

---

# Kubernetes in Action

## chapter-15 포드와 클러스터 노드의 오토스케일링

Sunggon Song

---

# 소개

- ❑ 포드에서 실행되는 애플리케이션은 확장 가능한 리소스 (리플리케이션 컨트롤러, 리플리카셋, Deployment, ...)의 복제본 개수를 늘려서 수동으로 스케일 가능
- ❑ 쿠버네티스 포드 및 노드의 오토스케일
  - ❑ CPU 사용량, 다른 측정 항목의 증가를 감지하면 포드를 모니터링하고 자동으로 스케일
  - ❑ 클라우드 인프라스트럭처에서 실행 중인 경우 기존 노드가 더이상 포드를 허용할 수 없는 경우 추가 노드를 스펀업

## 15.1. 수평 포드 오토 스케일링

- 오토스케일링 프로세스
  - CPU 활용률에 따른 스케일링
  - 메모리 사용량에 따른 스케일링
  - 기타 및 사용자 지정 메트릭을 기반으로 스케일링
  - 오토스케일링에 적합한 메트릭 결정
-

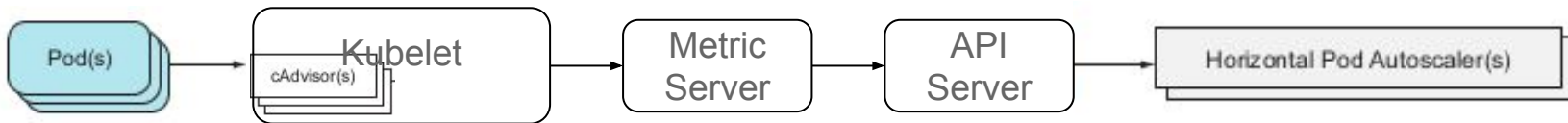
## 15.1.1. 오토스케일링 프로세스

Horizontal 컨트롤러가 HPA(HorizontalPodAutoscaler) 리소스를 만들어 컨트롤러가 관리하는 포드 복제본의 수를 자동으로 조정

- ❑ POD 메트릭 가져오기
- ❑ 필요한 포드 수 계산하기
- ❑ 확장가능한 리소스에서 원하는 복제본 개수 필드를 업데이트
- ❑ 전체 오토스케일링 프로세스 이해

## 15.1.1.1. POD 메트릭 가져오기

- ❑ cAdvisor가 수집한 포드 및 노드 메트릭을 힙스터를 통해서 획득



## 15.1.1.1. Kubelet 리소스 메트릭 API

- ❑ kubelet은 각 포드를 구성 컨테이너로 변환하고 컨테이너 런타임 인터페이스를 통해 개별 컨테이너 사용 통계를 가져옴
  - ❑ 기존 도커 통합의 경우, cAdvisor에서 이 정보를 가져옴
- ❑ Kubelet은 리소스 메트릭 API(/metrics/resource/v1alpha1)를 통해 집계 된 pod 리소스 사용 통계를 노출

# kubectl top

대부분의 **CPU** 및 메모리를 사용하는 포드를 볼 수 있음

- 현재 **cpu, memory** 사용량 만 표시

```
$ kubectl top pod --all-namespaces
NAMESPACE NAME CPU(cores) MEMORY(bytes)
kube-system kube-proxy-gke-rel3170-default-pool-3459fe6a 2m 12Mi
kube-system kube-proxy-gke-rel3170-default-pool-3459fe6a 2m 12Mi
kube-system fluentd-gcp-v2.0.9-5t9q6 8m 85Mi
kube-system fluentd-gcp-v2.0.9-pd4s9 10m 84Mi
kube-system kube-dns-3468831164-v2gqr 1m 26Mi
kube-system event-exporter-v0.1.7-1642279337-180db 0m 13Mi
kube-system kube-proxy-gke-rel3170-default-pool-3459fe6a 1m 12Mi
kube-system l7-default-backend-3623108927-tjm9z 0m 1Mi
kube-system kube-dns-3468831164-cln0p 1m 25Mi
kube-system fluentd-gcp-v2.0.9-sj3rh 9m 84Mi
kube-system kube-dns-autoscaler-244676396-00btn 0m 7Mi
kube-system kubernetes-dashboard-1265873680-8prcm 0m 18Mi
kube-system heapster-v1.4.3-3980146296-33tmw 0m 42Mi
```

