SK telecom

AppServing

Simple java app deployment on K8S

컨테이너솔루션개발팀 최태일

01 도입배경

02 요구사항 / 기능 및 특징

03 아키텍쳐 / Tool Chain

04 Stage별 상세 내용

05 Demo / To-do

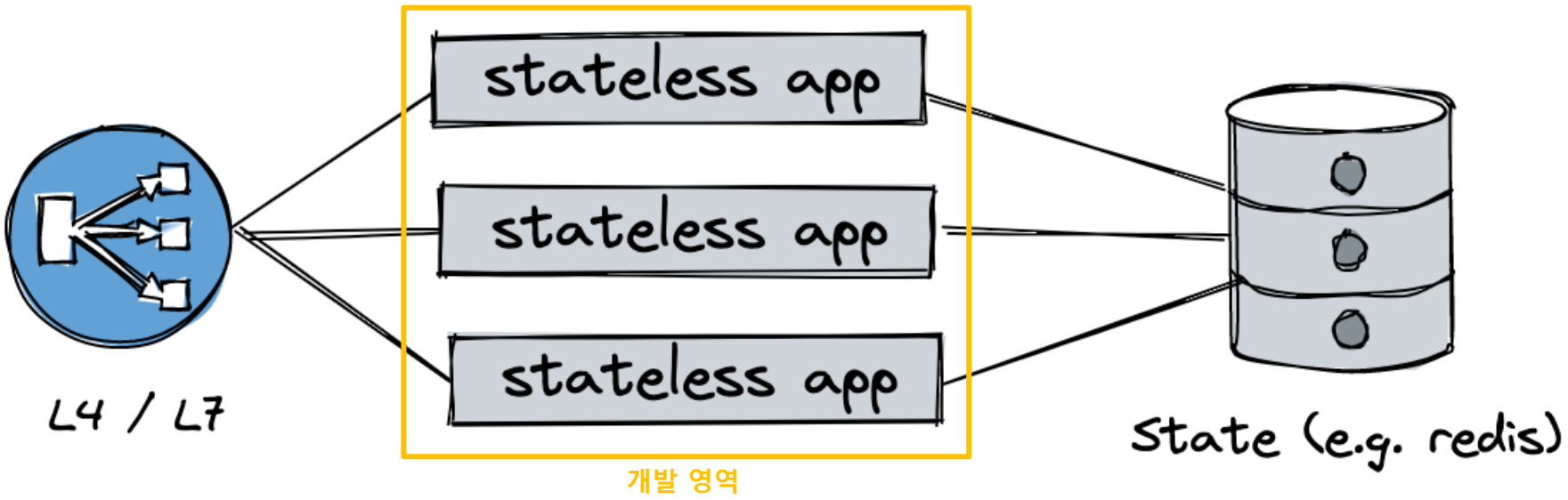
01

도입 배경

도입 배경

국내 SI SW 개발 현황

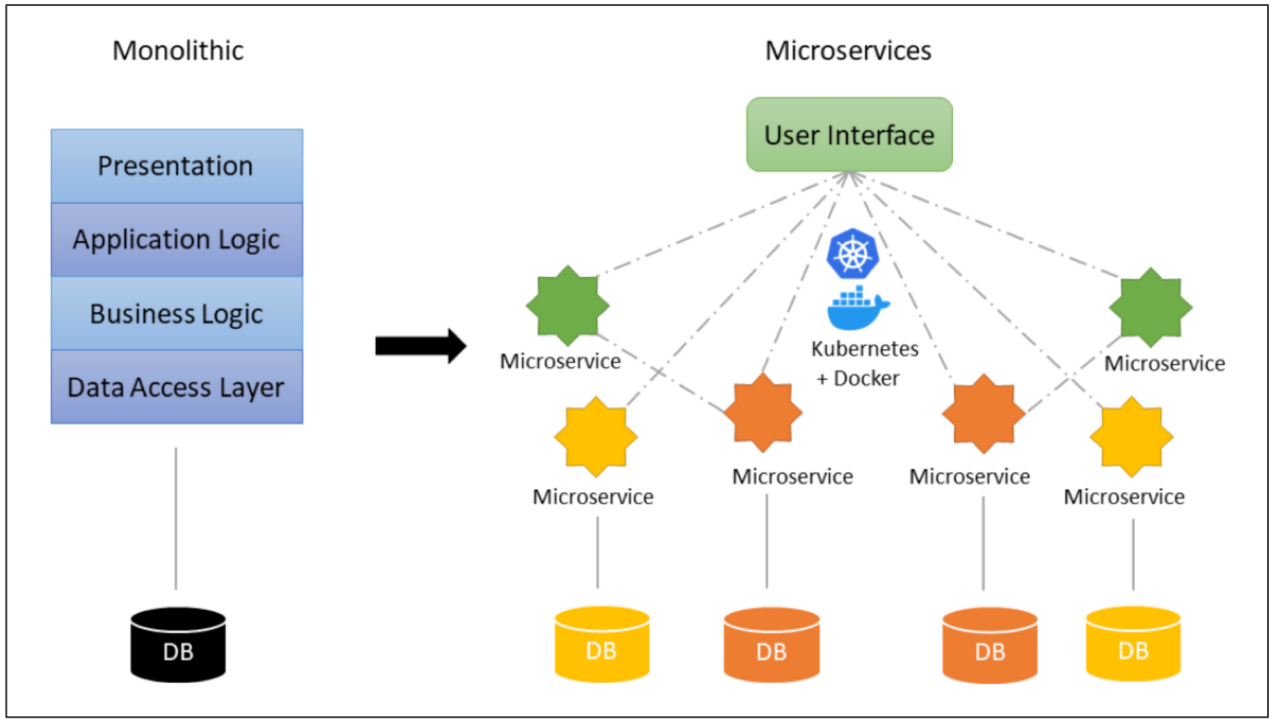
- 대부분의 SI app들은 MVC pattern을 지원하는 **Spring Boot** 등의 Java Framework을 주로 사용
  - 대부분 Web Service를 근간으로 하므로, REST API를 비롯한 WAS를 포함하는 경우 많음
  - 개발된 결과물은 jar 또는 war 확장자의 **단일 파일**로 제공
- Scaling/ HA나 SW 안정성은 Load balancer와 데이터베이스 등 Common하고 검증된 Solution를 통해 확보하고 SI 개발 자체는 **Business logic에만 집중**



