클라우드 네이티브 정보시스템 구축을 위한
발주자 안내서

NIA 한국지능정보사회진흥원

서울특별시 중구 청계천로 14 (04520)
대표전화 053-230-1114

클라우드 네이티브 정보시스템 구축을 위한

발주자 안내서

행정안전부

NIA 한국지능정보사회진흥원

본 안내서는 **클라우드 네이티브 정보시스템 구축 사업을 계획하는 발주자와 수행하는 개발자**에게 클라우드 네이티브에 대한 이해도를 높이고, 적극적인 활용을 지원하고자 발간하게 되었습니다.

일러두기

- **안내서를 왜 발간하게 되었는가?**

4차 산업혁명 시대를 맞아 국민의 일상에 디지털 전환이 가속화되고 있고, 이에 적합한 대국민 서비스 제공을 위해 공공서비스의 클라우드 전환이 추진되고 있습니다.

클라우드 전환 및 도입 효과를 높이기 위해 단순한 기술 인프라 위주의 클라우드 도입보다 클라우드 환경에 최적화된 새로운 형태의 클라우드 네이티브 정보시스템 구축이 필요합니다. 즉, 기존의 크고, 단일한 서비스 구조를 마이크로서비스 아키텍처로 구현하여 개발, 배포, 운영함으로써 빠르고 안정적인 대국민 서비스를 제공할 수 있습니다.

본 안내서는 공공부문 클라우드 네이티브 관련 정보화 사업에 참여하는 발주자와 개발자에게 관련 기술 정보와 활용 방안 등을 제공하여 성공적인 사업 추진을 지원하고자 발간되었습니다.

- **누가 안내서를 읽어야 할까?**

클라우드 관련 정보화 사업을 준비하는 중앙행정기관, 지방자치단체, 공공기관 등 발주자와 클라우드 네이티브 정보시스템 구축 및 운영 사업에 참여하거나 관심이 있는 개발자입니다.

- **언제 안내서를 활용할까?**

발주 기관에서 클라우드 기반 정보화 사업을 기획하고 발주하기 전에 발주자 안내서를 통해 클라우드 네이티브의 개념, 주요 기술 등을 이해하고 도입 적합성을 검토할 수 있습니다.

공공·민간 부문 클라우드 네이티브 정보시스템 구축을 위해 주요 기술과 구현·운영 방안을 학습하고자 하는 개발자들은 항시 개발자 안내서를 참고할 수 있습니다.

- **안내서는 어떻게 구성되나?**

발주자를 위한

PART I - 발주자 안내서

클라우드 네이티브 개요, 도입 필요성, 구성요소 및 원칙, 도입 적합성 검토, 구축사업 추진 시 고려사항으로 구성

정보시스템 개발자를 위한

PART II - 개발자 안내서

클라우드 네이티브 정보시스템 구축 절차와 구축 단계별 개발 방안으로 구성

목차

PART I 발주자 안내서

01 클라우드 네이티브 개요

02 클라우드 네이티브 도입 필요성

목차

PART I 발주자 안내서

03 클라우드 네이티브 구성요소 및 원칙

목차

PART I 발주자 안내서

03 클라우드 네이티브 구성요소 및 원칙

04 클라우드 네이티브 적합성 검토

목차

PART I 발주자 안내서

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

목차

PART II 개발자 안내서

06 클라우드 네이티브 정보시스템 개발 절차

07 클라우드 네이티브 정보시스템 분석 단계

08 클라우드 네이티브 정보시스템 설계 단계

목차

PART II 개발자 안내서

09 클라우드 네이티브 정보시스템 구현·운영 단계

목차

PART II 개발자 안내서

PART I

발주자 안내서

클라우드 네이티브 정보시스템 구축을 위한

발주자 안내서

01

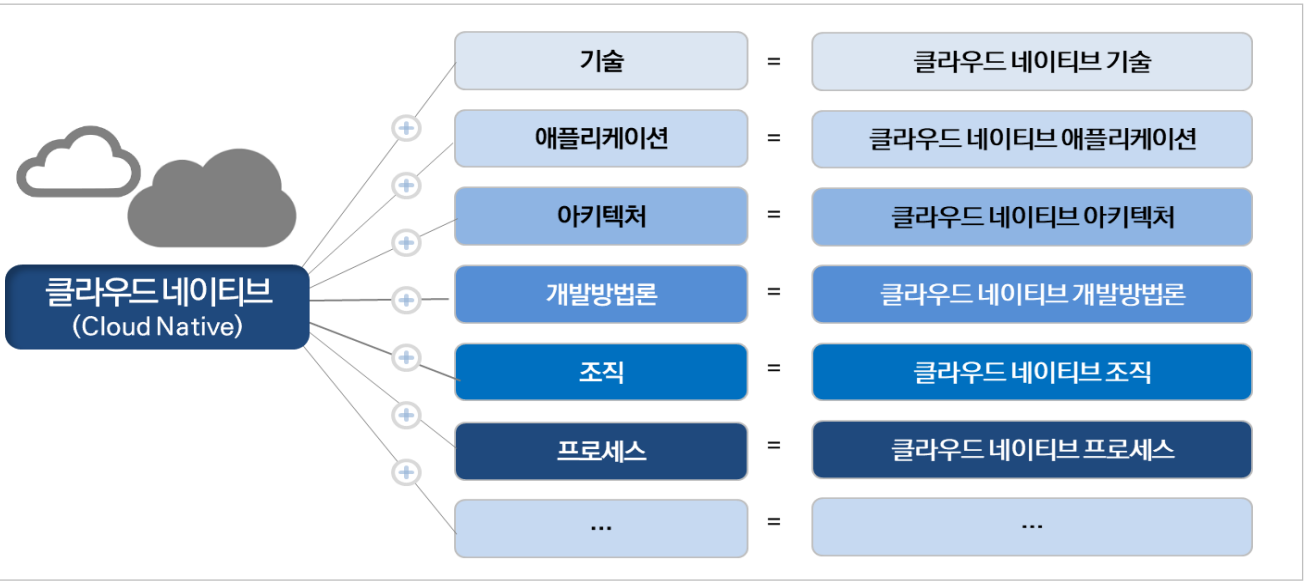
클라우드 네이티브 개요

01 클라우드 네이티브 개요

1.1 클라우드 네이티브 정의

- 클라우드 네이티브(Cloud Native)는 “클라우드의 장점을 최대한 활용하여 정보시스템을 구축 및 실행하는 환경”을 말한다
- 클라우드 네이티브는 기술, 애플리케이션, 아키텍처, 개발방법론, 조직, 프로세스 등 다양한 용어와 결합하여 다양한 의미로 사용되고 있다.

[그림 1-1] 클라우드 네이티브 용어의 다양한 사용



- 2015년 최초로 클라우드 네이티브라는 용어를 사용한 리눅스는 CNCF(Cloud Native Computing Foundation) 재단을 설립하여 클라우드 네이티브 관련 기술을 정의하고 오픈소스를 관리하고 있다. 이 재단에는 550개가 넘는 클라우드 공급자와 기술 기업들이 참여하고 있으며, 클라우드 네이티브에 대해 다음과 같이 정의하고 있다.

[그림 1-2] CNCF의 클라우드 네이티브 정의 V1.0

CLOUD NATIVE COMPUTING FOUNDATION

CNCF의
클라우드 네이티브 정의 V1.0

- 퍼블릭(Public), 프라이빗(Private), 하이브리드(Hybrid) 클라우드 환경에서 확장성 있는 애플리케이션을 만들고 운영할 수 있다.
- 컨테이너(Container), 서비스 메시(Service Mesh), 마이크로서비스(Microservice), 불변의 인프라스트럭처(Infrastructure), 그리고 선언적 API(Application Programming Interface)가 전형적인 접근 방식에 해당한다.
- 회복성이 있고, 관리 편의성을 제공하며, 가시성을 갖는 느슨하게 결합된 시스템을 사용할 수 있다.
- 견고한 자동화와 함께 사용하면, 엔지니어는 최소한의 수고로 영향력이 크고 예측 가능한 변경을 할 수 있다.

[출처 : https://github.com/cncf/toc/blob/master/DEFINITION.md]

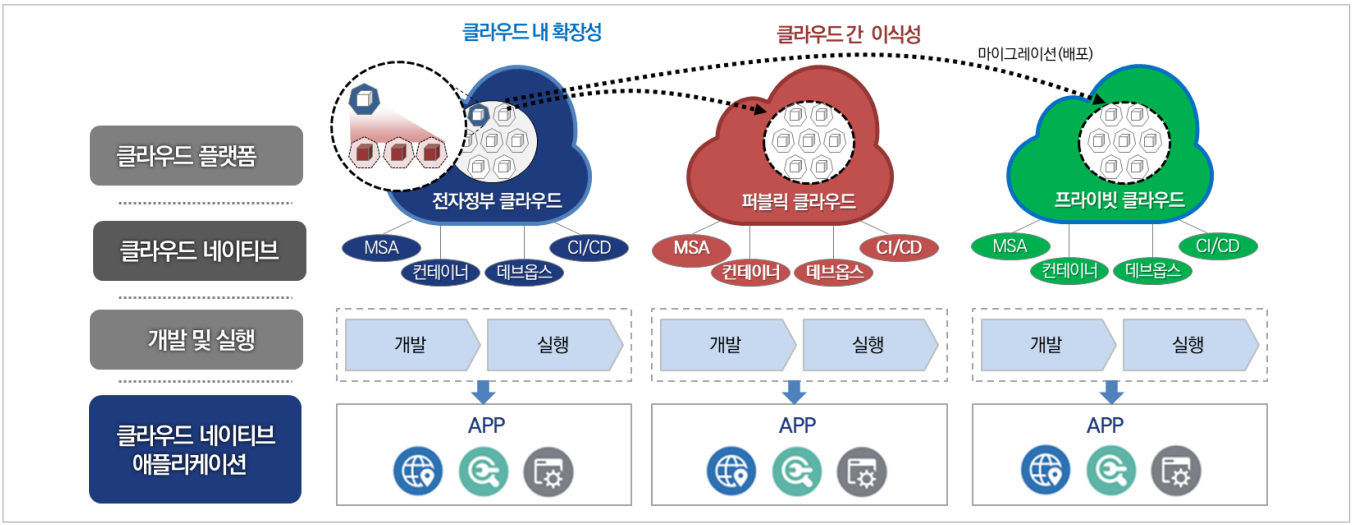
01 클라우드 네이티브 개요

1.2 클라우드 네이티브 애플리케이션

1.2.1 클라우드 네이티브 애플리케이션 정의

- 클라우드 네이티브 애플리케이션은 클라우드 네이티브 환경에서 개발, 실행되는 애플리케이션을 말한다.
- 클라우드 네이티브 애플리케이션은 프라이빗, 퍼블릭 및 하이브리드 클라우드 환경 전체에 지속적인 개발과 자동화된 관리 환경을 제공하기 위해 설계된 애플리케이션이므로 클라우드 내에서 확장이 가능하고, 어떤 클라우드에서도 이식이 가능하다

[그림 1-3] 클라우드 네이티브 애플리케이션



- 기존 애플리케이션은 장기간에 걸쳐 긴밀하게 결합된 모놀리식(Monolithic, 단일한) 구조로 물리 서버 기반 위에서 동작하는 반면, 클라우드 네이티브 애플리케이션은 소규모 서비스 단위의 마이크로서비스로 구성되며 가상 컨테이너 환경에서 동작되도록 설계되고 구현되었다.

[표 1-1] 기존 애플리케이션과 클라우드 네이티브 애플리케이션의 차이점

구분	기존 애플리케이션	클라우드 네이티브 애플리케이션
애플리케리션 구조	모놀리식 구조	마이크로서비스
결합	크고, 조밀한 결합	느슨한, 서비스 기반
실행환경	물리서버 중심	가상 컨테이너 중심
확장	수직 확장(Scale-Up)	수평 확장(Scale-Out)
인프라 의존성	인프라 의존	인프라 독립, 이식성 보장
개발방법	폭포수(Waterfall)	애자일(Agile, 민첩성)
빌드·배포	수작업, 긴 시간	CI/CD[1] 자동화, 짧은 시간 & 지속적
조직구조	단절된 개발, 운영, 보안 팀	데브옵스 협업

1) CI/CD (Continuous Integration/Continuous Deployment, 지속적 통합/연속적 배포)

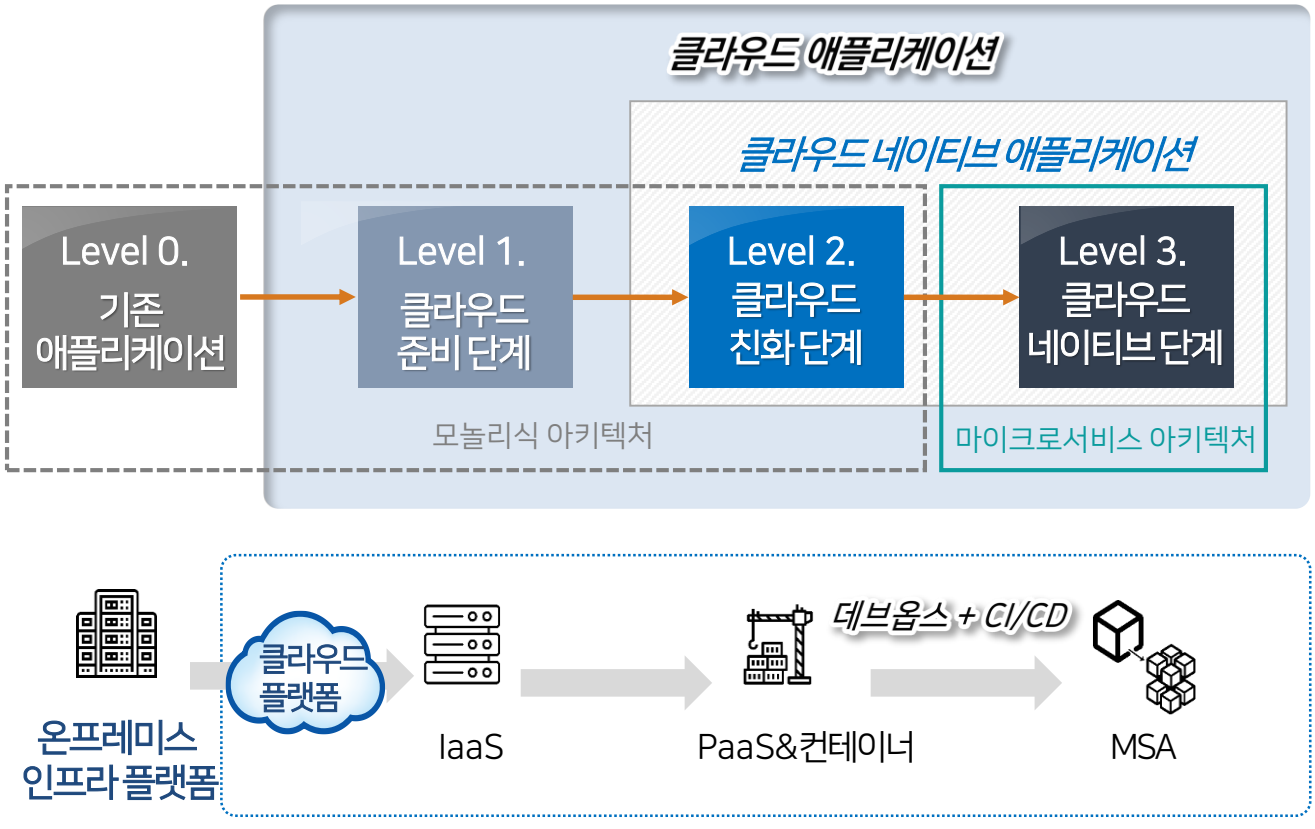
01 클라우드 네이티브 개요

1.2 클라우드 네이티브 애플리케이션

1.2.2 클라우드 애플리케이션 성숙도 단계

- 넓은 의미에서 클라우드 네이티브 애플리케이션은 PaaS(Platform-as-a-Service)를 기반으로, 컨테이너, 데브옵스, CI/CD 기술을 적용한 클라우드 친화 단계와 여기에 MSA(Microservice Architecture, 마이크로서비스 아키텍처)까지 포함한 클라우드 네이티브 단계의 애플리케이션을 모두 포함한다.
- 클라우드 애플리케이션의 성숙도 단계는 Level 1. 클라우드 준비 단계 → Level 2. 클라우드 친화 단계 → Level 3. 클라우드 네이티브 단계로 구분된다. 기존 애플리케이션을 IaaS(Infrastructure-as-a-Service) 환경에 구현하는 클라우드 준비 단계, PaaS와 컨테이너를 도입하는 클라우드 친화 단계, 데브옵스 및 CI/CD 도입과 함께 MSA를 적용하는 클라우드 네이티브 단계로 발전한다.
- 기존 애플리케이션, 클라우드 준비 단계와 친화 단계의 애플리케이션은 모놀리식 아키텍처로 설계되고, 클라우드 네이티브 단계의 애플리케이션은 MSA 기반으로 설계된다.

[그림 1-4] 클라우드 애플리케이션 성숙도 단계



※ 데브옵스와 CI/CD는 클라우드 친화 단계 또는 클라우드 네이티브 단계에 도입할 수 있음

01 클라우드 네이티브 개요

1.3 클라우드 네이티브 구성요소

- 클라우드 네이티브를 위한 4가지 구성요소는 마이크로서비스, 컨테이너, 데브옵스, CI/CD이다.

[그림 1-5] 클라우드 네이티브 구성요소

1 마이크로서비스

애플리케이션의 민첩성과 유지·관리 편의를 위해 기존의 모놀리식 아키텍처를 작은 기능 단위인 서비스로 분리하고 API[1]를 통해 서로 커뮤니케이션할 수 있게 하여 독립적으로 개발, 배포, 운영함

2 컨테이너

어떠한 클라우드 환경에도 즉각적인 배포가 가능하도록 확장성과 이식성이 필요하며 이를 위해 기존의 서버보다 훨씬 가벼운 프로세스 수준의 가상 서버, 컨테이너를 마이크로서비스의 배포 및 실행환경으로 제공함

3 데브옵스

클라우드 네이티브 환경에서는 하나의 대규모 릴리스 및 업데이트를 기다리는 것이 아니라 수많은 마이크로서비스 중 변경되는 서비스들이 하나의 배포단위로 애플리케이션을 쉽게 출시할 수 있음

4 CI/CD

수많은 마이크로서비스를 개별 팀에서 수시로, 빠른 속도로 변경하기 위해서 클라우드 플랫폼 내에 개발, 배포 프로세스의 자동화가 필요

1) API(Application Programming Interface) : 애플리케이션 SW의 개발 및 통합에 사용되는 정의 및 프로토콜 세트를 의미

01 클라우드 네이티브 개요

1.4 클라우드 네이티브 특장점

1.4.1 클라우드 네이티브 특징

- 클라우드 네이티브 구성요소인 마이크로서비스, 컨테이너, 데브옵스, CI/CD의 적용에 따른 클라우드 네이티브의 특징은 다음과 같다.

[표 1-2] 클라우드 네이티브 특징

구성요소	주요 특징	설명
마이크로서비스	소규모 서비스	• 결합도와 응집도가 높은 업무를 마이크로서비스 단위로 분리
	독립적 서비스 운영	• 마이크로서비스는 다른 서비스 기능과 분리되어 독립적으로 운영 • 특정 서비스 장애 시 해당 서비스만 격리함으로써 나머지 전체 서비스는 정상적으로 운영
	다양한 언어 및 기술 적용 - 폴리글랏(Polyglot)	• 마이크로서비스별로 고유한 언어와 기술요소 적용 가능 • 실험적으로 새로운 기능을 적용해보고 다른 마이크로서비스로 변경 가능
	유지보수 용이성	• 해당 마이크로서비스에 대한 개발 및 테스트가 이뤄지므로 유지보수 용이
컨테이너	효율적 개발환경 구축	• 컨테이너를 이용해 다른 애플리케이션과 분리된 효율적인 개발환경 구축 • 개발자가 개발환경 구성 및 튜닝 등에 소모하던 시간을 줄일 수 있음
	경량화	• 컨테이너는 애플리케이션과 런타임 SW만으로 작은 사이즈의 이미지 파일 생성 • VM(Virtual Machine, 가상머신)은 OS, 애플리케이션, 런타임을 포함하여 이미지 파일을 생성하므로 이미지 사이즈가 큼)
	높은 이식성	• 애플리케이션 시스템을 어떠한 클라우드 플랫폼에도 배포 가능
	확장성	• 리소스가 정해진 임계치를 초과할 경우 자동으로 스케일 아웃하는 오토스케일링(Autoscaling) 기능 제공
데브옵스	서비스 단위 개발·운영조직 협업	• 마이크로서비스 단위로 개발과 운영조직을 통합하여 조직 간 협업 가능
	신속한 서비스 개발 및 배포	• 개발조직과 운영조직 간 협업을 통해 문제점이나 오류를 조기에 해결함으로써 신속한 서비스 개발 및 배포 가능
CI/CD	프로세스 자동화	• 개발, 빌드, 테스트, 배포에 이르는 파이프라인 전체 과정에 대한 자동화
	소규모 서비스 단위 배포	• 프로세스 자동화를 통해 개발 및 배포 시간을 단축함으로써 배포 주기가 짧아짐

01 클라우드 네이티브 개요

1.4 클라우드 네이티브 특장점

1.4.1 클라우드 네이티브 특징

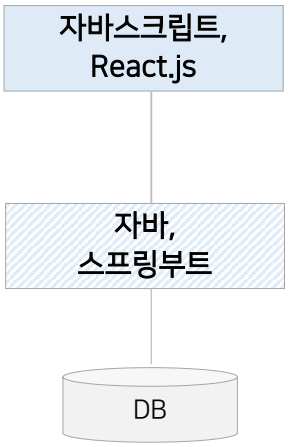
[그림 1-6] 참고. 폴리글랏(Polyglot) 아키텍처

폴리글랏(Polyglot) 아키텍처

다(多)언어 아키텍처, 즉 하나의 애플리케이션(시스템)을 개발하는 데 여러 개의 개발 언어를 사용하는 것을 말하며, 마이크로서비스 폴리글랏 아키텍처는 마이크로서비스별로 상이한 개발 언어를 적용 시 이와 관련된 프레임워크임

모놀리식 아키텍처의 폴리글랏

프론트엔드와 백엔드 개발 시 상이한 개발언어 사용



프론트엔드[1] (UI)

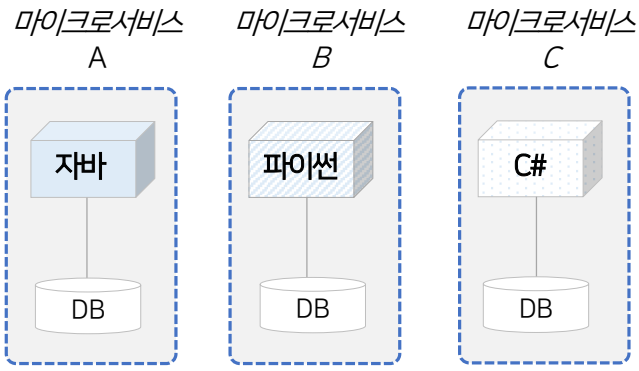
- 사용자에게 UI(User Interface, 사용자 인터페이스)를 제공하는 부분으로 신속한 개발이 중요함
- 자바스크립트, 파이썬(Python) 등의 언어와 관련 프레임워크를 주로 사용

백엔드[2] (서비스)

- 비즈니스 로직, DB 등을 처리하는 부분으로 트랜잭션 처리를 위한 안정성을 필요로 함
- 자바, 스프링 프레임워크 주로 적용

마이크로서비스 아키텍처의 폴리글랏

마이크로서비스별 상이한 개발언어 적용



- 독립적인 작은 단위인 마이크로서비스별로 최적화된 언어, 프레임워크 선택 사용 가능
- 해당 마이크로서비스의 특징에 대한 이해를 토대로 적합한 개발언어 등 개발환경 구성

1) 프론트엔드(Front-end) : 사용자가 이용하는 웹이나 앱의 모든 화면 요소를 의미
2) 백엔드(Back-end) : 서버나 미들웨어 등에서 데이터와 업무 로직 등을 처리

01 클라우드 네이티브 개요

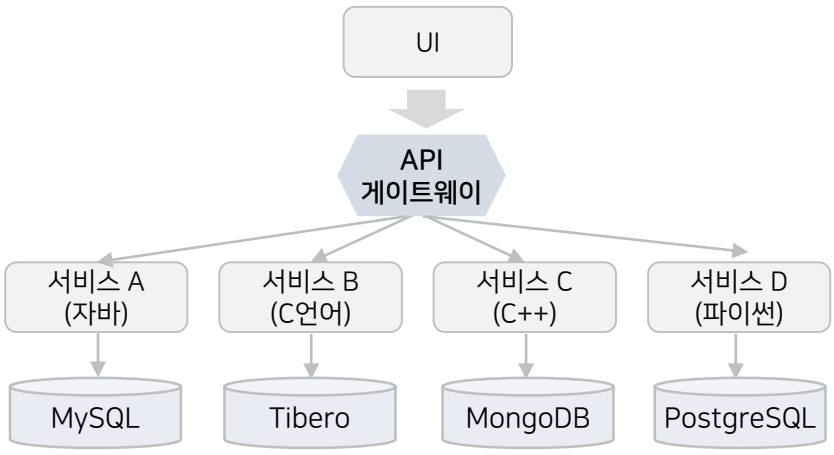
1.4 클라우드 네이티브 특장점

1.4.2 클라우드 네이티브 장점

- 클라우드 네이티브 도입을 통해 얻을 수 있는 장점은 마이크로서비스별 독립적 서비스 운영, 탄력적 시스템 운영 , 마이크로서비스 중심의 효율적인 조직 구성, 데브옵스 및 CI/CD를 통한 개발 기간 단축, 작은 서비스 단위의 신속한 요구사항 반영, 컨테이너를 활용한 이식성 확보 등으로 요약된다.

[그림 1-7] 장점1. 마이크로서비스별 독립적 서비스 운영

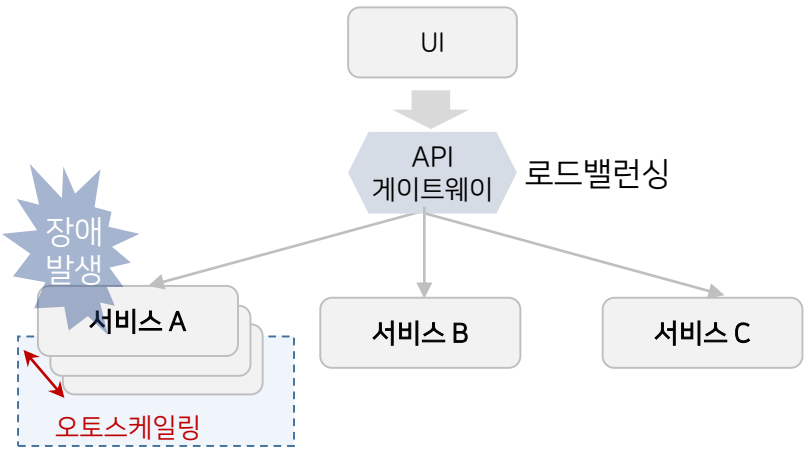
마이크로서비스별 독립적 서비스 운영



- 클라우드 네이티브 애플리케이션을 이루고 있는 각각의 마이크로서비스는 독립적으로 운영
- 서비스 간의 호출은 HTTP Restful API를 통해 느슨하게 결합
- 각 시스템의 프로그래밍 언어가 다르거나 이기종의 데이터베이스를 사용하여도 API 규약에 따른 통신 가능
- 각 서비스에 최적화된 언어와 아키텍처 구성 가능

[그림 1-8] 장점2. 탄력적인 시스템 운영

탄력적인 시스템 운영



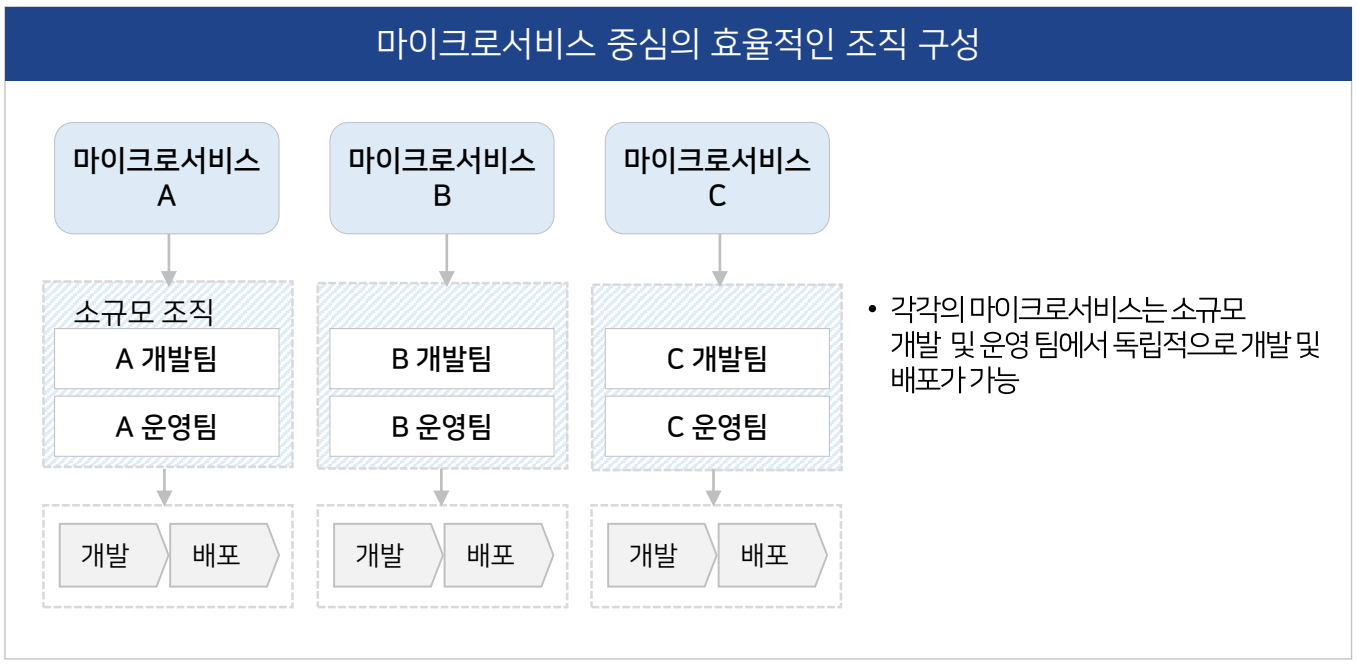
- 하나의 마이크로서비스에 대한 요청 증가 시 오토스케일링을 통해 동적으로 자원을 할당
- 특정 서비스에서 장애 발생 시 로드 밸런서(Load Balancer, 부하분산기)를 통해 장애가 발생된 인스턴스를 제외하여 서비스 정상 운영
- 특정 서비스 장애 시 서킷 브레이커 (Circuit Breaker, 회로차단기)를 통해 해당 서비스 요청 제외

01 클라우드 네이티브 개요

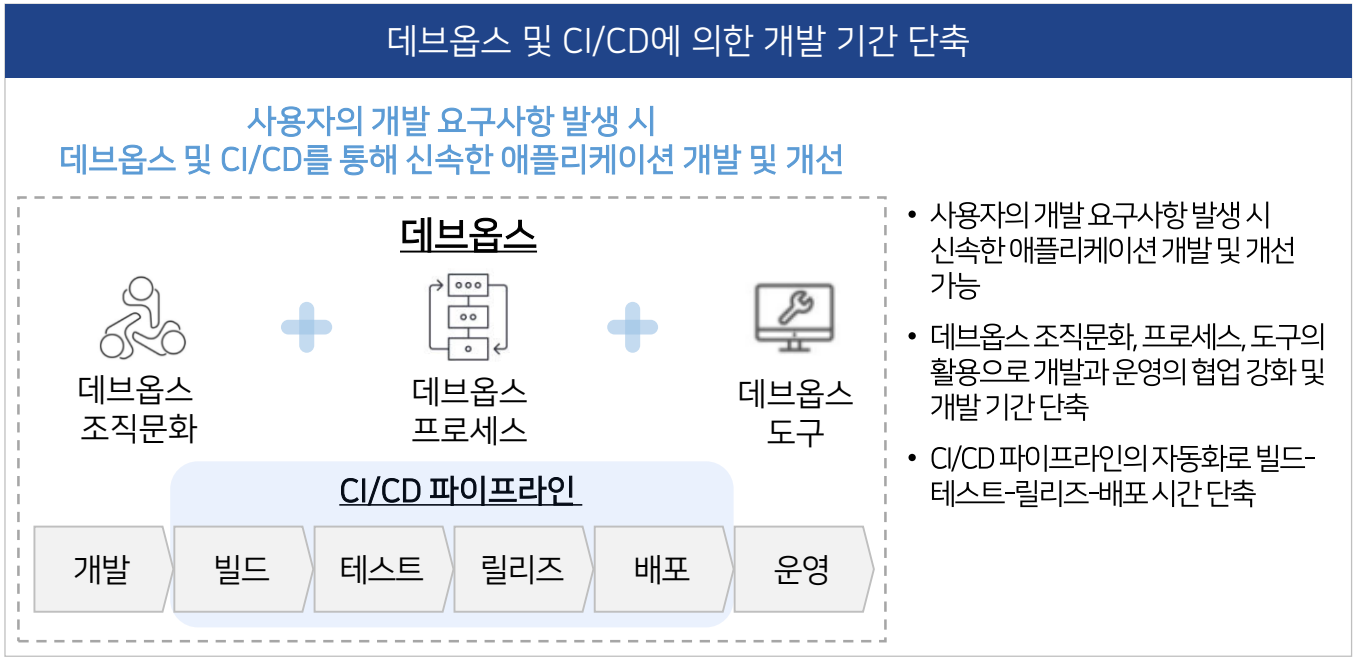
1.4 클라우드 네이티브 특장점

1.4.2 클라우드 네이티브 장점

[그림 1-9] 장점3. 마이크로서비스 중심의 효율적인 조직 구성



[그림 1-10] 장점4. 데브옵스 및 CI/CD에 의한 개발 기간 단축

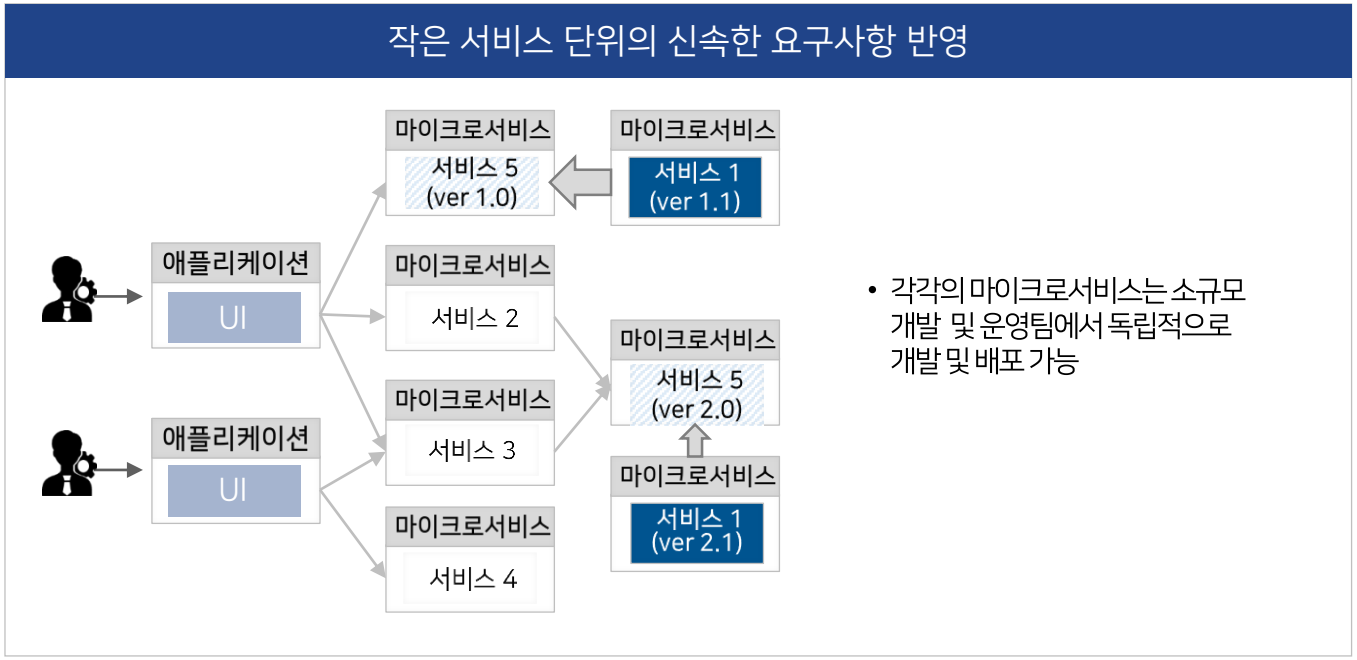


01 클라우드 네이티브 개요

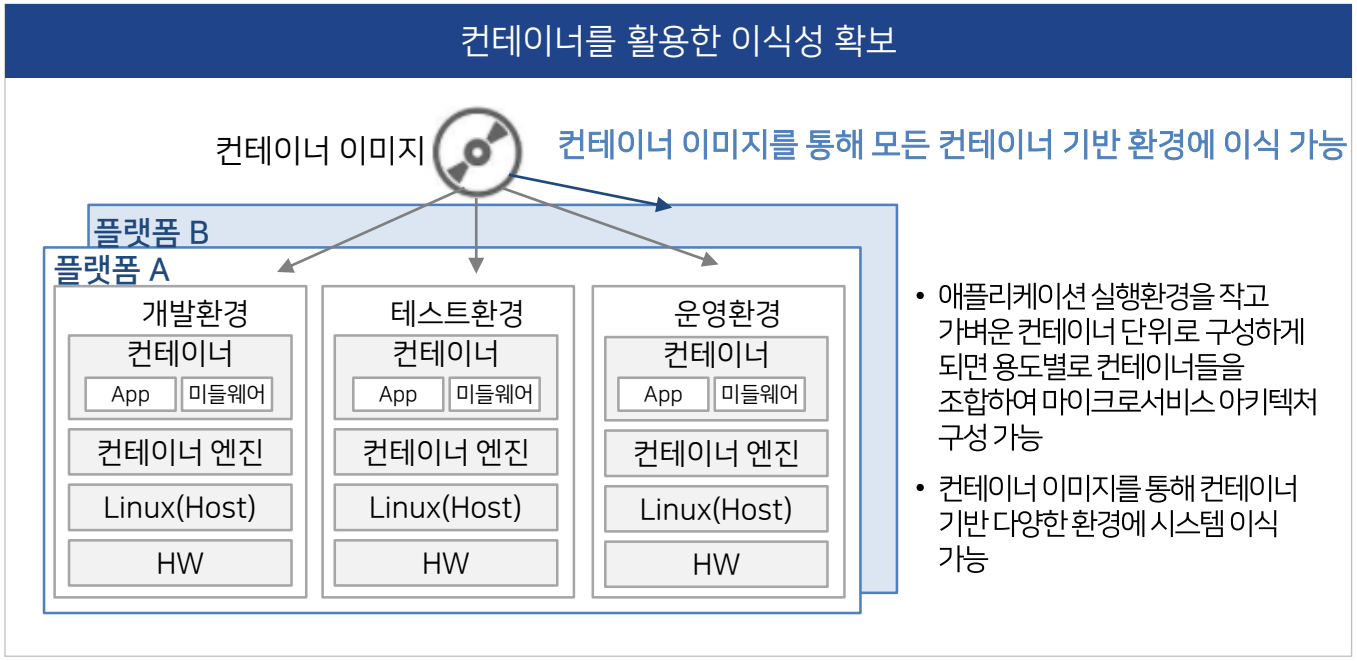
1.4 클라우드 네이티브 특장점

1.4.2 클라우드 네이티브 장점

[그림 1-11] 장점5. 작은 서비스 단위의 신속한 요구사항 반영



[그림 1-12] 장점6. 컨테이너를 활용한 이식성 확보



01 클라우드 네이티브 개요

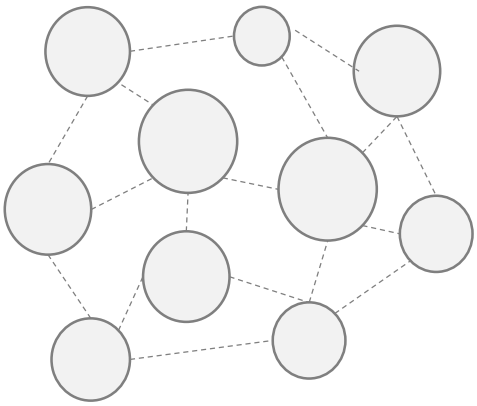
1.4 클라우드 네이티브 특장점

1.4.3 클라우드 네이티브 단점

- 클라우드 네이티브 도입 시 발생할 수 있는 단점은 마이크로서비스 분산에 따른 아키텍처 복잡성 증가, 분산 데이터베이스에 의한 데이터의 중복, 잦은 빌드 및 배포 시 보안 위험 노출, 클라우드 특정 서비스에 대한 종속성 등이 있다.