## 15.1.1.2. 필요한 포드 수 계산하기

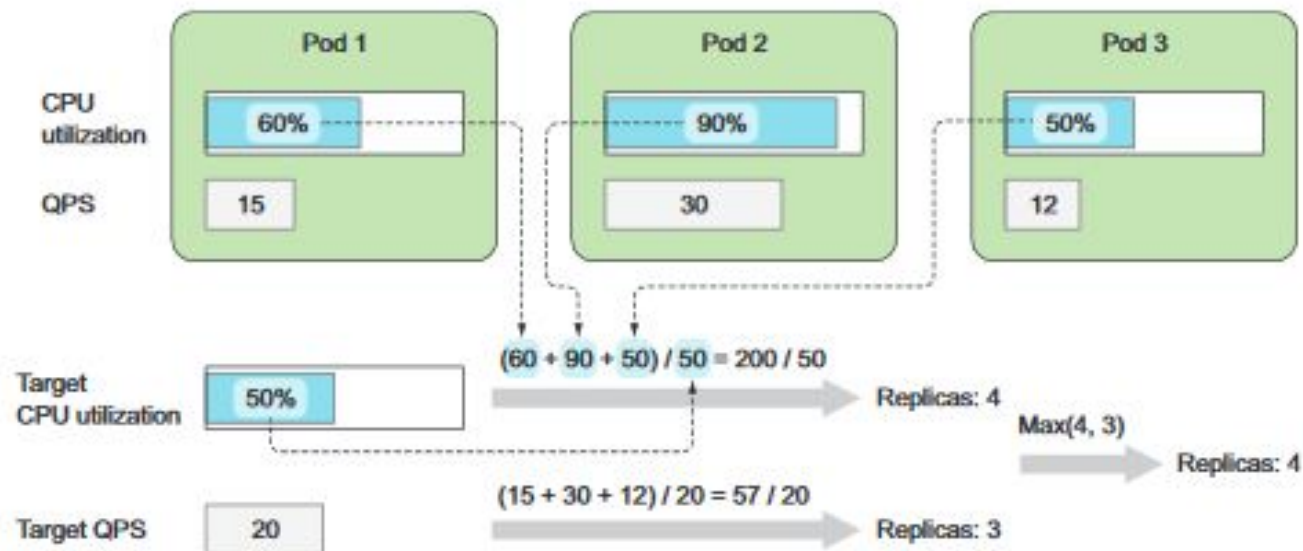
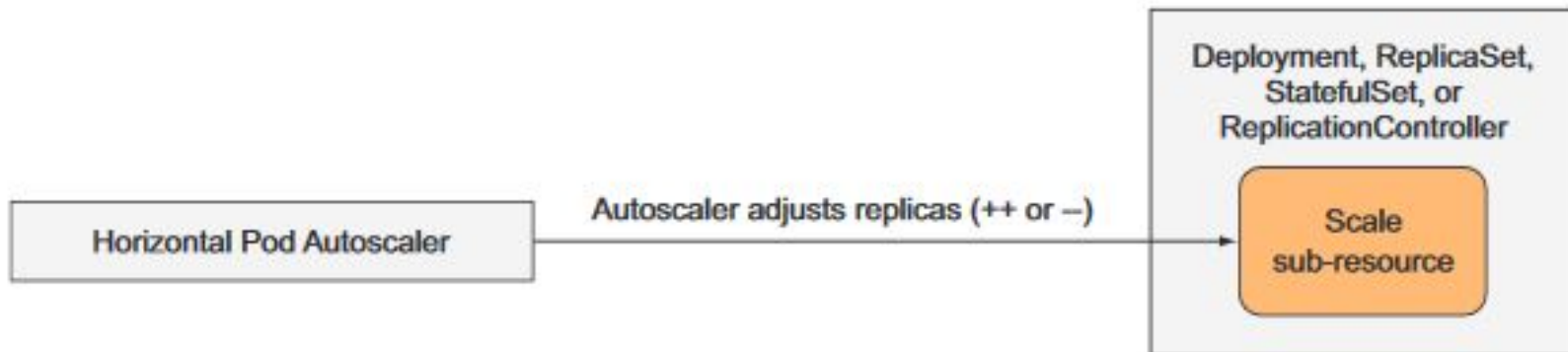


