Java 로 Kubernetes Operator를 만들 수 있다고?

11번가 Core Platform Development Team 최유진

Operator Pattern이란?

Operator Pattern

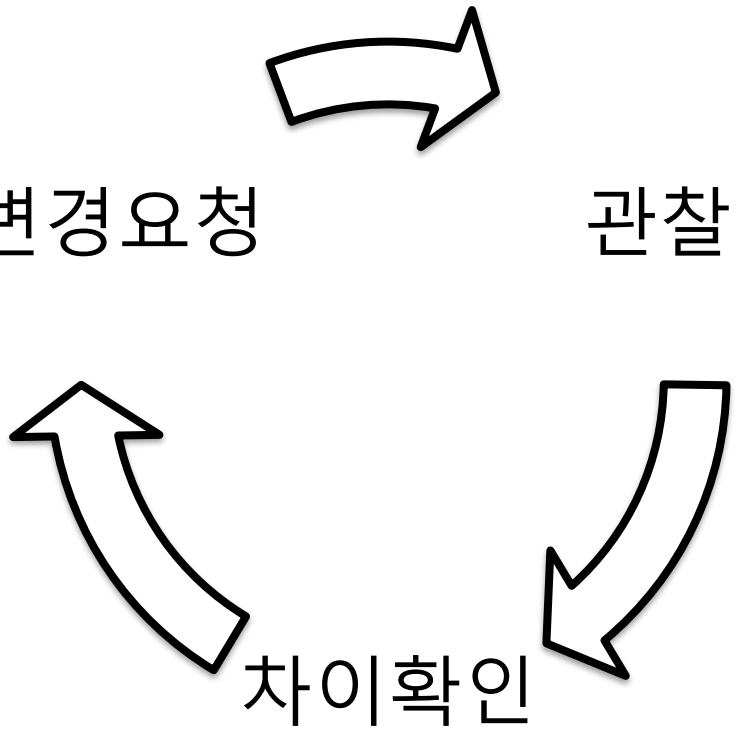
- Kubernetes 코드를 변경하지 않고, 클러스터의 동작을 확장하기 위한 패턴이다.
- 사용자가 정의한 리소스(**Custom Resource**)에 따라 동작하는 **컨트롤러**를 만들 수 있다.
- 사람(Operator)이 해왔던 반복적인 작업을 자동화 할 수 있다.

Custom Resource

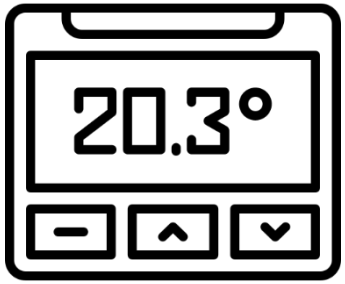
- 리소스는 특정한 API 객체 집합을 저장하는 엔드포인트를 나타낸다.
  - 예) pods : pod 객체 집합
- 커스텀 리소스는 쿠버네티스 API의 확장으로, 새로운 종류의 리소스를 정의할 수 있다.

컨트롤러

- desired state와 실제 상태를 비교하여 상태를 일치시키기 위한 동작을 수행하는 Control Loop
- Operator는 (커스텀) 컨트롤러이다.

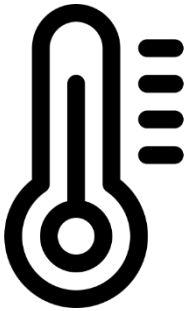


Control Loop (예시)



