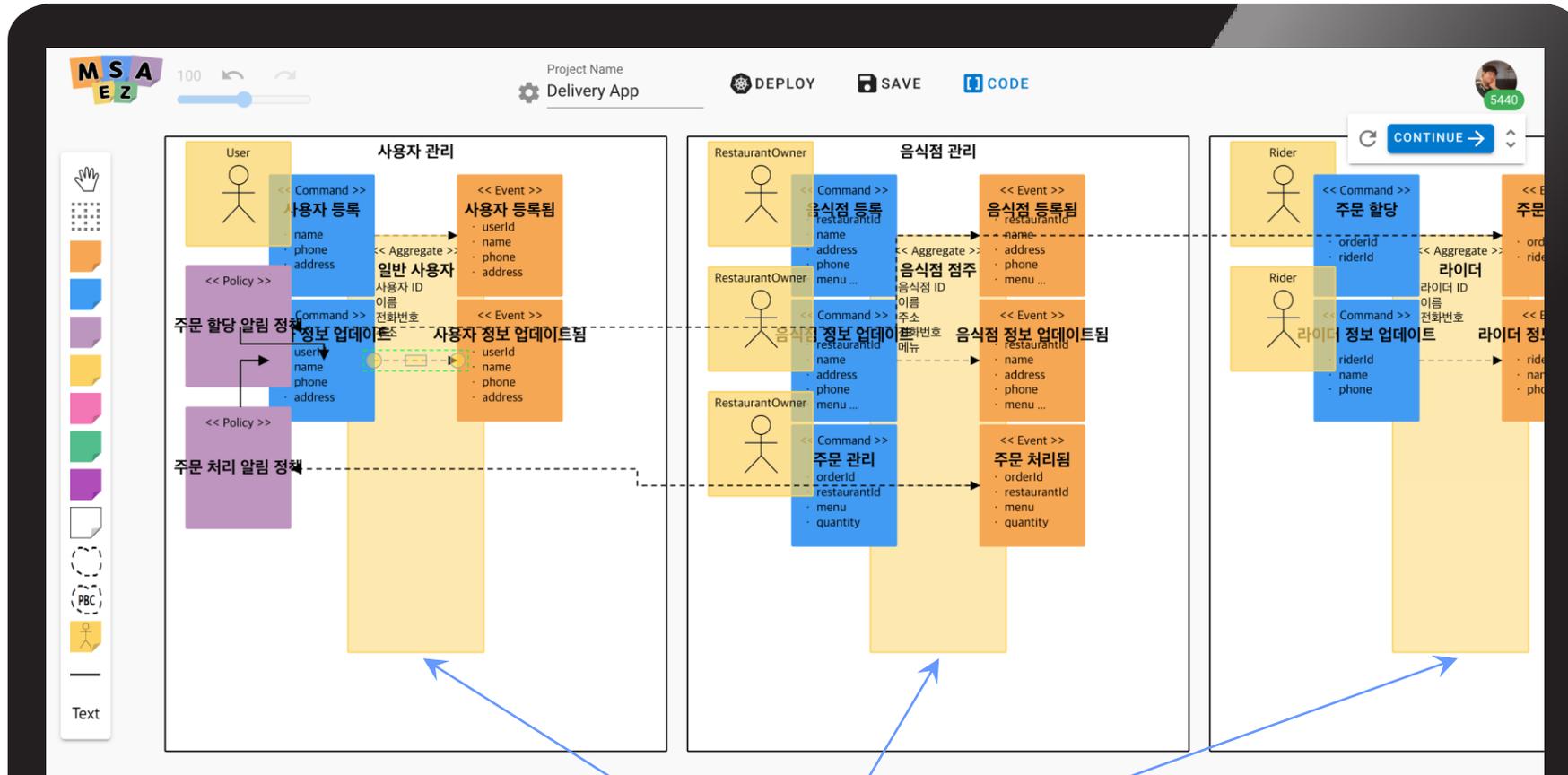
Derives

- Aggregates
- Policies Detail
- Read Model Detail

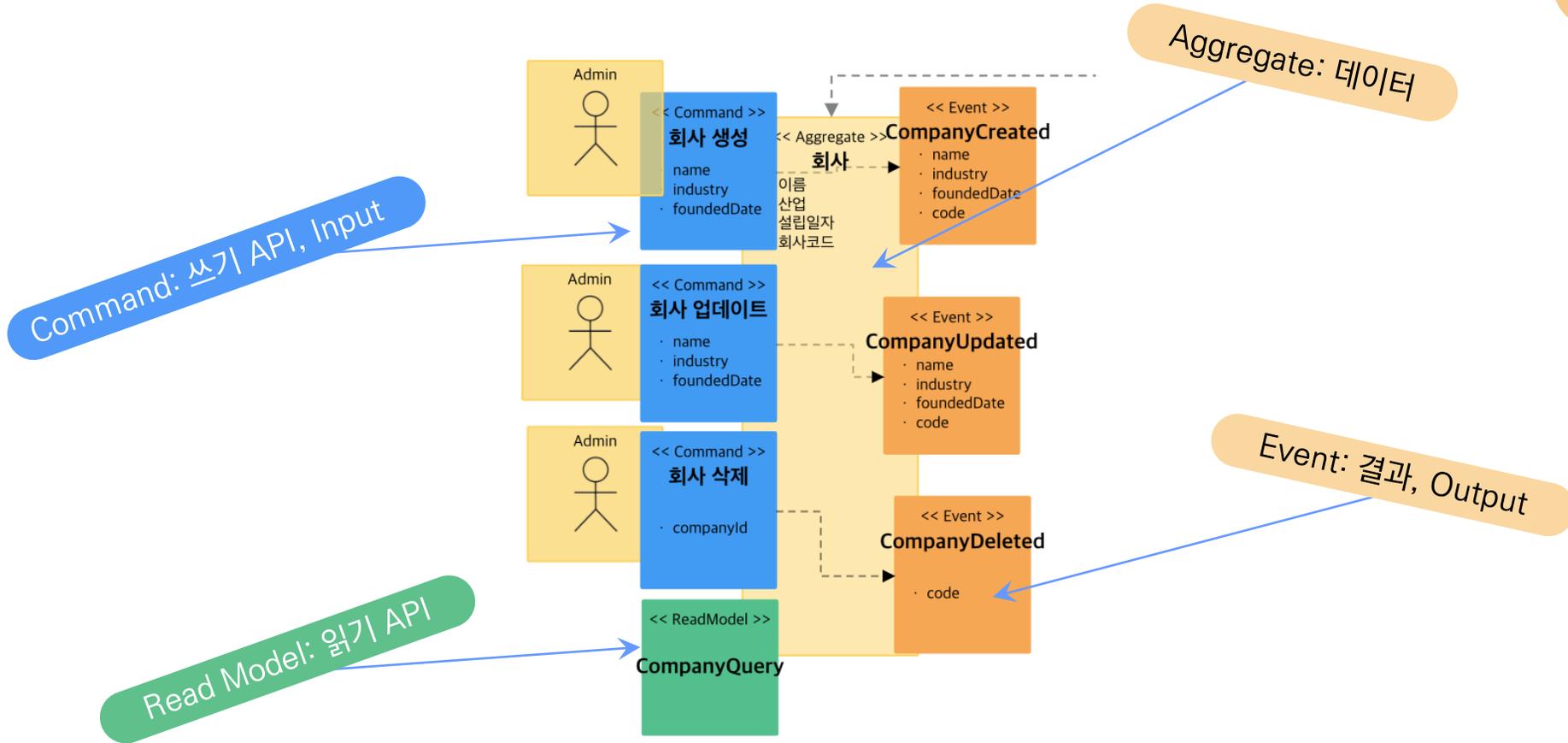
Related to

- Code Generation

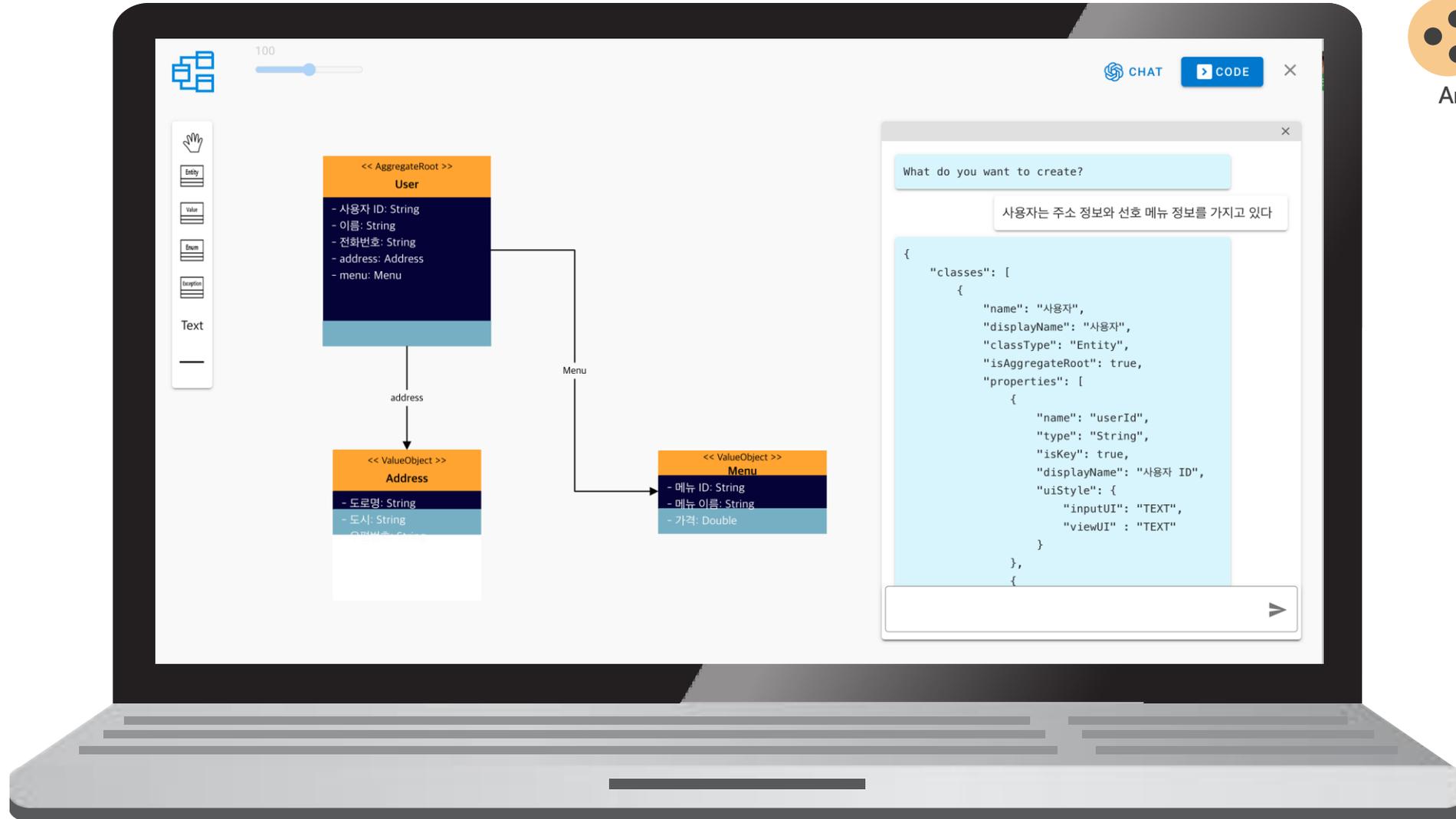
이벤트스토밍 모델 생성



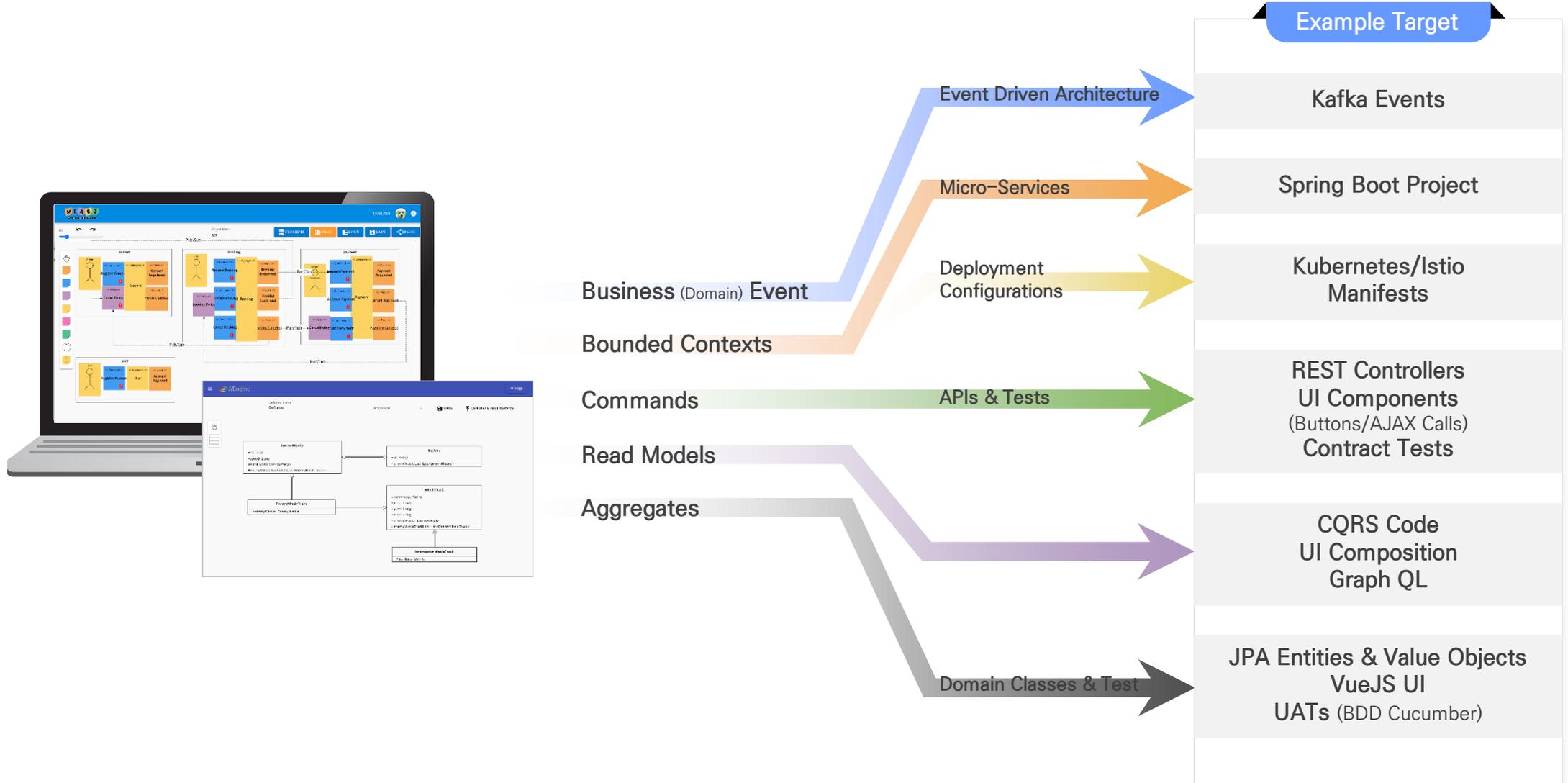
Bounded Contexts

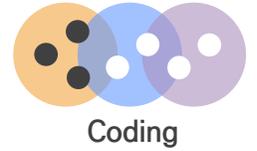


도메인 클래스 모델 생성



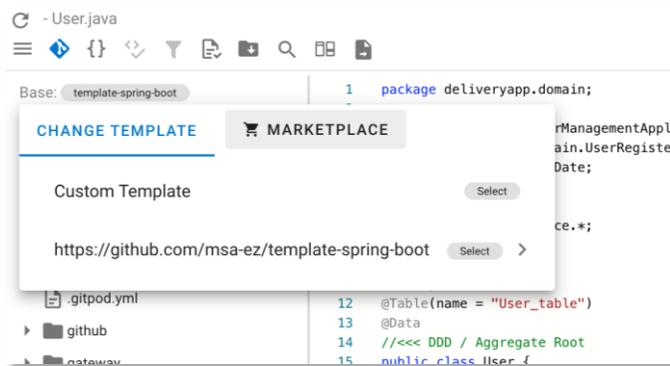
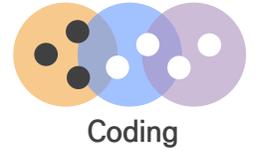
Model to Implementation





```
- User.java
TOPPINGS ▾
EXPLAIN CODE ►
1 package deliveryapp.domain;
2
3 import deliveryapp.UserManagementApplication;
4 import deliveryapp.domain.UserRegistered;
5 import java.time.LocalDate;
6 import java.util.Date;
7 import java.util.List;
8 import javax.persistence.*;
9 import lombok.Data;
10
11 @Entity