Figure 15.2 Calculating the number of replicas from two metrics



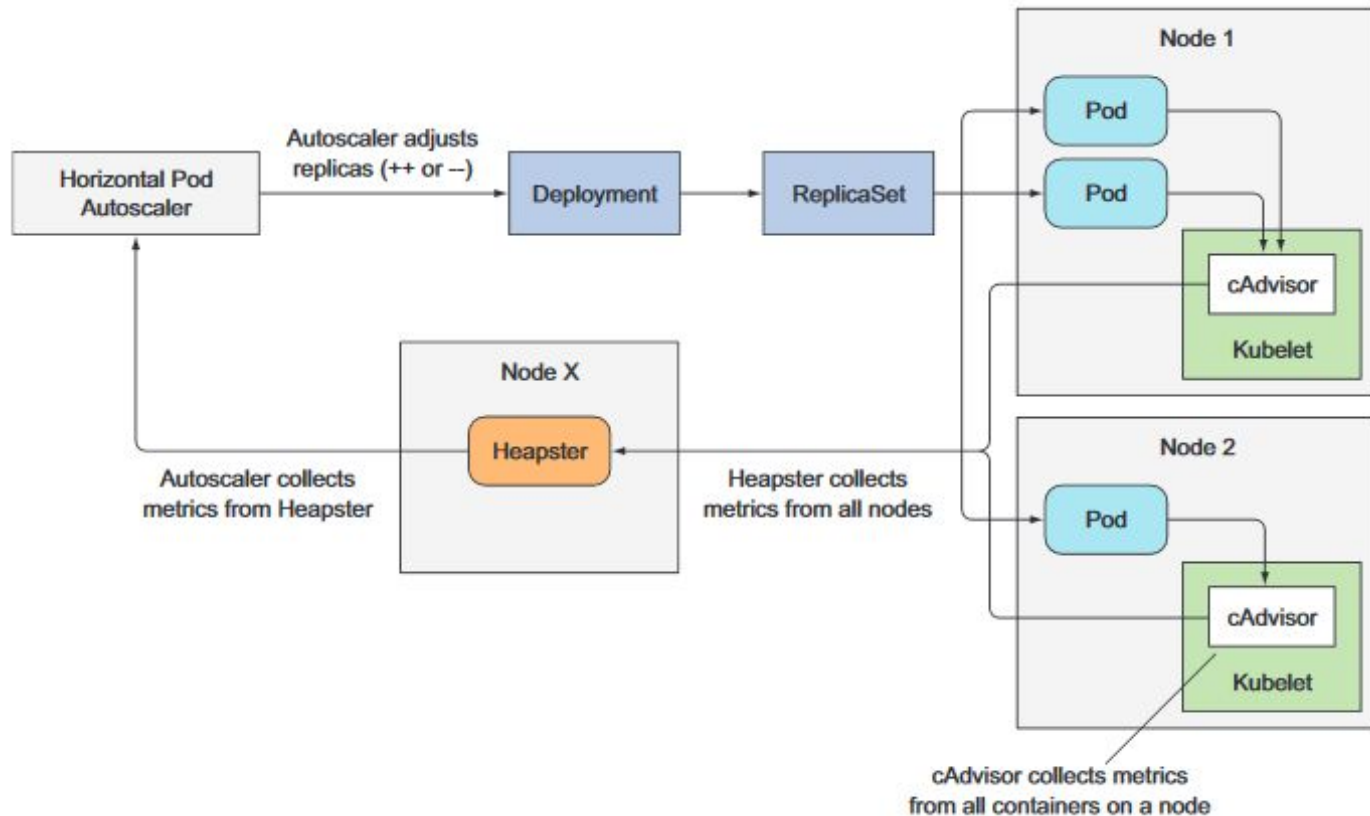
### 15.1.1.3. 확장가능한 리소스에서 원하는 복제본 개수 필드를 업데이트

- ❑ HPA는 Scale sub-resource를 통해 확장가능한 리소스의 복제본 개수 필드를 수정
- ❑ 확장가능한 리소스
  - ❑ 디플로이먼트, 레플리카셋, 리플리케이션 컨트롤러, 스테이트풀셋



## 15.1.1.4. 전체 오토스케일링 프로세스 이해

- ❑ 메트릭 데이터의 흐름
  - ❑ POD -> cAdvisor -> Heapster -> HPA
- ❑ 복제본수 조정 흐름
  - ❑ HPA -> Deployment-> ReplicaSet -> POD



오토스케일러가 메트릭을 얻고 디플로이먼트를 확장하는 방법

# 쿠버네티스 모니터링 아키텍처

## ❑ 핵심 메트릭 파이프라인

- ❑ Kubelet(cAdvisor) - metric server(slimmed-down heapster) - API server - HPA controller
- ❑ 시스템 메트릭을 기반으로하는 스케줄러와 HPA와 같은 핵심 시스템 구성 요소 (그리고 kubectl top)에서 사용

