NIA 한국지능정보사회진흥원

클라우드 네이티브 기반 행정·공공 서비스 확산 지원

클라우드 네이티브 추진시 고려사항

2차 교육 교재

INDEX

클라우드 네이티브 기반 행정·공공 서비스 확산 지원

클라우드 네이티브 발주자 가이드

클라우드 네이티브는 어떻게 발전하고 있는가?

NIA 한국지능정보사회진흥원

클라우드 네이티브 정의

클라우드 네이티브란?

클라우드 네이티브
(형용사/명사)

클라우드 컴퓨팅의 장점을 최대한 활용할 수 있는
(효율적인 자원이용, 탄력적 수요 대응 등)
정보시스템 분석·설계·구현 및 **실행하는 환경**

클라우드 네이티브
애플리케이션

클라우드 환경에서 실행되는 **애플리케이션**

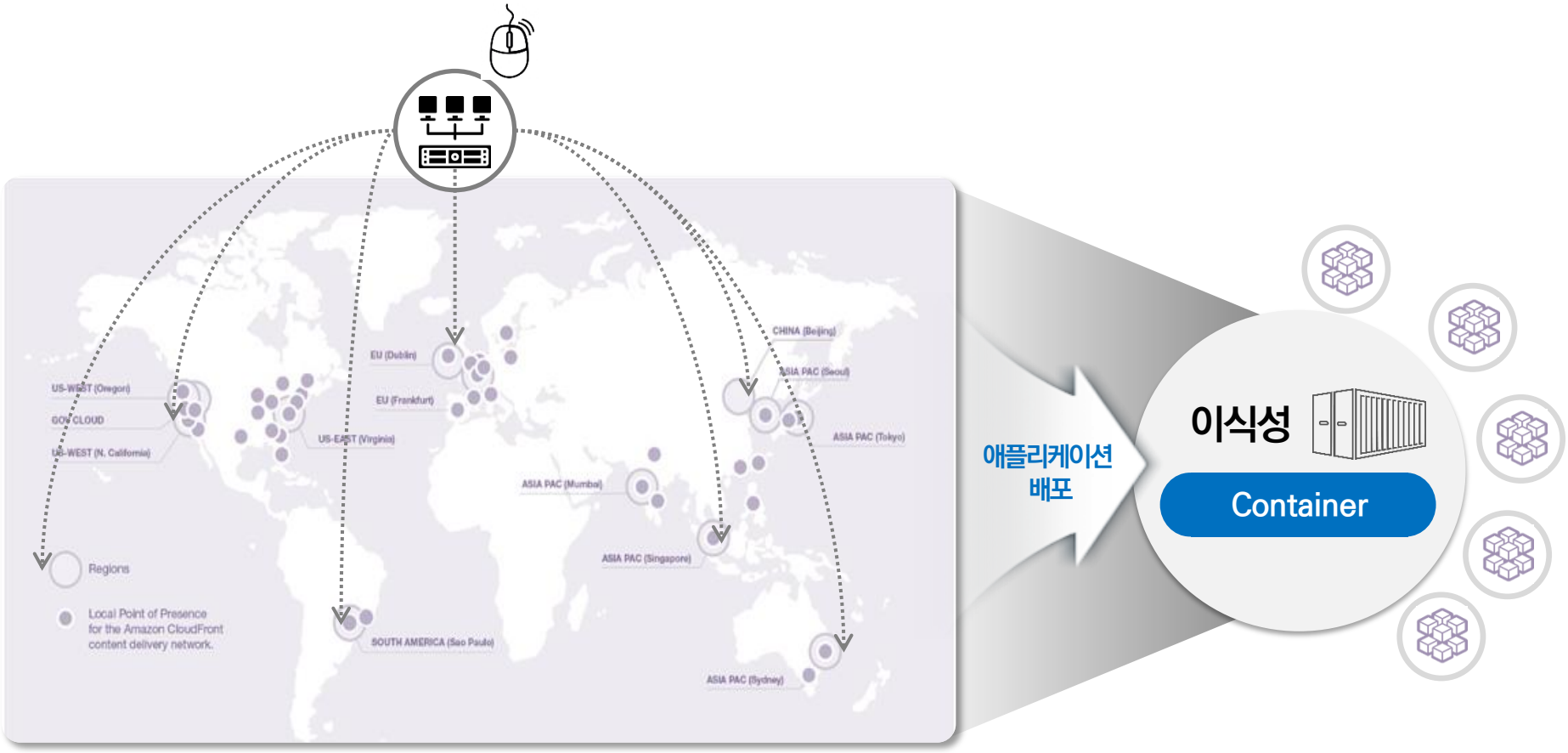
CNCF (Cloud Native Computing Foundation) v1.0
클라우드 네이티브 전환할 수 있는 기술 정의 및 오픈 소스를 관리하는 단체

- 퍼블릭, 프라이빗, 하이브리드 **클라우드 환경에서** 확장성 있는 **애플리케이션**
- 컨테이너, 서비스 메시(Mesh), **마이크로서비스(Micro Service) 인프라 구조**, 선언적 API로 접근
- 자동화, 회복성, 편리성, 가시성을 갖는 **느슨하게 결합된 시스템** (개발 및 실행 환경)
- 엔지니어는 최소한의 수고로, 영향력이 크고, 예측 가능한 변경을 할 수 있는 기술 정의

클라우드 네이티브 특징

클라우드 네이티브는 작고·가볍고 손쉽게 배포를 제공합니다.

클라우드 환경에서의 서비스 배포 → 전세계에 한번의 클릭으로 애플리케이션 배포 → 전 세계를 상대로 서비스 가능



"반대로 전 세계에서 국내 비즈니스 생태계에 대한 서비스 배포도 가능"

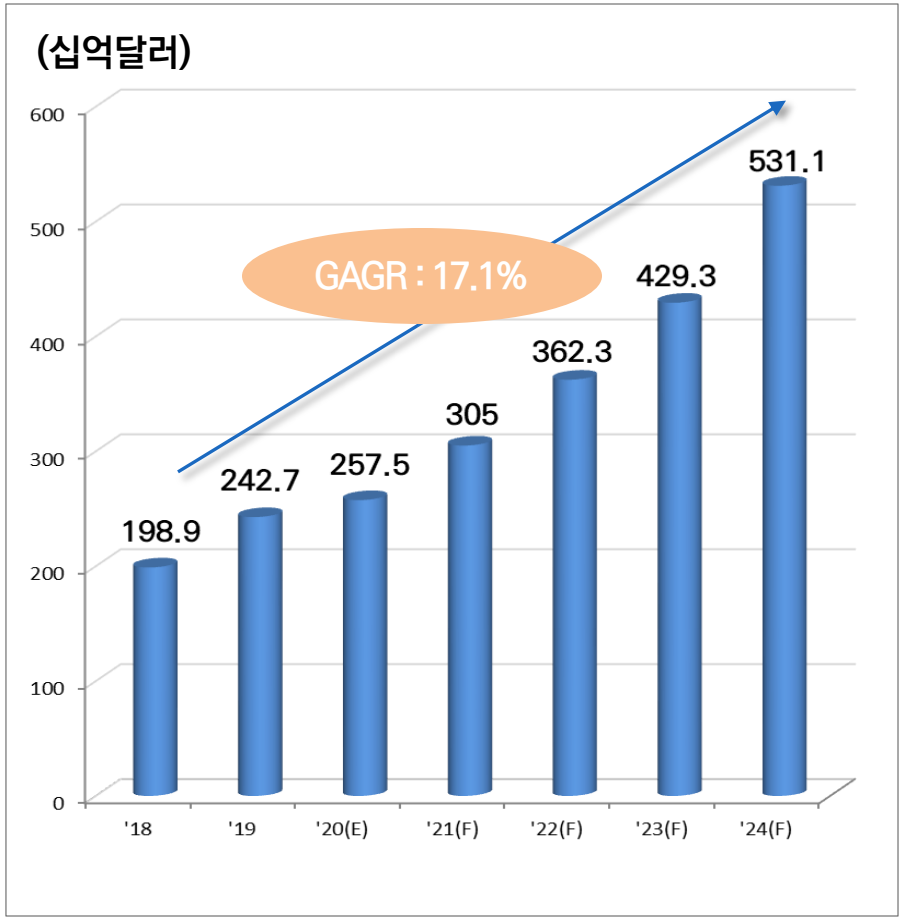
클라우드 네이티브 변화

클라우드 네이티브는 어떻게 발전하고 있는가?

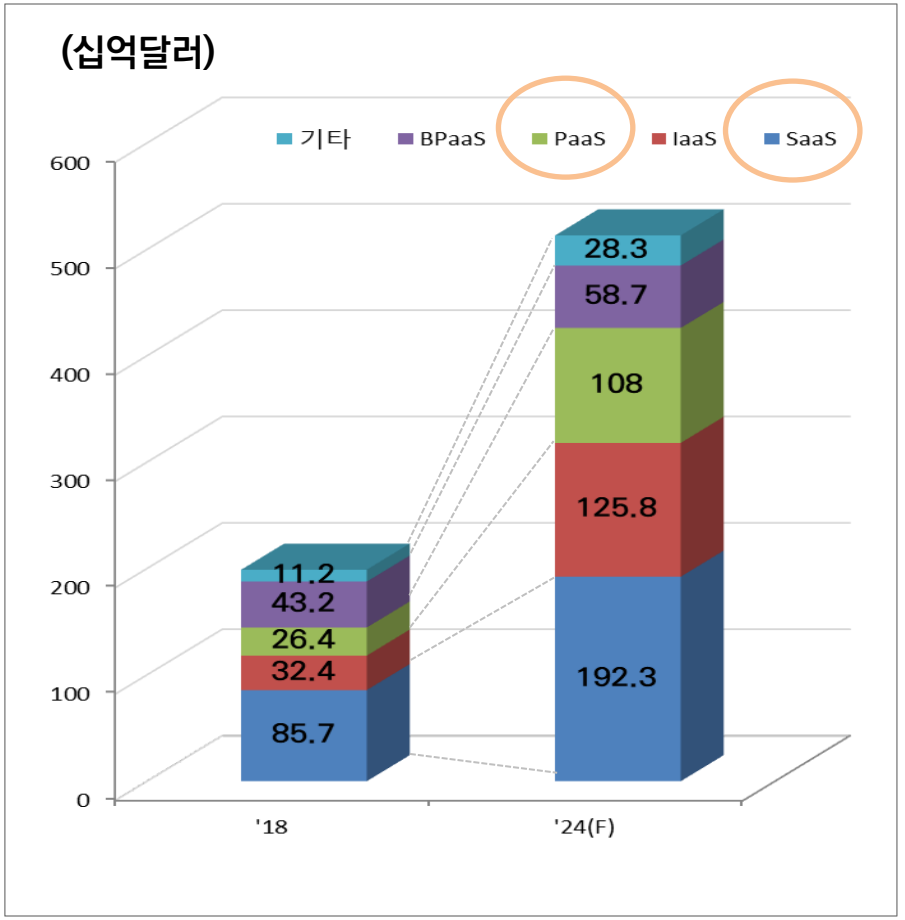
1단계 레거시	2단계 클라우드	3단계 클라우드 네이티브
물리(온프리미스)	IaaS	PaaS + 컨테이너
		MSA + 컨테이너
		DevOps + CI/CD
	Lift & Shift (OS, 데이터, 애플리케이션을 그대로 옮기는 것. Re-호스트와 동일 개념)	Re-Factor (클라우드 환경 아키텍처 제공)
기존 애플리케이션	클라우드 준비단계 애플리케이션	클라우드 친화단계 애플리케이션 → 클라우드 네이티브 애플리케이션
Re-호스트	Re-호스트 (Re-호스트 +a)	Re-Purchase (SaaS형태로 구매)

클라우드 네이티브 시장 변화

’24년 세계 시장규모는 5,131억달러로 ’18년 이후 매년 17.1%의 높은 성장세를 지속할 것으로 예상



자료 : 가트너 (‘20.10)



자료 : 가트너 (‘20.10)

클라우드 네이티브 기반 행정·공공 서비스 확산 지원

클라우드 네이티브 발주자 가이드

어떤 공공업무에 클라우드 네이티브의 적용이 가능할까요?

NIA 한국지능정보사회진흥원

클라우드 네이티브 대상 업무선정 방향 (전문가 의견)

클라우드 네이티브 업무는 학계, 업체, 정부정책을 반영하여 대상을 선정할 수 있습니다.

서비스 복잡도가 높은 시스템

- 마틴파울러(2015, 최초 용어정의) : "마이크로서비스는 복잡한 시스템에서 유용할 때 MSA전환"

명확한 경계가 가능한 시스템

- 샘뉴먼(2019, 저서) : "해당분야를 제대로 이해하지 못해 적절한 경계를 찾기 어렵다면 MSA전환 불리"

더 이상 확장할 수 없는 한계지점에 도달한 시스템

- 수잔파울러(2019, 저서) : "확장성 한계로 인해 심각한 안정 문제 발생하여, 개발생산성·효율성 저하 시 MSA 전환"

정부정책

클라우드 네이티브 적용 검토 (1/4)

클라우드 네이티브는 변화가 많은 대국민 서비스 중심으로 우선 검토하여야 합니다.

정부내 전환가능 업무 식별

SRM(서비스 참조 모델)의 전환가능 업무

정부내지원 서비스 →	업무변화 少 (적용 불리)
공통기술 서비스 →	시스템 연계 多 (적용 불리)
대국민 서비스 →	업무변화 多/ 폭주성 高 (적용 유리)

기관·시스템특성을 반영한 업무선정

시스템 특성 반영 (시스템 복잡성)

규모(大) × 부서(多) × 시스템 메뉴(多)

※국가 및 기초자치 단체 226, 공공기관 338개 기관 업무대상

정부정책 특성 반영 (제도개선이 많은 업무)

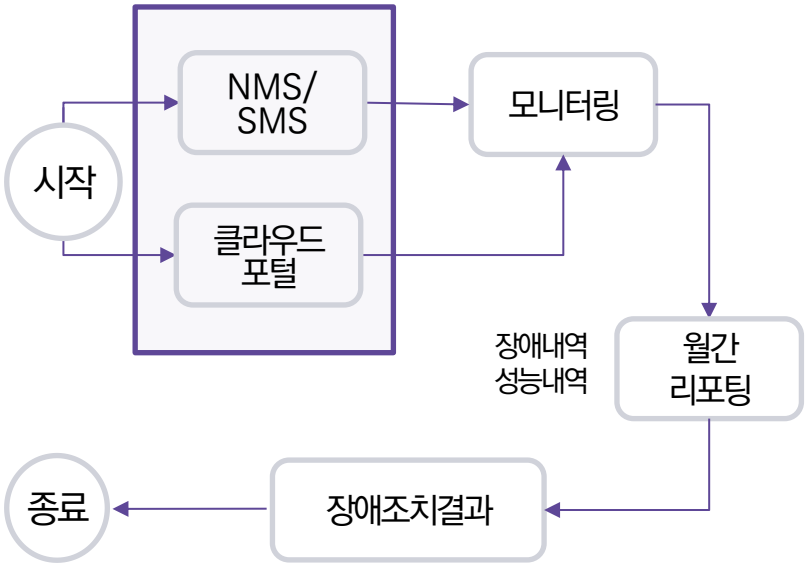
A기관	B기관	C기관	D기관

클라우드 네이티브 적용 검토 (2/4)

빈번한 장애, 폭주, 변경 및 배포가 존재하는 한계 지점에 도달한 시스템 중심으로 우선 검토하여야 합니다.

