효율적인 오픈소스 적용을 위한

IBM 플랫폼 활용

최주헌

IBM Cloud Solution Architect

juheon.choi@ibm.com

Agenda

1. 앱 현대화 이해와 방향성

2. 앱 현대화 오픈소스 적용 방안

3. 효율적인 오픈소스 적용을 위한 IBM 플랫폼 활용 방안

4. IBM Cloud 플랫폼을 활용한 데모

1

앱 현대화(Modernization)

이해와 방향성

Digital transformation은 더 나은 비즈니스 가치를 실현

9.5x

프로덕트 품질 개선

5x

비용 절감 개선

1.7x

사용자/근로자의 만족도 개선

1.6x

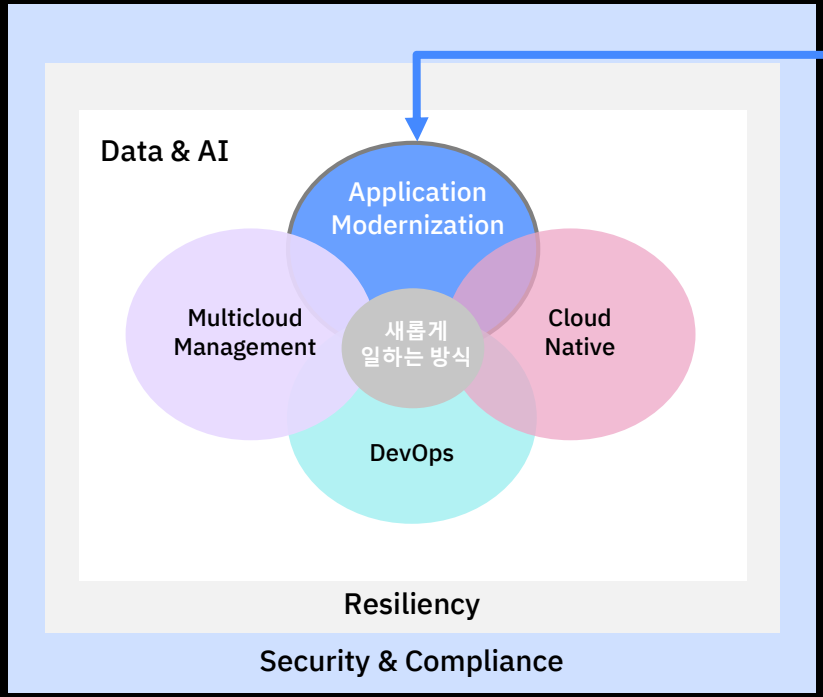
고객 유지 개선

현대화는 비즈니스 가치를 개선시킬 핵심 요인이다.

출처 : Forrester Consulting

App Modernization(현대화) 이란?

클라우드로 전환하는 광범위한 여정의 **시작점**으로, 애플리케이션을 **새로운 플랫폼**으로 이관하고 **아키텍처를 변경** 및 **DevOps**를 적용하여 **새로운 기능**을 빠르고 점진적으로 적용하는 **과정**



주요 변화:

- **컨테이너 플랫폼** 이관
- **마이크로 서비스 아키텍처(MSA)**로 변경
- **DevOps** 적용

예 – Provisioning

개발팀은 애플리케이션을 향상시키기 위해 새로운 기능이 적용된 환경이 필요하다.

프로비저닝 항목	Current (VMs)	After Containers
Database – provision a standalone database with a set of tables	2 weeks →	20 minutes
WebSphere	2-3 weeks →	60 seconds
MQ	2 weeks →	20 minutes
Development Pipeline	1-2 weeks →	2 hours

Before

effort: 7 weeks (35 days, 280 hours)
linear: 3 weeks (15 days, 120 hours)

After 2.41 hours (일부는 수동으로 인증한다고 가정)

ΔP = 117.3 시간 : 프로비저닝 인스턴스 당 절약된 작업 시간

앱 현대화(Modernization) 진행 과정

비즈니스, 고객, 기술 관점에 기존 자산을 평가하고 우선순위를 식별하여 전환유형별로 점진적인 현대화를 진행

2. 고객 관점

+ 고객중심
+ Data / insights
+ Self service
+ 데이터 처리에 대한 신뢰 구현

1. 비즈니스 관점

+ 비즈니스 우선순위
+ 매출 향상
+ 혁신과 비용 지불의 균형

최적화된
현대화
패턴

3. 기술 관점

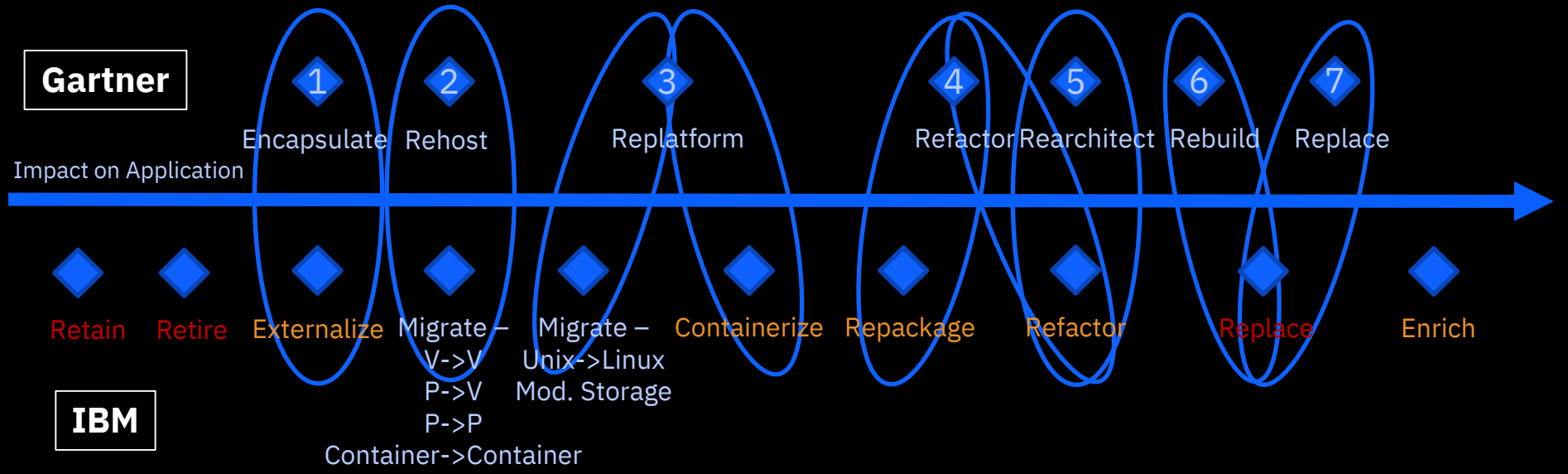
+ 프로세스 자동화
+ 최적화 기술을 적용한 프로세스 재구성

Virtual > Virtual
Physical > Virtual
Physical > Physical
Modernize Storage
Unix > Linux
Container > Container