도입 배경

MSA (Mirco Service Architecture) 전환기

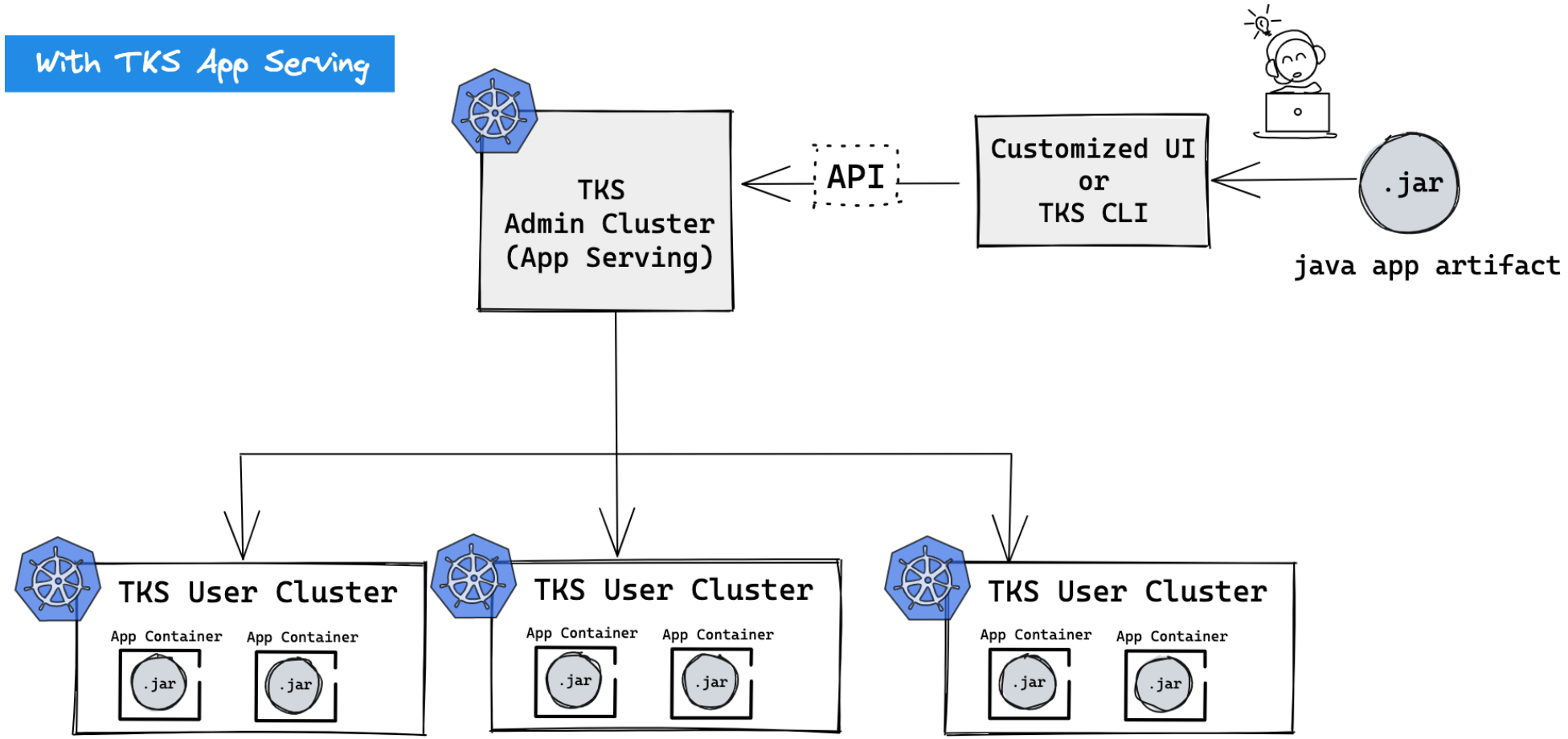
- SW를 컨테이너로 패키징하는 기술과, 다수의 컨테이너들을 효율적으로 관리할 수 있는 Kubernetes라는 플랫폼이 본격적으로 활용되기 시작하면서, MSA로 전환하고자 하는 기업들이 대폭 늘어나고 있음.
- Kubernetes
  - MSA 환경에서 핵심적인 container 관리 플랫폼 중 거의 표준으로 정립
  - 설치/운용/사용 측면에서 Learning Curve가 높음
  - **많은 Java 개발자 및 배포 담당자들은 아직 kubernetes에 대한 상세한 지식이 부족함**