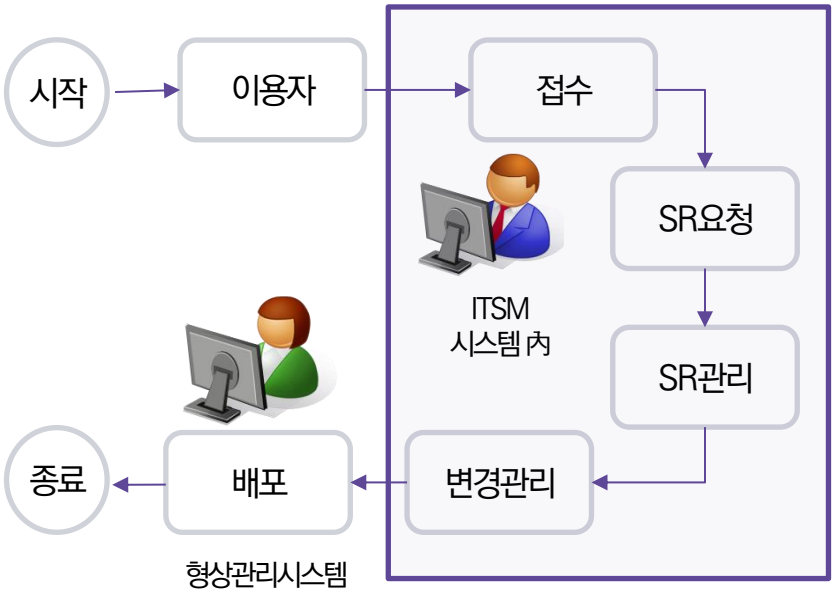
시스템 장애 및 폭주 발생 업무

시스템 장애 발생내역과 성능분석 절차



빈번한 시스템 변경과 배포 업무

SR 변경신청 및 배포횟수 식별

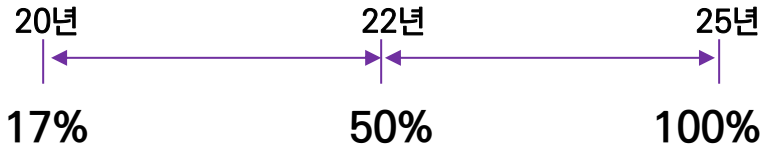


클라우드 네이티브 적용 검토 (3/4)

공공클라우드 전면 전환에 따른 클라우드 네이티브 상호 운용성 확보하여 서비스간 연결해야 합니다.

공공클라우드 전면 전환 사업진행

- 공공기관 전면전환 비율 (목표)

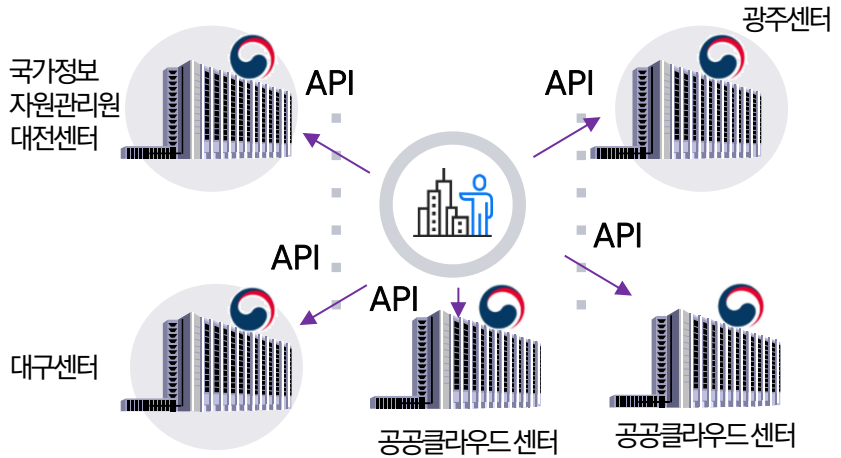


- 공공 (G-클라우드, 자체), 민간클라우드 센터

공공 클라우드센터	민간 클라우드센터
54%	46%

공공클라우드 센터 상호운용성 확보

- 디지털정부 서비스 개발환경인 클라우드 표준 플랫폼으로 고도화 필요



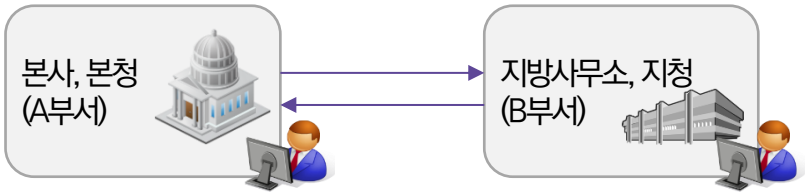
※ 마이크로 서비스 아키텍처에서는 API로 멀티센터의 서비스를 통합 제공

클라우드 네이티브 적용 검토 (4/4)

공공조직의 공통적인 특성(순환보직)을 감안하여, 담당 업무별 기능 단위로 서비스를 분리 구성하여 영향도를 최소화하여야 합니다.

공무원 인사규정 보직변경

통상적인 조직개편 및 잦은 순환보직



➔ 공무원 인사규정 :
동일 직위에서 2년 이상 근무한 경우 타 직위 또는 타 부서로 전보

내 업무 시스템만 관심

타 시스템 (多) – 관련부서 A, B, C, D ~ Z 부서

내 업무시스템(少)

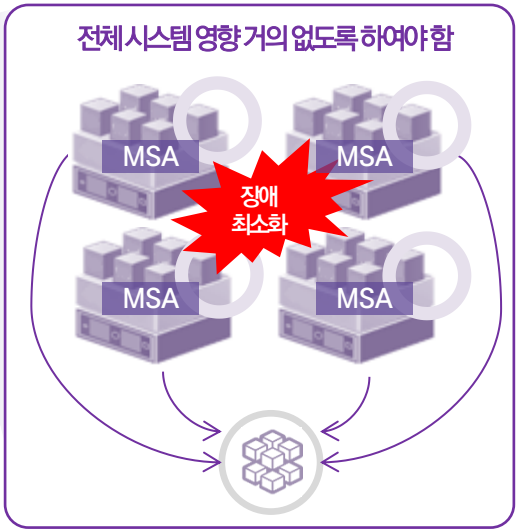
➔ 전체 시스템을 모두 이해하기 어려움. 업무담당 시스템 대비 타 시스템 관련(유관부서) 비중이 높음

전체시스템 변경협조는 차세대에서나 가능

전체 시스템 영향 없이, 내 업무 중심으로 관리하고 배포
(점차적으로 모든 부서가 동일하게 기능 및 서비스단위 분리)

내 업무시스템 변경

어떤 영향, 장애 걱정
➔
'내 프로그램 변경에 타 시스템이 영향 없도록 …'



정보시스템 자가진단 방법

클라우드 네이티브 구성요소의 특장점을 토대로, 현행 정보시스템에 대한 체크리스트를 도출 하였습니다.

정보시스템 1개씩 적합성 검토 수행

대상 정보시스템

클라우드 적용 대상 여부

Y

정보시스템 혁신필요 여부

Y

적합성 검토 수행 (체크리스트 활용)

체크리스트 검토결과

6개 이상 Yes

클라우드 네이티브 도입

N

기존 시스템 유지

N

기존 아키텍처 유지

6개 미만 Yes

기존 아키텍처 유지

도입 우선 순위 결정

- 시스템 규모 : 大
- 복잡도 : 高
- 확장성 : 低

정보시스템 자가 진단 체크리스트

현행 정보시스템에 대한 체크리스트를 도출하였으며, 6개 이상 "Y"응답시, 클라우드 네이티브 도입이 필요한 것으로 판단할 수 있습니다.

정보시스템 자가진단 체크리스트 예시

구분(목표)		자가진단 항목	답변	
안정적 서비스 운영	1	• 초기개발비의 약 15%이상을 매년 추가개발 및 유지보수 비용으로 사용하고 있습니까?	✓	6개 이상 "YES" 응답 시 도입 검토
	2	• 다양한 원인에 의한 장애 발생 시 장애복구(예.시스템 증설, 업그레이드 등)를 위해 서비스를 중단한 적이 있습니까?	✓	
	3	• 특정시점(년, 월, 주, 시)에 트래픽이 증가로 접속지연으로 불만이 제기된 적이 있습니까?		
업무 및 기술 변화 대응	4	• 수시로 정책, 업무 요건 등의 변화에 따른 요구사항에 대해 신속한 대응이 필요합니까?		
	5	• 디지털 신기술(빅데이터, AI, 블록체인, IoT 등) 적용 및 다양한 언어 및 다양한 오픈소스에 대한 요구사항 반영이 필요합니까?	✓	
	6	• 소규모 서비스 단위로 기능과 DB의 명확한 분리가 가능하고, 독립적 단위로 실행이 가능합니까? (공통 기능 및 데이터 사용, 타 시스템과의 연계성, 서비스 의존관계 등 확인)	✓	
개발 품질 향상	7	• 시스템 개발 및 운영시 개발 및 운영 조직의 분리에 따라 의사소통, 개발 및 배포 지연 등의 문제가 존재합니까?	✓	
	8	• 소스코드의 복잡성으로 서비스 확장이 곤란하여 서비스 분리 및 소스코드 개선이 필요합니까?	✓	
개발기간	9	• 개발된 SW를 형상관리 시스템에 커밋한 후 개발계, 검증계, 운영계 서버에서 빌드, 테스트, 배포하는 과정에 빌드·테스트·배포 도구를 사용하지 않거나 부분적으로 사용하고 있습니까 ?		
	10	• 현행 시스템의 배포주기를 단축하고 싶습니까?		

클라우드 네이티브 기반 행정·공공 서비스 확산 지원

클라우드 네이티브 발주자 가이드

클라우드 네이티브 도입 고려사항은 어떻게 되나요?

NIA 한국지능정보사회진흥원

정보화 사업관리 프로세스

공공기관에서 추진하는 정보화 사업관리는 기획·검토, 계획수립, 사업자 선정계약, 개발구축, 감리, 검사종료 단계에 따라 진행됩니다.

기획·검토

기획·검토
- 정보화사업 기획
- 정보화사업 검토

계획수립

사업계획서(안) 작성
- 사업계획서(안) 작성
- 기술적용계획 수립

사업계획서(안) 검토
- 기술평가 시행
- 보안성 검토

사업계획서(안)
- 사업계획서(안) 수정·확정

사업자 선정·계약

제안요청
- 제안요청서 작성
- 제안안내서 작성
- 입찰공고
- 제안요청설명회 개최(필요시)
- 입찰마감

사업자 선정 및 계약 체결
- 제안서 평가
- 협상 및 낙찰자 선정
- 제안서 보상
- 계약서류 준비 및 계약체결

개발·구축

사업착수
- 착수계획서 검토

진도/품질관리
- 진도/품질관리

변경관리
- 과업변경

예산관리
- 선금 지급
- 기성대가 지급

보안관리
- 준비단계 보안
- 수행단계 보안
- 종료단계 보안
- 보안적합성 검증필 정보시스템 도입
- 개인정보보호

하도급관리
- 하도급관리

감리

감리법인 선정
- 감리법인 선정

감리시행
- 예비조사 실시
- 감리계획 접수
- 감리시행
- 감리결과 조치
- 감리결과 조치 확인

감리 사업관리
- 감리 변경관리
- 감리 보안관리

감리 완료
- 검사 및 대가지급

검사종료

완료검사
- 완료검사

인수 및 하자보수
- 산출물 인수
- 하자보수 내용 확정

사업종료
- 지체상금 산정 및 정산(필요시)
- 잔금대가 지급

클라우드 네이티브 추진 관련 사항 검토 대상

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

사업발주를 위한 사업관리 단계별 고려사항 (1/2)

기획·검토, 계획수립, 사업자선정·계약, 개발·구축, 감리, 검사 종료, 사업자 선정·계약 이슈입니다.

발주 단계			클라우드 네이티브 사업단계별 고려사항
기획·검토	기획·검토	정보화사업 기획	• 클라우드 네이티브 시스템 구축 관련 성과계획 수립
계획수립	사업계획서(안) 작성	사업계획서(안) 작성	• 클라우드 네이티브 관련 요구사항 상세화 • 클라우드 네이티브 도입을 고려한 개발비 예산 산정
		기술적용계획 수립	• 클라우드 네이티브 애플리케이션 관련 기술적용계획 수립
	사업계획서(안) 검토	기술평가 시행	• 클라우드 네이티브 관련 요구사항 상세화
사업자 선정·계약	제안 요청	제안요청서 작성	• 클라우드 네이티브 관련 요구사항 상세화 • 클라우드 네이티브 도입을 고려한 개발비 예산 반영

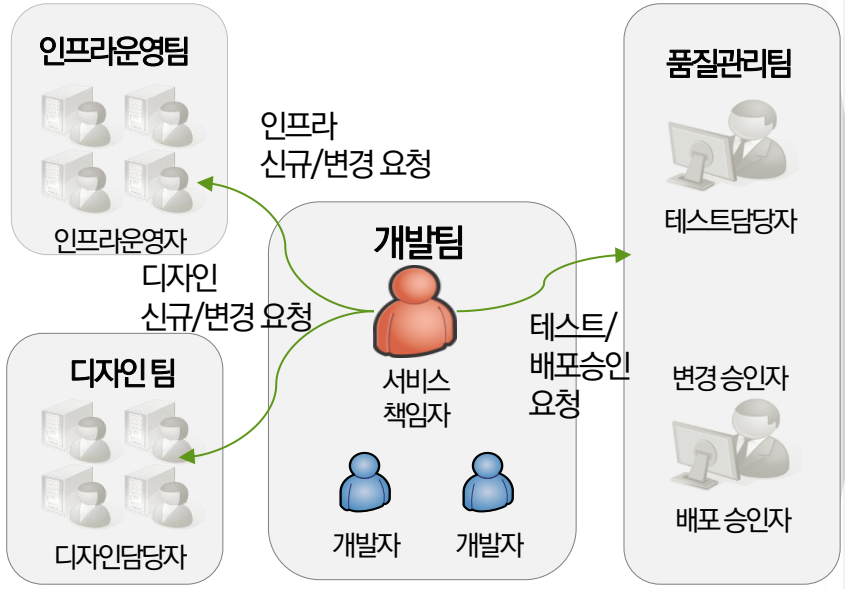
사업발주를 위한 사업관리 단계별 고려사항 (2/2)

발주 단계			클라우드 네이티브 사업단계별 고려사항
사업자 선정·계약	사업자 선정 및 계약 체결	제안서 평가	• 기술평가 수행 시 **클라우드 네이티브 적용 경험 및 기술력을 평가할 수 있는 항목 정의 및 배점** 부여
		협상 및 낙찰자 선정	• 기술협상 시 **클라우드 네이티브 관련 기술적 이행 사항에 대해 구체적으로 협상** 수행
개발·구축	사업착수	착수계획서 검토	• **착수계획서 내**에 클라우드 네이티브 관련 과업 내용이 누락 없이 잘 반영되어 있는지 확인
	진도/품질관리	진도/품질관리	• **방법론의 공정단계별 품질관리 활동 수행** – 개발방법론의 공정단계별 품질관리 활동 수행 – **클라우드 네이티브 관련 요구사항별 이행 과정 점검**
감리	감리시행	감리시행	• **감리계획서 및 수행결과보고서 검토 및 조치** – 감리계획서 및 수행결과보고서 내에 클라우드 네이티브 관련 요구사항에 대한 점검계획과 결과 검토 후 조치 요구
검사종료	완료검사	완료검사	• **완료검사 수행 시 클라우드 네이티브 관련 요구사항의 이행 여부 확인**

클라우드 네이티브 조직 변화

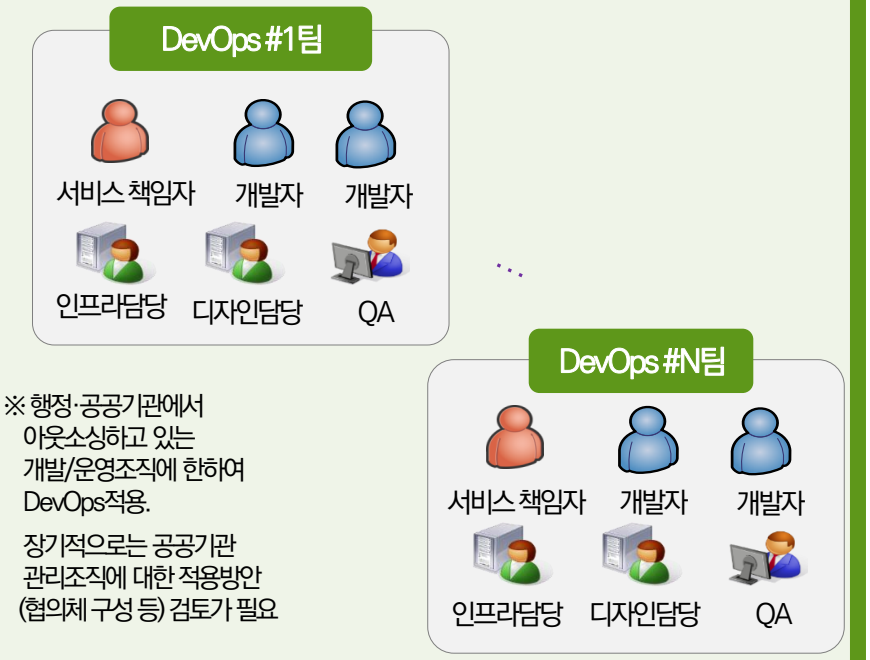
인프라·플랫폼/디자인/개발/운영 비즈니스에 민첩한 대응이 가능한 조직으로 변화해야 합니다.