Application
Migration & Modernization
+
Replace, Retain, Retire

Containerize
Repackage
Refactor
Externalize
Enrich

가트너(Gartner) 7 Rs 과 IBM의 Application Cloud Journeys 의 매핑



Application Modernization 유형

유형	설명	대상	기술 요소 변화	노력
Containerize	- 최소한의 노력으로 **컨테이너로 전환** - WebSphere, JBoss, WebLogic, Tomcat 를 컨테이너에 최적화된 Liberty로 전환	Kubernetes	• 처음에는 단순하지만 그보다 더 높은 수준의 변환 프로세스를 가짐 • 잠재적으로 Refactor 대상	Low
Repackage	- 모놀리식 애플리케이션을 **작은 패키지로 모듈화** - 모듈화된 패키지는 별도의 Liberty 컨테이너에 배포	Kubernetes	• 다수의 컨테이너화된 컴포넌트로 분할 (War 파일) • 컴포넌트는 독립적으로 실행 • 워크로드는 잠재적 Enrich 대상 • 컴포넌트는 잠재적으로 제거 대상일 수 있음	Low
Refactor	- 서비스 메시가 적용된 **MSA**로 애플리케이션을 재설계 - 작은 서비스를 재 작성하여 기능을 분리해 내는 방식으로 시작할 수 있음 - 한번이 아니라 점진적으로 코드 수정을 수차례 진행할 수 있음	Kubernetes	• 도메인 지향 분석 • 12-factor 적용 (Cloud Native app) • 애플리케이션 사이클 주기 개선 • DevOps 문화 수용 • 업무 프로세스 변화	High
Externalize	- 애플리케이션의 **핵심 기능을 API로 제공**하여 다른 시스템이 이를 이용할 수 있도록 함 - API Connect를 이용하여 서비스를 노출할 수 있음	Any	• 레거시 컴포넌트는 중요한 기능을 포함 • 기존 레거시 인터페이스를 포함 • 잠재적 제거 대상 • API를 추상화 할 수 있는 API Connect 적용	Medium
Enrich	- 워크로드에 **새로운 가치**(예: chatbot, AI)를 추가하거나 기타 기능을 추가함 - 코드 변경이 필요함	Kubernetes	• 컨테이너화된 워크로드의 기능을 더 강화 시키기 위해 AI 분석 또는 기타 클라우드 기능의 추가가 필요	Medium

2

앱 현대화(Modernization)

오픈소스 적용 방안

시나리오

가 기업은 **인터넷 상거래 사이트** 입니다.
하루 주문 30,000건이 발생하고 한달에 90만, 1년에 약 1000만 건의 거래가 진행되고 있습니다.
주문부터 배달 까지 대부분 거래는 **1주일 이내**로 완료 됩니다.

애플리케이션은 **모놀리틱에 2중화된 서버** 그리고 **관계형 DB**로 구축 되어 있습니다.

늘어나는 거래에 **서비스는 빈번히 지연**되고 있습니다.
고객은 **과거 주문 내역 전체를 조회**하고 싶어하지만,
수억건의 데이터를 조회 서비스로 제공할 경우 **실제 거래에 영향**을 미치기 때문에 조회 서비스를 제공하지 못하는 상태입니다.

계속 증가하는 주문 건수와 조회 서비스를 처리하지 못하여 시스템 지연이 자주 발생하고 있습니다.

분석

Pain Points

- 비즈니스 변화 대응을 위한 신속한 **신규 서비스 적용**의 어려움
- 모듈의 복잡도가 높은 단일 서비스로 **변경과 확장**의 어려움
- **애플리케이션** 환경의 **확장 및 이동**이 유연하지 못함
- 관계형 DB로 인하여 **빅데이터 활용**하여 다양한 서비스와 분석의 한계
- **시스템 피크 대응 한계** 및 다운 타임에 따른 업무 피해 발생

해결방안

1. MSA 전환

모놀로틱 앱을 MS로 분리하여 손쉬운 기능 변경 및 서비스 별 스케일링 적용으로 생산성 및 자원 효율화를 향상함

2. DB 전환

거래중인 데이터는 RDB에 유지하고 거래가 완료된 데이터는 NoSQL로 이관하여 목적에 맞는 DB 구성

3. 플랫폼 전환

애플리케이션 이식성, 확장성을 고려하여 기존 호스트 방식을 컨테이너로 전환함

4. DevOps 적용

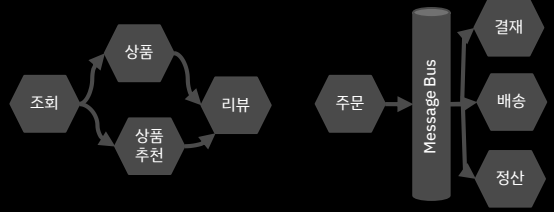
DevOps Tools Chain을 적용하여 코드 품질을 향상하고 주기적이고 반복적인 배포 과정을 자동화 함

5. 보안 및 모니터링 구축

소스코드와 이미지에 대한 보안성 검토 및 마이크로 서비스를 모니터링하기 위한 체계를 구축함

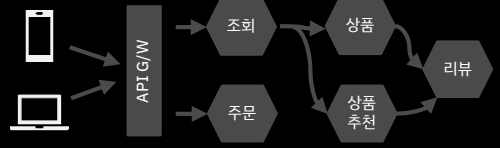
MSA 전환

method call -> API Call, 메시지 전송 변경



- 하나의 모놀리틱 시스템이 다수의 mSvc로 분리됨
- 호출 방식이 메소드 콜에서 API, 메시지 방식으로 전환
- 메시지 버스를 이용한 데이터 전달

