Resource Limits

What are resource limits?

One of the challenges of every distributed system designed to share resources between applications, like Kubernetes, is, paradoxically, how to properly share the resources. When Kubernetes schedules a Pod, it's important that the containers have enough resources to actually run. If you schedule a large application on a node with limited resources, it is possible for the node to run out of memory or CPU resources and for things to stop working!

Requests and limits are the mechanisms Kubernetes uses to control resources such as CPU and memory. Requests are what the container is guaranteed to get. If a container requests a resource, Kubernetes will only schedule it on a node that can give it that resource. Limits, on the other hand, make sure a container never goes above a certain value. The container is only allowed to go up to the limit, and then it is restricted.

CPU

CPU resources are defined in millicores. If your container needs two full cores to run, you would put the value "2000m". If your container only needs ¼ of a core, you would put a value of "250m".

Memory

Memory resources are defined in bytes. Normally, you give a mebibyte value for memory (this is basically the same thing as a megabyte), but you can give anything.

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: demo
spec:
  hard:
    requests.cpu: 500m
    requests.memory: 100Mib
    limits.cpu: 700m
    limits.memory: 500Mib
```

requests.cpu is the maximum combined CPU requests in millicores for all the containers in the Namespace. In the above example, you can have 50 containers with 10m requests, five containers with 100m requests, or even one container with a 500m

request. As long as the total requested CPU in the Namespace is less than 500m!

requests.memory is the maximum combined Memory requests for all the containers in the Namespace. In the above example, you can have 50 containers with 2MiB requests, five containers with 20MiB CPU requests, or even a single container with a 100MiB request. As long as the total requested Memory in the Namespace is less than 100MiB!

limits.cpu is the maximum combined CPU limits for all the containers in the Namespace. It's just like requests.cpu but for the limit.

limits.memory is the maximum combined Memory limits for all containers in the Namespace. It's just like requests.memory but for the limit.

Hands On – Testing a Resource Limit

I. Create a new namespace with the imperative command:

Let's first create a new namespace and then verify it exists using the kubens command. Notice the default namespace will be highlighted yellow as that's the current namespace you're in.

```
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$ kubectl create ns rl-dom-test-100mi
namespace/rl-dom-test-100mi created
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$ kubens
default
kube-node-lease
kube-public
kube-system
rl-dom-test-100mi
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$
```

II. Create a memory limit of 100mi for the namespace:

Now, we can edit the namespace to have a resource memory limit of 100 mi.

```
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$ cat doms-namespace.yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: rl-dom-test
  namespace: rl-dom-test-100mi
spec:
  limits:
  - max:
      memory: 100Mi
    type: Container
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$ kubectl apply -f doms-namespace.yaml
limitrange/rl-dom-test created
```

III. Create a pod with a memory request of 100mi, verify the error in the logs:

Last, lets create a pod that is asking for over 100mi of memory. If our resource limit is working in the namespace, we should receive an error.

```
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$ cat doms-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: rl-test
  namespace: rl-dom-test-100mi
spec:
  containers:
  - name: rl-test
    image: nginx
    resources:
      requests:
        memory: 150Mi
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$ kubectl apply -f doms-pod.yaml
The Pod "rl-test" is invalid: spec.containers[0].resources.requests: Invalid value: "150Mi": mus
t be less than or equal to memory limit
dominickhrndz314@cloudshell:~ (sandbox-io-289003)$
```

If you received this error, then you have completed this training.