[그림 1-13] 단점1. 마이크로서비스의 분산에 따른 아키텍처 복잡성 증가

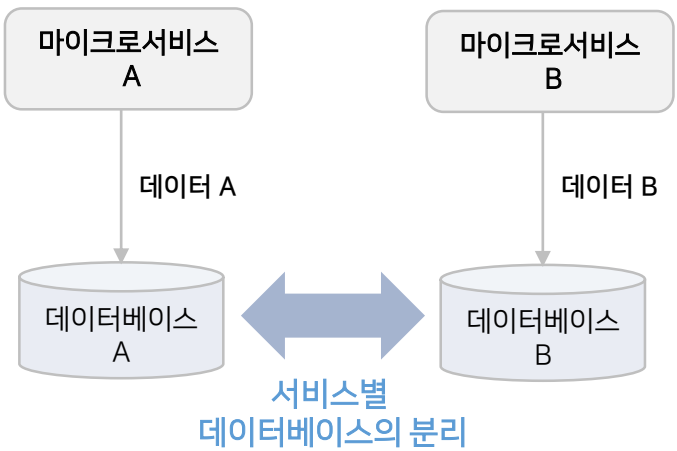
마이크로서비스의 분산에 따른 아키텍처 복잡성 증가



- 시스템의 각 서비스들이 마이크로서비스로 분산되어 관리 포인트의 증가
- 전체 시스템에 대한 아키텍처 복잡도가 증가하여 전체적인 구조 파악의 어려움
- 분산 시스템 추가 시 네트워크 대기 시간, 내결함성, 직렬화 등 고려해야 할 요소 증가

[그림 1-14] 단점2. 분산 데이터베이스에 의한 데이터 중복

분산 데이터베이스에 의한 데이터 중복

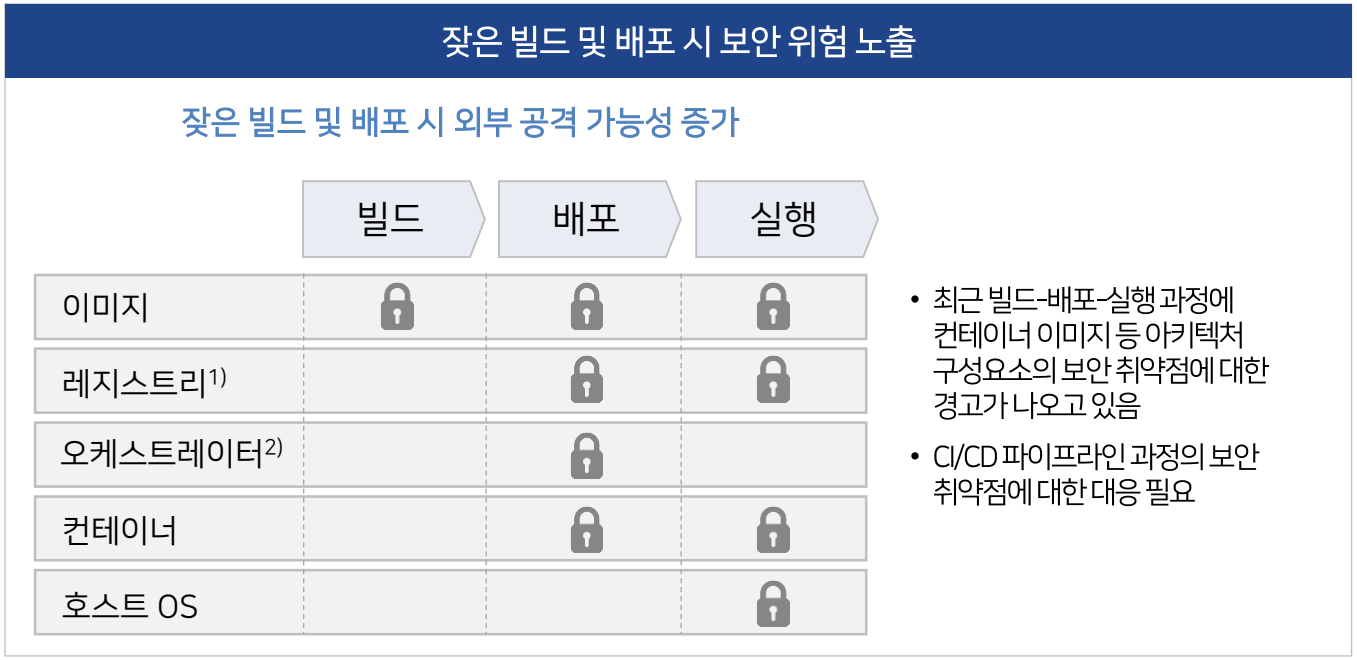


- 분산 데이터베이스로 인하여 중복 데이터 저장 필요(반정규화)
- 각 데이터베이스에 각각 데이터가 저장되므로 데이터 일관성이 깨질 가능성 증가
- 멀티 데이터베이스에 대한 트랜잭션 처리가 필요하며, 트랜잭션 관리에 어려움이 발생할 가능성 증가

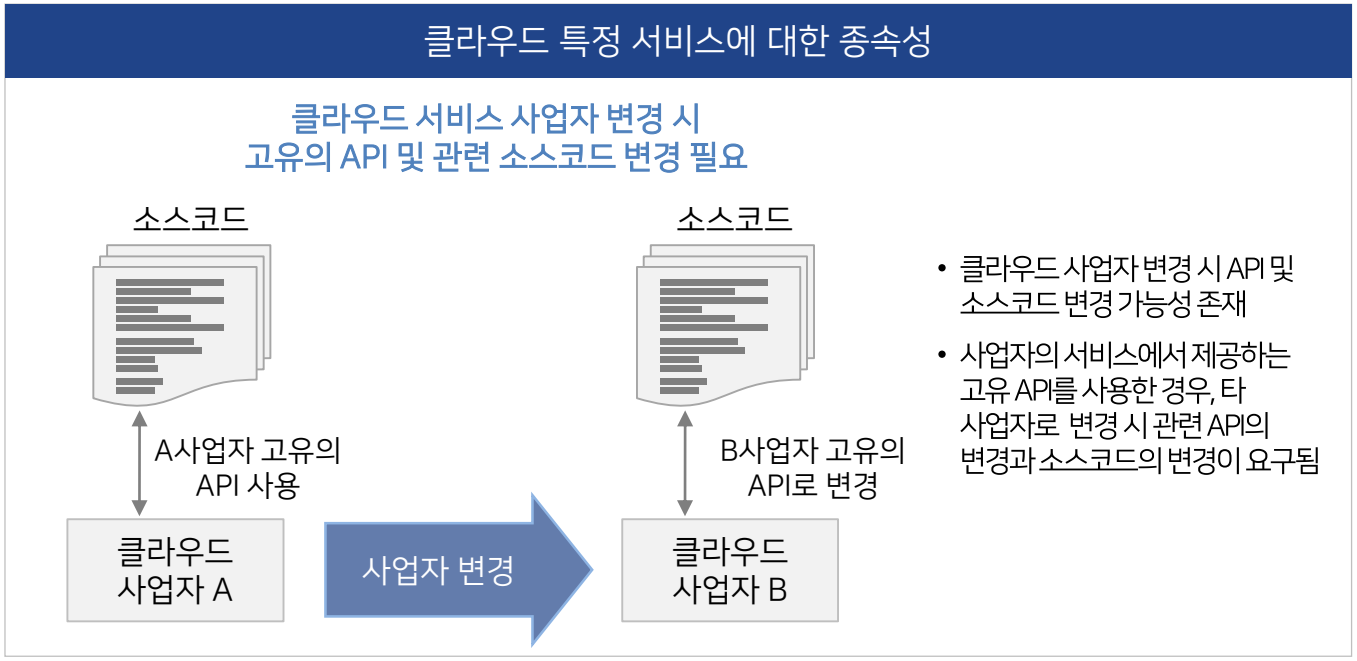
1.4 클라우드 네이티브 특장점

1.4.3 클라우드 네이티브 단점

[그림 1-15] 단점3. 잦은 빌드 및 배포 시 보안 위험 노출



[그림 1-16] 단점4. 클라우드 특정 서비스에 대한 종속성



1) 레지스트리(Registry) : 컨테이너 이미지를 저장하는 저장소
2) 오케스트레이터(Orchestrator) : 배포된 컨테이너들이 정상적으로 동작하는지 관리해주는 기술

클라우드 네이티브 정보시스템 구축을 위한

발주자 안내서

02

클라우드 동향 및 클라우드 네이티브 **도입 필요성**

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.1 클라우드 동향

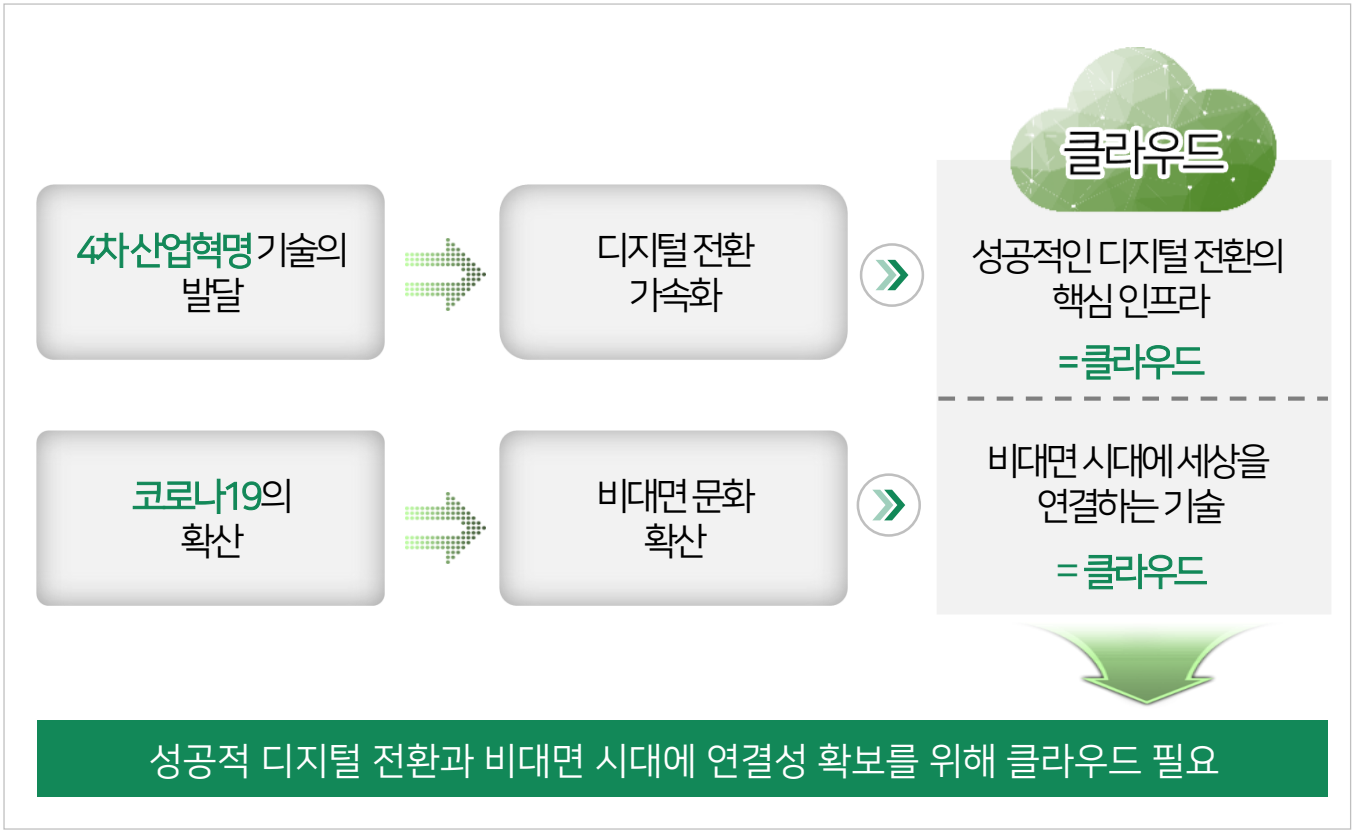
2.1.1 클라우드 확산 배경

- 4차 산업혁명 기술의 발달에 의한 디지털 혁신의 가속화, 코로나19에 따른 비대면 문화의 확산 등에 따라 클라우드가 디지털 전환의 핵심 인프라이자 중요한 전략적 수단이 되고 있다.
- 클라우드, 빅데이터, AI(Artificial Intelligence, 인공지능), IoT(Internet of Things, 사물인터넷), 블록체인, 모바일 등 4차 산업혁명 기술의 발달은 기업과 기관의 전략, 조직, 프로세스, 사업모델, 문화, 커뮤니케이션, 정보시스템 등을 근본적으로 변화시키는 계기가 되었다.
- 재택근무 확산에 따른 비대면 업무처리의 증가, 코로나19 관련 온라인 서비스 증가 등에 대응하기 위한 핵심 인프라로 클라우드가 활용되면서 클라우드의 중요성이 부각되고 있다.

디지털 혁신(DX : Digital Transformation)

기업이 진행하거나 추진하는 혁신 과정 중 하나로 클라우드, 빅데이터, 인공지능(AI), IoT, 블록체인, 가상현실, 모바일 등 방대한 디지털 기술을 하나로 통합해 전사적인 혁신을 추진하는 것임

[그림 2-1] 클라우드의 중요성



02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.1 클라우드 동향

2.1.2 클라우드 정책 동향

2.1.2.1 디지털 정부혁신 추진·발전계획

- 정부는 디지털 전환 시대에 전자정부의 한계를 극복하고자 '19년 디지털 정부혁신 추진계획, '20년 디지털정부혁신 발전계획을 발표하였으며, 민간 또는 공공 클라우드 전환, 클라우드 플랫폼 고도화, 디지털서비스 전문계약 제도 신설에 관한 내용을 포함하고 있다.

[그림 2-2] 디지털 정부혁신 추진계획('19.10)

디지털 정부혁신 추진계획('19.10) – 과제5. 클라우드와 디지털서비스 이용 활성화

민간 클라우드 이용 확대	• 안보수사, 내부시스템을 제외한 **전체 시스템을 이용 대상**으로 확대
개방형 플랫폼	• AI, 클라우드 등을 활용한 서비스를 쉽게 개발·운영할 수 있도록 **개방형 전자정부 클라우드 플랫폼 구축**
서비스 전문계약	• 우수한 민간 서비스를 정부가 이용할 수 있도록 **디지털서비스 전문계약 제도**를 마련하고, 유통 플랫폼을 구축

[그림 2-3] 디지털 정부혁신 발전계획('20.06)

디지털 정부혁신 발전계획('20.06) – 과제3-4. 공공부문 클라우드 전면 전환

클라우드 전환	• **소규모 전산실 위주의 정보시스템**을 클라우드로 전환
클라우드 플랫폼 고도화	• **전자정부서비스 개발환경을 클라우드 플랫폼으로 고도화** • 공공부문 서비스 개발 시 인공지능, 빅데이터 등의 지능형 기술을 손쉽게 적용할 수 있도록 클라우드 기반의 개발·운영 플랫폼 확산
디지털서비스 전문계약제도 신설	• 클라우드 등 민간의 검증된 디지털서비스를 공공에서 신속하게 활용하고, 디지털서비스 산업 육성 • 우수한 중소 소프트웨어 기업의 공공시장 진출 지원

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.1 클라우드 동향

2.1.2 클라우드 정책 동향

2.1.2.2 전자정부 클라우드 플랫폼 구축·확산

- 정부는 전자정부 정보자원의 활용 효율성 제고, 지능정보기술을 활용한 지능형 전자정부의 신속 구현을 지원하기 위해 전자정부 클라우드 플랫폼을 구축하고 있다.
- 전자정부 클라우드 플랫폼은 공공정보 서비스 개발에 필요한 전 영역(인프라, 개발 및 실행환경, 정보자원)을 제공하는 클라우드 환경으로 PaaS 및 SaaS(Software-as-a-Service) 서비스를 제공하고 IaaS를 연계하는 플랫폼이다.

[그림2-4] 전자정부 클라우드 플랫폼 구성도

[출처: 행정안전부, 전자정부 클라우드 플랫폼 구축 현황, 2020.12.3]

[그림2-5] 전자정부 클라우드 플랫폼의 특징

공통기반

- 정보자원을 서비스 형태로 범정부적으로 공용하는 플랫폼
- AI, 빅데이터 등 지능형 서비스 개발 도구 제공

PaaS·SaaS 플랫폼

- 멀티 PaaS 플랫폼 구현 지원
  - 쿠버네티스, 클라우드 파운드리 활용
- 공공 및 민간 SaaS 제공 (연계형, 설치형)

국가정보자원관리원

- 국가정보자원관리원의 지능형 클라우드 인프라 활용

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.1 클라우드 동향

2.1.2 클라우드 정책 동향

2.1.2.3 민간 클라우드(SaaS) 선도이용 지원

- 정부는 급변하는 디지털 수요에 탄력적으로 대응할 수 있도록 공공부문 정보시스템에 대해 클라우드 환경으로 전환을 추진하고 있으며, 그 일환으로 전문성·효율성이 높은 민간서비스(SaaS 등)를 선도 이용할 수 있도록 지원하고 있다.

[그림2-6] 민간 클라우드(SaaS) 선도이용 지원 추진 배경

급변하는 디지털 수요에 탄력적으로 대응할 수 있도록
공공부문 정보시스템을 전문성·효율성을 갖춘
클라우드 환경으로 전환 추진중('21년 ~ '25년)

정부주도 구축운영 방식은 다양성·창의성·혁신성 적기 반영에 한계
기관 단위의 서비스 개발에서 서비스 즉시 이용 형태로의 패러다임 변화가 필요한 시점

↓

행정·공공기관에 활용성·효율성 높은
민간 클라우드 서비스 선도이용 지원

추진 근거			
	전자정부법 제54조 정보자원 통합관리 및 동법 시행령 제66조	행정기관 및 공공기관 정보자원 통합기준 (행안부 고시 제2020-31호)	클라우드컴퓨팅 제12조 및 제20조 (클라우드컴퓨팅 도입/이용 촉진)

[그림2-7] 민간 클라우드(SaaS) 선도이용 지원사업

사업 목적	• 행정·공공기관 대상 민간 클라우드 서비스 선도이용 지원을 통해 공공부문의 민간 클라우드 서비스(SaaS 등) 이용 활성화
사업 내용	• 민간 클라우드 서비스를 이용하고자 하는 기관에 대해 초기 이용료 지원(~ '22년 2월)
지원 대상	• 전자정부법 제2조에 따른 행정기관 및 공공기관
대상 서비스	• 공공 부문에서 이용 가능한 민간 클라우드 서비스

[출처: 행정안전부, 민간 클라우드(SaaS) 선도이용 지원 사업 추진 현황, 2021.12.2]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.1 클라우드 동향

2.1.2 클라우드 정책 동향

2.1.2.4 전자정부 수출

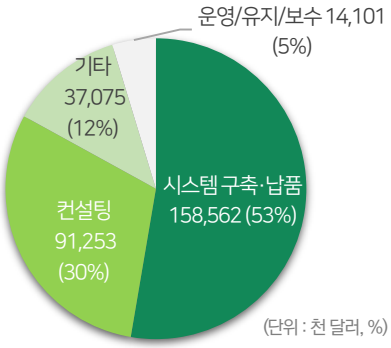
- 전자정부 수출은 해외 정부가 전자정부를 구현하는 데 필요한 인적·물적 및 지적 자원을 제공하는 행위를 말한다. 정부는 범정부 협업 및 민관 협력, 국제협력 등을 통해 전자정부의 해외수출을 촉진하고 있다.
- '19년 전자정부 수출은 175개 업체가 70개 국가를 대상으로 총 368건, 약 3억 달러의 실적을 달성하였으며, 매년 수출실적은 꾸준히 증가하고 있다.
- '19년 전자정부 수출실적은 시스템 구축 납품이 53%, 컨설팅 30%, 기타 12%, 운영/유지/보수가 5%를 차지한다.

[그림 2-8] 전자정부 수출실적('17년-'19년)

행정·공공기관 정보시스템 및 정보자원 현황('20.10월 기준)

구분	2017년	2018년	2019년
수출금액	236,053천 달러	258,318천 달러	300,992천 달러
수출건수	180건	201건	268건
진출국가	72개	60개	70개
진출업체	79개	112개	175개

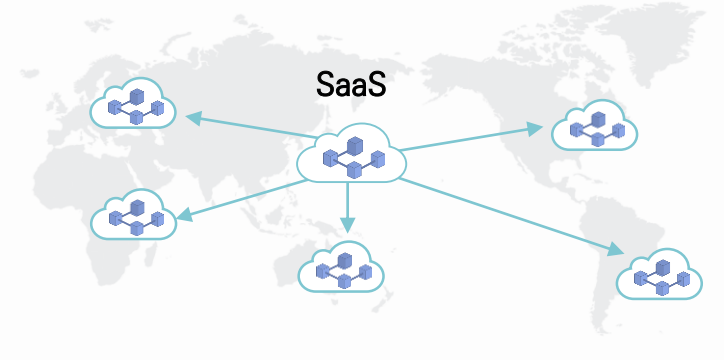
사업 유형별 수출실적 ('19년, 금액 기준)



[출처: 한국지능정보사회진흥원, 2019년도 전자정부 수출실적 조사 결과보고서, 2020.12]

[그림 2-9] 전자정부 수출 방향성

전자정부 시스템을 SaaS의 형태로 개발하여 수출



- 전체 수출액의 50% 이상이 시스템 구축 납품이므로 향후 효율적인 시스템 구축 및 수출(이식)을 위해 SaaS 형태로 전자정부 시스템을 개발할 필요가 있음
- 즉 클라우드 기반으로 전자정부 시스템을 구축하여 해외 정부의 클라우드 플랫폼에 배포할 수 있는 형태의 개발이 요구됨

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.1 클라우드 동향

2.1.2 클라우드 정책 동향

2.1.2.5 해외 주요국가의 클라우드 정책

- 2010년 이후 전 세계적으로 공공부문 클라우드 도입이 주요 관심사로 떠오르며 현재까지 해외 주요 국가에서 클라우드 도입 확대를 위해 국가 전략을 수립하고 활발하게 추진하고 있다.
- 미국, 영국, EU는 10년 전부터 클라우드 퍼스트 정책을 수립하여 클라우드의 활성화를 촉진하였다. 특히 미국과 영국은 공공 부문의 퍼블릭 클라우드 활용을 추진하였고, 영국과 호주는 클라우드에 적합한 서비스 제공을 위해 클라우드 네이티브 추진 정책을 시행하고 있다.

[그림2-10] 주요 국가의 클라우드 정책

미국

- 2010년 중앙정부의 클라우드 퍼스트 정책
- 2019년 6월 '연방 클라우드 컴퓨팅 전략' 보고서를 발표
  - 기존의 '클라우드 퍼스트'에서 새로운 전략인 '클라우드 스마트'로 전환
  - 보안, 조달, 인력 부문 스마트 전략 수립
  - 효율적인 클라우드 도입 가이드 만드는데 주력

영국

- 2011년 3월 '정부 클라우드 전략(Government Cloud Strategy)' 수립
- 2013년 클라우드 퍼스트 정책 발표
- 2013년 전문 계약 제도인 'G-클라우드 프레임워크' 신설 및 매년 갱신
- 2017년 2월 클라우드 퍼스트 정책을 재정립
  - **클라우드 퍼스트에서 클라우드 네이티브로 전환하도록 장려**

EU

- 2018년 GAIS-X 프로젝트 추진
  - 프랑스-독일 중심으로 아마존, MS, 구글 등 미국 클라우드 회사에 대한 의존도를 줄이고, 유럽의 디지털 주권을 회복하고 클라우드 컴퓨팅 생태계를 구축하고자 GAIA-X 프로젝트 추진
- 2021년 3월 '2030 디지털 전환' 발표
  - 기업 비즈니스의 디지털 전환과 공공 서비스의 디지털화 강조

호주

- 2017년 시큐어 클라우드 전략 발표
  - 공공부문 클라우드 전환을 위한 원칙 제시, 민간 클라우드 우선 활용 촉진
  - **클라우드에 적합하도록 서비스 설계**, 퍼블릭 클라우드 서비스 기본 적용
  - 클라우드 최대한 사용 등 민간 클라우드 도입 확대 정책 지속 시행

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

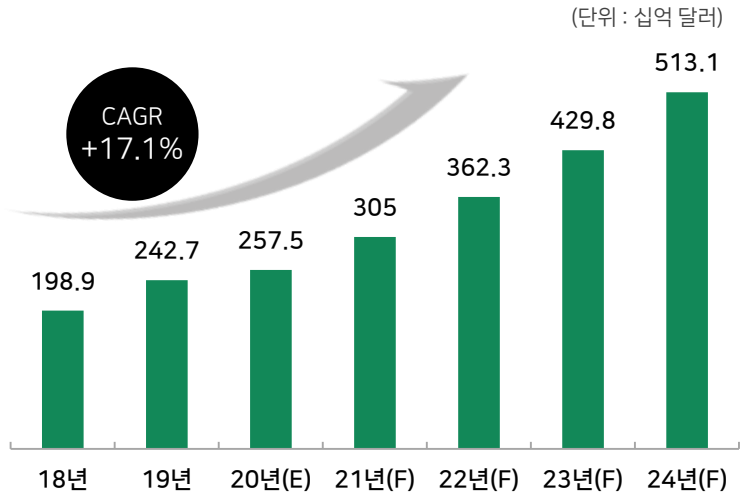
2.1 클라우드 동향

2.1.3 클라우드 시장 동향

2.1.3.1 세계 클라우드 시장 규모

- 세계 클라우드 시장은 '18년 이후 연평균 17.1% 수준으로 급성장하여 '24년에는 5,131억 달러 규모로 예측되며, 세계 퍼블릭 클라우드 시장점유율은 '19년 기준으로 아마존 AWS와 MS Azure가 1, 2위를 차지하여 전체 시장의 71.2%를 점유한다. (가트너, 2020년 10월)

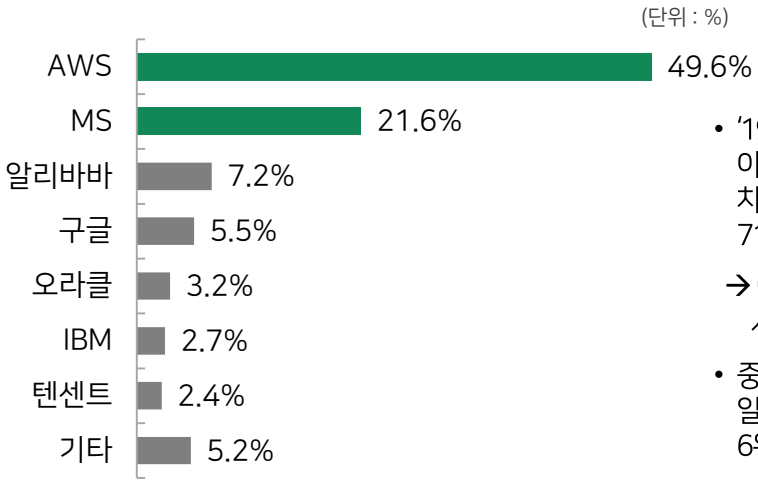
[그림 2-11] 세계 클라우드 시장규모 전망



- 세계 클라우드 시장은 '18년 이후 연평균 17.1% 수준으로 급성장하여 '21년 3,050억 달러, '24년 5,131억 달러 규모로 예측됨

[출처: 가트너, 2020.10.]

[그림 2-12] 세계 퍼블릭 클라우드 시장 점유율('19년)



- '19년 말 기준 세계 퍼블릭 클라우드 시장점유율은 아마존 AWS가 49.6%, MS Azure는 21.6%를 차지하며, 이 2개 벤더의 시장점유율은 전체 시장의 71.2% 를 차지함

→ 아마존 AWS는 규모의 경제를 바탕으로 시장지배적 위치를 계속 유지하고 있음

- 중국의 클라우드 시장이 급성장하고 있으며, 알리바바와 텐센트의 시장점유율은 세계 3위와 6위를 차지함

[출처: 가트너, 2020.10.]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

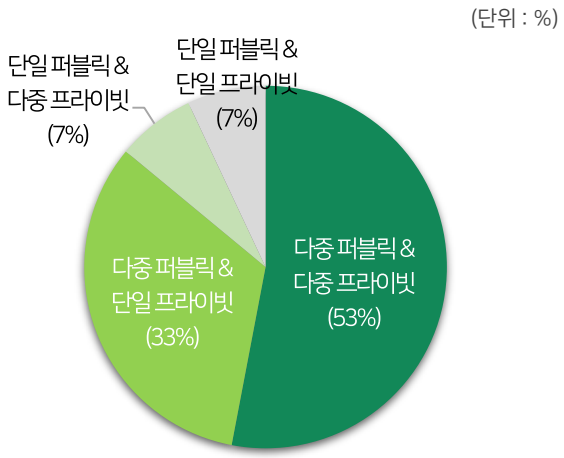
2.1 클라우드 동향

2.1.3 클라우드 시장 동향

2.1.3.2 세계 클라우드 사용 현황

- 2020년 플렉세라 보고서에 따르면 93% 이상의 기관은 2개 이상 클라우드를 이용하고, 86%는 하이브리드 클라우드를 적용하고 있으며, 클라우드 네이티브의 핵심기술은 컨테이너로 도커(Docker)와 쿠버네티스(Kubernetes)를 가장 많이 사용하는 것으로 나타났다.

[그림 2-13] 하이브리드·멀티 클라우드 사용 현황

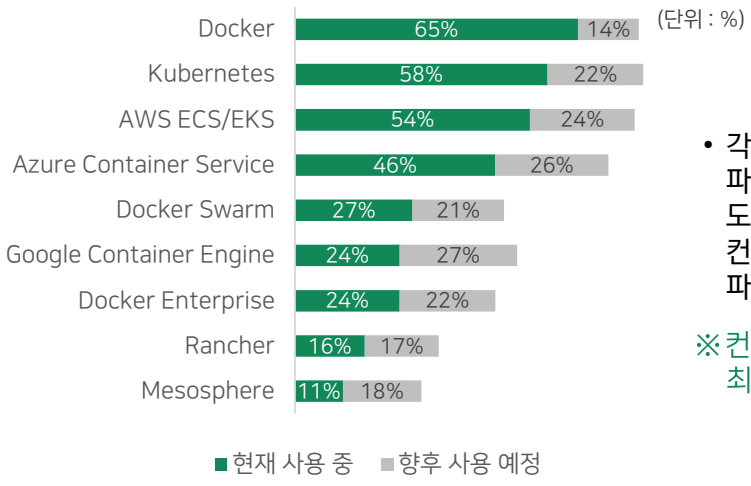


- 2020년 플렉세라 보고서에 따르면 클라우드 사용 기업의 7%만이 단일 퍼블릭 또는 단일 프라이빗 클라우드를 사용하는 것으로 응답함
- 93%의 기관은 2개 이상의 클라우드를 이용하고, 86%는 하이브리드 클라우드를 적용함
  - 클라우드 간 워크로드 이식 경험이 있는 기업은 36% 수준으로 상당히 높음

※ 클라우드 간 높은 이식성이 요구됨

[출처: 플렉세라 보고서, 750개 기업 대상 조사 결과, 2020.]

[그림 2-14] 클라우드 네이티브 기술(컨테이너) 사용 현황



- 각 기업의 클라우드 네이티브 기술의 적용 수준 파악을 위해 컨테이너 도구 사용 현황을 조사한 결과, 도커, 쿠버네티스, AWS ECS/EKS, MS Azure 컨테이너 서비스의 순으로 사용하고 있는 것으로 파악됨

※ 컨테이너 도구는 주로 배포용 도구로 활용되었으나 최근에는 클라우드 네이티브의 핵심 기술이 됨

[출처: 플렉세라보고서, 750개 기업 대상 조사 결과, 2020.]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

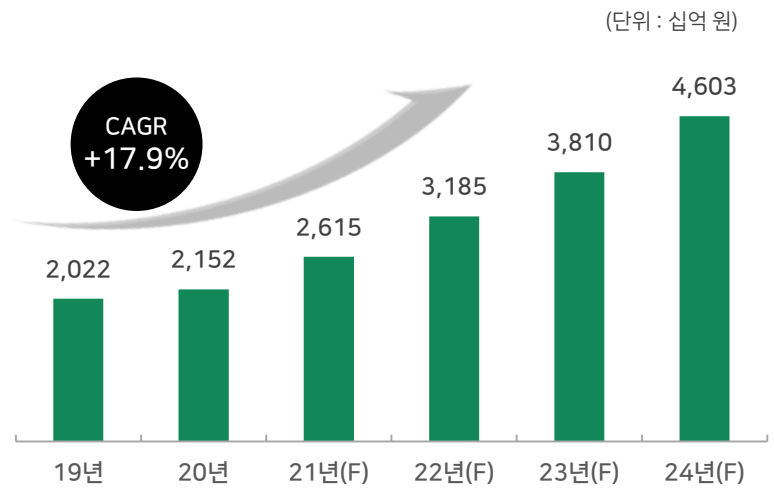
2.1 클라우드 동향

2.1.3 클라우드 시장 동향

2.1.3.3 국내 클라우드 시장 규모

- 국내 퍼블릭 클라우드 시장은 연평균 17.9% 수준의 성장이 예측되며, 국내 퍼블릭 PaaS 시장은 전체 클라우드 시장보다 가파른 성장으로 연평균 성장률이 28.8% 수준이 될 것으로 기대되고 있다.
- PaaS는 개발자들이 애플리케이션 실행에 필요한 환경 구성에 대한 부담을 줄이고, 빠르게 애플리케이션을 배포할 수 있도록 돕기 때문에 클라우드 네이티브 도입을 위해 필수적이다.

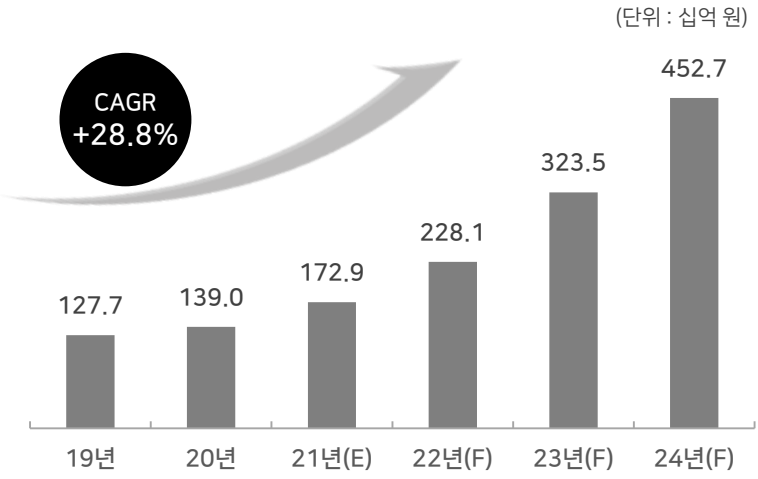
[그림 2-15] 국내 퍼블릭 클라우드 시장규모



- 국내 퍼블릭 클라우드 시장 규모는 '19년 2조 원에서 '24년 4.6조 원으로 연평균 17.9% 성장할 것으로 예측됨
- 이는 전 세계 클라우드 시장 성장률은 17.1%보다 약간 높은 수준임

[출처: 가트너, 2020.10.]

[그림 2-16] 국내 퍼블릭 PaaS 시장규모



- 국내 퍼블릭 PaaS 시장은 '19년 1,277억 원에서 '24년 4,527억 원으로 연평균 28.8% 성장할 것으로 예측됨
- 국내 퍼블릭 PaaS 시장은 국내 퍼블릭 클라우드 성장률 17.9%보다 높은 28.8%로 급격한 성장이 이뤄질 것으로 보여 짐

[출처: IDC, 2020.]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

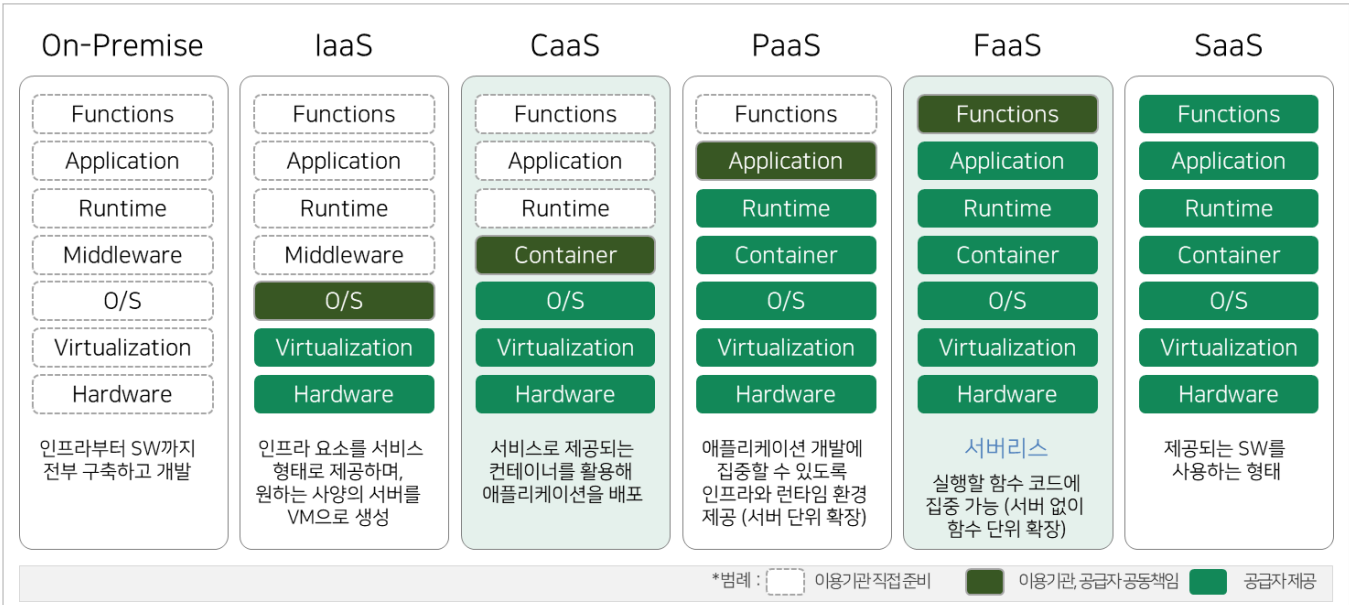
2.1 클라우드 동향

2.1.4 클라우드 기술 동향

2.1.4.1 클라우드 서비스 모델

- 클라우드 서비스 모델은 사용자가 클라우드 서비스 제공업체로부터 얼마만큼의 자원을 제공받고, 얼마만큼의 자원을 직접 관리하는가에 따라 IaaS, CaaS, PaaS, FaaS, SaaS의 5가지 모델로 구분된다.

[그림 2-17] 클라우드 서비스 모델의 유형



[표 2-1] 클라우드 서비스 모델 설명

구분	설 명
IaaS	• 서비스로서의 인프라스트럭처(Infrastructre-as-a-Service) • 하드웨어 자원을 네트워크를 통해 이용하는 형태 • 대표적으로 가상 서버나 온라인 스토리지 등을 제공
CaaS	• 서비스로서의 컨테이너(Containers-as-a-Service) • 컨테이너 기반 추상화를 통해 사용자가 애플리케이션을 배포하고 관리하도록 지원
PaaS	• 서비스로서의 플랫폼(Platform-as-a-Service) • 기업의 애플리케이션 실행환경 및 애플리케이션 개발 환경을 서비스로 제공하는 모델
FaaS	• 서비스로서의 기능(Function-as-a-Service) • 서버리스(serverless) 컴퓨팅을 구현하는 방식으로, 개발자가 비즈니스 로직을 작성하면 플랫폼이 관리를 전담하는 리눅스 컨테이너에서 이를 실행
SaaS	• 서비스로서의 소프트웨어(Software-as-a-Service) • 주로 업무에서 사용하는 소프트웨어의 기능을 인터넷 등의 네트워크를 통해 필요한 만큼 서비스로 이용할 수 있도록 제공하는 형태

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

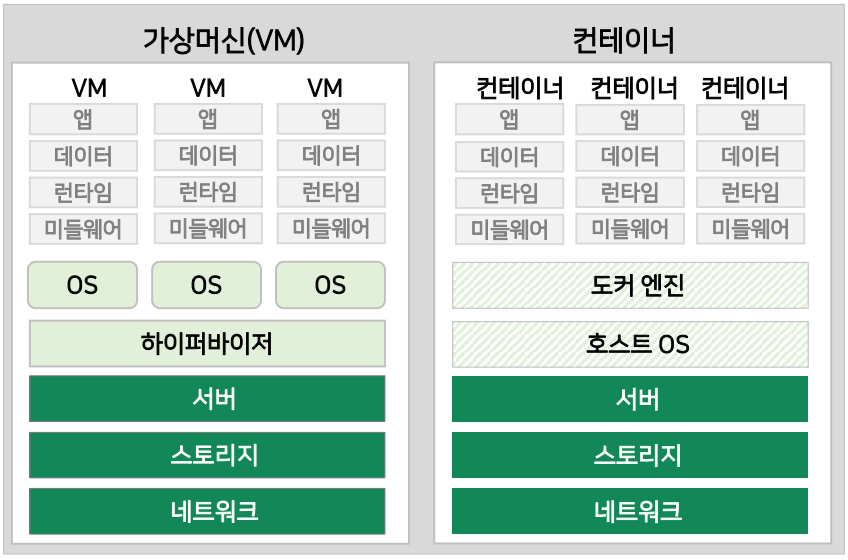
2.1 클라우드 동향

2.1.4 클라우드 기술 동향

2.1.4.1 클라우드 서비스 모델

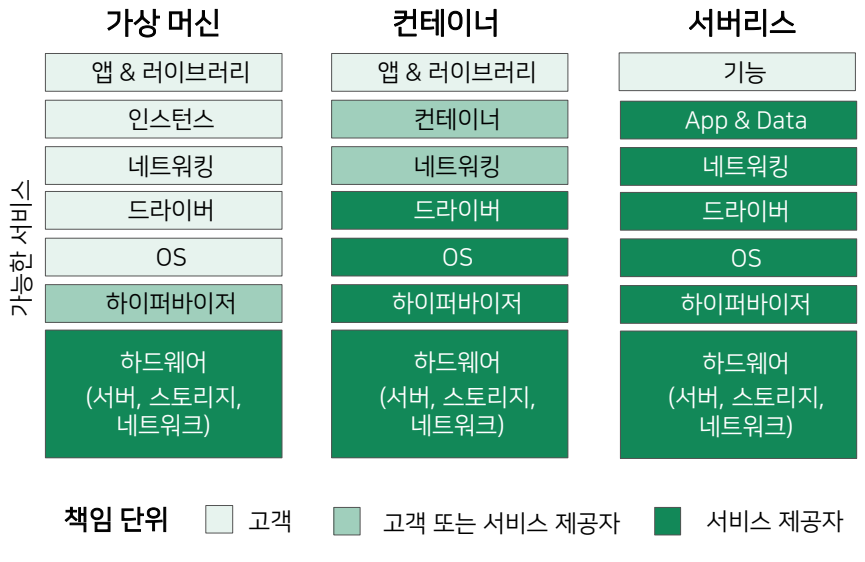
- 최근 컨테이너와 서버리스 등의 최신 기술이 발달하면서 CaaS와 FaaS 서비스 모델이 부각되고 있으며, CaaS는 컨테이너 기반 가상화의 한 형태이고, FaaS는 함수(Function)를 서비스로 제공하는 모델이다.

[그림 2-18] CaaS 서비스 모델



- 가상머신은 하드웨어를 가상화 게스트 OS 위에서 가상 머신 구동
- 컨테이너는 운영체제 레벨에서 자원을 격리하여 할당하기 때문에 게스트 OS 불필요

[그림 2-19] FaaS 서비스 모델(서버리스)



- 서버를 관리할 필요 없이 특정 이벤트에 반응하는 함수를 등록하고 해당 이벤트가 발생하면 함수가 실행되는 구조
- 데이터 처리, 시스템 통합, 간단한 API 및 마이크로서비스 구축에 활용

  예) AWS Lambda, Azure Functions, Google Cloud Functions 등

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

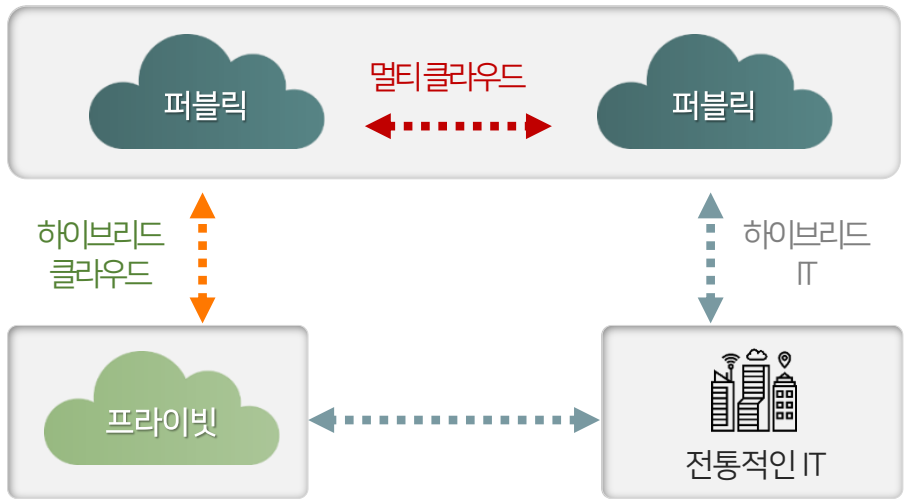
2.1 클라우드 동향

2.1.4 클라우드 기술 동향

2.1.4.2 클라우드 이용 모델

- 클라우드 이용방식에 따라 퍼블릭, 프라이빗, 하이브리드, 멀티 클라우드로 구분된다. 기관의 상황에 따라 필요한 이용 모델을 가져갈 수 있으며, 최근에는 컨테이너 기반의 하이브리드 클라우드에 대한 관심이 커지고 있다.
- 퍼블릭, 프라이빗, 멀티, 하이브리드 클라우드 중 어떠한 이용 모델에서도 개발과 실행이 가능한 클라우드 네이티브 도입이 필요하다.

[그림 2-20] 클라우드 이용 모델의 유형



[표 2-2] 클라우드 이용 모델 설명

구분	설 명
퍼블릭 클라우드 (Public Cloud)	• 클라우드 서비스 공급자가 서버 및 클라우드 리소스 제공 • 소규모, 빠른 서비스가 필요한 조직에 적합
프라이빗 클라우드 (Private Cloud)	• 단일 조직에서 독점적으로 사용되는 컴퓨팅 리소스 제공 • 대규모, 보안이 중요한 조직에 적합
멀티 클라우드 (Multi Cloud)	• 여러 퍼블릭 클라우드를 함께 쓰는 방식 • 안정성 확보를 위한 클라우드 분산 운영이 필요한 조직에 적합
하이브리드 클라우드 (Hybrid Cloud)	• 퍼블릭 클라우드와 프라이빗 클라우드 장점을 모두 필요로 하는 조직에 적합 • 컨테이너(OS 가상화) 기반 클라우드 관리 • 가장 진화된 방식의 클라우드 이용 모델

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

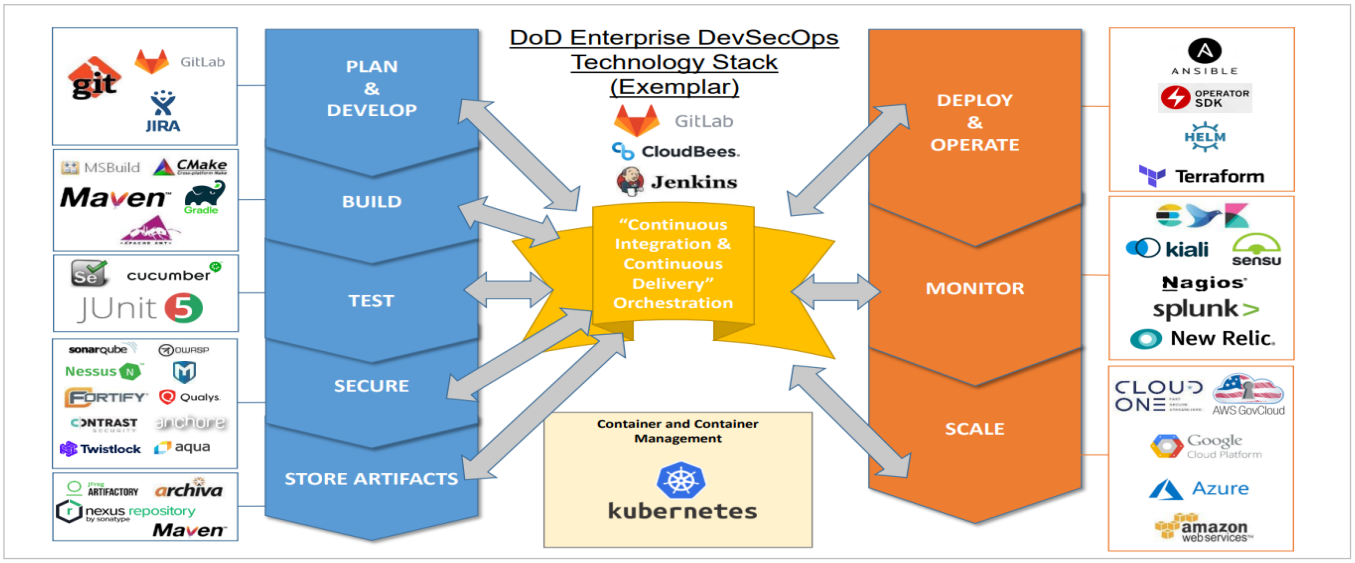
2.2 클라우드 네이티브 도입 사례

2.2.1 해외 공공 부문

2.2.1.1 미국 국무부(DoD)의 클라우드 네이티브 도입 사례

- 미국 국무부(DoD)는 3년 ~10년 가량 소요되는 대형 무기시스템의 소프트웨어 배포 시간을 단축하고자 CNCF 호환 쿠버네티스와 오픈소스를 활용하여 엔터프라이즈 DevSecOps 에코시스템(ONE 플랫폼)을 구축하였다.
- 기존의 무기시스템 개발은 폭포수 방식에 따른 중간 결과물의 확인 곤란, CI/CD 체계 미흡, 사용자의 피드백 반영 미흡, AI 머신러닝(Machine Learning, 기계학습) 및 사이버 보안 관련 사항 반영 곤란 등의 문제가 있었다. 이러한 문제를 해결하기 위해 DevSecOps 플랫폼을 도입하여 배포시간 단축, SW 품질 향상, 비용 절감 등의 효과를 얻었다.

[그림 2-21] DoD 엔터프라이즈 DevSecOps 기술 스택



[표 2-3] DoD 엔터프라이즈 DevSecOps 도입 효과

구분	도입 효과
배포시간 단축 및 신속한 장애 복구	• 106배 더 빠른 개발→ 배포 소요 시간 • 208배 더 빈번한 코드 배포 • 평균 2,604배 빠른 장애 복구
SW 품질 향상	• 7배 더 낮은 변경 실패율 • 비계획 작업 및 재작업 22% 감소 • 보안문제 해결시간 50% 단축
비용절감	• 개발비용 40% 절감
혁신 집중	• 프로토타이핑 시간 44% 감소

[출처: DoD„Enterprise DevSecOps Initiative & Platform One]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

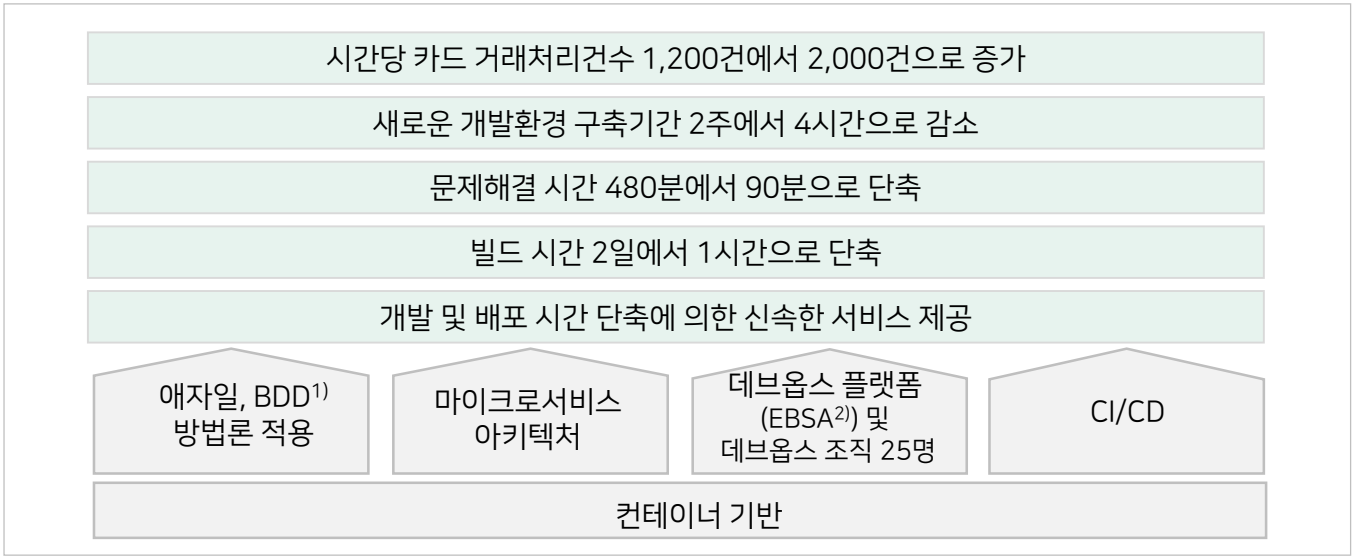
2.2.1 해외 공공 부문

2.2.1.2 영국 내무부(Home Office)의 클라우드 네이티브 도입 사례

- 영국 내무부(Home Office) 이민국은 이민 및 여권, 마약 정책, 범죄, 화재, 테러 대응 및 경찰 등 안전과 보안 관련 행정 업무를 담당하는 기관이다.
- 이민국의 기존 시스템은 기존 호스팅 플랫폼의 확장성 및 안정성 부족으로 디지털 혁신 추진 시 한계점에 직면하게 되어 이를 해결하고자 영국 정부의 사업 중 가장 큰 클라우드 혁신 프로젝트를 추진하였다.
- 기존 호스팅 플랫폼을 아마존 AWS로 전환하면서 클라우드 네이티브 기반으로 플랫폼과 MSA를 적용하여 기존 문제 해결, 빌드 시간 단축 및 개발환경 구축 시간 감소 등의 효과를 얻었다.

