< Custom Controller Operator Pattern >

Operator-SDK

애플리케이션을 패키징, 배포 및 관리하는 것

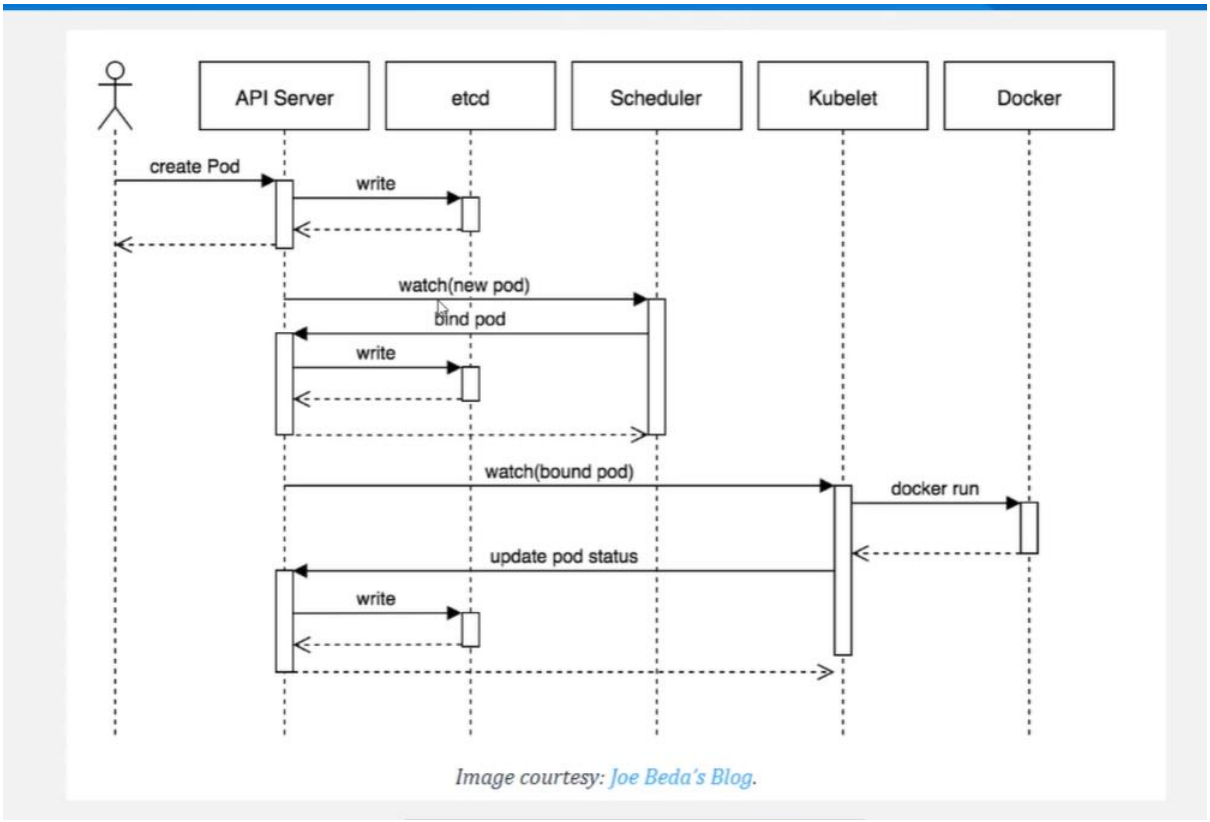
컨트롤러, 커스텀 리소스 생성, 등록을 자동화해 놓은 것

1) CRD(Custom Resource Definition) 란?

직접 리소스의 종류를 정의해서 사용할 수 있음.

2) K8s 컨트롤러

선언형(Declarative) 방식 : 최종적으로 도달해야 하는 **바람직한 상태(Desired State)**를 정의 한 뒤, **현재 상태(Current State)**가 바람직한 상태와 다를 경우 이를 일치하도록 만드는 방법이다.