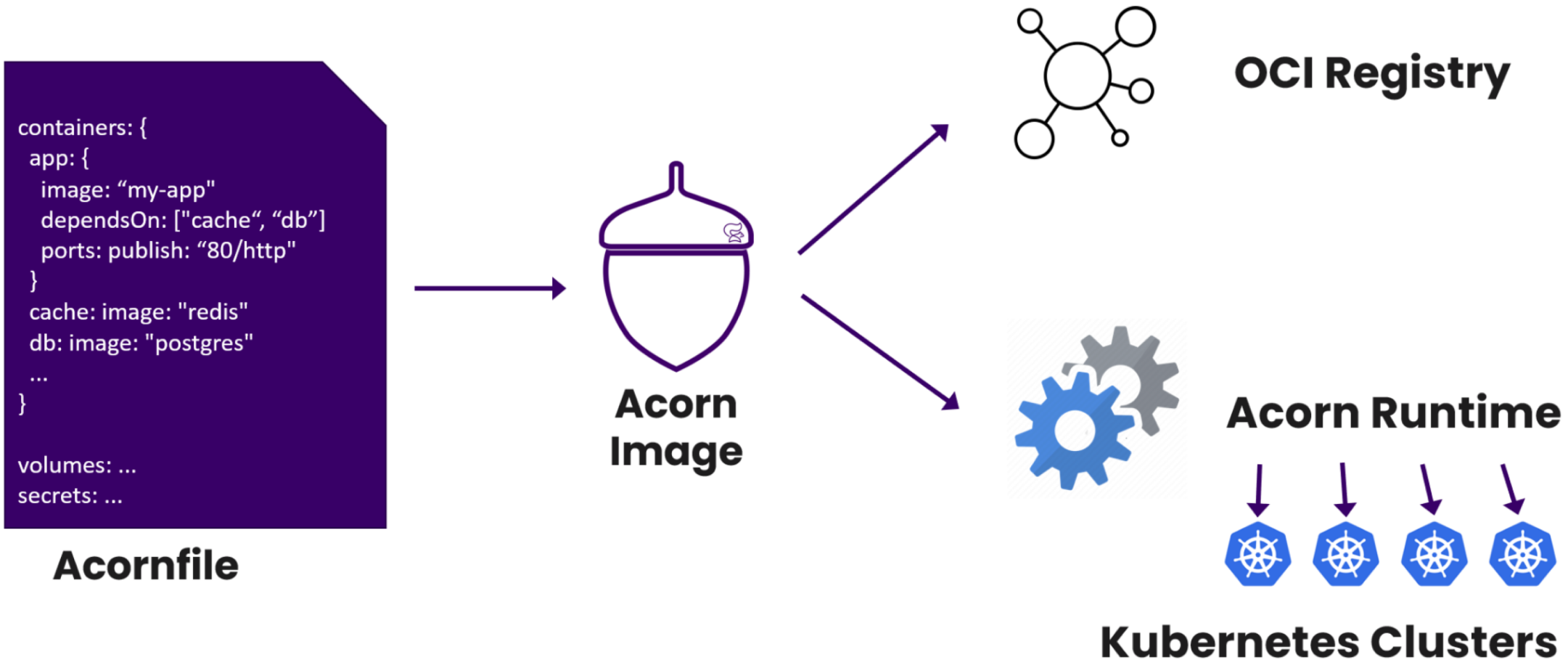
그림출처:
https://velog.io/@jkim2791/Micro-Service-ArchitectureMSA-%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C-%EC%84%9C%EB%B9%84%EC%8A%A4-%EC%95%84%ED%82%A4%ED%85%8D%EC%B3%90%EB%9E%80

도입 배경

- 따라서 이러한 사용자들이 **컨테이너와 Kubernetes에 대한 충분한 지식이 없이도** 쉽게 Legacy Java App을 컨테이너화하여 Kubernetes상에 배포할 수 있는 서비스를 제공



유사 서비스: Acorn



- Dockerfile 작성과 kubernetes manifest 작성에 대한 기초지식을 필요로 함
- k8s manifest 문법 대신 Acornfile 이라는 자체 DSL을 익혀야함 (https://docs.acorn.io/getting-started)

02

요구사항 / 기능 및 특징

요구사항

- Configmap / secret mount 기능이 꼭 필요함
  - ➢ DB 등 다른 요소들과의 접속 설정 등 환경에 따라 변경되는 값들을 동적으로 설정
- 기존에 빌드해놓은 이미지를 사용하여 배포만 하고 싶다
  - ➢ 특히 상용 환경 적용시 이미지 보안 점검을 통과한 이미지를 사용해야 함
- Blug/green, canary 배포 등이 가능했으면 좋겠다
- 배포 후 모니터링 기능이 제공되고, 실패시에는 에러 정보를 볼 수 있으면 좋겠다
- Nexus 등 repository 없이 개발 머신의 Jar/war 파일로 직접 배포가 가능하면 좋겠다 (개발 환경)
- Python 언어도 지원했으면 좋겠다