일반적 개발·운영 방식



개발팀이 인프라/디자인/품질관리팀에 연락하여 신규·변경을 요청

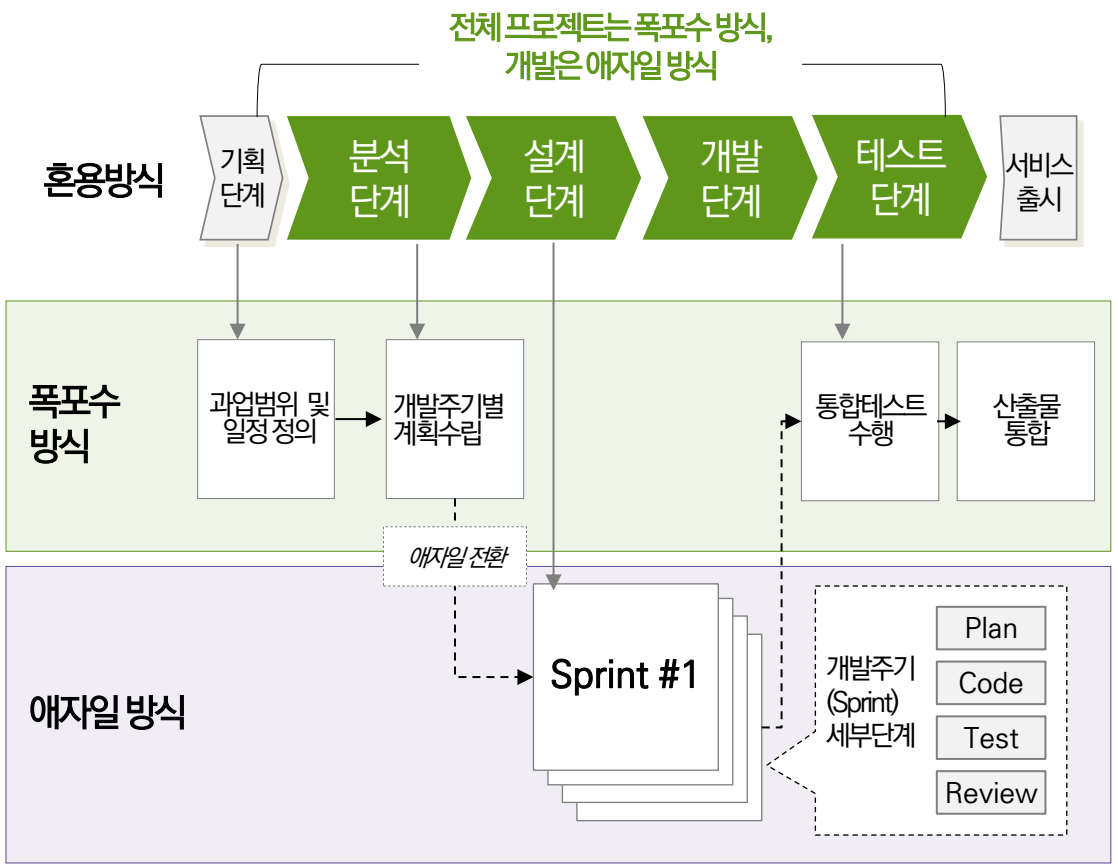
마이크로서비스 제공을 위한 개발·운영 방식



서비스 단위로 DevOps조직을 운영하고 운영환경에 직접배포

클라우드 네이티브 환경조성을 위한 애자일 방법론 적용

공공부문 애자일 적용은 현실적으로 애자일 방법론을 폭포수 방법론의 보완적인 역할로 폭포수-애자일 혼용방식(하이브리드)의 적용이 합리적입니다.



계약방식	• 확정된 예산금액 만큼 고정가 계약 • 과업범위, 수행인력 및 계약기간 명시
과업범위	• 요구사항 및 품질기준을 최대한 확정 후 프로젝트 착수
개발방식	• 개략적인 개발일정으로 프로젝트 착수, 개발주기(Sprint)별 상세계획에 따라 설계,개발, 테스트 및 평가 수행
결과물 (산출물)	• 개발주기(Sprint)별 산출물 분할 작성 후 프로젝트 종료시점까지 산출물 통합 및 검수 진행
수행기간	• 계약서상의 도급계약 기간 준수

※국내 공공부문의 SW 조달체계는 기본적으로 과업범위(요구사항 확정)와 예산이 확정된 고정가 계약을 전제로 하고 있어, 계약에 명시된 목적물(시스템 및 관련 산출물)을 정해진 기간 안에 제공.
국외 공공부문의 경우 애자일을 도입하는 사례가 점점 증가하고 있음

클라우드 네이티브 기반 행정·공공 서비스 확산 지원
클라우드 네이티브 발주자 가이드

클라우드 네이티브 도입 절차는 어떻게 되나요?

NIA 한국지능정보사회진흥원

클라우드 네이티브 프로세스

행정기관을 위한 정보화 사업 단계별 관리 및 점검가이드를 참조하여 수립한
클라우드 네이티브 발주자(기획자) 프로세스입니다.

MSA

정보화전략계(ISP) 추진

① 정보화의 필요성

② 현황분석

③ 개선방안 수립

④ 사업실행계획 수립

모놀리틱 vs.
마이크로 서비스

클라우드 네이티브 적합성 검토

⑤ 사업 주요 고려사항

MSA 필요한 기술보유
비즈니스 규모 인력
배포도구 선정
DevOps 조직 구성

분석, 설계,
테스트 산출물
(클라우드 네이티브 적용여부)

전자정부 클라우드
플랫폼에서 제공하는 서비스 이용,
전자정부 표준프레임워크
적용검토

MSA개발비
1.2~1.4배 산정

⑩ 인수관리

⑨ 사업발주

⑧ 사업계획서 작성

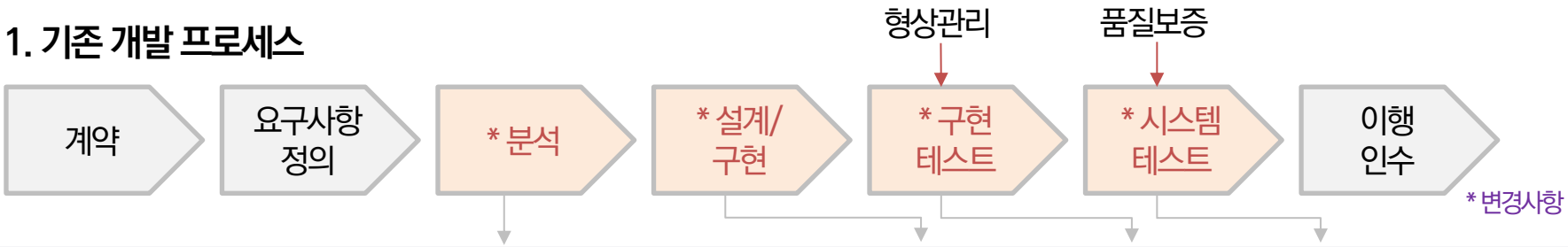
⑦ 기술적용계획/기술평가
/보안대책수립

⑥ 사업비 산정

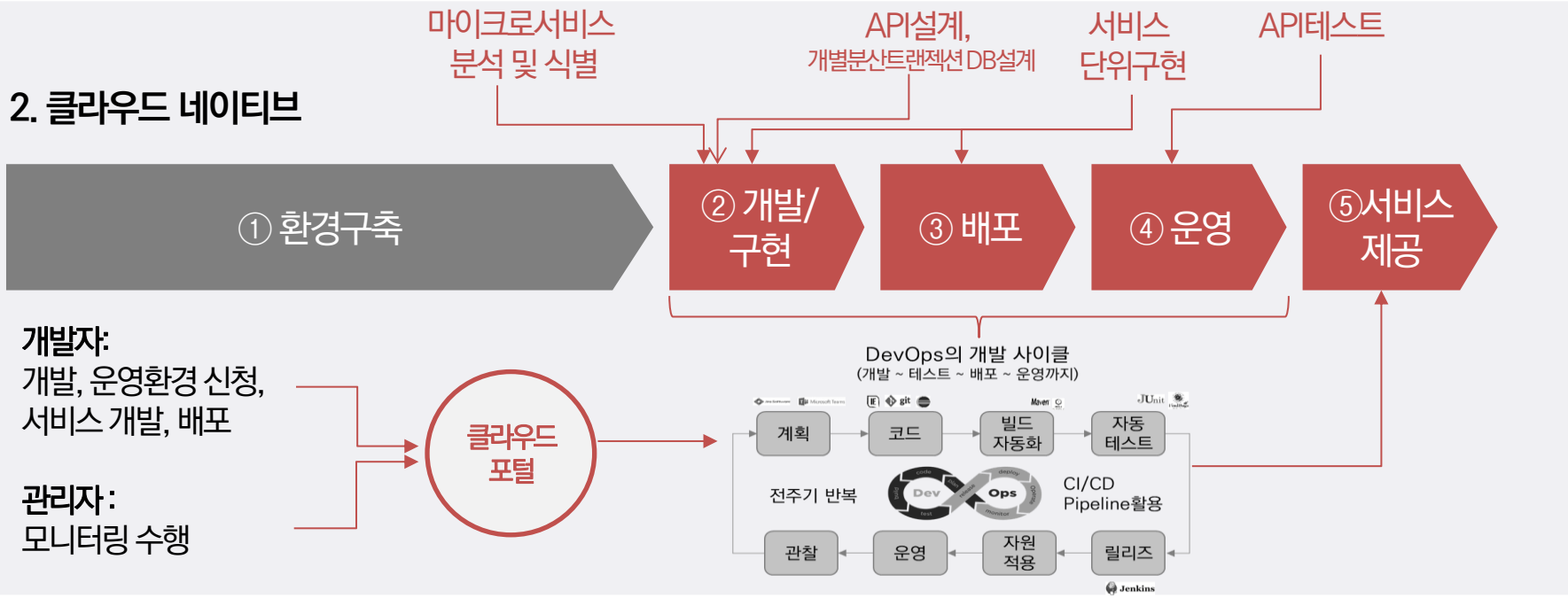
클라우드 네이티브 프로세스

행정기관을 위한 정보화 사업 단계별 관리 및 점검가이드를 참조하여 수립한 클라우드 네이티브 개발자 프로세스입니다.