## ❑ 모니터링 파이프 라인

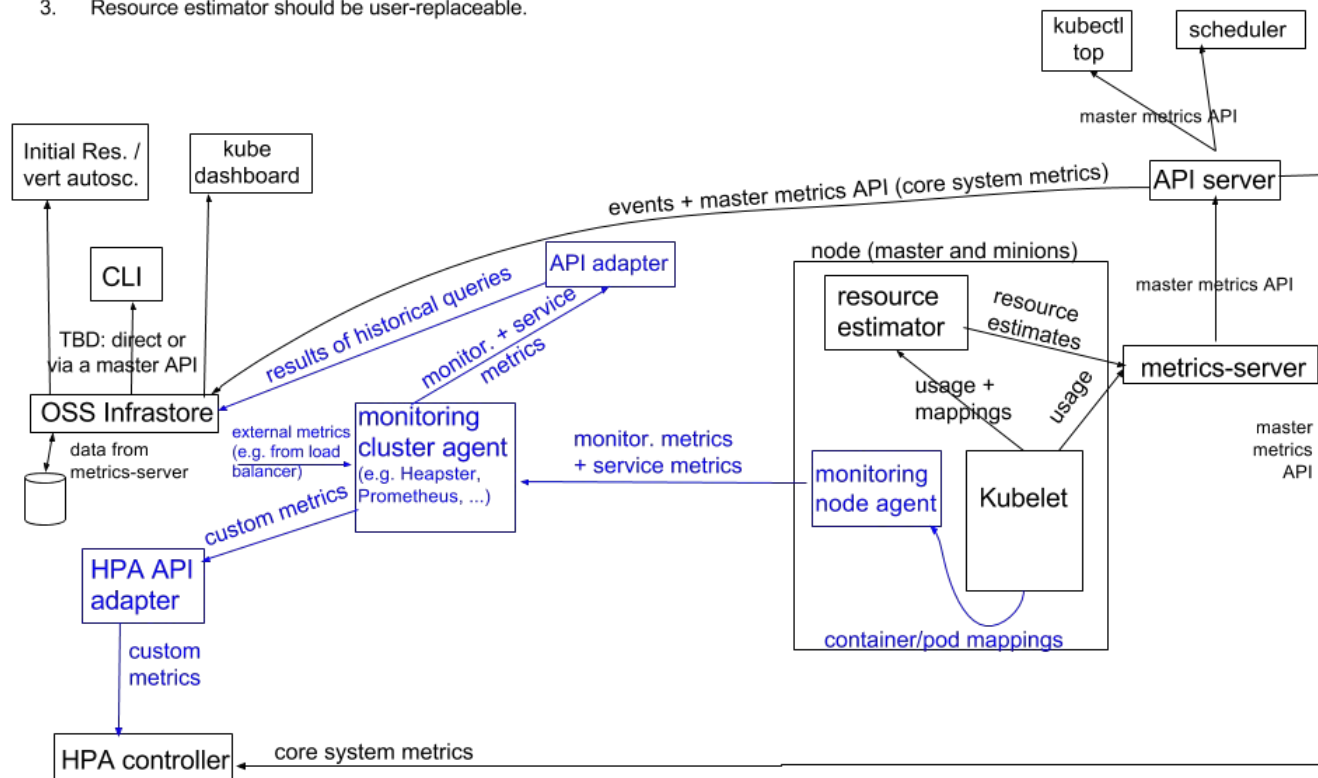
- ❑ 시스템에서 다양한 메트릭을 수집하여 End-User, 인프라스토어(via adapters) 및 HPA(사용자 정의 메트릭에 대해서)에 노출하는데 사용
- ❑ 여러 모니터링 시스템 공급업체(프로메테우스, 힙스터 등) 중 하나를 선택
  - ❑ 쿠버네티스는 모니터링 파이프 라인을 제공하지 않음
  - ❑ 노드 단위 에이전트와 클러스터-레벨 aggregator로 구성

## Monitoring architecture proposal: OSS

(arrows show direction of metrics flow)

### Notes

1. Arrows show direction of metrics flow.
2. **Monitoring pipeline is in blue.** It is user-supplied and optional.
3. Resource estimator should be user-replaceable.



## 15.1.2. CPU 활용률에 따른 스케일링

- 갑작스런 피크부하를 처리할 수 있는 충분한 여유를 확보하려면 target CPU 사용량 90%를 넘지 않도록 해야 함
- 포드는 CPU사용률 백분율을 오토스케일러가 결정할 수 있도록 CPU의 요청을 설정해야 함
- 오토스케일러는 포드의 실제 CPU 소비량과 CPU 요구량을 비교

## 15.1.2. CPU 활용률에 따른 스케일링

- ❑ CPU 사용량에 기반한 HorizontalPodAutoscaler 생성
- ❑ 첫 번째 오토 리스케일 이벤트 보기
- ❑ 스케일업 트리거
- ❑ 오토스케일러가 디플로이먼트를 확장하는 것 보기
- ❑ 스케일링의 최대 비율 이해
- ❑ 기존 HPA object에서 타겟 metric 값 변경