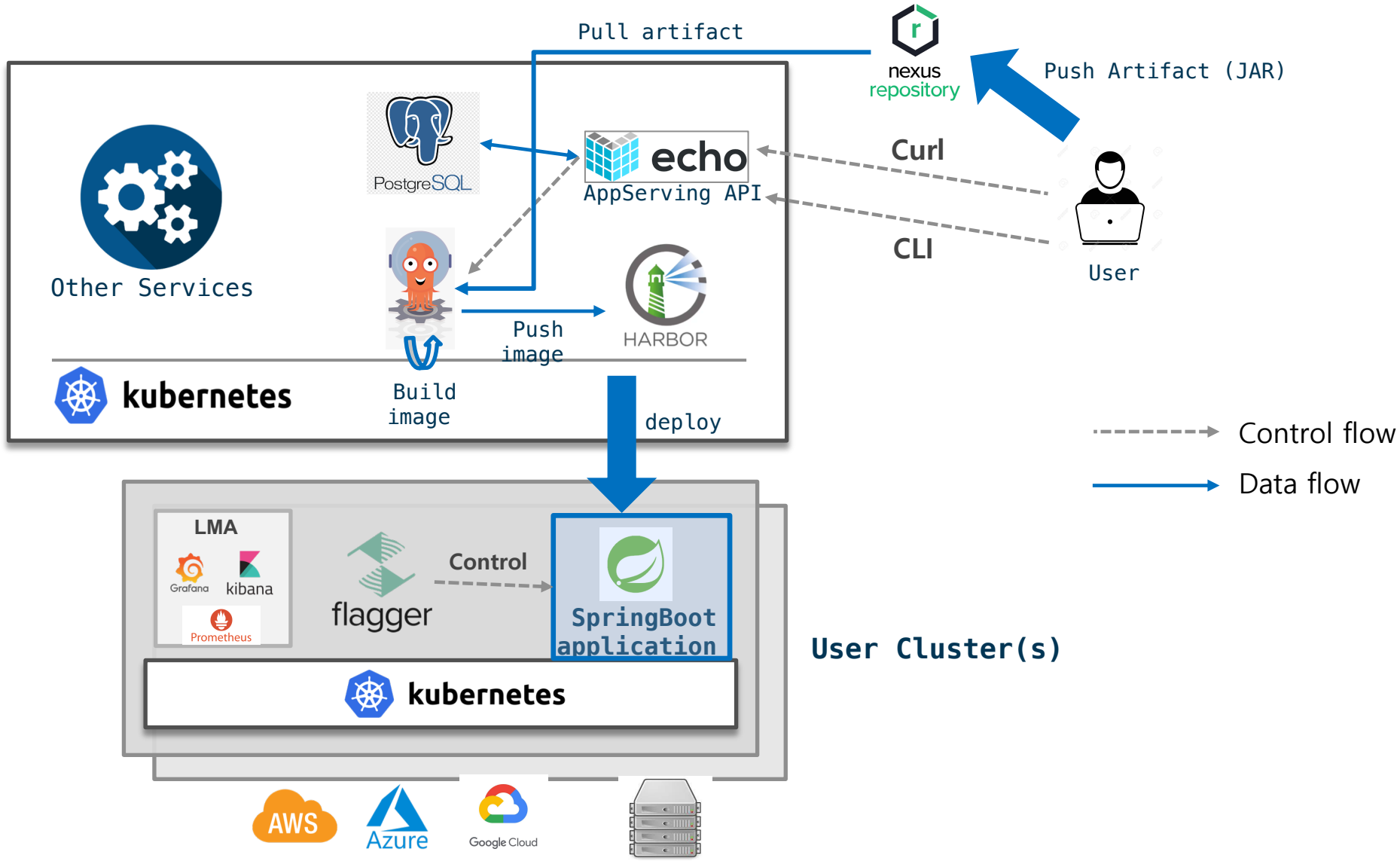
기능 및 특징

- 다양한 형태의 작업 수행 가능
  - Image build only
  - Deployment only
  - Image build & deployment
- 간편한 앱 버전 업그레이드 및 배포 히스토리 제공
- Blue/green 및 canary 배포 등 지원
- REST API 형태로 호출 가능 (향후 CLI 및 UI 제공 예정)
- Configmap/secret 및 PV mount 지원 (예정)
- Logging/monitoring 대시보드 제공 (예정)
- Vulnerability scanning for image (예정)

03 Architecture / Tool chain

Architecture

App Serving API가 요청을 받아 Argo Workflow를 통해 실제 작업을 수행하는 구조



Tool Chain

대부분 오픈소스를 활용

PostgreSQL (DB)

- AppServing 및 기타 서비스 관련 데이터 저장

Echo (Web framework)