12 @Table(name = "User_table")
13 @Data
14 //<<< DDD / Aggregate Root
15 public class User {
16
17     @Id
18     private String userId;
19
20     private String name;
21
22     private String phone;
23
24     private Address address;
25
26     @PostPersist
27     public void onPostPersist() {
28         UserRegistered userRegistered = new UserRegistered(this);
29         userRegistered.publishAfterCommit();
30     }
31
32     @PrePersist
33     public void onPrePersist() {}
34
35     public static UserRepository repository() {
36         UserRepository userRepository = UserManagementApplication.applicationCon
37         UserRepository.class
38     }
39 }
```

마켓플레이스 - 템플릿의 선택



Marketplace

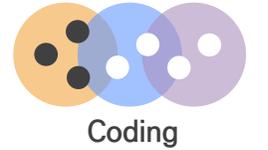
TEMPLATES

TOPPINGS

- All
- Java
- Python
- Go
- JavaScript
- Spring boot

Template Name	Language	Rating
GPT Engineer for Spring Boot	Java	★★★★★
Spring boot 3	Java	★★★★★
Spring boot monolith	Java	★★★★★
Axon	Java	★★★★★
Eventuate	Java	★★★★★
Python	Python	★★★★★
Spring boot clean	Java	★★★★★
Spring boot mybatis	Java	★★★★★
Go	Go	★★★★★
Nodejs	JavaScript	★★★★★
Spring boot	Java	★★★★★

GPT-Engineer 템플릿 선택

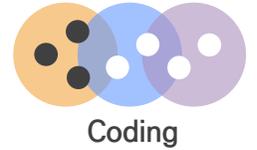


The screenshot displays the MSA-Easy GPT-Engineer interface. On the left, a panel titled "GPT Engineer for Spring Boot" features a robot icon and instructions: "Let the GPT engine (AI) generates the clean code in Spring Boot 3". It includes a five-star rating and tags for "Java" and "Spring boot", with an "APPLY" button. The main area shows a code generation window with the following content:

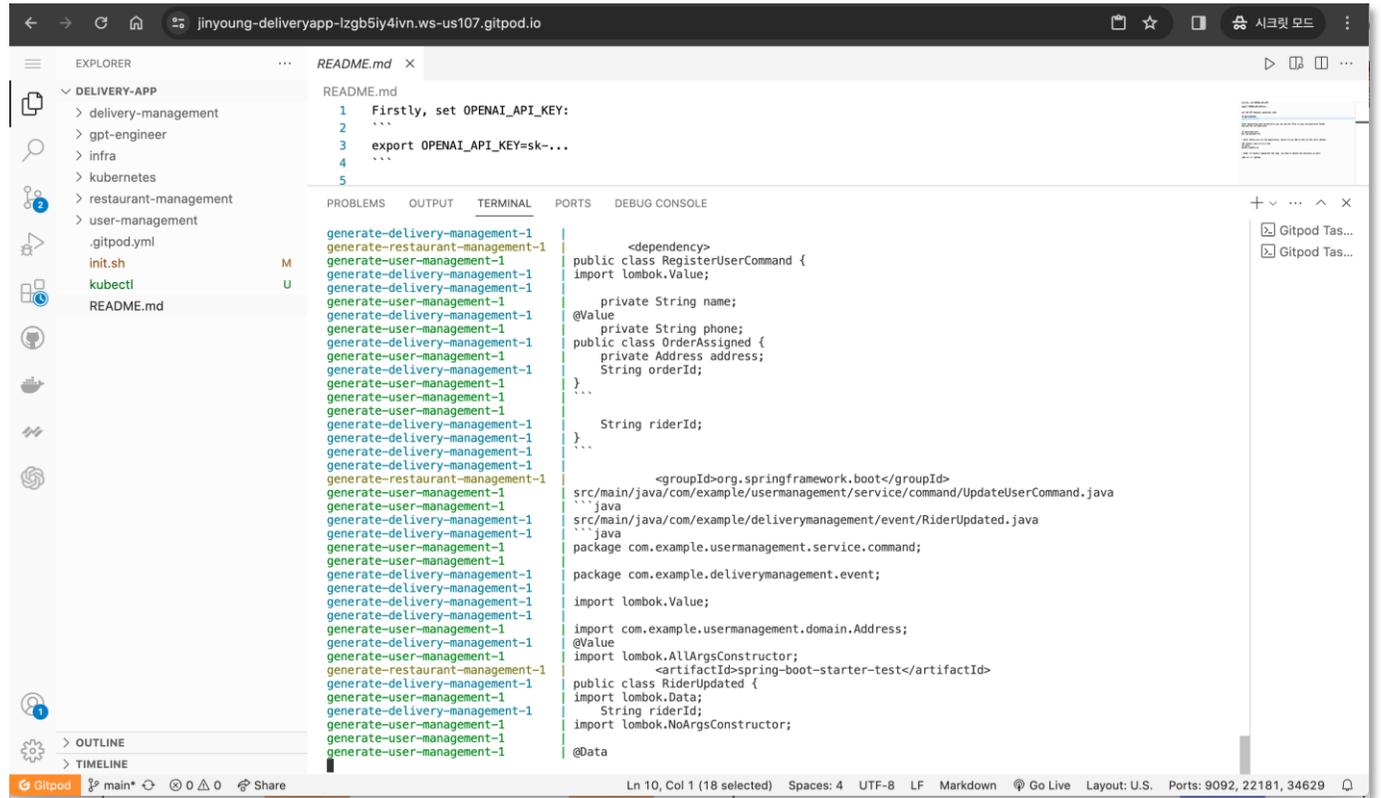
```
Firstly, set OPENAI_API_KEY:  
  
export OPENAI_API_KEY=sk-...  
  
Let the GPT Engineer generates code:  
  
cd gpt-engineer  
docker-compose up  
  
After generating code succes  
And you can run them with:  
  
cd {microservice}  
mvn spring-boot:run  
  
Note: Before you run the ap  
  
sdk install java 17.0  
cd infra  
docker-compose up  
  
Node: To totally regenerate
```

Below the code is a vertical toolbar with icons for a hand, a grid, a document, a PBC icon, a person icon, and a "Text" button. The right side of the interface shows a code editor with a "prompt" file containing the following text:

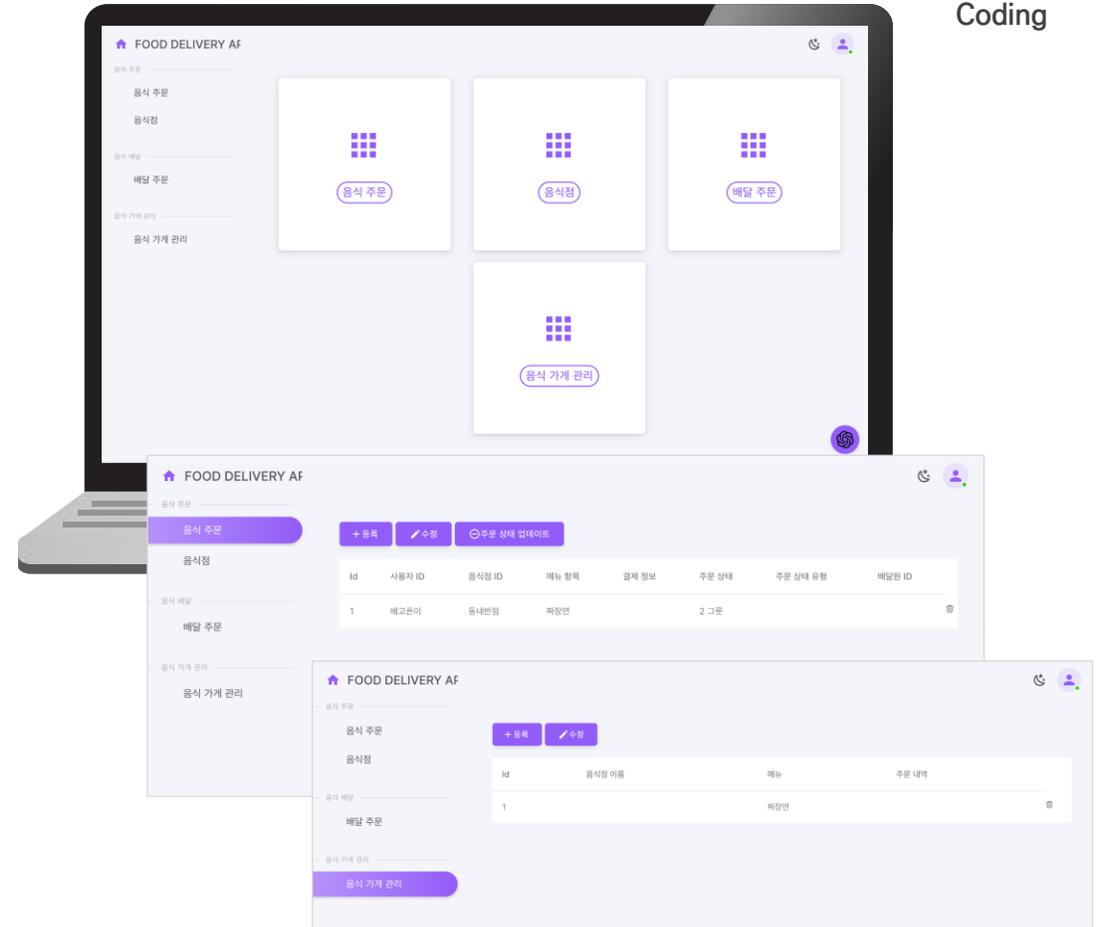
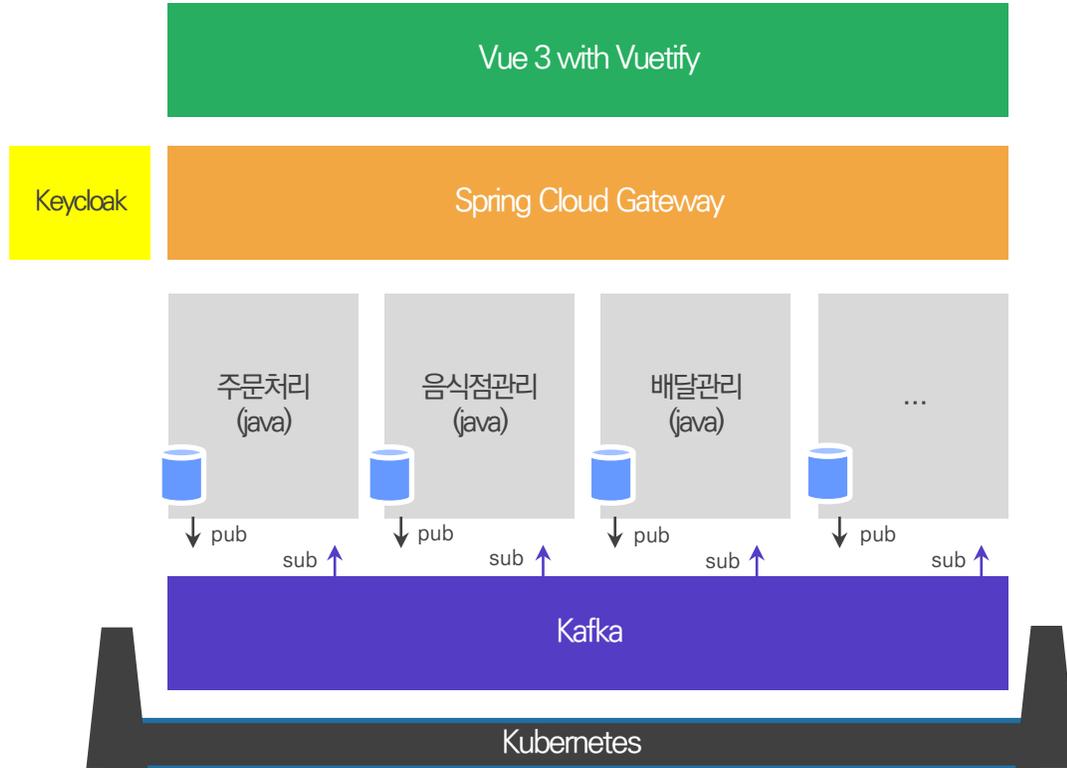
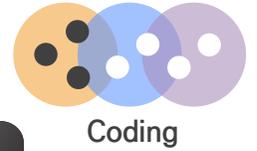
```
- prompt  
Base: gpt-engineer  
Source Tree  
prompt  
1 Please implement for this Event Storming Model:  
2  
3 {"_type":"BoundedContext","name":"order","aggregates":[{"_type":"Aggregate","name":"Order","aggregateRoot":{"_type":"AggregateRoot","field  
4  
5  
6 I want to utilize following techniques:  
7  
8 - clean architecture  
9 - spring-boot3  
10 - spring-data-rest  
11 - JPA (must use package jakarta.persistence.* instead of javax.persistence.*)  
12 - H2 DB  
13 - Lombok  
14 - Kafka with spring-cloud-streams library  
15 - JUnit Test  
16  
17  
18 SELF-CRITICISM:  
19 - when you create pom.xml, USE correct parent project to spring boot 3  
20 - when you create pom.xml, you MUST specify the version: 3.1.3 ONLY for the spring-cloud-streams  
21 - Please generate start from pom.xml  
22  
23  
24 when you create execution shell (run.sh), please add '--no-transfer-progress' option not to output transfer progress if there's mvn comman  
25  
26  
27 mvn [command] --no-transfer-progress  
28  
29  
30
```



```
docker-compose.yml X
gpt-engineer > docker-compose.yml > {} services > {} generate-user-manage
docker-compose.yml - The Compose specification establishes a standard for the defini
1  version: "3"
2
3  services:
4
5    generate-user-management:
6      tty: true
7      stdin_open: true
8      environment:
9        - OPENAI_API_KEY=${OPENAI_API_KEY}
10     image: ghcr.io/jinyoung/gpt-engineer:v5
11     # command: ["-sh"]
12     volumes:
13       - ../user-management:/project
14
15     generate-restaurant-management:
16       tty: true
17       stdin_open: true
18       environment:
19         - OPENAI_API_KEY=${OPENAI_API_KEY}
20     image: ghcr.io/jinyoung/gpt-engineer:v5
21     # command: ["-sh"]
22     volumes:
23       - ../restaurant-management:/project
24
25     generate-delivery-management:
26       tty: true
27       stdin_open: true
28       environment:
29         - OPENAI_API_KEY=${OPENAI_API_KEY}
30     image: ghcr.io/jinyoung/gpt-engineer:v5
31     # command: ["-sh"]
```



