# 19. Resource Control

# Resource Request 및 Limit

- CPU 및 메모리 에 대해 리소스 설정 가능
- CPU 는 하이퍼쓰레딩 된 가상 코어 단위

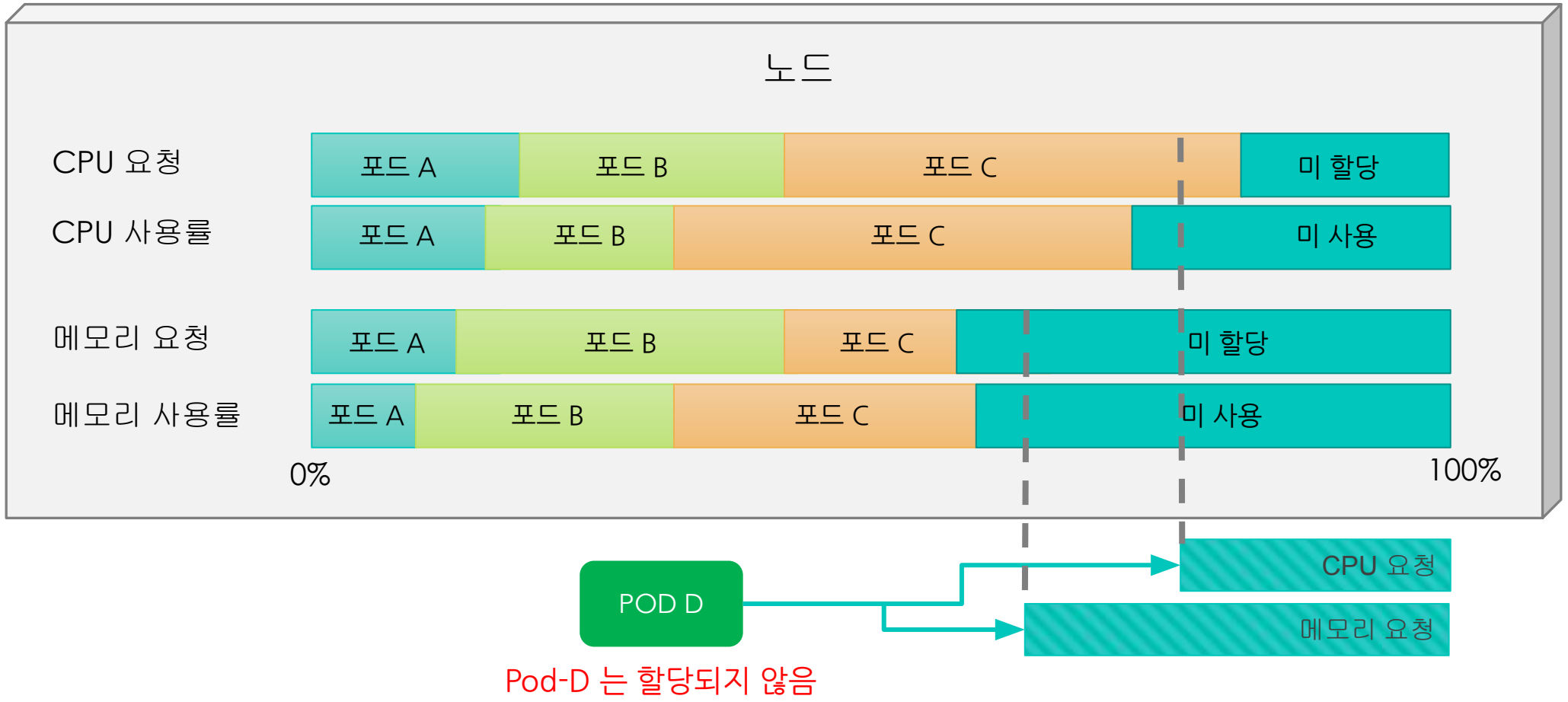
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
  - name: wp
    image: wordpress
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