서비스 간의 통신을 위한 Service Mesh 적용

MSA	
정의	• 서비스간 통신을 추상화, 전용 인프라 레이어
기능	• 서비스관리, 라우팅,트레이싱, 서킷브레이크 등
오픈소스	• Spring Cloud(VM), Istio(K8s)

처리 방식 분리

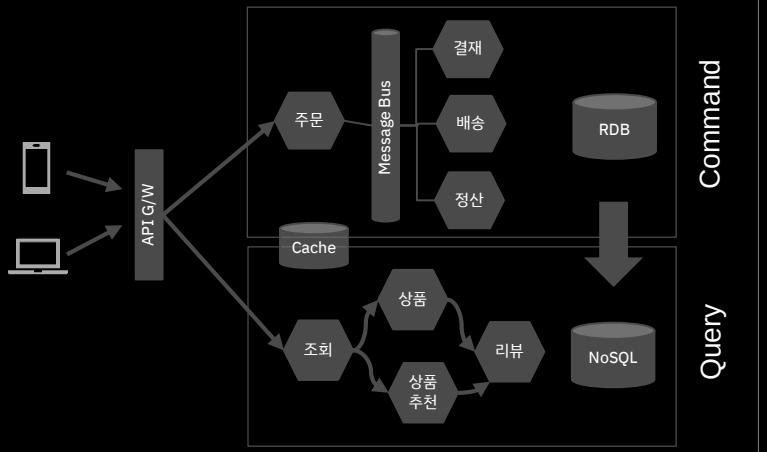
	CPU Intensive	Network Intensive
설명	• 대용량 데이터 처리 • 비즈니스 로직을 사용 • 안정성	• 사용자 요청 담당 • 빠른 요청 처리
기술	• Sync, Blocking	• Async, Non-Blocking, Websocket
사용 예	• 대용량 데이터 처리 • 백엔드 서비스 • 주요 로직 처리 서비스	• 대용량 트래픽 처리 • 프론트엔드 서비스 • 사용자의 요청, 라우팅 처리
오픈소스	• Tomcat, Undertow	• Nginx, Netty

❑ 적용 가능한 오픈소스

- 대용량 트래픽 처리 - nginx
- 대용량 분산 메세지 처리 - kafka
- 분산 네트워크 환경을 관리 - istio

DB & 플랫폼 전환

목적, 용도에 맞는 DB 활용 (예:CQRS 패턴 적용)

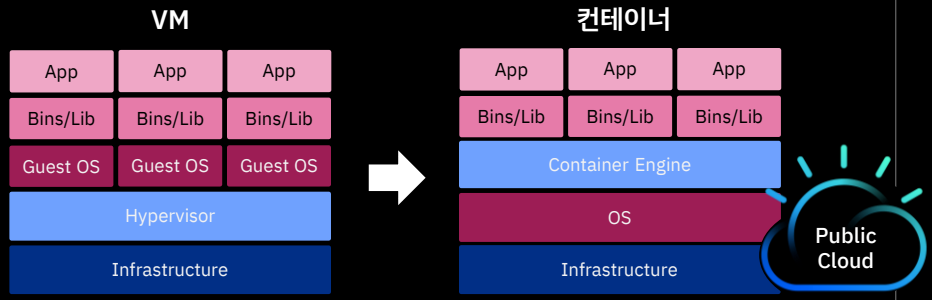


- CQRS 패턴[1] 적용으로 안정성(RDB) 과 성능(NoSQL) 을 고려한 설계
- 장애 격리로 워크로드 보호
- 쓰기 저장소(RDB) 에서 읽기 저장소(NoSQL) 로 데이터 동기화 (Eventual Consistency)

주1) CQRS(Command Query Responsibility Segregation) 패턴 이란?

CUD 작업과 R 작업 처럼 트랜잭션 특성에 따라 서비스를 분리하여 성능과 안정성을 극대화 시키는 패턴

Cloud Native에 최적화된 컨테이너 플랫폼 및 Cloud 전환



- 인프라스트럭처에 초점
- 컴퓨팅 환경을 가상화
- 서버 한대에 여러개의OS 를 구성함
- 서버 공간을 효율적으로 사용

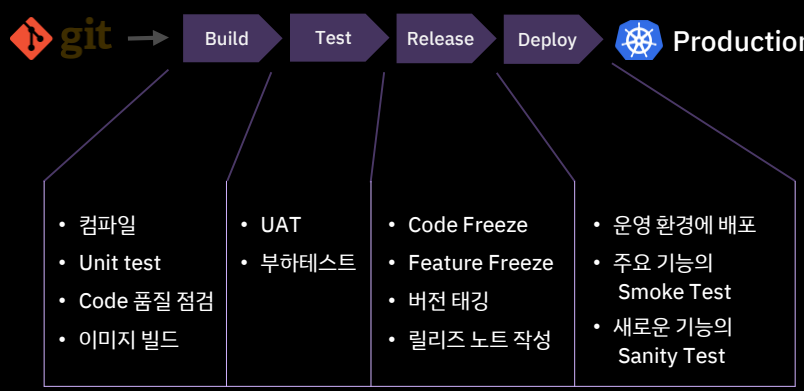
- 애플리케이션에 초점
- 애프리케이션을 가상화
- VM 보다 작고 내부 처리 프로세스가 적어 빠름
- 어디든(anywhere) 손쉬운 워크로드 이동
- MSA 분산 네트워크에 최적합

❑ 적용 가능한 오픈소스

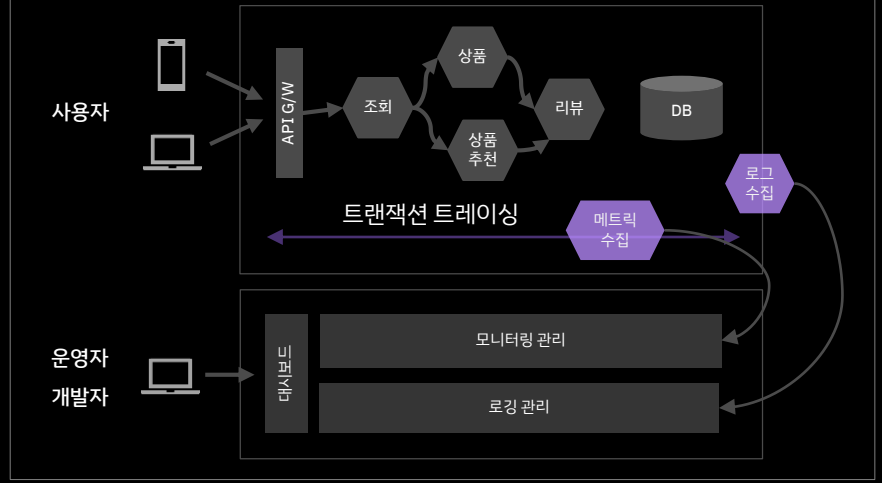
- 안전성 (CUD) 고려한 DB – Mariadb / Postgre / cubrid
- 성능 (R) 고려한 DB – MongoDB(NoSQL) / Redis (케시) / Elasticsearch (검색)
- 컨테이너 플랫폼 – Docker / Kubernetes / Openshift