[그림 2-22] 클라우드 혁신 추진 내용

[그림 2-23] 클라우드 혁신 효과



[출처:Capgemini, "Home Office completes one of the UK Government's largest cloud transformation projects", 2017.12.]

1) BDD(Behavior Driven Development) : 행동 주도 개발
2)EBSA(Environment Build Support Administration) : 데브옵스 플랫폼

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

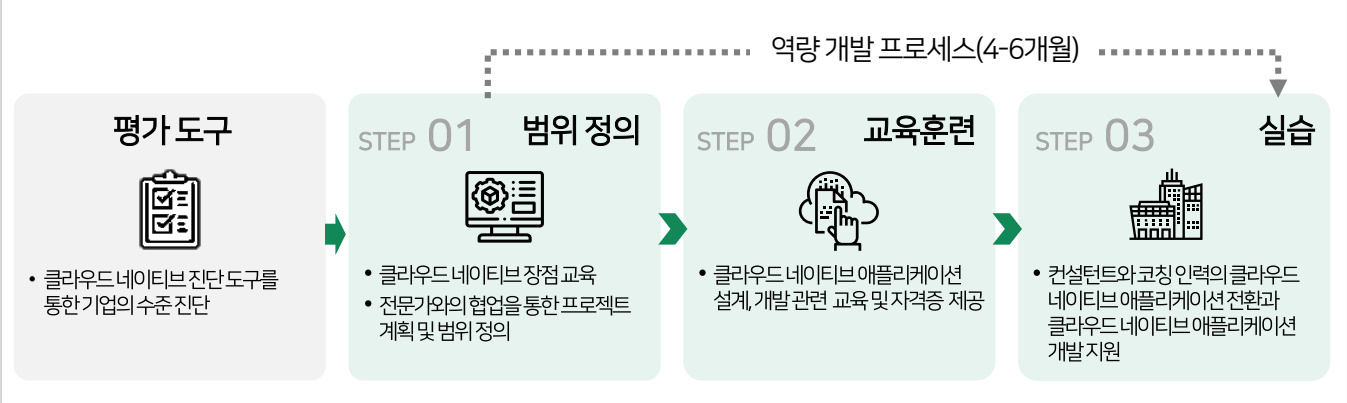
2.2 클라우드 네이티브 도입 사례

2.2.1 해외 공공 부문

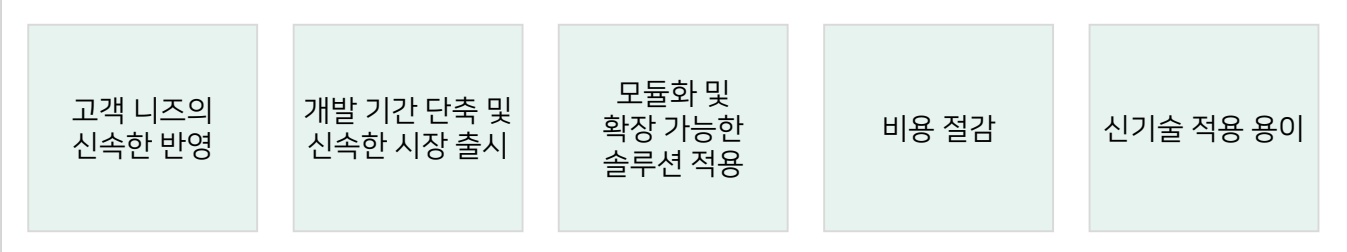
2.2.1.3 싱가포르 정보통신미디어개발청(IMDA)의 클라우드 네이티브 지원 사례

- 싱가포르 정보통신미디어개발청(IMDA)은 통합형 정보통신과 미디어 부문을 전체적으로 개발 및 규제하며 인재, 연구, 혁신 및 기업에 중점을 두는 국가 산업 감독관리 기관이다.
- 싱가포르 IMDA는 싱가포르의 ICT 중소기업의 혁신을 통해 고객에게 보다 효과적인 서비스를 제공하도록 기존 소프트웨어의 혁신을 지원하고자 GoCloud 사업을 추진하고 있다.
- GoCloud의 추진 과제 중 GoCloud 네이티브는 싱가포르 현지 ICT 중소기업이 클라우드 네이티브 기반의 마이크로서비스와 데브옵스의 적용을 통한 개발과 배포환경을 구축하도록 지원한다.
- 2020년 6월 1일부터 GoCloud 수준을 향상시키고자 클라우드 서비스 사업자를 지정하여 해당 서비스 수수료의 최대 80%까지 지원하고 있다.
- 지정된 클라우드 사업자의 역할은 다음과 같다.
  - ICT 중소기업의 개발팀에게 클라우드 네이티브, MSA, 데브옵스 관련 컨설팅과 교육 제공
  - 클라우드 네이티브 애플리케이션 개발 관련 교육을 통해 실제 프로젝트를 수행할 수 있도록 지원

[그림 2-24] 클라우드 네이티브 역량 개발 지원 절차



[그림 2-25] GoCloud 사업의 효과



[출처:https://www.imda.gov.sg/programme-listing/gocloud]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

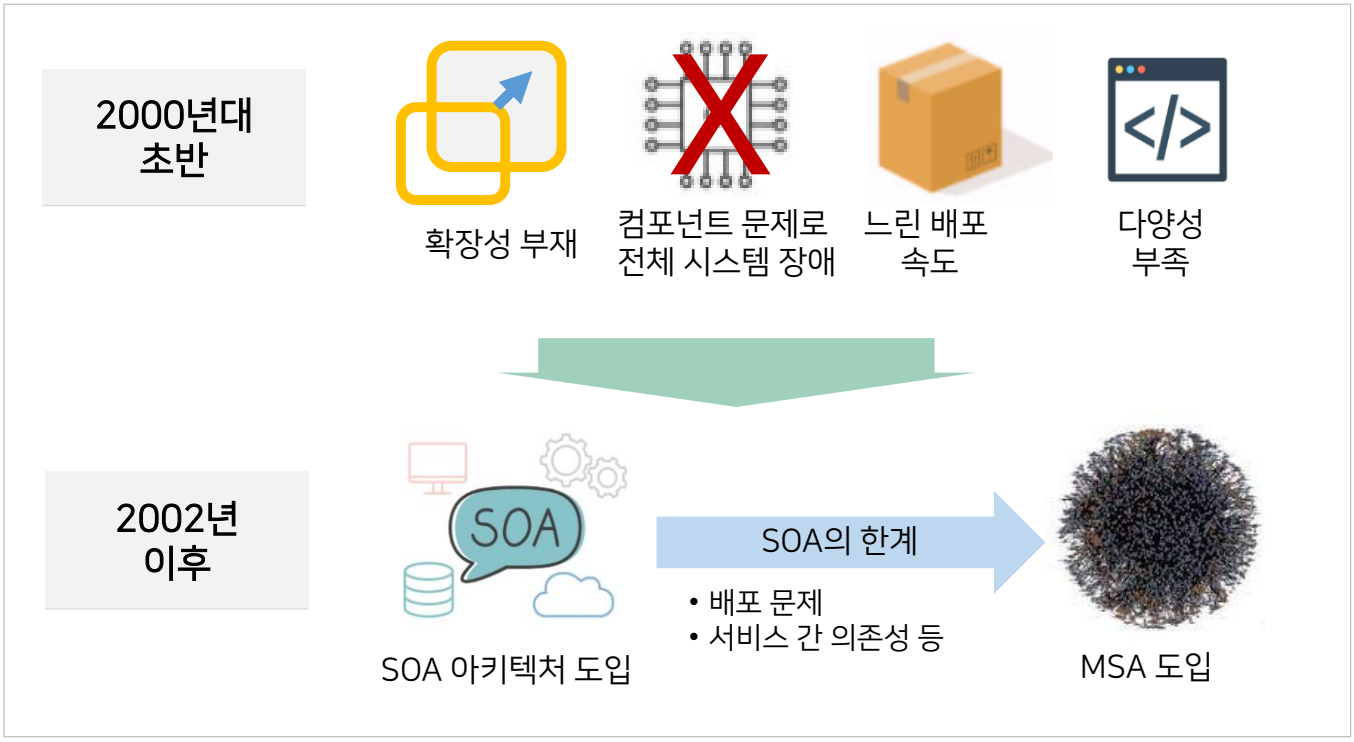
2.2 클라우드 네이티브 도입 사례

2.2.2 해외 민간 부문

2.2.2.1 아마존의 클라우드 네이티브 도입

- 2000년대 초반까지 아마존은 거대한 모놀리식 아키텍처 기반의 시스템을 유지함에 따라 데이터베이스도 점점 비대해지고 있었다. 아마존은 다양한 카테고리의 상품을 제공하고자 하였으나 IT 시스템의 아키텍처 문제에 직면하게 되었다.
- 아마존은 IT 시스템의 확장성 부재, 컴포넌트 문제로 인한 전체 시스템 장애, 느린 배포 속도, 개발도구의 다양성 부족 등의 문제를 해결하고자 DB를 분리하고 SOA(Service Oriented Architecture : 서비스 지향 아키텍처)를 도입하였다. 하지만 여전히 배포 문제와 서비스 간 의존성에 따른 병목현상을 해결하지 못했다.
- 아마존은 SOA의 한계를 해결하기 위해, 고객 지향적 서비스에 대한 담당자의 책임 및 권한 부여, 빠른 혁신을 추구하기 위해 MSA를 도입하게 되었다.

[그림 2-26] MSA 도입 배경



[출처 : 아마존 AWS 코리아, 데브옵스 문화와 모범 사례 및 필수도구]

- 데브옵스팀은 기획, 개발, 운영, 보안 등 각 부문의 인력으로 구성되었고, 1개 팀은 피자 2판으로 점심 식사를 할 수 있는 5 ~ 7명 정도의 인원으로 만들어졌다.

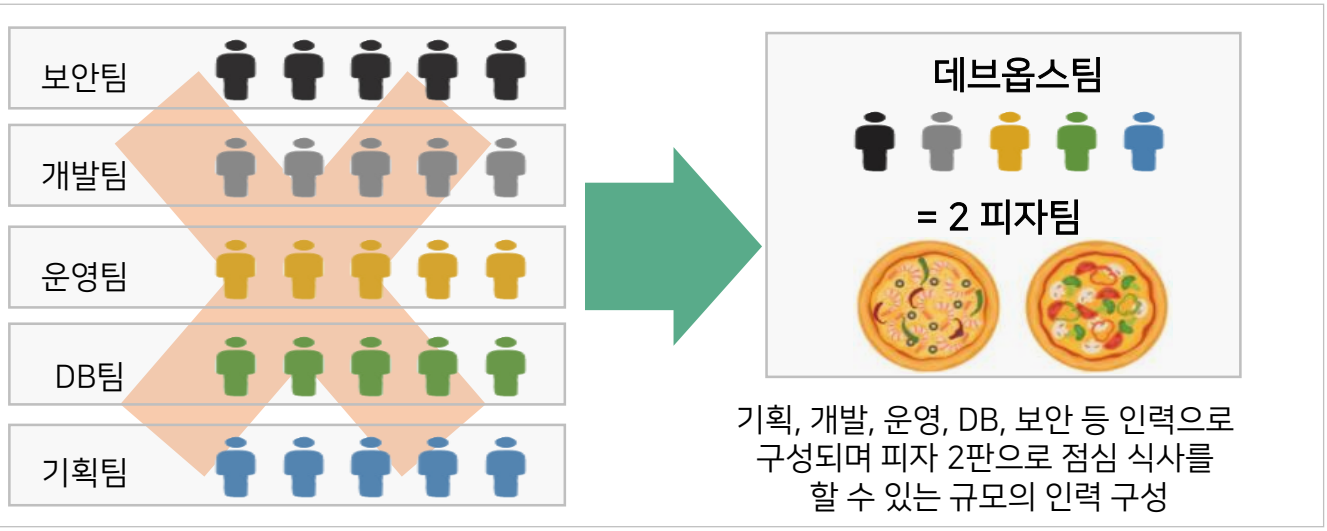
02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.2 해외 민간 부문

2.2.2.1 아마존의 클라우드 네이티브 도입

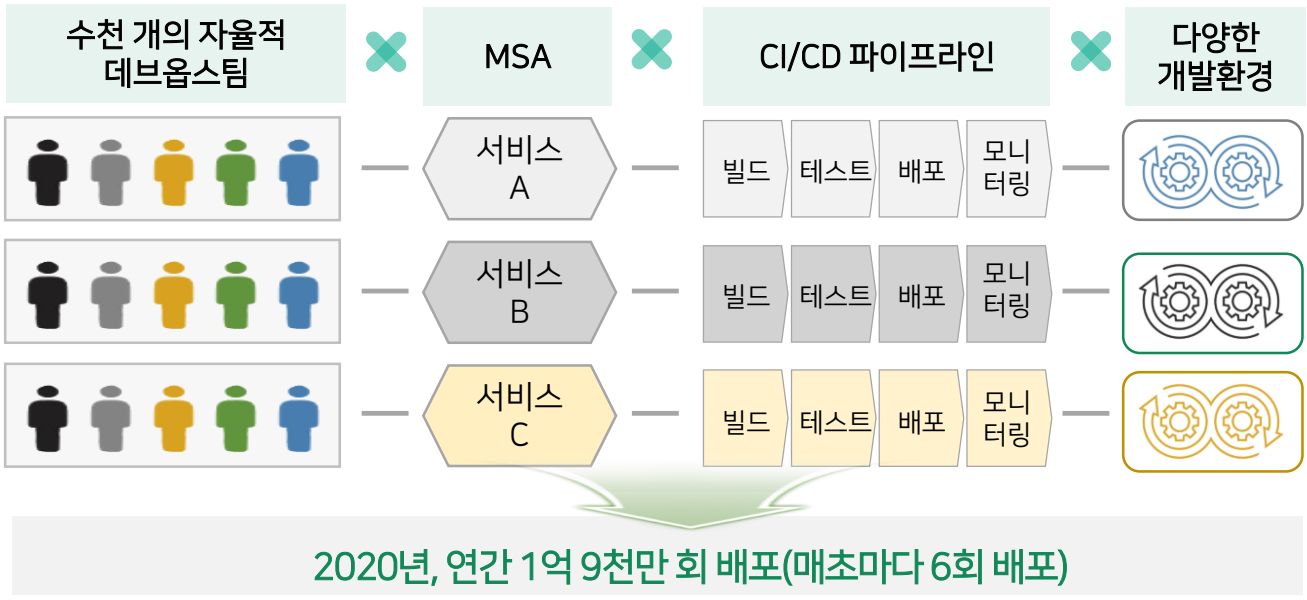
[그림 2-27] 데브옵스팀의 구성



[출처: 아마존 AWS 코리아, 데브옵스 문화와 모범 사례 및 필수도구]

- 수천 개의 자율적 데브옵스팀 구성, MSA 적용, 팀별 CI/CD 파이프라인 자동화를 통한 지속적인 통합 및 배포, 다양한 개발환경 구성 등을 통해 신속한 서비스 배포 체계를 구축하였다. 2020년 기준 연간 1억 9천만 회(매초 6회) 배포를 하고 있다.

[그림 2-28] 클라우드 네이티브 체계 구현



[출처: 아마존 AWS 코리아, 데브옵스 문화와 모범 사례 및 필수도구 & AWS Fargate와 Amazon ECS를 활용한 CI/CD 모범 사례]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.2 해외 민간 부문

2.2.2.2 넷플릭스의 클라우드 네이티브 도입

- 넷플릭스(Netflix)는 2009년 ~ 2015년까지 7년간 IT 인프라를 데이터센터로 이전하였다. 그리고 영상 콘텐츠 로그 및 데이터 기록, 웹 로그인 및 검색 기능, 빅데이터 분석, 결제 기능을 단계적으로 아마존 AWS로 옮겼다.
- 넷플릭스는 2016년 데이터센터의 문을 닫고 "클라우드 네이티브 기업이 되었다."고 선언을 하였다. 기존에 모놀리식 구조로 서비스를 개발하고 있었으나 서비스가 복잡해지고 커지면서 단점이 드러났고, 이를 해결하기 위해 MSA를 적용하여 작은 단위의 서비스를 개발하는 체계로 전환했기 때문이다.

[그림 2-29] 넷플릭스의 클라우드 네이티브 도입 배경

개발속도 저하

- 기능 변경, 신규 기능 추가 시 연관된 모든 코드 수정 필요
- 모든 개발자의 참여가 요구되어 서비스 개발 및 개선 속도가 저하됨

데이터센터 비대화

- 트래픽 증가 시 접속량을 감당할 HW 추가 필요
- 영상 스트리밍이라는 본질보다 데이터센터가 커질 수 있는 상황

서버 점검의 문제

- 일반적인 기업은 새벽 시간에 서버 점검을 하지만 넷플릭스는 다양한 지역에서 24시간 서비스됨
- 서버 점검을 위한 대기시간 적용의 어려움 존재

서비스 중단

- 서비스가 하나의 데이터베이스 위에 만들어졌기 때문에 DB 장애 시 서비스가 중단됨
- 데이터센터 화재 시 3일간 서비스가 마비된 적 있음

[출처 : 카카오TV, 넷플릭스는 어떻게 '클라우드 네이티브'가 됐나]

[그림 2-30] 넷플릭스의 클라우드 네이티브 도입 내용

조직적 변화 – 일하는 방식의 변화

- 클라우드 네이티브 환경에서 애플리케이션을 개발, 운영하기 위해 개발팀에게 권한과 책임을 부여 독립적, 자율적으로 운영되도록 조직문화를 변경

You Build it.

개발조직에 의한 클라우드 실행

You Run it.

데브옵스를 위한 몇 명의 직원

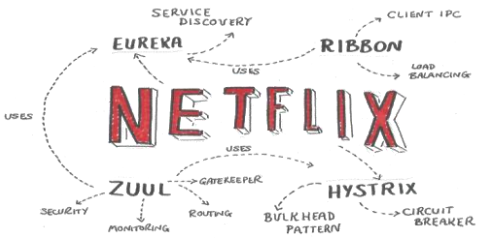
You Support it.

개발자들이 운영환경에 직접 배포

기술적 변화 – 오픈소스 개발, 배포

- 수백 개의 마이크로서비스를 AWS에서 운영, 전 세계 대상으로 24시간 운영
- 자체적인 노하우, 기술역량을 바탕으로 다양한 라이브러리 및 서비스 개발하여 오픈소스로 공개

→ Netflix OSS(Open Source Software)



02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.2 해외 민간 부문

2.2.2.2 넷플릭스의 클라우드 네이티브 도입

- 넷플릭스는 클라우드 네이티브를 도입하는 과정에서 경험한 노하우와 문제해결 방법을 공유하기 위해 관련 기술을 넷플릭스 OSS라는 이름으로 오픈소스로 공개하였다.
- 넷플릭스는 2015년에 스프링 클라우드 넷플릭스(Spring Cloud Netflix)1.0을 출시했고, 2018년부터 스프링 클라우드 넷플릭스를 통한 커뮤니티 산출물을 이용하여 자바 프레임워크를 스프링 부트(Spring Boot)로 전환하였다.

[표 2-4] 넷플릭스의 OSS(오픈소스 SW) 목록

NETFLIX | OSS Spring Cloud

오픈소스	역할	설명
Spring Boot	컨테이너 기반 프레임워크	• 스프링 기반 애플리케이션을 쉽게 구현할 수 있도록 지원하고, 단독 실행이 가능한 내장 웹애플리케이션서버(WAS) 컨테이너를 포함한 프레임워크
Spring Cloud Config	설정 서비스	• 중앙집중식 설정(Config) 서비스로 애플리케이션 설정에 필요한 파일을 마이크로서비스와 분리 저장하는 역할을 담당 • 각 마이크로서비스 설정을 깃(Git) 또는 설정 파일로 저장관리
Spring Security	보안 기능 탑재된 스프링 하위 프레임워크	• 서비스에 접근할 수 있는 인증/인가 처리 프레임워크로 인증서버가 발행한 토큰을 기반으로 서비스 통신 처리 • 자바 스크립트 웹 토큰(JWT, Java script Web Token)을 지원
Spring Sleuth/Zipkin	분산환경 로그 추적	• 슬루쓰(Sleuth)를 이용하여 트랜잭션의 고유 추적 식별자를 부여하며, 집킨(Zipkin)서버를 구성하여 추적 로그를 수집 및 모니터링
Netflix Eureka	서비스 디스커버리	• 서비스 디스커버리(Discovery, 탐색)를 제공하는 오픈소스로, 서버와 클라이언트로 이루어져 있음 • 마이크로서비스 기동 시 자신의 정보를 유레카(Eureka) 서버에 등록하고, 등록된 정보를 다른 클라이언트에게 전달하여 서로 통신이 가능하도록 함
Netflix Zuul	API 게이트웨이	• 마이크로서비스에 대한 서비스 라우팅 기능을 제공 • 모든 호출이 한 곳을 통하도록 서비스의 단일 엔드포인트(End point)를 제공하는 API 게이트웨이(Gateway) 역할 • 공통적인 인증/인가, 필터링, 라우팅 규칙 등을 적용
Netflix Hystrix	서킷 브레이커	• 마이크로서비스의 회복성 패턴을 위한 서킷 브레이커
Netflix Ribbon	클라이언트 측 로드밸런서	• 클라이언트에서 서비스 호출에 대한 분산 처리가 되도록 함
Netflix Turbin	히스트릭스 스트림 통합	• 여러 컨테이너의 히스트릭스(Hystrix) 정보를 통합/조회

[출처: 넷플릭스, 오픈소스 소프트웨어 센터 등]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.3 국내 민간 부문

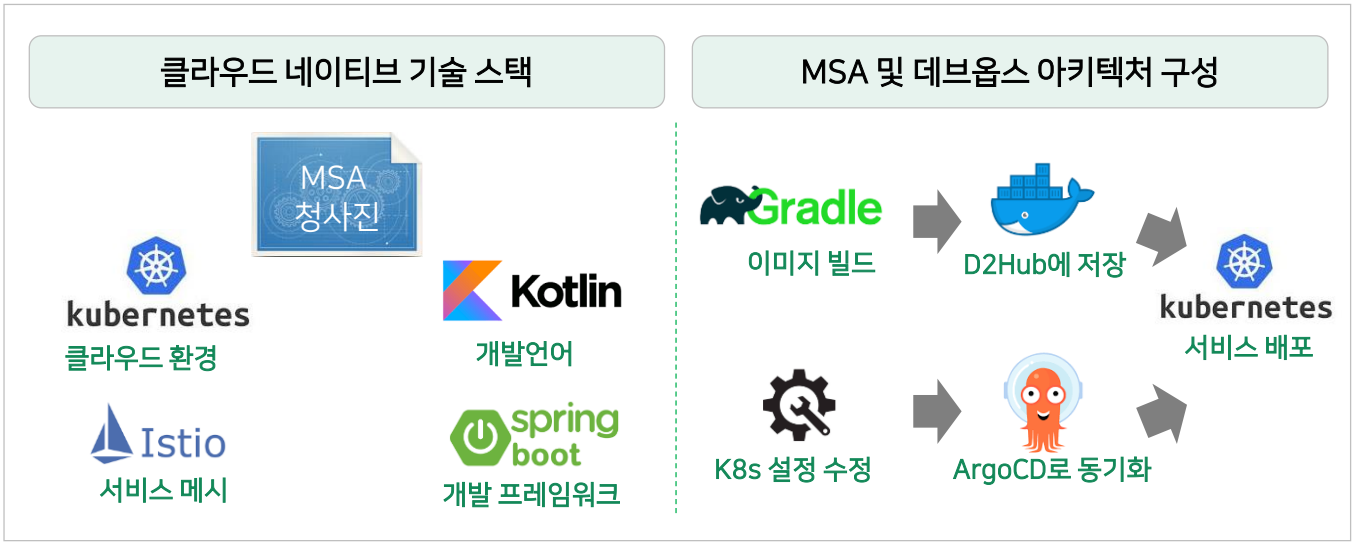
2.2.3.1 카카오 이모티콘서비스

- 카카오 이모티콘서비스는 단일 서버 내 큰 코드베이스 존재, 배포 및 코드 버전 관리 문제, 서비스 설계 변경 등의 문제를 안고 있었으며, 이를 해결하고자 MSA와 데브옵스 아키텍처를 도입하였다.
- MSA와 데브옵스의 도입을 통해 물리적인 서비스 및 코드의 분리, 새로운 배포시스템 구현, 신규 서비스 추가 용이성 확보 등의 효과를 얻게 되었다.

[그림 2-31] 카카오 이모티콘서비스의 클라우드 네이티브 도입 배경 및 효과

클라우드 네이티브 도입 배경	• 코드 이슈 : 단일 서버에 존재하는 큰 코드베이스와 기능의 분할 필요 • 운영 이슈 : 배포의 어려움, 코드의 버전 관리 복잡, 메인 DB 트래픽 증가 • 기술 이슈 : 서비스가 너무 커서 설계 변경의 어려움, 과거 기술에 종속 → 배포 부담을 줄이기 위해 배포 단위를 작게 만들고, 서비스 크기를 줄임
클라우드 네이티브 적용 내용	서비스 배포환경 통합, 간단한 리소스 설정 / 서비스 간 연결관계 관리, 쉬운 모니터링 환경 / 서비스 나누기
클라우드 네이티브 도입 효과	• 물리적인 서비스의 분리, 코드의 분리에 따른 업무 역할 분담의 명확화 • 새로운 배포 시스템으로 배포 부담 감소 • 신규 서비스 기능 추가 용이 및 서비스 민첩성 확보

[그림 2-32] 클라우드 네이티브 기술 스택 및 데브옵스 파이프라인



[출처 : https://tech.kakao.com/2021/09/14/msa/]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.3 국내 민간 부문

2.2.3.2 카카오뱅크 모바일앱

- 카카오뱅크 모바일앱은 변화를 추구하는 개발 조직과 안정을 추구하는 운영 조직 간의 대립과 소통 문제를 해결하고자 데브옵스를 도입하게 되었다.
- 카카오뱅크는 금융권이므로 운영의 안정성도 중요하지만 고객 지향적인 서비스를 제공하고자 변화의 위험을 낮추기 위해 데브옵스 도구를 도입하고 데브옵스 문화를 만들었다.

[그림 2-33] 카카오뱅크 모바일 앱의 데브옵스

[출처 : https://mk-v1.kakaocdn.net/dn/if-kakao/conf2019/conf_video_2019/1_105_03_m1.mp4]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.3 국내 민간 부문

2.2.3.3 11번가

- 11번가 쇼핑몰 시스템은 2016년까지 거대한 모놀리식 구조로 되어 있어서 배포 지연과 장애 문제, 노후화된 기술 스택, 느린 개발환경, 트래픽 폭증 대응 곤란 등의 문제를 안고 있었다.
- 2017년에 업무 도메인별로 서비스를 작게 나누고, 이에 따라 API 서버를 분리하는 등 MSA를 도입하여 개발 및 배포 속도 향상, 주문 및 결제 서비스 속도 향상 및 서비스 장애 감소 등의 효과를 얻었다.

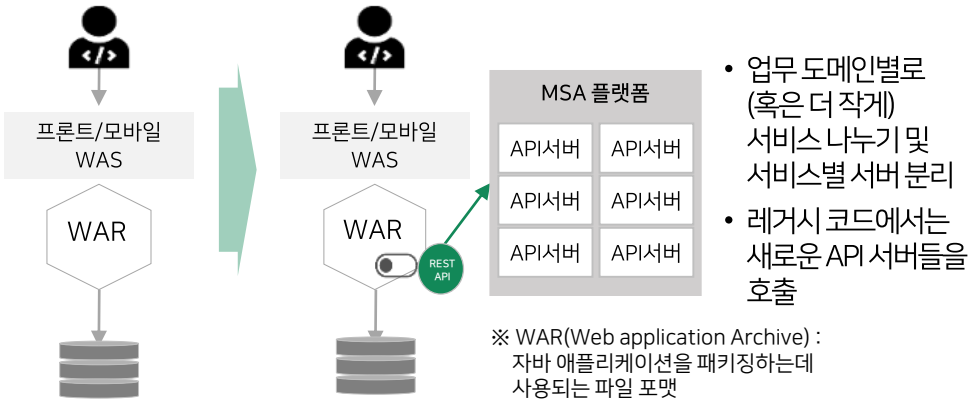
[그림 2-34] 11번가 쇼핑몰의 MSA 전환 배경 및 도입 효과

MSA 도입 배경

- 2016년 말 11번가 쇼핑몰 시스템
  - 거대한 모놀리식 시스템 → 많은 개발팀의 코드를 매주 목요일 정기 배포
  - 8년 이상 사용한 노후화된 SW 스택 → 개발자들이 새로운 기술을 접할 기회 부족
  - 비대해진 200만 라인의 공통 모듈 → IDE(Integrated Development Environment, 통합개발환경)를 띄우기 어려운 개발환경
  - 폭증하는 트래픽으로 수년 내 감당하기 힘들 것으로 예상

MSA 도입 시 변화사항

MSA 도입 전(~ '16년) MSA 도입 후('17년 ~ '18년)



- 업무 도메인별로 (혹은 더 작게) 서비스 나누기 및 서비스별 서버 분리
- 레거시 코드에서는 새로운 API 서버들을 호출

※ WAR(Web application Archive) : 자바 애플리케이션을 패키징하는데 사용되는 파일 포맷

MSA 플랫폼 솔루션 선정

MSA를 위한 선진사례를 빠르게 적용하는 것이 필요

NETFLIX OSS

개발자들이 가장 친숙해 하는 환경

Spring Boot

클라우드 네이티브 도입 효과

- 개발 및 배포 속도 향상뿐만 아니라 주문과 결제 서비스에 대한 응답 속도를 높이고 서비스 장애도 줄였음

[출처 : 11번가 Spring Cloud 기반 MSA로의 전환, https://youtu.be/J-VP0WFEQsY]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.3 국내 민간 부문

2.2.3.4 우아한형제들(배달의민족)

- 배달의민족은 2015년 1일 주문건수 5만 건 이하 수준에서 2020년 약 200만 건으로 주문건수가 폭증하였으며, 주문 증가에 대응하고 장애 대응을 하기 위해 2016년부터 단계적으로 마이크로서비스 아키텍처로 전환하였다. MSA 아키텍처 전환 시 성능, 장애 격리, 데이터 동기화 관련 문제를 해결함으로써 안정적인 서비스를 제공할 수 있게 되었다.

[그림 2-35] 배달의민족 MSA 도입 배경 및 도입 효과

MSA 도입 배경

- '15년 1일 주문건수 5만 건 이하에서 '20년 200만 건으로 매년 주문건수 폭증
- '15년 시스템 현황
  - MS SQL + PHP, ASP
  - 대부분 MS SQL(루비 DB) 스토어드 프로시져 사용, 루비 DB 장애 시 전체 서비스 장애
- '16년 시스템 현황
  - PHP를 자바로 변경
  - 결제, 주문중계 서비스를 독립시키면서 마이크로서비스 도전 시작
  - IDC(Internet Data Center, 인터넷 데이터 센터)에서 AWS 클라우드 인프라로 이전 시작

MSA 전환 (단계별 서비스 분리)

MSA 전환 과정('16년 – '19년)

프론트 서버	회원인증	리뷰
쿠폰	포인트	정산
주문	결제	주문중계
가게/업주	광고	메뉴

- '16년 : 1일 주문건수 10만 건
  - 결제, 주문중계 시스템 분리
- '17년 : 1일 주문건수 20만 건 돌파
  - 메뉴, 정산, 가게 시스템 분리
- '18년 : 광고 증가 → 장애 대응 필요
  - 쿠폰, 포인트, 주문, 리뷰 시스템 분리
- '19년 이후
  - 광고, 가게/업주 시스템 등 분리
  - '19년 말 MSA 전환 완료

MSA 도입 고려사항

성능	장애 격리	데이터 동기화
• 대용량 트래픽 대응 • 일부 시스템 초당 15,000회 API 호출	• 가게, 광고 같은 내부 서비스나 DB 장애 시에도 고객 서비스를 유지하고 주문도 가능해야 함	• 데이터 분산 필요
▼	▼	▼
명령과 조회 서비스 분리	**이벤트 전파와 동기화**	**최소 데이터 보관 원칙**
• 핵심 명령 시스템과 조회 중심 사용자 서비스 분리	• 데이터는 최종적인 일관성 유지(이벤트 발행 방식)	• 데이터 논리적 의존관계 최소화

[출처 : 배달의민족 마이크로서비스 여행기, https://www.youtube.com/watch?v=BnS6343GTkY&t=19s]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

2.2 클라우드 네이티브 도입 사례

2.2.3 국내 민간 부문

2.2.3.5 마켓컬리

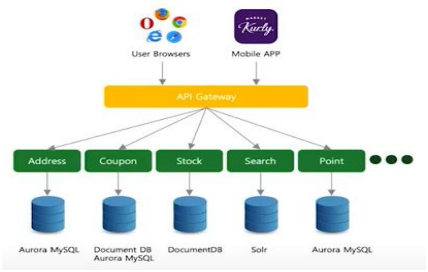
- 마켓컬리는 온프레미스 환경에서 서비스 구성을 위한 고민 해결을 위해 컨테이너, MSA, 데브옵스, CI/CD를 포함하여 클라우드 네이티브를 도입하였다.
- 기존 서비스의 마이크로서비스화 , 컨테이너를 통한 서비스 구성 관리 및 배포 관리, CI/CD를 통한 배포 안정성 확보, 데브옵스 조직 체계에 의한 개발과 운영 효율화를 통해 고객 서비스 대응력이 강화되었다.

[그림 2-36] 마켓컬리의 클라우드 네이티브 도입 배경 및 효과

클라우드 네이티브 도입 배경	• 온프레미스(On-Premise) 환경에서 신규 서비스 구성을 위한 고민을 해결하기 위해 클라우드와 클라우드 네이티브 도입 결정 • 클라우드 환경에 맞게 변화하기 위해 컨테이너, 마이크로서비스, 데브옵스, CI/CD를 도입하여 클라우드 네이티브 환경 구성
클라우드 네이티브 적용 내용	 Containers 서비스 구성관리 및 배포관리  CI / CD 배포 안정성과 빠른 서비스 릴리즈  Microservices 기존 서비스의 마이크로서비스화  DevOps 운영과 개발의 효율화
클라우드 네이티브 도입 효과	• 신속한 고객 요구사항 대응, 업무시간 중 인프라관리 관련 업무 감소, 개발 및 배포 시간 단축, 장애 감소 등

[그림 2-37] 마켓컬리의 MSA 기술 스택 및 데브옵스 파이프라인

마이크로서비스 아키텍처



- 모놀리식 아키텍처를 마이크로서비스 아키텍처로 전환
- 기존 서비스를 주소록, 쿠폰, 재고, 포인트 등 마이크로서비스와 DB로 분리

데브옵스 및 CI/CD 기술 스택

- 운영 환경 : AWS 등
- 개발환경 : RANCHER
- CI/CD : Jenkins, Github, AWS 코드 시리즈 등
- 인프라 코드관리 : Terraform
- 개발 모니터링 : ELK Stack

[출처 : https:// www.youtube.com/watch?v=a2q6eJ0-DKE&t=57s]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

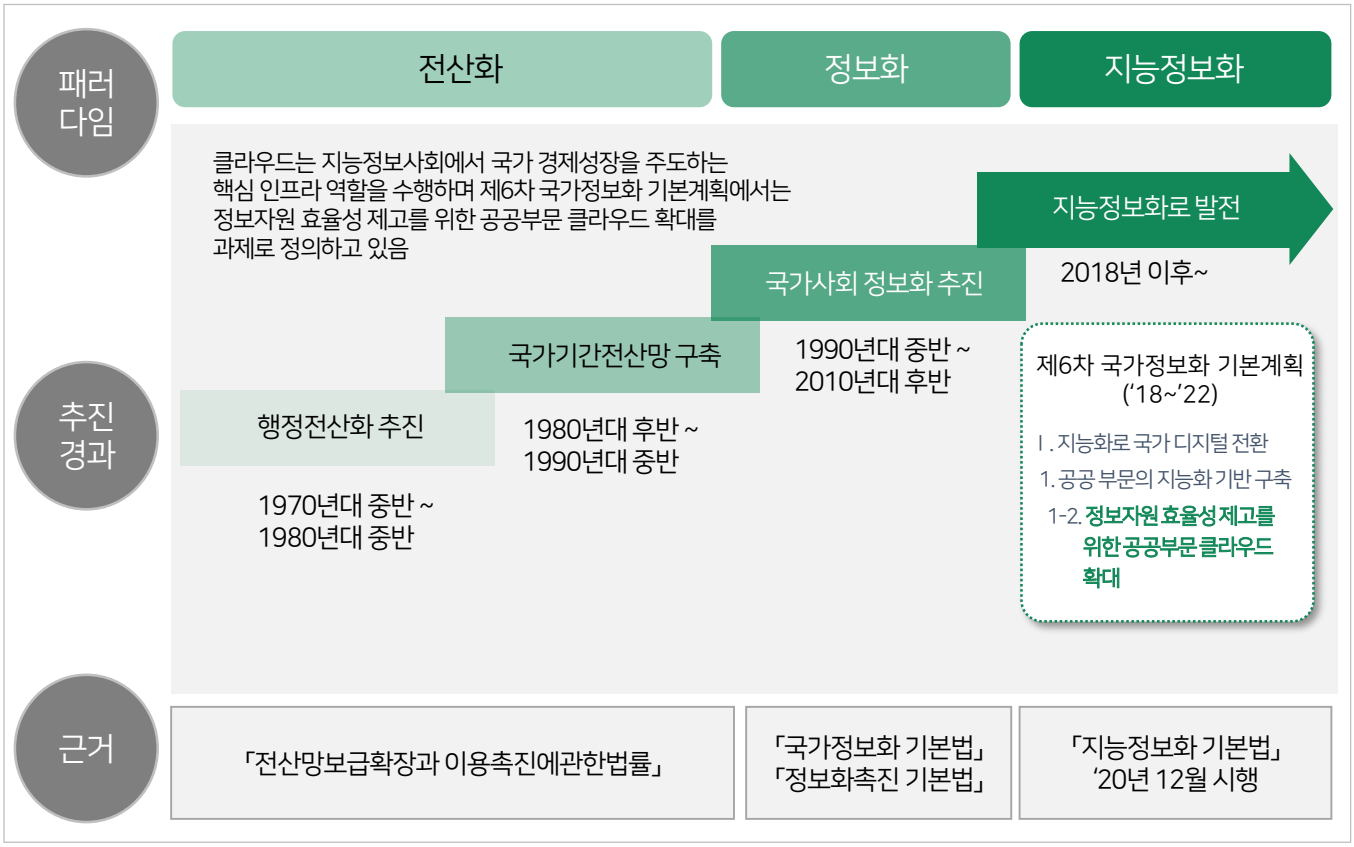
2.3 클라우드 네이티브 도입 필요성

2.3.1 지능정보화의 추진

2.3.1.1 정부의 지능정보화 계획

- 지능정보화란 정보의 생산·유통 또는 활용을 기반으로 지능정보기술이나 그 밖의 다른 기술을 적용 및 융합하여 사회 각 분야의 활동을 가능하게 하거나 그러한 활동을 효율화·고도화하는 것을 말한다.
- 정부는 지능정보화를 추진하기 위해 클라우드, 빅데이터, AI, IoT 등 최신 기술 기반과 산업 생태계를 강화하고, 국가 경쟁력 강화와 국민의 삶의 질 향상에 기여하기 위해 지능정보화 로드맵을 마련하였다.
- 2018년 정부는 4차 산업혁명의 기회를 극대화하고 지능화 혁신의 편익을 제공하도록 5년간의 국가정보화 미래상(비전)을 제시하는 '제6차 국가정보화 기본계획(2018~2022)' 을 수립·발표했다.
- 기본계획은 13개 중점 추진과제와 42개 정보화 세부과제를 제시했으며, 이중 '국민의 삶을 책임지는 지능국가' 수립을 위해 AI, 빅데이터, 클라우드 등 지능정보기술을 적용하는 정보화사업 비중을 2022년까지 35%로 확대한다는 계획이 포함되어 있다.

[그림 2-38] 지능정보화 로드맵



02 클라우드 동향 및 클라우드 네이티브 도입 필요성

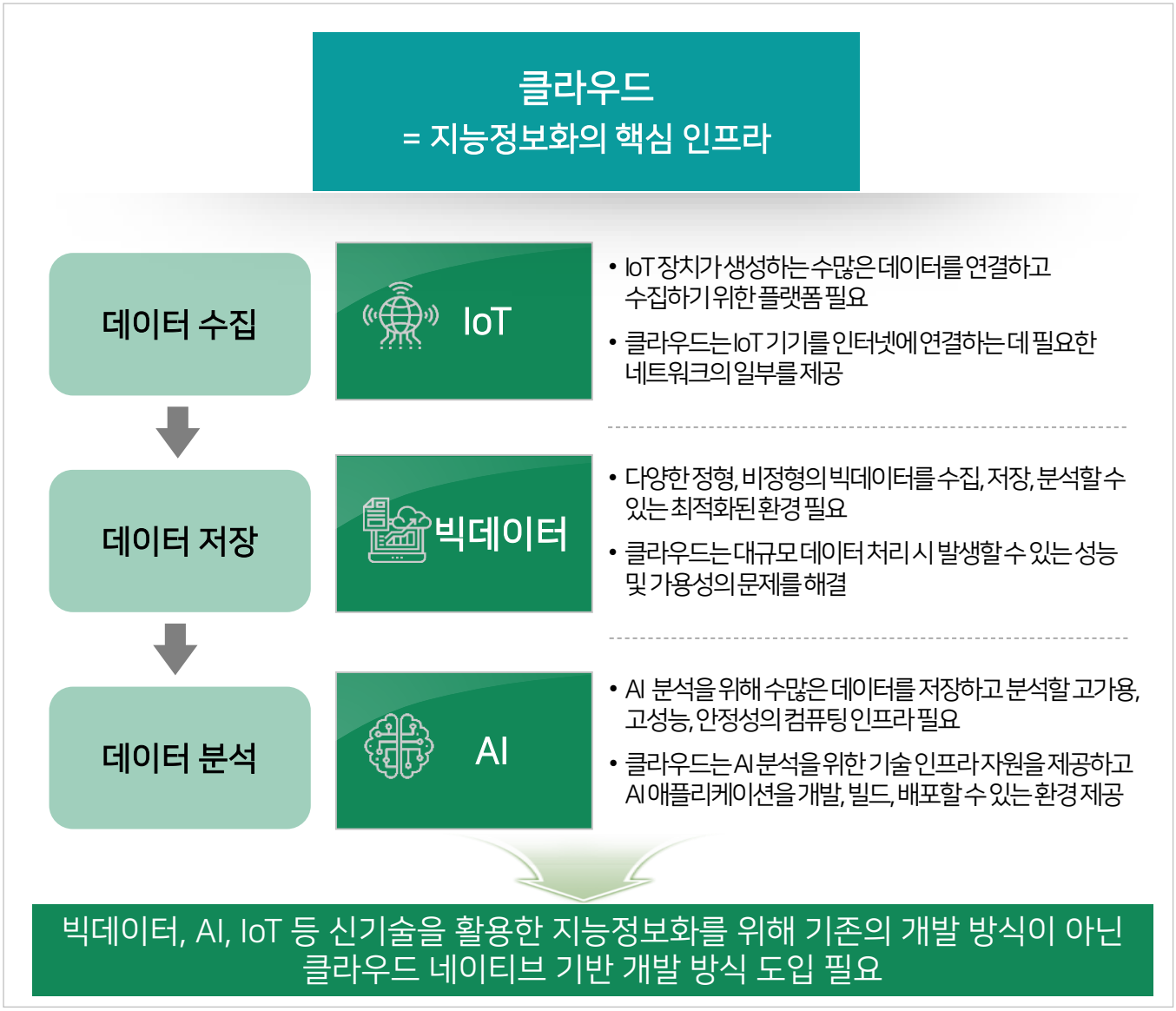
2.3 클라우드 네이티브 도입 필요성

2.3.1 지능정보화의 추진

2.3.1.2 지능정보화를 위한 클라우드 역할

- 클라우드는 인간과 인간, 인간과 사물, 사물과 사물 간의 연결을 기하급수적으로 증가시키는 초연결성 기반의 플랫폼이다.
- 클라우드는 빅데이터, AI, IoT 등 다양한 신기술을 활용한 각종 서비스에 대해 최적의 성능, 안정성, 가용성을 제공하여 지능정보화의 핵심 인프라로서 국가와 산업경쟁력 확보를 위해 지능처리 기반의 데이터의 수집, 저장, 분석 등의 작업을 가능하게 한다.

[그림 2-39] 지능정보화를 위한 클라우드의 역할



02 클라우드 동향 및 클라우드 네이티브 도입 필요성

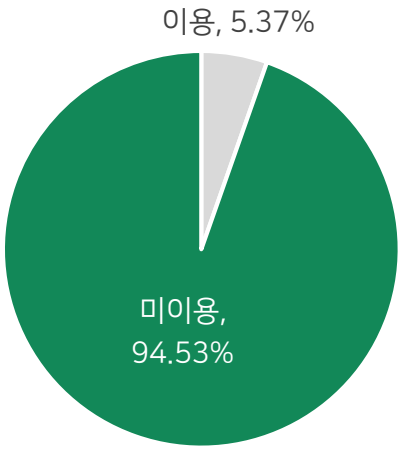
2.3 클라우드 네이티브 도입 필요성

2.3.2 공공부문 정보화 현황

2.3.2.1 정보시스템 클라우드 이용현황(2019년)

- 2019년 공공부문에서 추진한 정보화사업 중 약 5.37%만 클라우드를 이용하고 있으며, 이 중 중앙행정기관의 클라우드 이용률이 21.87%로 가장 높았고, 입사헌법/독립기관의 이용률이 0%로 가장 낮은 것으로 나타났다.