- Rest API server 로 사용

Argo Workflows

- Workflow engine 으로서 단계적인 task 수행

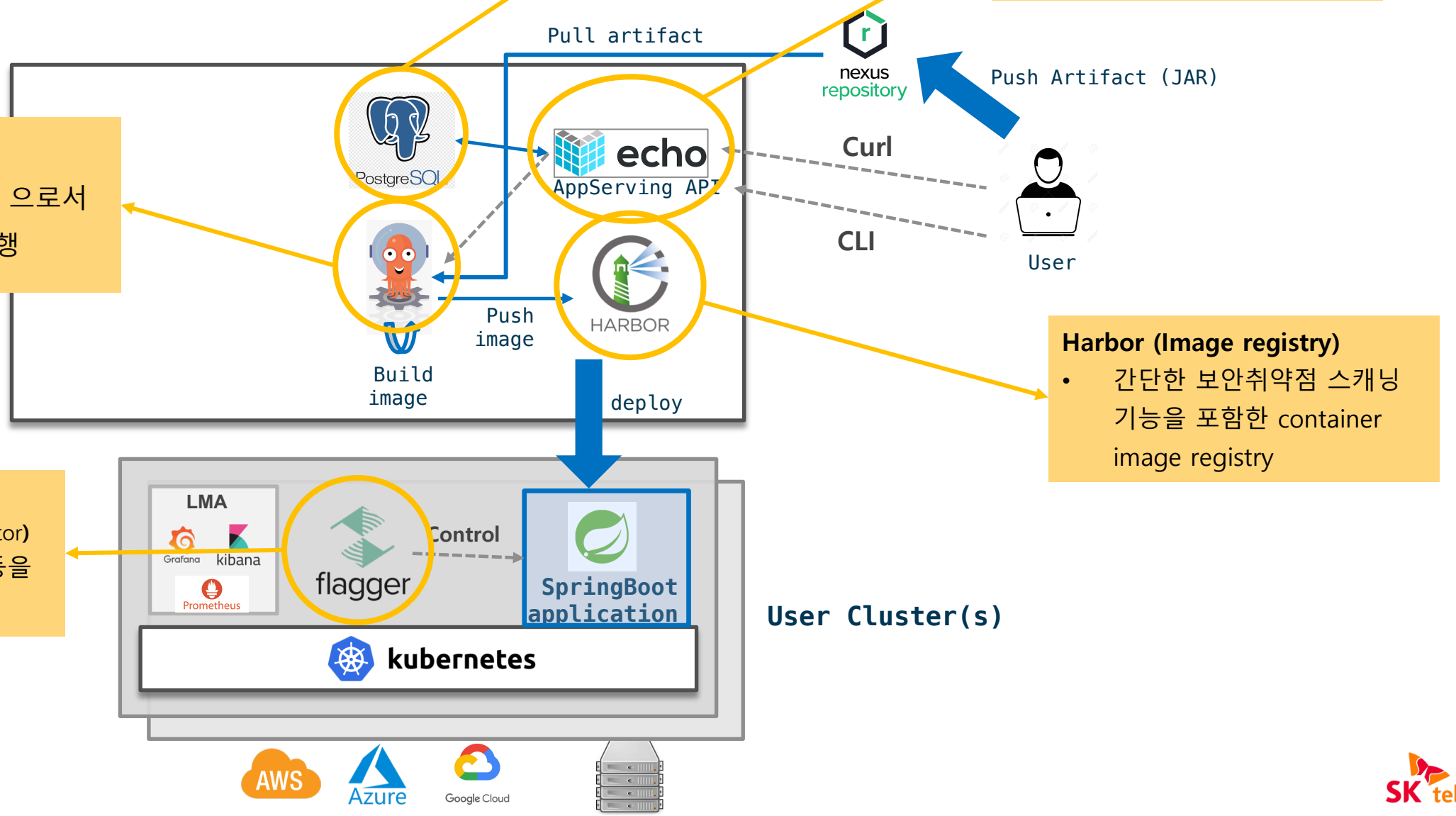
Harbor (Image registry)

- 간단한 보안취약점 스캐닝 기능을 포함한 container image registry

Flagger

(Progressive Delivery operator)

- B/G, canary 배포 등을 수행



04

Stage별 상세 내용

Request params

REST API 호출시 C/R/U/D 각 작업별로 parameter 필요

- 최초 App 생성 (Create) 시 필요한 parameter 예시

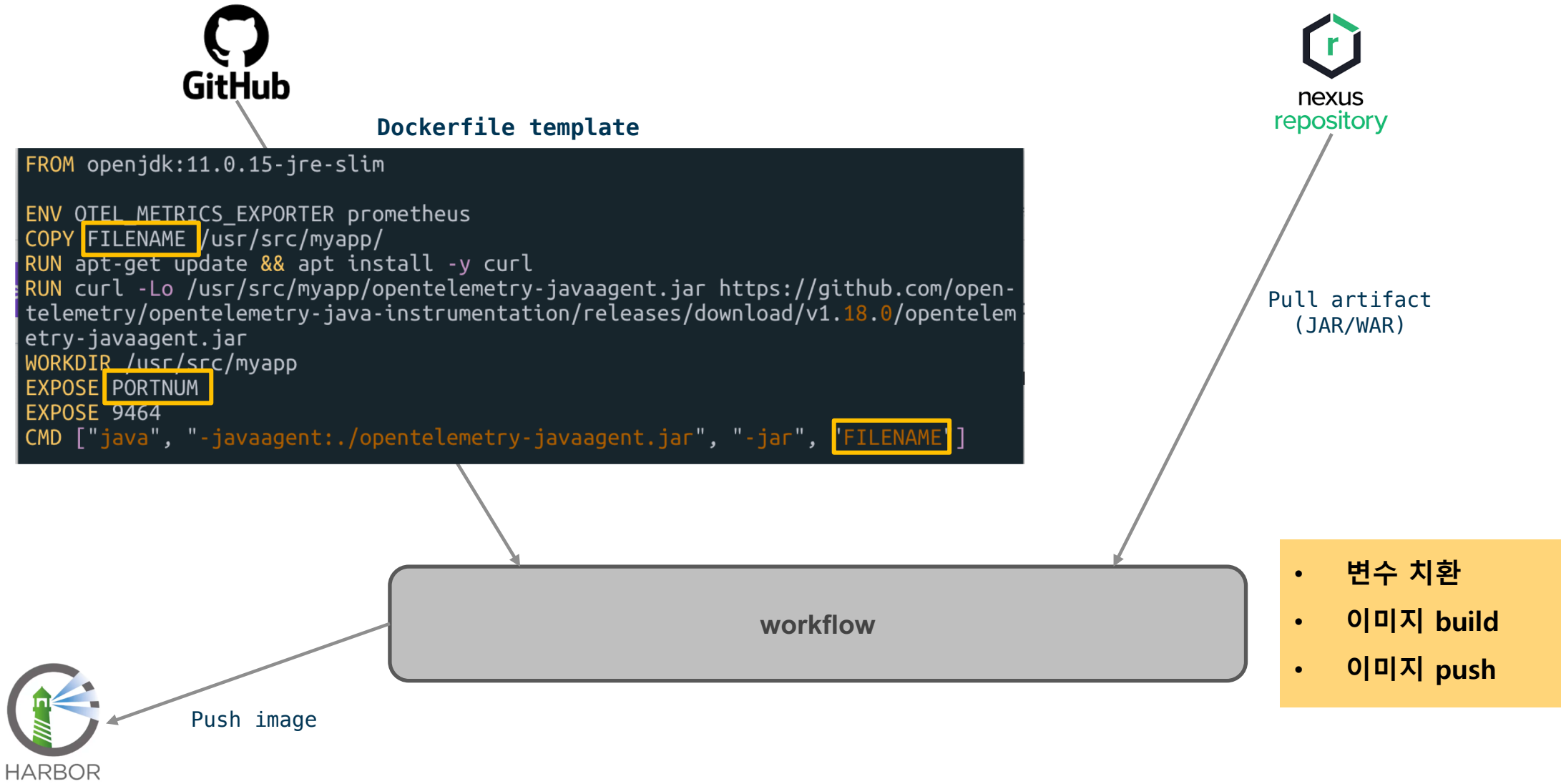
```
"name": "my-petclinic",
"version": "v1",
"target_cluster_id": "cdozqn511",
"task_type": "all",
"artifact_url": "http://OOO/repository/my-release-repo/default/petclinic/1.0/petclinic-1.0.jar",
"port": "8080",
"resource_spec": "medium",
"profile": "default"
```