원하는 온도로 설정
desired state

!=

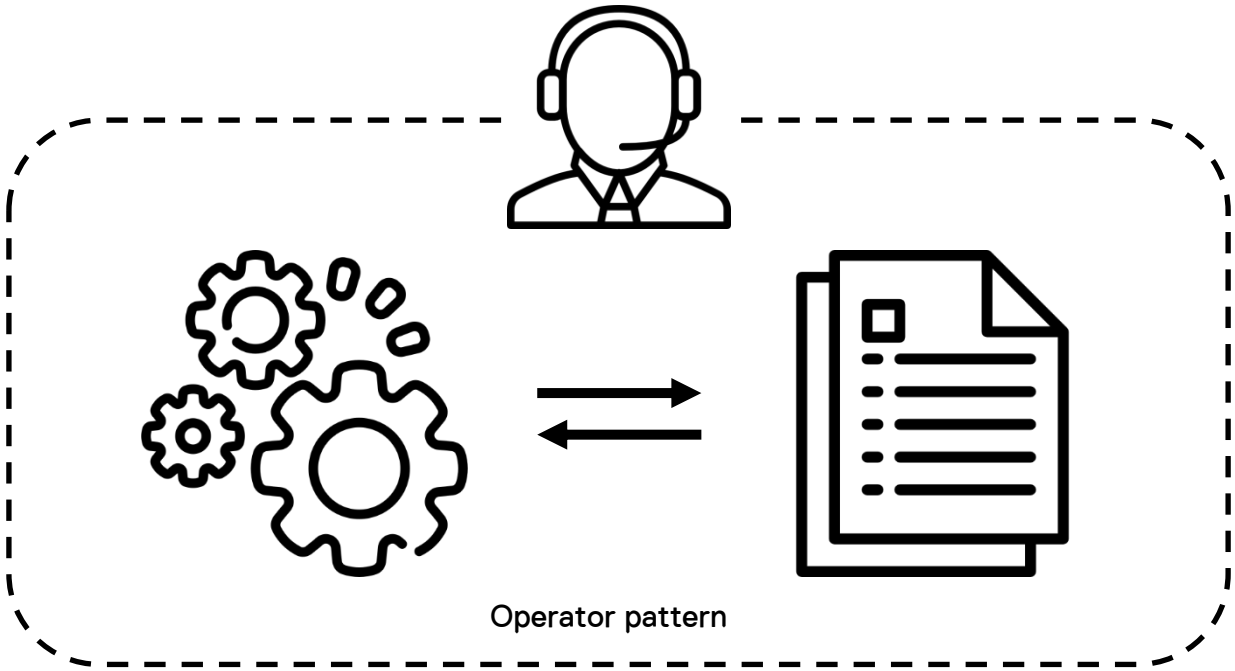


현재 온도
current state

온도 조절
controller

Operator Pattern

커스텀 컨트롤러가 사용자가 생성한 Custom Resource를 watch 하고,
Custom Resource 에 정의된 desired state와 현재 상태를 일치시키기 위한
Custom Resource에 특화된 동작을 하는 것