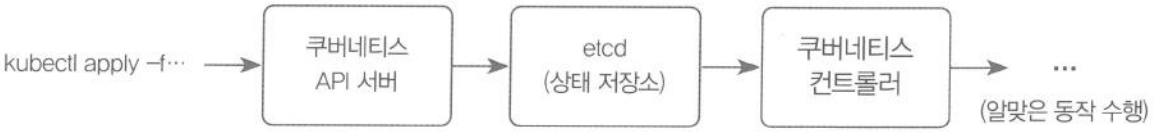
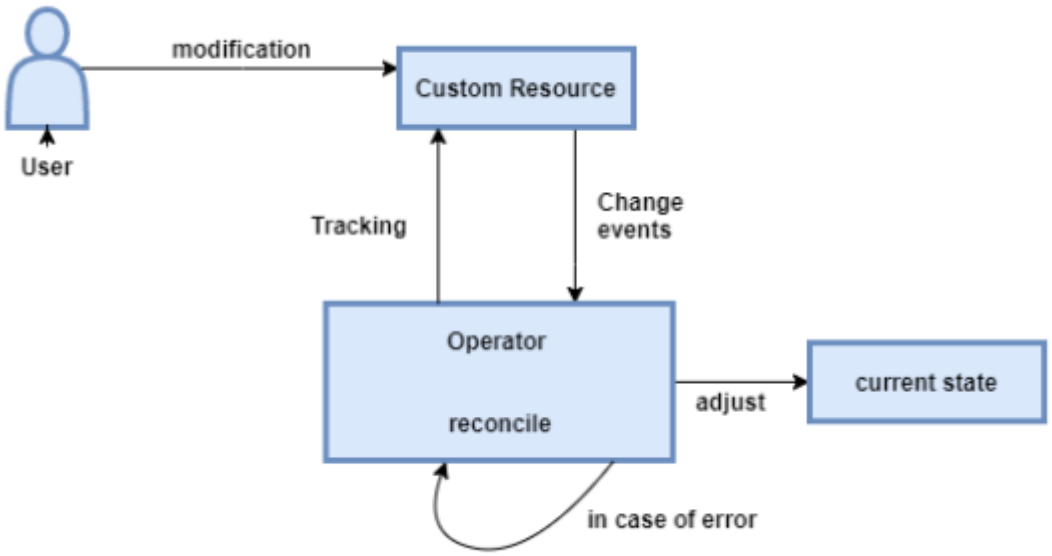
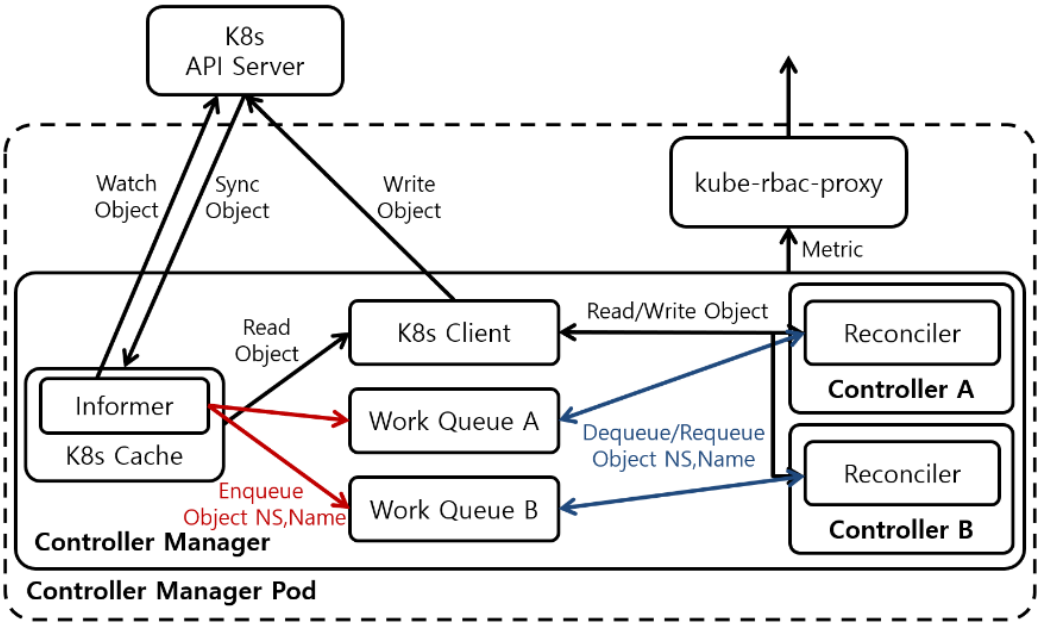


그림 12.2 선언형 방식의 예

3) Operator 패턴 : CRD를 사용할 수 있도록 컨트롤러를 구현하는 방법

4) Reconcile: 현재 상태가 desired State 가 되도록 특정 동작을 수행 하는 것





[그림 1] Controller Manager Architecture

1. Operator-sdk 디렉터리 구조

1) Main.go

Outcluster 방식으로 동작 시킬 수 있도록 파일 생성되어 있음

2) Api / v1

Api/<version>/<kind>_types.go CRD의 API 정의되어 있음

3) Controllers/

컨트롤러 구현하는 디렉터리이다. Controller/<kind>_controller.go 파일을 편집하여 지정된 리소스 유형을 처리하도록 컨트롤러의 Reconsile logic을 정의한다.

4) Confg/bases/~.yaml

CRD <kind>.yaml 파일 위치이다.

5) Config/crd/kustomization.yaml

오브젝트 생성과 설정이 정의되어 있다.

6) Makefile

컨트롤러를 빌드하고 배포하는데 사용한다.