## 15.1.2.1. CPU 사용량에 기반한 HPA 생성

### ❑ 디플로이먼트의 Pod템플릿에 CPU리소스 요청을 추가

```
❑ Deployment.yaml
  ❑ Spec:
    ❑ Replicas: 3
    ❑ Template:
      ❑ Metadata:
        ❑ ...
      ❑ Spec:
        ❑ Containers:
          ❑ -image: luksa/kubia:v1
            ❑ Name: nodejs
            ❑ Resources:
              ❑ Requests:
                ❑ Cpu: 100m
```

### ❑ HPA 객체를 생성하고 디플로이먼트를 가리켜야 한다.

```
❑ $ kubectl autoscale deployment kubia --cpu-percent=30 --min=1 --max=5
Deployment "kubia" autoscaled
```



# HPA yaml 정의

```
$ Kubectl get hpa.v2beta1.autoscaling kubia -o yaml
Kind: HorizontalPodAutoscaler
Metadata:
  Name: kubia
  ...
  Spec:
    maxReplicas: 5
    Metrics:
      Resource:
        Name: cpu
        targetAverageUtilization: 30
      Type: Resource
    minReplicas: 1
    scaleTargetRef:
      apiVersion: extension/v1beta1
      Kind: Deployment
      Name: kubia
    Status:
      currentMetrics: []
      currentReplicas: 3
      desiredReplicas: 0
```

## 15.1.2.2. 첫 번째 오토 리스케일 이벤트 보기

❏ \$ Kubectl get hpa

❏	NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS
❏	Kubia	Deployment/kubia	<unknown> / 30%	1	5	0

❏ \$ kubectl get deployment

❏	NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
❏	Kubia	1	1	1	1	23m

### 15.1.2.2. 첫 번째 오토 리스케일 이벤트 보기

```
$ kubectl describe hpa
```

 Name: kubia



 Reference: [Deployment/kubia](#)

Metrics: (current / target)


## Resource cpu on pods

📄 (as a percentage of request): 0%(0) / 30%

Min replicas: 1

Max replicas: 5

## Events:

 **From** **Reason** **Message**

.....

```
horizontal-pod-autoscaler    SuccessfulRescale      New size: 1; reason : All metrics  
                                below target
```

## 15.1.2.3. 스케일업 트리거

❑ POD를 서비스를 통해서 노출

❑ `$ kube expose deployment kubia --port=80 --target-port=8080`

❑ Service “kubia” exposed

❑ 병렬로 다수의 리소스 감시

❑ `$ watch -n 1 kubectl get hpa,deployment`

❑ Every 1.0s: kubectl get hpa,deployment

❑

❑	NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
❑	Kubia	Deployment/kubia	0% / 30%	1	5	0	45m

❑

❑	NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
❑	Kubia	1	1	1	1	56m

## 15.1.2.3. 스케일업 트리거

### ❑ 부하 생성기 포드

```
❑ $ kubectl run -it --rm --restart=Never loadgenerator --image=busy/box  
❑ -- sh -c "while true; do wget -O - -q http://kubia.default; done"
```

❑ --rm 옵션: 사용 후 포드 삭제

❑ --restart=Never 옵션: 디플로이먼트 객체를 통하지 않고 직접 관리되지 않는 포드 생성

## 15.1.2.4. 오토스케일러가 디플로이먼트를 확장하는 것 보기.

- ❑ CPU부하가 30%를 초과하지 않으면 추가 부하 생성기 실행  
컨테이너의 CPU 사용률 : 실제 CPU사용량을 요청한 CPU로 나눈 값

❑ \$ kubectl describe hpa

❑ Name: kuba  
❑ ...  
❑

❑ **Events:**

❑ From	Reason	Message
❑ ----	-----	-----

h-p-a	SuccessfulRescale	New size: 1; reason : All metrics below target
-------	-------------------	--

h-p-a	SuccessfulRescale	New size: 4; reason : Cpu resource utilization (percentage of request) above target
-------	-------------------	--

## 15.1.2.5. 스케일링의 최대 비율 이해

- 단일 스케일업 단계에서 복제본 수를 최대 2배까지만 스케일아웃.
- 이전의 오토스케일링 작업 이후에 후속 오토스케일링 작업을 얼마나 빨리 수행할 수 있는지에 대한 제한이 존재
- 최근 3분 이내에 리스케일 이벤트가 발생하지 않은 경우만 스케일아웃을 수행하고, 스케일다운은 5분간격으로 덜 자주 수행됨

## 15.1.2.6. 기존 HPA object에서 타겟 metric 값 변경

❑ **kubectl edit**를 이용해 CPU 사용률 목표를 수정할 수 있음

❑ `$ Kubectl edit hpa.v2beta1.autoscaling kuba`

❑ ...