Operator pattern

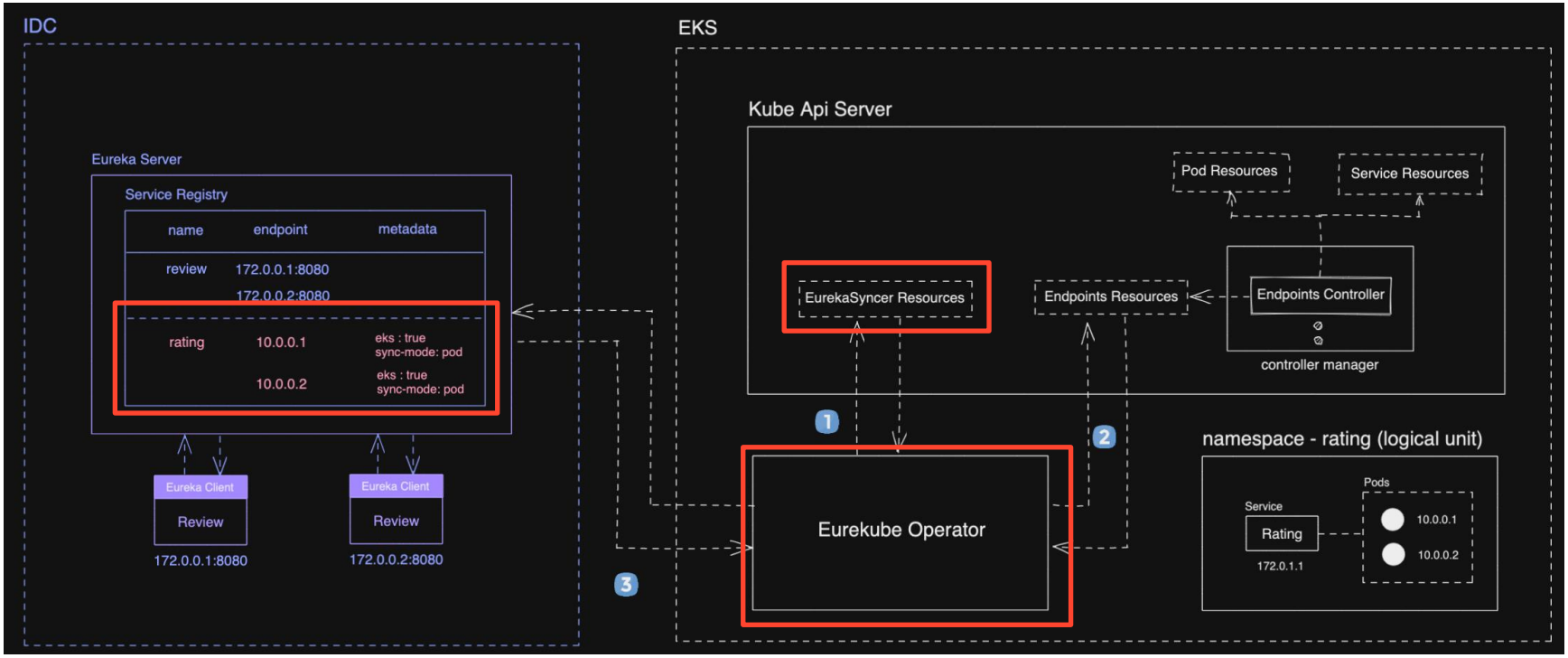
Operator Pattern을 통해 할 수 있는 것들

- 애플리케이션의 상태를 백업하고 복원
- 데이터베이스 스키마 또는 추가 구성 설정과 같은 관련 변경 사항에 따른 애플리케이션 코드 업그레이드 처리
- 쿠버네티스 API를 지원하지 않는 애플리케이션에 서비스를 게시하여 검색을 지원
- 클러스터의 전체 또는 일부에서 장애를 시뮬레이션하여 가용성 테스트

Operator Pattern 예시 (in 11st)

Eurekube Operator

- IDC와 AWS의 서비스 디스커버리를 통합하기 위한 Operator
- Eureka Syncer라는 Custom Resource의 변경사항을 watch한다.
- 변경사항이 생기면, IDC내의 Eureka에 AWS내의 Endpoint를 서비스로 추가한다.
- 자세한 내용은 **서비스 디스커버리 붙였다 뗐다, 참 쉽죠?** 세션 참고



