Operator pattern을 활용한 Autoscaler 개발

한양대학교 컴퓨터소프트웨어학부

2018008495 손병재

2018008595 신희원

2023.10.20

목차

Ⅰ. 요약

Operator pattern을 활용하여 Application specific한 autoscaler 개발을 목표로 하고, Operator pattern에 대한 연구 및 애플리케이션 설정 및 개발을 진행하였습니다.

Ⅱ. 배경지식

Operator pattern

CRD를 정의하고, 커스텀 리소스를 생성하는데, 이 커스텀 리소스를 위한 컨트롤러를 구현하여 커스텀 리소스를 제어하는 방식입니다. Operator SDK와 같은 도구를 활용하여 Operator를 개발할 수 있고, 다른 방법으로 저수준의 API를 이해하여 개발하는 방법이 있습니다. 전자가 훨씬 유용한 방법이므로 그것을 활용하였습니다.

애플리케이션

제안서에 작성하였던 Redis Cluster의 Autoscaling을 위한 애플리케이션이 개발되어야 했습니다. 개발해야하는 애플리케이션의 특징으로, 이벤트가 있을 때 트래픽이 폭등하여 캐시 접근이 늘어난다는 조건이 필요하였습니다. 그래서 초기 애플리케이션으로 게시판을 생각하여 개발을 진행하였으나, 다음과 같은 사유로 Redis Cluster에 대한 개발을 포기하게 되었습니다.

- Redis Cluster에 대한 근본적인 지식 부족
- 애플리케이션 복잡도의 증가로 인한 주제 이탈 가능성

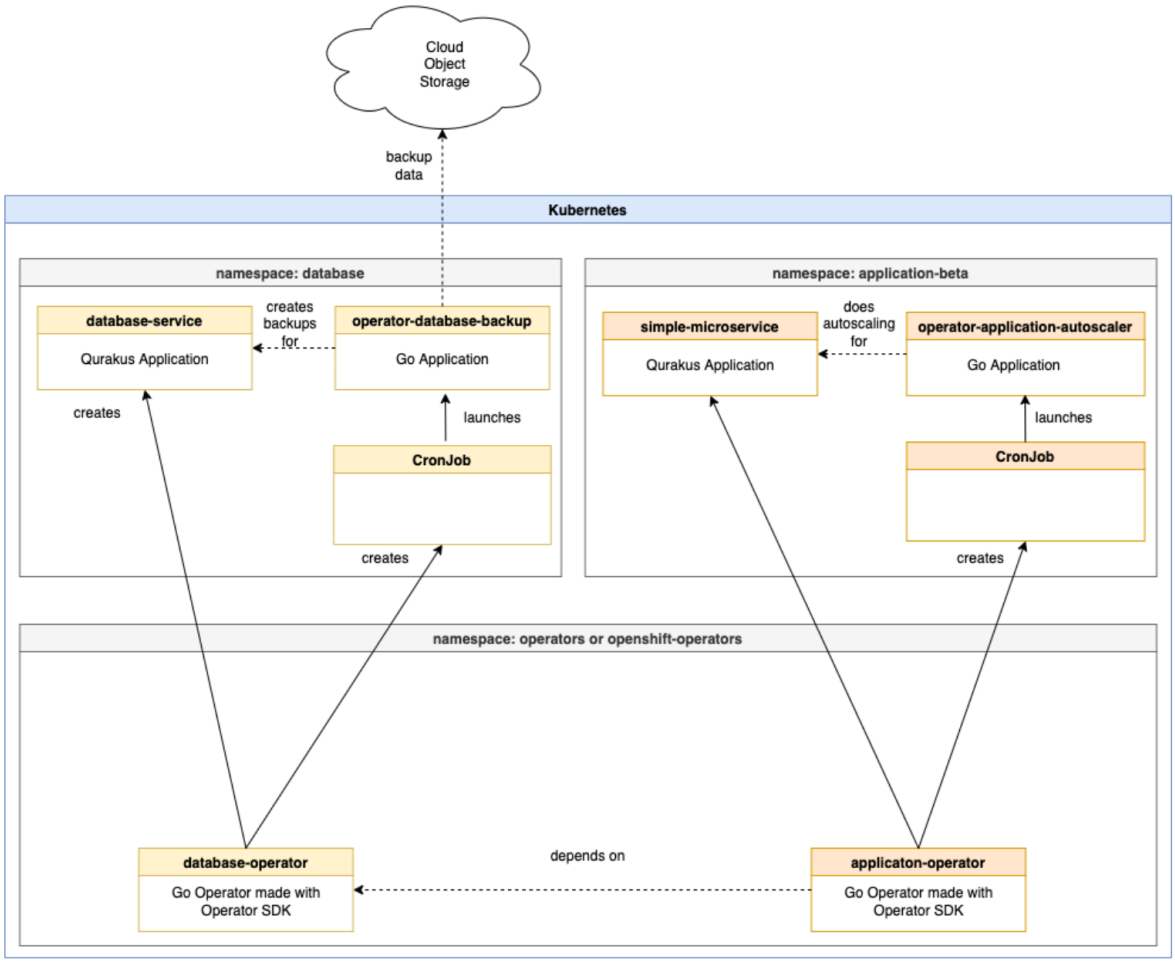
결과적으로 훨씬 단순한 애플리케이션을 기획하고, 오직 Operator pattern을 활용한 Autoscaling 로직 개발에 집중할 수 있도록 하였습니다. 그 결과는 다음과 같습니다.

