Gestión de recursos en Kubernetes

KEEPCODING
Tech School

KeepCoding®. Todos los derechos reservados

Requests y Limits en pods

KEEPCODING
Tech School

KeepCoding®. Todos los derechos reservados

Recursos en Kubernetes

- Cuándo creamos un Pod, de manera (casi) opcional podemos especificar cuánta CPU y cuánta memoria RAM necesita cada contenedor.
- Diferenciamos 2 tipos:
  - **Requests**: Lo mínimo que el contenedor demanda al sistema para poder funcionar → Se reservan recursos (reserved resources).
  - **Limits**: El límite máximo de los recursos que puede consumir.
- Cuando establecemos "**requests**", el scheduler de Kubernetes puede tomar mejores decisiones a la hora de saber en qué nodo debe colocar nuestro POD.
- Al especificar "**limits**" evitaremos que algún contenedor devaste los recursos del Clúster,

```
spec.containers[].resources.limits.cpu
spec.containers[].resources.limits.memory
spec.containers[].resources.requests.cpu
spec.containers[].resources.requests.memory
```

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Límites de CPU

- Cada CPU de un nodo de Kubernetes es equivalente a:
  - 1 AWS vCPU
  - 1 GCP Core
  - 1 Azure vCore
  - 1 Hyperthread on a bare-metal Intel processor with Hyperthreading
- Es una unidad absoluta, trabajaremos habitualmente con milicores:
  - 0.1 es equivalente a 100m.
  - 1000m = 1 CPU
- No podremos reservar más CPU que la que haya físicamente disponible en el nodo de mayor tamaño.
- Valores **por defecto** (configurable)**:**
  - 100m request y sin límite

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Límites de Memoria

- Se mide en bytes.
- Podemos usar los sufijos (E, P, T, G, M, K) y en potencia de 2 (Ei, Pi, Ti, Gi, Mi, Ki).
- No podremos asignar más requested memory que la que haya físicamente disponible en el nodo más grande
- Valores **por defecto** (configurable)**:**
  - 0Mb request y sin límite

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Diferentes casuísticas

- Si no se especifican límites:
  - CPU
  - Memoria

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

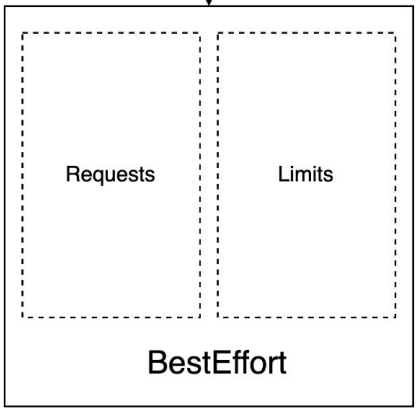
¿Cómo maneja Kubernetes los diferentes PODs?

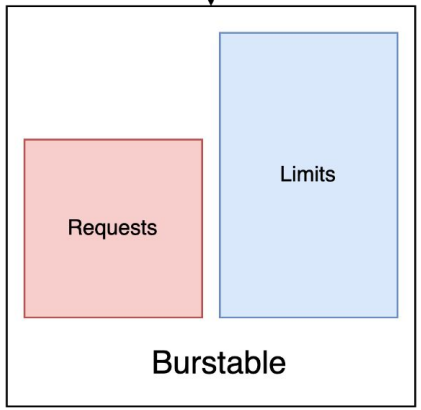
- Tipos de QoS aplicada por Kubernetes
  - Guaranteed → Todo tiene request y limit y son iguales
  - Burstable → Hay algún request.
  - BestEffort → Nada tiene request ni limit.
- Si un nodo está bajo mucha presión de memoria lo primero en matar son pods BestEffort y posteriormente los Burstable, nunca los Guaranteed (excepto si superan su propio límite de memoria por ejemplo).
- Si un Pod excede su límite de memoria se reinicia
- Si el Pod intenta usar CPU por encima del límite se frena → CPU throttling

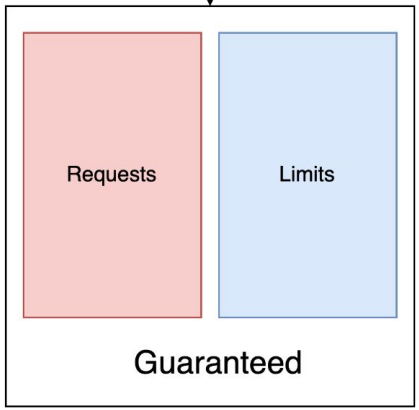
https://kubernetes.io/docs/tasks/configure-pod-container/quality-service-pod/

© All rights reserved. www.keepcoding.io

Quality of Service







© All rights reserved. www.keepcoding.io

Control de recursos - Demo

- Demo!