- 배포된 App 조회 (Get)

```
$ curl -X GET http://localhost:9112/apps/UUID
```

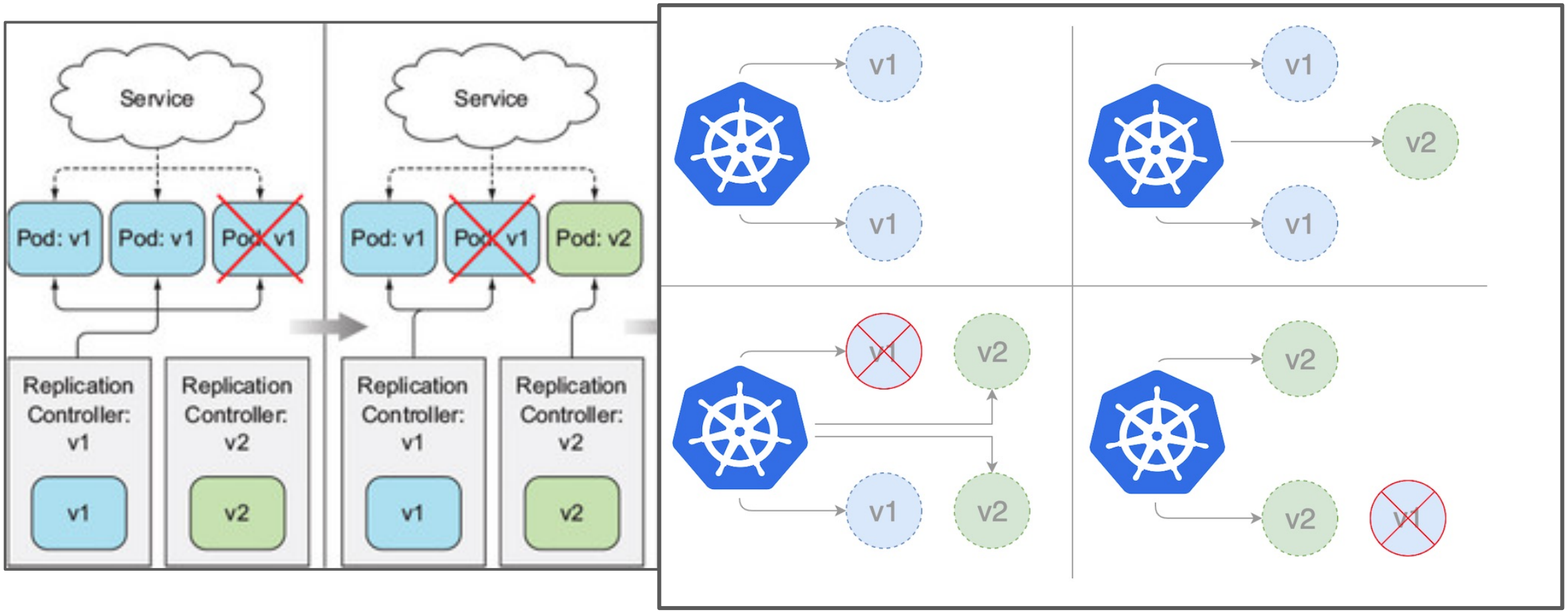
Image Build

Jar 등 artifact 파일과 심플한 Dockerfile 템플릿을 사용하여 이미지 build 수행



Deployment > rolling-upgrade

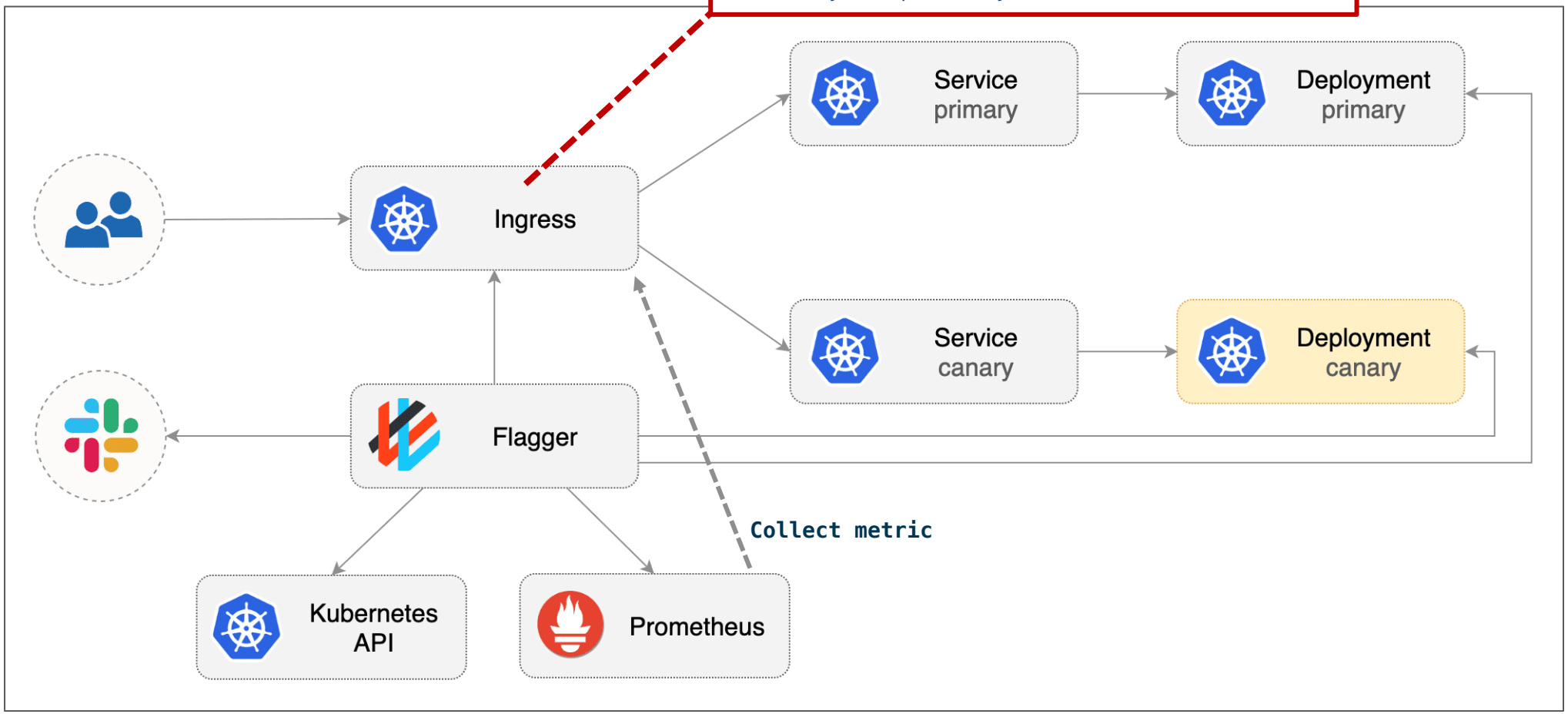
- Kubernetes 의 가장 기본적인 upgrade 방식으로써 무중단 업그레이드 수행
- App Serving 에서는 순수하게 helm upgrade 명령을 사용하여 rolling upgrade 수행



Deployment > Canary

- Flagger가 ingress 및 service 들을 제어
- Ingress의 canary-weight 를 사용하여 비중 조절

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: "nginx"
    nginx.ingress.kubernetes.io/canary: "true"
    nginx.ingress.kubernetes.io/canary-weight: "5"
  name: your-api-canary
```



SK telecom

Deployment > Challenge for B/G deployment

Metric 수집을 어떻게 할 것인가?

- Canary의 경우, nginx ingress controller 의 metric을 활용 가능
- Ingress를 사용하지 않는 B/G 의 경우는 pod 자체가 metric을 제공해야 함