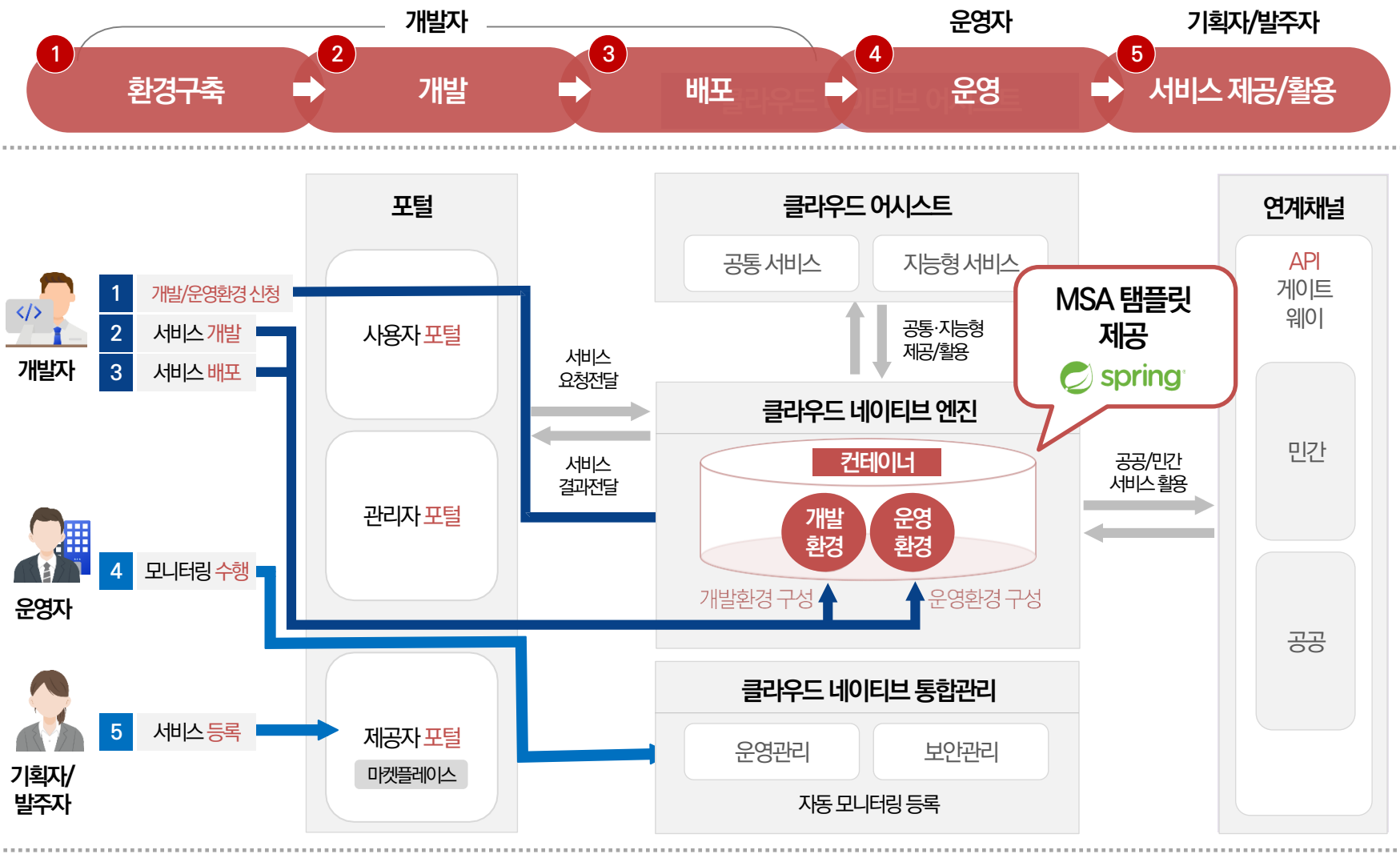
1. 기존 개발 프로세스



2. 클라우드 네이티브

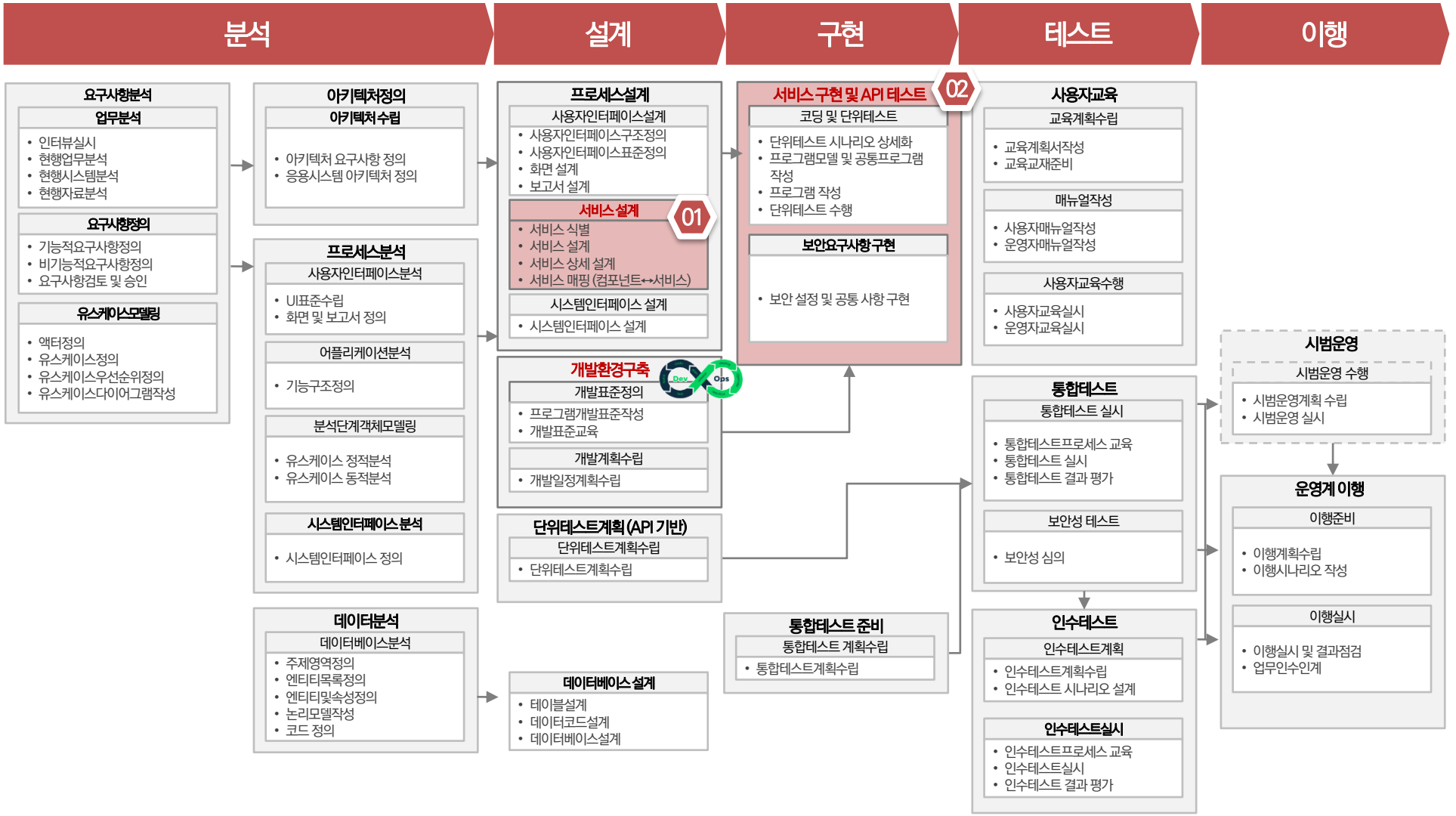


전자정부 클라우드 플랫폼 기반 클라우드 네이티브 애플리케이션 개발 절차



클라우드 네이티브 개발 절차 적용예시 (1/2)

CBD방법론을 기반한 마이크로 서비스 개발을 순차적으로 적용합니다.



클라우드 네이티브 개발 절차 적용예시 (2/2)

애자일 방법론을 중심으로 집중 적용합니다.

분석 → 설계 → 구현 → 테스트 → 이행

Sprint 실행 (설계 ~ 구현 ~ 테스트)

Life Cycle(WBS) vs. Sprint 실행 (1~N)

분석

요구사항분석
- 업무분석
  - 인터뷰실시
  - 현행업무분석
  - 현행시스템분석
  - 현행자료분석
- 요구사항정의
  - 기능적요구사항정의
  - 비기능적요구사항정의
  - 요구사항검토 및 승인
- **이벤트스토밍** 01
  - Scrum팀 구성
  - 이벤트 스토밍의 준비사항
  - 워크숍 방식 정의
  - 포스트-잇 유형별 의미
  - 요구사항 백로그 도출
  - 도메인 이벤트 도출
  - 커멘드 도출
  - 앤티티(어그리게잇) 도출 (대상업무)
  - 컨텍스트 경계 그리기
  - 컨텍스트 매핑하기

산출물 축소

아키텍처정의
- 아키텍처수립
  - 아키텍처 요구사항 정의
    - –서비스 분해지침
    - –API 설계원칙
    - –표준 및 명명규칙
  - 응용시스템 아키텍처 정의
  - **인프라 구조 정의** 02
    - –서비스 메쉬, G/W, CI/CD
- **마이크로서비스 입력물정리** (API, DATA, Event, Interface)

산출물 축소

프로세스분석
- 사용자인터페이스분석
  - UI표준수립
  - 화면 및 보고서 정의
- 어플리케이션분석
  - 기능구조정의
- 시스템인터페이스분석
  - 시스템인터페이스 정의

산출물 축소

데이터분석
- **데이터베이스분석**
  - 주제영역정의
  - 엔티티 목록정의
  - 엔티티 및 속성정의
  - 논리모델작성, 코드 정의
  - **분산트랜젝션 처리방안**

설계

산출물 축소

프로세스설계
- Front-End사용자인터페이스설계
  - 사용자인터페이스구조정의
  - 사용자인터페이스표준정의
  - 화면 설계
  - 보고서 설계
- **Back-End서비스설계** 03
  - 서비스 식별
  - 서비스 설계 (컴포넌트 X)
  - 서비스 상세 설계 (API설계, 도메인 모델링)
- Front-End시스템인터페이스설계
  - 시스템인터페이스 설계

개발환경구축 (Dev Ops)
- 개발표준정의
  - 프로그램개발표준작성
  - 개발표준교육
- 개발계획수립
  - 개발일정계획수립

단위테스트계획(API기반)
- 단위테스트계획수립
  - 단위테스트계획수립

개별분산데이터베이스설계 04
- 테이블설계
- 데이터코드설계
- 데이터베이스설계

구현

서비스구현및API테스트 05
- **코딩및단위테스트**
  - 단위테스트 시나리오 상세화
  - 프로그램모델 및 공통프로그램 작성
  - 프로그램 작성
  - 단위테스트 수행

보안요구사항구현
- 보안 설정 및 공통 사항 구현

통합테스트준비
- 통합테스트계획수립
  - 통합테스트계획수립

테스트

사용자교육
- 교육계획수립
  - 교육계획서작성
  - 교육교재준비
- 매뉴얼작성
  - 사용자매뉴얼작성
  - 운영자매뉴얼작성
- 사용자교육수행
  - 사용자교육실시
  - 운영자교육실시

통합테스트
- 통합테스트실시
  - 통합테스트프로세스 교육
  - 통합테스트 실시
  - 통합테스트 결과 평가
- 보안성테스트
  - 보안성 심의

인수테스트
- 인수테스트계획
  - 인수테스트계획수립
  - 인수테스트 시나리오 설계
- 인수테스트실시
  - 인수테스트프로세스 교육
  - 인수테스트실시
  - 인수테스트 결과 평가

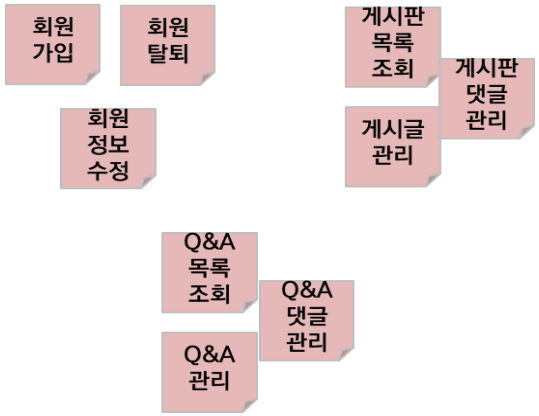
이행

시범운영
- 시범운영수행
  - 시범운영계획 수립
  - 시범운영 실시

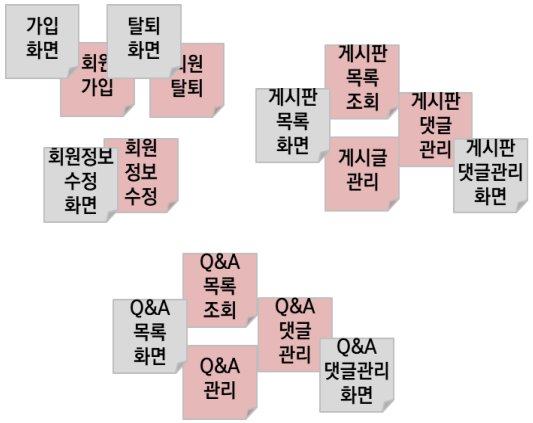
운영계이행
- 이행준비
  - 이행계획수립
  - 이행시나리오 작성
- 이행실시
  - 이행실시 및 결과점검
  - 업무인수인계

클라우드 네이티브 전환을 위한 마이크로 서비스 식별 결과 예시

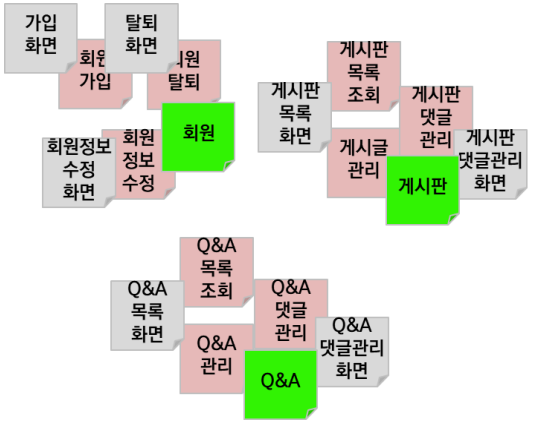
① 이벤트 도출



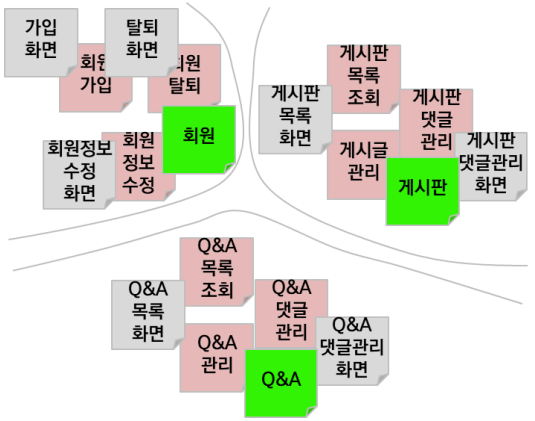
② 커멘드 도출



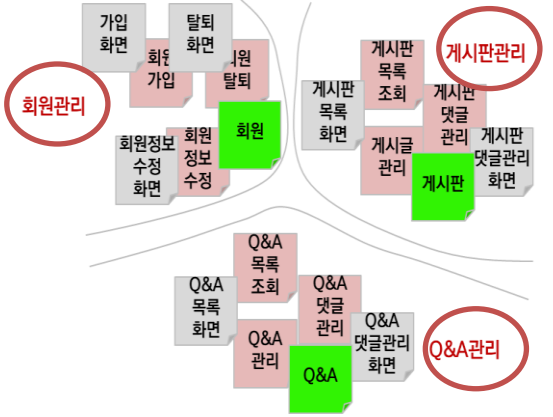
③ 엔티티(에그리게잇) 도출



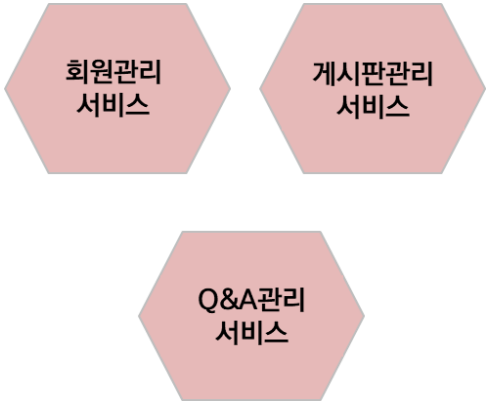
④ 컨텍스트 경계 그리기



⑤ 마이크로 서비스 식별(컨테스트 매핑)



⑥ 마이크로 서비스 식별 결과



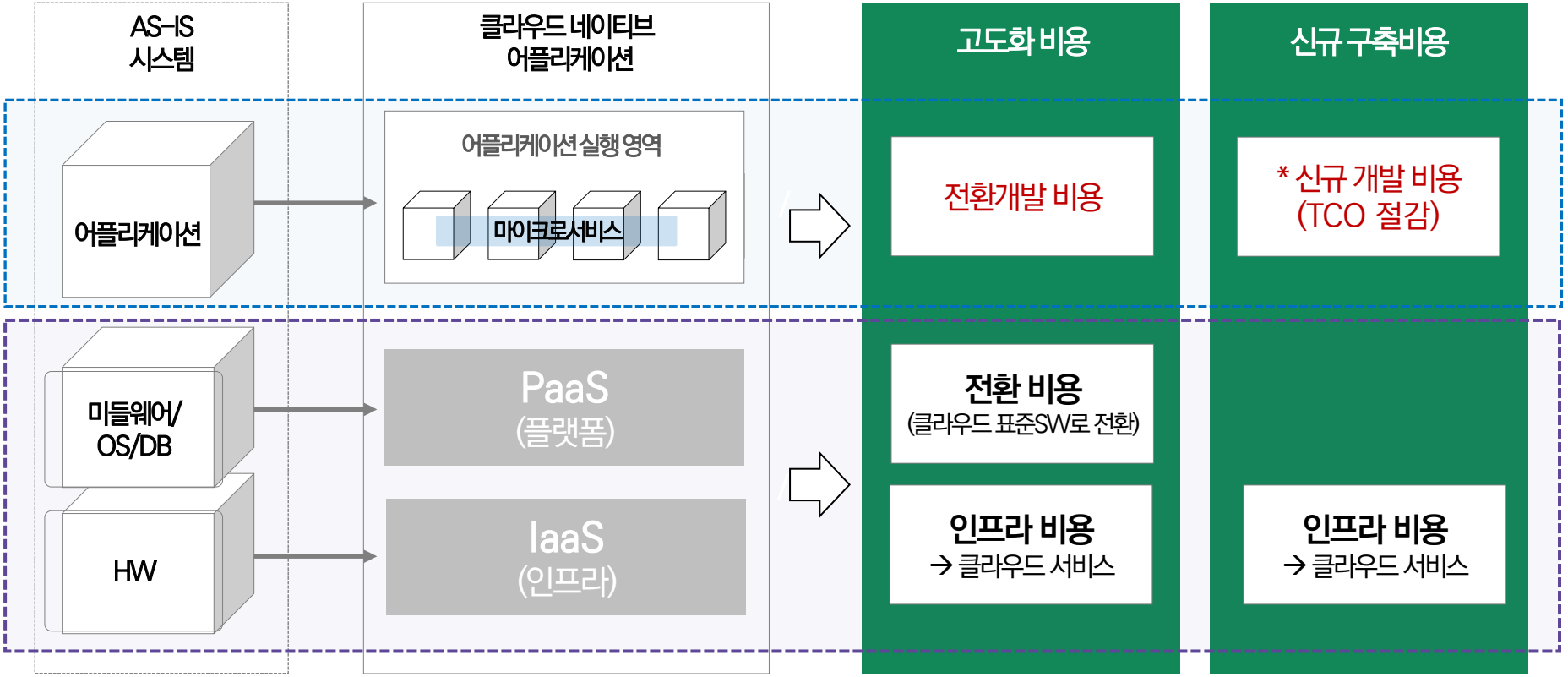
클라우드 네이티브 기반 행정·공공 서비스 확산 지원
클라우드 네이티브 발주자 가이드

클라우드 네이티브 도입 비용산정은 어떻게 해야 하나요?