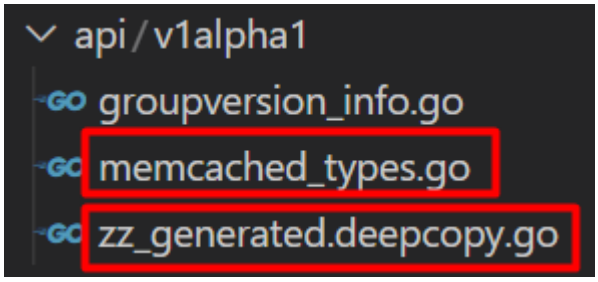
2. Operator-sdk 로 CRD, CR 생성

Github : https://github.com/limes22/operator-sdk

1) Crd spec 정의 (api -> crd)

$ make generate

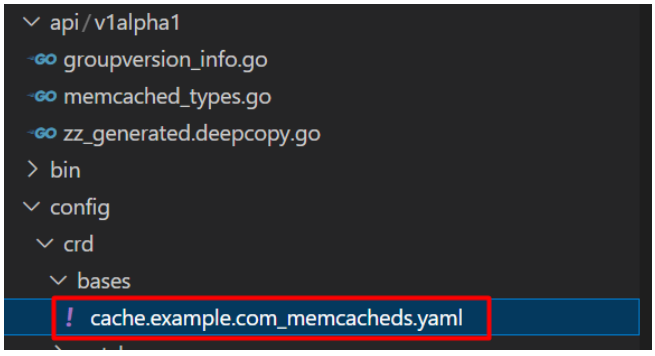
Memcahched_type 파일은 기본적인 내용만 있어서 상세 내용 추가하여 zz_generated.deepcopy.go 파일을 생성한다.



- **CRD 작업 사항 template 코드로 떨구기**

  $make manifests

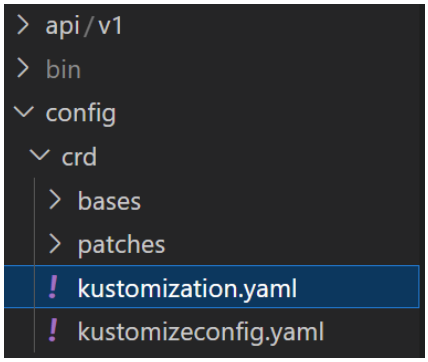
  zz_generated.deepcopy.go 기반으로 config base 파일이 만들어짐



  Crd manifest 파일 생성

- **CRD를 ETCD에 등록**

  $ make install //CRD 생성



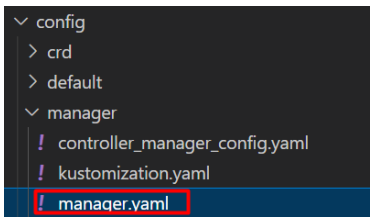
Kustomization 파일을 통해서 CRD 생성됨

Crd 가 api server 에 전달되서 etcd에 저장된다.

Config/crd/bases/<kind>.yaml -> etcd 에 등록

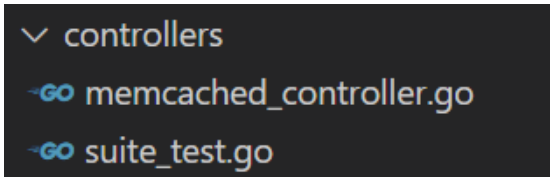
2) Controller 생성

Config 디렉토리에 manager.yaml 파일 내용대로 만들어진다.




```yaml
apiVersion: v1
kind: Namespace
metadata:
  labels:
    control-plane: controller-manager
  name: system
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: controller-manager
  namespace: system
  labels:
    control-plane: controller-manager
spec:
  selector:
    matchLabels:
      control-plane: controller-manager
  replicas: 1
  template:
    metadata:
      annotations:
        kubectl.kubernetes.io/default-container: manager
      labels:
        control-plane: controller-manager
```

Namespace와 deployment 가 선언되어 있다.



Controller.go 파일도 reconcile, createdeployment 함수 추가해서 수정한다.

3) Docker operator's image registry 설정

도커 이미지 이름변경 : docker registry/imagename : tag

$ make docker-build docker-push

4) Operator 실행

- **Outside cluster 모드로 실행**

terminal에 process 개념으로 띄우는 것

$ make install // 아웃클러스터 모드로 디플로이먼트 생성

$ go run main.go //main.go 파일 실행

- **Inside cluster deployment 로 실행**

$ make deploy

5) CR 생성

```yaml
apiVersion: cache.example.com/v1alpha1
kind: Memcached
metadata:
  name: memcached-sample
spec:
  size: 3
```

$ kubectl apply -f config/samples/cache_v1alpha1_memcached.yaml

Reference

1. 오퍼레이터 패턴 - https://kubernetes.io/ko/docs/concepts/extend-kubernetes/operator/
2. Operator-sdk - https://sdk.operatorframework.io/docs/building-operators/golang/tutorial/
3. Operator-sdk - https://sdk.operatorframework.io/docs/overview/project-layout/
4. Operator-sdk - https://sdk.operatorframework.io/docs/building-operators/golang/installation/
5. Kube-api - https://book.kubebuilder.io/cronjob-tutorial/gvks.html