리소스 요청

리소스 한계

# Pod 스케줄링 시 리소스 고려 사항

- 스케줄러는 요청 사항의 합해서 미 할당된 리소스를 계산함
- 실제 사용량은 여유가 있어도 아래와 같이 Pod가 할당 되지 않을 수 있음



# 노드의 유효한 리소스 량 확인하기

- 노드에 describe 명령으로 리소스 용량 확인 가능
- 스케줄러는 Allocatable 항목만 확인해서 스케줄링 을 수행함

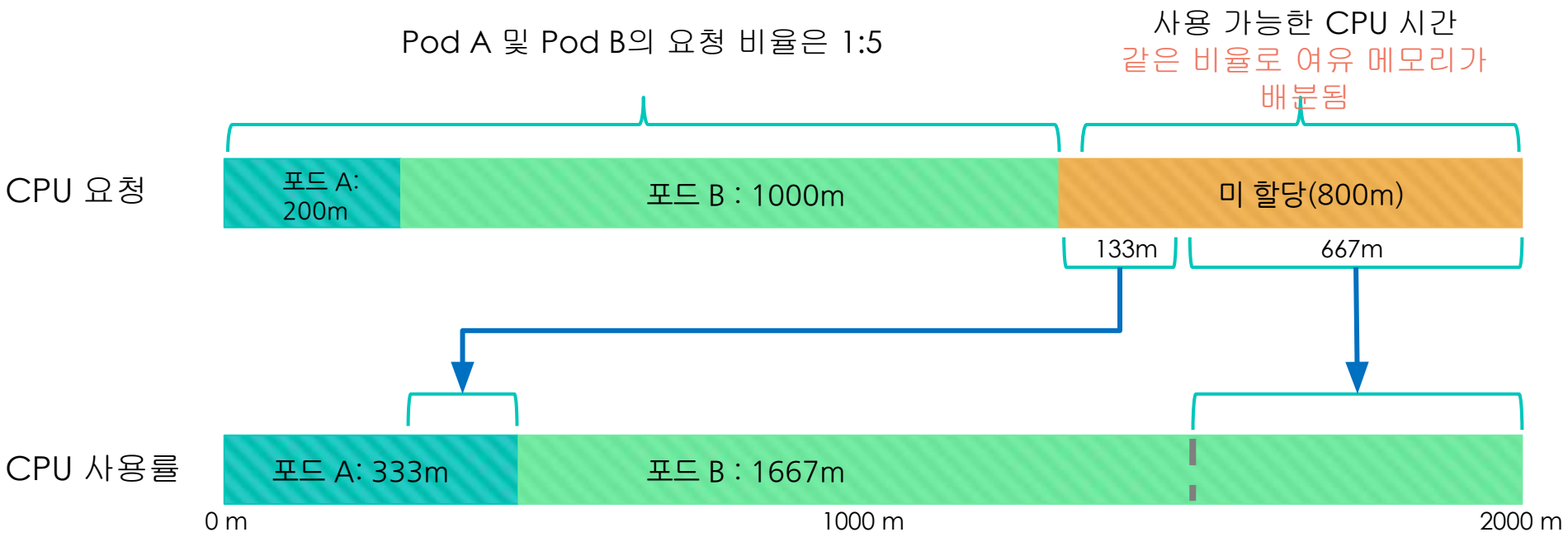
$ Kubectl describe nodes

```
Capacity:
  cpu:                2
  ephemeral-storage:  10252564Ki
  hugepages-2Mi:      0
  memory:             2040816Ki
  pods:               110
Allocatable:
  cpu:                2
  ephemeral-storage:  9448762967
  hugepages-2Mi:      0
  memory:             1938416Ki
  pods:               110
```