- 간단한 사용자 인증 서버 개발

- 스케일링 될 서비스

사용자 인증만 확인할 수 있도록 API 호출을 통한 가입, 로그인, 로그아웃이 가능한 서버를 개발하고, 스케일링이 될 서비스는 어떤 것이든 작동하는 애플리케이션이기만 하면 되므로 기존에 데모로 활용하였던 Kiada를 사용하였습니다.

전체 아키텍처



다음 아키텍처에서 application-operator 부분을 참고하여 개발을 진행하였습니다. Go를 사용하지 않고, 익숙한 JAVA를 사용하기 위해 JAVA Operator SDK를 활용하였고, 기본

애플리케이션은 FastAPI를 활용, MySQL에 접근하여 로그인 여부를 파악할 수 있도록 개발하였습니다.

Ⅲ. 개발과정

애플리케이션 개발

사용자 인증 서버는 다음과 같이 구성되어 있습니다.

- 회원 가입 기능
- 로그인 기능
- 로그아웃 기능

이때, Active한 사용자 수를 확인하여 사용자에게 제공될 서비스 파드의 개수를 늘리는 Autoscaling 로직을 사용할 것입니다. 단순히 POST를 통해 모든 사용자 인증이 진행되고, API가 호출될 때 마다 DB를 체크하여 Active한 사용자 수를 확인합니다. 사용자 수와 파드 수의 관계는 다음과 같습니다.

- Active 사용자 수 < 1000 : 2개의 파드
- Active 사용자 수 < 10000 : 3개의 파드
- 이후 20000명 추가될 때마다 1개씩 증가

이 로직을 다음과 같이 적용하여 커스텀 리소스의 상태를 변경할 수 있습니다.

```python
async def __try_change_kiada_autoscalar_spec():
    enabled_user_count = await __aggregate_enabled_user_count()
    if enabled_user_count < 1000:
        target_replicas = MIN_REPLICAS
    elif enabled_user_count < 10000:
        target_replicas = 3
    else:
        additional_users = (enabled_user_count - 10000) // 20000
        target_replicas = 3 + additional_users

    # config.load_kube_config()
    load_incluster_config()
    custom_api = client.CustomObjectsApi()
    kiada_service_count_crd = custom_api.get_namespaced_custom_object(group=group, version='v1', namespace=namespace,
                                                                      plural=plural, name=resource_name)

    kiada_service_count_crd['spec']["kiadaServiceCount"] = target_replicas

    custom_api.replace_namespaced_custom_object(group, version='v1', namespace=namespace, plural=plural,
                                                name=resource_name, body=kiada_service_count_crd)


async def __aggregate_enabled_user_count():
    query = select(func.count(User.id)).where(User.enabled == True)
    enabled_user_count = session.execute(query).scalar()
    return enabled_user_count
```

또한, 기본 DB 설정이 진행되어야 합니다. MySQL에 애플리케이션에서 사용할 DB 이름대로 DB를 생성하고 테이블을 적절하게 구성하여야 합니다.

Operator 및 CRD

JAVA Operator SDK를 활용하여 Operator를 개발하고, 사용된 자바 객체를 CRD로 만들어주는 CRDGenerator를 통해 CRD를 작성합니다. 작성된 CRD는 다음과 같습니다.

Generated by Fabric8 CRDGenerator, manual edits might get overwritten!

apiVersion: apiextensions.k8s.io/v1

kind: CustomResourceDefinition

Metadata:

name: kiadaservicecounts.hanyang.ac.kr

Spec:

group: hanyang.ac.kr

names:

kind: KiadaServiceCount

plural: kiadaservicecounts

```
        singular: kiadaservicecount
    scope: Namespaced
    versions:
    - name: v1
      schema:
        openAPIV3Schema:
            properties:
                spec:
                    properties:
                        owner:
                            type: string
                        kiadaServiceCount:
                            type: integer
                    type: object
                status:
                    properties:
                        ready:
                            type: boolean
                    type: object
            type: object
      served: true
      storage: true
      subresources:
```

```
status: {}
```

다음 yaml 파일을 kubectl apply를 통해 CRD를 생성할 수 있습니다.