Operator 만들기

Operator 개발 환경

흔한 자바 서버 개발자의 기술 스택

Custom Resource를 watch 하는 Operator 제작

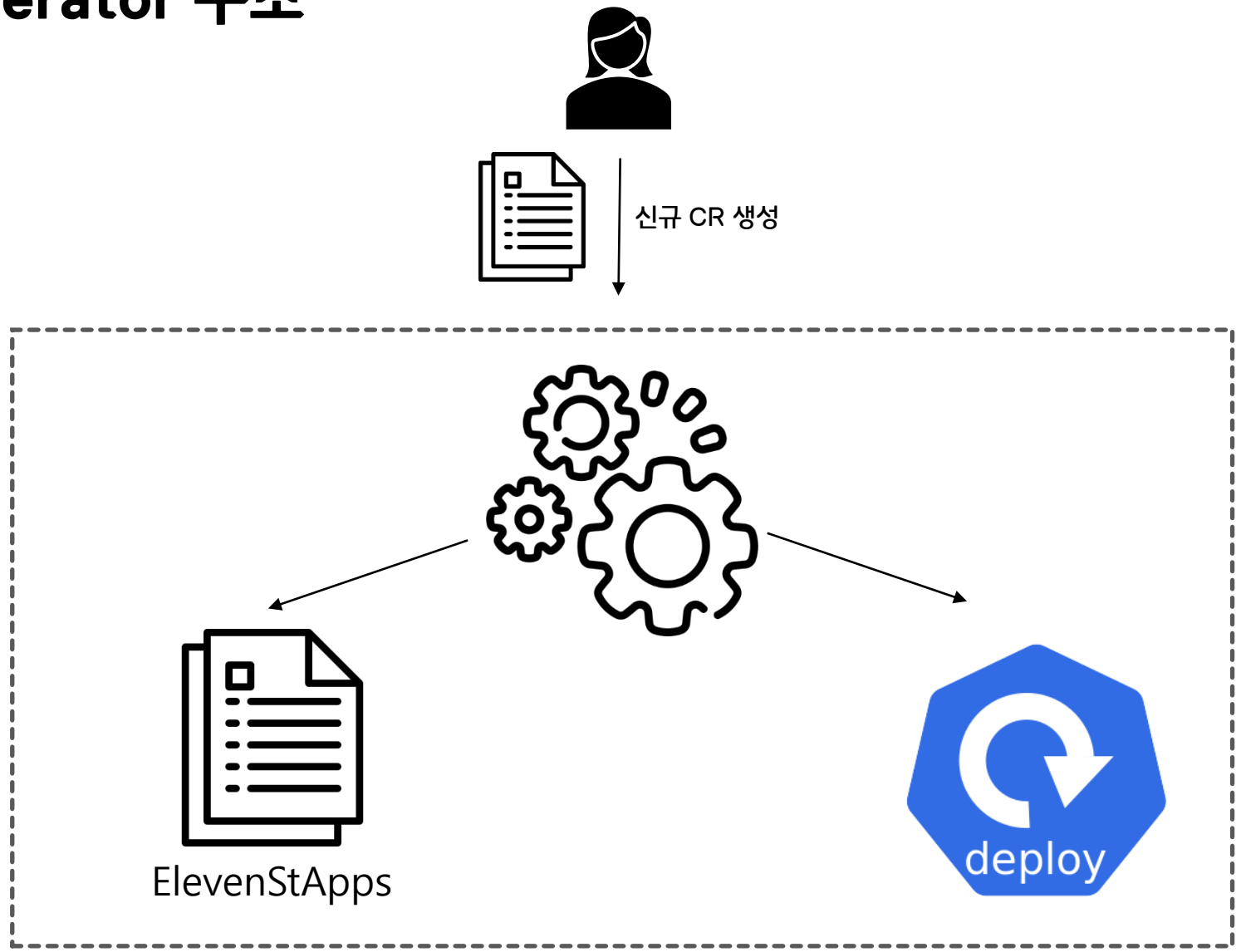
- **Java Operator SDK 사용**
  - fabric8 Kubernetes Client를 기반으로 만들어진 SDK
- Spring Boot Starter를 제공
- 자바로 Operator를 개발하면서 마주할 수 있는 공통적인 문제들이 처리되어 있음

[JAVA OPERATOR SDK]

Fabric8 Kubernetes client

- REST API를 언어마다 구현해 둔 공식 / 비공식 라이브러리가 존재
- 비공식 라이브러리
- 자바 중 가장 오래된 라이브러리
- 공식 라이브러리보다 2년 앞선 2015년에 시작

Operator 구조

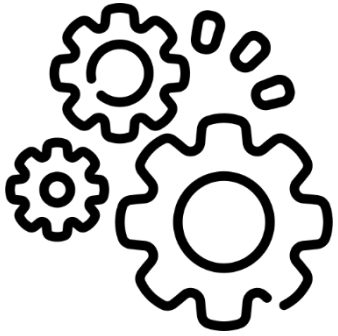


개발 과정

1

Custom Resource Definition 생성

2



Operator 제작 (Java Operator SDK)

개발 과정

1

Custom Resource Definition 생성

2

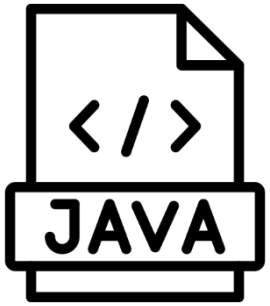
Operator 제작 (Java Operator SDK)

Custom Resource Definition란?