[그림 2-40] 정보시스템 클라우드 이용 여부(2019년)



- '19년 기준, 공공기관의 정보화사업 중 클라우드를 이용한 비율은 5.37%이고, 이용하지 않는 비율은 94.53%임

  → 클라우드 이용도가 낮은 편임

[출처 : 행정안전부, : 2020년도 범정부EA 기반 공공부문 정보자원 현황 통계 보고서]

[표 2-5] 기관 유형별 정보시스템 클라우드 이용현황(2019년)

(단위 : 개, %)

구분	이용		미이용		합계
	시스템 수	비율(%)	시스템 수	비율(%)	
중앙행정기관	381	21.87	1,361	78.13	1,742
입사헌법/독립기관	0	0.00	140	100.00	140
광역자치단체	78	5.07	1,461	94.93	1,539
기초자치단체	77	1.04	7,350	98.96	7,427
공공기관	335	6.23	5,045	93.77	5,380
합계	871	5.37	15,357	94.63	16,228

[출처 : 행정안전부, : 2020년도 범정부EA 기반 공공부문 정보자원 현황 통계 보고서]

02 클라우드 동향 및 클라우드 네이티브 도입 필요성

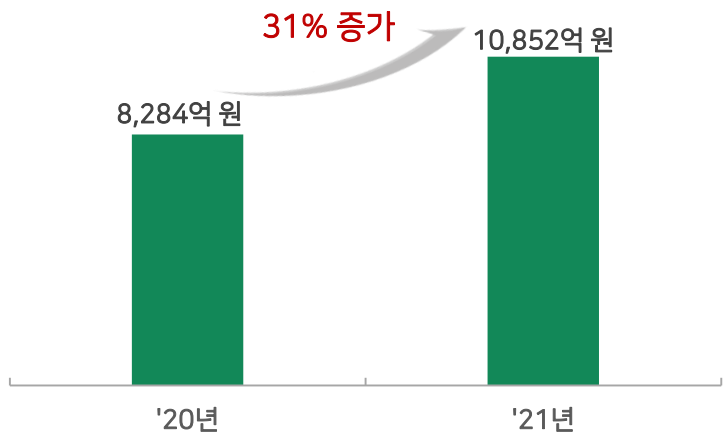
2.3 클라우드 네이티브 도입 필요성

2.3.2 공공부문 정보화 현황

2.3.2.2 정보시스템 클라우드 예산 규모

- 총 56개 중앙부처 등의 정보화사업의 클라우드 예산 규모는 '20년 8,284억 원에서 '21년 10,852억 원으로 약 31% 증가하여 공공부문 전체 예산의 약 10 ~ 12% 수준을 차지하는 것으로 나타났다.

[그림 2-41] '21년 공공부문 클라우드 예산 규모



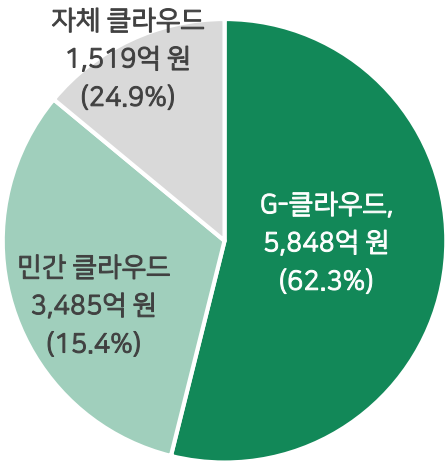
'21년 클라우드 예산은
'20년 대비 31% 증가

대상 : 총 56개 중앙부처, 헌법기관 등이 제출한 '21년도 정보화사업 총 2,864개

[출처 : 한국지능정보사회진흥원, 2021년 국가정보화 시행계획을 위한 공공부문 클라우드 사업규모 조사 결과]

- '21년도 56개 중앙부처의 공공부문 클라우드 예산 중 G-클라우드는 62.3%, 민간 클라우드는 15.4%, 자체 클라우드는 24.9%를 차지하며, 향후 민간 클라우드 활용 비중은 점차 증가할 것으로 예상된다.

[그림 2-42] '21년 공공부문 클라우드 유형별 예산 규모



'21년 총 클라우드 예산
10,852억 원 중
민간 클라우드 이용 규모는
3,485억 원(15.4%) 수준

대상 : 총 56개 중앙부처, 헌법기관 등이 제출한 '21년도 정보화사업 총 2,864개

[출처 : 한국지능정보사회진흥원, 2021년 국가정보화 시행계획을 위한 공공부문 클라우드 사업규모 조사 결과]

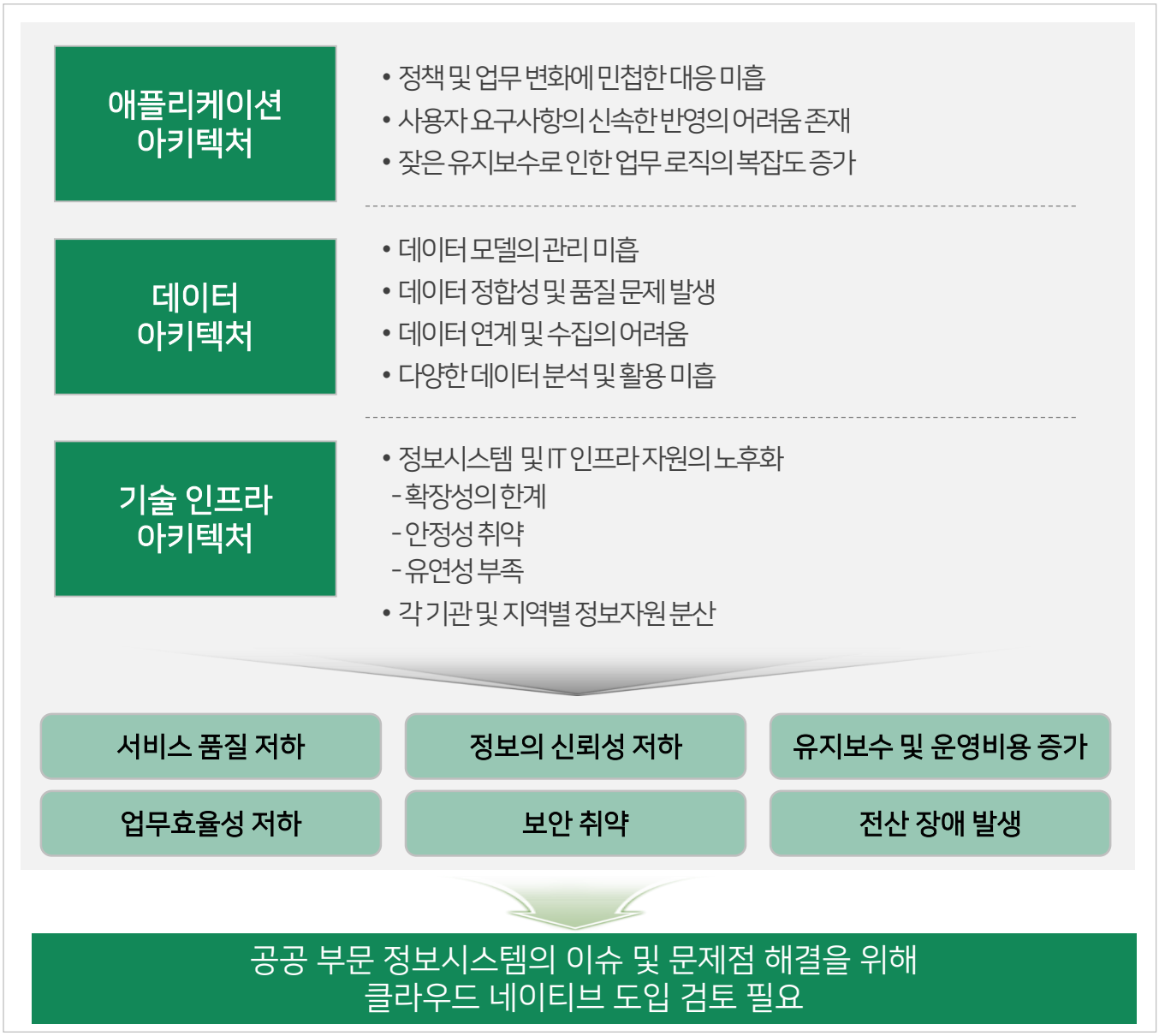
2.3 클라우드 네이티브 도입 필요성

2.3.2 공공부문 정보화 현황

2.3.2.3 정보시스템 주요 이슈 및 문제점

- 공공부문 정보화사업계획에 포함된 각 기관의 정보시스템은 공통적으로 기관 간 애플리케이션, 데이터, 기술 인프라 관점의 유사한 이슈 및 문제점을 가지고 있다.
- 주요 이슈 및 문제점은 서비스 품질 저하, 업무 효율성 저하, 데이터 품질 문제, 보안 취약성, 유지보수 및 규모 증가, 전산 장애 발생 등으로 요약된다.

[그림 2-43] 공공부문 정보시스템 주요 이슈 및 문제점



02 클라우드 동향 및 클라우드 네이티브 도입 필요성

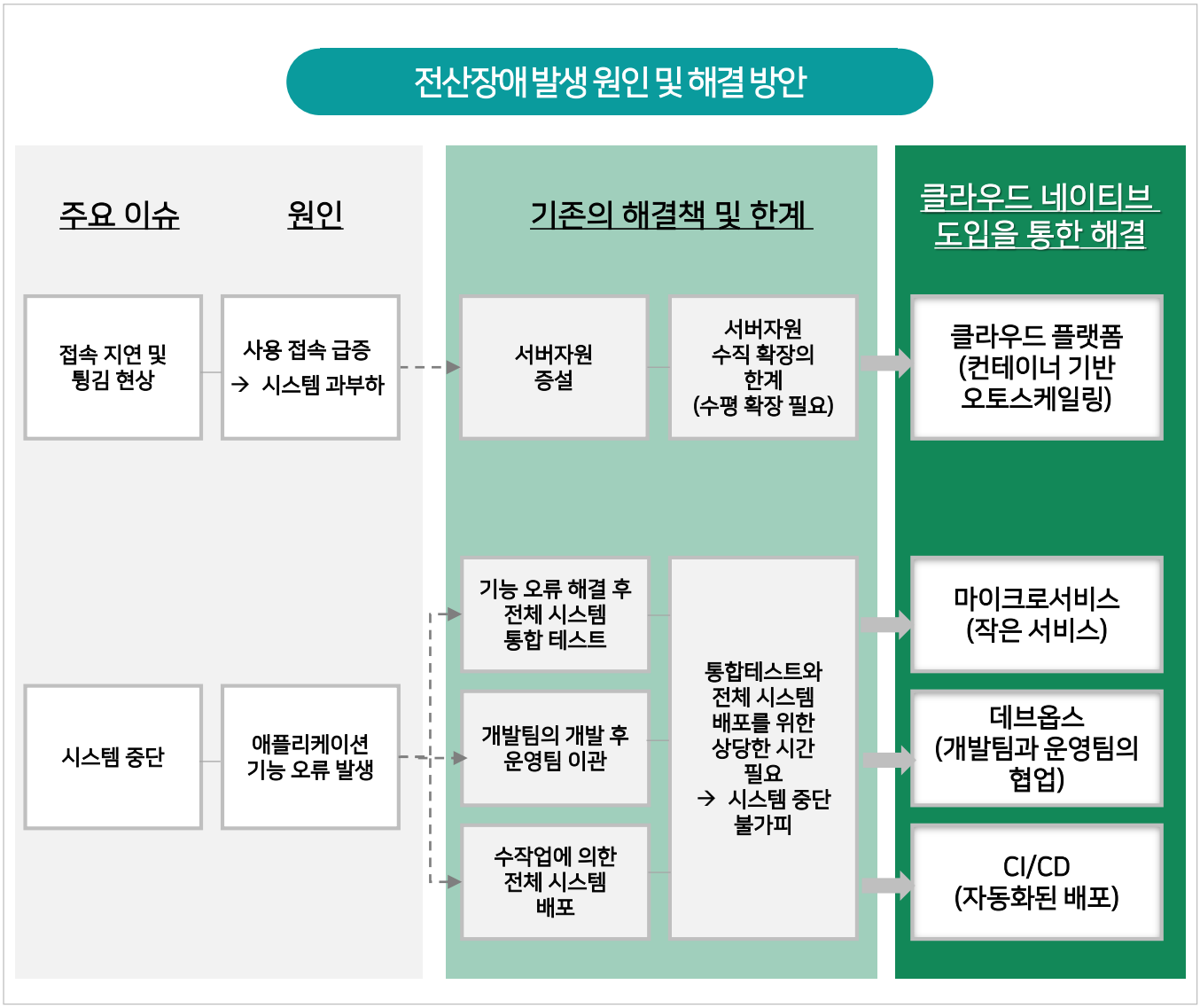
2.3 클라우드 네이티브 도입 필요성

2.3.2 공공부문 정보화 현황

2.3.2.4 클라우드 네이티브 도입을 통한 해결방안

- 공공부문에서 각종 대국민 서비스 제공 시 전산장애가 발생할 경우 수직확장(Scale-up) 방식의 자원 증설, 애플리케이션 기능 오류 해결 등의 작업을 수행하고 있다. 하지만 수직확장의 한계, 신속한 오류 해결의 어려움 등의 문제점은 여전히 존재한다.
- 이러한 문제점을 해결하기 위해 컨테이너 기반 클라우드 플랫폼, MSA, 데브옵스, CI/CD를 포함한 클라우드 네이티브 기술 도입이 필요하다.

[그림 2-44] 클라우드 네이티브 도입을 통한 전산장애 해결 방안



클라우드 네이티브 정보시스템 구축을 위한

발주자 안내서

03

클라우드 네이티브 구성요소 및 원칙

03 클라우드 네이티브 구성요소 및 원칙

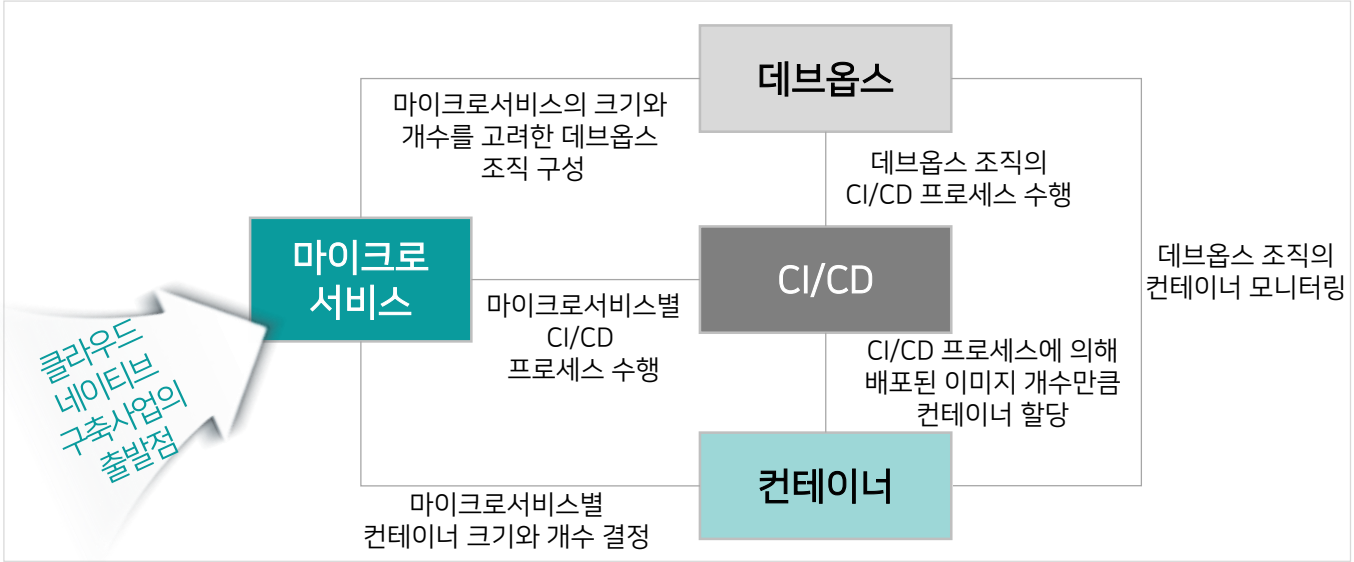
3.1 개요

3.1.1 클라우드 네이티브 구성요소 및 원칙

- 클라우드 네이티브 구성요소는 마이크로서비스, 컨테이너, 데브옵스, CI/CD이고. 클라우드 네이티브 애플리케이션 개발 시 적용되는 개발방법론은 애자일(Agile), 원칙은 12가지 요소(12 Factors)이다.

[그림 3-1] 클라우드 네이티브 구성요소 및 원칙

[그림 3-2] 클라우드 네이티브 구성요소의 연관도



03 클라우드 네이티브 구성요소 및 원칙

3.2 마이크로서비스

3.2.1 마이크로서비스 아키텍처 정의

- 마이크로서비스 아키텍처(MSA, Microservice Architecture)란 하나로 구성된 애플리케이션을 여러 개의 작고 느슨한 형태의 서비스로 변경과 조합이 가능한 아키텍처이며, 독립적으로 서비스하고 배포할 수 있도록 구성된다.
- 마이크로서비스는 최근 REST API의 일반화, 도커와 같은 컨테이너 기술, 클라우드 컴퓨팅 환경의 발전 등에 힘입어 좀 더 손쉽게 구현될 수 있게 되었다.
- MSA 전문가들의 마이크로서비스 아키텍처에 대한 정의는 다음과 같다.

[그림 3-3] 참고. 전문가의 마이크로서비스 아키텍처 정의

아드리안 콕크로프트 (Adrian Cockcroft)

아마존 AWS, 클라우드 아키텍처 전략 부사장(2016.10 ~)
넷플릭스, 클라우드 아키텍처 담당(2007.5~2013.12)

마이크로서비스는 명확한 경계를 가진 느슨하게 결합된 서비스 지향 아키텍처

Microservices are loosely-coupled service oriented architecture with bounded contexts

느슨한 결합(Loosely-coupled)
서비스는 독자적으로 업데이트되며, 서로 영향을 주지 않음

서비스 지향 아키텍처(Service oriented architecture)
서비스들이 네트워크를 통해 서로 API 통신

명확한 경계(Bounded contexts)
서비스는 도메인, 비즈니스의 경계를 갖고 있음

마틴 파울러 (Martin Fowler)

영국의 소프트웨어 개발자, 저술가

마이크로서비스는 분산개발모델을 활용하여 지속적으로 사용 가능한 분산시스템을 구축하기 위한 아키텍처

Microservices is an architecture for building a continuously available distributed system with a distributed development model.

- 마이크로서비스 아키텍처 스타일은 **단일 애플리케이션을 소규모 서비스로 개발하는 방법으로**, 각각 **자체 프로세스에서 실행**되고 **가벼운 메커니즘 통신을 함**
- 종종 HTTP 리소스 API와 같은 마이크로서비스는 비즈니스 기능을 중심으로 구축되며 **완전 자동화된 배포 도구를 통해 독립적으로 배포**할 수 있음
- 마이크로서비스는 최소한의 중앙집중식으로 관리가 되며, **다른 프로그래밍 언어로 작성되고 다른 데이터 저장 기술을 사용**할 수 있음

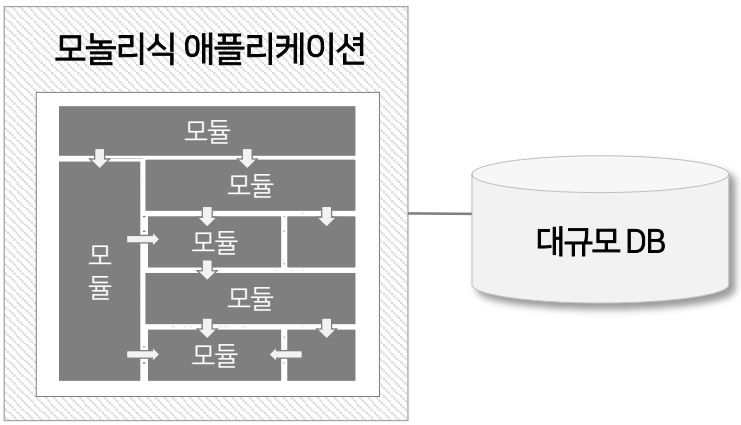
03 클라우드 네이티브 구성요소 및 원칙

3.2 마이크로서비스

3.2.2 마이크로서비스 아키텍처와 모놀리식 아키텍처의 차이점

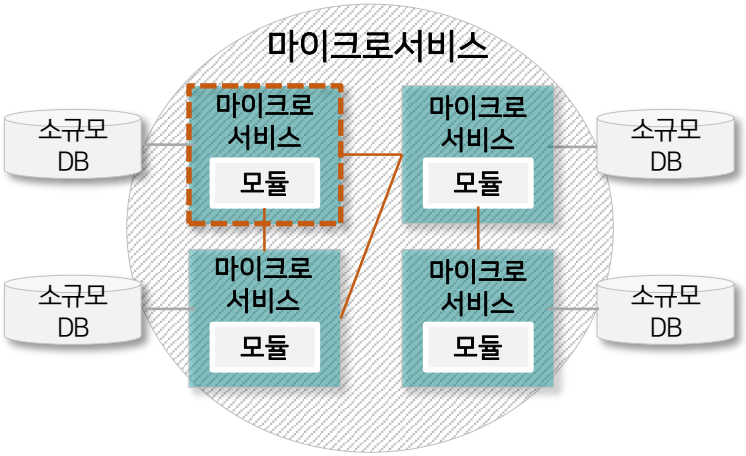
- 기존의 모놀리식 아키텍처는 큰 규모의 다수 모듈로 구성된 단일 애플리케이션과 대규모 DB로 구성되고, 애플리케이션 내부 모든 모듈의 데이터는 단일한 대규모 데이터베이스에 저장된다.
- 마이크로서비스 아키텍처는 여러 개의 서비스가 각각 개별 DB를 갖고 API로 통신하는 구조이며, 마이크로서비스 단위별로 배포되고 독립적으로 실행·운영된다.

[그림 3-4] 모놀리식(Monolithic) 아키텍처



- 큰 규모의 다수 기능(모듈)으로 구성된 단일 애플리케이션
- 애플리케이션 내부 모든 모듈(영역)의 데이터는 단일한 데이터베이스에 저장
  - 하나의 애플리케이션 안에서 모듈 간 직접 호출이 이뤄지고 하나의 대규모 단일 DB를 공유하는 구조

[그림 3-5] 마이크로서비스 아키텍처



- 기존 애플리케이션 모듈(기능, 하위 업무 등) 단위를 마이크로서비스로 정의
  - 여러 개의 서비스가 각각 개별 DB를 갖고 API로 통신하는 구조로 변화(API를 통해 다른 서비스에 접근하고 외부와 연계함)
- 마이크로서비스 단위별로 분리 배포, 독립적으로 실행·운영
- 마이크로서비스는 소규모의 개발팀 단위로 자율적으로 운영

03 클라우드 네이티브 구성요소 및 원칙

3.2 마이크로서비스

3.2.2 마이크로서비스 아키텍처와 모놀리식 아키텍처의 차이점

- 마이크로서비스 아키텍처는 변경사항의 신속한 적용, 트래픽 증가에 유연한 대응, 다양한 기술 적용 등의 장점이 있지만, 분산 아키텍처 구조로 인해 인터페이스가 증가하고 데이터 처리 구조가 복잡해지는 단점도 있다.

[표 3-1] 마이크로서비스 아키텍처와 모놀리식 아키텍처의 장단점

구분	마이크로서비스 아키텍처	모놀리식 아키텍처
장점	• 독립적인 작은 서비스들을 느슨한 결합으로 분산 구성하기 때문에 변경사항을 유연하고 빠르게 적용 가능 • 마이크로서비스는 각각의 서비스를 개별적으로 확장하거나, 축소할 수 있어 트래픽 변화에 유연하게 대응 가능 • 서비스별 플랫폼/신기술의 도입 및 확장이 용이	• 하나의 애플리케이션으로 구성되기에 배포 및 테스트 용이 • 프로세스 내 모듈 간 호출 방식으로 원격 호출 인터페이스로 인한 성능 저하 없음 • 통합 DB 구조로 데이터의 일관성 유지가 쉬움
단점	• 네트워크 통신 구간의 증가에 따른 성능 지연 요소 존재 • 서비스 간 데이터 처리의 독립성으로 트랜잭션 처리가 복잡함 • 서비스 간 연계와 서로 다른 다양한 기술의 사용으로 테스트, 장애추적, 모니터링이 어려움	• 일체형 애플리케이션이기에 일부 모듈의 변경 사항에도 전체 애플리케이션 개발/운영 프로세스와 패키징에 영향을 줌 • 컴포넌트별, 기능별 특성에 맞는 신기술 또는 구조를 적용하기 어려움 • 모듈단위 개별 확장이 어려움

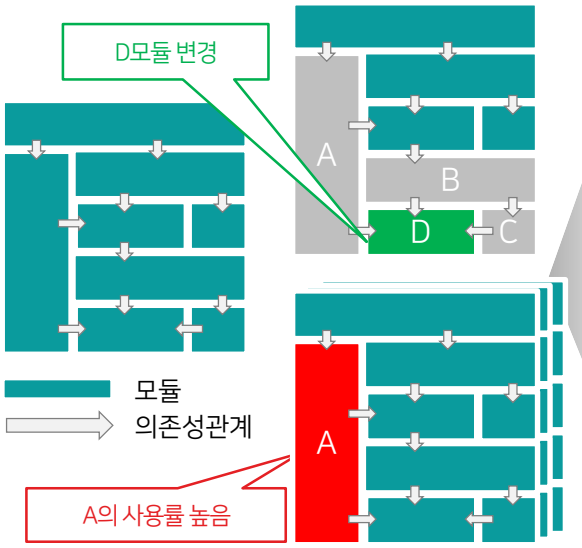
03 클라우드 네이티브 구성요소 및 원칙

3.2 마이크로서비스

3.2.3 마이크로서비스 아키텍처의 필요성

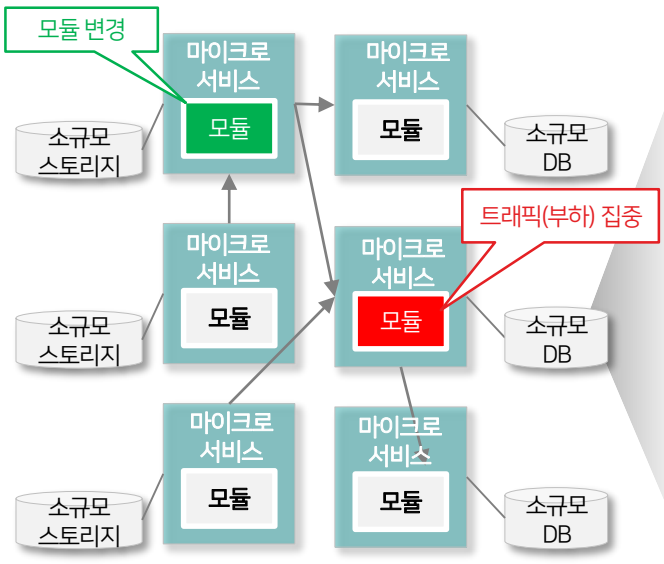
- 클라우드와 같은 유연한 인프라 환경의 확산과 더불어 기존 모놀리식 애플리케이션이 근본적으로 확장성, 장애 전파, 규모의 증가에 따른 운영상 어려움 등의 문제점들을 가지고 있다.
- 모놀리식 아키텍처의 한계를 해결하기 위해 마이크로서비스 아키텍처로 전환함으로써 애플리케이션 개발 시 신속한 업데이트, 독립적인 확장성, 높은 가용성, 성능 향상 등의 효과를 얻을 수 있다.

[그림 3-6] 모놀리식 아키텍처의 한계



- D모듈 변경 시 A,B,C모듈의 추가 테스트 필요
- D모듈의 작은 변화도 전체 애플리케이션 배포 요구
- D모듈만 테스트하고 릴리즈 할 수 없음
- A, B, C, D모듈이 동시에 변경될 경우 부가적인 작업 필요
- 결론적으로 매우 느린 릴리즈 사이클 형성
- A모듈이 집중적으로 많이 사용되는 경우에도 어쩔 수 없이 전체 애플리케이션을 확장
- A모듈이 문제가 생기면 나머지 모듈에도 장애의 영향이 미칠 수 있음

[그림 3-7] 마이크로서비스 아키텍처 전환 이유



- 여러 팀이 독립적으로 릴리즈 사이클을 결정하며, 요구사항을 더 빠르게 반영 — 신속한 업데이트
- 부하가 발생하는 모듈만 독립적으로 스케일-아웃/인이 가능한 구조 — 독립적인 확장성
- 장애를 하나의 모듈에만 고립시켜, 전체적인 서비스에는 영향을 미치지 않음 — 높은 가용성
- 애플리케이션의 사이즈가 축소됨에 따라 시작 시간과 스케일 시간이 단축 — 성능과 유저 경험

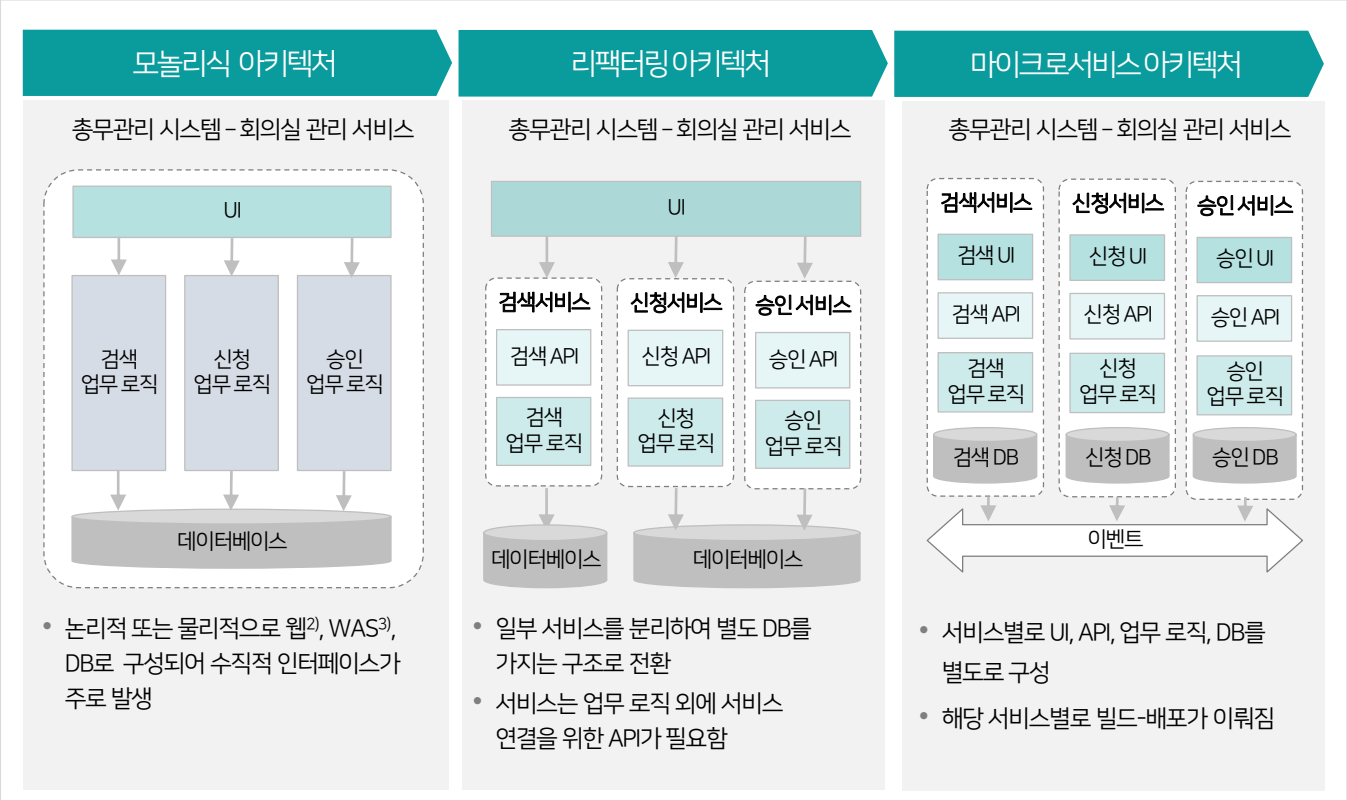
03 클라우드 네이티브 구성요소 및 원칙

3.2 마이크로서비스

3.2.4 마이크로서비스 아키텍처의 구성

- 모놀리식 아키텍처는 논리적, 물리적 측면에서 웹, WAS, DB로 구성되어 수직적 인터페이스가 주로 발생한다.
- 모놀리식 아키텍처에서 마이크로서비스 아키텍처로 전환하는 과도기 과정에 리팩터링[1](Refactoring) 아키텍처를 가져갈 수 있다. 리팩터링 아키텍처는 일부 서비스에 대해 별도 DB를 가지고, 기존의 업무 로직 서비스 외에 서비스 간 연계를 위한 API가 필요하다.
- 마이크로서비스 아키텍처로 전환하게 되면 서비스별로 UI, API, 업무 로직, DB가 별도로 구성되고, 해당 서비스별로 빌드와 배포가 이뤄지게 된다.

[그림 3-8] 마이크로서비스 아키텍처로의 전환 예시



신속한 서비스 제공이 필요한 업무에 대해 마이크로서비스 아키텍처를 설계하고, 변화가 적은 업무는 기존 아키텍처를 유지하도록 함

1) 리팩터링(Refactoring) : SW 공학 관점에서 결과의 변경 없이 코드의 구조를 재조정함을 뜻함
2) 웹(WEB, World Wide Web) : 인터넷에 연결된 컴퓨터를 통해 정보를 공유할 수 있는 공간을 의미하고, 웹 서버는 웹 브라우저에 요청하는 문서나 오브젝트를 전송해주는 역할
3) WAS(Web Application Server) : DB 조회나 다양한 로직 처리를 요구하는 동적인 컨텐츠를 위해 만들어진 애플리케이션 서버

3.2 마이크로서비스

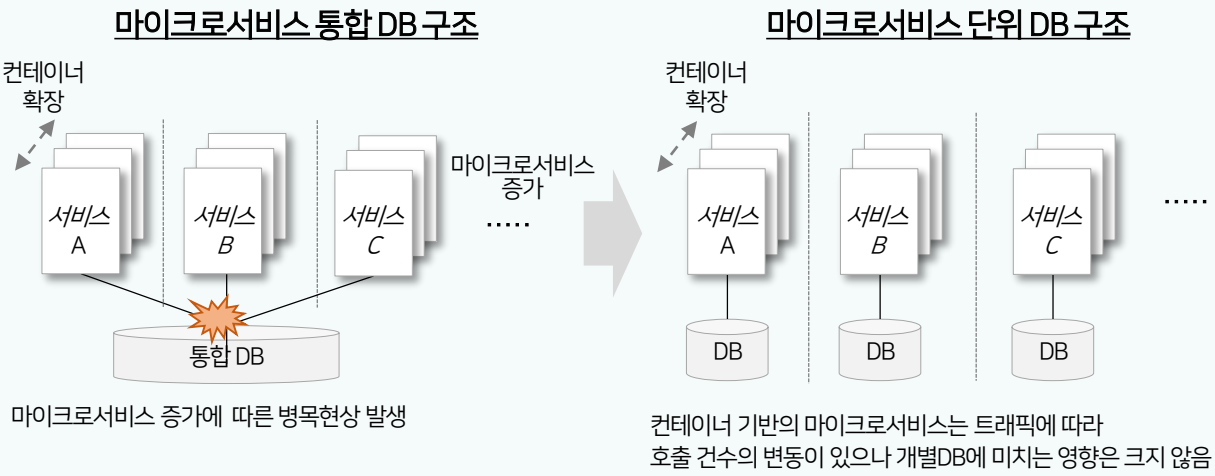
3.2.4 마이크로서비스 아키텍처의 구성

- 기존의 모놀리식 아키텍처는 통합 DB를 사용하므로 마이크로서비스의 개수가 증가할 경우 컨테이너가 증가하여 통합DB에 요청건수가 늘어나 병목현상이 발생할 수 있으므로 이를 해결하기 위해 마이크로서비스 단위의 DB 구성을 원칙으로 한다.

[그림 3-9] 참고. 클라우드 네이티브 환경에서의 DB 구성

마이크로서비스 DB 분리 필요성

- 기존의 모놀리식 아키텍처에서 단일한 통합 DB를 유지한 채 MSA로 전환할 경우의 문제점
  - 마이크로서비스 개수의 증가 시 트래픽의 증가에 의해 컨테이너 동적할당(Scale-out)이 이뤄지고, 이에 따라 컨테이너가 증가할 경우 통합 DB는 늘어난 요청에 의해 심각한 병목현상 발생 가능
  - 또한 통합된 DB구조로 서비스 간 독립적인 배포,운영이라는 MSA의 장점을 살릴 수 없음
- 이러한 문제점을 해결하기 위해 MSA에서는 마이크로서비스 단위의 DB 구성을 원칙으로 함



클라우드 네이티브 환경에서의 DB 구축 방안

1안) 기존의 DB 라이선스를 활용하여 직접 DB를 구축, 운영 하는 방식
- 마이크로서비스 단위로 DB분리가 권고되기 때문에 시스템 규모보다 커질수록 라이선스 비용 추가 뿐만 아니라 늘어난 DB를 직접 관리해야 하는 부담이 있음

2안) 클라우드에서 제공하는 완전 관리형 DB 서비스 사용하는 방식
- 관리형 DB서비스의 경우, 상용 SW보다 오픈소스 기반의 SW를 주로 사용
- 고가의 고가용성 제품이 아니므로 Read-Write DB 분리 등으로 성능을 확보

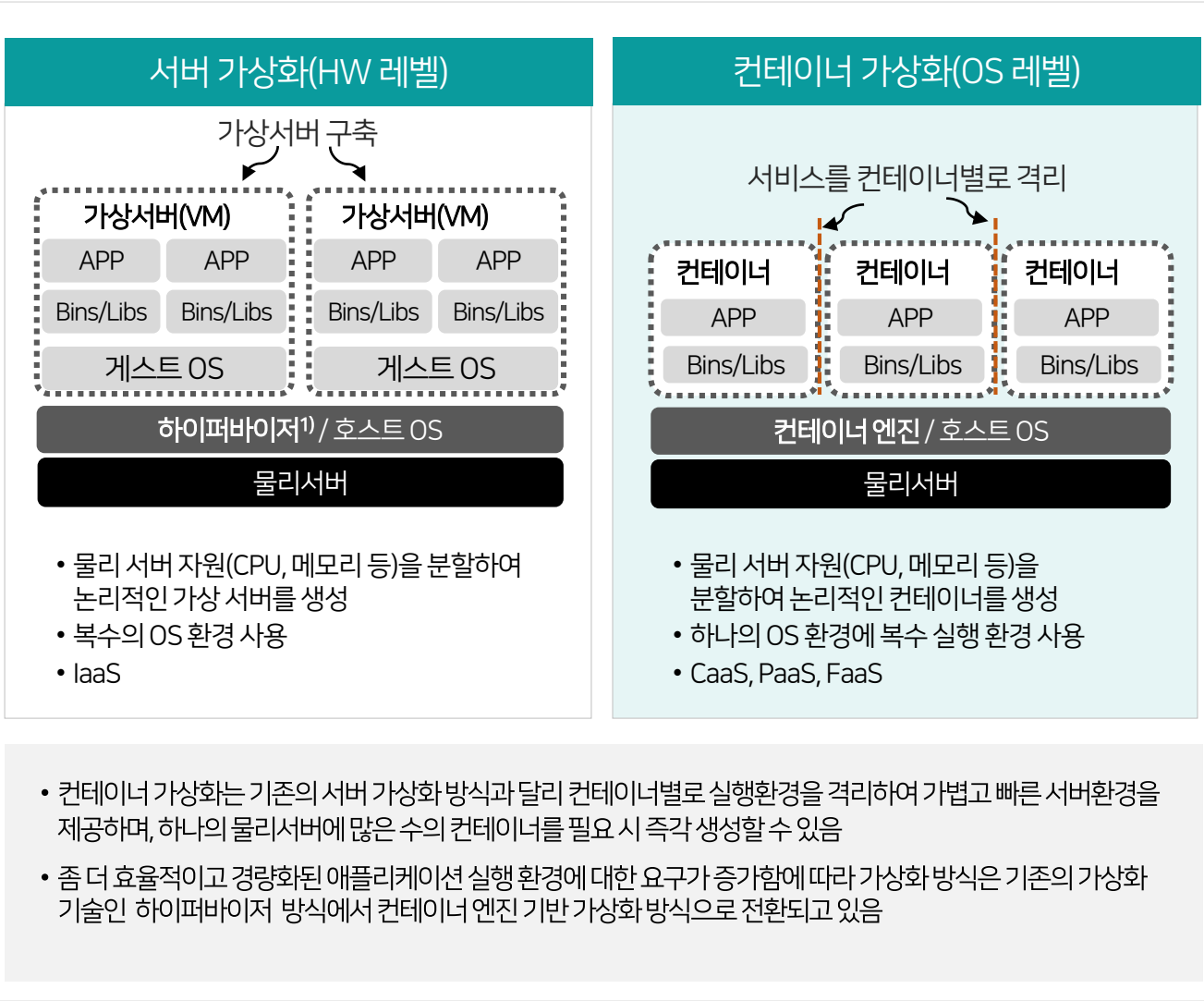
03 클라우드 네이티브 구성요소 및 원칙

3.3 컨테이너

3.3.1 컨테이너 정의

- 컨테이너는 어떤 환경에서도 실행 가능한 경량화된 소프트웨어 패키지이며, 애플리케이션 코드, 런타임 모듈, 라이브러리 등의 모든 요소를 포함한다.
- 컨테이너는 개발자와 IT 전문가가 여러 환경에서 애플리케이션을 원활하게 배포할 수 있도록 한다.
- 실제 해운항만물류업에서 컨테이너를 사용하여 선박 및 각종 운송 수단을 통해 운송할 다양한 화물을 분리하는 것처럼 IT 분야에서도 컨테이너화라는 접근방식이 점점 더 많이 사용되고 있다.

[그림 3-10] 서버 가상화와 컨테이너 가상화 기술 비교



1) 하이퍼바이저(Hypervisor) : 가상머신(VM)을 생성하고 구동하는 소프트웨어로, 서로 다른 여러 개의 운영체제를 나란히 구동할 수 있음

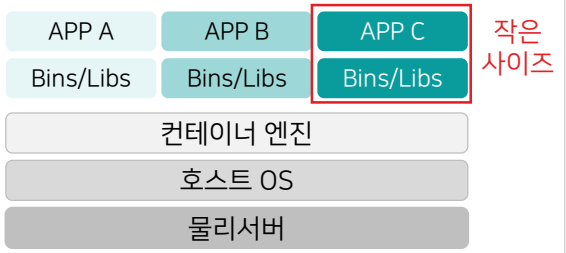
03 클라우드 네이티브 구성요소 및 원칙

3.3 컨테이너

3.3.2 컨테이너와 가상머신의 차이

- 컨테이너와 가상머신의 개념, 구성, 이미지 사이즈, 시작 소요시간, 보안, 시스템 부하, 성능 등의 관점에서 차이점은 다음과 같다.

[표 3-2] 컨테이너와 가상머신의 차이

구분	컨테이너	가상머신(VM)
개념	• 하이퍼바이저 없이 리눅스 컨테이너기술을 바탕으로 애플리케이션을 격리된 상태에서 실행하는 가상화 솔루션	• 컴퓨팅 환경을 소프트웨어로 구현 • 컴퓨터를 에뮬레이션[1]하는 소프트웨어로 가상머신 상에서 운영체제나 응용프로그램을 설치 및 실행
구성		
기본 이미지 사이즈	• MB (VM보다 경량)	• GB
시작 소요시간	• 즉각 (VM보다 빠름)	• 수 분 이내
보안	• 커널 취약점 공유	• 컨테이너보다 보안이 강함
기반	• 호스트 커널	• 하이퍼바이저
HOST OS	• 일반 리눅스, CoreOS, Atomic(RedHat), Photon(V사) 등	• 일반적인 리눅스, 윈도우즈
종류	• 도커, LXC	• KVM, V사, Xen, Hyper-V, Virtual Box
시스템 부하	• 낮음 (VM보다 낮음)	• 높음
성능	• 우위 (VM보다 우수)	• 하이퍼바이저로 인한 성능저하 발생
구동 애플리케이션 수	• 많음	• 적음

1) 에뮬레이션(emulation) : 완전히 똑같은 방법으로 다른 대상을 하드웨어 동작까지도 흉내내는 것을 말함

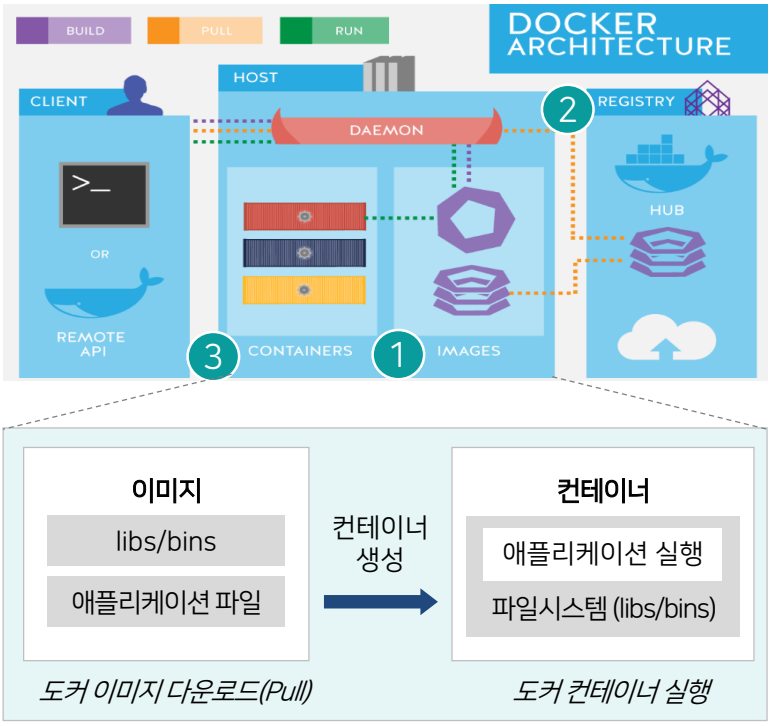
03 클라우드 네이티브 구성요소 및 원칙

3.3 컨테이너

3.3.3 도커(Docker)

- 도커(Docker)는 컨테이너 기반의 오픈소스 가상화 플랫폼으로 컨테이너를 관리하고, 애플리케이션을 신속하게 개발, 테스트, 배포할 수 있는 플랫폼이다.
- 소프트웨어를 컨테이너라는 표준화된 유닛으로 패키징하며, 컨테이너는 라이브러리, 시스템 도구, 코드 등 소프트웨어 실행에 필요한 모든 것을 포함한다.
- 도커는 컨테이너 환경에서 독립적으로 애플리케이션을 실행할 수 있도록 컨테이너를 만들고 관리하는 것을 도와준다.
- 도커를 통해 애플리케이션을 실행하면 독립적인 환경에서 일관된 결과를 보장한다.
- 도커는 이미지, 이미지 저장소, 컨테이너 실행환경으로 구성된다.