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Ejemplo

- Este POD tiene 2 contenedores.
- El contenedor “db” requiere un mínimo de 64MiB de memoria y un mínimo de 250milicores de CPU.
- El contenedor “db” como máximo podrá usar 128MiB de memoria y 500milicores de CPU.
- El contenedor “wordpress” requiere un mínimo de 128MiB de memoria y un mínimo de 500milicores de CPU.
- El contenedor “wordpress” como máximo podrá usar 128MiB de memoria y 500milicores de CPU.

```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:
      limits:
        memory: "128Mi"
        cpu: "500m"
```

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Consideraciones

- ¿Qué pasa si pongo un límite más grande de lo que dispongo en el Clúster? ¿Y un request?
- ¿Es importante establecer limits y requests?
- ¿Cómo actúa Kubernetes cuándo sobrepasamos los límites?
- ¿Qué es un OOMKill?
- ¿Y un OOM Killer del sistema?
- ¿Cómo podemos otorgar QoS?

El control de estos recursos es **FUNDAMENTAL**

© All rights reserved. www.keepcoding.io

Límites y Cuotas

KEEPCODING
Tech School

KeepCoding®. Todos los derechos reservados

Límites y requests por defecto para namespaces

- Ya hemos visto la importancia que tiene establecer límites y requests en nuestros PODs.
- Podemos establecer límites por defecto **a nivel de namespace** (para cada POD que no los defina).
- Para ello usaremos el recurso "**LimitRange**"

```
apiVersion: v1
kind: LimitRange
metadata:
  name: mem-limit-range
spec:
  limits:
  - default:
      memory: 512Mi
    defaultRequest:
      memory: 256Mi
      type: Container
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-limit-range
spec:
  limits:
  - default:
      cpu: 1
    defaultRequest:
      cpu: 0.1
      type: Container
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: mem-min-max-lr
spec:
  limits:
  - max:
      memory: 1Gi
    min:
      memory: 500Mi
    type: Container
```

```
apiVersion: v1
kind: LimitRange
metadata:
  name: cpu-min-max-lr
spec:
  limits:
  - max:
      cpu: 1
    min:
      cpu: 0.1
    type: Container
```

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Quotas de CPU y Memoria por Namespace

- Podemos limitar la cantidad total de CPU y Memoria que se pueden asignar en un namespace.
- Será la suma de lo que soliciten cada uno de los PODs del namespace.
- Usaremos el recurso “**ResourceQuota**”

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-demo
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
     limits.memory: 2Gi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: quota-mem-cpu-demo
spec:
  containers:
  - name: quota-mem-cpu-demo-ctr
    image: nginx
    resources:
      limits:
        memory: "800Mi"
        cpu: "800m"
      requests:
        memory: "600Mi"
          cpu: "400m"
```

```
Error from server (Forbidden): error when creating "quota-mem-cpu-pod-2.yaml": pods "quota-mem-cpu-demo-2" is
forbidden: exceeded quota: mem-cpu-demo, requested: requests.memory=700Mi,used: requests.memory=600Mi,
limited: requests.memory=1Gi
```

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/tasks/administer-cluster/quota-api-object/

Quotas de objetos del API de Kubernetes

- Podemos limitar la cantidad de objetos de un tipo que se pueden crear **por Namespace**.
- Hay algunos objetos que pueden incurrir en costes y queremos poder limitarlos.

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: object-quota-demo
spec:
  hard:
    pods: "2"
    services.loadbalancers: "1"
    services.nodeports: "0"
```

String	API Object
"pods"	Pod
"services	Service
"replicationcontrollers"	ReplicationController
"resourcequotas"	ResourceQuota
"secrets"	Secret
"configmaps"	ConfigMap
"persistentvolumeclaims"	PersistentVolumeClaim
"services.nodeports"	Service of type NodePort
"services.loadbalancers"	Service of type LoadBalancer

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/tasks/administer-cluster/quota-api-object/

Consultar consumo de recursos

- Podemos ver lo que se está consumiendo en los nodos
- Podemos ver lo que se está consumiendo en un Namespace.
- Estas métricas las proporciona el metricsServer de Kubernetes a través de cAdvisor.
- No son valores instantáneos.

```
$ kubectl top node
$ kubectl top pod
```

```
$ minikube addons enable metrics-server
```

x

© All rights reserved. www.keepcoding.io

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Ejercicios cpu y memoria

KEEPCODING
Tech School

KeepCoding®. Todos los derechos reservados

Ejercicios control de recursos

1. Crea un POD nginx que esté limitado a 10m de CPU y 64MiB de RAM
2. Crea un POD de nginx que esté limitado a 1MiB de RAM
3. Crea un POD al que se asigne una clase QoS de 'Guaranteed'. **Verifícalo**
4. Crea un POD que requiera un número muy elevado de CPUs (20). **Observa el comportamiento.**

1. Crea un namespace “keepcoding” y establece un valor por defecto de 10m de CPU y 64MiB de RAM
2. En el namespace de Keepcoding, pon un límite máximo de 128MiB de memoria.
3. Establece una cuota máxima de memoria de 256MiB en el namespace de Keepcoding.
4. Establece una cuota de máximo 2 Pods en el namespace de Keepcoding.

© All rights reserved. www.keepcoding.io

KEEPCODING

Tech School

Madrid | Barcelona | Bogotá

KeepCoding®. Todos los derechos reservados