- Custom Resource를 정의할 때 사용된다.
- Pods, ReplicaSets, ConfigMaps등과 같은 빌트인 리소스처럼 CRD(Custom Resource Definition)을 통해 Kubernetes에 새로운 리소스를 추가할 수 있다.
- 새로운 리소스의 이름과 스키마를 정의할 수 있다.
- https://kubernetes.io/docs/tasks/extend-kubernetes/custom-resources/custom-resource-definitions/

Custom Resource Definition 생성 방법

yaml 파일을 통한 생성 (기본)



Java 코드를 통한 생성

yml을 통한 CRD 생성

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: elevenstapps.11st.example.com
spec:
  group: 11st.example.com
  names:
    kind: ElevenStApp
    plural: elevenstapps
    singular: elevenstapp
  scope: Namespaced
  versions:
  - name: v1
    schema:
      openAPIV3Schema:
        properties:
          spec:
            properties:
              name:
                type: string
              label:
                type: string
            type: object
          status:
            type: object
        type: object
    served: true
    storage: true
    subresources:
      status: {}
```

개별 항목에 대한 설명은 링크를 참고

https://kubernetes.io/docs/reference/kubernetes-api/extend-resources/custom-resource-definition-v1/

자바 코드를 통한 CRD 생성

- Fabric8에서 제공하는 crd-generator 사용
- https://github.com/fabric8io/kubernetes-client/blob/master/doc/CRD-generator.md

Gradle 의존성 추가

```
annotationProcessor 'io.fabric8:crd-generator-apt:5.12.2'
```

Maven 의존성 추가

```
<dependency>
  <groupId>io.fabric8</groupId>
  <artifactId>crd-generator-apt</artifactId>
  <scope>provided</scope>
</dependency>
```

자바 코드를 통한 CRD 생성

- CRD Generator는 컴파일시 CustomResource를 상속 받은 클래스를 기반으로 CRD를 생성
- CustomResource는 클래스명이 나타내듯 CustomResource를 나타낼 때 사용하는 클래스

```
@Version("v1")
@Group("11st.example.com")
public class ElevenStApp extends CustomResource<ElevenStAppSpec, ElevenStAppStatus> implements Namespaced {
}
```

CustomResource 나타내기

- 타입 파라미터
  1. Spec: 유저가 적용하려고 하는 desired state
  2. Status: custom resource의 현재 상태

io.fabric8.kubernetes.client

Class CustomResource<S,T>

java.lang.Object
  io.fabric8.kubernetes.client.CustomResource<S,T>

Type Parameters:

`S - the class providing the Spec part of this CustomResource`

`T - the class providing the Status part of this CustomResource`

All Implemented Interfaces:

`io.fabric8.kubernetes.api.model.HasMetadata, io.fabric8.kubernetes.api.model.KubernetesResource, Serializable`

CustomResource 나타내기

```java
public class ElevenStAppSpec {
    String appId;
    String owner;

    public String getAppId() { return appId; }

    public void setAppId(String appId) { this.appId = appId; }

    public String getOwner() { return owner; }

    public void setOwner(String owner) { this.owner = owner; }
}
```

```java
public class ElevenStAppStatus {
    private boolean ready;

    public boolean isReady() { return ready; }

    public void setReady(boolean ready) { this.ready = ready; }
}
```

```java
@Version("v1")
@Group("11st.example.com")
public class ElevenStApp extends CustomResource<ElevenStAppSpec, ElevenStAppStatus> implements Namespaced {
}
```

CustomResource 나타내기

- 필수 어노테이션
  - Versions & Group
    - API path를 만드는데에 사용된다.
    - /apis/<group>/<version>
      - (example) /apis/stable.example.com/v1

```
@Version("v1")
@Group("stable.example.com")
public class ExposedApp extends CustomResource<ExposedAppSpec, ExposedAppStatus> implements Namespaced {
}
```

CustomResource 나타내기

- Namespaced
  1. Custom resource의 범위를 결정한다.
  2. 설정하지 않으면 기본으로 cluster 범위가 설정된다.