[그림 3-11] 도커 아키텍처



1 도커 이미지

- 컨테이너 실행에 필요한 모든 파일과 설정 값 등을 포함하고 있는 실행파일 집합
- 개발 바이너리와 환경 설정을 패키징
- 도커 파일로 이미지 생성 과정을 작성한 후 빌드 명령어를 통해 이미지 생성

2 이미지 저장소(Registry)

- 생성한 도커 이미지의 저장, 공유를 위한 저장소
- 개발자가 도커 이미지를 업로드(Push)하고, 도커 실행 시 다운로드(Pull)
- 공개 도커 저장소(hub.docker.com)와 사설 도커 저장소 이용 가능

3 컨테이너 실행환경

- 도커 이미지를 기반으로 실행되는 일반 리눅스 컨테이너
- 컨테이너는 도커 실행 호스트에서 실행되는 하나의 프로세스
- 컨테이너는 호스트 및 타 프로세스와 격리

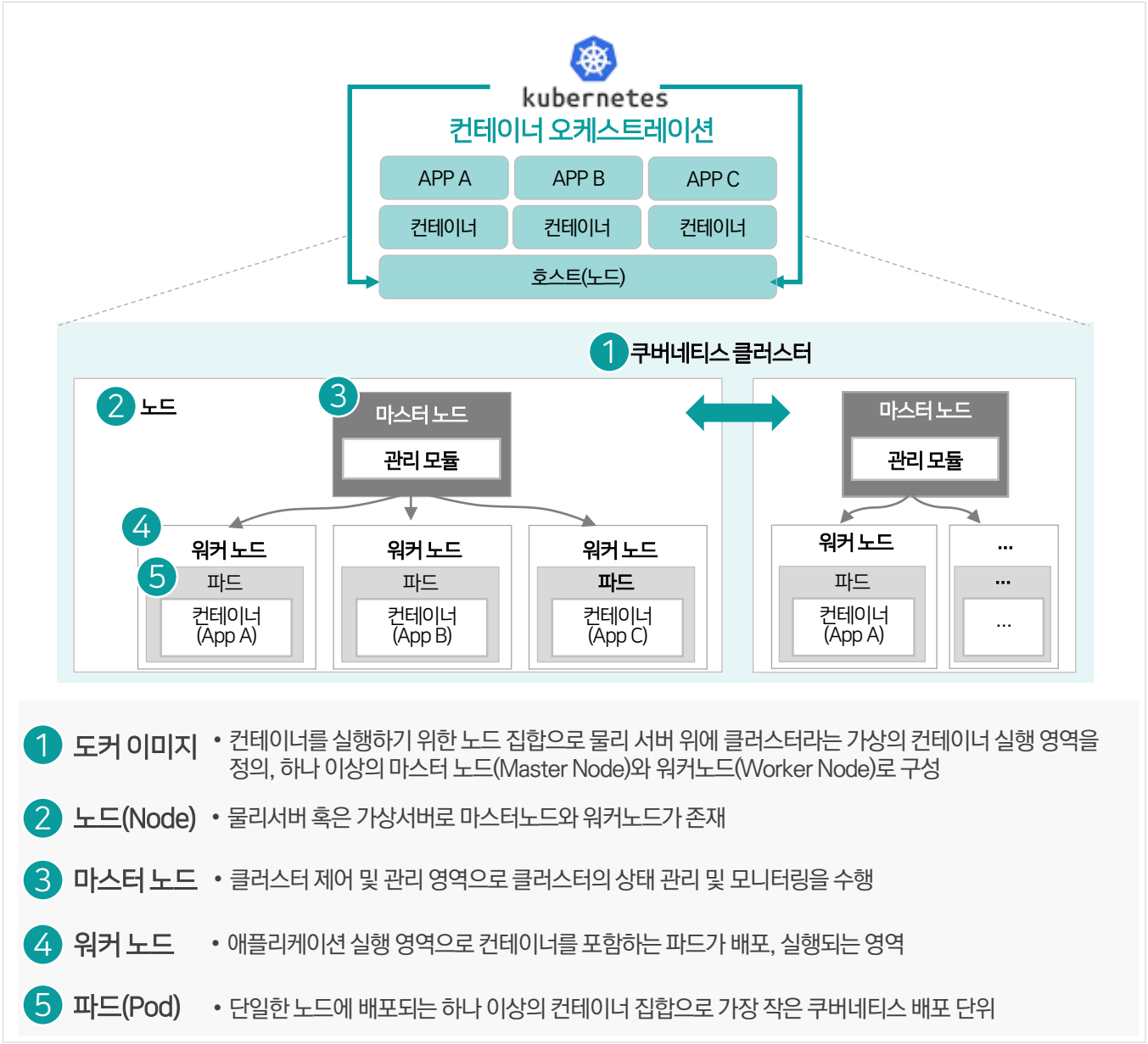
03 클라우드 네이티브 구성요소 및 원칙

3.3 컨테이너

3.3.4 쿠버네티스(Kubernetes)

- 쿠버네티스(Kubernetes)는 컨테이너화된 애플리케이션을 자동으로 배포, 스케일링, 관리해주는 컨테이너 오케스트레이션[1] 오픈소스 플랫폼으로, 구글 내부의 컨테이너 서비스를 오픈소스로 공개한 것이다.
- 쿠버네티스라는 명칭은 조타수나 파일럿을 뜻하는 그리스어에서 유래되었으며, k와 s 사이에 8글자를 나타내는 K8s로 약식 표기한다.

[그림 3-12] 쿠버네티스 아키텍처



1 **도커 이미지** • 컨테이너를 실행하기 위한 노드 집합으로 물리 서버 위에 클러스터라는 가상의 컨테이너 실행 영역을 정의, 하나 이상의 마스터 노드(Master Node)와 워커노드(Worker Node)로 구성

2 **노드(Node)** • 물리서버 혹은 가상서버로 마스터노드와 워커노드가 존재

3 **마스터 노드** • 클러스터 제어 및 관리 영역으로 클러스터의 상태 관리 및 모니터링을 수행

4 **워커 노드** • 애플리케이션 실행 영역으로 컨테이너를 포함하는 파드가 배포, 실행되는 영역

5 **파드(Pod)** • 단일한 노드에 배포되는 하나 이상의 컨테이너 집합으로 가장 작은 쿠버네티스 배포 단위

1) 컨테이너 오케스트레이션(Container Orchestration) : 컨테이너의 배포, 관리, 확장, 네트워킹을 자동화하는 기술

03 클라우드 네이티브 구성요소 및 원칙

3.4 데브옵스

3.4.1 데브옵스 개요

- 데브옵스(DevOps)는 개발(Development)과 운영(Operations)의 합성어로, 개발과 운영 프로세스의 통합 및 자동화를 통해 신속한 서비스를 제공하도록 지원하는 통합 프로세스, 도구, 조직문화 등을 포함한 체계를 말한다.
- 데브옵스 구현을 통한 개발팀(Dev)과 운영팀(Ops)의 협업은 개발 계획부터 코딩, 테스트, 배포, 운영 및 지속적인 모니터링에 이르는 모든 단계에 걸쳐 계속된다. 이러한 협업 체계는 추가 개선, 개발, 테스트 및 구축에 대한 지속적인 피드백을 반영하는 기반이 된다.
- 데브옵스를 통해 얻을 수 있는 가장 중요한 효과는 필요한 기능 변경 또는 추가 기능을 더 빠르고, 지속적으로 배포할 수 있다는 것이다.

[그림 3-13] 데브옵스 개념도

※ 데브옵스 도구는 CI/CD 도구와 동일한 개념으로 사용되고 있음

03 클라우드 네이티브 구성요소 및 원칙

3.4 데브옵스

3.4.2 데브옵스 프로세스

- 데브옵스 프로세스는 계획(plan), 코딩(code), 빌드(build), 테스트(test), 릴리즈(release), 배포(deploy), 운영(operate), 모니터링(monitor)의 8단계로 구성된다.

[그림 3-14] 데브옵스 8단계 프로세스

1 계획(plan)	• 현업팀과 개발팀 간 프로젝트를 목표로 협의하여 계획을 수립
2 코딩(code)	• 개발팀은 애플리케이션 설계 및 개발을 수행하고, 깃(Git) 같은 도구를 이용하여 코드를 저장
3 빌드(build)	• 메이븐(Maven), 그레이들(Gradle) 등의 빌드 도구를 이용하여 여러 저장소의 코드를 통합하여 애플리케이션을 빌드
4 테스트(test)	• 셀레늄(Selenium), 제이유닛(Junit) 등의 테스트 자동화 도구를 이용하여 애플리케이션을 테스트
5 릴리즈(release)	• 배포 도구 등을 이용하여 통합된 애플리케이션을 개발서버에서 운영 서버로 배포
6 배포(deploy)	• 젠킨스(Jenkins) 등의 도구를 이용하여 구현한 기능들을 자동으로 애플리케이션으로 통합
7 운영(operate)	• 소프트웨어가 배포되면 운영팀은 환경 설정을 하거나, 필요한 리소스를 통해 프로비저닝(Provisioning) 등의 활동을 수행
8 모니터링(monitor)	• 운영상 사용자에게 충격을 줄 만한 특정 이슈 등의 식별을 위해 모니터링 수행

1) 프로비저닝(Provisioning) : 사용자의 요구사항에 맞게 시스템 자원을 할당, 배치, 배포해 두었다가 필요 시 시스템을 즉시 사용할 수 있는 상태로 미리 준비해 두는 것을 말함

03 클라우드 네이티브 구성요소 및 원칙

3.4 데브옵스

3.4.3 데브옵스 문화

- 일반적으로 개발팀에서 개발이 끝나면 테스트를 통해 시스템은 운영팀에 이관되고, 운영팀이 시스템을 배포, 관리 및 운영한다. 일단 운영팀에 이관된 시스템은 개발팀이 관여하지 않고, 운영팀은 시스템 운영을 담당한다. 시스템 운영 과정에 장애가 발생하면 개발팀과 운영팀은 서로 책임 공방을 하게 된다.
- 데브옵스는 개발과 운영의 분리에서 오는 이러한 문제점을 해결하기 위해 개발과 운영을 하나의 조직처럼 움직일 수 있도록 협업과 소통을 통해 투명한 개발 및 운영 문화를 만든다.
- 데브옵스 문화는 보이지 않는 것이며 기관의 규모, 조직 구성원, 서비스 등에 따라 달라질 수 있다. 그리고 데브옵스 문화는 한번에 만들어질 수 없고 여러 번 시행착오의 과정을 통해 만들어진다.
- 데브옵스 적용을 위해 영국 정부가 제공하는 "데브옵스를 위한 좋은 습관(Good habit for DevOps)"의 내용을 살펴본다.

[그림 3-15] 영국 정부의 "데브옵스를 위한 좋은 습관"의 주요 내용

교차 기능 조직 (Cross Functional Team)	• 하나의 팀에 각각 다른 역할을 할 수 있는 팀원들로 구성하여 End-to-End 서비스 조직 구성 • 한 팀 내에 서비스의 기획에서부터 개발, 운영 등 해당 서비스와 관련된 모든 것을 할 수 있는 구조로 팀을 구성
광범위한 공통 지표 (Widely Shared Metrics)	• 팀 전체가 기준으로 삼을 수 있는 서비스에 대한 공통적인 지표 필요 • 서비스를 개발하고 개선했을 때, 이를 평가하고 현재 진행 상태를 인지할 수 있는 기준 필요 예) 응답 시간, 일 방문자수, 평균 체류시간, CPU 사용률 등
반복적인 작업의 자동화 (Automating Repetitive Tasks)	• CI/CD 도구 도입을 통한 빌드, 배포, 테스트 자동화 • 반복적인 작업의 자동화를 통해 작업의 효율화, 배포 및 테스트 시간 단축, 신속한 서비스 출시 등 가능
사후 검증 (Post mortems)	• 장애나 이슈가 있을 때, 처리 후에, 그 내용을 전체 팀과 공유 • 서비스를 운영하는 팀의 문제점은 이슈 등에 대한 심각도가 얼마나 높은지를 인지하지 못하는 경우가 있으므로 사후 검증을 통해 이슈의 영향도를 이해
정기적 릴리즈 (Regular Release)	• 시스템 릴리즈는 개발, 테스트, 배포 과정을 거쳐야 하므로 많은 협업이 필요한 작업임 • 정기적으로 릴리즈 주기를 설정하면 어떻게 협업해야 할지 파악할 수 있음

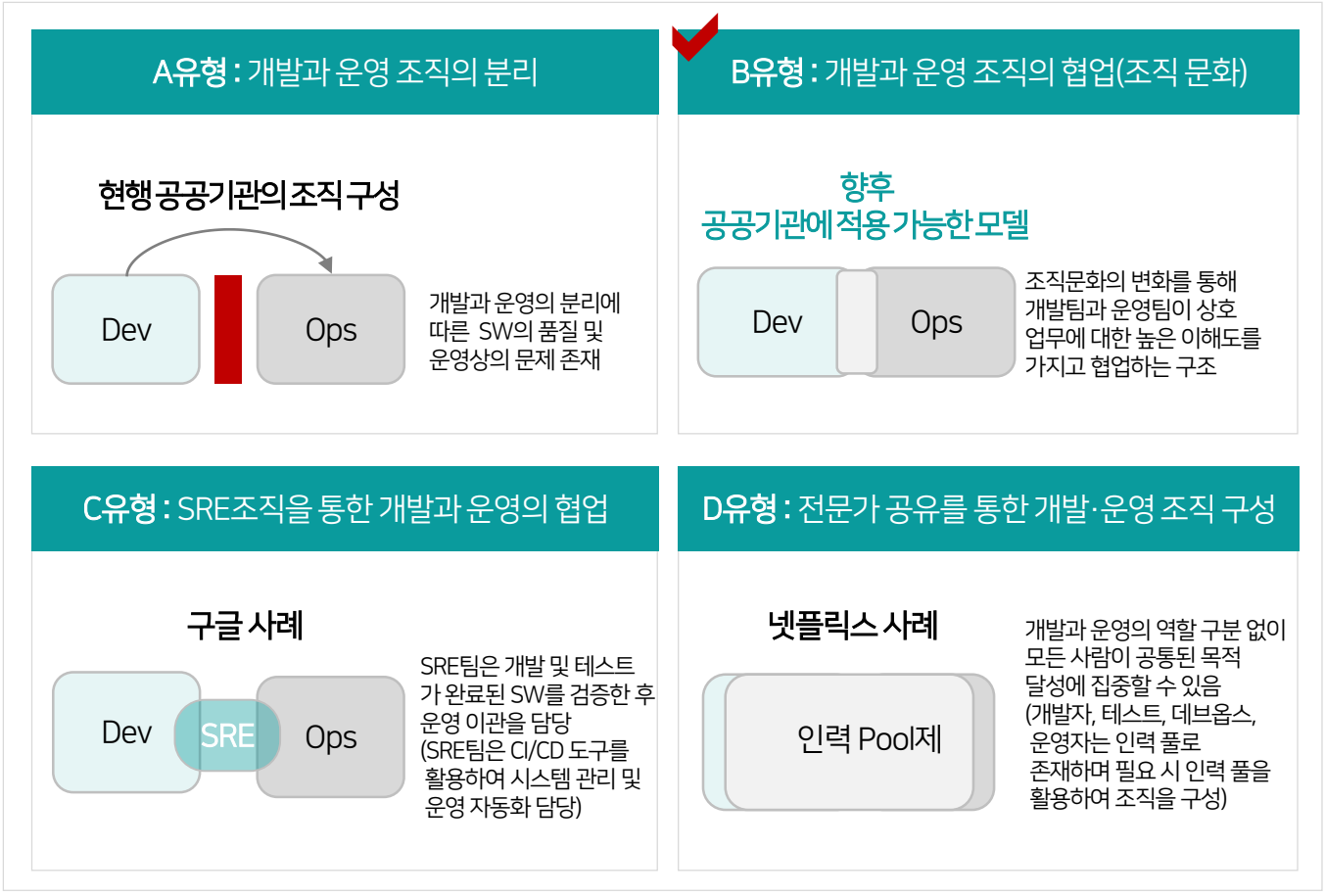
[출처 : 영국 정부(GOV.UK), Good habit for DevOps]

3.4 데브옵스

3.4.4 데브옵스 조직

- 데브옵스를 지원하기 위한 조직은 크게 4개 유형이 있으며, 각 조직의 일하는 방식과 조직 구성 원칙에 따라 적합한 조직 유형을 참고하여 조직을 구성하도록 한다.
- 데브옵스 조직 A 유형은 개발과 운영 조직을 분리하는 방식으로, 현재 국내 대부분의 공공기관은 이러한 형태로 정보화 조직을 구성하고 있다. 전반적으로 소프트웨어 품질과 운영상의 문제가 존재한다.
- 데브옵스 조직 B 유형은 개발과 운영 조직의 협업이 가능한 조직 문화를 만드는 모델이며, 개발팀과 운영팀 상호 간에 높은 이해에 기반하여 협업이 가능하다.
- 데브옵스 조직 C 유형은 구글 사례에 적용된 SRE(Site Reliability Engineering, 사이트 신뢰성 엔지니어링) 팀을 통한 개발과 운영의 협업체계를 유지한다.
- 데브옵스 조직 D 유형은 넷플릭스에서 적용하고 있는 모델로 전문가 인력의 공유를 통해 개발·운영 인력을 1개의 팀으로 구성한다.

[그림 3-16] 데브옵스 조직 유형



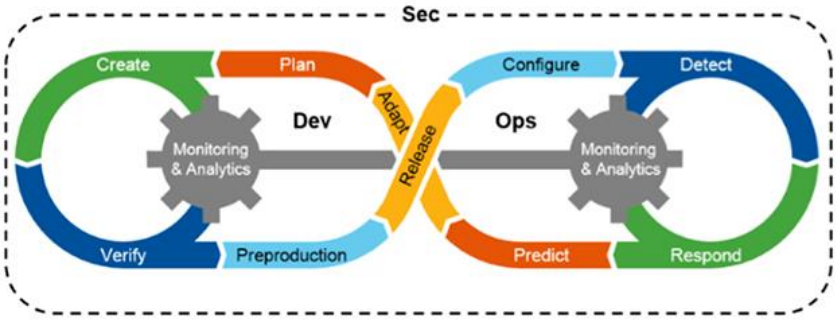
03 클라우드 네이티브 구성요소 및 원칙

3.4 데브옵스

3.4.5 데브섹옵스(DevSecOps) 개념

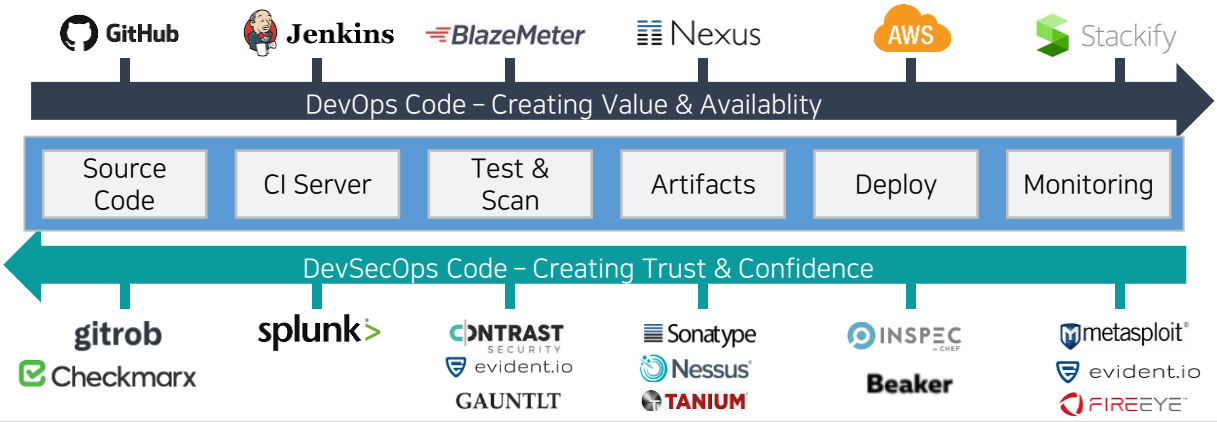
- 데브섹옵스(DevSecOps)는 데브옵스 프로그램 내에서 애플리케이션 개발 및 배치 파이프라인에 보안(Security) 관점을 추가한 개념이다.
- 협업을 중시하는 데브옵스 프레임워크에 보안을 공동의 책임으로 간주하고 개발 처음부터 운영까지의 전체 라이프사이클에 걸쳐 내재화된 보안 기반을 구축하는 것이다.
- 보안이 데브옵스 파이프라인 상에 유기적으로 결합되어 있지 않고 개발 최종 단계에 적용된다면 점검 과정에 심각한 보안 결함의 발견 시 상당한 전체 소스코드의 재작성이 필요할 수도 있다.
- 데브섹옵스는 정보보안팀, 개발팀, 운영팀 간의 지속적인 협업을 통해 모니터링, 평가 및 분석을 진행하고 보안상의 결함과 취약점을 조기에 노출시킴으로써 프로그램 수정에 필요한 시간을 단축시키고, 재작업 비용을 줄이고, 보안사고의 위험도 감소시킨다.

[그림 3-17] 데브섹옵스 라이프사이클



[출처: 가트너, DevSecOps Life Cycle]

[그림 3-18] DevSecOps 적용 툴체인



[출처: Stackify, DevSecOps Toolset]

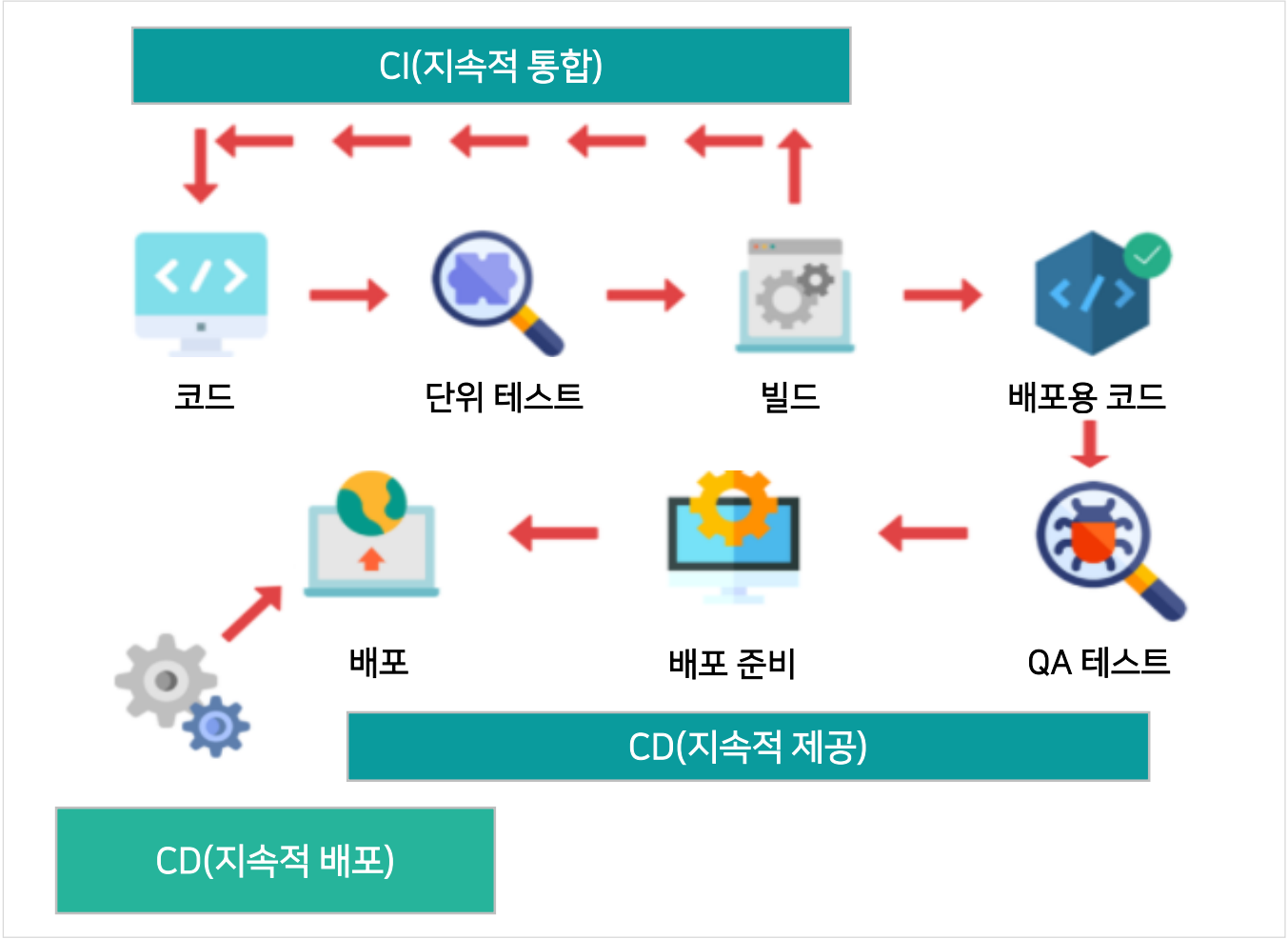
03 클라우드 네이티브 구성요소 및 원칙

3.5 CI/CD

3.5.1 CI/CD 개념

- CI는 지속적 통합(Continuous Integration)을 의미하며, 애플리케이션의 코드 변경사항이 빌드 및 테스트되어 공유 리포지토리에 통합되므로 여러 명의 개발자에 의한 통합 개발이 가능해진다.
- CD는 지속적 제공(Continuous Delivery)과 지속적 배포(Continuous Deployment)를 의미하며 이 두 용어는 상호 교환적으로 사용된다.
- 지속적 제공은 변경 사항이 테스트를 거쳐 리파지토리에 자동으로 업로드되는 것을 말한다.
- 지속적 배포는 개발자의 변경사항을 리파지토리에서 고객이 사용 가능한 운영 환경까지 자동으로 릴리즈하는 것을 의미한다. CD를 통해 운영팀은 이 리포지토리에서 애플리케이션을 실시간 운영환경으로 배포할 수 있게 된다,

[그림 3-19] CI/CD 개념도



[출처:https://blog.oursky.com/2019/08/19/how-to-build-cicd-pipeline/]]

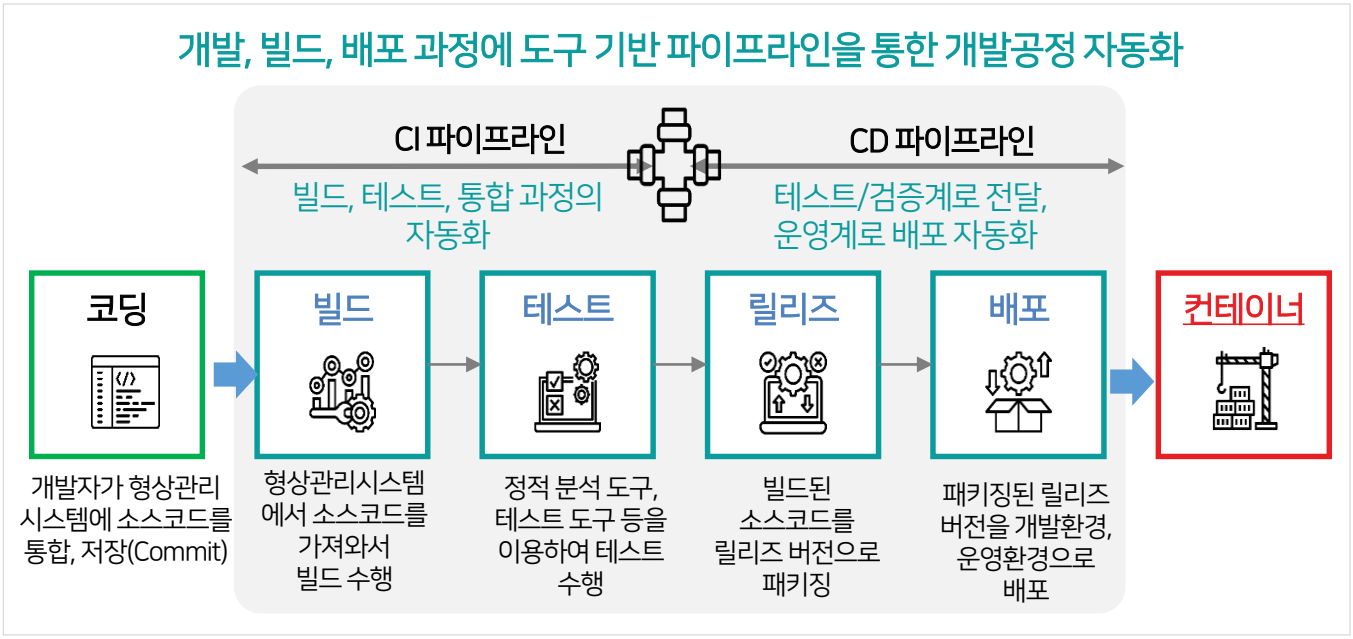
03 클라우드 네이티브 구성요소 및 원칙

3.5 CI/CD

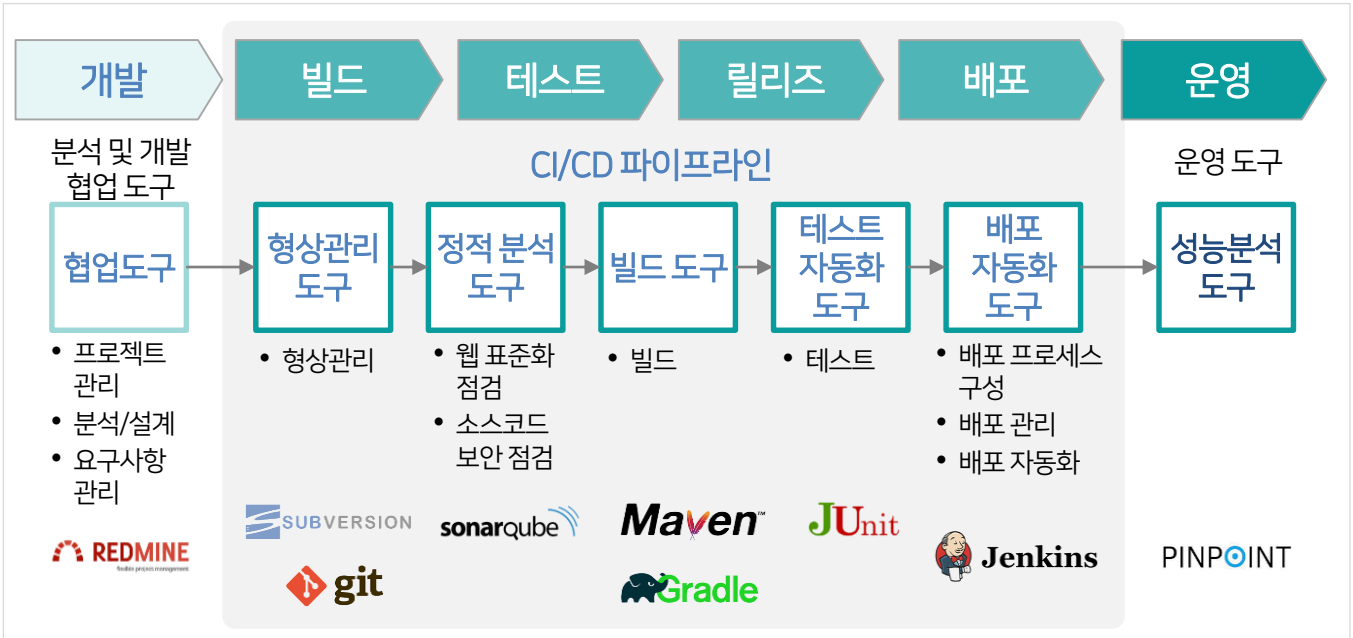
3.5.2 CI/CD 파이프라인

- CI/CD 파이프라인은 애플리케이션 코드가 개발 환경에서 테스트 환경을 거쳐 최종적으로 운영 환경에 배포되고, 서비스의 형태로 사용자에게 전달되는 일련의 과정을 말하며, 이 과정에 대한 자동화를 통해 개발 및 배포 시간을 단축하고, 품질을 개선할 수 있게 된다

[그림 3-20] CI/CD 파이프라인



[그림 3-21] 전자정부 클라우드 플랫폼의 CI/CD 파이프라인 예시



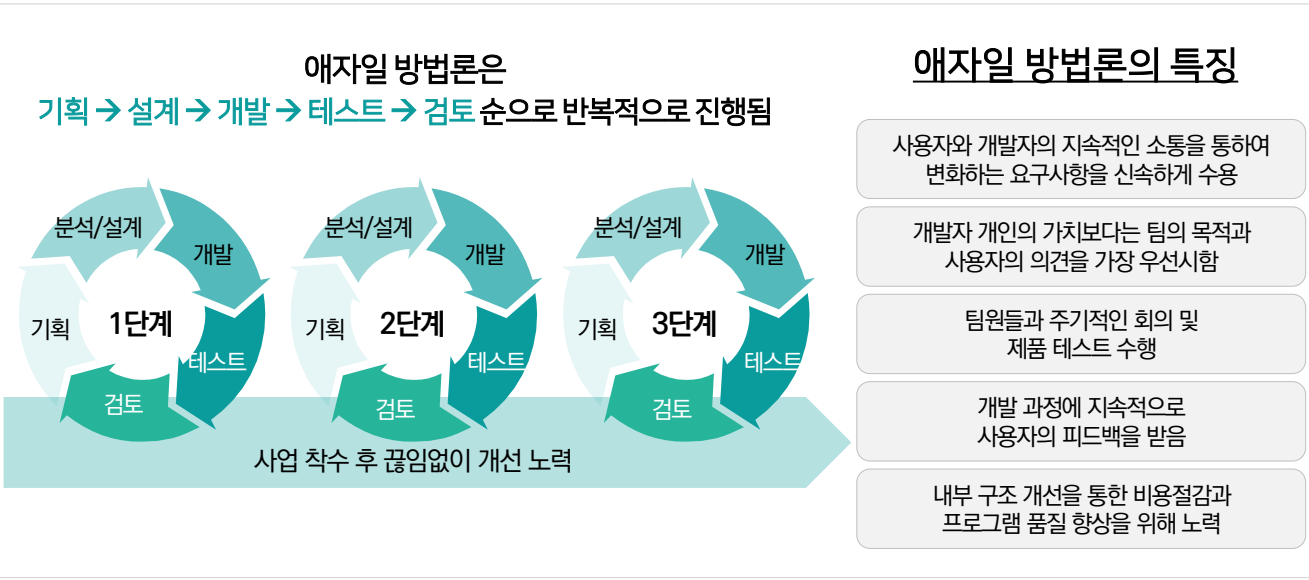
03 클라우드 네이티브 구성요소 및 원칙

3.6 애자일 방법론

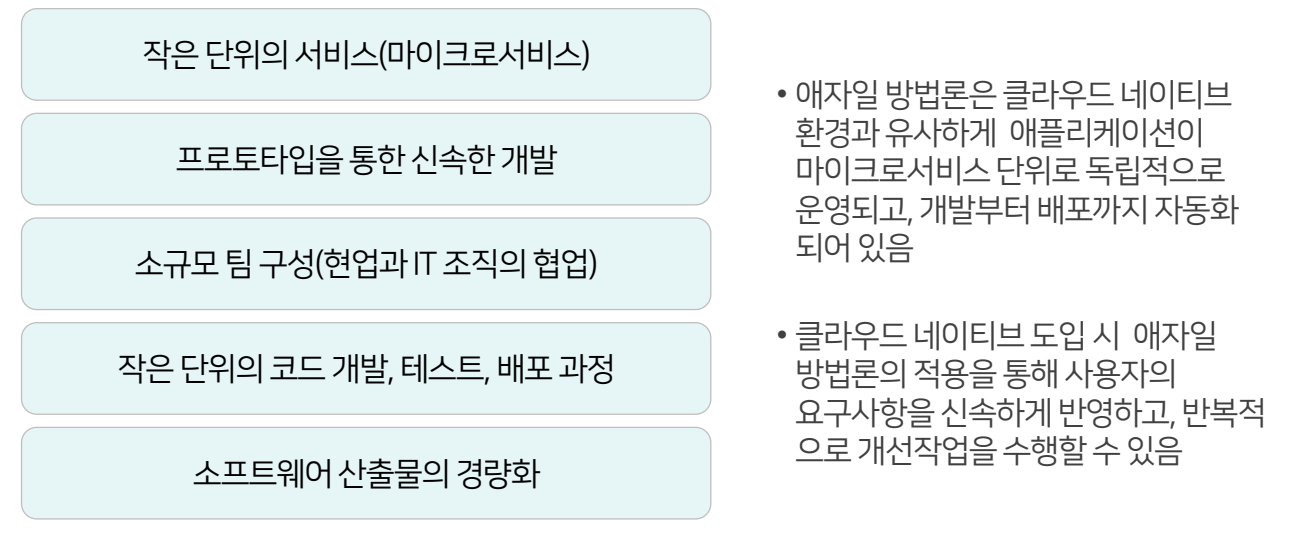
3.6.1 애자일 방법론 개요

- 애자일(Agile) 방법론은 사용자의 요구사항을 신속하게 반영하기 위해 기획, 분석·설계 과정에 많은 시간과 노력을 기울이지 않고 빠르게 프로토타입을 개발하여 사용자의 피드백과 방향성을 확인하고 지속적인 개선 작업을 짧은 주기로 반복하는 개발 방법론이다.
- 애자일 방법론과 클라우드 네이티브의 공통점은 작은 서비스 단위로 사용자의 요구사항을 지속적이고, 반복적으로 반영하여 개선 작업을 수행하는 것이다.

[그림 3-22] 애자일 방법론의 진행 과정 및 특징



[그림 3-23] 클라우드 네이티브와 애자일 방법론의 공통점



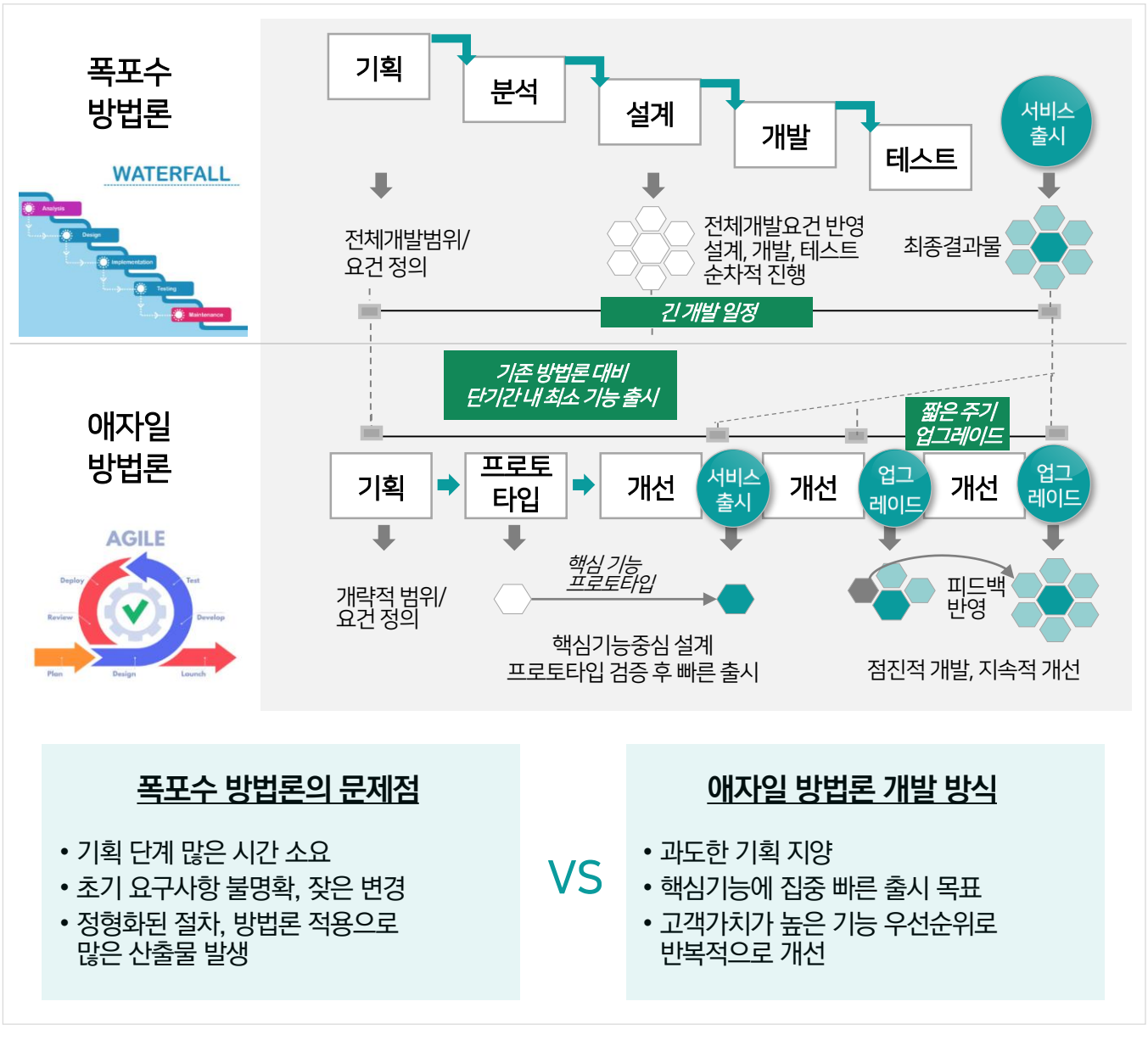
03 클라우드 네이티브 구성요소 및 원칙

3.6 애자일 방법론

3.6.2 애자일 방법론과 폭포수 방법론의 차이점

- 애자일은 1970년대 윌리엄 로이스의 논문에 처음 등장했으며, 스프린트(Sprint)라는 짧고 점진적인 개발 주기로 구성된 프로젝트 관리 방법론이다.
- 기존의 폭포수 방법론의 개발 공정은 기획, 분석, 설계, 개발, 테스트 단계가 위에서 아래로 순차적으로 진행되지만, 애자일 방법론은 각 개발 공정을 명확하게 구분하지 않고 각 단계를 반복적으로 수행하면서 요구사항을 추가하거나 수정하면서 개발을 수행하는 방법론이다.

[그림 3-24] 폭포수 방법론과 애자일 방법론 비교



폭포수 방법론의 문제점

- 기획 단계 많은 시간 소요
- 초기 요구사항 불명확, 잦은 변경
- 정형화된 절차, 방법론 적용으로 많은 산출물 발생

VS

애자일 방법론 개발 방식

- 과도한 기획 지양
- 핵심기능에 집중 빠른 출시 목표
- 고객가치가 높은 기능 우선순위로 반복적으로 개선

03 클라우드 네이티브 구성요소 및 원칙

3.6 애자일 방법론

3.6.2 애자일 방법론과 폭포수 방법론의 차이점

- 애자일 방법론과 폭포수 방법론의 개발 형태, 계획 수립, 적용 가능 사업, 사용자 참여, 리더십, 업무 책임 관점의 차이점은 다음과 같다.

[표 3-3] 애자일 방법론과 폭포수 방법론 비교

구분	애자일 방법론	폭포수 방법론
개발 형태	• 점진적 반복적 개발	• 폭포수 개발
계획 수립	• 사업 초기 : 개략적인 전체 계획 • 사업 진행 : 주기적인 상세 계획 ※ 계획보다 변화 대응 위주	• 사업 초기 : 상세하고 정확한 전체 계획 • 사업 진행 : 전체 계획에 대한 부분적인 변경관리
적용 가능 사업	• 요구사항이 명확하지 않고, 시간과 비용의 예측이 어려운 사업 • 개발 과정을 통해 사업의 목표와 방향을 정함	• 요구사항이 명확하며, 개발 기간 동안 요구사항의 변화가 크지 않은 사업 • 사업 목표와 방향이 명확한 경우 적용 ※ 공공기관은 계획과 예산에 의해 사업이 수행되고 있음
사용자 참여	• 개발기간 동안 사용자의 지속적 참여 ※ 공공기관의 경우, 인력의 부족으로 개발기간 동안 지속적 참여가 가능한지 사전 파악 필요	• 공정단계별 산출물 검토 및 사용자 테스트 단계에 참여
리더십	• 개발자 중심 - 기획자, 설계자, 개발자 등이 수평적인 관계 ※ 개발자의 의지대로 기획자, 설계자가 끌려갈 수 있음	• 기획자, 설계자 중심 - 계획, 설계에 따라 개발이 진행되는지 추적 관리함
업무책임	• 팀 책임	• 개인 책임

- 공공기관의 경우 명확한 사업 목표와 방향, 정해진 예산에 의해 정보화사업을 수행하므로 애자일 방법론이 클라우드 네이티브 사업에 적합하지만, 전면 도입보다는 필요 시 부분적 적용을 고려하도록 한다.

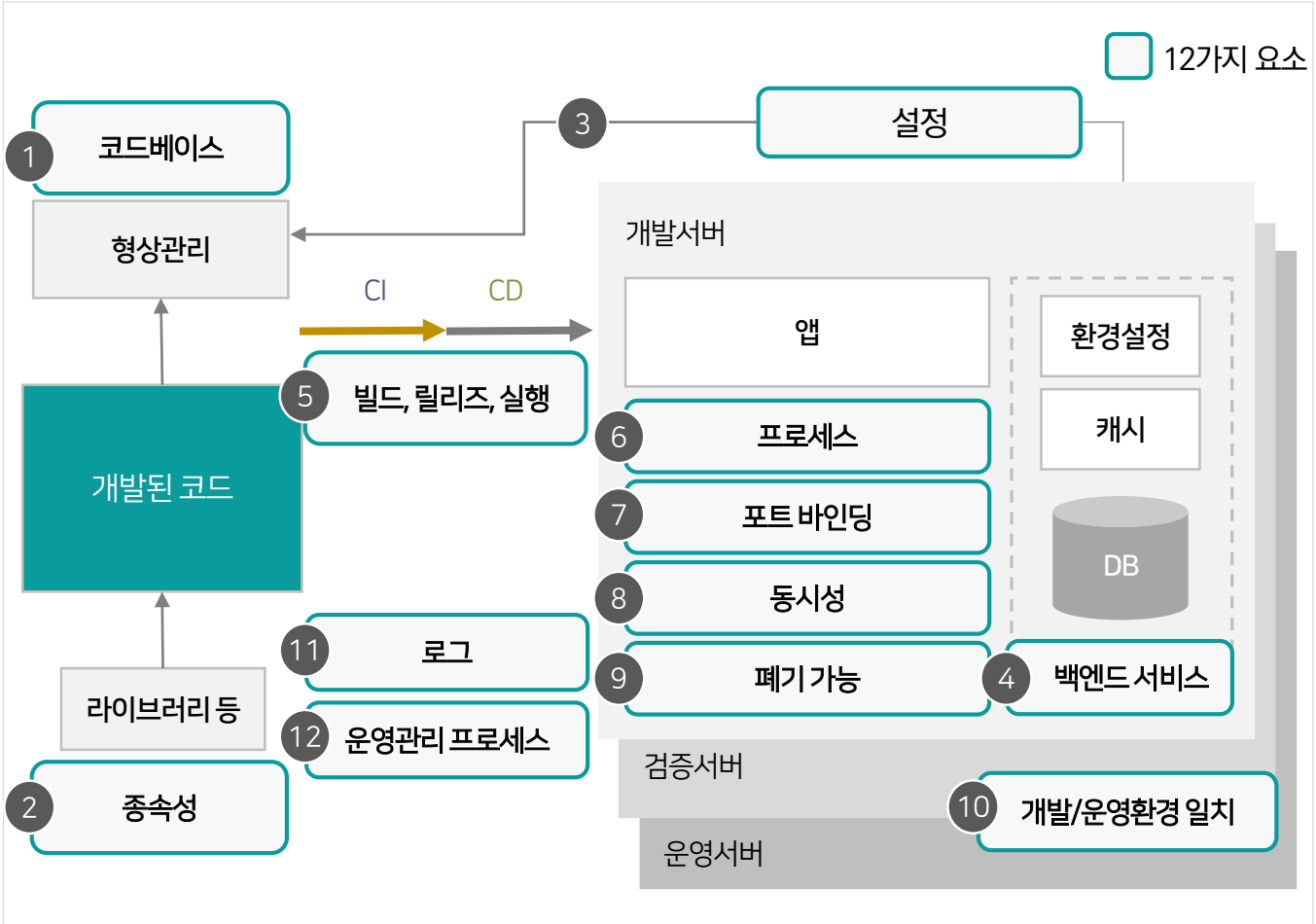
03 클라우드 네이티브 구성요소 및 원칙

3.7 12가지 요소

3.7.1 12가지 요소(12 Factors)개념

- 12가지 요소 원칙은 2012년 헤로쿠(Heroku)에서 일하던 개발자들이 클라우드에 적합한 SaaS 애플리케이션 개발과 배포 방법에 맞는 12가지 원칙을 개념화한 것이다. 12가지 요소는 클라우드 네이티브 환경에 적합하게 적용되어야 할 부분들을 명확하게 정의하고 있다.
- 클라우드 환경에서 효과적인 서비스 운영을 위해 클라우드 네이티브 애플리케이션 형태의 구성은 필수적이다. 기존 온프레미스 환경에 적합하게 설계되어 동작하는 애플리케이션만으로는 클라우드라는 환경을 효율적으로 사용하기 어렵기 때문이다.
- 12가지 요소 원칙은 클라우드 환경에 적합하게 적용되어야 할 부분과 지켜져야 할 부분들을 원칙으로 정의하며, 클라우드 네이티브 애플리케이션 개발 시 이 원칙을 준수하도록 해야 한다.