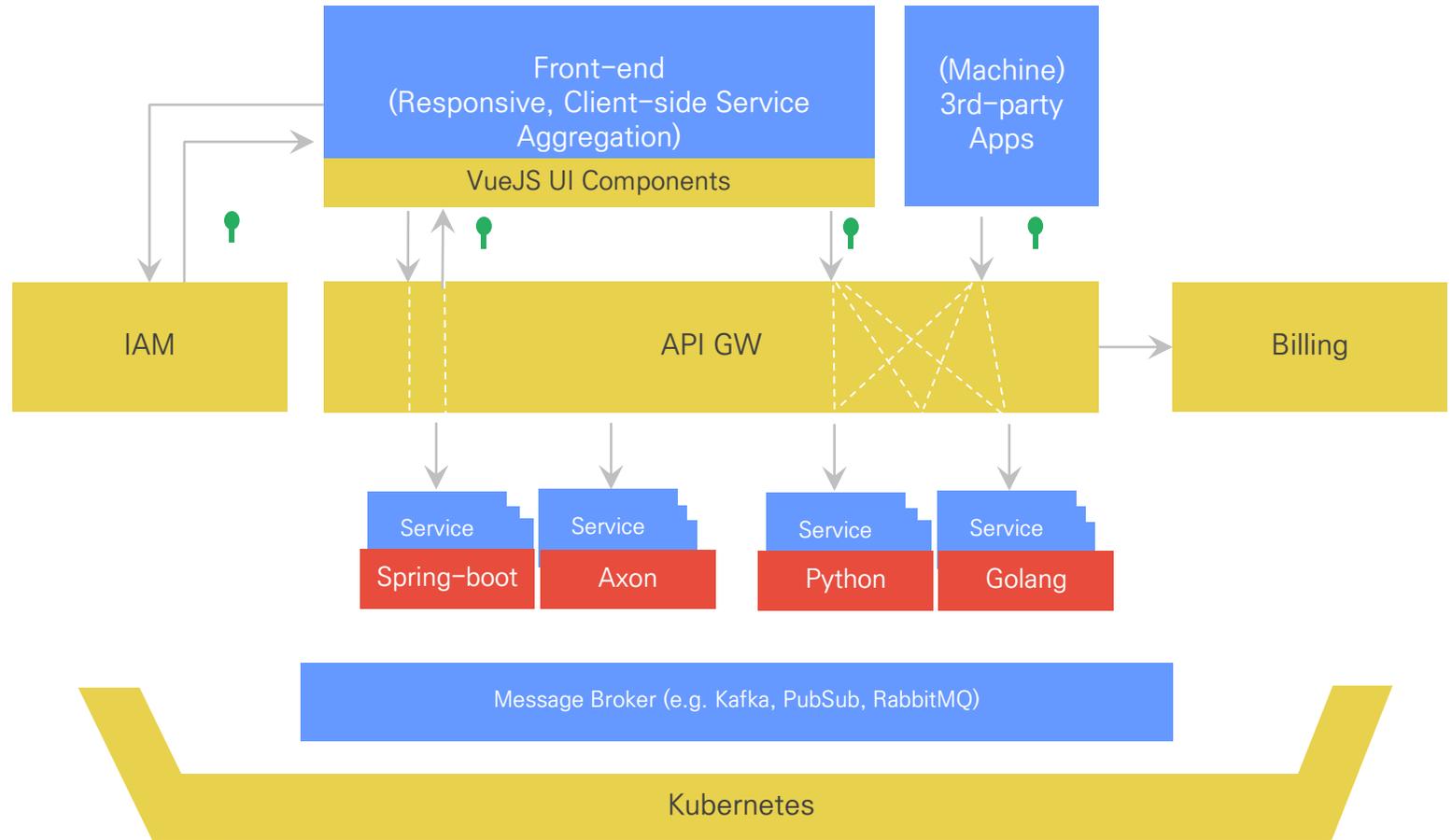
각 마이크로서비스별 에이전트가 병렬로 코드를 생성



Generated Inner/Outer Architecture

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

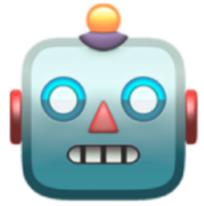


Marketplace

TEMPLATES

TOPPINGS

- All
- Java
- Python
- Go
- JavaScript
- Spring boot

 GPT Engineer f or Spring Boot ★★★★★	 Spring boot 3 ★★★★★	 Spring boot m onolith ★★★★★	 Axon ★★★★★	 Eventuate ★★★★★	 Python ★★★★★
 Spring boot cl ean	 Spring boot m ybatis	 Go ★★★★★	 Nodejs ★★★★★	 Spring boot ★★★★★	

Marketplace

TEMPLATES

TOPPINGS

- All
- Multitenancy
- Kubernetes
- Security
- Service Mesh
- DevOps
- Data Projection
- Frontend

