3.3 HPA

❖HPA(Horizontal Pod Autoscaler)란?

- 수평적 파드 오토스케일러로써 pod의 갯수를 자동으로 확장, 축소시키는 기능을 제공함
- 사용하는 기반 지표 예
  - Default : CPU 사용률
  - 확장 : 메모리 또는 사용자 정의 지표(예: Prometheus 에서 수집된 지표)
- 사전 조건
  - k8s-metrics-server 가 설치되어 있어야 함
  - 대상 리소스는 Deployment, Replicaset, Statefulset 이어야 함
  - 리소스 요구사항이 정의되어 있어야 함
    - CPU, Memory에 대한 requests 가 설정되어 있어야 함
    - limits 만 있고 requests가 설정되지 않은 경우는 오작동할 수 있음

3.3 HPA

❖HPA 테스트

- CPU 부하를 발생시키는 Deployment 배포

```
## hpa-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hpa-deploy
spec:
  replicas: 1       # 시작 : replicas 1
  selector:
    matchLabels:
      app: hpa-deploy
  template:
    metadata:
      labels:
        app: hpa-deploy
```

```
    spec:
      containers:
      - name: hpa-deploy
        image: vish/stress   # CPU 부하생성 이미지
        args:
          - -cpus
          - "1"              # 1 CPU를 지속 사용
        resources:
          requests:
            cpu: 100m
          limits:
            cpu: 200m
```

2.9 HPA

- HPA 리소스 생성

```
## hpa.yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-test
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: hpa-deploy
  minReplicas: 1
  maxReplicas: 8
  metrics:
  - type: Resource
    resource:
      name: cpu
      # 타겟이 50에 도달하면 replicas 가 확장됨
      target:
        type: Utilization
        averageUtilization: 50
```

3.3 HPA

- 예제 적용

```
## deployment 배포
kubectl apply -f hpa-deploy.yaml

## Pod 1개 생성된 것 확인
kubectl get pods

## HPA 리소스 적용
kubectl apply -f hpa.yaml

## Pod 목록 지속 확인하여 Pod 동적 생성 확인. 중단은 CTRL+C
watch kubectl get pods -l app=hpa-deploy

## HPA 상태 실시간 모니터링
watch kubectl get hpa

## 테스트 완료 후 리소스 정리
kubectl delete -f hpa.yaml
kubectl delete -f hpa-deploy.yaml
```

3.4 VPA

❖VPA(Vertical Pod Autoscaler)란?

- Pod가 사용하는 CPU와 메모리의 사용량을 자동으로 조절해주는 기능
- VPA는 HPA와는 달리 Pod의 갯수는 유지하면서 리소스 요구사항(requests, limits) 값을 조정함
- 리소스를 너무 적게 또는 너무 많이 요구하는 Pod를 최적화하여 리소스 낭비를 줄이고 안정성을 높이도록 함

❖작동 방식

- metrics-server 와 Recommender가 CPU, 메모리 사용량을 수집
- 일정 시간 동안의 데이터를 기반으로 추천 리소스 용량(requests, limits) 계산
- Updater가 리소스 조정이 필요한 Pod를 재시작하도록 하여 새로운 리소스 반영

❖VPA updateMode

- Auto : 리소스를 자동 조정하고 pod도 자동 재시작
- Initial : 처음 Pod를 생성할 때만 추천값 적용
- Off : 권장값을 추천만 할 뿐 실제 적용하지 않음(리소스 낭비를 분석하기 위한 용도)

3.4 VPA

❖VPA 구성요소 설치

- recommender, updater, Admission Controller

```
## Git Clone 후 Shell Script 실행
git clone https://github.com/Kubernetes/autoscaler.git
cd autoscaler/vertical-pod-autoscaler/
./hack/vpa-up.sh

## 설치 구성요소 확인
## 3가지 구성요소 : vpa-admission-controller, vpa-recommender, vpa-updater
kubectl get deployment -n kube-system

## Pod가 정상 실행중인지 확인
kubectl get pods -n kube-system | grep vpa

## 디렉토리를 작업 경로로 다시 이동
cd ../..
```

3.4 VPA

❖VPA 테스트

- 테스트용 Deployment
  - kubectl apply -f vpa-deploy.yaml

```
## vpa-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vpa-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      app: vpa-deploy
  template:
    metadata:
      labels:
        app: vpa-deploy
    spec:
      containers:
      - name: stress
        image: vish/stress
        args:
        - -cpus
        - "1"
        ## 초기설정값 확인
        resources:
          requests:
            cpu: "100m"
            memory: "128Mi"
          limits:
            cpu: "200m"
            memory: "256Mi"
```

3.4 VPA

- VPA 리소스 생성
  - kubectl apply -f vpa.yaml

```
## vpa.yaml
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: vpa-demo
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind: Deployment
    name: vpa-deploy
  updatePolicy:
    updateMode: "Auto"
```

- VPA가 추천하는 값을 확인
  - kubectl describe vpa vpa-demo 명령실행
  - 출력 결과에서 Recommendation 영역을 확인
    - 지속적으로 실행하면 CPU, 메모리 추천값이 증가하는 것을 확인할 수 있음. 약간의 시간(2-3분)이 소요됨

3.4 VPA

- Pod 재시작과 조정된 리소스 확인 : 약간의 시간 소요
  - kubectl get pods -l app=vpa-deploy --watch
  - kubectl get pods -o json | jq '.items[].spec.containers[].resources'

```
stepano@laptop:~/k8s/vpa$ kubectl get pods -o json | jq '.items[].spec.containers[].resources'
{
  "limits": {
    "cpu": "1174m",
    "memory": "500Mi"
  },
  "requests": {
    "cpu": "587m",
    "memory": "262144k"
  }
}
```

  - 재시작할지 여부는 vpa updater가 판단하는데 추천값이 현재의 requests 값과 현저하게 차이가 날 때 재시작
  - 막연하게 기다리기 힘들면 다음 명령을 실행해 Pod의 갯수를 늘려볼 것
    - kubectl scale deployment vpa-deploy --replicas=2
    - 신규로 생성되는 Pod에 추천값 적용됨을 확인

3.4 VPA

❖ 리소스 정리

- kubectl delete -f vpa.yaml
- kubectl delete -f vpa-deploy.yaml
- cd autoscaler/vertical-pod-autoscaler/
- ./hack/vpa-down.sh