DevOps, 모니터링 / 로깅

빠른 배포를 위해 DevOps 적용



MSA 환경을 위한 모니터링/로깅 도입



❑ 적용 가능한 오픈소스

- 빌드 – Jenkins / tekton
- 테스트 – Jinit (단위) / Selenium (기능테스트) / Jmeter (부하)

❑ 적용 가능한 오픈소스

- 로깅 – Elasticsearch / Logstash / Kibana
- 모니터링 – Zipkin / Prometheus / kiali

보안

컨테이너 환경에 최적화된 보안 구성

코드 보안

- 오픈소스 라이브러리 코드 취약성 점검
- 버그 및 개인 키 인증 정보 공개 스캔
- 빌드 시 코드 보안 품질 점검
- 코드 저장소의 정기적인 스캔
- 라이선스 검출 및 잠재적 충돌 인식

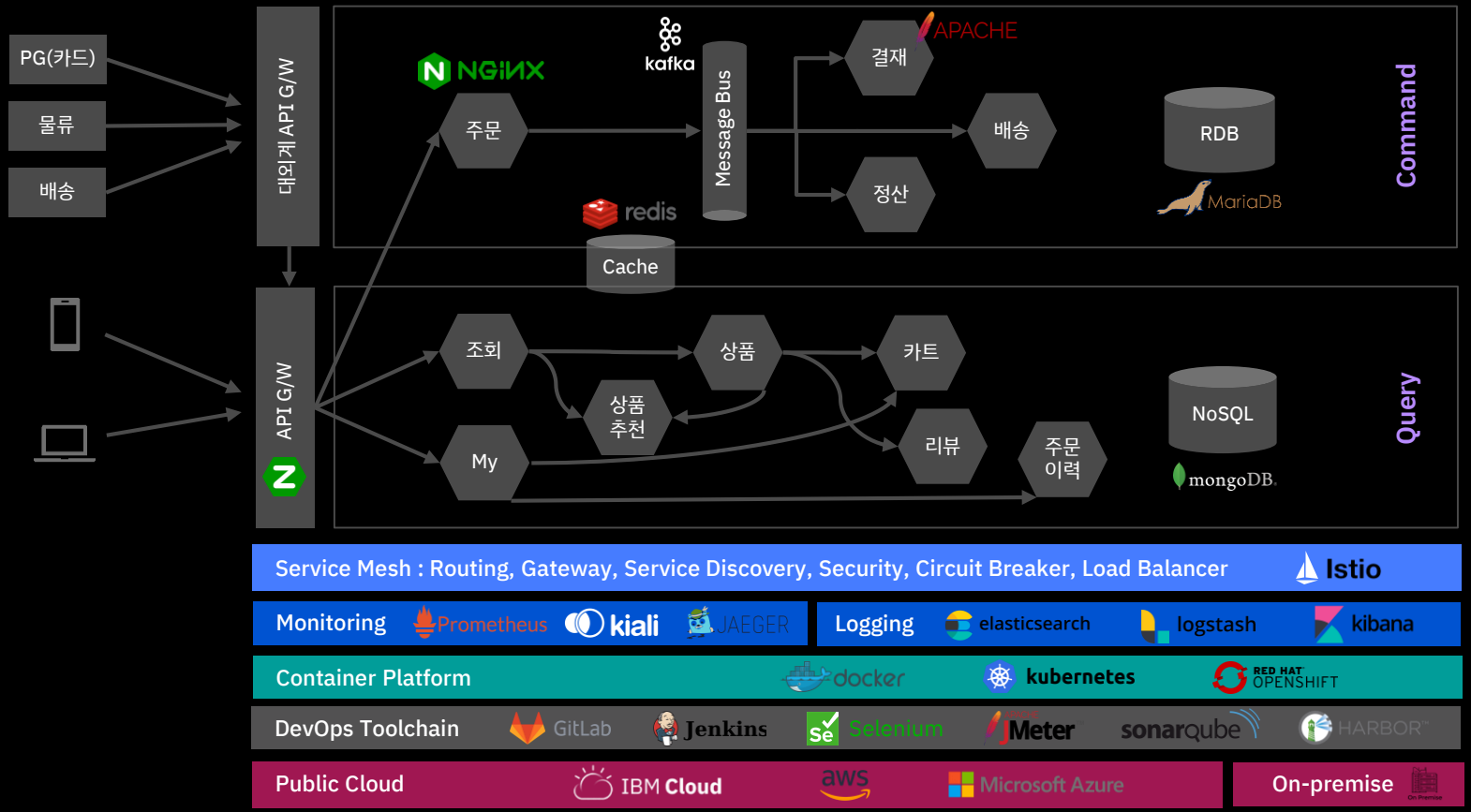
이미지 보안

- 검증된 베이스 이미지 사용
- 베이스 이미지의 취약점, 구성결함, 악성코드 스캔
- 필요한 권한으로 컨테이너 사용 (Privileged container 권한 방지)
- container breakout, 호스트 공격
- 저장소 인증/권한 취약점에 따른 이미지 변경
- 이미지 취약점 스캔 및 주기적인 업데이트
- 보안이 강화된 런타임 환경 사용(rkt, gvisor, kata container)

❑ 적용 가능한 오픈소스

- **코드 보안 – SonarQube**
- **이미지 보안 – Horber 저장소 / Anchore 스캐닝 도구**

오픈소스를 활용한 아키텍처



Challenge :

오픈소스기술

관리

보안

…

Istio
Postgre
MongoDB
Docker
Kubernetes
Openshift
Redis
MariaDB
Nginx
Tomcat
Kafka
Anchore
Apache
Sonarcube
Prometheus
Harbor
Kiali
JMeter
Junit
Elasticsearch
Jaeger
Selenium
Kibana
Logstash
Gitlab
Jenkins
Logstash
Kibana

