

The Mismatch Scenario

In this lesson, we will figure out what happens when a Namespace's defined limits are violated.

WE'LL COVER THE FOLLOWING ^

- Looking into the Definition
- Creating Resources
- Creating the Pods Directly
- Destroying the Namespace

Let's see what happens when resources are defined, but they do not match the Namespace `min` and `max` limits.

Looking into the Definition

We'll use the same `go-demo-2.yml` we used before.

```
cat res/go-demo-2.yml
```



The **output**, limited to the relevant parts, is as follows.

```
...
apiVersion: apps/v1
kind: Deployment
metadata:
  name: go-demo-2-db
spec:
  ...
  template:
    ...
    spec:
      containers:
      - name: db
        image: mongo:3.3
        resources:
          limits:
            memory: 100Mi
            cpu: 0.1
          requests:
```



```

        requests:
          memory: 50Mi
          cpu: 0.01
    ...
apiVersion: apps/v1
kind: Deployment
metadata:
  name: go-demo-2-api
spec:
  ...
  template:
    ...
    spec:
      containers:
      - name: api
        ...
        resources:
          limits:
            memory: 10Mi
            cpu: 0.1
          requests:
            memory: 5Mi
            cpu: 0.01
    ...

```

What matters is that the `resources` for both Deployments are defined.

Creating Resources

Let's create the objects and retrieve the events. They will help us understand better what is happening.

```

kubectl --namespace test apply \
  -f res/go-demo-2.yml \
  --record

kubectl --namespace test \
  get events \
  --watch

```

The **output** of the latter command, limited to the relevant parts, is as follows.

```

... Error creating: pods "go-demo-2-db-868dbbc488-s92nm" is forbidden: maximum memory usage p
...
... Error creating: pods "go-demo-2-api-6bd767ffb6-96mbl" is forbidden: minimum memory usage
...

```

We can see that we are forbidden from creating either of the two Pods. The difference between those events is in what caused Kubernetes to reject our request.

The `go-demo-2-db-*` Pod could not be created because its `maximum memory usage`

The `go-demo-2-api-*` Pod could not be created because its `maximum memory usage` per Container is `80Mi`, but limit is `100Mi`. On the other hand, we are forbidden from creating the `go-demo-2-api-*` Pods because the `minimum memory usage` per Container is `10Mi`, but request is `5Mi`.

All the containers within the `test` Namespace will have to comply with the `min` and `max` limits. Otherwise, we are forbidden from creating them.

Container limits cannot be higher than the Namespace `max` limits. On the other hand, container resource requests cannot be smaller than Namespace `min` limits.

If we think about Namespace limits as lower and upper thresholds, we can say that container requests cannot be below them, and that container limits can't be above.

! Please press the `CTRL` and `c` keys to stop watching the events.

Creating the Pods Directly

It might be easier to observe the effects of the `max` and `min` limits if we create Pods directly, instead of through Deployments.

```
kubectl --namespace test run test \
  --image alpine \
  --requests memory=100Mi \
  --restart Never \
  sleep 10000
```



We tried to create a Pod with the memory request set to `100Mi`. Since the Namespace limit is `80Mi`, the API returned the error message stating that the `Pod "test" is invalid`. Even though the `max` limit refers to container `limit`, memory `request` was used in its absence.

We'll run a similar exercise but, this time, with only `1Mi` set as memory request.

```
kubect1 --namespace test run test \  
  --image alpine \  
  --requests memory=1Mi \  
  --restart Never \  
  sleep 10000
```



This time, the error is slightly different. We can see that `pods "test" is invalid: spec.containers[0].resources.requests: Invalid value: "100Mi": must be less than or equal to memory limit`. What we requested is below the `min` limit of the `test` Namespace and, therefore, we are forbidden from creating the Pod.

Destroying the Namespace

We'll delete the `test` Namespace before we move into the next subject.

```
kubect1 delete namespace test
```



In the next lesson, we will explore how to define resource quotas for a Namespace.