```
@Version("v1")
@Group("stable.example.com")
public class ExposedApp extends CustomResource<ExposedAppSpec, ExposedAppStatus> implements Namespaced {
}
```

CustomResource 나타내기

- 이 외 자동으로 생성되는 필드 값들에 대해서도 annotation으로 설정가능
- 설정하지 않으면 클래스명을 기반으로 값이 자동으로 설정됨

```
# Generated by Fabric8 CRDGenerator, manual edits might get overwritten!
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: elevenstapps.11st.example.com
spec:
  group: 11st.example.com
  names:
    kind: ElevenStApp
    plural: elevenstapps
    singular: elevenstapp
  scope: Namespaced
  versions:
  - name: v1
    schema:
      openAPIV3Schema:
        properties:
          spec:
            properties:
              appId:
                type: string
              owner:
                type: string
            type: object
          status:
            properties:
              ready:
                type: boolean
            type: object
        type: object
    served: true
    storage: true
    subresources:
      status: {}
```

CRD 리뷰


그룹명
범위
버전
spec
status
Generated by Fabric8 CRDGenerator, manual edits might get overwritten!
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
name: elevenstapps.11st.example.com
spec:
group: 11st.example.com
names:
kind: ElevenStApp
plural: elevenstapps
singular: elevenstapp
scope: Namespaced
versions:
- name: v1
schema:
openAPIV3Schema:
properties:
spec:
properties:
appId:
type: string
owner:
type: string
type: object
status:
properties:
ready:
type: boolean
type: object
type: object
served: true
storage: true
subresources:
status: {}

CR 작성

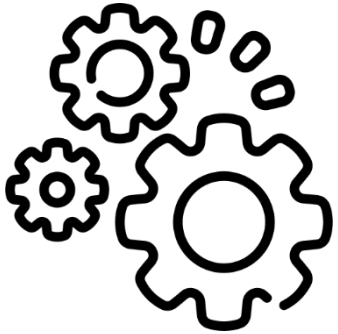
```yaml
apiVersion: "11st.example.com/v1"
kind: ElevenStApp
metadata:
  name: elevenst-service
spec:
  appId: elevenst-service
  owner: core-platform-team
```

개발 과정

1

Custom Resource Definition 생성

2



Operator 제작 (Java Operator SDK)

Java Operator SDK - Reconciler

- 타입 파라미터를 통해 watch 할 리소스를 설정한다.
- Resource에 변경사항이 생길 때 마다 reconcile이 호출된다.

```
public interface Reconciler<R extends HasMetadata> {

    The implementation of this operation is required to be idempotent. Always use the UpdateControl
    object to make updates on custom resource if possible.
    Params: resource – the resource that has been created or updated
            context – the context with which the operation is executed
    Returns: UpdateControl to manage updates on the custom resource (usually the status) after
            reconciliation.
    UpdateControl<R> reconcile(R resource, Context<R> context) throws Exception;

}
```

Java Operator SDK - reconcile

- Resource: reconcile을 트리거 한 리소스
- Context: 현 reconcile에 대한 context 정보

```
public interface Reconciler<R extends HasMetadata> {

    The implementation of this operation is required to be idempotent. Always use the UpdateControl
    object to make updates on custom resource if possible.
    Params: resource – the resource that has been created or updated
            context – the context with which the operation is executed
    Returns: UpdateControl to manage updates on the custom resource (usually the status) after
             reconciliation.
    UpdateControl<R> reconcile(R resource, Context<R> context) throws Exception;

}
```

Reconcile Logic

- **Custom Resource 내용 로깅**
- Deployments 리소스 생성
  - CR에 사용자가 정의한 정보를 활용
- status 변경

```
// CR 관련 내용 로깅
logger.info("Start Reconcile Logic!");
logger.info("CRD name : " + resource.getCRDName());
logger.info("metadata.name : " + resource.getMetadata().getName());
logger.info("spec.label : " + resource.getSpec().getLabel());
logger.info("spec.name : " +resource.getSpec().getName());
```

Reconcile Logic

- Custom Resource 내용 로깅
- **Deployments 리소스 생성**
  - CR에 사용자가 정의한 정보를 활용
- status 변경