Complexity of hybrid and multi-cloud

3

효율적인 오픈소스 적용을 위한

IBM 플랫폼 활용 방안

IBM 오픈소스 기술지원 현황

" IBM은 1999년부터 아파치 오픈소스 재단을 시작으로 약 4조원을 오픈소스 개발 생태계 구축에 투자했다.
이제는 유연성과 빠른 구축의 장점이 있는 오픈 소스 기반 시스템은 피할 수 없는 숙명이며, 우리는 그러한 개발자를 적극 지원한다. "

- 크리스 페리스, IBM CTO, Open Technology

오픈소스 기술에 대한 투자	오픈소스 Ecosystem 참여	오픈소스를 위한 기술 지원 서비스
• **40억 달러** 규모 튜자 • **70,000명**의 오픈소스 엔지니어 보유 • 매년 1,000여명의 IBM 개발자들이 오픈소스 Innovation과 Code를 Contribute하고 있음	• **100**개 이상 **오픈소스 단체**에서 활동 • Foundation : Apache Foundation, Cloud Foundry, Cloud Native Linux Foundation, Open Container Initiative 등 • Project : Knative, Kubernetes, Kubeflow, TensorFlow, Zowe 등	• **오픈소스 기술지원 체계**를 제공 • 커뮤니티가 제공하지 못하는 **기술지원 서비스** 제공 • 주요 오픈소스에 대한 Global Alliance 를 통하여 경쟁력 있는 가격의 **서브스크립션 제공** 및 기술 지원

IBM Cloud

가장 개방적이고 안전한, 기업 비즈니스용 퍼블릭 클라우드

Open Innovation

- Cloud Native Open Source Project의 주도 및 기여
- K8s on IBM Cloud 1K+ 기업 고객 지원, 21K+ clusters 운영
- 고객의 Open Source 기반 기술 경험 연장 및 최신 기술 적용
- Cloud Services anywhere

Innovation ↑ Cost ↓

Security leadership

- 데이터 암호화에 대한 업계 최고 규정 준수 (FIPS 140-2 Level4)
- IBM Security 오퍼링 통합을 통한 End-to-End 보안 서비스 제공
- 컨테이너 보안의 업계 리더

Risk ↓ Agility ↑

Enterprise grade

- #1. VMware public cloud (2,000곳 이상 고객 보유)
- Power AIX, IBM i, Z 기반 미션 크리티컬한 워크로드 클라우드전환 지원
- 가장 광범위한 컴퓨팅 포트폴리오 : 베어메탈, VPC, 가상서버, 컨테이너 및 서버리스 서비스 등