NIA 한국지능정보사회진흥원

클라우드 네이티브 비용 산정 개요

클라우드 네이티브 어플리케이션 전환 및 신규 구축을 위한 예산을 수립한 후 발주절차를 진행합니다.



* 신규 구축 개발비용(용역비용) = 기존과 유사하게 정보시스템의 기능점수(FP)를 집계하는 방식으로 비용을 산정하되 클라우드 네이티브 어플리케이션 유형을 반영하도록 보정계수의 조정이나 기능점수(FP)의 추가

클라우드 네이티브 애플리케이션 개발 비용의 포지셔닝

'21년 정보화전략계획 수립 공통가이드에 의하면, 클라우드 네이티브 애플리케이션 구축 시 SW 개발비 산정에 대한 검토가 필요합니다.

대분류	중분류	소분류	관련 기준	비고
시스템 구축비	HW 구매비	HW 구매/설치비	정보시스템 HW 규모산정 지침(한국정보통신기술협회)	클라우드 서비스
	SW 구축비	상용 SW 구입비		클라우드 서비스
		SW 개발비	SW사업 대가산정 가이드 (한국소프트웨어산업협회)	
	DB 구축비	DB 설계비	SW사업 대가산정 가이드(한국소프트웨어산업협회)	
		데이터 제작		
	시스템 운용환경 구축비	운용환경 설계비	엔지니어링 사업대가의 기준(산업통상자원부)	클라우드 서비스
		운용환경 공사(시설/통신망)		
	기존 시스템 이전비	HW 이전비		
		데이터 이전비		
부대비	감리비		정보시스템 감리기준(행정안전부)	

출처 : 정보화전략계획(ISP) 수립 공통가이드(2021년) 참조

클라우드 네이티브 애플리케이션 구축과 관련된 개발 비용

클라우드 네이티브 애플리케이션 개발 비용 산정

클라우드 네이티브 도입 시 측정 단위는
트랜잭션과 데이터 관점에서 마이크로서비스 단위로 변화가 필요합니다.

현행 기능 분류체계

애플리케이션
(예. 총무관리 시스템)

세부 업무
(예. 회의실 관리)

단위 프로세스
(예. 회의실 예약)

마이크로서비스 기반 개발 시 업무 및 프로세스 재정의 필요

클라우드 네이티브 도입 시 기능 식별의 변화

트랜잭션(EI, EO, EQ)

FP 산정을 위한 업무 및 프로세스의 변화 예상

- 기존의 업무 및 프로세스 단위를 마이크로서비스 단위로 변경할 것인지 의사결정 필요
- 마이크로서비스 기반 개발 시 업무 및 프로세스가 세분화될 수 있음
- 기관의 결정에 따라 기존의 업무 및 프로세스 체계대로 작업할 수 있음 (단, 마이크로서비스 관련 백엔드 작업에 대한 작업 비중은 보정계수로 조정할 수 있음)

데이터(ILF, EIF)

데이터 처리 및 연계 관련 프로세스의 변화 예상

- 일반적으로 마이크로서비스 단위별로 자체 데이터를 관리하므로 내부데이터 처리(ILF)와 외부데이터 처리(EIF) 관련 기능점수가 높아질 수 있음
- 기관에서 기존의 데이터 처리방식대로 FP 작업을 수행할 가능성 있음

클라우드 네이티브 기반 행정·공공 서비스 확산 지원
클라우드 네이티브 발주자 가이드

클라우드 네이티브의 아키텍처 참조모델은?

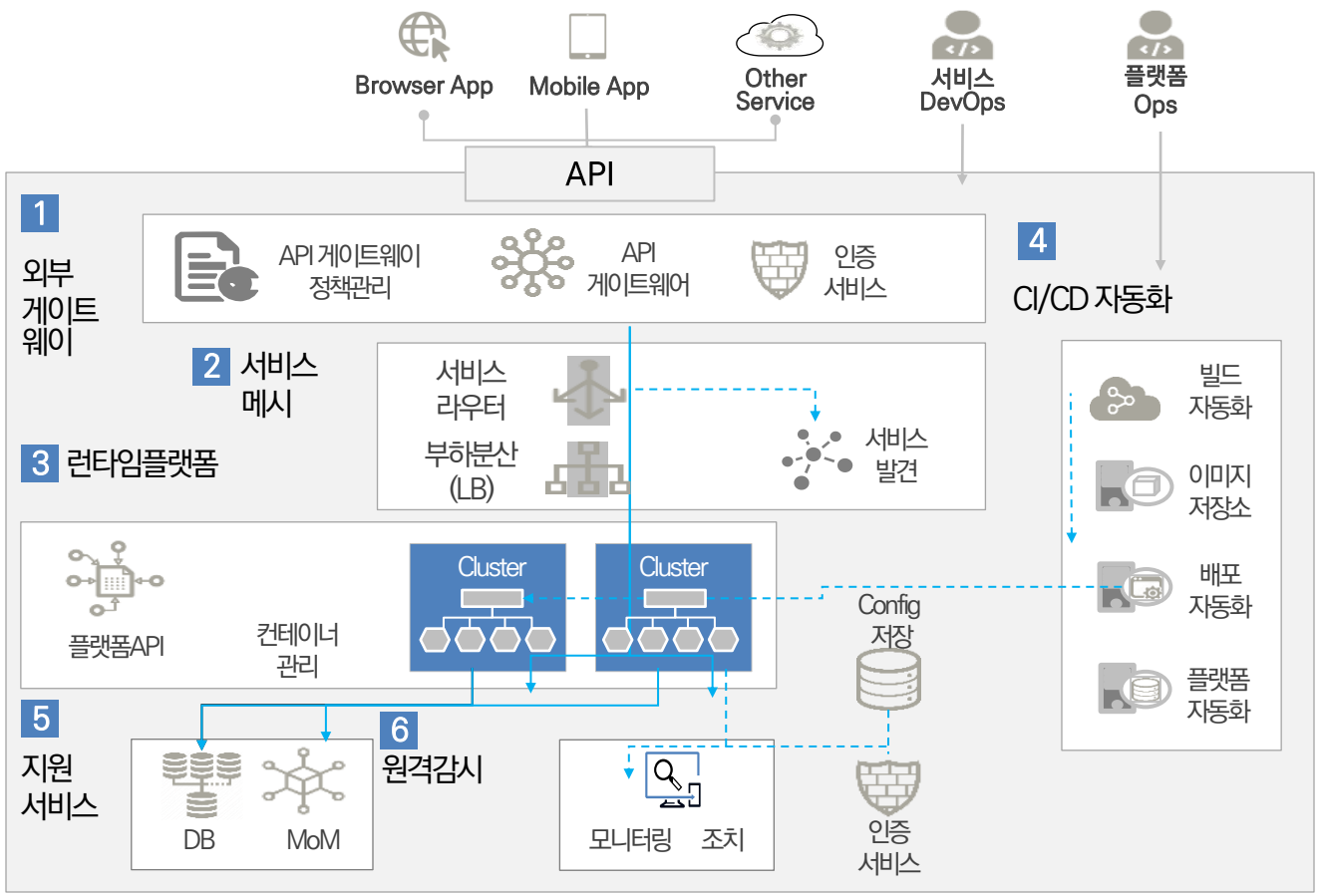
NIA 한국지능정보사회진흥원

가트너 참조모델

GARTNER

가트너는 클라우드 네이티브 아키텍처 참조모델을 제공합니다.

외부 게이트웨이, 서비스 메시, 런타임 플랫폼, CI/CD, 지원, 원격감시 구성



1 외부로부터의 서비스 요청을 내부 구조를 드러내지 않고 처리하기 위한 구성요소

2 마이크로서비스 구성 요소 간의 네트워크를 제어

3 마이크로서비스를 실행하기 위한 컨테이너와 그 컨테이너를 관리하는 역할

- Inner 아키텍처: 개별 마이크로 서비스 구축을 위한 아키텍처
- Outer 아키텍처: 개별 마이크로 서비스가 개발, 배포, 실행되는 전체 운영, 관리환경

4 마이크로서비스의 지속적인 통합, 지속적인 전달, 지속적인 배포 체계

5 애플리케이션이 실행하기 위해 지원되는 모든 서비스

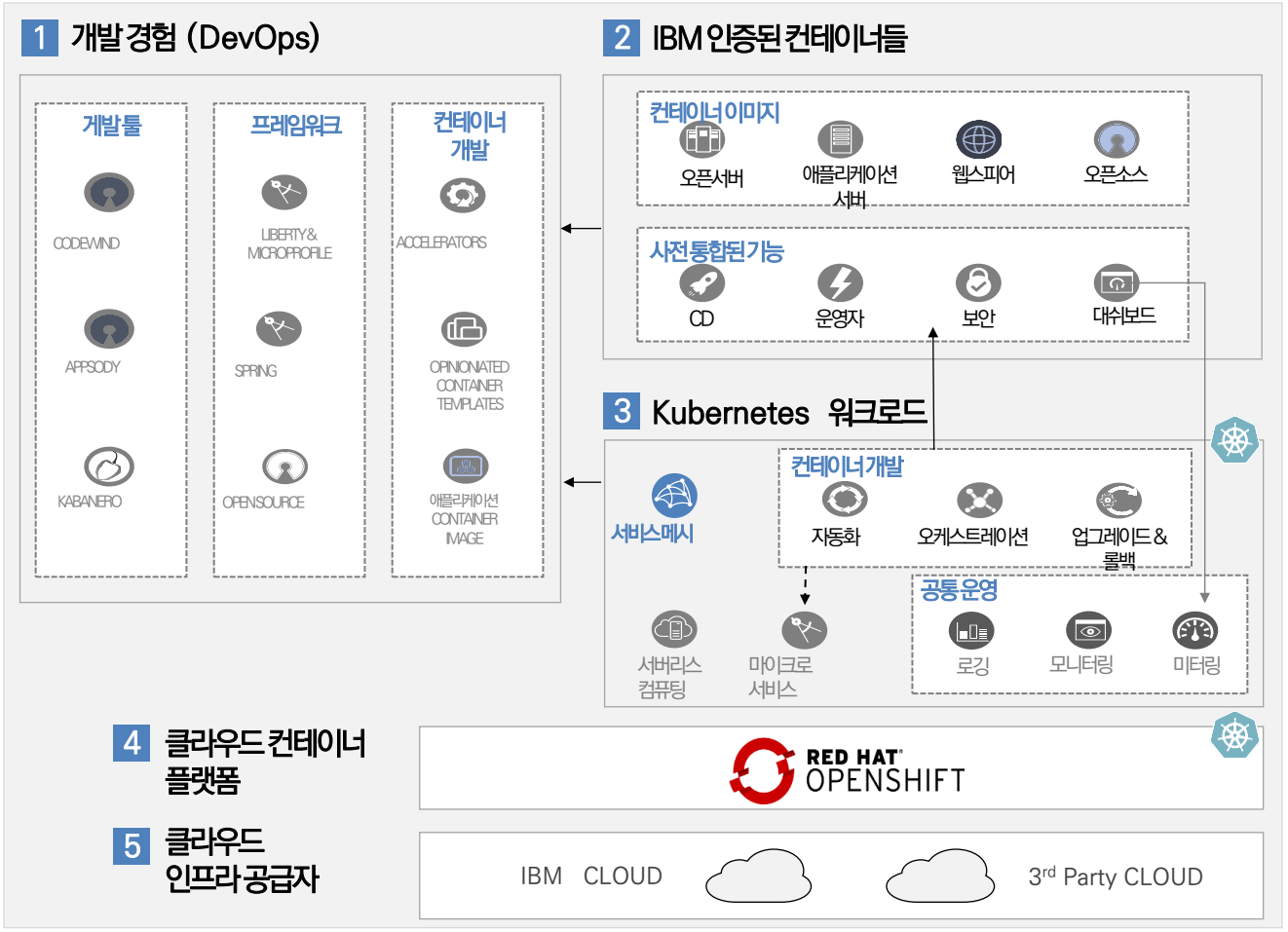
6 분산환경에서 서비스 모니터링 및 헬스체크

IBM 참조모델

IBM

IBM은 클라우드 네이티브 아키텍처 참조모델을 제공합니다.