```
kubernetesClient.apps()
    .deployments()
    .create(new DeploymentBuilder()
        .withMetadata(new ObjectMetaBuilder()
            .withName(resource.getSpec().getAppId() +"-nginx") // CR에 사용자가 설정한 정보를 사용
            .withLabels(Map.of( k1: "app",  v1: "label"))
            .build())
        .withSpec(new DeploymentSpecBuilder()
            .withReplicas(3)
            .withTemplate(new PodTemplateSpecBuilder()
                .withMetadata(new ObjectMetaBuilder()
                    .withLabels(Map.of( k1: "app",  v1: "nginx"))
                    .build())
                .withSpec(new PodSpecBuilder()
                    .withContainers(new ContainerBuilder()
                        .withName("nginx")
                        .withImage("nginx:1.14.2")
                        .withPorts(new ContainerPortBuilder().withContainerPort(80).build())
                        .build())
                    .build())
                .build())
            .withSelector(new LabelSelectorBuilder()
                .withMatchLabels(Map.of( k1: "app",  v1: "nginx"))
                .build())
            .build())
        .build());
```

Kubernetes Resource 생성하기

- Fabric8 Kubernetes Client를 이용하여 작성
- fluent 인터페이스를 제공하여 쉽게 작성할 수 있다.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

메타데이터 설정

컨테이너 설정

```
kubernetesClient.apps()
    .deployments()
    .create(new DeploymentBuilder()
        .withMetadata(new ObjectMetaBuilder()
            .withName(resource.getSpec().getAppId() +"-nginx")
            .withLabels(Map.of( k1: "app",  v1: "label"))
            .build())
        .withSpec(new DeploymentSpecBuilder()
            .withReplicas(3)
            .withTemplate(new PodTemplateSpecBuilder()
                .withMetadata(new ObjectMetaBuilder()
                    .withLabels(Map.of( k1: "app",  v1: "nginx"))
                    .build())
                .withSpec(new PodSpecBuilder()
                    .withContainers(new ContainerBuilder()
                        .withName("nginx")
                        .withImage("nginx:1.14.2")
                        .withPorts(new ContainerPortBuilder().withContainerPort(80).build())
                        .build())
                    .build())
                .build())
            .withSelector(new LabelSelectorBuilder()
                .withMatchLabels(Map.of( k1: "app",  v1: "nginx"))
                .build())
            .build())
        .build());
```

메타데이터 설정

설정한 정보를 사용

컨테이너 설정

Reconcile Logic

- Custom Resource 내용 로깅
- Deployments 리소스 생성
  - CR에 사용자가 정의한 정보를 활용
- **status 변경**
  - **Custom Resource를 변경하지 않은 경우 UpdateControl.noUpdate()**

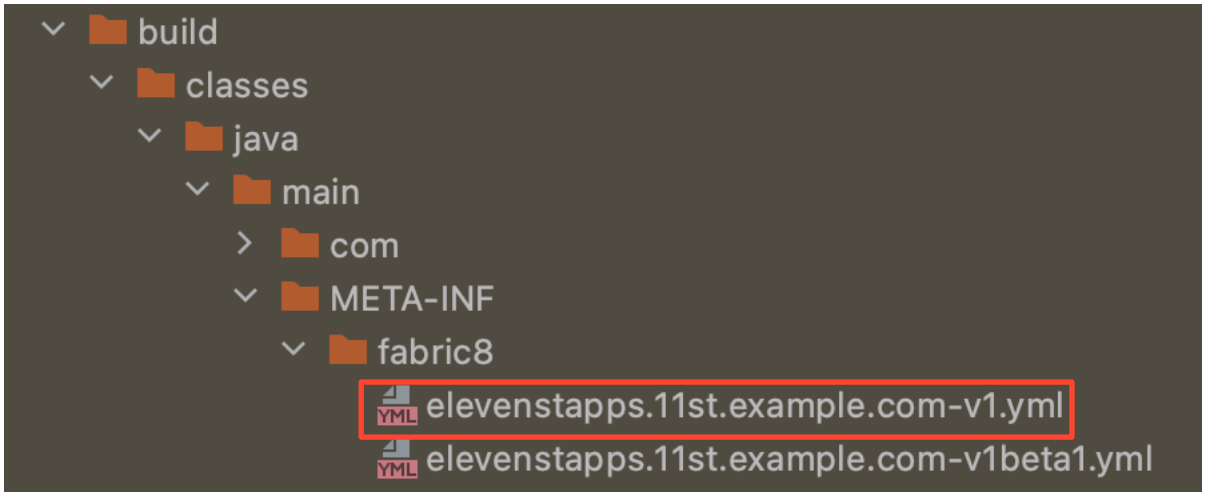
```
var status = new ElevenStAppStatus();
status.setReady(true);
resource.setStatus(status);