스케줄러는 Allocatable 만 확인 함

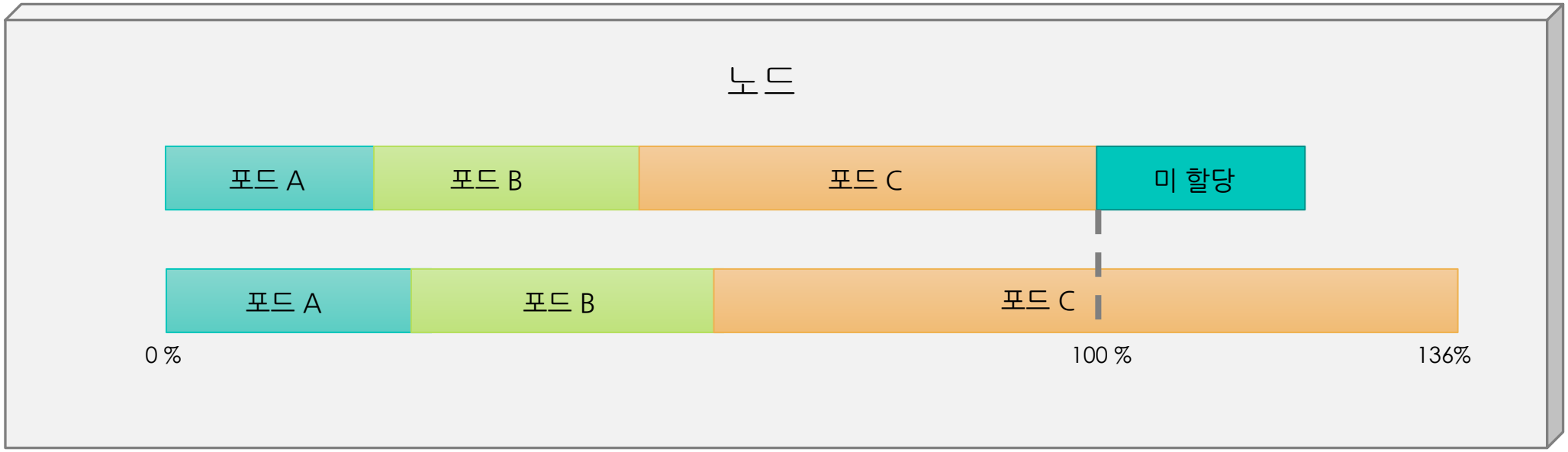
# 미 할당 리소스에 대한 분배

- 미 할당된 CPU 시간은 요청 하는 POD 에게 기존 요청한 비율만큼 추가 할당 가능 합니다.
- 단, limit 항목 이전 까지만 가능함



# Limit 값 초과

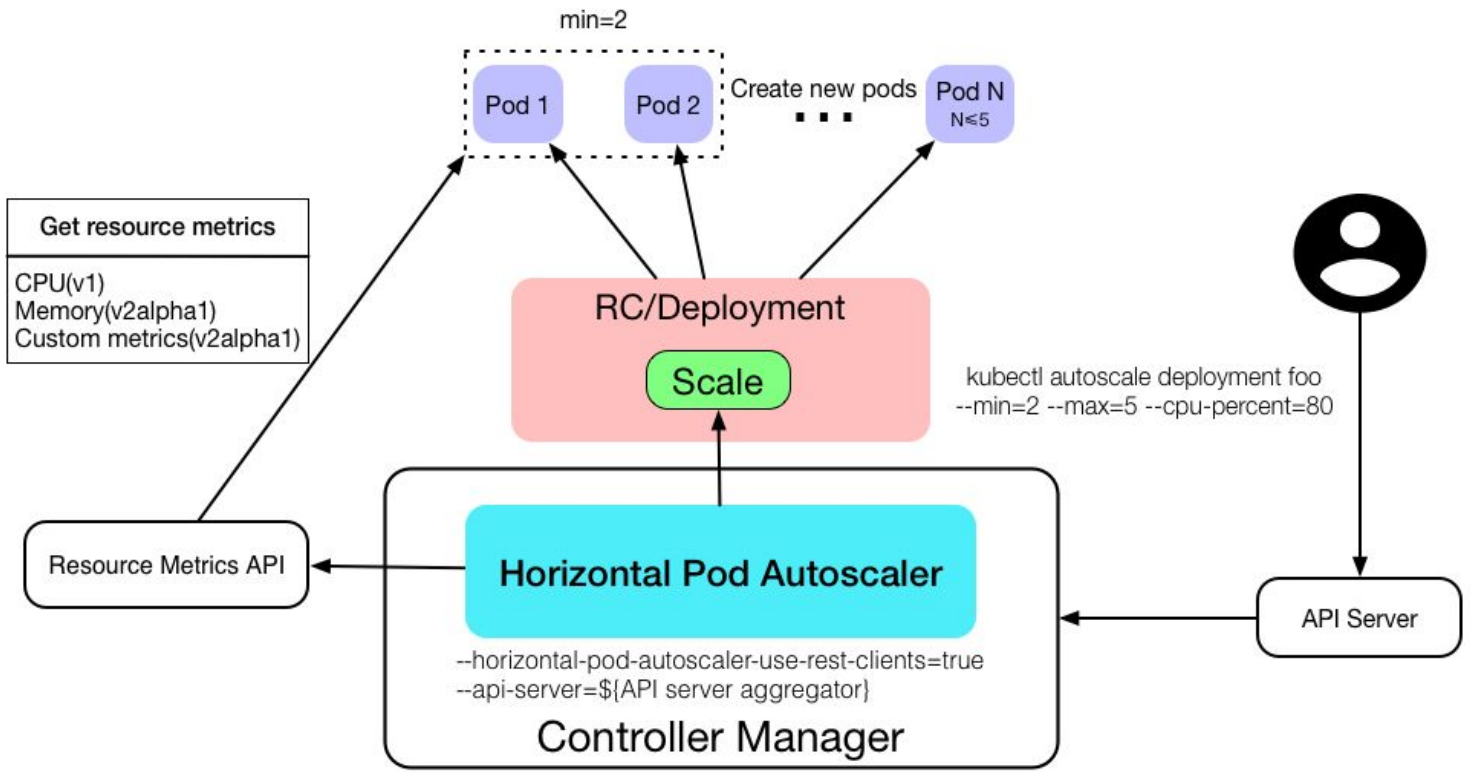
- 한 노드의 모든 POD 의 한계(limit)의 합이 노드 용량의 100%를 초과 할 수 있다



# 20. HPA (Horizontal Pod Auto-Scaler)

# HPA 작동 원리 – Metric 수집

- Horizontal 컨트롤러가 HorizontalPodAutoscaler(HPA) 리소스를 만들어 수평 스케일링을 수행 함
- 스케일링 대상은 리플리케이션 컨트롤러 / 디플로이먼트 / 리플리카셋 / 스테이트풀셋
- Kubelet 의 cAdvisor 를 통해 메트릭이 수집됨
- HPA 컨트롤러는 –horizontal-pod-autoscaler-use-rest-clients=true / --api-server=$(api server aggregator) 를 통해 메트릭 수집




Source https://github.com/rootsongjc/kubernetes-handbook

# HPA 작동 원리 – 계산 방법

- HPA 가 Metric 을 수집해서 계산한 결과는 복제본의 수
- 단일 메트릭 항목만 고려 하도록 구성된 경우, 모든 Pod의 메트릭 값을 합산후 HPA 에 설정된 목표 값으로 나눈 후 큰 정수로 반올림
- 여러 메트릭을 고려 하도록 구성된 경우 계산이 훨씬 복잡 해짐.