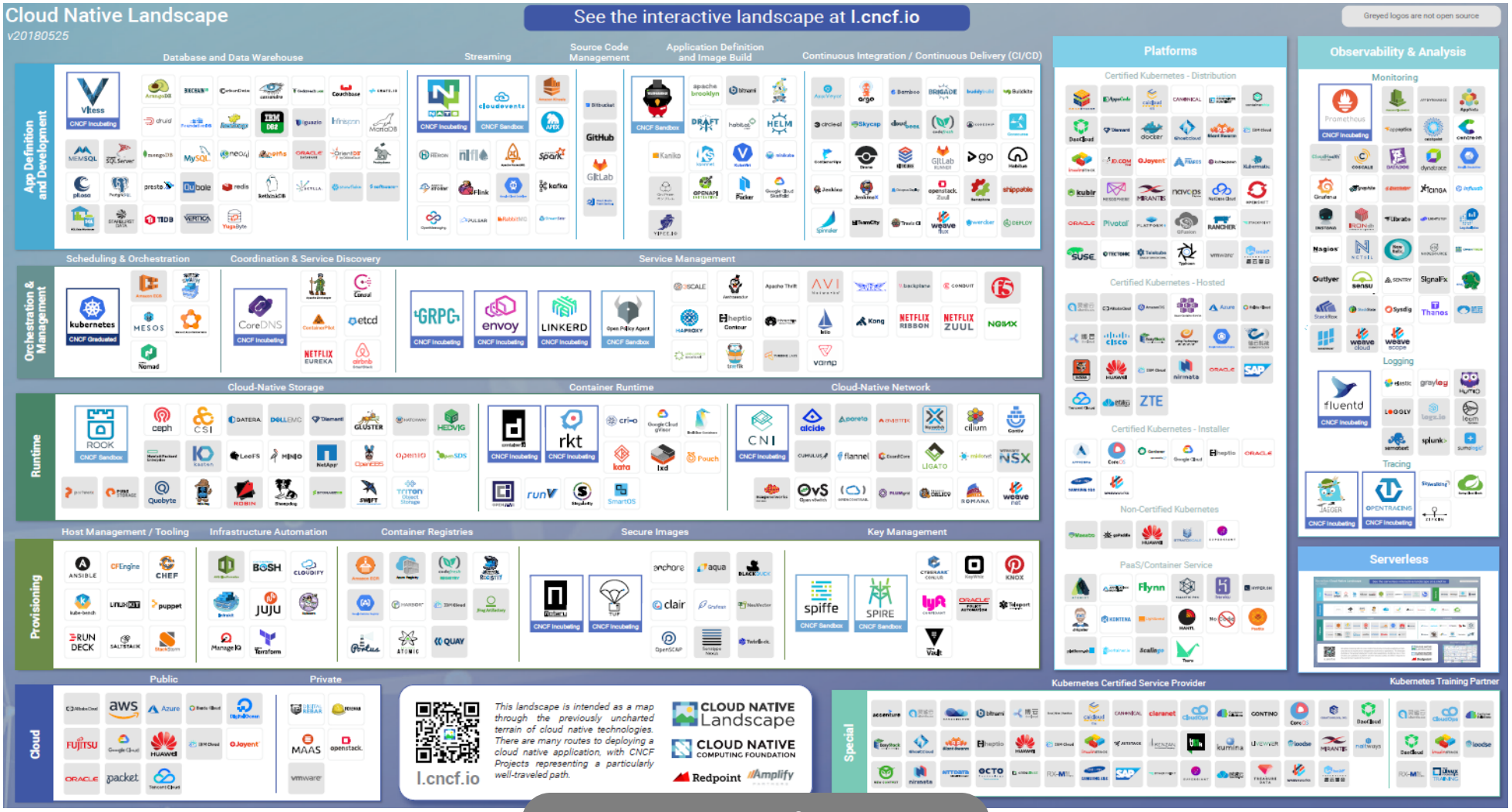
컨테이너/ 개발경험(DevOps) 중심으로 구성, 클라우드 인프라와 플랫폼은 서비스 임차



1. 개발자 및 아키텍트를 위한 로컬 개발, 테스트 영역
   개발을 위한 프레임워크, 런타임 사전 제공, 검증
2. 레드햇 오픈시프트 인증된 컨테이너 이미지 제공
   오픈소스에 대한 지속적인 통합 및 배포가 가능하도록 툴 제공
3. 서비스 메시를 사용할 수 있도록 Config 저장, 실행영역, 컨테이너 개발 및 운영팀을 위한 모니터링, 대시보드 등 관리기능 제공
4. 쿠버네티스 기반 컨테이너 오케스트레이션 및 실행 영역
   컨테이너 배포, 업데이트 및 워크로드별 리소스 관리
5. 마이크로서비스 운영하기 위한 인프라 스트럭처 영역
   컨테이너 플랫폼에 VM 및 베어메탈 자원을 제공

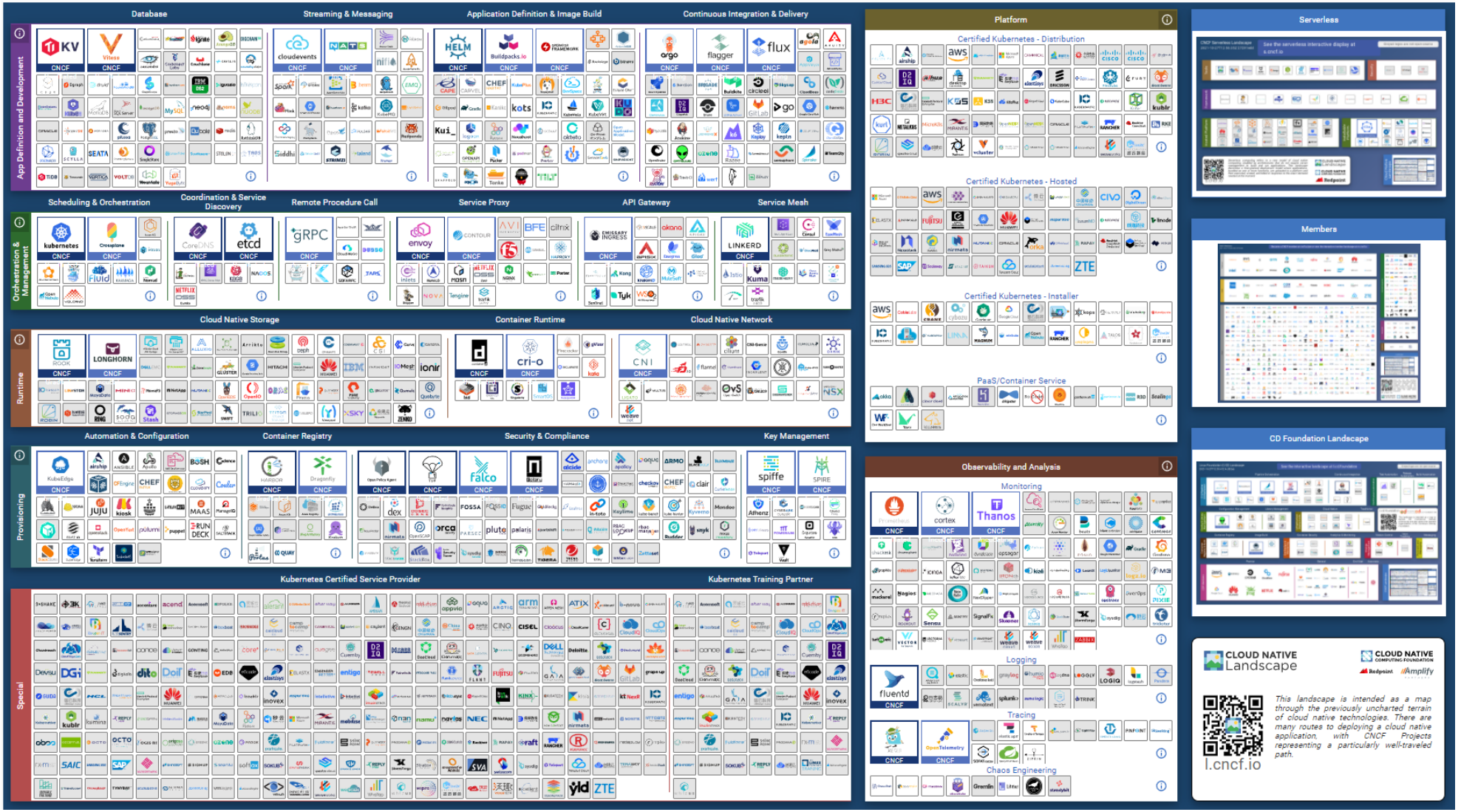
클라우드 네이티브 한눈에 알아 보기

2018년



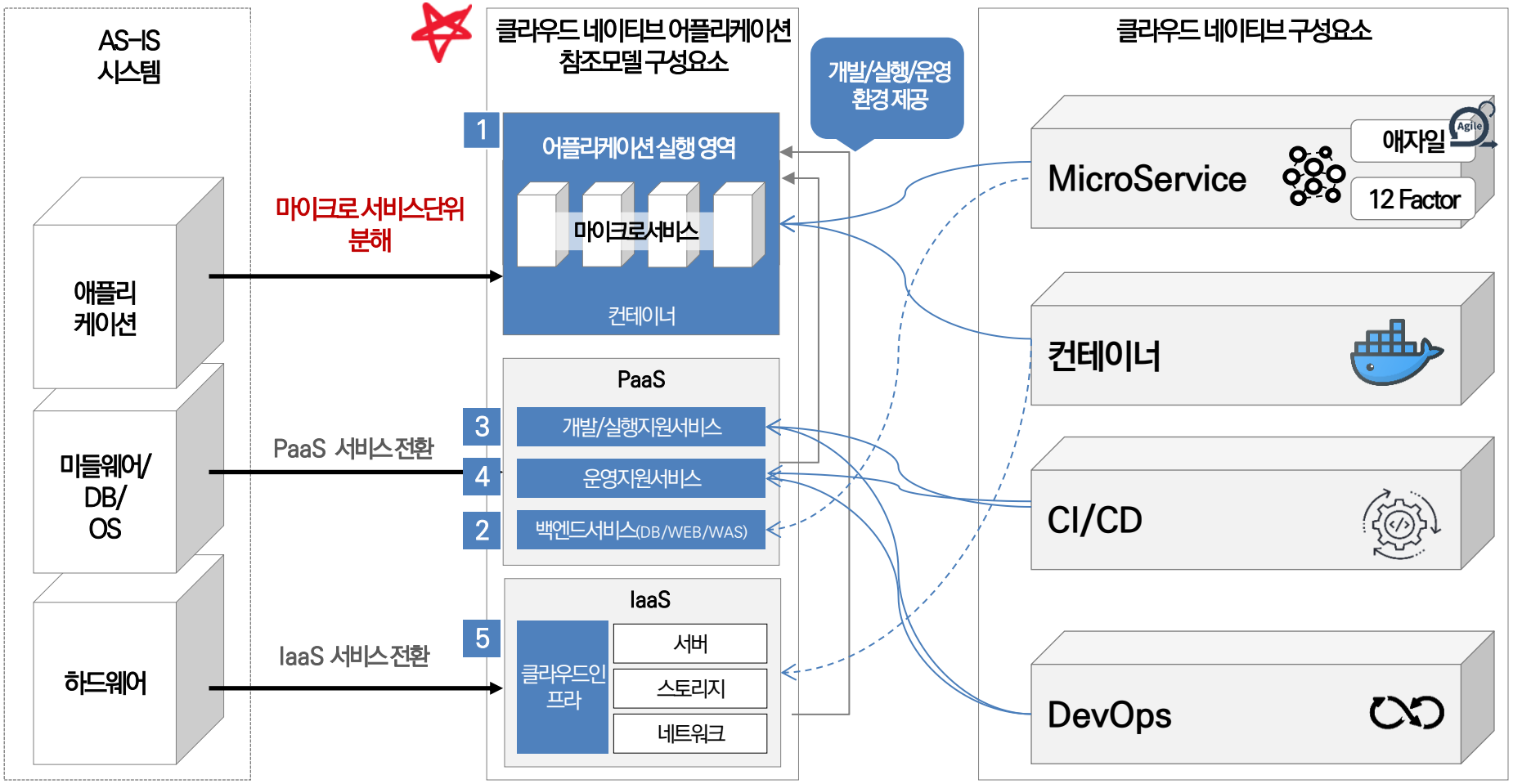
클라우드 네이티브 한눈에 알아 보기

2021년



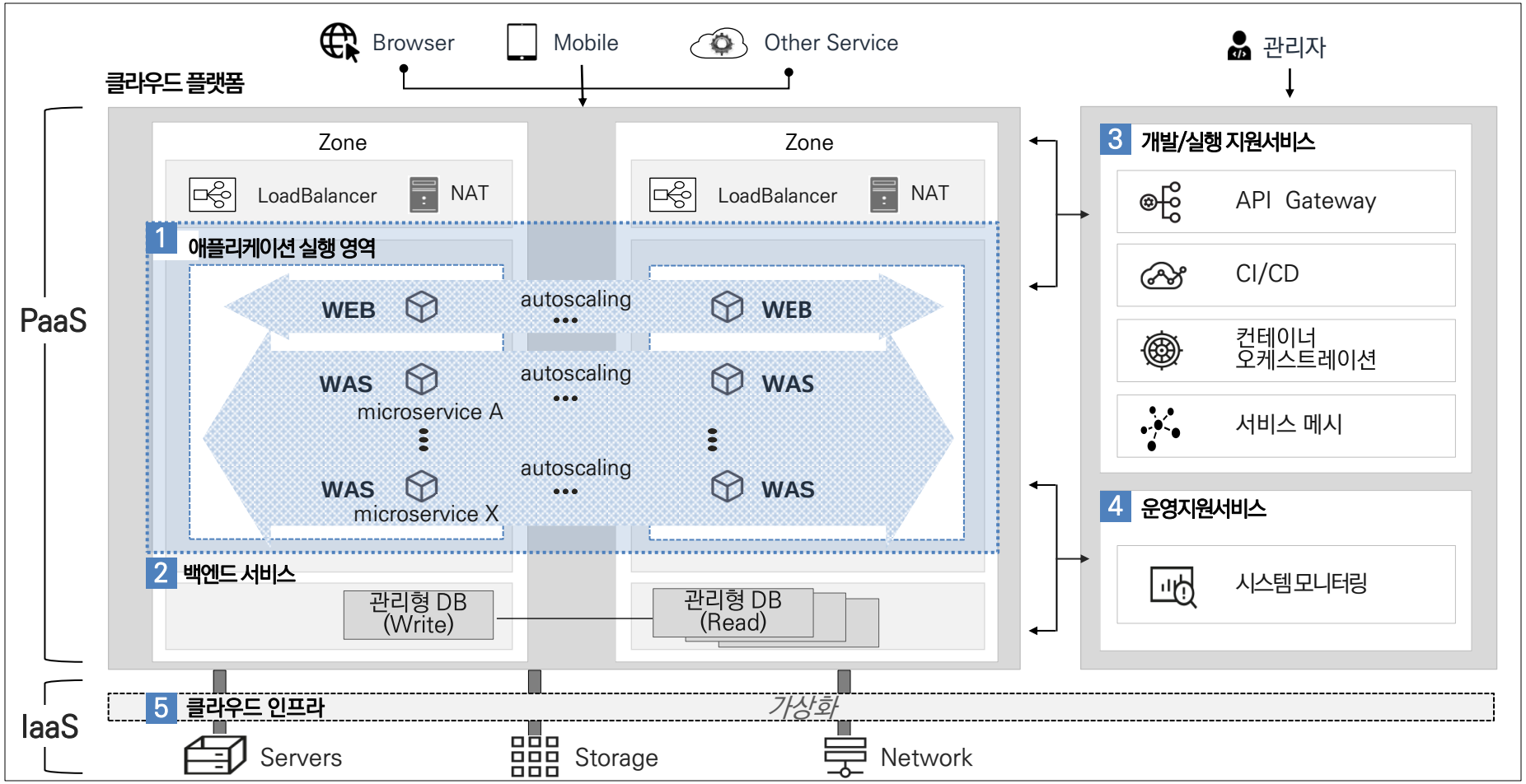
참조모델 구성요소

선진사례를 통하여 아키텍처 참조모델 구성요소를 정립합니다.