Investment protection ↑

Highest level of encryption
FIPS 140-2 Level 4

Isolation for cloud native
ROKS and containers on bare metal

No data egress charges with Cloud Databases
No vendor lock in and lower TCO

No-cost bandwidth between regions
Significantly lower TCO

Enhanced availability SLAs
HA: 99.99%, Non-HA: 99.9%

Higher SLA payouts versus market
25% of monthly at 60 minutes

Audit transparency to bare metal
Traceable serial number compliance

Full control to bare-metal level
Full admin control of compute

Good Design Award for VPC

Good Design Award for API Connect

Customer Choice Award for Cloud IaaS

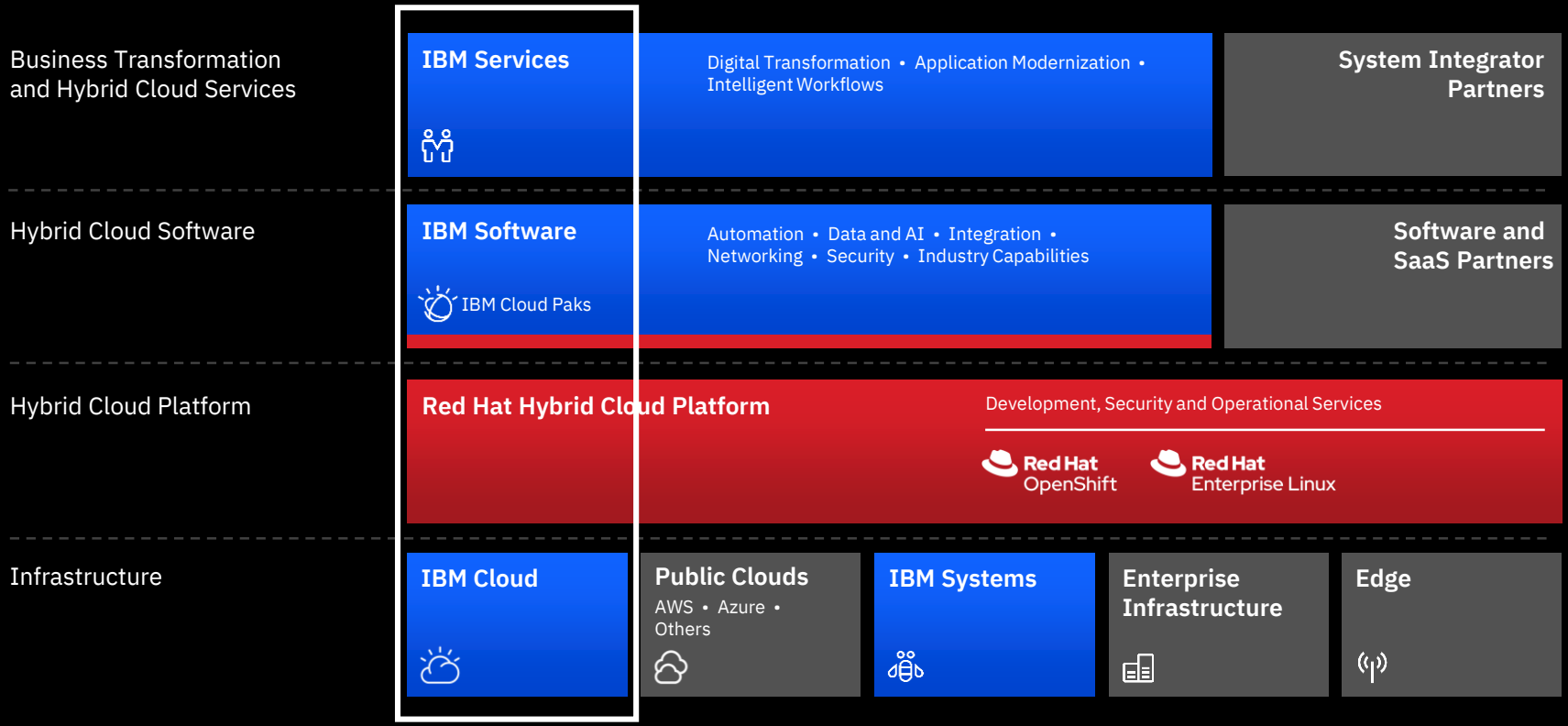
Stratus Award for User Experience

Gold Indigo Award for IBM Cloud App ID and Assist Me

Build Your Own Platform by

Collaboration with IBM

IBM Cloud 는 Enterprise DT 플랫폼을 위한 최적 구성 지원



Build Your Own Platform on

IBM Cloud

엔터프라이즈의 성공적인 클라우드 전환 여정을 지원하는 IBM Cloud

- VMware, SAP & Power 전환 오퍼링
- 컨테이너 서비스
- IBM Cloud에서 제공되는 Red Hat OpenShift

- IBM Cloud Paks
- 클라우드 카탈로그
- 클라우드 네이티브 서비스
- IBM Cloud Satellite

IBM

현재 애플리케이션 유형

47% 커스텀

38% 패키지 애플리케이션

15% Cloud + SaaS

클라우드 결정 프레임워크

- 현 상황은?
- 2-3년 후 목표치, 도달점은?
- 가치 창출 요소는?