[그림 3-25] 12가지 요소 개념도



[출처: Microservices Arch 12 Factor App, https://www.youtube.com/watch?v=N45VJ20wvCw]

03 클라우드 네이티브 구성요소 및 원칙

3.7 12가지 요소

3.7.2 12가지 요소 원칙 설명

- 12가지 요소 원칙에 대한 설명은 다음과 같다.

[표 3-4] 12가지 요소 원칙 설명

구분	설명
1 코드베이스 (Codebase)	• 하나의 코드베이스(소스코드)로 버전 관리하며, 이를 여러 곳에 배포 ※ 코드베이스는 애플리케이션의 소스코드 전체 집합, 그것을 담고 있는 저장소를 말하며 SVN(Subversion)과 같은 중앙관리형 버전관리 시스템에서 하나의 저장소, 깃과 같은 분산형 버전 관리 시스템에서 다수의 저장소일 수 있음
2 종속성 (Dependencies)	• 패키지, 라이브러리 등 종속이 필요한 경우 명시적으로 선언하고 분리시켜 실행환경의 종속성 제거
3 설정 (Configuration)	• 소스 코드와 설정 정보를 분리, 실행 시 코드에서 읽어서 사용
4 백엔드서비스 (Backing Services)	• 애플리케이션 작동에 필요한 서비스(DB, 메시지큐, 캐시 등)를 연결된 리소스로 취급하여 연결과 분리가 용이
5 빌드, 릴리즈, 실행 (Build, Release, Run)	• 빌드, 릴리즈, 실행단계를 엄격히 분리
6 프로세스 (Processes)	• 애플리케이션 실행 시 하나 혹은 여러 개의 무상태(stateless) 프로세스로 실행
7 포트바인딩 (Port-binding)	• 애플리케이션은 독립적이며, http 같은 포트 바인딩을 통해서 외부에 서비스 제공
8 동시성 (Concurrency)	• 애플리케이션을 수평적으로 확장하며, 무상태 특성이 이런 확장을 단순하게 만듦
9 폐기가능 (Disposability)	• 빠른 시작과 안정적 종료(Graceful shutdown)을 통한 안정성 극대화
10 개발/운영 환경 일치 (Dev/Prod Parity)	• 개발/검증/운영 환경을 가능한 비슷하게 유지함으로써 지속적인 개발 및 배포 가능
11 로그 (Log)	• 로그 파일을 이벤트 스트림으로 취급하여 이를 취합, 인덱싱, 분석할 수 있어야 함
12 운영관리 프로세스 (Admin Process)	• 시스템관리 작업은 일회성 프로세스로 만들어서 실행

[출처: https://12factor.net/ko/]

03 클라우드 네이티브 구성요소 및 원칙

3.7 12가지 요소

3.7.3 12가지 요소 원칙의 특징

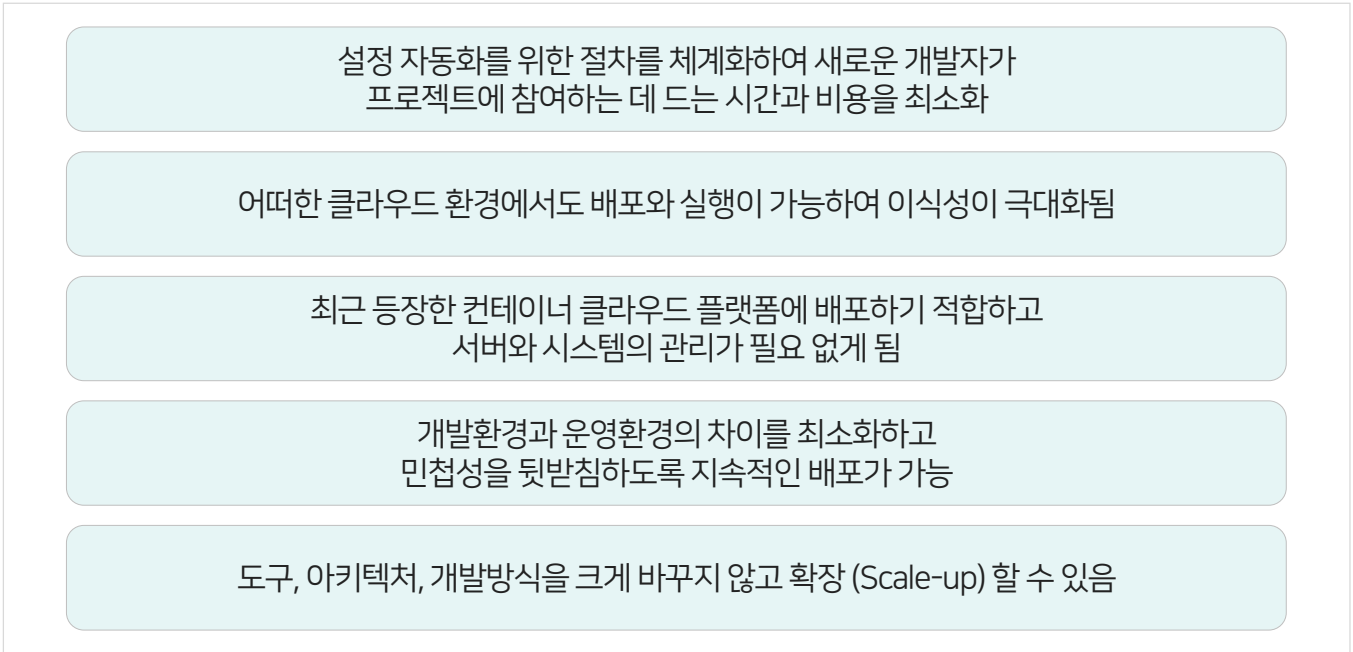
- 12가지 요소 원칙은 클라우드 네이티브는 무상태, 느슨한 결합, 독립적/자율적 운영, 자동화 측면에서 공통점이 있다.

[표 3-5] 12가지 요소 원칙과 클라우드 네이티브의 공통점

구분	공통점
무상태	• 컨테이너 단위
느슨한 결합	• http, Restful API
독립적/자율적	• 바인드, 분산 DB 지향
자동화	• 데브옵스, CI/CD

- 12가지 요소 원칙은 SaaS 애플리케이션을 개발하기 위한 방법론이므로 어떠한 언어로 작성된 앱에도 적용 가능하고, 백엔드 서비스와 다양하게 조합할 수 있다.

[그림 3-26] 12가지 요소 앱의 특징



[출처: https://12factor.net/ko/]

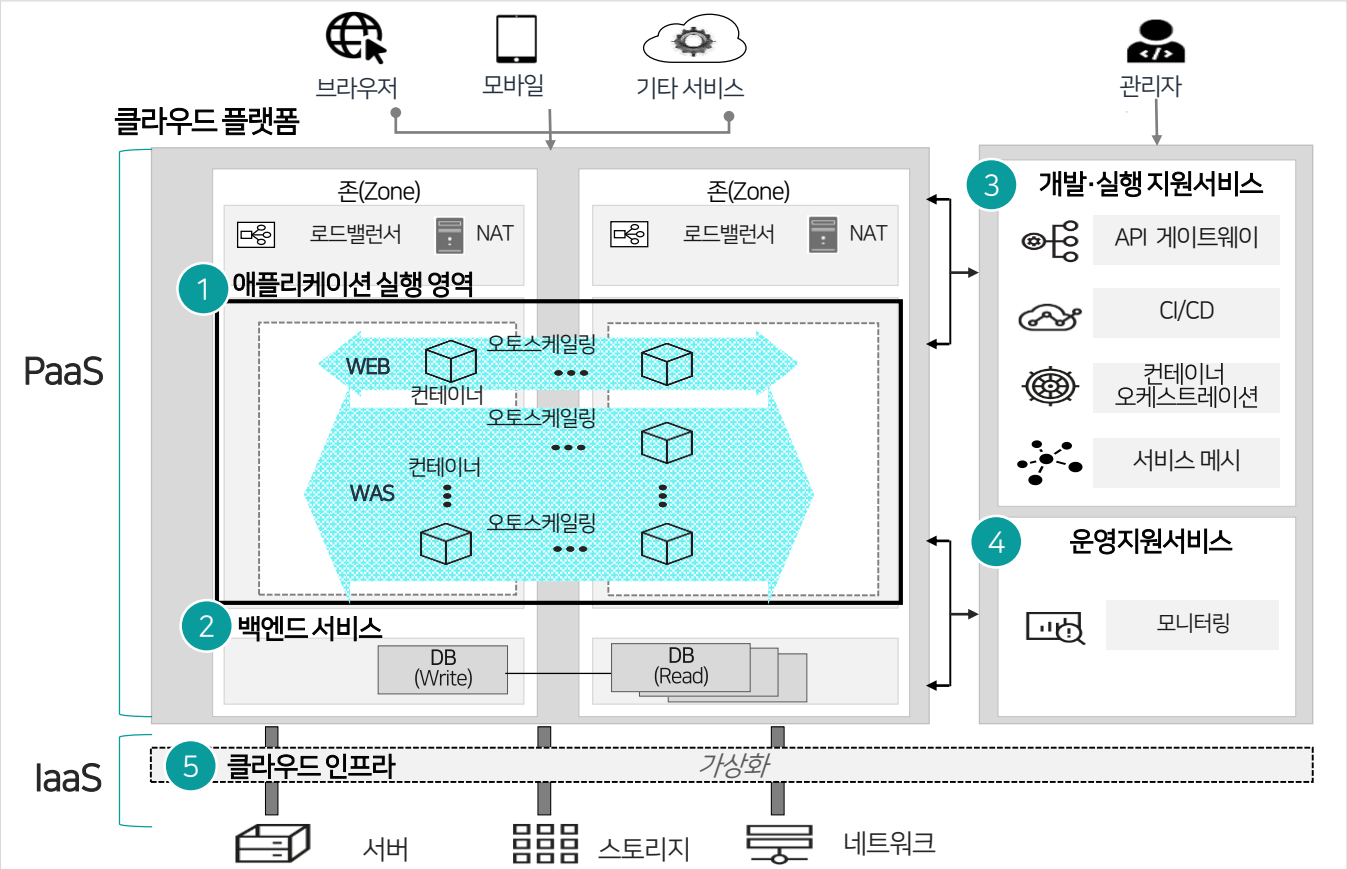
03 클라우드 네이티브 구성요소 및 원칙

3.8 클라우드 네이티브 애플리케이션 아키텍처

3.8.1 클라우드 네이티브 애플리케이션 아키텍처 개요

- 클라우드 네이티브 애플리케이션 아키텍처는 애플리케이션 실행 영역, 백엔드 서비스, 개발·실행 지원 서비스, 운영지원 서비스, 클라우드 인프라 영역으로 구성된다.

[그림 3-27] 클라우드 네이티브 애플리케이션 아키텍처



영역	설명
1 애플리케이션 실행 영역	• 기능 단위로 분해된 애플리케이션이 컨테이너 형태로 실행되는 영역 • 트래픽의 변동에 따라 컴퓨팅 자원(컨테이너)의 동적 확장 적용 • 클라우드 네이티브 애플리케이션의 전환, 구축 시 직접 설계, 개발 되는 영역 • 이외 영역은 클라우드 플랫폼에서 제공하는 서비스 활용
2 백엔드 서비스	• 애플리케이션이 실행되기 위해 필요한 DB, 캐시, 메시지큐 등 클라우드 서비스
3 개발·실행 지원 서비스	• 클라우드 네이티브 애플리케이션의 개발 및 실행 환경을 제공하는 클라우드 서비스
4 운영지원 서비스	• 로깅, 성능관리 및 모니터링 등 애플리케이션 운영을 위한 클라우드 서비스
5 클라우드 서비스	• 클라우드에서 제공되는 인프라 서비스, IaaS

03 클라우드 네이티브 구성요소 및 원칙

3.8 클라우드 네이티브 애플리케이션 아키텍처

3.8.2 애플리케이션 실행 영역

- 애플리케이션 실행 영역은 클라우드 네이티브 애플리케이션이 컨테이너 단위로 배포, 실행되는 영역이다.
- 애플리케이션 실행 영역은 애플리케이션, 실행 단위인 컨테이너, 컨테이너 구동을 위한 컨테이너 런타임으로 구성된다.

[그림 3-28] 애플리케이션 실행 영역

영역	설명
1 클라우드 네이티브 애플리케이션	• 미들웨어 (웹 서버, WAS)를 포함하여 배포하여 개별적 실행 가능 • 배포 대상에 따라 아래의 2가지 컨테이너로 구성 - 웹 서버 : 이미지 등 정적 콘텐츠 서비스 - WAS : 마이크로서비스 단위 애플리케이션 서비스
2 컨테이너	• 클라우드 네이티브 애플리케이션 배포 및 실행 단위
3 컨테이너 런타임	• 클라우드 실행환경 제공 예) 도커, 쿠버네티스 등

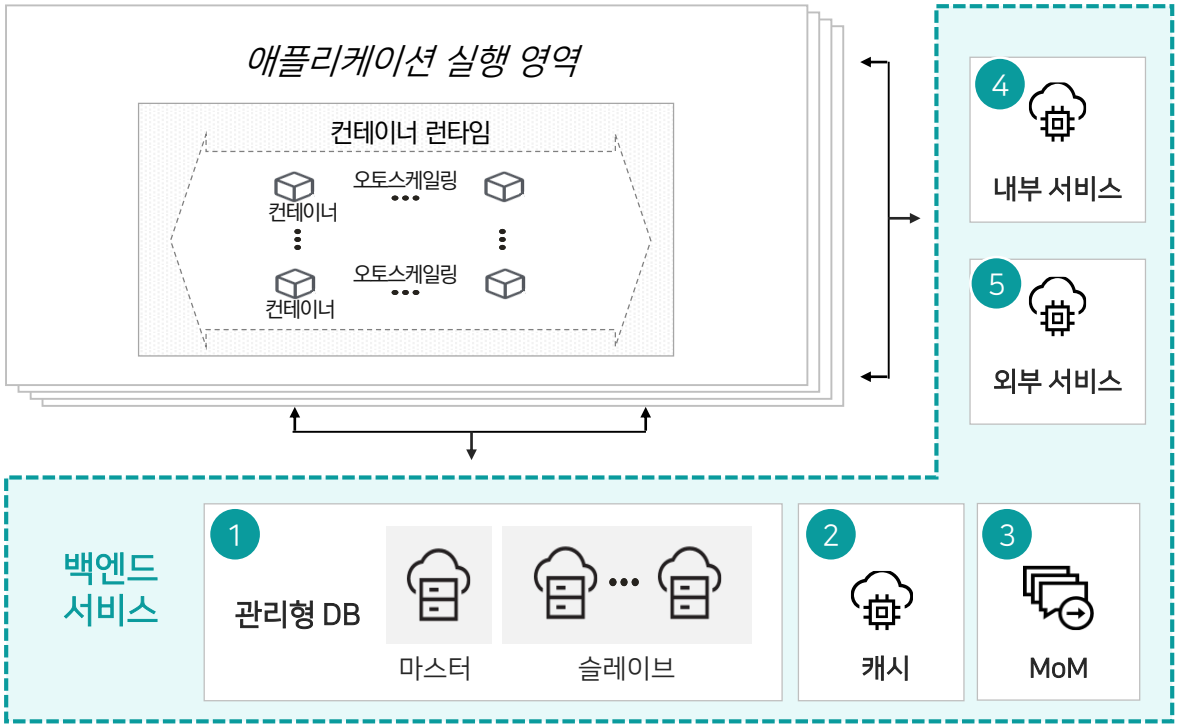
03 클라우드 네이티브 구성요소 및 원칙

3.8 클라우드 네이티브 애플리케이션 아키텍처

3.8.3 백엔드 서비스

- 클라우드 네이티브 애플리케이션을 실행하기 위해 네트워크로 연결된 모든 리소스를 백엔드 서비스(Backing Service)라고 하며, 클라우드 플랫폼에서 제공하는 서비스, 외부 연계 서비스 및 직접 구축한 서비스를 모두 포함한다.
- 애플리케이션 실행영역에 대한 수정을 하지 않고 다른 백엔드 서비스 전환이 가능하도록 구성되어야 한다.

[그림 3-29] 백엔드 서비스 종류



구분		설명
1	DB	• 클라우드 플랫폼에서 제공 및 관리되는 DB 서비스
2	캐시	• 세션관리, 성능향상 등의 용도를 위한 메모리 캐싱 서비스
3	MoM	• MoM(Message Oriented Middleware, 메시지 지향 미들웨어) • 비동기 메시징(Messaging) 서비스
4	내부 서비스	• AI, 빅데이터, 챗봇 등의 클라우드 플랫폼 제공 서비스
5	외부 서비스	• 인증, 간편결제, OpenAPI 형태 외부 연계 서비스

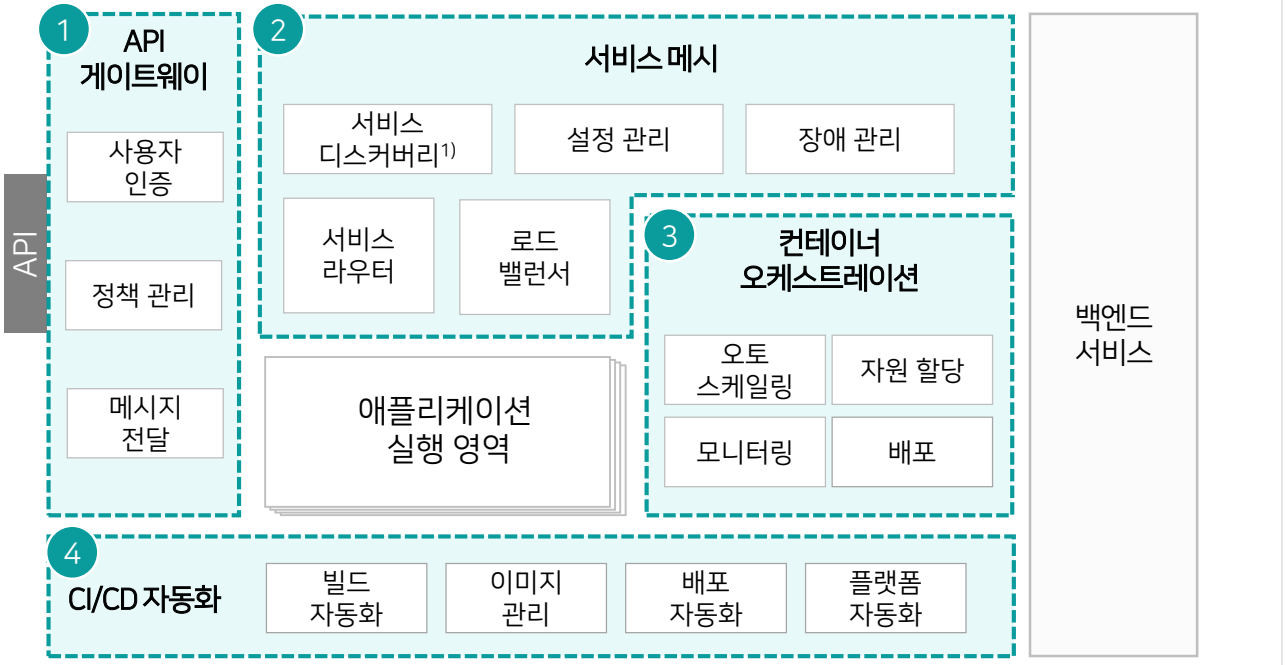
03 클라우드 네이티브 구성요소 및 원칙

3.8 클라우드 네이티브 애플리케이션 아키텍처

3.8.4 개발·실행 지원 서비스

- 클라우드 네이티브 애플리케이션을 실 환경에서 효율적으로 배포하고 안정적으로 운영하기 위해서는 클라우드 네이티브 애플리케이션에 최적화된 개발·실행환경을 구성해야 한다.
- 개발·실행환경은 클라우드 플랫폼이 제공하는 서비스를 이용하거나 오픈소스 또는 상용 SW를 활용하여 자체 구축이 가능하다.

[그림 3-30] 개발·실행 지원 서비스



구분	서비스	설명	주요 솔루션
실행 환경	1 API 게이트웨이	• API 형태의 서비스를 제공하는 마이크로서비스 앞단에서 엔드포인트를 단일화하여 외부 사용자에 제공 • API에 대한 인증과 인가 및 여러 서버로 라우팅하는 기능 등을 담당함	스프링 클라우드 게이트웨이, 줄(Zuul), 콩(Kong) 등
	2 서비스 메시	• 여러 서버에 걸쳐 다수의 컨테이너에 대한 배포, 컨테이너 단위의 오토스케일링, 자원 할당, 장애 복구 및 모니터링 등의 운영 자동화를 제공하는 서비스	쿠버네티스, 클라우드 파운드리, 도커스웜(DockerSwarm)
	3 컨테이너 오케스트레이션	• 대규모의 마이크로서비스를 실시간으로 제어, 관리 • 서비스 간에 통신을 하기 위해 서비스 디스커버리, 로드밸런싱, 장애 제어, 설정관리 등 제공	스프링 클라우드, 이스티오(Istio) 등
내부 서비스	4 CI/CD	• 개발, 테스트, 배포 프로세스에 대한 도구 기반 자동화 및 모니터링 제공	젠킨스, 밤부(Bamboo), 깃, SVN 등

1) 서비스 디스커버리(Service Discovery) : MSA로 구성되어 있는 서비스들은 각자 다른 IP와 포트를 가지고 있으며, 다른 서비스의 IP와 포트를 탐색하는 것을 말함

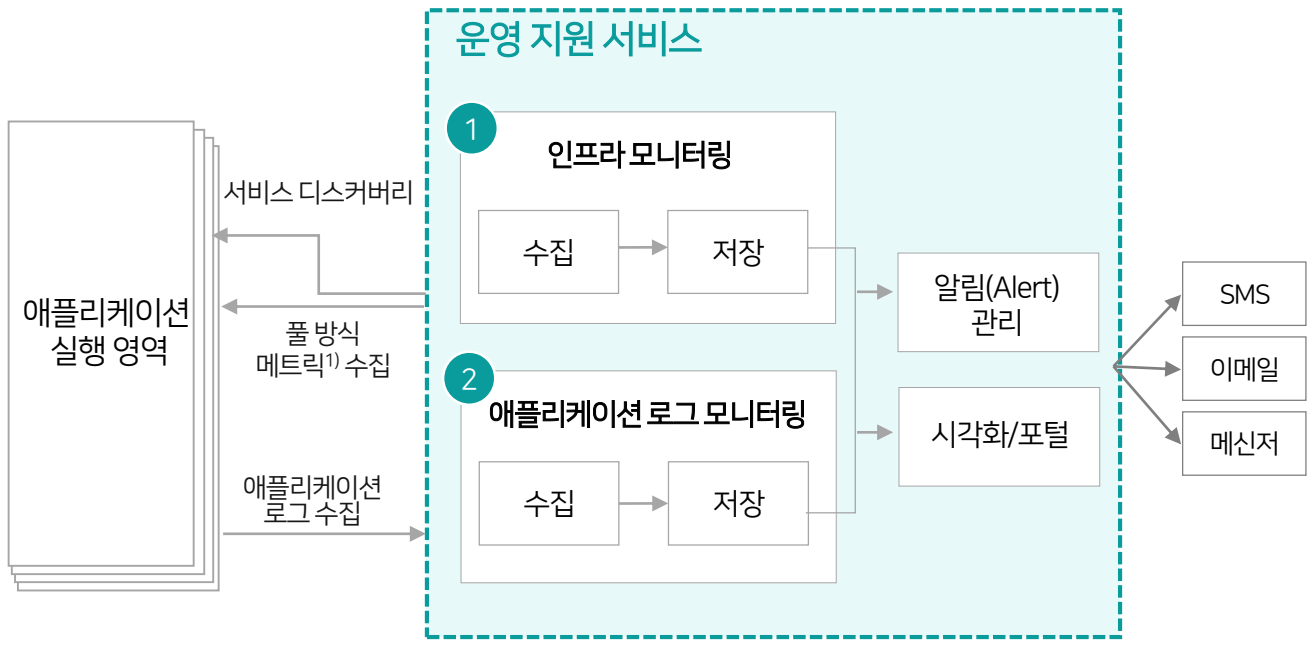
03 클라우드 네이티브 구성요소 및 원칙

3.8 클라우드 네이티브 애플리케이션 아키텍처

3.8.5 운영 지원 서비스

- 운영 지원 서비스는 클라우드 네이티브 애플리케이션 및 인프라 리소스 (디스크, CPU, 메모리 등)에 대한 로그 및 사용 데이터를 수집, 분석, 시각화 등 애플리케이션 전반의 모니터링 기능을 제공한다.
- 클라우드 네이티브 애플리케이션은 마이크로서비스 단위로 OS 수준의 가상화된 환경에서 컨테이너가 동적으로 생성되므로 기존 모니터링과 달리 서비스 디스커버리 및 풀(Pull) 방식의 데이터 수집이 적용된다.

[그림 3-31] 운영 지원 서비스



구분	설명
1 인프라 모니터링	• 컨테이너 클러스터, 노드, 컨테이너 등 상태, 리소스 가용량 및 사용량 등을 수집, 제공
2 애플리케이션 로그 모니터링	• 애플리케이션 헬스체크, 에러, 재시작 횟수 등의 이벤트 정보

1) 메트릭(Metric) : 효율적인 라우팅 결정을 위해 라우터가 사용하는 특정 데이터 집합

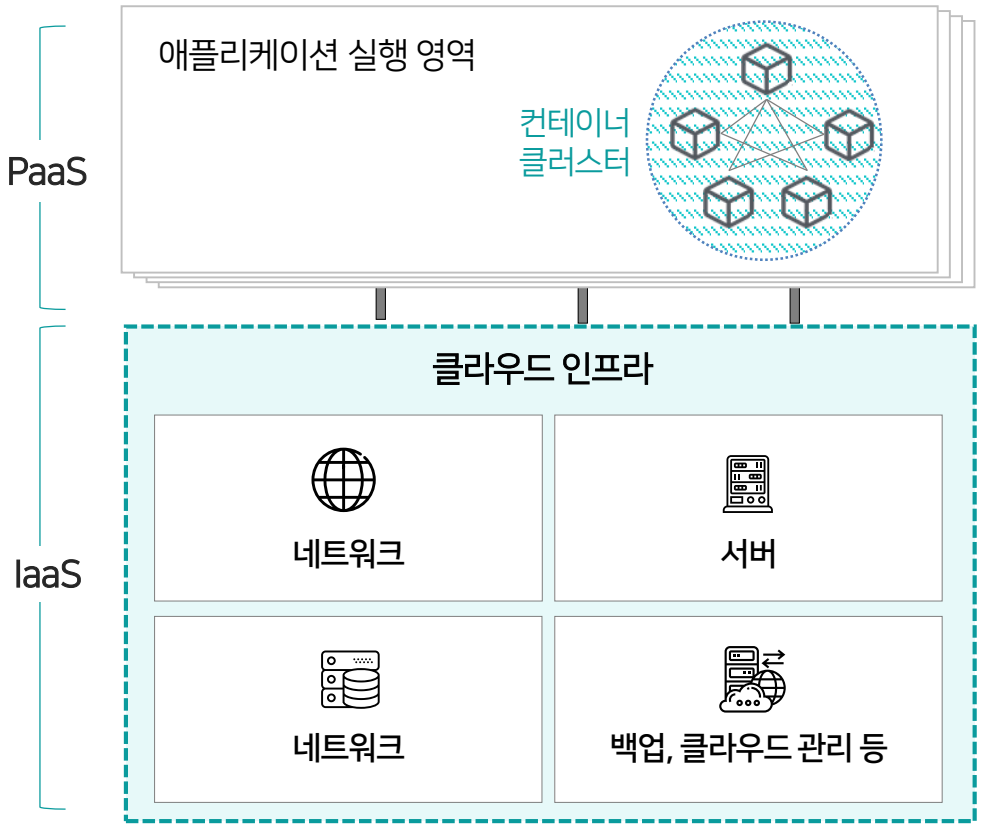
03 클라우드 네이티브 구성요소 및 원칙

3.8 클라우드 네이티브 애플리케이션 아키텍처

3.8.6 클라우드 인프라

- 클라우드 인프라는 서버, 스토리지, 네트워크 장비 등과 같은 인프라 구성요소로 클라우드의 IaaS에 해당한다.
- 클라우드 네이티브 환경에서 클라우드 인프라 위에 컨테이너와 같은 경량화된 가상화 기술 및 네트워크 및 스토리지 가상화를 적용하여 컨테이너 클러스터를 구축한다.

[그림 3-32] 클라우드 인프라



클라우드 네이티브 애플리케이션은 컨테이너 기반의 실행환경 외에도
마이크로서비스 구조의 애플리케이션을 개발, 운영할 수 있는
클라우드 플랫폼 서비스가 갖춰진 클라우드 환경이 필요함

클라우드 네이티브 정보시스템 구축을 위한

발주자 안내서

04

클라우드 네이티브 **적합성 검토**

04 클라우드 네이티브 적합성 검토

4.1 개요

- 공공부문 정보화사업 수행 시 클라우드 네이티브를 도입할 것인지 사전 의사결정이 필요하다. 정보화사업은 정보화사업계획, ISP 수립의 과정을 거쳐서 발주가 진행되므로 이 단계에 클라우드 네이티브 도입 적합성 검토 작업을 수행하도록 한다.
- 정보화사업계획 수립 시 기관 내부의 요구사항, 정보시스템 현황 및 이슈 등을 고려하여 클라우드 네이티브 적합성 검토를 수행하고, 도입 여부를 결정한다.

[그림 4-1] 단계별 클라우드 네이티브 적합성 검토

	주요 태스크	수행 내용	수행 주체
사업기획 단계	• 정보시스템별 상위 수준의 클라우드 네이티브 도입 적합성 검토	• 클라우드 네이티브 도입 적합성 검토 체크리스트를 통한 검토 수행 • 정보시스템 구축, 전환, 고도화 등의 배경, 주요 현황 및 이슈 요약 참조	• 발주자
	클라우드 네이티브 도입 목표 기반의 적합성 검토 체크리스트 활용		
ISP 컨설팅 단계	• 정보시스템별 서비스 영역별 클라우드 네이티브 적합성 검토 및 도입 방안 수립	• 정보시스템의 애플리케이션, 데이터, 기술 아키텍처 관점의 상세 현황 및 이슈 분석 - 예) 마이크로서비스 분해, 업무-데이터 상관 분석, CRUD 매트릭스, 서비스 호출 관계 등 • ISP 수행사는 상세 분석 내용을 참조하여 클라우드 네이티브 도입 방안 수립	• ISP 수행사 (제3자 관점)
시스템 구축 단계	• 정보시스템별 서비스 영역별 클라우드 네이티브 도입 최종 결정 • 서비스별 상세 설계 및 개발	• 클라우드 네이티브 도입이 결정된 상태에서 업무 분석 및 설계 시 최종적인 의사결정 수행 - 클라우드 네이티브 도입, 미도입 구분 • 사용자의 상세 요구사항, 정보시스템 상세 분석 등을 통해 서비스별 상세 설계 및 개발	• 발주자, 구축사업 수행사 - 업무 분석 및 설계 담당

※ 본 안내서는 사업기획 단계의 클라우드 네이티브 적합성 검토 방안을 제시함

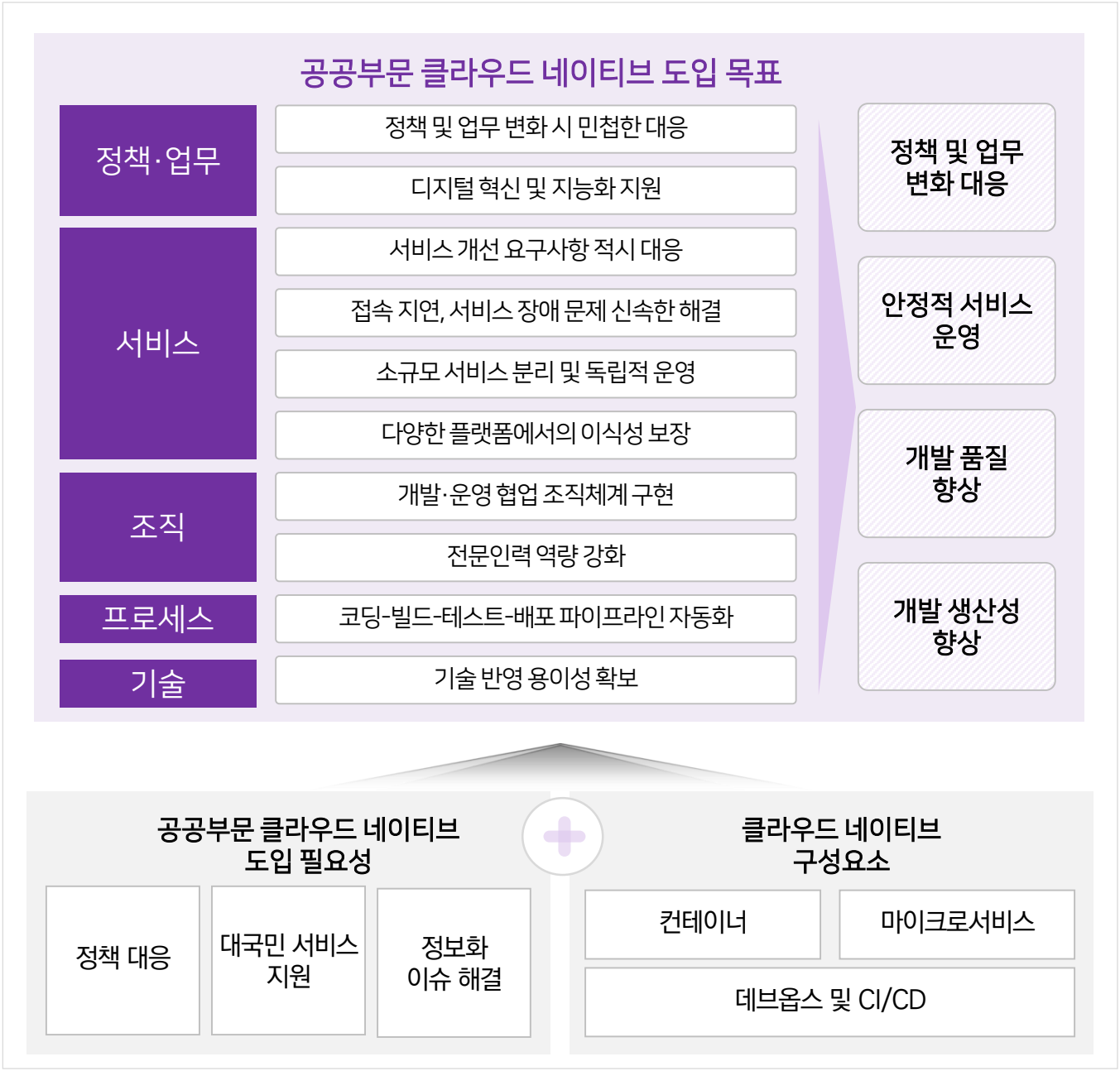
04 클라우드 네이티브 적합성 검토

4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.1 공공부문 클라우드 네이티브 도입 목표

- 클라우드 네이티브 도입 필요성과 구성요소의 특장점을 고려하여 정책·업무, 서비스, 조직, 프로세스, 기술 관점에서 공공부문 클라우드 네이티브 도입 목표를 정의하였다.
- 도입 목표는 정책 및 업무 변화 대응, 안정적 서비스 운영, 개발 품질 향상, 개발 생산성 향상이다.

[그림 4-2] 공공부문 클라우드 네이티브 도입 목표 도출



04 클라우드 네이티브 적합성 검토

4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.2 클라우드 네이티브 적합성 검토 항목 도출

- 공공부문 클라우드 네이티브 도입 목표를 토대로 클라우드 네이티브 적합성 검토 항목을 10가지 도출하였다.
- 클라우드 네이티브 적합성 검토 항목에 대한 설명은 다음과 같다.

[표 4-1] 클라우드 네이티브 적합성 검토 항목 설명

관점	적합성 검토 항목	설명
정책·업무	정책 및 업무 변화에 대한 민첩한 대응	• 한국판 뉴딜 등 새로운 정책, 기관별 업무계획 등에 따른 다양한 정보화 요구사항 변화에 대한 신속한 대응 수준
	디지털 혁신 및 지능화 지원	• 디지털 혁신 및 지능화 관련 사업(빅데이터, AI, IoT 등)을 효과적으로 도입하기 위해 클라우드 네이티브 기술이 필요한 정도
서비스	서비스 개선 요구사항 적시 대응	• 클라우드 네이티브 아키텍처, 개발·운영 프로세스 및 환경 등 구현을 통해 사용자의 다양한 서비스 개선 요구사항을 적시에 반영이 가능한 정도
	접속 지연, 서비스 장애 문제 신속한 해결	• 사용자의 증가에 따른 접속 지연, 서비스 장애 등 문제의 신속한 해결이 필요한 수준
	소규모 서비스 분리 및 독립적 운영	• 서비스 규모가 큰 경우, 소규모 서비스로 분리하여 손쉽게 개발 및 유지보수하고 독립적으로 운영하려는 요구의 수준
	다양한 플랫폼 환경에서의 이식성 보장	• 개발-테스트-운영 환경, 다양한 플랫폼 등에 서비스를 배포하고 이식이 필요한 정도
조직	개발·운영 협업 조직체계 구현	• 신속한 서비스 개발 및 서비스 제공을 위한 개발-운영 협업체계 지원 수준
	전문인력 역량 강화	• 클라우드 네이티브 환경에서 조직 간 협업을 지원하고, CI/CD 자동화 도구를 활용할 수 있는 전문인력의 확보 및 역량 수준
프로세스	코딩-빌드-테스트-배포 파이프라인 자동화	• 코딩-빌드-테스트-배포 파이프라인 상의 반복 작업의 자동화를 통한 배포주기 단축 및 애플리케이션 전달 속도 향상의 정도
기술	기술 반영 용이성 확보	• 오픈소스 SW 등 다양한 기술의 도입, 업그레이드, 업데이트 등 작업이 용이한 정도

04 클라우드 네이티브 적합성 검토

4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.2 클라우드 네이티브 적합성 검토 항목 도출

- 클라우드 네이티브 적합성 검토 항목과 클라우드 네이티브 구성요소인 컨테이너, 데브옵스 및 CI/CD, 마이크로서비스는 연관성을 가진다.
- 적합성 검토 항목에 대한 답변과 구성요소 간 연관성을 확인하여 최종적으로 도입할 구성요소를 결정한다.

[표 4-2] 클라우드 네이티브 적합성 검토 항목과 구성요소의 연관성

관점	적합성 검토 항목(도입 목표)	컨테이너	데브옵스 및 CI/CD	마이크로서비스
정책·업무	정책 및 업무 변화에 대한 민첩한 대응	●	●	●
	디지털 혁신 및 지능화 지원	●	●	●
서비스	서비스 개선 요구사항 적시 대응	●	●	●
	접속 지연, 서비스 장애 문제 신속한 해결	●	●	●
	소규모 서비스 분리 및 독립적 운영	●	●	●
	다양한 플랫폼 환경에서의 이식성 보장	●	●	●
조직	개발·운영 협업 조직체계 구현	-	●	●
	전문인력 역량 강화	●	●	●
프로세스	코딩-빌드-테스트-배포 파이프라인 자동화	●	●	-
기술	기술 반영 용이성 확보	●	-	●

04 클라우드 네이티브 적합성 검토

4.2 클라우드 네이티브 적합성 검토 체크리스트

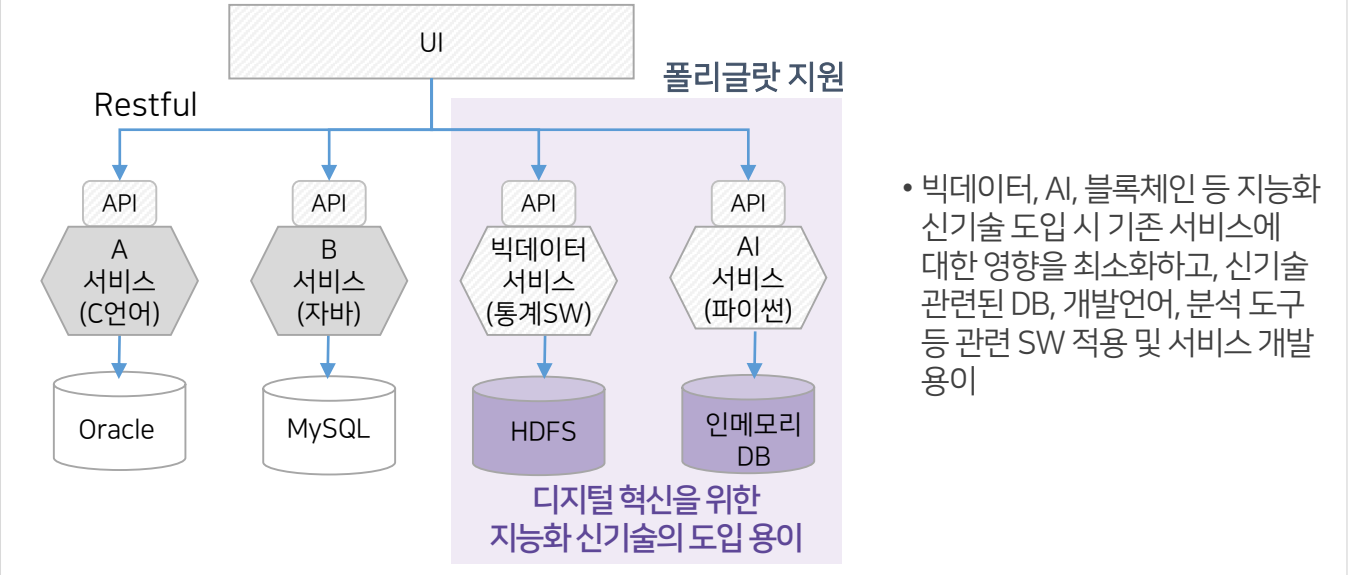
4.2.3 클라우드 네이티브 적합성 검토 항목 설명

4.2.3.1 정책 · 업무 관점

- 정책 및 업무 관점의 검토 항목은 정책 및 업무 변화에 대한 민첩한 대응, 디지털 혁신 및 지능화 지원이다.

[그림 4-3] 정책 및 업무 변화에 대한 민첩한 대응

[그림 4-4] 디지털 혁신 및 지능화 지원



04 클라우드 네이티브 적합성 검토

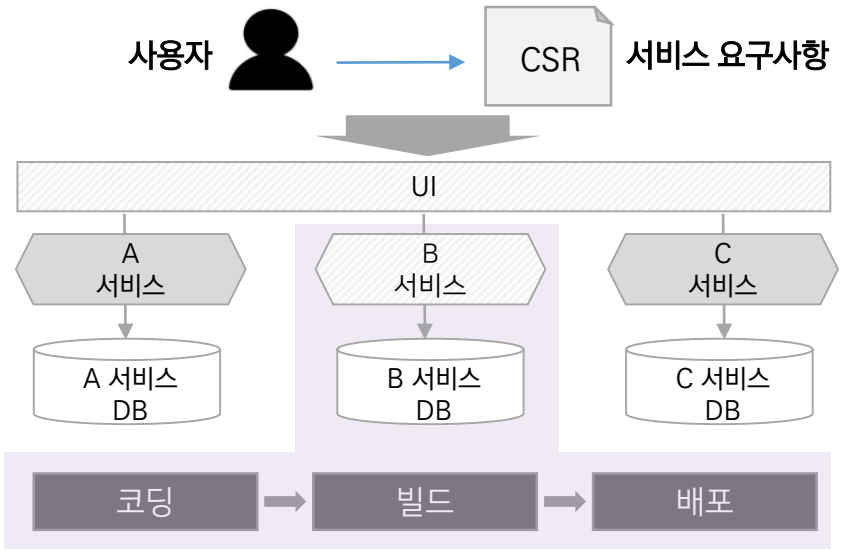
4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.3 클라우드 네이티브 적합성 검토 항목 설명

4.2.3.2 서비스 관점

- 서비스 관점의 검토 항목은 서비스 요구사항 적시 대응, 접속 지연 및 서비스 장애 문제 신속한 해결, 소규모 서비스 분리 및 독립적 운영, 다양한 플랫폼에서의 이식성 보장이다.