Controller는 이 CRD를 통해 생성된 CR의 변경을 감지하여 리컨사일 작업을 수행할 수 있습니다. 이 작업은 크게 세가지로 나뉩니다.

- CR 내용 로깅
- 스케일링
- CR의 Status 변경

Dockerize

모든 애플리케이션이 개발되었다면, 쿠버네티스에 적재할 수 있도록 도커화가 필요합니다. mysql의 경우, 이미 dockerhub에 적재되어 있으므로, 사용자 인증 서버와 서비스, Operator만 도커화시키면 됩니다. Dockerfile을 작성하여 도커화하고, 로컬에서 빌드 후 실행하여 정상적으로 작동하는지 확인할 수 있습니다.

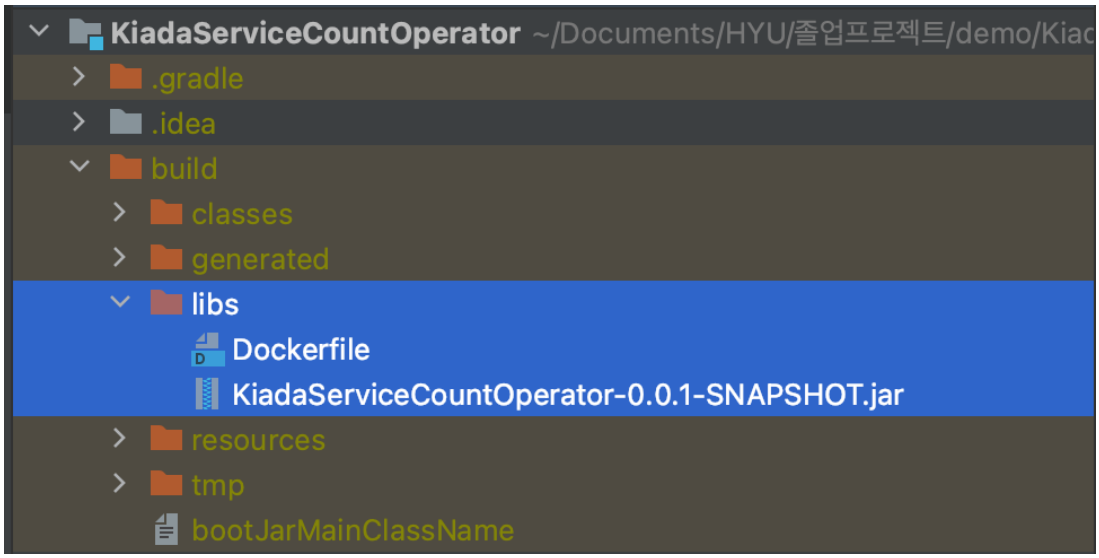
Ⅳ. 실행과정

개발 과정에서 모든 작업이 마무리된 후, 실제로 실행하는 과정을 확인하겠습니다.

1. CRD 생성

```
sonbyeongjae ~/Documents/HYU/졸업프로젝트/demo/KiadaServiceCountOperator
❯ kubectl get kiadaservicecounts
No resources found in default namespace.
```

2. Operator 실행



3. CR 생성

```
INFO 22650 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : Start Reconcile Logic!
INFO 22650 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : CRD name : kiadaservicecounts.hanyang.ac.kr
INFO 22650 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : metadata.name : hanyang-kiada-autoscalar
INFO 22650 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.owner : hanyang-university-graduation-37-team
INFO 22650 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.kiadaServiceCount : 1
```

리컨사일이 로직이 동작하여 초기 kiada deployment의 레플리카 수가 1로 설정되는 것을 확인할 수 있음.

```
sonbyeongjae ~/Documents/HYU/졸업프로젝트/demo/KiadaServiceCountOperator/kubernetes main ±+
kubectl get deployment
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
kiada   1/1     1            1           8h
vote    1/1     1            1           8h
```

4. 회원가입 및 로그인 API 호출

5. 스케일링 확인

```
sonbyeongjae ~/Documents/HYU/분산컴퓨팅/kubernetes-in-action-
kubectl get deploy
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
kiada   2/2     2            2           10h
vote    1/1     1            1           9h
```