수행방법

22% 현행 유지/변경무

23% 리프트앤 시프트

31% 컨테이너화

8% 리팩토링

18% 사용중단

종착점: 하이브리드 멀티클라우드


퍼블릭 클라우드


프라이빗 클라우드

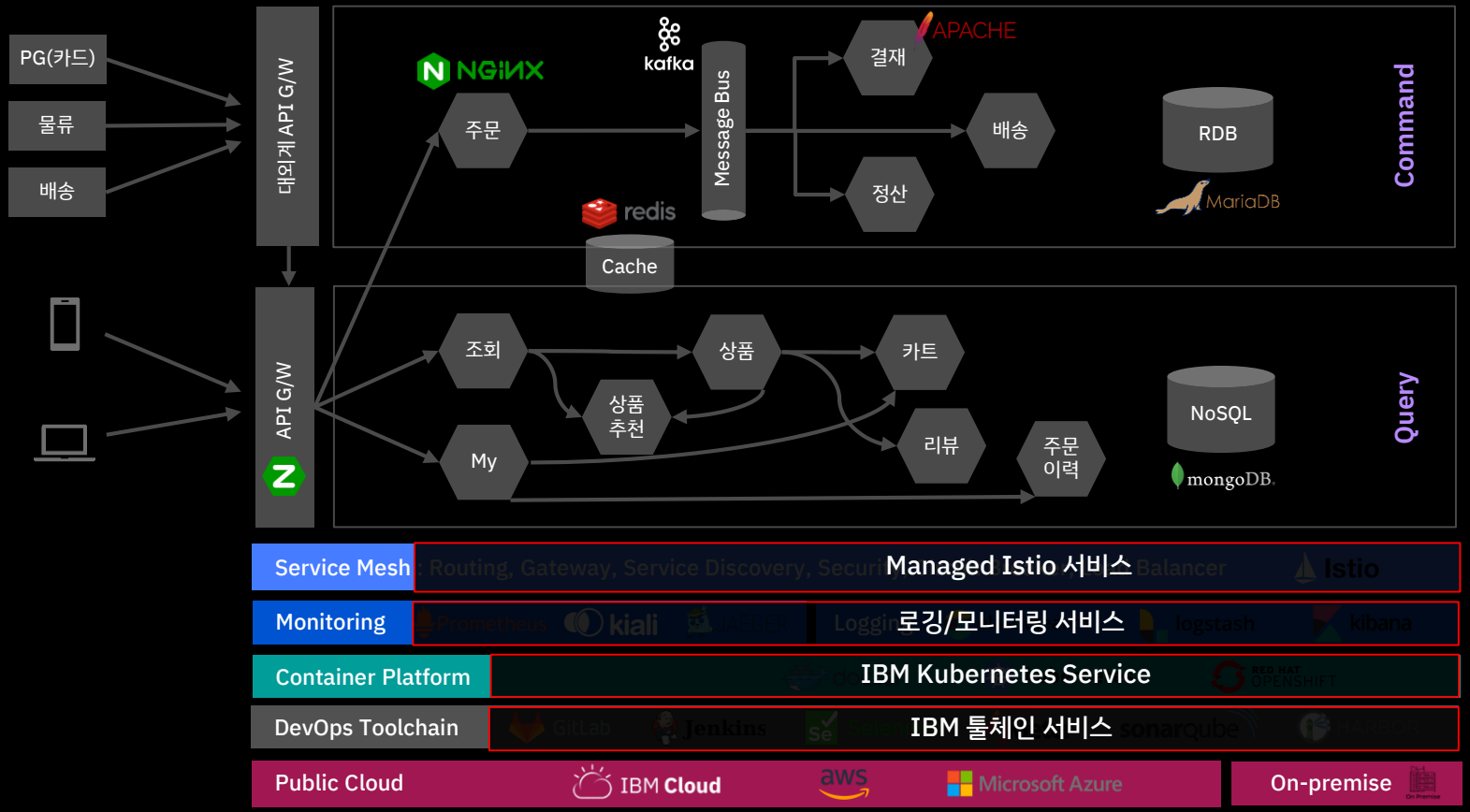

온프레미스

타 클라우드 제공사

4

IBM Cloud 플랫폼을 활용한 데모

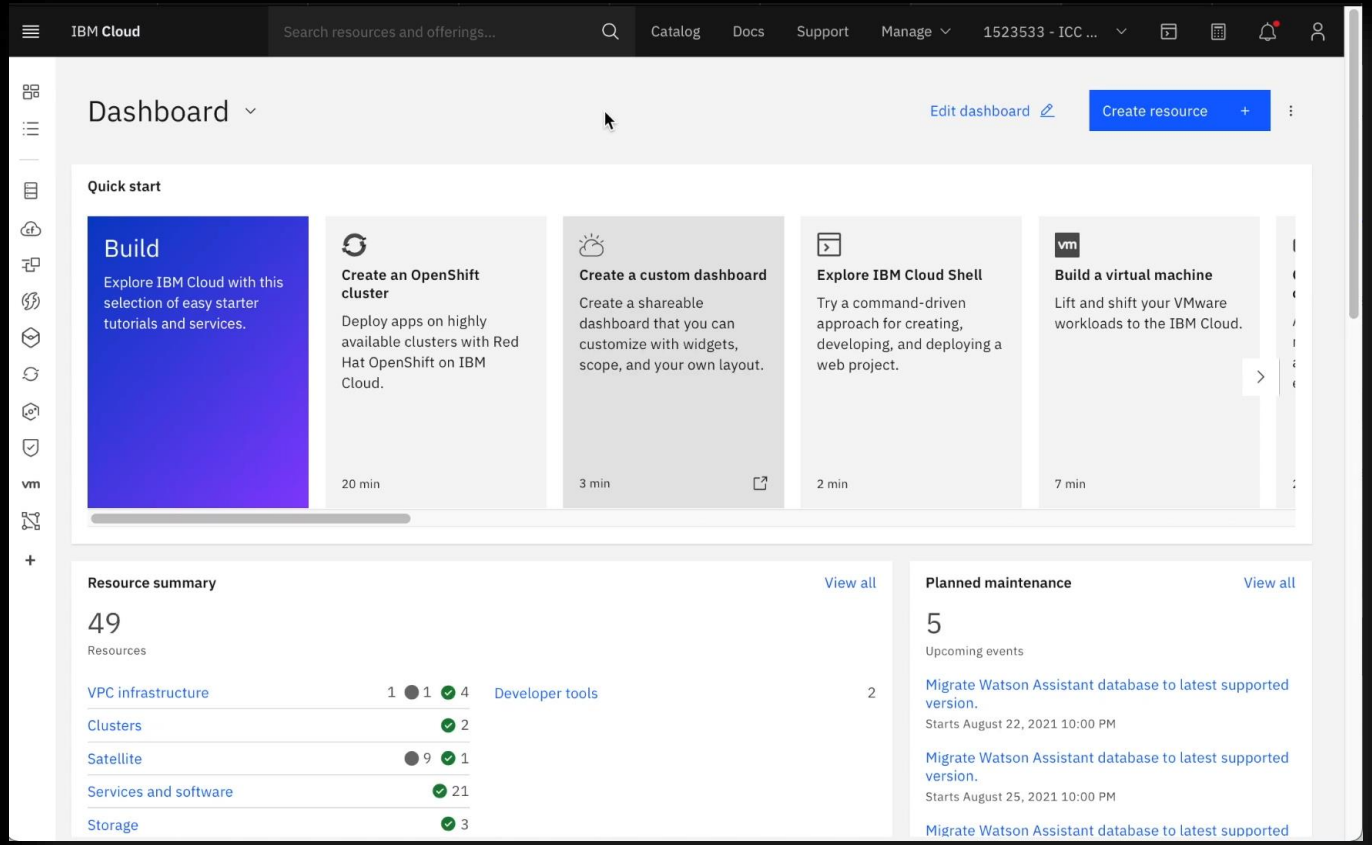
데모 – IBM 플랫폼 활용


PG(카드)
물류
배송
대외계 API G/W
API G/W
NGINX
kafka
Message Bus
주문
결재
배송
정산
APACHE
RDB
MariaDB
Command
redis
Cache
조회
상품
카트
상품 추천
My
리뷰
주문 이력
NoSQL
mongoDB.
Query
Service Mesh
Managed Istio 서비스
Monitoring
로깅/모니터링 서비스
Container Platform
IBM Kubernetes Service
DevOps Toolchain
IBM 툴체인 서비스
Public Cloud
IBM Cloud
aws
Microsoft Azure
On-premise

IBM 플랫폼 활용 – IKS (IBM Kubernetes Service)

IBM Cloud Kubernetes Service

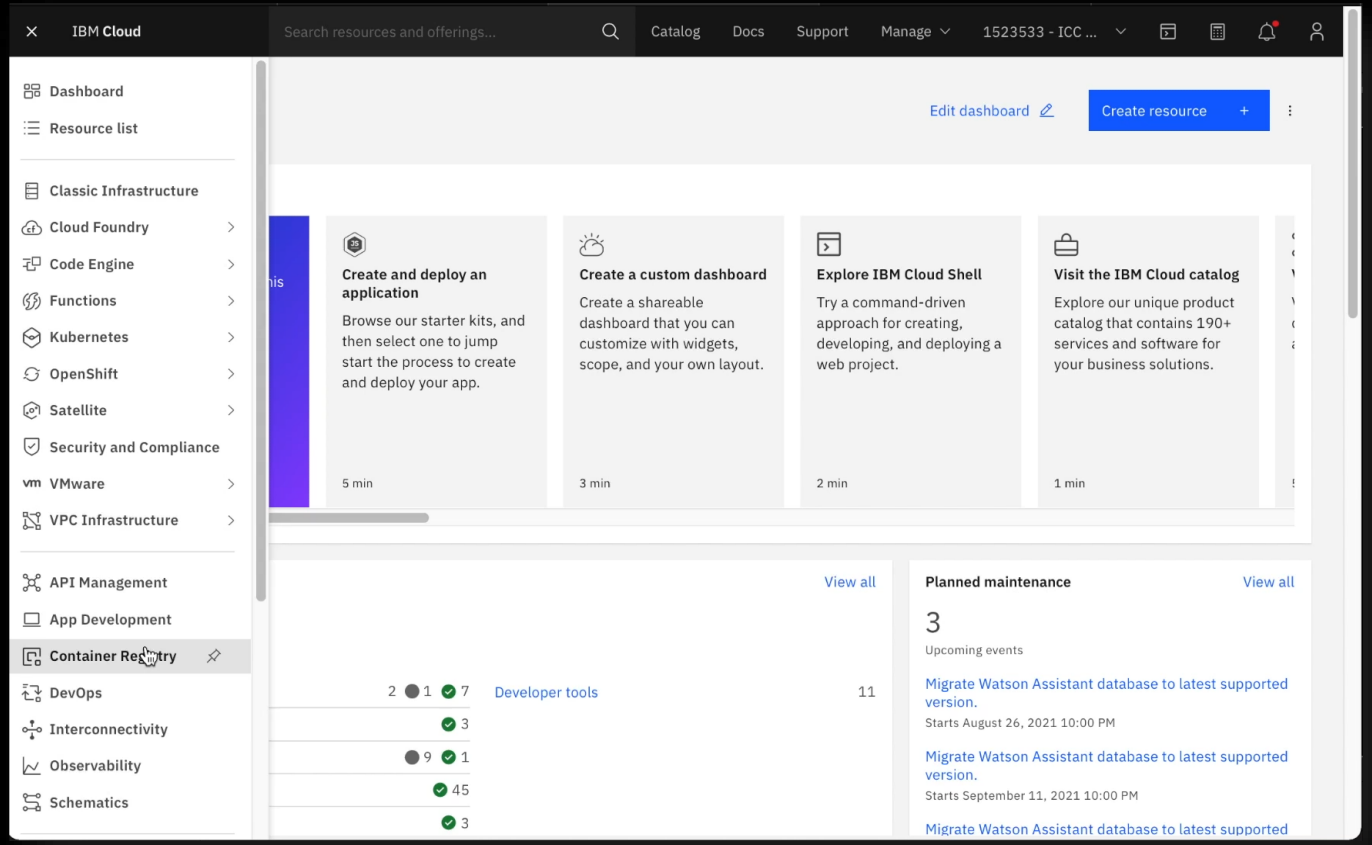
- **마스터 노드 고가용성(HA)**
- **무료 마스터 노드**
- 최신 Kubernetes 버전
- IBM Cloud Service 확장
  (Watson AI, IoT, 블록체인)