클라우드 네이티브 애플리케이션 아키텍처 참조모델

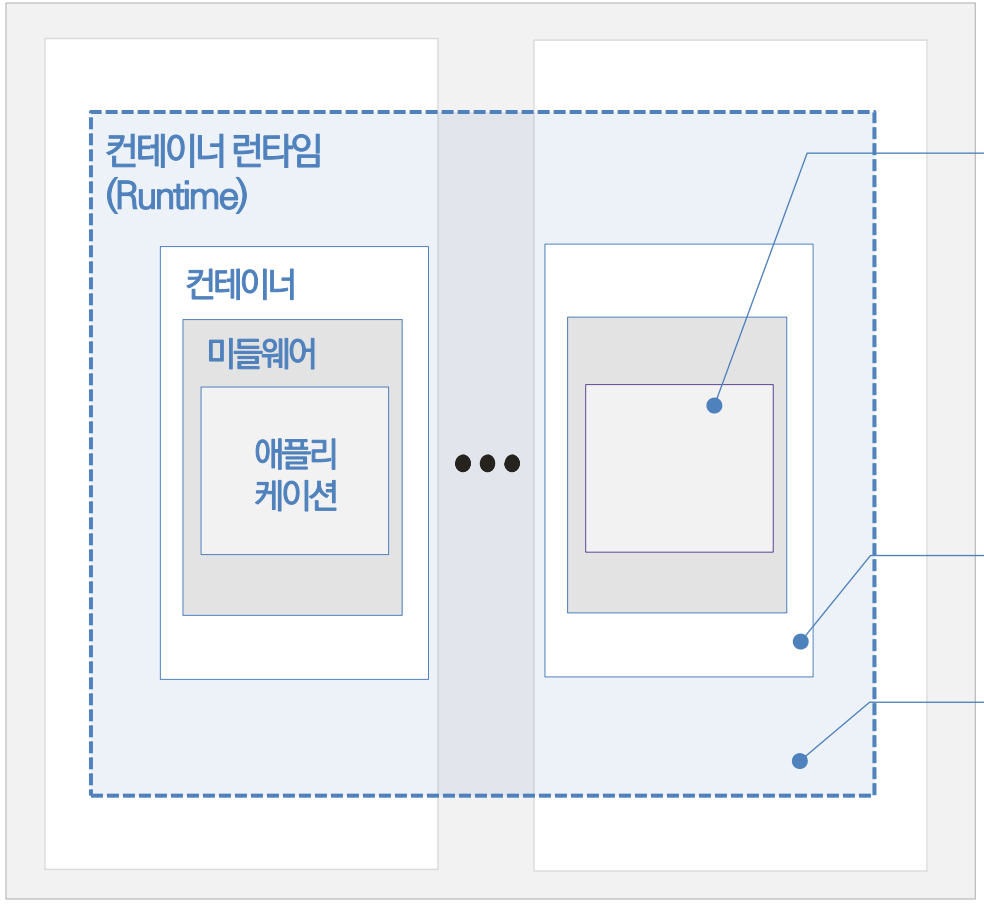
클라우드 네이티브 애플리케이션 아키텍처 참조모델 및 구성영역을 체계화합니다.



애플리케이션 실행 영역

1 아키텍처 참조모델의 애플리케이션 실행 영역입니다.

애플리케이션 실행 영역은 클라우드 네이티브 애플리케이션이 컨테이너 단위로 배치, 실행되는 영역으로
클라우드 네이티브 애플리케이션과 실행단위인 컨테이너 및 컨테이너 구동을 위한 컨테이너 런타임으로 구성됨



애플리케이션 실행 영역 구성

클라우드 네이티브 애플리케이션

- 미들웨어 (Web Server, WAS)를 포함하여 배포하여 개별적 실행 가능
- 배포대상에 따라 아래의 2가지 컨테이너로 구성
  - Web Server : 이미지 등 정적 컨텐츠 서비스
  - WAS : 마이크로서비스 단위 애플리케이션 서비스

컨테이너(Container)

- 클라우드 네이티브 애플리케이션 배포 및 실행 단위

컨테이너 런타임(Container Runtime)

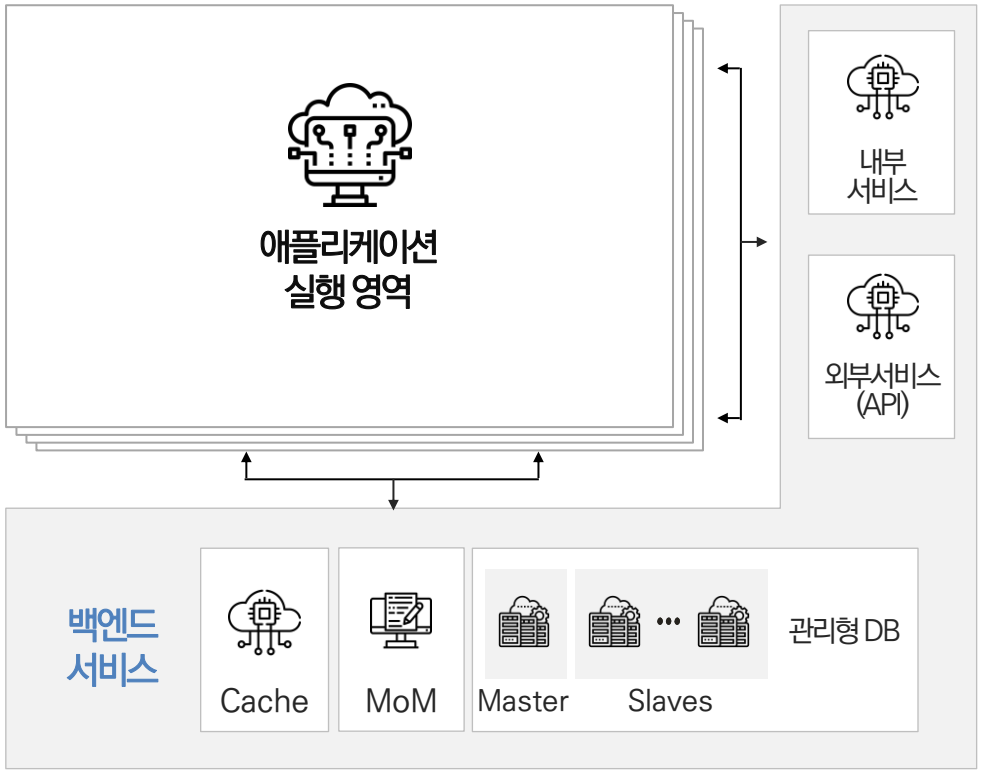
- 클라우드 실행환경 제공

  예) 도커, 쿠버네티스 등

백앤드 서비스 영역

2 아키텍처 참조모델의 백앤드 서비스 영역입니다.

클라우드 네이티브 애플리케이션을 실행하기 위해 네트워크로 연결된 모든 리소스를 백엔드 서비스라고 하며,
클라우드 플랫폼에서 제공하는 서비스, 외부 연계 서비스 및 직접 구축한 서비스를 모두 포함함



백엔드 서비스 종류

DB	• 클라우드 플랫폼에서 제공 및 관리되는 DB 서비스
Cache	• 세션관리, 성능향상 등의 용도를 위한 메모리 캐싱 서비스
MoM	• 비동기 메시징(Messaging) 서비스
외부 서비스	• 인증, 간편결제, OpenAPI 형태 외부 연계 서비스
내부 서비스	• AI, 빅데이터, 챗봇 등의 클라우드 플랫폼 제공 서비스

클라우드 네이티브 애플리케이션을 수정하지 않고 다른 백엔드 서비스로 전환이 가능하도록 구성되어야 함

개발·실행 지원서비스 영역

3 아키텍처 참조모델의 개발·실행 지원서비스 영역입니다.

클라우드 네이티브 애플리케이션을 실 환경에서 효율적으로 배포하고 안정적으로 운영하기 위해서는
클라우드 네이티브 애플리케이션에 최적화된 개발 및 실행환경을 구성해야 함

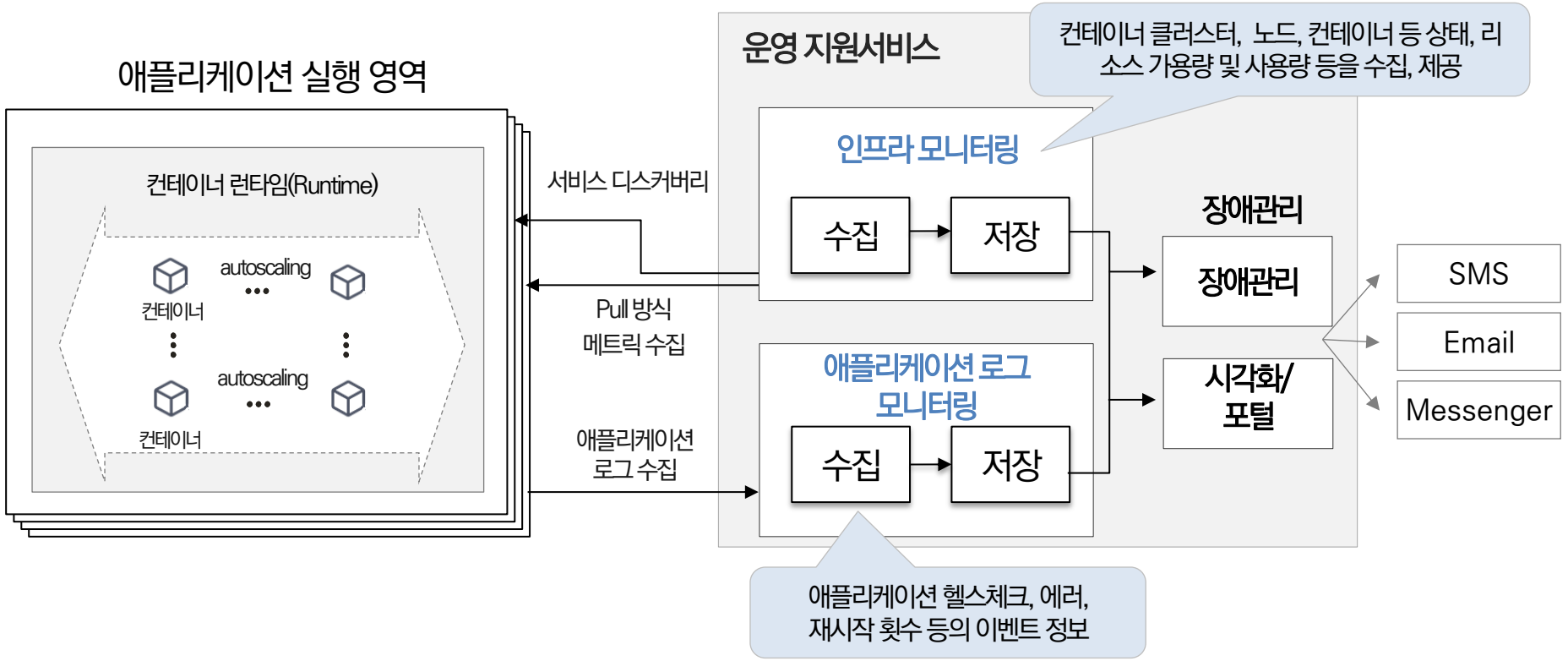
구분	지원서비스	서비스 내용	주요 솔루션
실행 환경	API Gateway	API 형태의 서비스를 제공하는 마이크로서비스 앞단에서 앤드포인트를 단일화하여 외부 사용자에 제공 API에 대한 인증과 인가 및 여러 서버로 라우팅 하는 기능 등을 담당함	Spring Cloud Gateway, zuul, kong 등
	컨테이너 오케스트레이션	여러 서버에 걸쳐 다수의 컨테이너에 대한 배포, 컨테이너 단위의 Auto scaling, 자원할당 , 장애 복구 및 모니터링 등의 운영 자동화를 제공하는 서비스	Kubernetes, Cloud Foundry, Docker Swarm 등
	서비스 메시	대규모의 마이크로서비스를 실시간으로 제어, 관리서비스 간에 통신을 하기 위해 Service discovery, 로드밸런싱, 장애 제어, Config 관리 등 제공	Spring Cloud, Istio 등
개발 환경	CI/CD	개발, 테스트, 배포 프로세스에 대한 도구 기반 자동화 및 모니터링 제공 CI(Continuous Integration)는 코드 변경 사항이 지속적으로 통합(Continuous Integration)하는 것이고 CD(Continuous Delivery)는 지속적인 서비스 제공(Continuous Delivery)을 의미	Jenkins, bamboo, git, svn 등

개발 및 실행환경 구성을 위해서 지원서비스의 활용도가 높아지고 있으며
1) 클라우드 플랫폼에서 제공되는 서비스를 이용하거나 2) 오픈소스 및 상용 SW를 활용하여 자체 구축이 가능함

운영 지원서비스 영역

4 아키텍처 참조모델의 운영 지원서비스 영역입니다.

운영 지원서비스는 클라우드 네이티브 애플리케이션 및 인프라 리소스 (CPU, Memory 등)에 대한
로그 및 사용데이터를 수집, 분석, 시각화 등 애플리케이션 전반의 모니터링 기능을 제공함



클라우드 네이티브 애플리케이션은 마이크서비스 단위로 작아지고 OS 수준에서 가상화 된 환경에서 모니터링 대상 컨테이너 또한 동적으로 생성되므로 기존 모니터링과 다르게 서비스 디스커버리 및 Pull 방식의 데이터 수집이 적용됨

클라우드 인프라 영역

5 아키텍처 참조모델의 클라우드 인프라 영역입니다.

클라우드 네이티브 환경에서 클라우드 인프라 위에
컨테이너와 같은 경량화된 가상화 기술 및 네트워크 및 스토리지 가상화를 적용하여 클러스터를 구축함

서버, 스토리지, 네트워크 장비
등과 같은 인프라 구성요소로
클라우드의 IaaS 에 해당하며,
클라우드 네이티브 애플리케이션은
컨테이너 기반의 실행환경 외에도
마이크로서비스 구조의 애플리케이션을
개발, 운영할 수 있는
클라우드 플랫폼 서비스(PaaS)가
갖춰진 클라우드 환경이 필요함

구성요소 , 방법론 및 원칙 등에 관련된 용어설명

- **MSA(Micro Service Architecture)** : 하나로 구성된 어플리케이션을 여러 개의 작고 느슨한 형태의 서비스로 쪼개어 **독립적으로 서비스하고 배포할 수 있도록 구성하는 아키텍처**

- **컨테이너** : 호스트 서버의 운영체제 수준의 **경량화 된 가상화 기술** 및 그 결과물인 격리된 인스턴스(빠른 수평적 확장)

- **도커(Docker)** : 리눅스 컨테이너 기술을 적용한 가장 대표적인 컨테이너 제공 솔루션

- **쿠버네티스** : 컨테이너 오케스트레이션 도구들이 출현하였으며 대표적인 도구

- **DevOps** : 개발과 운영 프로세스의 통합 및 자동화를 통해 신속한 서비스를 제공하도록 지원하는 **통합프로세스, 조직문화, 도구 등을 포함한 체계**

- **DevOps 툴체인(Tool Chain)** : DevOps를 효과적으로 적용하기 위해서 DevOps 전 프로세스를 여러 가지 도구(Tool)로 연결하여 자동화하는 과정

- **CI/CD** : 반복적인 개발, 테스트, 배포 과정에 대한 자동화와 모니터링을 제공하는 **도구기반 프로세스**

- **애자일(Agile)** : 사용자의 요구사항을 이해, 신속하게 반영하기 위해 기획, 설계 과정에 많은 시간과 노력을 기울이지 않고 빠르게 프로토타입을 개발하여 사용자의 피드백과 방향성을 확인하고 지속적인 **개선 과정을 짧은 주기로 반복하는 개발방법론**

- **12 Factor** : 클라우드 환경에 최적화되어 클라우드 모든 기능을 제대로 사용할 수 있게 위해서 지켜야 할 규칙

12 Factor

감사합니다.