6. Locust를 활용한 부하테스트 진행

```
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  15m   deployment-controller  Scaled up replica set kiada-6dc7ff4c97 to 3 from 2
  Normal  ScalingReplicaSet  44s   deployment-controller  Scaled down replica set kiada-6dc7ff4c97 to 2 from 3
```

```
2023-10-19 18:14:41.034  INFO 1 --- [           main] a.k.h.k.KiadaServiceCountOperator       : Starting KiadaServiceCountOperator using Java 17.0.2 on kiada-service-count-operator-59b9b88d85-2clpt with PID 1 (/app.jar started by root in
 /)
2023-10-19 18:14:41.037  INFO 1 --- [           main] a.k.h.k.KiadaServiceCountOperator       : No active profile set, falling back to 1 default profile: "default"
2023-10-19 18:14:42.965  WARN 1 --- [           main] ault ConfigurationService implementation : Configuration for reconciler 'kiadaservicecountreconciler' was not found. Known reconcilers: None.
2023-10-19 18:14:42.967  INFO 1 --- [           main] ault ConfigurationService implementation : Created configuration for reconciler ac.kr.hanyang.kiadaservicecountoperator.reconciler.KiadaServiceCountReconciler with name kiadaservicecou
ntreconciler
2023-10-19 18:14:43.054  INFO 1 --- [           main] io.javaoperatorsdk.operator.Operator     : Registered reconciler: 'kiadaservicecountreconciler' for resource: 'class ac.kr.hanyang.kiadaservicecountoperator.cr.KiadaServiceCount' for n
amespace(s): [all namespaces]
2023-10-19 18:14:43.055  INFO 1 --- [           main] io.javaoperatorsdk.operator.Operator     : Operator SDK 3.0.3 (commit: ab111e7) built on Fri Jun 24 13:23:42 UTC 2022 starting...
2023-10-19 18:14:43.068  INFO 1 --- [           main] io.javaoperatorsdk.operator.Operator     : Client version: 5.12.2
2023-10-19 18:14:43.070  INFO 1 --- [           main] i.j.operator.processing.Controller       : Starting 'kiadaservicecountreconciler' controller for reconciler: ac.kr.hanyang.kiadaservicecountoperator.reconciler.KiadaServiceCountReconci
ler, resource: ac.kr.hanyang.kiadaservicecountoperator.cr.KiadaServiceCount
2023-10-19 18:14:43.666  INFO 1 --- [           main] i.j.operator.processing.Controller       : 'kiadaservicecountreconciler' controller started, pending event sources initialization
2023-10-19 18:14:43.670  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : Start Reconcile Logic!
2023-10-19 18:14:43.670  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : CRD name : kiadaservicecounts.hanyang.ac.kr
2023-10-19 18:14:43.670  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : metadata.name : hanyang-kiada-autoscalar
2023-10-19 18:14:43.671  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.owner : hanyang-university-graduation-37-team
2023-10-19 18:14:43.671  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.kiadaServiceCount : 1
2023-10-19 18:14:43.968  INFO 1 --- [           main] a.k.h.k.KiadaServiceCountOperator       : Started KiadaServiceCountOperator in 3.955 seconds (JVM running for 5.44)
2023-10-19 18:19:53.414  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : Start Reconcile Logic!
2023-10-19 18:19:53.414  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : CRD name : kiadaservicecounts.hanyang.ac.kr
2023-10-19 18:19:53.414  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : metadata.name : hanyang-kiada-autoscalar
2023-10-19 18:19:53.415  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.owner : hanyang-university-graduation-37-team
2023-10-19 18:19:53.415  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.kiadaServiceCount : 2
2023-10-19 21:10:29.742  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : Start Reconcile Logic!
2023-10-19 21:10:29.748  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : CRD name : kiadaservicecounts.hanyang.ac.kr
2023-10-19 21:10:29.749  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : metadata.name : hanyang-kiada-autoscalar
2023-10-19 21:10:29.749  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.owner : hanyang-university-graduation-37-team
2023-10-19 21:10:29.749  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.kiadaServiceCount : 3
2023-10-19 21:25:31.043  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : Start Reconcile Logic!
2023-10-19 21:25:31.045  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : CRD name : kiadaservicecounts.hanyang.ac.kr
2023-10-19 21:25:31.045  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : metadata.name : hanyang-kiada-autoscalar
2023-10-19 21:25:31.045  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.owner : hanyang-university-graduation-37-team
2023-10-19 21:25:31.045  INFO 1 --- [countreconciler] a.k.h.k.r.KiadaServiceCountReconciler    : spec.kiadaServiceCount : 2
```

전체 로그를 확인하여 오토 스케일링이 정상적으로 진행됐음을 확인할 수 있습니다.

추가사항

쿠버네티스 외부 Metrics를 제외하고 HPA에서 사용할 수 있는 지표는 다음과 같습니다.

- Resource
- Pods
- Object

이 중에서, Pod의 메트릭에 HTTP Request 등이 있습니다. Prometheus는 메트릭을 수집하는 역할을 할 뿐, 실제 데이터는 애플리케이션 레벨에서 생성되어야 합니다.

Ⅶ. 참고문헌

Java operator sdk: https://github.com/operator-framework/java-operator-sdk

Operator pattern: https://ibm.github.io/operator-sample-go-documentation/