Metric 수집을 위한 다양한 방법들 검토

- Prometheus exporter를 만들어 sidecar 형태로 붙인다? -> 구현의 부담
- Spring Boot 가 제공하는 starter-actuator 사용

```
dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-actuator’
  Implementation 'io.micrometer:micrometer-registry-prometheus'
}
```

  - App 개발자가 pom.xml, applicaion.yaml 등의 파일을 직접 수정해야함.
- **Opentelemetry-java-agent** (https://github.com/open-telemetry/opentelemetry-java-instrumentation)
  - Java [illegible]ation에 attach되어 실행되는 agent로서 bytecode 삽입을 통해 metric data를 수집/제공

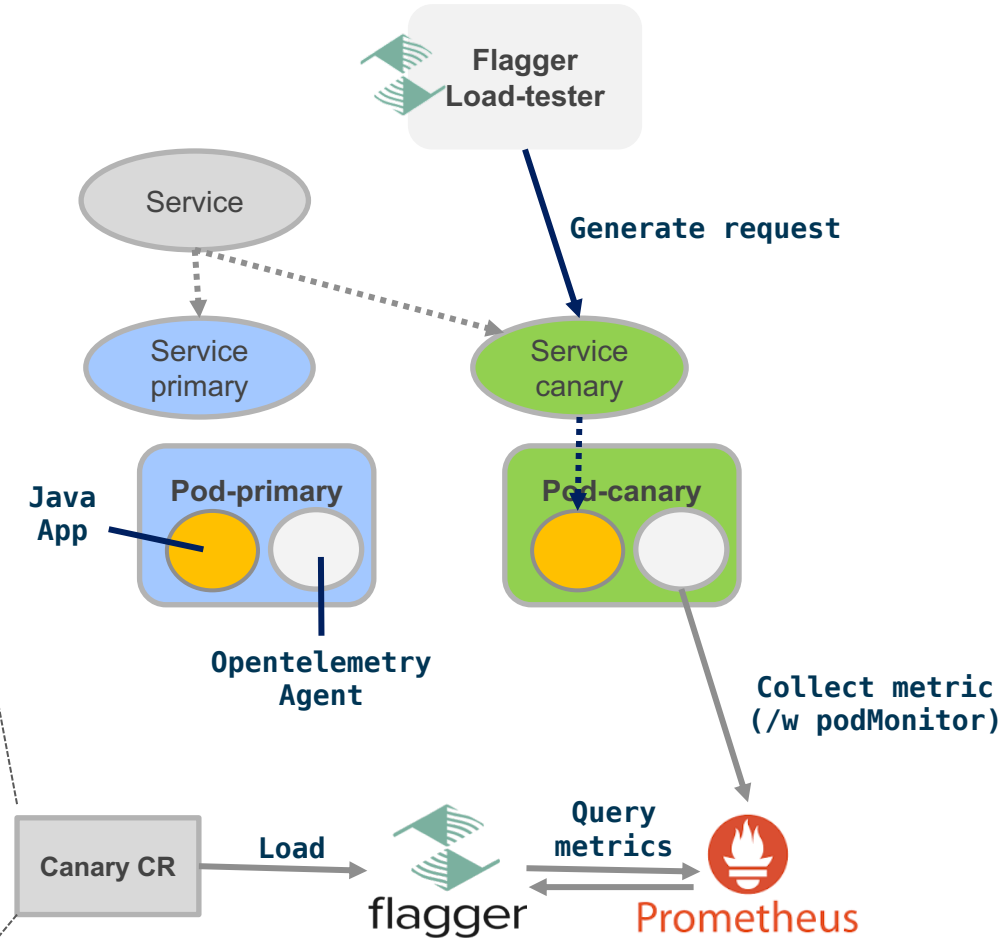
A single, open source standard and a set of technologies to capture and export metrics, traces, and logs from your cloud-native applications and infrastructure.

Deployment > Blue-green (in detail)

- Flagger가 deployment 및 service를 제어 (primary & canary)
- opentelemetry-agent 및 podMonitor를 통한 메트릭 수집 -> canary 성공 여부 판단