❑ Spec:

❑ maxReplicas: 5

❑ Metrics:

❑ Resource:

❑ Name: cpu

❑ targetAverageUtilization: 60

❑ Type: Resource



### 15.1.3. 메모리 사용량에 따른 스케일링

- 메모리 스케일업 후 오래된 포드가 어떻게 든 메모리를 해제해야 함
  - 메모리 해제 작업은 실 메모리를 사용 주체인 앱에서 수행해야 함
  - 시스템이 할수 있는일은 앱을 재시작하는것이지만 앱이 다시 이전과 같이 많은 메모리를 사용한다면 오토스케일러는 다시 스케일업을 수행한다. 이러한 과정이 계속 반복될수 있음
- 메모리 사용량 기반 오토스케일은 CPU사용률 기반 오토스케일링보다 훨씬 문제가 많음

## 15.1.4. 기타 및 사용자 지정 메트릭을 기반으로 스케일링

HPA 메트릭 필드에서 사용할 메트릭을 두 개 이상 정의할 수 있음

```
❑ Spec:
  ❑ maxReplicas: 5
    ❑ Metrics:
      ❑ Resource:
        ❑ Name: cpu
        ❑ targetAverageUtilization: 30
      ❑ Type: Resource
```

- ❑ RESOURCE METRIC TYPE
- ❑ PODS METRIC TYPE
- ❑ OBJECT METRIC TYPE

## 15.1.4.1. RESOURCE METRIC TYPE 이해

- 리소스 메트릭 유형은 오토스케일러의 베이스를 컨테이너의 리소스 요청에 지정된 것과 같은 리소스 메트릭에 대한 오토 스케일 결정을 만든다.

## 15.1.4.2. PODS METRIC TYPE 이해

- ❑ 포드 메트릭 유형은 포드와 관련된 다른 메트릭을 직접 참조하는데 사용된다.
  - ❑ 예를들어, QPS(Query Per Second) 또는 메시지 브로커 대기열의 메시지 수(메시지 브로커가 포드에 실행 중일 때)
- ❑ Spec:
  - ❑ maxReplicas: 5
    - ❑ Metrics:
      - ❑ Resource:
        - ❑ metricName: qps
        - ❑ targetAverageValue: 100
      - ❑ Type: Pods

## 15.1.4.3. OBJECT METRIC TYPE 이해

- ❑ 객체 메트릭 유형은 오토스케일링의 스케일 포드를 해당 포드와 직접 관련이 없는 메트릭을 기반으로 만들려는 경우에 사용
  - ❑ 예를들어, 잉레스 객체와 같은 다른 클러스터 객체의 메트릭에 따라 포드의 크기를 조정할 수 있다.
- ❑ Spec:
  - ❑ maxReplicas: 5
    - ❑ Metrics:
      - ❑ Resource:
        - ❑ metricName: latencyMillis
        - ❑ Target:
          - ❑ apiVersion: extensions/v1beta1
          - ❑ Kind: Ingress
          - ❑ Name: frontend
        - ❑ targetValue: 20
      - ❑ Type: Object

## 15.1.5. 오토 스케일링에 적합한 메트릭 결정하기

- ❑ 메모리 사용량같은 경우 오토스케일링을 위한 좋은 메트릭이 아니다.
  - ❑ 복제본수를 늘려도 측정중인 메트릭 평균값이 선형적으로 감소하지 않아 오토스케일링이 제대로 동작되지 않는다.
- ❑ QPS는 복제본 수를 늘리면 항상 QPS가 비례해 감소한다.
- ❑ 오토스케일러를 앱의 사용자 지정 메트릭에 따라 결정하기 전에 포드 수가 증가하거나 감소할 때 그 메트릭 값이 어떻게 작동할지 생각해야 한다.

## 15.1.6. 복제본을 0으로 축소하기

- ❑ Idling 및 un-idling
  - ❑ 며칠에 한 번만 요청을 받는 서비스를 실행하면 항상 실행되도록하고 다른 포드에서 사용할 수 있는 리소스를 점유해서는 안된다.
  - ❑ 그러나 고객의 요청이 발생하면 즉시 해당 서비스를 사용할수 있기를 원할것이다.
- ❑ HPA의 minReplicas필드를 0 으로 설정한다면,이 경우 포드가 시작될 때까지 첫 번째 요청을 처리하는 데 지연이 있음

## 15.2. 수직 포드 자동 스케일링

- 리소스 요청 자동 구성
- 포드가 실행되는 동안 리소스 요청 수정



## 15.2. 포드의 수직적 오토스케일링

- 수평으로 확장할 수 없는 애플리케이션이라면....
- 수직적 스케일링을 하여 더 많은 메모리와 CPU를 제공.