return UpdateControl.patchStatus(resource);
```

Operator 실행

1. CRD 생성
2. Operator 실행
3. CR 작성 및 생성

Custom Resource Definition 생성

- crd-generator를 통해 CRD yaml 파일 생성
- 컴파일 시 yaml 파일이 생성됨



컴파일 결과 생성된 yml 파일

- kubectl 명령어를 통한 CRD 생성

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample kubectl apply -f
build/classes/java/main/META-INF/fabric8/elevenstapps.11st.example.com-v1.yml
customresourcedefinition.apiextensions.k8s.io/elevenstapps.11st.example.com cr
eated
```

생성된 Custom Resource Definition 확인

생성 후 kubectl api-resources 커맨드를 통해 custom resource definition이 정상적으로 생성되었는지 확인

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample kubectl api-resources --api-group=11st.example.com
NAME           SHORTNAMES   APIVERSION            NAMESPACED   KIND
elevenstapps                11st.example.com/v1   true         ElevenStApp
```

아직 생성한 리소스는 없는 상태

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample kubectl get elevenstapps
No resources found in default namespace.
```

(참고) 존재하지 않는 리소스의 경우

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample kubectl get notexistresourcenames
error: the server doesn't have a resource type "notexistresourcenames"
```

Operator 실행 후 CR 생성

1. Operator 실행
   (발표) IDE를 이용한 실행
   (실제) Pod 형태로 운영

2. CR 생성

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample/kubernetes kubectl apply -f cr.yml

elevenstapp.11st.example.com/elevenst-service created
```

3. Operator 로그 확인

```
2022-11-14 14:53:45.680  INFO 33220 --- [edappreconciler] c.e.j.reconciler.ExposedAppReconciler    : Start Reconcile Logic!
2022-11-14 14:53:45.680  INFO 33220 --- [edappreconciler] c.e.j.reconciler.ExposedAppReconciler    : CRD name : elevenstapps.11st.example.com
2022-11-14 14:53:45.681  INFO 33220 --- [edappreconciler] c.e.j.reconciler.ExposedAppReconciler    : metadata.name : elevenst-service
2022-11-14 14:53:45.681  INFO 33220 --- [edappreconciler] c.e.j.reconciler.ExposedAppReconciler    : spec.appId : elevenst-service
2022-11-14 14:53:45.681  INFO 33220 --- [edappreconciler] c.e.j.reconciler.ExposedAppReconciler    : spec.owner : core-platform-team
```

생성된 리소스 확인

4. elevenstapps 리소스(CustomResource) 확인

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample/kubernetes kubectl get elevenstapps
NAME               AGE
elevenst-service   88m
```

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample kubectl describe elevenstapps elevenst-service
Spec:
  App Id:  elevenst-service
  Owner:   core-platform-team
Status:
  Ready:  true
```

5. deployments 리소스 확인

```
a1101580@1101580M01 ~/IdeaProjects/java-operator-sample/kubernetes kubectl get deployments
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
eleventst-service-nginx    3/3     3            3           88m
```

마무리

- Operator?
  - 사용자가 정의한 리소스에 따라 동작하는 컨트롤러
- Java Operator SDK를 사용한 Operator 제작
- 실제로는 Java Operator SDK를 사용하여 더 다양한 컨트롤 가능
  - Custom Resource가 아닌 reconcile 로직 내에서 생성된 리소스로 reconcile 호출 등
- 11 번가의 Operator 활용 사례는 **서비스 디스커버리 붙였다 뗐다, 참 쉽죠?** 세션에서 확인 가능

데모 코드

- GitHub
  https://github.com/yujinchoi-94/java-operator-sample

Q&A

E-mail) choi.yujin@11stcorp.com

Thank you