```yaml
apiVersion: flagger.app/v1beta1
kind: Canary
metadata:
  name: petclinic-bg
  namespace: petclinic
```

```yaml
  analysis:
    # schedule interval (default 60s)
    interval: 60s
    # max number of failed checks before rollback
    threshold: 5
    # number of checks to run before rollback
    iterations: 5
    # Prometheus checks based on
    # http_request_duration_seconds histogram
    metrics:
      - name: my-request-success-rate
        templateRef:
          name: my-request-success-rate
          namespace: flagger
        # minimum req success rate (non 5xx responses)
        # percentage (0-100)
        thresholdRange:
          min: 95
        interval: 1m
    # acceptance/load testing hooks
    webhooks:
      - name: load-test
        url: http://flagger-loadtester.flagger/
        timeout: 5s
        metadata:
          type: cmd
          cmd: "hey -z 1m -q 10 -c 2 http://petclinic-canary.petclinic:8080/"
```



Deployment > blue-green in detail

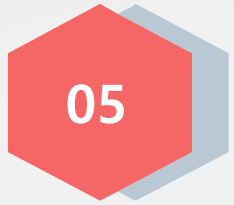
Canary 배포 성공 여부 판단을 위해 **custom metric** 생성

Canary CR

```yaml
analysis:
  # schedule interval (default 60s)
  interval: 60s
  # max number of failed checks before rollback
  threshold: 5
  # number of checks to run before rollback
  iterations: 5
  # Prometheus checks based on
  # http_request_duration_seconds histogram
  metrics:
    - name: my-request-success-rate
      templateRef:
        name: my-request-success-rate
        namespace: flagger
      # minimum req success rate (non 5xx responses)
      # percentage (0-100)
      thresholdRange:
        min: 95
      interval: 1m
  # acceptance/load testing hooks
  webhooks:
    - name: load-test
      url: http://flagger-loadtester.flagger/
      timeout: 5s
      metadata:
        type: cmd
        cmd: "hey -z 1m -q 10 -c 2 http://petclinic-canary.petclinic:8080/"
```

Custom metric

```yaml
apiVersion: flagger.app/v1beta1
kind: MetricTemplate
metadata:
  name: my-request-success-rate
  namespace: flagger
spec:
  provider:
    type: prometheus
    address: http://lma-prometheus.lma:9090
  query: |
    sum(
      rate(
        http_server_requests_count{
          namespace="{{ namespace }}",
          pod=~"{{ target }}-[0-9a-zA-Z]+(-[0-9a-zA-Z]+)",
          status~"2.*"
        }[{{ interval }}]
      )
    )
    /
    sum(
      rate(
        http_server_requests_count{
          namespace="{{ namespace }}",
          pod=~"{{ target }}-[0-9a-zA-Z]+(-[0-9a-zA-Z]+)"
        }[{{ interval }}]
      )
    )
    * 100
```



Demo / To-do

Demo Scenario

준비된 Jar 파일로 이미지를 빌드하고 조회 및 업그레이드하는 과정 시연

- Artifact repositoy (Eg, nexus) 에 Jar 파일 준비
- App 배포 API 호출 (CREATE)
- 배포된 App 상태 조회 (GET) 및 브라우져를 통한 접속 확인
- Update API 호출 (UPDATE)
- 배포된 App 상태 조회 및 브라우져를 통한 접속 확인

To-do

향후 개발 예정 기능

- 사용자 앱 관련 정보를 보여주는 Logging/Monitoring 대시보드 제공
- Configmap/secret 및 PV mount 지원
- 사용자용 CLI 및 UI 제공
- Vulnerability scanning for image
- 단일 어플리케이션 외에 마이크로 서비스 그룹 배포 지원
- Artifact repository를 input으로 받는 대신, Java 개발자가 app을 개발한 로컬 환경에서 바로 API/CLI 호출로 즉시 이미지 빌드 및 배포를 수행하는 기능

Q & A

감사합니다