IBM 플랫폼 활용 – IBM Cloud Container Registry

IBM Cloud
Container Registry

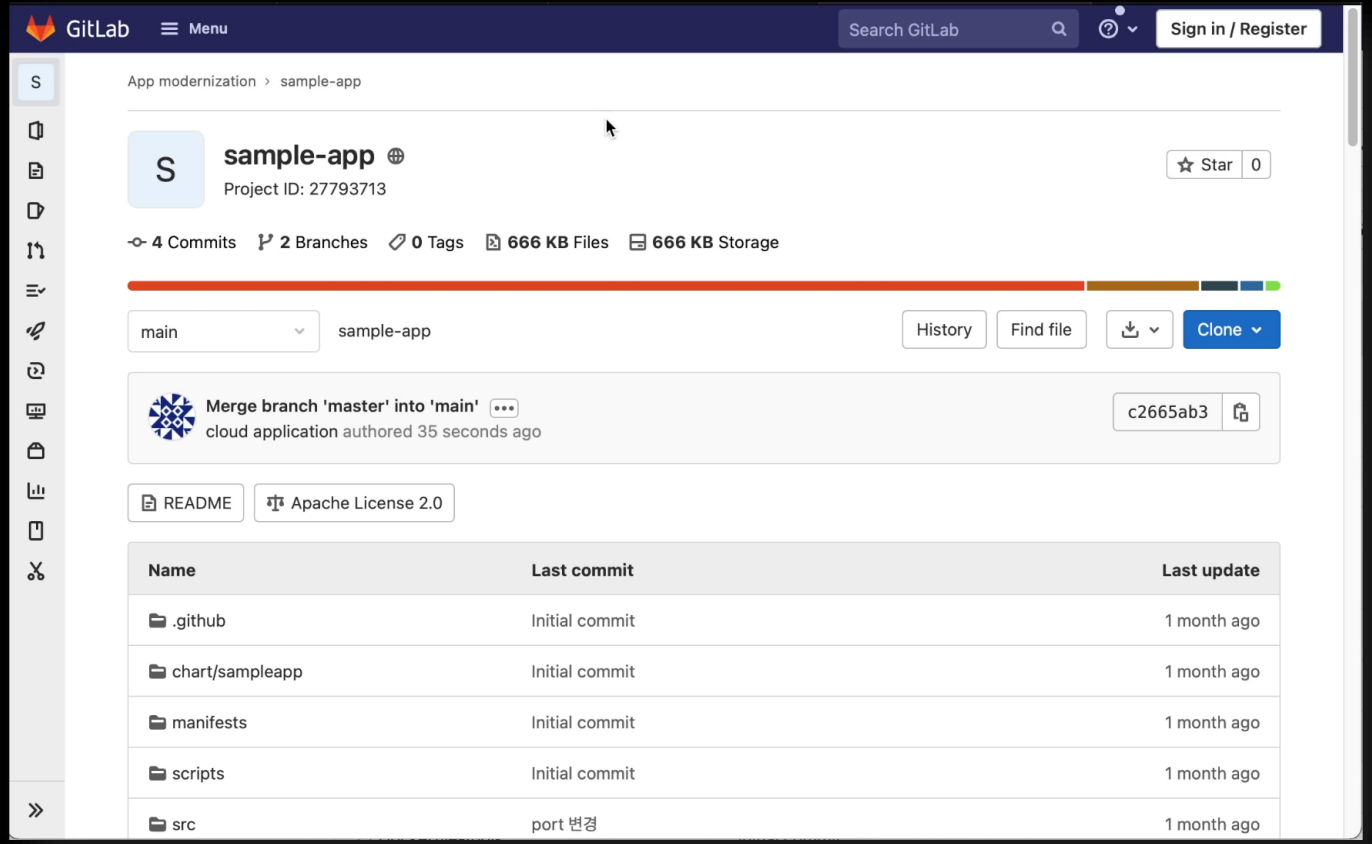
- 완전 관리형 이미지 저장소
- 멀티테넌트, 스케일링
- 강력한 보안 기능, 보안 인사이트
- IKS, Toolchain 사전 통합
- 스토리지(500MB) ,
  트래픽 (5GB/월) 무료 제공



IBM 플랫폼 활용 – IBM Cloud DevOps

IBM Cloud DevOps

- 오픈소스, 3rd 파트 지원
- 다양한 배치 템플릿 제공
- 다양한 빌드 환경 제공
  (VM, Openshift, CloudFoundry, Other Public Cloud)
- IBM Cloud IAM 보안 통합
- 배포 자동화 및 품질 보장



IBM 플랫폼 활용 – IBM Cloud Observability

로깅 & 모니터링

- 엔터프라이즈 스케일링
  (Not hidden costs)
- 실시간 로그 분석 제공
- 인프라 및 컨테이너 가시성
- 인사이트 기반 MSA 모니터링
- 메이저 제품에 대한 애드온 지원
  (Prometheus, Weave, Sysdig, fluentd)