eGov Default App
★★★★★

GPT Engineer for Spring Boot
★★★★★

Materio
★★★★★

Micro Wijmo
★★★★★

Unit Test for Microservices
★★★★★

Microfrontend with Single-SPA + VueJS
★★★★★

Local Microservice Development Dependencies
★★★★★

Spring boot multitenancy schema
★★★★★

Spring boot multitenancy partitioned
★★★★★

Github action
★★★★★

Wijmo
★★★★★

IsVanillaK8s
★★★★★

Istio
★★★★★

Ingress
★★★★★

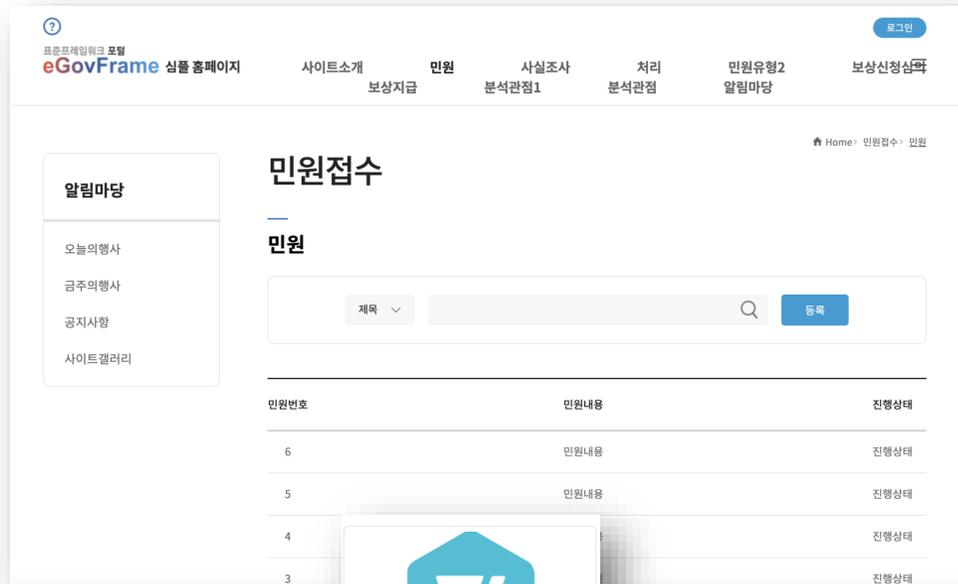
Argo
★★★★★

Spring security
★★★★★

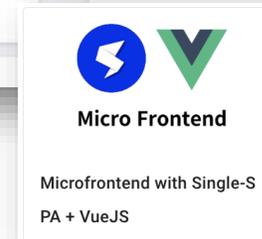
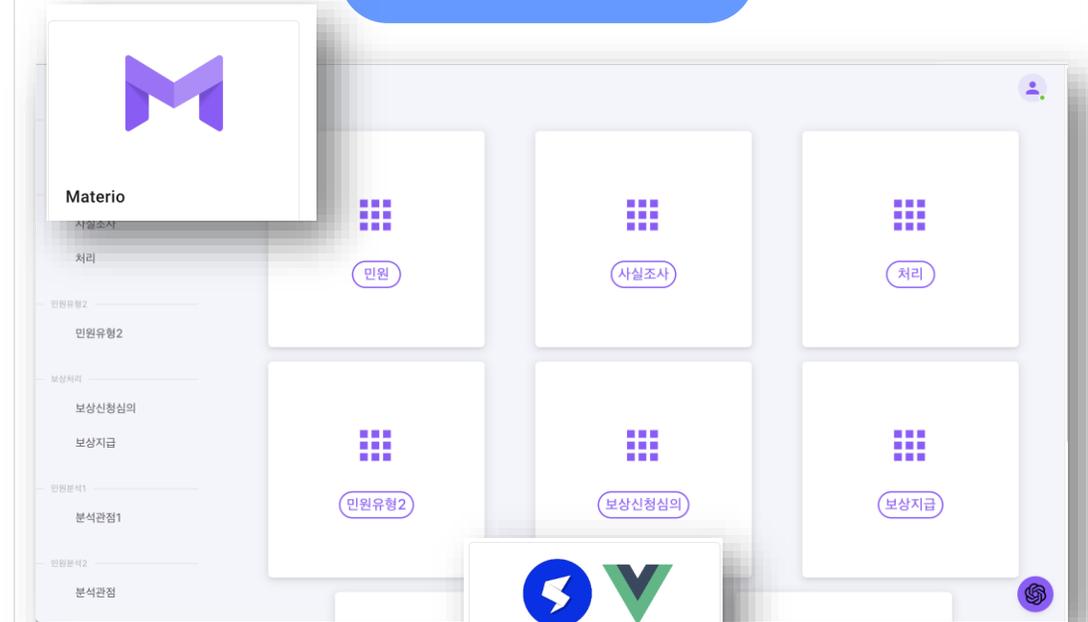
Apollo graphql
★★★★★

Keycloak security
★★★★★

React



Vue3+Vuetify+LLM



✓ 테스트



Microservices

Local Microservice Development Dependencies

★★★★★

✓ API 문서화



Unit Testina

Unit Test for Microservices

★★★★★

✓ 쿠버네티스



IsVanillaK8s

★★★★★

✓ 서비스 메시



Istio

★★★★★

✓ 깃 오피스



Argo

★★★★★



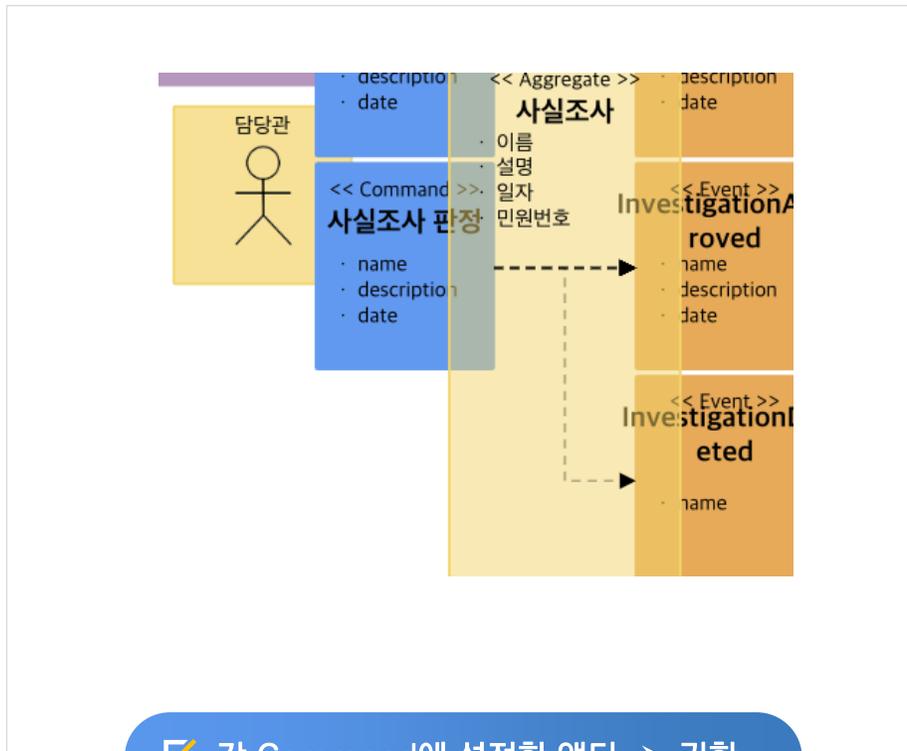
Ingress

★★★★★



Github action

★★★★★



✓ 각 Command에 설정한 액터 → 권한

✓ 스프링 시큐리티

Spring security
★★★★★

✓ 키클락

Keycloak security
★★★★★

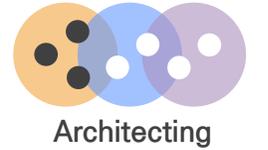
✓ Frontend와 Backend의 보안설정 (@AllowedScopes)으로 코드 생성

☑ MSA Ez는 전자정부 표준 프레임워크의 호환성 확인 가이드라인을 준수합니다.

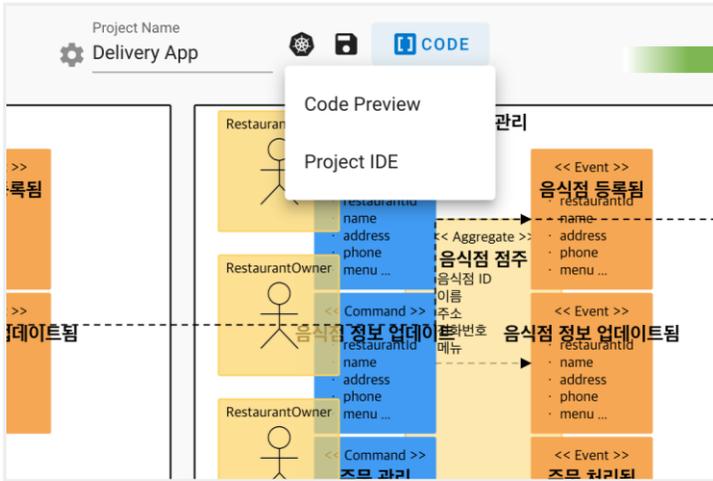
전자정부 표준프레임워크
eGovFrame
Open Source Open Platform



- ☑ 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.