컨테이너 정의 및 구조

컨테이너

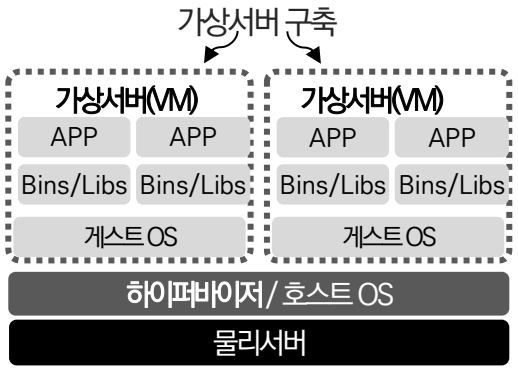
컨테이너는 어떤 환경에서나 실행하기 위해 필요한 애플리케이션 코드, 런타임 모듈, 라이브러리 등의 모든 요소를 포함하는 경량화된 소프트웨어 패키지임

기존 방식과 컨테이너 기술 비교

베어메탈

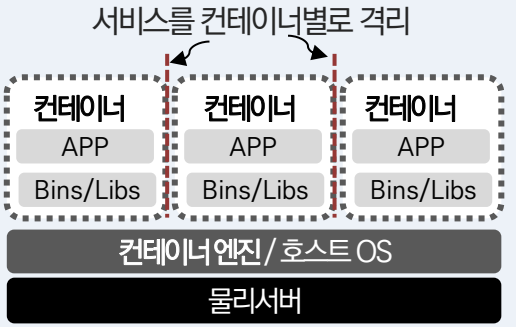
- 물리서버 전체를 하나의 OS 환경으로 사용

서버 가상화(HW 레벨)



- 물리서버 자원(CPU, 메모리 등)을 분할하여 논리적인 가상서버를 생성
- 복수의 OS 환경 사용
- IaaS (Infrastructure as a Service)

컨테이너 가상화(OS 레벨)



- 물리서버 자원(CPU, 메모리 등)을 분할하여 논리적인 컨테이너를 생성
- 하나의 OS 환경에 복수 실행 환경 사용
- CaaS, PaaS, FaaS

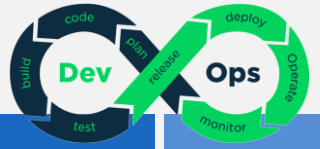
- 컨테이너 가상화는 기존의 서버 가상화 방식과 달리 컨테이너별로 실행환경을 격리하여 가볍고 빠른 서버환경을 제공하며, 하나의 물리서버에 많은 수의 컨테이너를 필요 시 즉각 생성할 수 있음
- 좀 더 효율적이고 경량화된(light weight) 애플리케이션 실행 환경에 대한 요구가 증가함에 따라 가상화 방식은 기존의 가상화 기술인 하이퍼바이저 방식에서 컨테이너 엔진 기반 가상화 방식으로 전환되고 있음

DevOps 개요

DevOps

- DevOps는 개발(Development)과 운영(Operatios)의 합성어로, 개발과 운영 프로세의 통합 및 자동화를 통해 신속한 서비스를 제공하도록 지원하는 **통합프로세스, 도구, 조직문화** 등을 포함한 체계
- **개발팀과 운영팀의 협업이 가능해져 업무 병목 구간을 최소화하고, 애플리케이션을 신속하게 개발, 배포할 수 있음**

DevOps = 개발과 운영의 협업체계



	개발단계(Dev)					운영 단계(Ops)		
프로세스	계획	개발 (코딩)	빌드	테스트	릴리즈	배포	운영	모니터링
도구 (툴체인)	Jira Software	Confluence, git	Maven, Gradle	JUnit	Jenkins	docker, Jenkins	kubernetes, mantis BUG TRACKER	splunk>, Nagios
조직	기획 담당	개발자	테스트 담당	개발자		운영자		
	사용자 (요청자)	변경관리 담당	사용자 (요청자)	변경심의위원회		작업심의위원회		

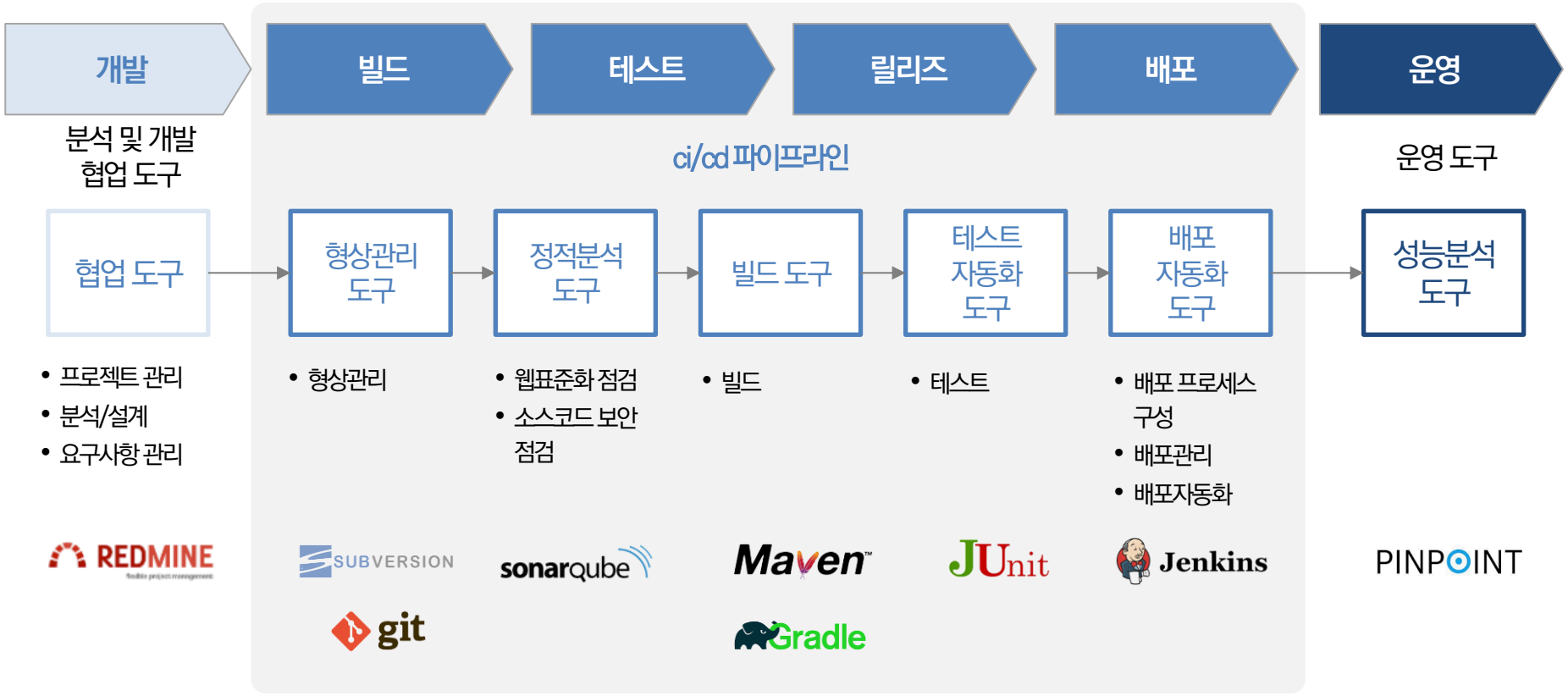
CI/CD 영역

CI/CD

CI/CD

CI/CD 파이프라인은 개발 이후 빌드, 테스트, 릴리즈, 배포 과정에 자동화 도구를 활용하여 애플리케이션 배포주기를 단축하여, 서비스 요구에 신속하게 대응하도록 지원함

CI/CD 파이프라인 예시



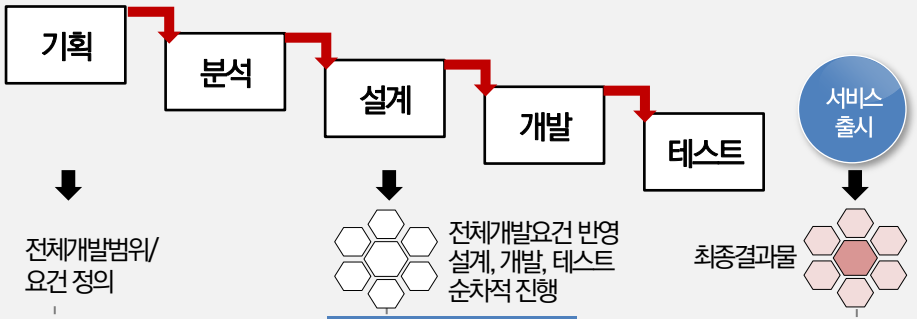
애자일 방법론

애자일 방법론

폭포수 방법론의 개발 공정은 기획, 분석, 설계, 개발, 테스트 단계가 위에서 아래로 순차적으로 진행되며, 애자일 방법론은 각 개발 공정을 명확하게 구분하지 않고 각 단계를 반복적으로 수행하면서 요구사항을 추가하거나 수정하면서 개발을 수행함

폭포수 방법론과 애자일 방법론 비교

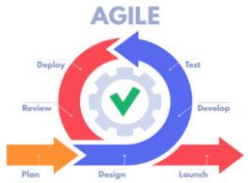
폭포수 방법론

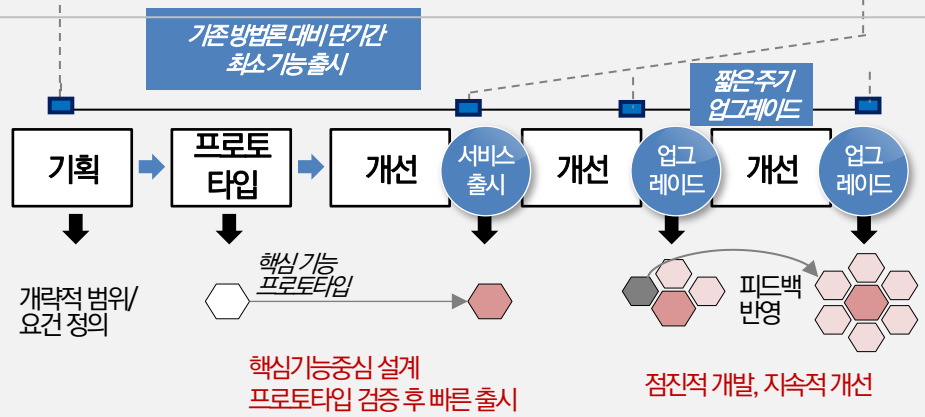


폭포수 방법론의 문제점

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

애자일 방법론





애자일 방법론 개발 방식

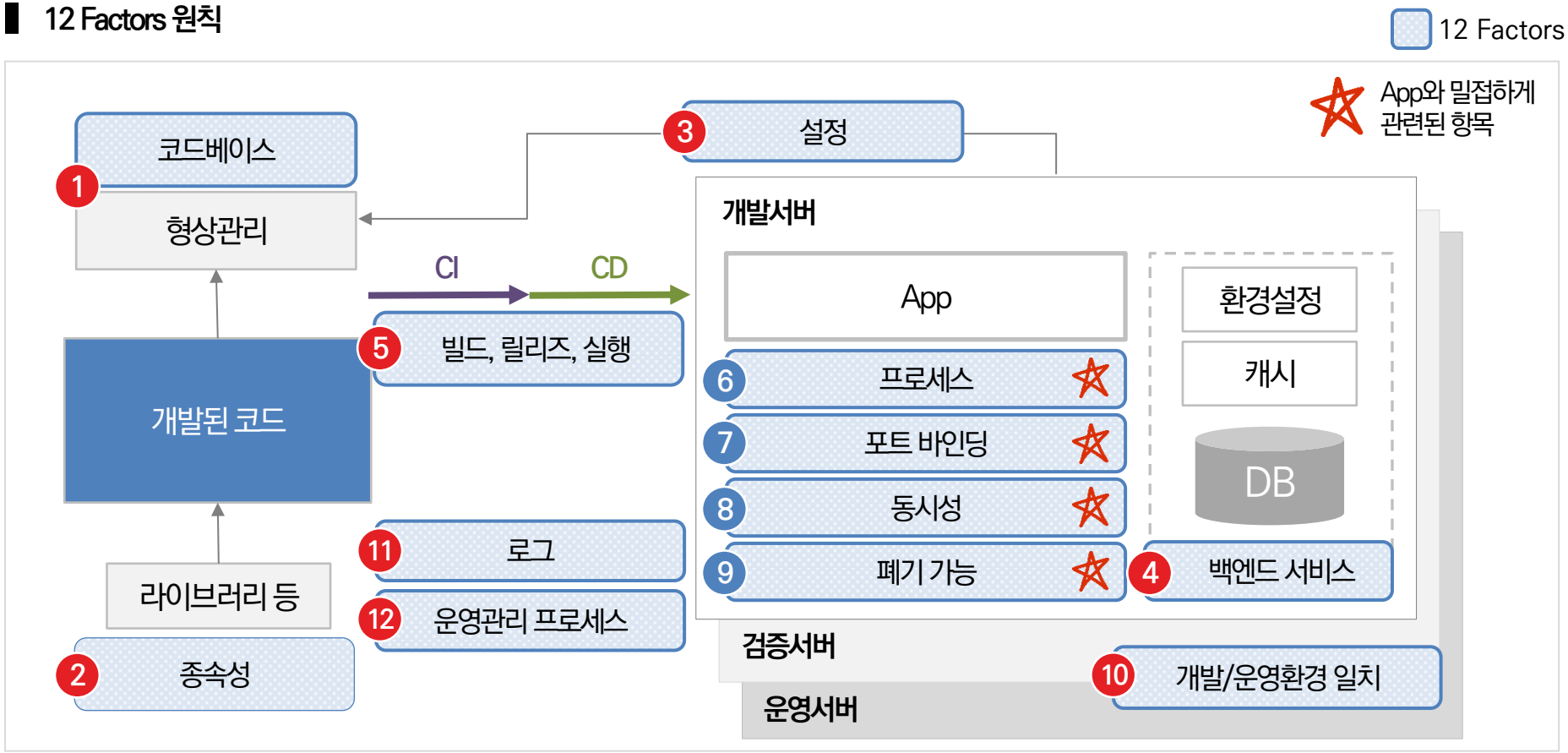
- 과도한 기획 지양
- 핵심기능에 집중 빠른 출시 목표
- 고객가치가 높은 기능 우선순위로 반복적으로 개선

12 Factors

12 Factors

12 Factors App 원칙은 2012년 Heroku에서 일하던 개발자들이 클라우드에 적합한 SaaS 애플리케이션 개발과 배포 방법에 맞는 12가지 원칙을 개념화한 것으로 클라우드 네이티브 환경에 적합하게 적용되어야 할 부분들을 명확하게 정의하고 있음

12 Factors 원칙



원칙 설명

원칙 설명

12 Factors App 원칙은 코드베이스, 종속성, 설정, 백엔드서비스, 빌드/릴리즈/실행, 프로세스, 포트바인딩, 동시성, 폐기가능, 개발/운영 환경 일치, 로그, 운영관리 프로세스로 정의되어 있으며, 해당 내용 설명은 다음과 같음

12 Factors 원칙 설명

1	**코드베이스**[1] (Codebase)	• 하나의 코드베이스(소스코드)로 버전관리하여, 이를 여러 곳에 배포
2	**종속성** (Dependencies)	• 패키지, 라이브러리 등 종속이 필요한 경우 명시적으로 선언하고 분리시켜 실행환경 종속성 제거
3	**설정** (Configuration)	• 소스 코드와 설정 정보를 분리, 실행시 코드에서 읽어서 사용
4	**백엔드서비스** (Backing Services)	• 애플리케이션 작동에 필요한 서비스(DB, 메시지큐, 캐시등)를 연결된 리소스로 취급하여 연결/분리가 용이
5	**빌드, 릴리즈, 실행** (Build, Release, Run)	• 빌드, 릴리즈, 실행단계를 엄격히 분리
6	**프로세스** (Processes)	• 애플리케이션 실행시 하나 혹은 여러 개의 stateless 프로세스로 실행
7	**포트바인딩** (Port-binding)	• 애플리케이션은 독립적이며, http 같은 포트 바인딩을 통해서 외부에 서비스 제공
8	**동시성** (Concurrency)	• 애플리케이션을 수평적으로 확장하며, 무상태(Stateless) 특성이 이런 확장을 단순하게 만듦
9	**폐기가능** (Disposability)	• 빠른 시작과 그레이스풀 셧다운(Graceful shutdown)을 통한 안정성 극대화
10	**개발/운영 환경 일치** (Dev/Prod Parity)	• 개발계/검증계/운영계 환경을 가능한 비슷하게 유지함으로써 지속적인 개발/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