[그림 4-5] 서비스 요구사항 적시 대응

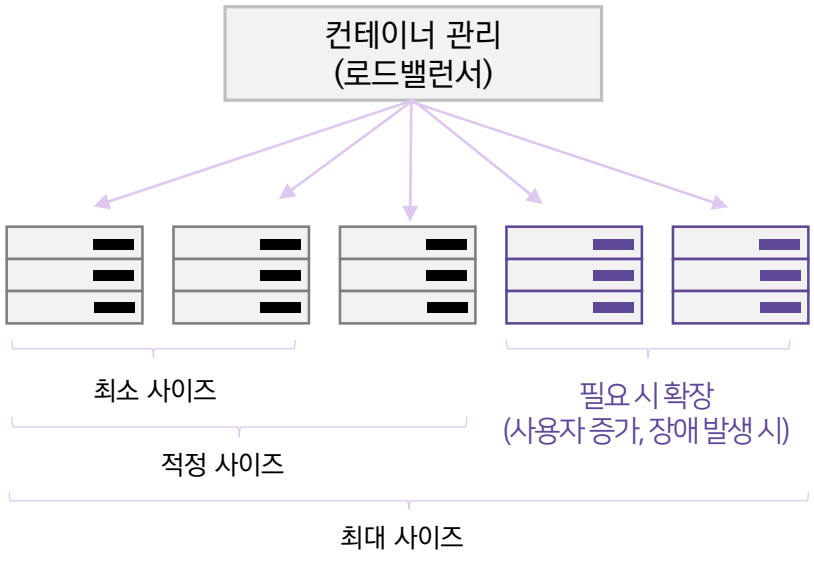


서비스 요구사항 발생 시 소규모 서비스에 대한 신속한 코딩, 빌드, 배포

- 사용자의 개선 요구사항(CSR)에 대해 전체 서비스가 아닌 해당 마이크로서비스에 대한 코딩, 빌드, 배포를 통한 적시 대응 가능

※ CSR(Customer Service Request, 고객의 서비스 요구사항)

[그림 4-6] 접속 지연, 서비스 장애 문제 신속한 해결



컨테이너 오토스케일링을 통한 가용성 보장

- 서비스 이용자의 증가, 서비스 장애 등 다양한 원인에 의해 서비스 및 자원 확장이 필요한 경우 컨테이너 관리 기능을 활용해 서비스와 해당 리소스를 즉시 확장 가능함
- 서비스의 규모가 작을수록 필요한 리소스가 줄어듦

04 클라우드 네이티브 적합성 검토

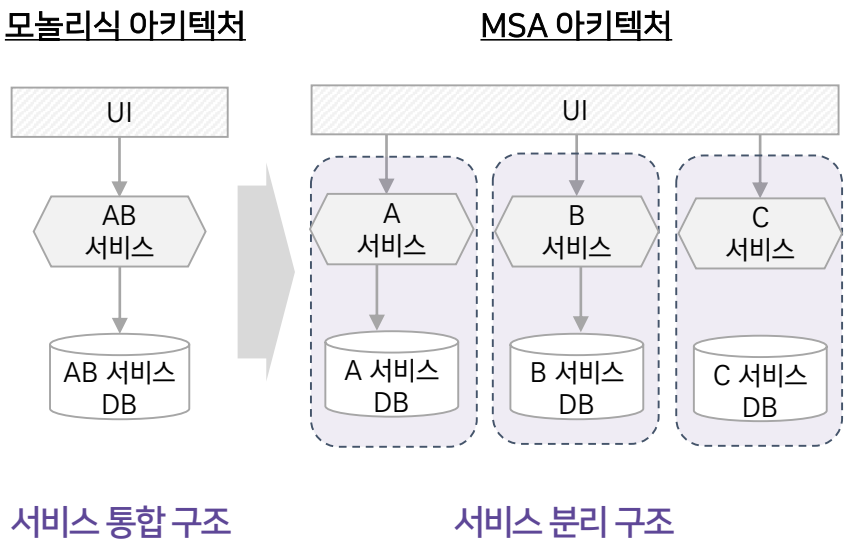
4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.3 클라우드 네이티브 적합성 검토 항목 설명

4.2.3.2 서비스 관점

- 서비스 관점의 검토 항목은 서비스 요구사항 적시 대응, 접속 지연 및 서비스 장애 문제 신속한 해결, 소규모 서비스 분리 및 독립적 운영, 다양한 플랫폼에서의 이식성 보장이다.

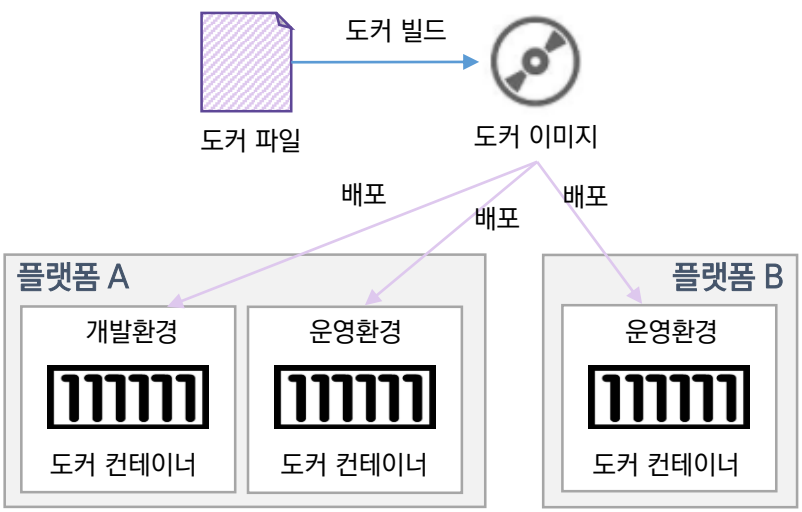
[그림 4-7] 소규모 서비스 분리 및 독립적 운영



MSA 아키텍처 기반 소규모 서비스 운영

- 기존의 모놀리식 아키텍처의 통합 서비스를 작은 서비스로 분리하여 독립적으로 운영될 수 있는 구조로 설계
- 각 서비스는 API를 이용하여 다른 서비스와 연결됨

[그림 4-8] 다양한 플랫폼에서의 이식성 보장



도커 컨테이너를 활용한 배포 또는 이식(예시)

- 도커 이미지는 서비스 운영에 필요한 서버 프로그램, 소스코드 및 라이브러리, 컴파일된 실행파일 등을 패키징한 것임
- 도커 이미지는 다양한 플랫폼의 개발·운영환경에 배포되고, 도커 이미지를 실행할 컨테이너를 생성하고 실행함

04 클라우드 네이티브 적합성 검토

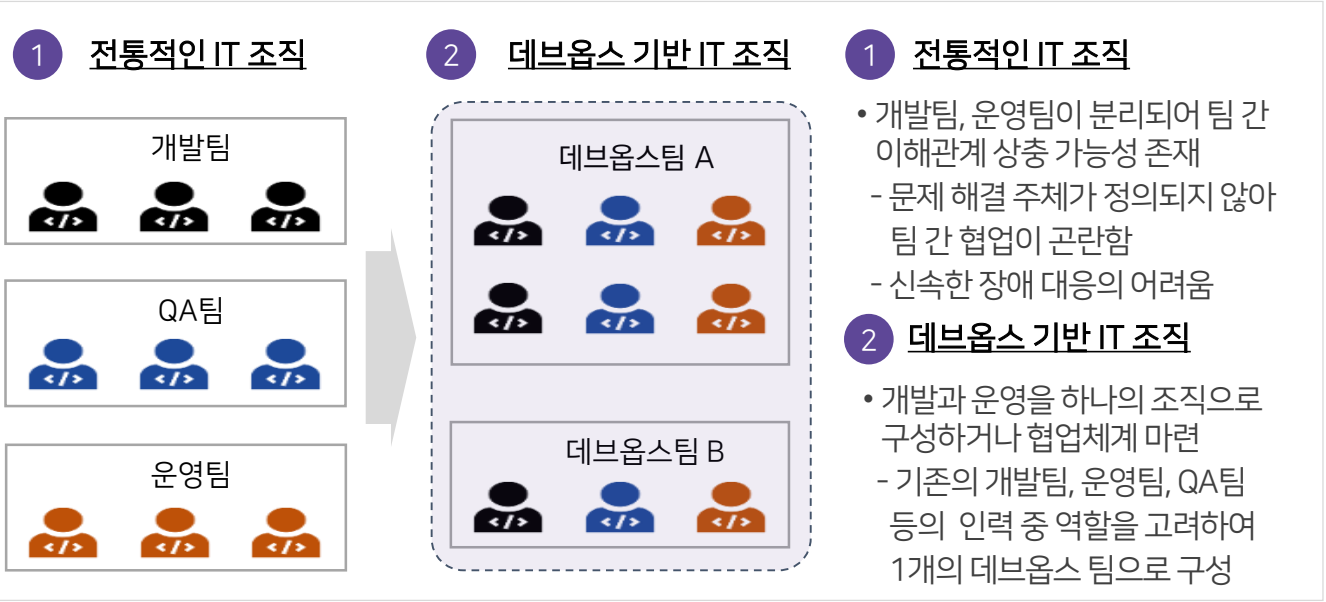
4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.3 클라우드 네이티브 적합성 검토 항목 설명

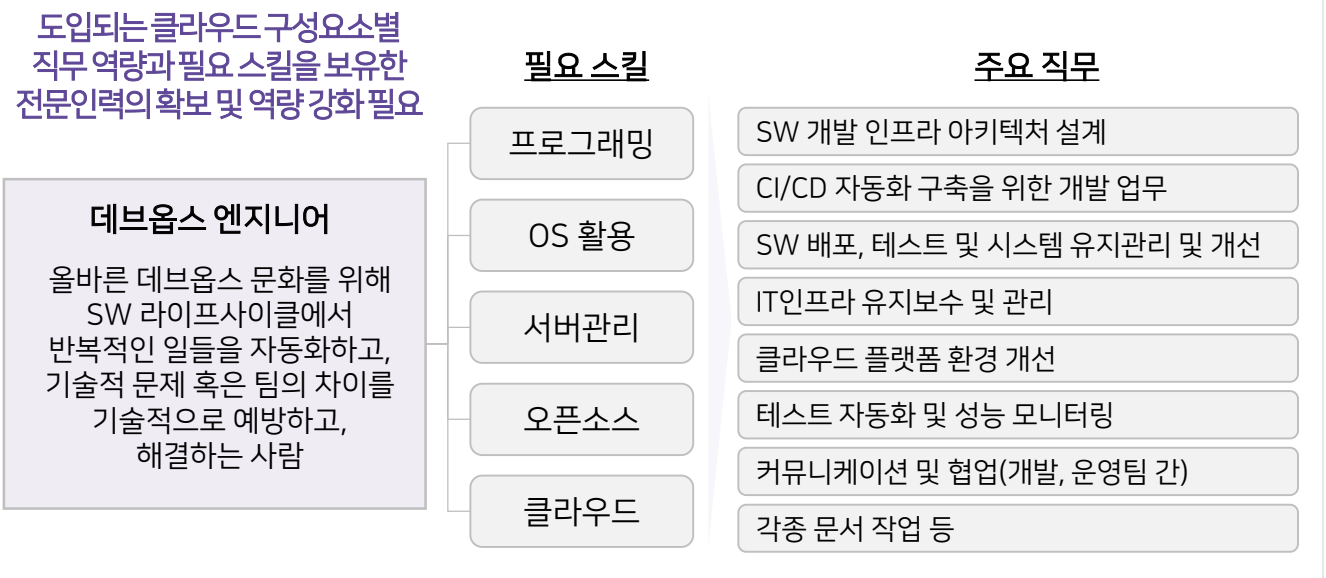
4.2.3.3 조직 관점

- 조직 관점의 검토 항목은 개발·운영 협업 조직체계 구현과 전문인력 역량 강화이다.

[그림 4-9] 소규모 서비스 분리 및 독립적 운영



[그림 4-10] 전문인력 역량 강화



04 클라우드 네이티브 적합성 검토

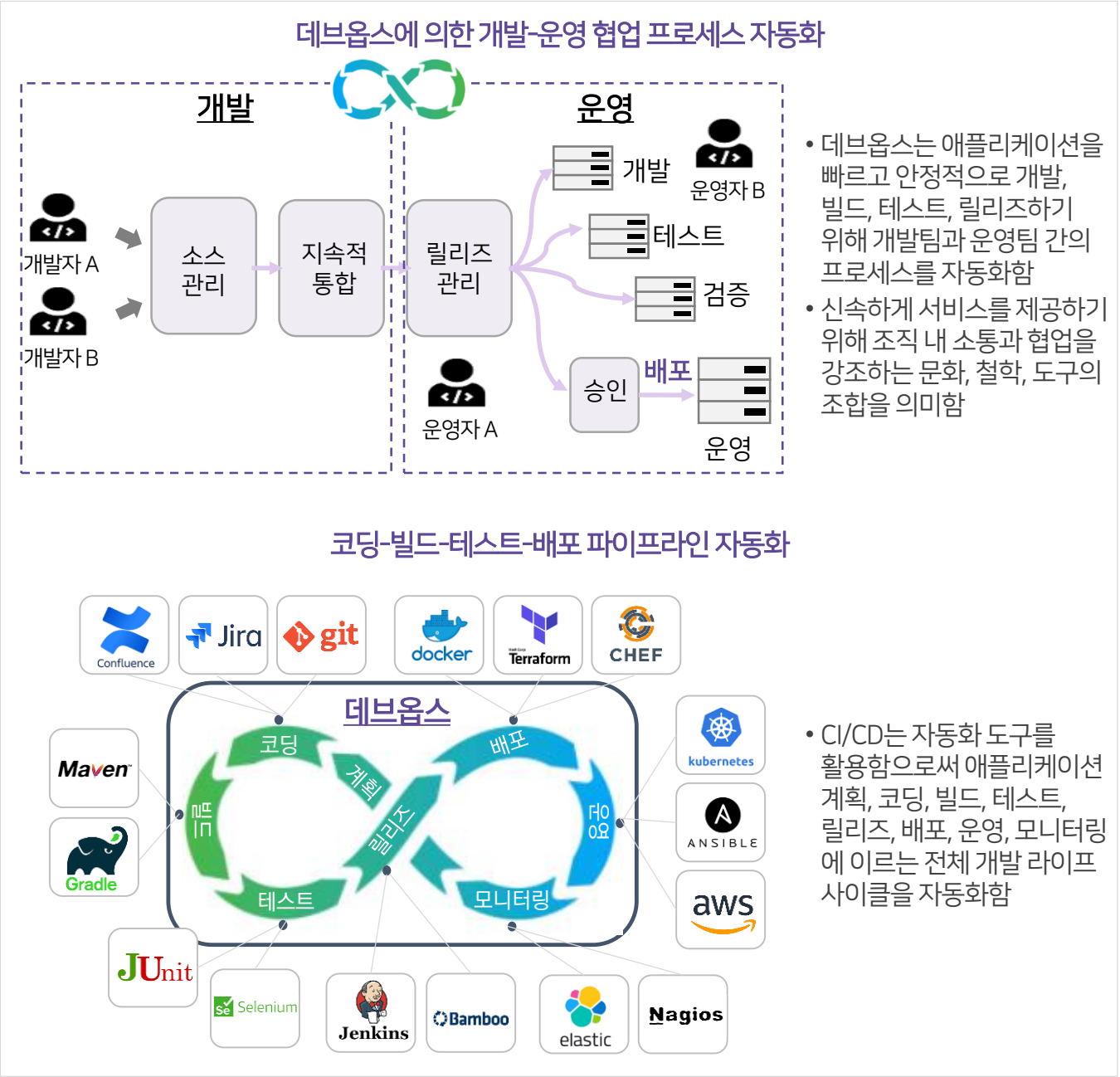
4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.3 클라우드 네이티브 적합성 검토 항목 설명

4.2.3.4 프로세스 관점

- 프로세스 관점의 검토 항목은 데브옵스 협업체계 기반 코딩-빌드-테스트-배포 파이프라인의 자동화이다.

[그림 4-11] 코딩-빌드-테스트-배포 파이프라인 자동화



04 클라우드 네이티브 적합성 검토

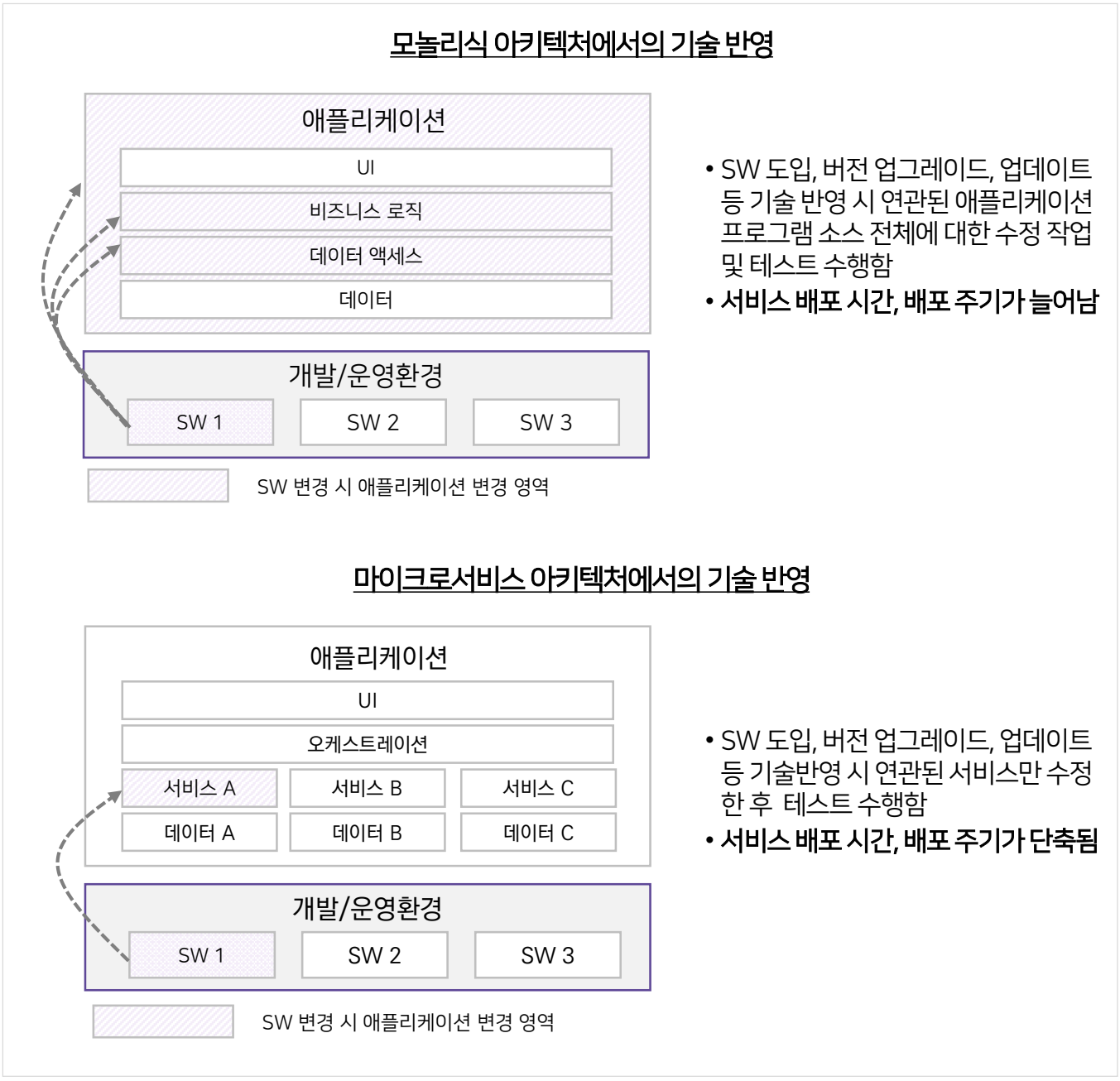
4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.3 클라우드 네이티브 적합성 검토 항목 설명

4.2.3.5 기술 관점

- 기술 관점의 검토 항목은 기술 반영의 용이성이다.

[그림 4-12] 기술 반영의 용이성



04 클라우드 네이티브 적합성 검토

4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.4 클라우드 네이티브 적합성 검토 체크리스트

4.2.4.1 적합성 검토 체크리스트

- 각 기관에서 클라우드 네이티브 도입이 필요한지 적합성을 검토할 수 있도록 체크리스트(질의사항)를 도출하였다.
- 적합성 검토 체크리스트는 정책 및 업무 변화 대응, 안정적 서비스 운영, 개발 품질 향상, 개발 생산성 향상 등 4개 목표에 대해 총 10개 질의사항으로 구성된다.

[표 4-3] 적합성 검토 체크리스트 도출 결과

목표 구분	적합성 검토 항목	질의사항
정책 및 업무 변화 대응	정책 및 업무 변화에 대한 민첩한 대응	1-1. 한국판 뉴딜과 같은 새로운 정책, 기관별 업무계획 등 다양한 정보화 요구사항 변화에 대해 신속한 대응이 필요합니까?
	디지털 혁신 및 지능화 지원	1-2. 디지털 혁신 및 지능 정보화 관련 신기술(빅데이터, AI, 블록체인, IoT 등)의 도입이 필요하고, 연관된 애플리케이션의 개발 또는 개선이 요구됩니까?
안정적 서비스 운영	서비스 개선 요구사항 적시 대응	2-1. 서비스 이용자의 잦은 CSR(Customer Service Request, 고객의 서비스 요구사항)에 대해 적시 대응이 필요합니까? 또는 초기 개발비의 약 15% 이상을 매년 추가개발 및 유지보수 비용으로 사용하고 있습니까?
	접속 지연, 서비스 장애 문제 신속한 해결	2-2. 다양한 원인에 의한 장애 발생 시 장애복구(예.시스템 증설, 업그레이드 등)를 위해 서비스를 중단한 적이 있습니까? 또는 이용자의 폭증에 의한 접속지연으로 이용자의 불만이 제기된 적이 있습니까?
	소규모 서비스 분리 및 독립적 운영	2-3. 소규모 서비스 단위로 기능과 DB가 명확하게 분리되고, 독립적으로 서비스를 실행할 수 있습니까? (공통 기능 및 데이터 사용 유무, 타 시스템과의 연계성, 서비스 의존 관계 등 확인)
	다양한 플랫폼 환경에서의 이식성 보장	2-4. 개발-테스트-운영 환경에서 안정적이고 지속적인 배포가 필요하거나 다양한 플랫폼 환경에 애플리케이션 배포가 요구됩니까?
개발 품질 향상	개발·운영 협업 조직체계 구현	3-1. 시스템 개발 및 운영 시 개발팀과 운영팀의 분리에 의한 의사소통의 문제, 개발 및 배포 지연 등의 문제가 존재합니까?
	전문인력 역량 강화	3-2. 개발-빌드-배포 과정의 품질 향상을 위해 전문 인력을 확보하거나 기존 인력에 대한 전문적인 교육이 필요합니까?
개발 생산성 향상	코딩-빌드-테스트-배포 파이프라인 자동화	4-1. 개발된 SW를 형상관리 시스템에 커밋한 후 개발/검증/운영 서버에서 각각 빌드, 테스트, 배포하는 과정 전체에 대해 자동화 도구를 도입하거나 추가할 필요가 있습니까?
	기술 반영 용이성 확보	4-2. 오픈소스 SW를 비롯한 다양한 기술의 도입, 업그레이드, 업데이트 등 작업 수행 시 연관된 서비스에 대한 변경 작업이 복잡하여 개발 상의 어려움을 겪은 적이 있습니까?

04 클라우드 네이티브 적합성 검토

4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.4 클라우드 네이티브 적합성 검토 체크리스트

4.2.4.2 클라우드 진단 관련 참조 도구

[그림4-13] 공공 부문 클라우드 도입 적합성 자가진단도구(TTA)

TTA에서 발표한 공공부문 클라우드 도입 적합성 자가진단 및 유형별 도입 지침 해설서는 클라우드 서비스 도입 시 실무 담당자가 활용할 수 있는 체크리스트와 활용방안을 제시하고 있으며, 세부목표별로 10개의 중분류 항목을 정의하고 있다.

대분류	중분류	중분류 설명
정책적 요인	정책 부합성	• 도입하고자 하는 클라우드 서비스가 정부 정책 방향에 부합하는 정도
	목표 부합성	• 도입하고자 하는 클라우드 서비스가 추진 기관의 조직 목표와의 부합 정도 및 새로운 가치 창출 여부
	법·제도 부합성	• 클라우드 서비스를 추진하기 위해 현재 법·제도하에서의 적용 가능성이나 향후 개선 가능성에 대한 정도
프로세스 및 조직적 요인	업무 적합성	• 도입하고자 하는 클라우드 서비스가 기존 업무 특성 및 업무 프로세스를 반영하고 있는지 여부
	프로세스 및 표준화	• 클라우드 서비스 도입을 위한 내부 프로세스, 계약·조달 프로세스 및 관리 체계, 표준화 수준 정도
	조직 인지성	• 클라우드 도입에 필요한 내부 조직의 준비 상황 및 교육 수준 정도
	비용 경제성	• 클라우드 도입에 따른 도입, 운용상의 효율성, 유지 관리 측면의 비용 경제성 여부
기술적 요인	가용성 및 유연성	• 도입하고자 하는 클라우드 서비스의 부하 패턴 변화에 대한 대응, 안정적인 서비스 제공 가능성 여부
	기술 용이성	• 도입하고자 하는 클라우드 서비스가 일반적인 정보화 기술요소를 수용하기에 용이한 정도
	보안 충족성	• 도입하고자 하는 클라우드 서비스 특성에 따른 데이터의 저장 위치, 물리적인 시스템 접근 통제 등 데이터 보안 관련 요인에 대한 충족 여부

[출처: TTA 공공부문 클라우드 서비스 도입적합성 자가진단 및 유형별 도입 지침 해설서]

04 클라우드 네이티브 적합성 검토

4.2 클라우드 네이티브 적합성 검토 체크리스트

4.2.4 클라우드 네이티브 적합성 검토 체크리스트

4.2.4.2 클라우드 진단 관련 참조 도구

[그림4-14] 클라우드 애플리케이션 성숙도 체크리스트

넷플릭스 플랫폼팀의 Andrew Spyker는 클라우드 네이티브 애플리케이션 성숙도에 관한 체크리스트를 제시하며, 아래의 10개 질문에 모두 "YES"로 응답할 경우, 클라우드 네이티브 수준이라고 설명했다.

구분	질문
재배포	① 전체 애플리케이션을 수 분내에 재배포할 수 있는가? Can you redeploy your entire application in minutes?
설치	② 애플리케이션이 자동 설치의 일부가 아닌 특정 IP, 포트, 파일 시스템에 종속적인가? Does your application depend on specific IP addresses, ports, file systems, that are not part of the automated installation?
장애복구	③ 애플리케이션이 중지되지 않고, 인프라 장애를 자동 복구할 수 있는가? Can you application survive, and auto-recover from, infrastructure(compute, network, storage) failures?
업그레이드/ 다운그레이드	④ 사용자에게 영향을 주지 않고 애플리케이션의 업그레이드 또는 다운그레이드가 이뤄지는가? Can you upgrade and downgrade, your application(or parts of the application) without any impact to users?
애플리케이션 실행	⑤ 동일한 환경에서 동시에 여러 버전의 애플리케이션 서비스를 실행할 수 있는가? Can you run multiple versions of your application services, in the same environment at the same time?
테스트	⑥ 프로덕션 환경에서 안전하게 테스트할 수 있는가? Can you Safely test in production?
장애 대응	⑦ 애플리케이션의 일부에 장애가 발생하더라도 다른 부분으로 정상 가동할 수 있는가? If a part of an application fails, will other parts continue to operate?
자동 스케일업/다운	⑧ 애플리케이션의 일부가 사용자 부하 또는 기타 요소에 따라 자동으로 스케일업 및 스케일다운할 수 있습니까? Can parts of your application scale-up and scale-down automatically, based on user load or other factors?
클라우드 제공자 관련 배포	⑨ 클라우드 제공자 간 애플리케이션 구성요소들을 배포할 수 있는가? Can you deploy application components across cloud providers?
	⑩ 하나의 애플리케이션 구성요소를 다른 클라우드 제공자에게 배포할 수 있는가? Can you deploy an application component on a different cloud provider?

[출처: Andrew Spyker, 넷플릭스 플랫폼팀, 전 IBM 소속]

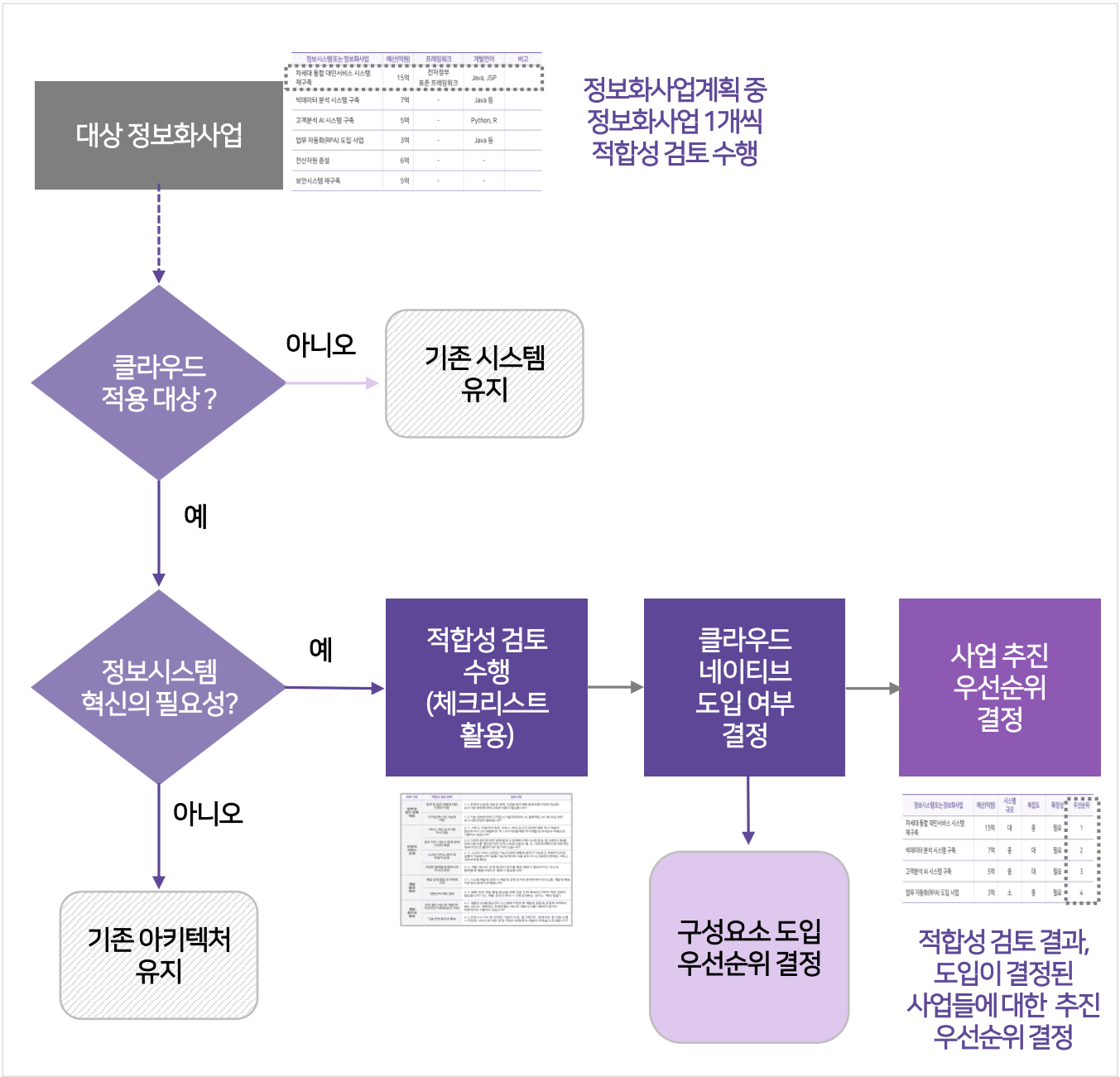
04 클라우드 네이티브 적합성 검토

4.3 클라우드 네이티브 적합성 검토 수행

4.3.1 클라우드 네이티브 적합성 검토 수행 개요

- 클라우드 네이티브 기반 정보화사업을 추진하고자 할 경우, 클라우드 네이티브 도입 적합성 검토 체크리스트를 활용하여 자가진단을 수행한 후, 진단 결과에 따라 클라우드 네이티브 도입 여부와 구성요소를 결정하도록 한다.

[그림 4-15] 클라우드 네이티브 적합성 검토 수행 절차



04 클라우드 네이티브 적합성 검토

4.3 클라우드 네이티브 적합성 검토 수행

4.3.2 클라우드 적합성 검토 후 도입 여부 결정

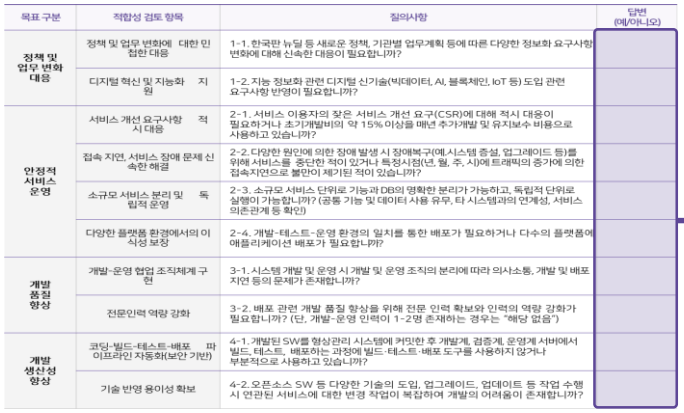
- 공공기관은 클라우드 네이티브 적합성 검토 체크리스트를 통해 자가진단을 수행한 후 “예” 응답 수에 따라 클라우드 네이티브 도입 여부를 결정하도록 하며, 5개 이상 “예”로 응답된 경우에 클라우드 네이티브를 도입하도록 한다.

[그림 4-16] 클라우드 네이티브 적합성 검토 후 도입여부 결정

클라우드 네이티브 적합성 검토 수행

정보화사업별 클라우드 네이티브 적합성 검토

정보화사업
1개씩 적합성 검토 수행



목표 구분	적합성 검토 항목	질의사항	답변 (예/아니오)
정책 및 업무 변화 대응	정책 및 업무 변화에 대한 민첩한 대응	1-1. 한국판 뉴딜 등 새로운 정책, 기관별 업무계획 등에 따른 다양한 정보화 요구사항 변화에 대해 신속한 대응이 필요합니까?	
	디지털 혁신 및 지능화 지원	1-2. 지능 정보화 관련 디지털 신기술(빅데이터, AI, 블록체인, IoT 등) 도입 관련 요구사항 반영이 필요합니까?	
안정적 서비스 운영	서비스 개선 요구사항 적시 대응	2-1. 서비스 이용자의 잦은 서비스 개선 요구(CSR)에 대해 적시 대응이 필요하거나 초기개발비의 약 15% 이상을 매년 추가개발 및 유지보수 비용으로 사용하고 있습니까?	
	접속 지연, 서비스 장애 문제 신속한 해결	2-2. 다양한 원인에 의한 장애 발생 시 장애복구(예, 시스템 증설, 업그레이드 등)를 위해 서비스를 중단한 적이 있거나 특정시점(년, 월, 주, 시)에 트래픽의 증가에 의한 접속지연으로 불만이 제기된 적이 있습니까?	
	소규모 서비스 분리 및 독립적 운영	2-3. 소규모 서비스 단위로 기능과 DB의 명확한 분리가 가능하고, 독립적 단위로 실행이 가능합니까? (공통 기능 및 데이터 사용 유무, 타 시스템과의 연계성, 서비스 의존관계 등 확인)	
	다양한 플랫폼 환경에서의 이식성 보장	2-4. 개발-테스트-운영 환경의 일치를 통한 배포가 필요하거나 다수의 플랫폼에 애플리케이션 배포가 필요합니까?	
개발 품질 향상	개발-운영 협업 조직체계 구현	3-1. 시스템 개발 및 운영 시 개발 및 운영 조직의 분리에 따라 의사소통, 개발 및 배포 지연 등의 문제가 존재합니까?	
	전문인력 역량 강화	3-2. 배포 관련 개발 품질 향상을 위해 전문 인력 확보와 인력의 역량 강화가 필요합니까? (단, 개발-운영 인력이 1-2명 존재하는 경우는 “해당 없음”)	
개발 생산성 향상	코딩-빌드-테스트-배포 파이프라인 자동화(보안 기반)	4-1. 개발된 SW를 향상관리 시스템에 커밋한 후 개발계, 검증계, 운영계 서버에서 빌드, 테스트, 배포하는 과정에 빌드-테스트-배포 도구를 사용하지 않거나 부분적으로 사용하고 있습니까?	
	기술 반영 용이성 확보	4-2. 오픈소스 SW 등 다양한 기술의 도입, 업그레이드, 업데이트 등 작업 수행 시 연관된 서비스에 대한 변경 작업이 복잡하여 개발의 어려움이 존재합니까?	

체크리스트를 활용하여 자가진단을 수행한 후 “예”로 응답한 개수 집계

클라우드 네이티브 도입여부 결정 기준

적합성 검토 자가진단을 수행한 결과, “예” 응답 수가 5개 이상인 경우에 클라우드 네이티브 도입을 검토하도록 함

“예” 응답 수	도입여부 결정 기준	비고
1개 ~ 3개	• 기존의 애플리케이션 아키텍처 유지	
4개	• 기존의 애플리케이션 아키텍처 유지 • 필요시 클라우드 네이티브 구성요소 일부 도입 가능	• 적합성 검토 항목의 답변을 기준으로 클라우드 네이티브 구성요소를 부분적으로 도입
5개	• 클라우드 네이티브 도입 가능	
6개 ~ 8개	• 적극적으로 클라우드 네이티브 도입	
9개 ~ 10개	• 매우 적극적으로 클라우드 네이티브 도입	• 우선적으로 정보화사업 추진

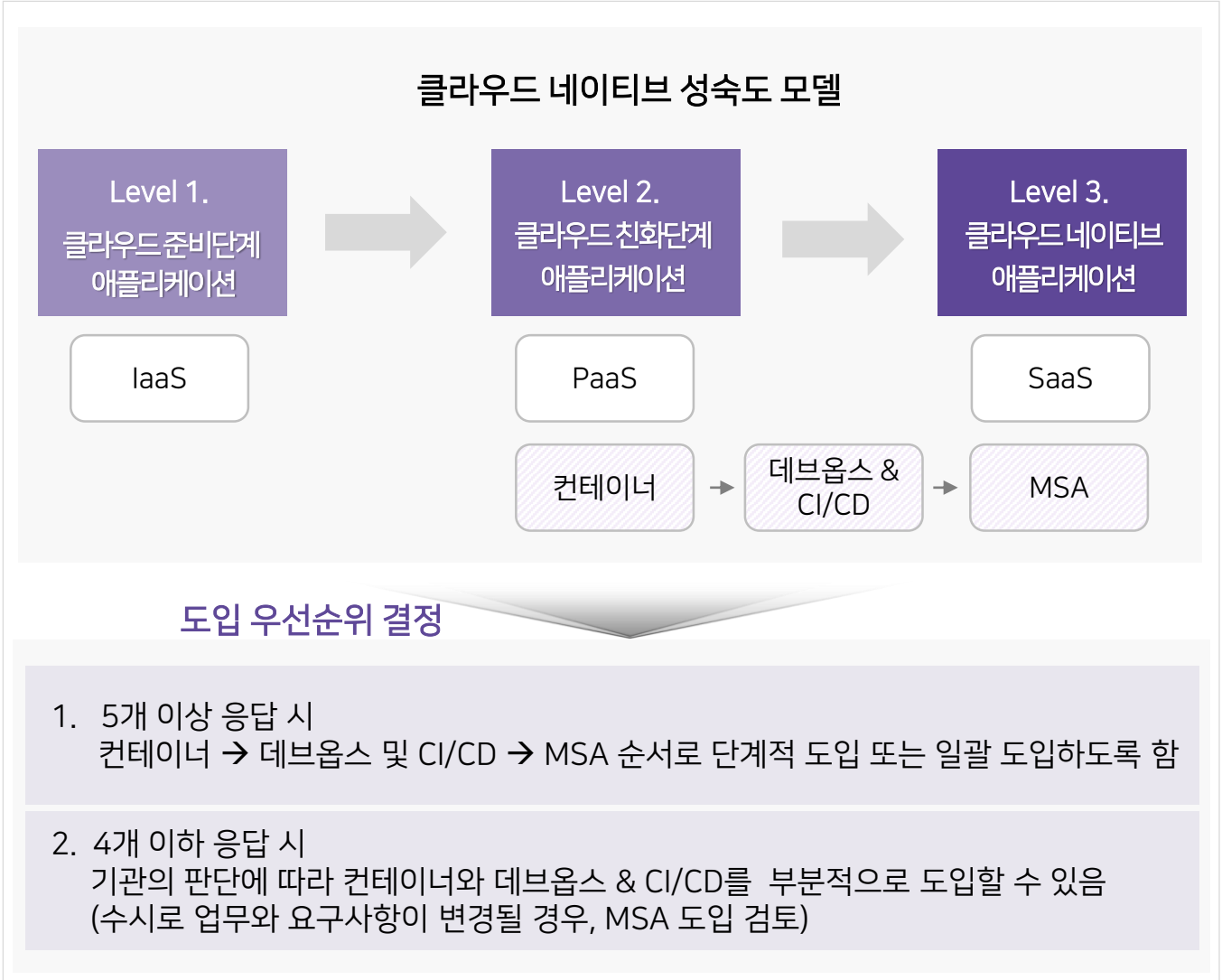
04 클라우드 네이티브 적합성 검토

4.3 클라우드 네이티브 적합성 검토 수행

4.3.3 클라우드 네이티브 도입 우선순위 결정

- 클라우드 네이티브 적합성 검토를 통해 클라우드 네이티브 도입이 결정되면 클라우드 네이티브 성숙도 모델을 고려하여 구성요소의 도입 우선순위를 결정하도록 한다.
- 구성요소별 도입 우선순위는 컨테이너 → 데브옵스 및 CI/CD → MSA의 순서로 단계적으로 도입하거나 일괄 도입도 가능하다.
- 적합성 검토 결과, "예" 응답 수가 4개 이하인 경우에 기관의 판단에 따라 컨테이너, 데브옵스 및 CI/CD를 부분적으로 도입할 수 있다. MSA는 업무와 요구사항이 자주 변경되는 경우 또는 컨테이너와 데브옵스 및 CI/CD를 구축한 이후에 도입하도록 한다.

[그림 4-17] 클라우드 네이티브 구성요소 도입 우선순위 결정



04 클라우드 네이티브 적합성 검토

4.3 클라우드 네이티브 적합성 검토 수행

4.3.4 클라우드 네이티브 사업 추진 우선순위 결정

- 클라우드 네이티브 도입이 적합한 것으로 결정된 정보화사업들에 대해 시스템 규모, 복잡도, 확장 필요성을 고려하여 사업 추진 우선순위를 정한다. 사업의 규모가 크고, 복잡하며, 확장의 한계가 있는 경우 정보화사업을 우선적으로 수행하도록 하며, 각 기관의 사정에 따라 사업 추진 우선순위는 조정 가능하다.

[그림 4-18] 클라우드 네이티브 사업 추진 우선순위 결정

클라우드 네이티브 사업 추진 우선순위 결정 요소

클라우드 네이티브 사업 추진 우선순위 결정 예시

정보시스템 또는 정보화사업	예산 (억원)	시스템 규모	복잡도	확장 필요성	합계	우선순위
차세대 통합 대민서비스 시스템 재구축	15억	5	5	5	15	1
빅데이터 분석 시스템 구축	7억	4	4	4	12	2
고객분석 AI 시스템 구축	5억	3	3	4	10	3
업무 자동화(RPA) 도입 사업	3억	2	3	3	8	4

- 시스템 규모, 복잡도, 확장 필요성을 5점 척도 기준으로 평가하여 합산 점수의 순서대로 사업 추진 우선순위 결정
- 각 기관의 사정에 따라 우선순위 조정 가능

04 클라우드 네이티브 적합성 검토

4.4 클라우드 네이티브 적합성 검토 예시

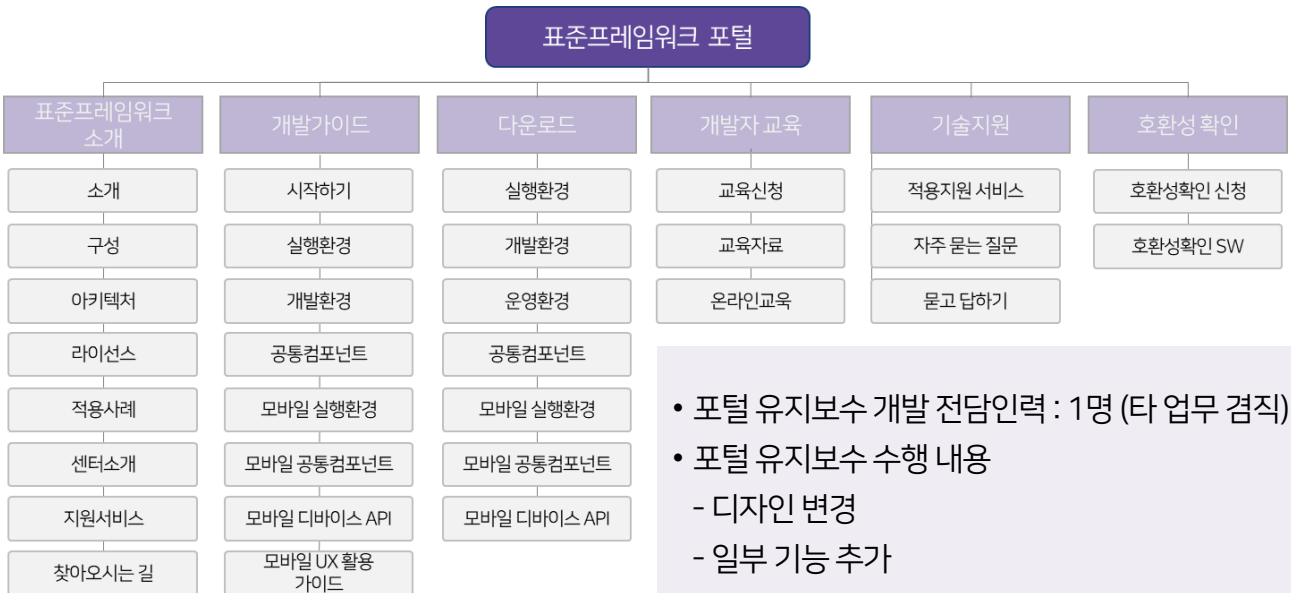
4.4.1 전자정부 표준프레임워크 포털

- 전자정부 표준프레임워크 포털은 공공 정보시스템 구축 시 활용할 수 있는 개발·운영 표준 환경을 제공하기 위해 개발 가이드, 다운로드, 개발자 교육, 기술지원 등의 서비스를 제공하고 있다.

[그림 4-19] 전자정부 표준프레임워크포털 화면 및 주요 기능

홈페이지 화면

홈페이지 구성 체계



- 포털 유지보수 개발 전담인력 : 1명 (타 업무 겸직)
- 포털 유지보수 수행 내용
  - 디자인 변경
  - 일부 기능 추가
  - 보안점검, 호환성 체크 관련 사항 수정

04 클라우드 네이티브 적합성 검토

4.4 클라우드 네이티브 적합성 검토 예시

4.4.1 전자정부 표준프레임워크 포털

- 전자정부 표준프레임워크 포털에 대해 클라우드 네이티브 적합성 검토 결과, “예” 응답 수가 3개로 집계되어 기존 아키텍처의 유지가 필요한 것으로 보여진다.