☑ 도메인 모델



☑ 표준 프레임 토픽 적용

Java/Spring Version MARKETPLACE

JAVA 8 JAVA 15

Kubernetes

- Vanilla Kubernetes
- OAuth by Spring Security + Spring GW
- OAuth by Keycloak + Spring GW

Service Mesh

- Istio
- Ingress

DevOps

- Argo + Istio

Data Projection

- Apollo GraphQL
- JAVA GraphQL

Custom Toppings

- egov-framework ×
- egov-default ×
- github-action ×

CUSTOM TOPPING

eGov Framework Toppings

☑ 코드 자동 생성

```
- OrderServiceImpl.java
package small.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 small.domain.CancelCommand;
import small.domain.Order;
import small.domain.OrderCommand;
import small.domain.OrderRepository;
import small.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());
    }
}
```



eGov 토픽	설 명	비 고
Egov-framework	이벤트스토밍으로 설계된 도메인 모델에 대해 표준 프레임을 지원하는 Spring boot 기반의 클라우드 네이티브 코드를 자동 생성한다.	Spring-boot JPA
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/

✔ 프론트 엔드 생성 예시



표준프레임워크 포털 **eGovFrame** 심플 홈페이지

로그인

사이트소개 민원 사실조사 처리 민원유형2 보상신청삼학
보상지급 분석관점1 분석관점 알림마당

Home > 민원접수 > 민원

알림마당

- 오늘의행사
- 금주의행사
- 공지사항
- 사이트갤러리

민원접수

민원

제목 ▾

민원번호	민원내용	진행상태
6	민원내용	진행상태
5	민원내용	진행상태
4	민원내용	진행상태
3	민원내용	진행상태

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 Archicecture (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	<ul style="list-style-type: none"> GraphQL fetch RestAPI composition 	If you have many Javascript developers
Java GraphQL	Java	Spring Boot GraphQL	<ul style="list-style-type: none"> GraphQL fetch RestAPI composition 	If you have many Java Developers

Topping Name	Platform	Functionalities	Recommended For
Keycloak + Spring Gateway	<ul style="list-style-type: none"> Spring Gateway Auth provider: Keycloak 	OIDC / OAuth2 authn/authzn	If most of your microservices are written in Java and Spring.
Spring Security + Spring Gateway	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> 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

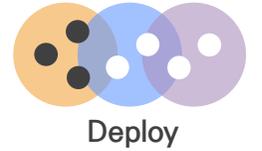
Topping Name	Platform	Functionalities	Recommended For
Ingress	Kubernetes	<ul style="list-style-type: none"> Traffic Management 	Simple Configuration
Istio	Kubernetes	<ul style="list-style-type: none"> Traffic Management Circuit Breaking / Rate Limiting Access Control Distributed Monitoring 	Advanced Resiliency, Dynamic Deployment, Monitoring
Argo CD + Rollout	Kubernetes	<ul style="list-style-type: none"> Canary Deployment GitOps 	Advanced Deployment

M S A
E Z

03
—

Ops Phase

 **WEngine**



The screenshot displays the MSA-Easy interface. The top part shows UML diagrams for '사용자 관리' (User Management) and '음식점 관리' (Restaurant Management). The bottom part shows a Kubernetes Service manifest editor with the following configuration:

```
1 apiVersion: "v1"
2 kind: "Service"
3 metadata:
4   name: "delivery-management"
5   labels:
6     app: "delivery-management"
7   annotations:
8     msaez.io/x: "551"
9   spec:
10    ports:
11      -
12        port: 8080
13        targetPort: 8080
14    selector:
15      app: "delivery-management"
16
```

✔ 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

- Namespace
- Deployment
- ReplicaSet
- Pod
- StatefulSet
- DaemonSet
- Horizontal Pod Autoscaler
- Service
- Ingress
- ConfigMap
- Secret
- PersistentVolume
- PersistentVolumeClaim
- StorageClass
- Job
- CronJob
- Role
- RoleBinding
- ClusterRole
- ClusterRoleBinding
- ServiceAccount
- Service
- Gateway
- VirtualService
- DestinationRule
- ServiceEntry
- Sidecar
- Quota
- Rule
- QuotaSpec
- QuotaSpecBinding
- MemQuota
- Application
- Workflow
- Workflow - Steps
- Workflow - Dag
- Rollout
- EventSource
- Sensor
- WorkflowTemplate

The screenshot displays the K8S-Easy web interface. At the top, there's a navigation bar with 'K8S-Easy' and 'Kubernetes Easy' logos. Below it, a sidebar contains icons for Service, Ingress, and other resources. The main area shows a deployment diagram for an 'api-gateway' service. The diagram illustrates the flow from the gateway to three services: 'order-svc', 'product-svc', and 'delivery-svc'. Each service is linked to a corresponding 'Deployment' (order, product, delivery), which shows a status of '3/3' (3 pods out of 3 desired). Below the diagram, a terminal window shows the command 'kubectl get pod -n default' and its output, listing various pods in a 'Running' state.

```
nobody@webkubect1-574768c64b-6e7p1:~$ kubectl get pod -n default
NAME                                READY   STATUS    RESTARTS   AGE
delivery-b59d8b575-d21nh            1/1    Running   0          75s
delivery-b59d8b575-dszk2            1/1    Running   0          75s
delivery-b59d8b575-fw924            1/1    Running   0          75s
order-7cbf64887-94fxg               1/1    Running   0          75s
order-7cbf64887-9qjqp              1/1    Running   0          75s
order-7cbf64887-cz722              1/1    Running   0          75s
product-5d4788fd74-2hxpj            1/1    Running   0          75s
product-5d4788fd74-69pg1            1/1    Running   0          75s
product-5d4788fd74-lhddn            1/1    Running   0          75s
nobody@webkubect1-574768c64b-6e7p1:~$
```

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, 새도우 디플로이 모델의 설정과 Scheduling을 통한 디플로이
- ArgoCD와의 연동을 통한 통합 모니터링

M S A
E Z

04
—

Values

 **WEngine**

- ✓ 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