## 15.2.1. 리소스 요청 자동 구성

- ❑ 리소스 요청이 없는 새 포드가 작성되면 `InitialResources` 플러그인은 포드 컨테이너의 히스토리 리소스 사용 데이터를 보고 이에 따라 요청을 설정
- ❑ 리소스 요청을 지정하지 않고 포드를 배치하고 쿠버네티스를 사용해 결국 각 컨테이너의 리소스가 필요한 것을 파악할 수 있음
- ❑ 예를 들어, 컨테이너의 메모리가 부족한 경우 다음 번에 컨테이너 이미지가 있는 포드를 만들 때 메모리의 리소스 요청이 자동으로 높게 설정된다.

## 15.2.2. 포드가 실행되는 동안 리소스 요청 수정

- ❑ 포드가 실행되는 동안 수직으로 스케일링
- ❑ **VerticalPodAutoscaler**
  - ❑ 1.12.6 이상 버전의 모든 클러스터에서 사용

```
apiVersion: autoscaling.k8s.io/v1beta2
kind: VerticalPodAutoscaler
metadata:
  name: my-rec-vpa
spec:
  targetRef:
    apiVersion: "extensions/v1beta1"
    kind:      Deployment
    name:      my-rec-deployment
  updatePolicy:
    updateMode: "Off"
```

## 15.3. 클러스터 노드의 수평적 스케일링

- 클러스터 오토-스케일러 소개
- 클러스터 오토-스케일러  
활성화
- 클러스터 축소 도중의 서비스  
중단 제한

## 15.3.1. 클러스터 오토-스케일러 소개

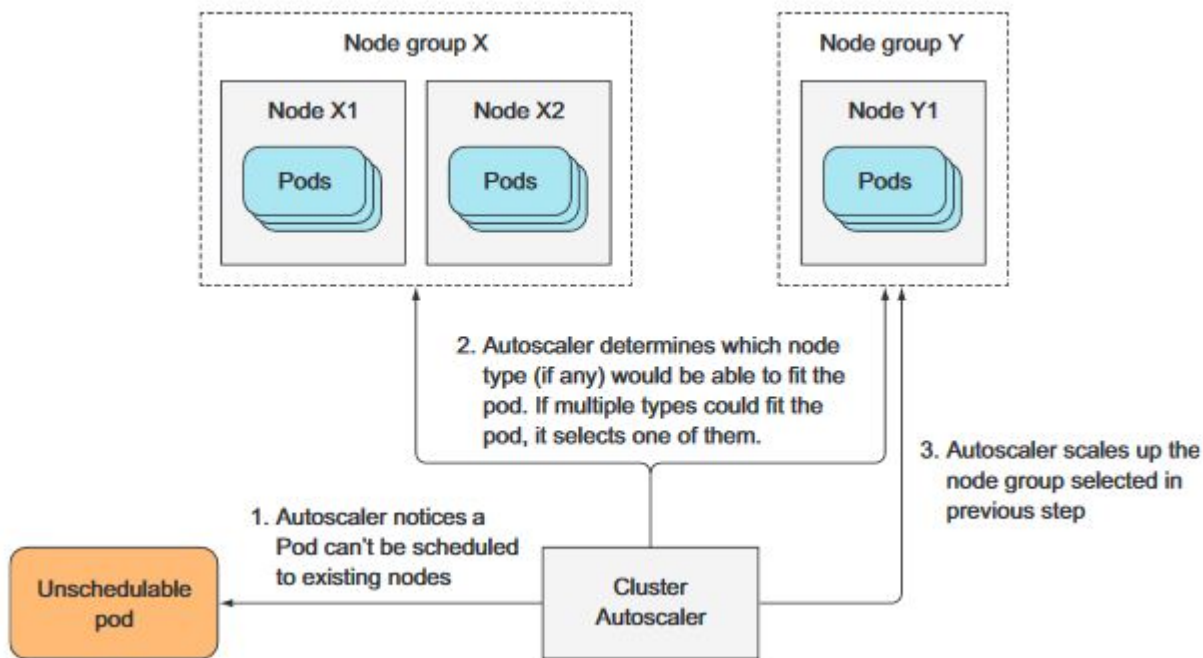
클러스터 자동 스케일러는 추가 노드가 필요함을 감지하자마자 자동으로 클라우드 공급자에 추가 노드를 요청함

- ❑ 클라우드 인프라 스트럭처에서 추가 노드 요청
- ❑ 노드 종료
- ❑ 수동으로 노드를 정리하고 배출

## 15.3.1. 클라우드 인프라 스트럭처에서 추가 노드 요청

- ❑ 클러스터 오토스케일러는 스케줄할 수 없는 포드를 확인하고 클라우드 제공 업체에게 추가 노드를 시작하라고 요청
- ❑ 새 노드가 포드를 수용할 수 있는지 여부를 확인
- ❑ 새 노드 그룹이 여러개이면 포드에 맞는 노드 유형을 선택

