



리소스 관리

리소스 제한

- 쿠버네티스에서는 컨테이너 단위로 리소스 제한 설정이 가능
- 서비스에 맞는 성능을 내기 위해서도 리소스 제한 설정은 필요
- 제한이 가능한 리소스는 CPU, Memory, Ephemeral 스토리지지만 DevicePlugin 사용
→ GPU 등 다른 리소스에 대해서도 제한 설정 가능

CPU 와 메모리 제한

- CPU는 클럭 수로 지정하지 않고 1vCPU(가상 CPU)를 1,000m(millicores) 단위로 지정
- 3GHz CPU라고 해도 1코어 정도를 지정할 경우 3,000m가 아닌 1,000m (메모리는 1G = 1000M)
- 리소스 제한은 파드 정의 내부의 각 컨테이너 정의 부분에 포함
- spec.containers[].resources의 requests.cpu/requests.memory 또는 limits.cpu/limits.memory를 지정하는 형태로 제한

```
# sample-resource yaml

apiVersion: apps/v1
kind:Deployment
metadata:
  name: sample-resource
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
      labels:
        app: sample-app
    spec:
      containers:
      - name: nginx-container
        image: nginx
        resources:
          requests:
            memory: "1024Mi"
            cpu: "500m"
          limits:
            memory: "2048Mi"
            cpu: "100m"
```

- Requests는 사용하는 리소스 최솟값을 지정
- 빈 노드에 Requests로 지정 한 양의 리소스가 존재하지 않으면 스케줄링되지 않음
- Limits는 사용할 리소스의 최댓값을 지정
- Requests와 달리 노드에 Limits로 지정한 리소스가 남아 있지 않아도 스케줄링
- Requests 리소스 양 < Limits 리소스 양이지만 Requests와 Limits의 차이가 아주 큰 컨테이너에서 부하가 상승하면 실제로 사용하려는 리소스 양과 차이가 있으므로 주의

```
# sample-resource-only-requests.yaml
apiVersion: apps/v1
kind: Pod
metadata:
  name: sample-resource-only-requests
spec:
  containers:
  - name: nginx-container
    image: nginx
    resources:
      requests:
        memory: "256Mi"
        cpu: "200m"

# 리소스 설정 확인
kubectl get pod sample-resource-only-requests -o json
```

Ephemeral 스토리지 리소스 제어

- 일반적으로 스토리지는 용량 문제나 영속성 관점에서 영구 볼륨 사용해 데이터를 쓰는 것이 바람직
- 컨테이너 재기동 시 삭제돼도 상관없는 데이터라면 container 내부의 디스크 영역 사용 가능

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-ephemeral-storage
spec:
  containers:
  - name: nginx-container
    image: nginx:1.17
    resources:
      requests:
        ephemeral-storage: "128Mi"
      limits:
        ephemeral-storage: "256Mi"

# 파드에 파일 기록
kubectl exec -it sample-ephemeral-storage -- dd if=/dev/zero of=/dummy bs=1M count=1024

# 리소스 실시간 확인
kubectl get pods --watch
```

- emptyDir은 파드에서 공유하고 사용이 가능해 여러 파드에 설정된 limits 합계 초과한 경우 Evict

시스템에 할당된 리소스와 Eviction Manager

- CPU / Mem / Ephemeral Storage의 일반적 리소스 고갈 시 k8s 동작이나 노드에 영향을 미침
 - 각 노드에는 kube-reserved, systme-reserved 라는 두 가지 리소스가 시스템 용으로 확보됨
 - kube-reserved : k8s 시스템 구성 요소나 컨테이너 런타임에 확보된 리소스
 - system-reserved: OS에 깊이 관련된 데몬 등에 확보된 리소스
 - 실제 할당 가능한 양은 노드의 전체 리소스 중 위 두 리소스 제외한 영역

```
# 노드 리소스 양과 할당 가능한 양 확인하는 명령어
kubectl get nodes -o
custom-columns="NAME:.metadata.name, CPU Capacity:.status.capacity.cpu, CPU
Allocatable:.status.allocatable.cpu, Mem Capacity:.status.capacity.memory, Mem
Allocatable:.status.allocatable.memory"
```

NAME	CPU_CAP	CPU_ALLOC	MEM_CAP	MEM_ALLOC
master	8	8	4004224Ki	3901824Ki
worker1	8	8	4004224Ki	3901824Ki
worker2	8	8	4004216Ki	3901816Ki

- k8s 내부에서는 Eviction Manager가 Allocatable, system-reserved, kube-reserved에서 실제로 사용되는 리소스 합계가 초과하지 않는지 확인하고, 초과 된 경우 파드 Evict
- Eviction Manager가 Evict 하는 파드 우선순위
 - requests에 할당된 양보다 초과해 리소스를 소비하고 있는 것
 - PodPriority가 더 낮은 것
 - requests에 할당된 양보다 초과하여 리소스를 소비하고 있는 리소스 양이 더 많은 것

GPU 등의 리소스 제한

- Device Plugins 기능이 지원돼 GPU나 그 외 리소스도 requests나 limits 이용해 제한 설정 가능

```
apiVersion: apps/v1
kind:Deployment
metadata:
  name: sample-resource
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-app
  template:
    metadata:
```

```
  labels:
    app: sample-app
spec:
  containers:
  - name: nginx-container
    image: nginx
    resources:
      requests:
        memory: "1024Mi"
        cpu: "500m"
      limits:
        memory: "2048Mi"
        cpu: "1000m"
```

```
kubectl scale deployment --replicas 6 sample-resource
```

```
kubectl get deployments
```

```
kubectl describe node worker2
```

하나의 파드에 여러 컨테이너가 존재하는 경우

- 여러 컨테이너 초기화 컨테이너를 가진 경우 스케줄링 될 때 파드 단위로 수행
- 필요한 리소스는 여러 컨테이너를 기반으로 계산
- 필요한 리소스 양은 모든 컨테이너 리소스 합계 또는 모든 초기화 컨테이너 리소스 최댓 값 중 큰 쪽 선택