[표 4-4] 전자정부 표준프레임워크포털의 클라우드 네이티브 적합성 검토 결과

목표 구분	적합성 검토 항목	질의사항	답변	근거
정책 및 업무 변화 대응	정책 및 업무 변화에 대한 민첩한 대응	1-1. 한국판 뉴딜과 같은 새로운 정책, 기관별 업무계획 등 다양한 정보화 요구사항 변화에 대해 신속한 대응이 필요합니까?	아니오	• 해당 없음
	디지털 혁신 및 지능화 지원	1-2. 디지털 혁신 및 지능 정보화 관련 신기술(빅데이터, AI, 블록체인, IoT 등)의 도입이 필요하고, 연관된 애플리케이션의 개발 또는 개선이 요구됩니까?	아니오	• 해당 없음
안정적 서비스 운영	서비스 개선 요구사항 적시 대응	2-1. 서비스 이용자의 잦은 CSR(Customer Service Request, 고객의 서비스 요구사항)에 대해 적시 대응이 필요합니까? 또는 초기 개발비의 약 15% 이상을 매년 추가개발 및 유지보수 비용으로 사용하고 있습니까?	예	• 행안부와 NIA의 요구사항 발생 시 (보안 점검, 호환성 체크 등)
	접속 지연, 서비스 장애 문제 신속한 해결	2-2. 다양한 원인에 의한 장애 발생 시 장애복구(예.시스템 증설, 업그레이드 등)를 위해 서비스를 중단한 적이 있습니까? 또는 이용자의 폭증에 의한 접속지연으로 이용자의 불만이 제기된 적이 있습니까?	아니오	• 장애 발생하지 않음 (G-클라우드 구성 관련 서비스 중단 시 사전 공지한 적 있음)
	소규모 서비스 분리 및 독립적 운영	2-3. 소규모 서비스 단위로 기능과 DB가 명확하게 분리되고, 독립적으로 서비스를 실행할 수 있습니까? (공통 기능 및 데이터 사용 유무, 타 시스템과의 연계성, 서비스 의존 관계 등 확인)	예	• 서비스 단위로 기능과 서비스 분리 가능
	다양한 플랫폼 환경에서의 이식성 보장	2-4. 개발-테스트-운영 환경에서 안정적이고 지속적인 배포가 필요하거나 다양한 플랫폼 환경에 애플리케이션 배포가 요구됩니까?	아니오	• 개발-테스트-운영 환경에 배포하지만 타 플랫폼 이식은 없음
개발 품질 향상	개발·운영 협업 조직체계 구현	3-1. 시스템 개발 및 운영 시 개발팀과 운영팀의 분리에 의한 의사소통의 문제, 개발 및 배포 지연 등의 문제가 존재합니까?	아니오	• 개발자 1명 (유지보수 10%, 운영 90%)
	전문인력 역량 강화	3-2. 개발-빌드-배포 과정의 품질 향상을 위해 전문 인력을 확보하거나 기존 인력에 대한 전문적인 교육이 필요합니까?	아니오	• 해당 없음
개발 생산성 향상	코딩-빌드-테스트-배포 파이프라인 자동화	4-1. 개발된 SW를 형상관리 시스템에 커밋한 후 개발/검증/운영 서버에서 각각 빌드, 테스트, 배포하는 과정 전체에 대해 자동화 도구를 도입하거나 추가할 필요가 있습니까?	예	• G-클라우드에서 제공하지 않음(KT 내부 정책) - Jenkins 등 지원하지 않음 ※ 자체적으로 Jira, SVN, Git-hub, Git-Lab, Jenkins 사용
	기술 반영 용이성 확보	4-2. 오픈소스 SW를 비롯한 다양한 기술의 도입, 업그레이드, 업데이트 등 작업 수행 시 연관된 서비스에 대한 변경 작업이 복잡하여 개발 상의 어려움을 겪은 적이 있습니까?	아니오	• 어려움 없음

표준프레임워크 포털에 대한 클라우드 네이티브 적합성 검토 결과, 3개 항목에 대해 “예”로 응답하여 기존 아키텍처를 유지하도록 함

04 클라우드 네이티브 적합성 검토

4.4 클라우드 네이티브 적합성 검토 예시

4.4.2 공영홈쇼핑 영업시스템과 온라인몰

- 공영홈쇼핑의 정보시스템은 크게 영업시스템과 온라인몰 시스템으로 구성된다.
- 온라인몰은 EC(인터넷)과 MC(모바일) 영역으로 구분되고, 영업시스템은 주문, 고객, 방송, 상품 중 주요 업무영역별 서브 시스템이 구성된다.

[그림 4-20] 공영홈쇼핑 영업시스템 화면 및 주요 기능

영업시스템 EC 사이트 화면

영업시스템과 온라인몰 구성도

04 클라우드 네이티브 적합성 검토

4.4 클라우드 네이티브 적합성 검토 예시

4.4.2 공영홈쇼핑 영업시스템과 온라인몰

- 공영홈쇼핑 영업시스템과 온라인몰(EC/MC)을 대상으로 클라우드 네이티브 적합성 검토 결과, “예” 응답 수가 5개로 집계되어 클라우드 네이티브 도입이 필요한 것으로 보여진다.

[표 4-5] 공영홈쇼핑 영업시스템의 클라우드 네이티브 적합성 검토 결과

목표 구분	적합성 검토 항목	질의사항	답변	근거
정책 및 업무 변화 대응	정책 및 업무 변화에 대한 민첩한 대응	1-1. 한국판 뉴딜과 같은 새로운 정책, 기관별 업무계획 등 다양한 정보화 요구사항 변화에 대해 신속한 대응이 필요합니까?	예	• 매년 업무계획에 따른 정보화 요구사항 반영 필요
	디지털 혁신 및 지능화 지원	1-2. 디지털 혁신 및 지능 정보화 관련 신기술(빅데이터, AI, 블록체인, IoT 등)의 도입이 필요하고, 연관된 애플리케이션의 개발 또는 개선이 요구됩니까?	예	• 지속적 신기술 적용 필요
안정적 서비스 운영	서비스 개선 요구사항 적시 대응	2-1. 서비스 이용자의 잦은 CSR(Customer Service Request, 고객의 서비스 요구사항)에 대해 적시 대응이 필요합니까? 또는 초기 개발비의 약 15% 이상을 매년 추가개발 및 유지보수 비용으로 사용하고 있습니까?	예	• 홈쇼핑 특성 상 지속적 서비스 개선 필요 - 타 홈쇼핑과 유사
	접속 지연, 서비스 장애 문제 신속한 해결	2-2. 다양한 원인에 의한 장애 발생 시 장애복구(예.시스템 증설, 업그레이드 등)를 위해 서비스를 중단한 적이 있습니까? 또는 이용자의 폭증에 의한 접속지연으로 이용자의 불만이 제기된 적이 있습니까?	아니오	• 가용성 99.5% 이상
	소규모 서비스 분리 및 독립적 운영	2-3. 소규모 서비스 단위로 기능과 DB가 명확하게 분리되고, 독립적으로 서비스를 실행할 수 있습니까? (공통 기능 및 데이터 사용 유무, 타 시스템과의 연계성, 서비스 의존 관계 등 확인)	예	• 서비스별 기능 및 DB 분리 가능
	다양한 플랫폼 환경에서의 이식성 보장	2-4. 개발-테스트-운영 환경에서 안정적이고 지속적인 배포가 필요하거나 다양한 플랫폼 환경에 애플리케이션 배포가 요구됩니까?	아니오	• 개발-테스트-운영 환경에 배포하지만 타 플랫폼 이식은 없음
개발 품질 향상	개발·운영 협업 조직체계 구현	3-1. 시스템 개발 및 운영 시 개발팀과 운영팀의 분리에 의한 의사소통의 문제, 개발 및 배포 지연 등의 문제가 존재합니까?	아니오	• 정기/수시 배포 중, 절차적으로 문제 없도록 처리 - 단계별 승인 후 진행
	전문인력 역량 강화	3-2. 개발-빌드-배포 과정의 품질 향상을 위해 전문 인력을 확보하거나 기존 인력에 대한 전문적인 교육이 필요합니까?	아니오	• 유지보수팀 운영 중
개발 생산성 향상	코딩-빌드-테스트-배포 파이프라인 자동화	4-1. 개발된 SW를 형상관리 시스템에 커밋한 후 개발/검증/운영 서버에서 각각 빌드, 테스트, 배포하는 과정 전체에 대해 자동화 도구를 도입하거나 추가할 필요가 있습니까?	아니요	• 수시 상품 변경에 따른 신속한 배포 필요 - 빌드, 배포 도구 사용
	기술 반영 용이성 확보	4-2. 오픈소스 SW를 비롯한 다양한 기술의 도입, 업그레이드, 업데이트 등 작업 수행 시 연관된 서비스에 대한 변경 작업이 복잡하여 개발 상의 어려움을 겪은 적이 있습니까?	예	• 상용 SW 사용으로 수평 확장의 즉시 대응이 어려움

공영홈쇼핑의 영업/온라인몰 시스템에 대한 클라우드 네이티브 적합성 검토 결과, 5개 항목에 “예”로 응답
→ 클라우드 네이티브 도입 필요(컨테이너, 데브옵스 및 CI/CD , MSA 도입 검토)

04 클라우드 네이티브 적합성 검토

4.4 클라우드 네이티브 적합성 검토 예시

4.4.3 한국부동산원 감정평가 및 공시가격 정보체계

- 한국부동산원은 감정평가정보체계 시스템과 공시가격정보체계 시스템을 보유하고 있으며, 감정평가정보체계는 감정평가사 등 제한된 허가자가 사용하는 시스템이고, 공시가격정보체계는 대국민용 정보 제공 시스템이다.
- 감평가정보체계는 40개, 공시가격정보체계는 16개의 애플리케이션 기능으로 구성된다.

[그림 4-21] 한국부동산원 감정평가 및 공시가격 정보체계 화면 및 주요 기능

공시가격체계 사이트

가격평가 및 공시가격 정보체계 구성

04 클라우드 네이티브 적합성 검토

4.4 클라우드 네이티브 적합성 검토 예시

4.4.3 한국부동산원 감정평가 및 공시가격 정보체계

- 한국부동산원 감정평가 및 공시가격 정보체계 대상으로 클라우드 네이티브 적합성 검토 결과, “예” 응답 수가 6개로 집계되어 클라우드 네이티브 도입이 필요한 것으로 보여진다.

[표 4-6] 한국부동산원 감정평가 및 공시가격 정보체계 네이티브 적합성 검토 결과

목표 구분	적합성 검토 항목	질의사항	답변	근거
정책 및 업무 변화 대응	정책 및 업무 변화에 대한 민첩한 대응	1-1. 한국판 뉴딜과 같은 새로운 정책, 기관별 업무계획 등 다양한 정보화 요구사항 변화에 대해 신속한 대응이 필요합니까?	아니오	• 다양한 변화 요구 적음
	디지털 혁신 및 지능화 지원	1-2. 디지털 혁신 및 지능 정보화 관련 신기술(빅데이터, AI, 블록체인, IoT 등)의 도입이 필요하고, 연관된 애플리케이션의 개발 또는 개선이 요구됩니까?	예	• 빅데이터, AI 등 신기술 요구 존재
안정적 서비스 운영	서비스 개선 요구사항 적시 대응	2-1. 서비스 이용자의 잦은 CSR(Customer Service Request, 고객의 서비스 요구사항)에 대해 적시 대응이 필요합니까? 또는 초기 개발비의 약 15% 이상을 매년 추가개발 및 유지보수 비용으로 사용하고 있습니까?	예	• 지속적 유지 보수 필요
	접속 지연, 서비스 장애 문제 신속한 해결	2-2. 다양한 원인에 의한 장애 발생 시 장애복구(예.시스템 증설, 업그레이드 등)를 위해 서비스를 중단한 적이 있습니까? 또는 이용자의 폭증에 의한 접속지연으로 이용자의 불만이 제기된 적이 있습니까?	예	• 공동주택가격열람 시기 사용자 증가에 따른 시스템 가용성 확보 필요
	소규모 서비스 분리 및 독립적 운영	2-3. 소규모 서비스 단위로 기능과 DB가 명확하게 분리되고, 독립적으로 서비스를 실행할 수 있습니까? (공통 기능 및 데이터 사용 유무, 타 시스템과의 연계성, 서비스 의존 관계 등 확인)	예	• 감정평가, 공시가격, 토지 등으로 서비스가 명확하게 분리됨 • 기존 C/S 프로그램이 객체지향으로 개발되어 있음
	다양한 플랫폼 환경에서의 이식성 보장	2-4. 개발-테스트-운영 환경에서 안정적이고 지속적인 배포가 필요하거나 다양한 플랫폼 환경에 애플리케이션 배포가 요구됩니까?	예	• 개발-테스트-운영환경의 구성 및 일치 필요
개발 품질 향상	개발·운영 협업 조직체계 구현	3-1. 시스템 개발 및 운영 시 개발팀과 운영팀의 분리에 의한 의사소통의 문제, 개발 및 배포 지연 등의 문제가 존재합니까?	아니오	• 이슈 없음
	전문인력 역량 강화	3-2. 개발-빌드-배포 과정의 품질 향상을 위해 전문 인력을 확보하거나 기존 인력에 대한 전문적인 교육이 필요합니까?	아니오	• 이슈 없음
개발 생산성 향상	코딩-빌드-테스트-배포 파이프라인 자동화	4-1. 개발된 SW를 형상관리 시스템에 커밋한 후 개발/검증/운영 서버에서 각각 빌드, 테스트, 배포하는 과정 전체에 대해 자동화 도구를 도입하거나 추가할 필요가 있습니까?	예	• 수시 배포가 많으므로 전반적인 배포주기 단축 필요
	기술 반영 용이성 확보	4-2. 오픈소스 SW를 비롯한 다양한 기술의 도입, 업그레이드, 업데이트 등 작업 수행 시 연관된 서비스에 대한 변경 작업이 복잡하여 개발 상의 어려움을 겪은 적이 있습니까?	아니오	• 해당 없음

한국부동산원의 감정평가정보체계, 공시가격정보체계 시스템에 대한 클라우드 네이티브 적합성 검토 결과,
6개 항목에 “예”로 응답 → 클라우드 네이티브 도입 필요 (컨테이너, 데브옵스 및 CI/CD)

클라우드 네이티브 정보시스템 구축을 위한

발주자 안내서

05

클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.1 사업관리 단계별 고려사항

5.1.1 정보화 사업관리 프로세스

- 공공기관에서 추진하는 정보화 사업관리 프로세스는 기획·검토, 계획 수립, 사업자 선정·계약, 개발·구축, 감리, 검사 종료 단계로 진행된다.
- 클라우드 네이티브 관련 정보화사업 추진 시 발주 담당자는 계획 수립과 사업자 선정·계약 작업 시 사업 추진을 위해 많은 준비가 필요하다.

[그림 5-1] 정보화 사업관리 단계

클라우드 네이티브 추진 관련 사항 검토 대상

[출처:행정안전부 & NIA, 행정기관을 위한 정보화사업 단계별 관리점검 가이드 V2.0]

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.1 사업관리 단계별 고려사항

5.1.2 클라우드 네이티브 관련 발주자 업무

- 클라우드 네이티브 정보화사업 추진 시 사업관리 단계별로 발주자가 수행해야 할 주요 업무는 다음과 같다.

[표 5-1] 사업관리 단계별 클라우드 네이티브 관련 발주자의 업무

사업관리 단계			관리·점검 수행내용	클라우드 네이티브 관련 발주자 업무
기획·검토	기획·검토	정보화사업 기획	• 국가정보화 기본계획에 따른 정보화사업 추진 계획 수립 • 국가정보화전략위원회의 심의와 기획재정부의 예산확보를 통한 정보화 시행계획 확정	• 특이사항 없음
계획 수립	사업계획서(안) 작성	사업계획서(안) 작성	• 정보화사업 추진을 위한 요구사항 상세화 • 요구사항 실현을 위한 필요 기술 검토, 소요 자원 및 예산 수립 등	• 클라우드 네이티브 관련 요구사항 상세화 • 클라우드 네이티브 도입을 고려한 개발비 예산 산정
		기술적용계획 수립	• 정보시스템 구축 사업 또는 정보시스템 운영 및 유지보수 사업 추진 시 기술적용계획 수립	• 클라우드 네이티브 애플리케이션 관련 기술적용계획 수립
	사업계획서(안) 검토	기술평가 시행	• 정보시스템 상호 운용, 공동 활용, 편의성 및 효율성 측면의 사업계획검토 • 사업계획서 작성 시 상호운용성 확보를 위한 기술 평가 수행 및 제안요청서 반영	• 기술평가 수행 시 클라우드 네이티브 관련 상호운용성 확보 방안 작성
사업자 선정·계약	제안 요청	제안요청서 작성	• 사업계획서를 토대로 발주기관의 요구사항을 포함하여 제안요청서 작성 • 제안요청서에 요구사항, 소요예산, 사업기간, 제안서 평가기준, 계약조건 등 기술	• 클라우드 네이티브 관련 요구사항 상세화 • 클라우드 네이티브 도입을 고려한 개발비 예산 산정
	사업자 선정 및 계약 체결	제안서 평가	• 사업자가 제출한 제안서에 대한 평가 실시 - 기술평가점수와 가격평가를 통해 종합평가점수 산출 - 일반적인 기술평가와 가격평가 비율은 각각 80% : 20%의 비중	• 기술평가 수행 시 클라우드 네이티브 적용 경험 및 기술력을 평가할 수 있는 항목 정의 및 배점 부여
		협상 및 낙찰자 선정	• 종합평가점수가 높은 우선협상대상자 선정 및 기술협상 수행 - 제안한 사업내용, 이행방법, 이행일정, 제안가격 등을 대상으로 기술 및 가격 협상 실시	• 기술협상 시 클라우드 네이티브 관련 기술적 이행사항에 대해 구체적으로 협상 수행

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.1 사업관리 단계별 고려사항

5.1.2 클라우드 네이티브 관련 발주자 업무

[표 5-1] 사업관리 단계별 클라우드 네이티브 관련 발주자의 업무

사업관리 단계			관리·점검 수행 내용	클라우드 네이티브 관련 발주자 업무
개발·구축	사업착수	착수계획서 검토	• 사업자는 계약문서에서 따라 사업 착수 • 사업자는 착수계획서를 계약서에 명시된 기간 내에 발주기관에 제출	• 착수계획서 내에 클라우드 네이티브 관련 과업 내용이 누락 없이 잘 반영되어 있는지 확인
	진도/품질관리	진도/품질관리	• 사업자는 주간보고와 월간보고 등 진도 관리 및 보고 활동 수행 • 단계별 품질 검토회를 통한 사업 품질 목표의 달성 여부를 확인	• 방법론의 공정단계별 품질관리 활동 수행 - 개발방법론에 따른 공정단계별 품질 관리 활동 수행 - 클라우드 네이티브 관련 요구사항별 이행 과정 점검
감리	감리시행	감리시행	• 발주기관은 감리시행 동안 사업자로 하여금 감리에 협조하도록 지시 • 감리수행결과보고서(안)에 대해서는 감리계획서에 명시된 점검항목 등이 전부 점검되었는지 확인	• 감리계획서 및 수행결과보고서 검토 및 조치 - 감리계획서 및 수행결과보고서 내에 클라우드 네이티브 관련 요구사항에 대한 점검계획과 결과 검토 후 조치 요구
검사종료	완료검사	완료검사	• 발주기관은 용역을 완성하거나 계약이행을 완료한 경우 검사 요청 • 발주기관의 검사 수행 및 검사 완료 여부 통지	• 완료검사 수행 시 클라우드 네이티브 관련 요구사항의 이행 여부 확인

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.2 사업계획서 및 제안요청서 작성 시 고려사항

5.2.1 사업계획서 및 제안요청서 작성 개요

- 사업계획서와 제안요청서 작성 시 클라우드 네이티브 관련 사항이 포함되어야 할 목차는 사업 범위, 문제점 및 개선 방향, 목표시스템 개념도, 개발 대상 업무, 상세 요구사항, 소요예산 등이다.
- 사업계획서(안) 작성 시 기본적인 방향성과 요구사항을 정의한 후 제안요청서 작성 시에 상세 요구사항을 구체적으로 작성한다.

[표 5-2] 사업계획서(안) 작성

대목차	소목차
1. 사업 개요	가. 추진 배경 및 필요성 나. 서비스 내용 다. 사업 범위 라. 기대효과
2. 현황 및 문제점	가. 업무 현황 나. 정보화 현황 다. 선진사례 라. 문제점 및 개선 방향
3. 사업 추진방안	가. 추진 목표 나. 추진 전략 다. 추진 체계 라. 추진 일정 마. 추진 방안
4. 사업내용	가. 개요 나. 용어의 정의 다. 목표시스템 개념도 라. 개발 대상 업무 마. 상세 요구사항
5. 소요자원 및 예산	가. 소요예산 요약 나. 세부 산출 근거
6. 법·제도 정비 및 운영·발전계획	가. 법제도 정비 계획 나. 시스템 운영 방안 다. 시스템 발전계획

[출처 : 2021년 전자정부지원사업 사업계획서 작성방법 안내서]

[표 5-3] 제안요청서 작성

대목차	소목차
1. 사업 개요	가. 추진 배경 및 필요성 나. 서비스 내용 다. 사업 범위 라. 기대효과
2. 현황 및 문제점	가. 업무 현황 나. 정보화 현황 다. 선진사례 라. 문제점 및 개선 방향
3. 사업 추진방안	가. 추진 목표 나. 추진 전략 다. 추진 체계 라. 추진 일정 마. 추진 방안
4. 사업내용	가. 개요 나. 용어의 정의 다. 목표시스템 개념도 라. 개발 대상 업무 마. 상세 요구사항
5. 제안서 작성요령	가. 제안서의 효력 나. 제안서 작성 지침 및 유의사항 다. 제안서 목차 라. 세부 작성 지침
6. 제안안내 사항	가. 입찰 방식 나. 제안서 평가방법 다. 기술성 평가기준(이하 생략)

[출처 : 공공 SW사업 제안요청서 작성을 위한 요구사항 상세화 실무 가이드]

클라우드 네이티브 관련 사항 반영

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.2 사업계획서 및 제안요청서 작성 시 고려사항

5.2.2 사업 추진 방향성 및 사업 범위 작성

- 사업 추진 방향성과 사업 범위 작성 시 MSA, 컨테이너, 데브옵스 및 CI/CD 구성요소 관련 내용을 포함하여 클라우드 네이티브 사업임을 명시한다.

[그림 5-2] 사업 추진 방향성 작성 예시

MSA, 컨테이너 구성요소를 포함하여 사업 추진 방향성 작성

○ MSA 기반의 컨테이너 형태로 구현된 공간정보 서비스 기능(공간정보 표준 프레임워크)를 효율적으로 운영·관리하기 위한 개방형 공간정보 플랫폼 구축
- 서비스 수요 증감에 따라 유연하게 컨테이너가 확장 및 축소가 가능한 운영관리 기능 및 컨테이너 동작 여부에 대한 상태 모니터링 기능 제공

예시

[출처 : 디지털 관광자원 통합관리시스템 재구축 및 운영 제안요청서, 한국관광공사]

[그림 5-3] 사업 범위 작성 예시

MSA, 컨테이너, CI/CD 구성요소가 포함된 사업 범위 작성

○ 다중화 기반 마이크로 서비스 구축
- (마이크로 서비스 아키텍처 구축) 컨테이너 관리 기능, API 게이트웨이 관리 기능
- (전자정부 표준프레임워크 적용) 실행환경 구성, 개발환경 구성, 운영환경 구성, 관리환경 구성
- (인프라 구축) 인프라 가상화·자동화 구현, HW·SW 구축, 보안관리

○ (마이크로서비스) 잘 정의된 API를 통해 콘텐츠를 제공하는 콘텐츠 관리 기능 중심으로 시스템 구축

○ 마이크로서비스를 독립적으로 개발/배포/관리할 수 있는 프레임워크 제공

* 컨테이너 관리 : 여러 대의 서버에서 여러 개의 컨테이너를 편리하게 관리하도록 서비스 메시, 컨테이너 오케스트레이션 등 자동화 기반의 컨테이너 배포 구현

○ 마이크로서비스 구현을 위한 가상화/자동화 환경을 제공

* 가상화 : 물리적/논리적 서버 클러스터 구성을 통해 시스템 가용성 향상 및 가상 서버 복제 및 수평 확장을 통해 시스템 확장성 확보
* 자동화 : 서비스 요청관리, 수요관리, 변경관리 등 사용자의 서비스 요청에 대한 해당 서비스를 사용자에게 제공하기 위해 다음과 같은 자동화된 서비스 제공관리 환경 구축

예시

[출처 : 디지털 관광자원 통합관리시스템 재구축 및 운영 제안요청서, 한국관광공사]

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.2 사업계획서 및 제안요청서 작성 시 고려사항

5.2.3 상세 요구사항 작성

- 상세 요구사항 작성 시 정보화사업의 목적, 방향성, 구체적인 범위, 구현 방식 등을 고려하여 클라우드 서비스 전반, 컨테이너, MSA 기반 개발, 데브옵스 & CI/CD 관련 상세 요구사항을 작성한다.

[그림 5-4] 상세 요구사항 작성 예시

요구사항 분류		클라우드 서비스 요구사항
요구사항 고유번호		CSR-001
요구사항 명칭		클라우드 서비스 공통
요구사항 상세 설명	세부 내용	• 개발에 필요한 PaaS 기반의 클라우드 서비스 제공 • 개발에 필요한 서버, 스토리지, 네트워크, 보안, 인증 등 제반 클라우드 서비스 제공 • 가상화 및 클라우드 환경 구성 - 각 서버별 VM 생성 및 OS, 시스템 SW 설치 지원 - 클라우드 자원의 자동 생성, 관리 및 모니터링 기능 제공

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]

요구사항 분류		클라우드 서비스 요구사항
요구사항 고유번호		CSR-002
요구사항 명칭		컨테이너 기반 배포 기능 구현
요구사항 상세 설명	세부 내용	• 컨테이너화된 애플리케이션을 쉽고 안전하게 배포하기 위해 자동화 기반의 컨테이너 배포 구현 - 이미지 리파지토리 생성, 컨테이너별 이미지 지정 및 리소스 할당, 로드밸런서 생성 및 삭제, 포트 바인딩을 통해 서비스 공개 - 인스턴스 등록 및 삭제, 클러스터 구성 및 보안 설정, 컨테이너 테스트 및 실행 상태 확인, 컨테이너 이미지 저장 및 배포, 자동화된 원격 배포 관리 등 • 컨테이너 이미지를 쉽고 안전하게 저장/관리/배포하기 위해 컨테이너 레지스트리 제공 - HTTPS를 통해 암호화되어 전송 - 네임스페이스를 사용하여 컨테이너 레지스트리(Registry, 이미지 저장소) 정의 - 컨테이너 이미지의 목록 조회 및 메타데이터 관리 - 불필요한 이미지 삭제 등

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]

5.2 사업계획서 및 제안요청서 작성 시 고려사항

5.2.3 상세 요구사항 작성

[그림 5-5] 상세 요구사항 작성 예시

요구사항 분류		클라우드 서비스 요구사항
요구사항 고유번호		CSR-003
요구사항 명칭		컨테이너 기반 서비스 메시 및 오케스트레이션 구현
요구사항 상세 설명	세부 내용	• 여러 대의 서버에서 여러 개의 컨테이너를 편리하게 관리하도록 서비스 메시 기능 제공 - 컨테이너를 적절한 서버에 배포하고 상태를 유지하기 위한 스케줄링 - 여러 대의 서버를 1대의 서버처럼 관리하고, 가상 네트워크를 이용해 접근하기 위한 클러스터링 - 컨테이너의 IP/포트정보를 서비스 레지스트리에 저장하며, 동적으로 변화하는 리소스의 위치를 API 게이트웨이가 검색하기 위한 서비스 디스커버리 기능 제공 - API 요청에 대한 최적의 경로를 지원하기 위한 다양한 API 라우팅 구현 - 서비스 간 부하 분산을 위한 로드밸런싱 - 오토스케일링 시 서버 수 지정, 서버의 사양 정의, 서버 실행 시까지의 워밍업 시간 지정 등 트래픽 집중에 서버, 스토리지, 네트워크 등 인프라 자원의 자동 확장 및 축소를 자동화하여 서비스 상태에 따른 적정 서버 유지를 통해 유연한 서비스를 제공하도록 오토스케일링 지원 - 특정 서비스에 오류가 발생하거나 실행 실패 시 신속하게 이전 버전으로 되돌아가도록 복구(Rollback) 기능 지원 - 표준화된 로그 이벤트 수집 및 분석, 서비스 간 호출 추적 및 성능 관리 등 로깅 및 로그 분석 등

예시

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]

요구사항 분류		클라우드 서비스 요구사항
요구사항 고유번호		CSR-004
요구사항 명칭		API 게이트웨이 관리 기능
요구사항 상세 설명	세부 내용	• API 호출을 위한 토큰 발급 및 인증, 엔드포인트별 API 호출 인증 및 인가, 접근 정책(예, 특정 클라이언트의 API 호출 불허에 의한 접근제어 기능) 등 API 인증 및 인가 처리 • 동일 API를 클라이언트나 마이크로서비스에 따라 다른 엔드포인트를 통해 서비스를 제공하도록 API 라우팅을 지원하고, 동일 API를 여러 개의 클라이언트/마이크로서비스별로 엔드포인트 제공 • 로깅, 인증 등 공통 기능을 중복 개발 또는 처리하지 않도록 요청과 응답의 표준화 및 공통 로직 처리 • 동일 API를 HTTP, REST, XML 웹 서비스 등 클라이언트와 마이크로서비스별로 상이한 프로토콜로 서비스하기 위한 프로토콜 변환 처리 • 동기, 비동기 등 API를 호출하는 메시지 패턴을 변경할 수 있도록 메시지 호출 패턴 변환 기능 제공 • 호출 횟수, 전송 용량, 네트워크 대역폭 등 서비스 레벨을 클라이언트나 마이크로서비스별로 조정할 수 있도록 QoS(Quality of Service) 설정 기능 제공 • API 호출 패턴 분석, API 호출 실행 및 접근 상태 분석, 요청 IP/클라이언트/일시 등 API 호출에 대한 로깅 및 모니터링 등

예시

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.2 사업계획서 및 제안요청서 작성 시 고려사항

5.2.3 상세 요구사항 작성

[그림 5-6] 상세 요구사항 작성 예시

요구사항 분류		기능 요구사항
요구사항 고유번호		SFR-001
요구사항 명칭		개발 공통 사항
요구사항 상세 설명	세부 내용	• 클라우드 플랫폼은 사용자가 플랫폼 서비스를 조합·활용하여 유연하게 시스템 환경을 구성할 수 있도록 지원하여야 함 • 애플리케이션 실행환경 자동구성 및 라이프사이클 관리 기능을 제공하여야 하며, 모든 응용 소프트웨어는 이를 기반으로 관리되어야 함 • 응용 소프트웨어 개발 및 배포, 운영은 과제 개발 주기를 단축하고, 운영중단을 최소화하기 위하여 클라우드 플랫폼에서 제공하는 데브옵스 도구(CI/CD 도구) 등을 이용하여 관리되어야 함 ※ 세부 범위는 추진상황에 따라 발주기관과 협의하여 변경 가능

예시

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]

요구사항 분류		클라우드 서비스 요구사항
요구사항 고유번호		SFR-002
요구사항 명칭		클라우드 플랫폼 기능 구현
요구사항 상세 설명	세부 내용	• 클라우드 리소스 관리 기능 - 서버, 스토리지, 네트워크 관리 - 리소스 모니터링 - 확장 규칙에 따른 전산 자원의 확장 및 축소(오토스케일링) • CI/CD 파이프라인 기능 - 지속적 통합과 배포를 지원하는 파이프라인 기능 제공 - 파이프라인의 실행은 트리거에 의한 자동, 배포자에 의한 수동, 스케줄러에 의한 배치 형태 전체 지원 - 배포에 대한 이력관리 및 롤백 지원 - 파이프라인 흐름에 대한 가시성과 리포트 제공 - 수행된 작업에 대한 가시성과 감사 추적 제공 • 컨테이너 배포 기능 - 애플리케이션 파일 배포 시 실행환경의 자동 생성·구동 지원 - 배포된 애플리케이션의 전체 생명주기 관리 및 스케일링 가능 • 빌드 기능 - 설정 기반의 정형화된 빌드 시스템 제공 - 종속성 관리 지원 - 테스트 자동화 프레임워크 지원 - 테스트 결과 리포트 제공

예시

[출처: 나라장터, 클라우드 네이티브 관련 제안요청서 참조]

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.3 개발비 산정 관련 고려사항

5.3.1 개발비 구성요소

- 정보시스템 구축을 위한 총 구축비는 시스템 구축비와 부대비로 구성된다. 시스템 구축비는 HW 구매비, 시스템 운용환경 구축비, SW 구축비, DB 구축비, 기존 시스템 이전비 등으로 구성된다.

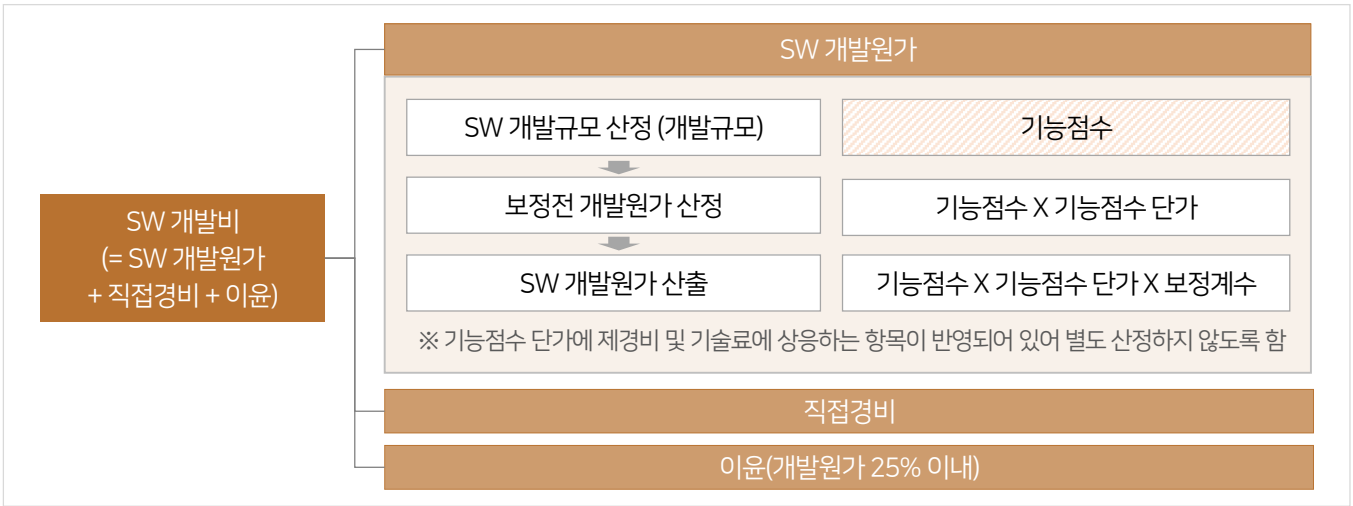
[표 5-4] 총 구축비 구성요소

대분류	중분류	소분류	관련 근거	비고
시스템 구축비	HW 구매비	HW 구매/설치비	정보시스템 하드웨어 규모산정 지침	장비비 (클라우드 이용 시 클라우드 서비스 비용으로 대체)
	시스템 운용환경 구축비	운용환경 설계비	엔지니어링 사업대가의 기준	
		운용환경 공사 (시설/통신망)		
	SW 구축비	상용 SW 구입비	정보시스템 하드웨어 규모산정 지침	
		SW 개발비	SW사업 대가산정 가이드	개발비
	DB 구축비	DB 설계비		
		데이터 제작		
	기존 시스템 이전비	HW 이전비		
		데이터 이전비		
부대비	감리비		정보시스템 감리 기준	기타

[출처: 기획재정부 & NIA, ISP 수립 공통 가이드, 2021]

- 클라우드 네이티브 정보시스템 개발비는 SW 개발비에 해당되며, SW 개발비는 크게 소프트웨어 개발원가, 직접경비, 이윤의 세 부분으로 구성되며 기능점수 기반으로 산출된다.

[그림 5-7] SW 개발비 구성요소



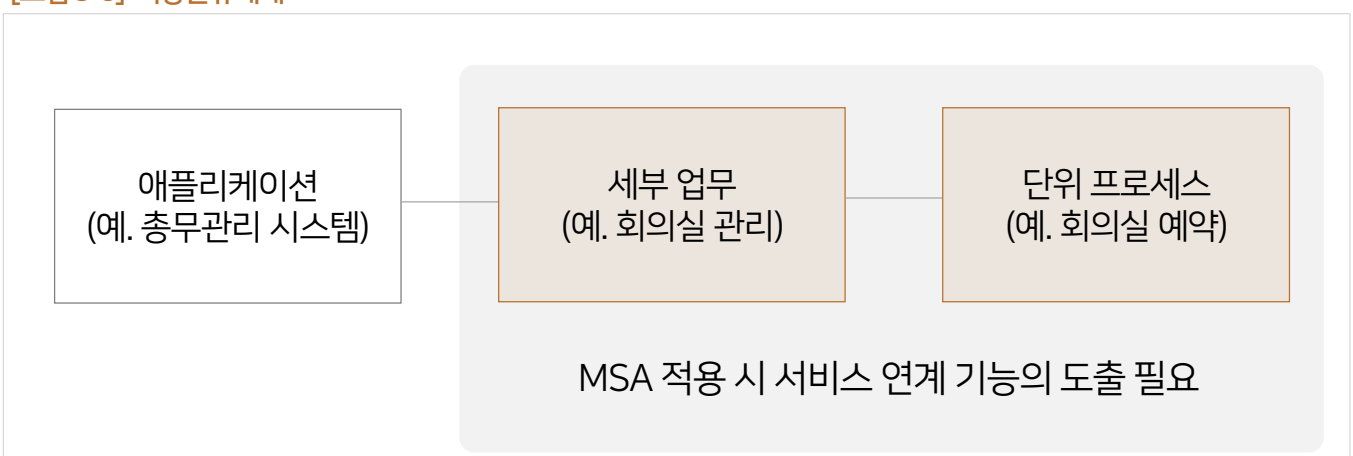
05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.3 개발비 산정 관련 고려사항

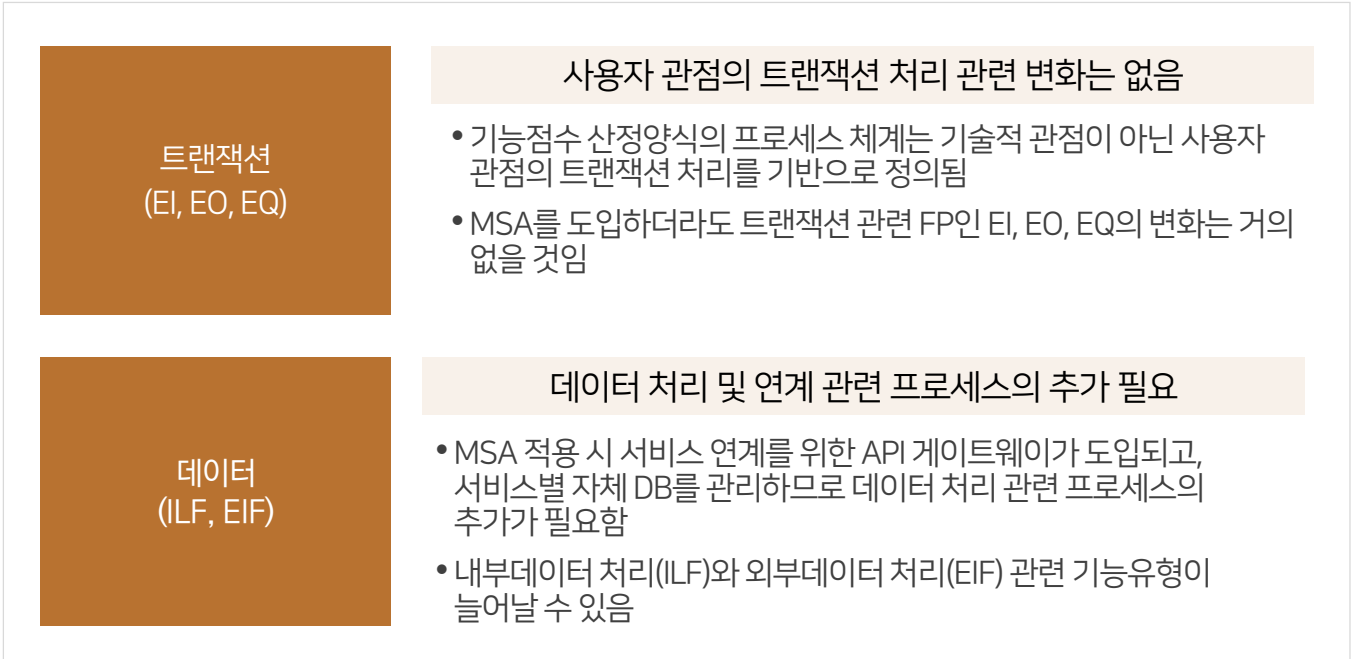
5.3.2 기능식별 방식의 변화

- 클라우드 네이티브 정보시스템 구축 시 MSA를 적용할 경우 서비스 연계를 위한 기능을 식별하도록 한다. 기능점수(FP, Function Point) 산정 방식은 기존과 동일하다.
- 마이크로서비스는 작은 서비스 단위로 자체 DB를 구성하고, API 게이트웨이를 통해 서비스 간 통신이 이뤄진다. 기능점수 산정 시 DB 관련 데이터 처리 기능과 서비스 간 연계 기능의 도출이 필요하다.

[그림 5-8] 기능분류체계



[그림 5-9] MSA 적용 시 기능식별 방식의 변화



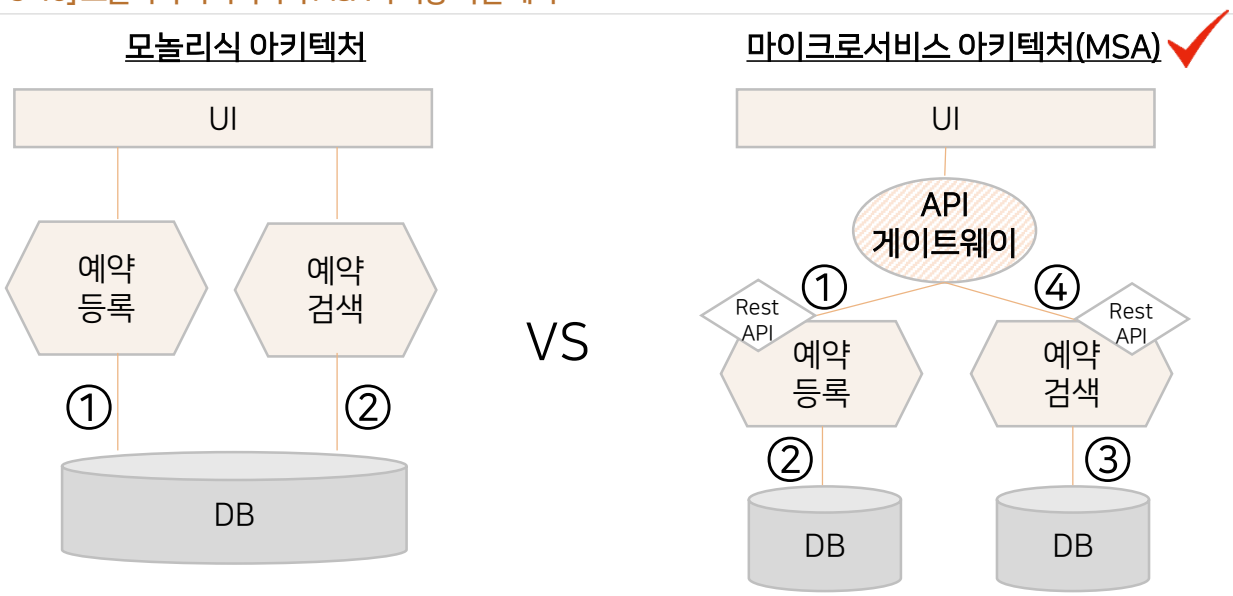
05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.3 개발비 산정 관련 고려사항

5.3.2 기능식별 방식의 변화

- MSA 도입 시 기존의 모놀리식 아키텍처 대비 서비스 연계와 DB 처리 관련 기능점수의 증가로 개발비 증가가 예상된다.

[그림 5-10] 모놀리식 아키텍처와 MSA의 기능 식별 예시



단위 프로세스	FP 유형
① 예약 등록	EI
② 예약 검색	EQ

단위 프로세스	FP 유형
① 예약 등록 API 연계	ILF
② 예약 등록	EI
③ 예약 검색	EQ
④ 예약 검색 API 연계	ILF 또는 EIF

- MSA는 API 게이트웨이를 통해 서비스 연계 작업을 수행하므로 이와 관련된 프로세스를 추가함
  - 예약 등록 서비스의 경우, 예약등록 API 연계를 내부논리파일(ILF)로 추가함
  - 예약 검색 서비스의 경우, 예약검색 API 연계를 내부논리파일(ILF)로 추가함

  ※ 예약검색 DB가 없는 경우, 예약등록 DB를 읽어야 하므로 외부연계파일(EIF)로 정의함

- MSA 적용 시 모놀리식 아키텍처 대비 API 연계 프로세스의 추가에 따라 기능점수가 증가하고, 개발비 증가가 예상됨

05 클라우드 네이티브 정보시스템 구축 사업 추진 고려사항

5.4 데브옵스 조직 관련 고려사항

- 공공기관에서 데브옵스 조직 구성 시 소통과 협업 관련 조직 문화, 조직 규모, 개발자 인원 및 역량 등을 고려하도록 한다.
- 공공기관의 현업 사용자가 정보화사업에 적극적으로 참여하기 어려울 수 있으므로, 해당 업무와 개발·운영 환경에 대해 경험이 풍부한 인력으로 데브옵스 조직을 구성하도록 한다.

[그림 5-11] 공공기관의 데브옵스 조직 구성 시 고려사항

인위적으로 데브옵스팀 만들지 않기	• 개발과 운영이 협업할 수 있는 체계와 소통 방안 마련 필요 • 인위적으로 개발과 운영을 모두 수행할 수 있는 데브옵스팀을 만들게 되면 오히려 추가적인 부담이 증가할 수 있음 • 데브옵스팀을 만드는 이유는 개발과 운영이 상호 협업하기 위해서임
조직의 규모를 고려한 데브옵스 조직 구성	• 소규모 공공기관의 기관의 경우, 이미 개발과 운영이 합쳐진 데브옵스 조직 운영이 이뤄지고 있으므로 기존 체계 유지 - 개발 및 운영 인력이 6-7인 이하로 구성된 전산실의 경우, 이미 데브옵스 조직의 형태로 업무를 수행하고 있음 - 1-2인이 개발과 운영 업무를 수행하는 기관도 상당수 있음 • 대규모 공공기관의 경우 개발과 운영의 단절에 따른 업무 효율성 저하 문제가 발생한다면 중장기적인 관점에서 데브옵스 조직 구성 검토 - 서비스 단위별로 약 6-7명 정도의 개발과 운영 인력을 1개의 데브옵스팀으로 구성하여 개발과 배포, 운영 업무를 수행하도록 함
개발자의 역량을 고려한 개발과 운영 업무 배정	• 개발팀과 운영팀을 하나의 팀으로 만들 때 개발자의 업무 수행 경험을 토대로 담당 업무를 배정해야 함 - 개발 경험자는 개발 업무, 운영 경험자는 운영 업무, 개발과 운영이 가능한 개발자에게 개발과 운영 연계 업무를 배정하도록 함
무분별한 데브옵스 엔지니어 채용 지양	• 데브옵스 엔지니어 한 사람을 채용한다고 조직문화가 바뀌지 않음 • 외부의 데브옵스 엔지니어 채용을 통해 데브옵스 환경을 만들려고 하지 말고, 중장기적으로 조직 내부에서 협업하는 문화를 만들어야 함

클라우드 네이티브 정보시스템 구축을 위한

발주자 안내서

발행일

2021년 12월 30일 발행

발행처

한국지능정보사회진흥원

담당팀

디지털정부기반지원팀