## 15.3.1. 클라우드 인프라 스트럭처에서 추가 노드 요청



## 15.3.1.1. 노드 종료

- ❑ 클러스터 오토스케일러는 충분히 활용되지 않을 때 노드 수를 줄여 스케일다운
  - ❑ CPU, Memory 요청이 50% 미만이면 불필요한 것으로 간주
  - ❑ 대상 노드에서 실행중인 시스템 포드 없는지 확인
  - ❑ 관리되지 않는 포드나 로컬 스토리지가 있는 포드가 노드에서 실행 중이지 않는지 경우
  - ❑ 노드를 종료하도록 선택하면 대상노드를 스케줄링 불가로 표시되고 실행 중인 모든 포드가 제거됨



## 15.3.1.2. 수동으로 노드를 정리하고 배출

- ❑ 노드를 예약 할 수없는 것으로 표시하고 수동으로 배출 할 수 있음
  - ❑ `$ kubectl cordon <node>`
    - ❑ 노드를 스케줄 할 수없는 것으로 표시 (그러나 해당 노드에서 실행중인 포드는 아무 것도하지 않습니다).
  - ❑ `$ kubectl drain <node>`
    - ❑ 노드를 스케줄 할 수없는 것으로 표시 한 다음 노드에서 모든 Pod을 제거합니다.
- ❑ 두 경우 모두 `kubectl uncordon <node>`을 사용하여 다시 `uncordon` 할 때까지 노드에 새 pod가 예약되지 않음

## 15.3.2. 클러스터 오토-스케일러 활성화

### ❑ 클러스터 오토스케일링 기능 사용가능

#### ❑ Google Kubernetes Engine (GKE)

- ❑ `$ gcloud container clusters update kubia --enable-autoscaling --min-nodes=3 --max-nodes=5`

#### ❑ Google Compute Engine (GCE)

- ❑ Kubeup.sh를 실행하기전 다음 3개의 환경변수를 설정해야 함

- ❑ `KUBE_ENABLE_CLUSTER_AUTOSCALER=true`
- ❑ `KUBE_AUTOSCALER_MIN_NODES=3`
- ❑ `KUBE_AUTOSCALER_MAX_NODES=5`

#### ❑ Amazon Web Services (AWS)

#### ❑ Microsoft Azure

- ❑ 클러스터 자동 스케너는 kube-system 네임 스페이스의 cluster-autoscaler-status ConfigMap에 상태를 게시합니다.

Reference : <https://github.com/kubernetes/autoscaler/tree/master/cluster-autoscaler>

## 15.3.3. 클러스터 축소 도중의 서비스 중단 제한

- ❑ 클러스터 자동 스케일러 또는 작업자가 자발적으로 노드를 종료 한 경우 작업이 추가 기능을 통해 해당 노드에서 실행되는 포드가 제공하는 서비스를 중단시키지 않도록 할 수 있습니다.
- ❑ 특정 서비스는 최소 개수의 포드가 항상 실행되도록 요구합니다.
- ❑ `$ kubectl create pdb kubia-pdb --selector=app=kubia --min-available=3`  
`poddisruptionbudget "kubia-pdb" created`
- ❑ `$ kubectl get pdb kubia-pdb -o yaml`  
`apiVersion: policy/v1beta1`  
`kind: PodDisruptionBudget`  
`Metadata:`  
 `name: kubia-pdb`  
`Spec:`  
 `minAvailable: 3`  
 `Selector:`  
 `matchLabels:`  
 `app: kubia`

---

## 15.4. Summary

- 
- ❑ 포드의 자동 수평 스케일링을 구성하는 것은 수평적인 "Pod-Autoscaler" 개체를 만들고 이를 "Deployment", "ReplicaSet" 또는 "Replication-Controller"로 가리키고 Pod의 목표 CPU 활용률을 지정하는 것만큼 쉽다.
  - ❑ HPA가 Pod의 CPU 사용률을 기반으로 스케일을 수행하도록 하는 것 외에도, 클러스터에 배포된 다른 개체와 관련된 사용자 지정 메트릭을 기준으로 스케일하도록 구성할 수 있다.
  - ❑ VerticalPodAutoscaler 사용 가능
  - ❑ Kubernetes 클러스터가 지원되는 클라우드 제공자를 실행하면 클러스터 노드도 자동으로 확장될 수 있다.
  - ❑ Pod에서 일회성 프로세스를 실행하면 "-it"와 "--rm" 옵션이 있는 "kubectl" 실행을 사용하여 Ctrl+C를 누르는 즉시 자동으로 포드를 중지하고 삭제할 수 있다.
-



# Referece

<https://livebook.manning.com/#!/book/kubernetes-in-action/chapter-2/24>