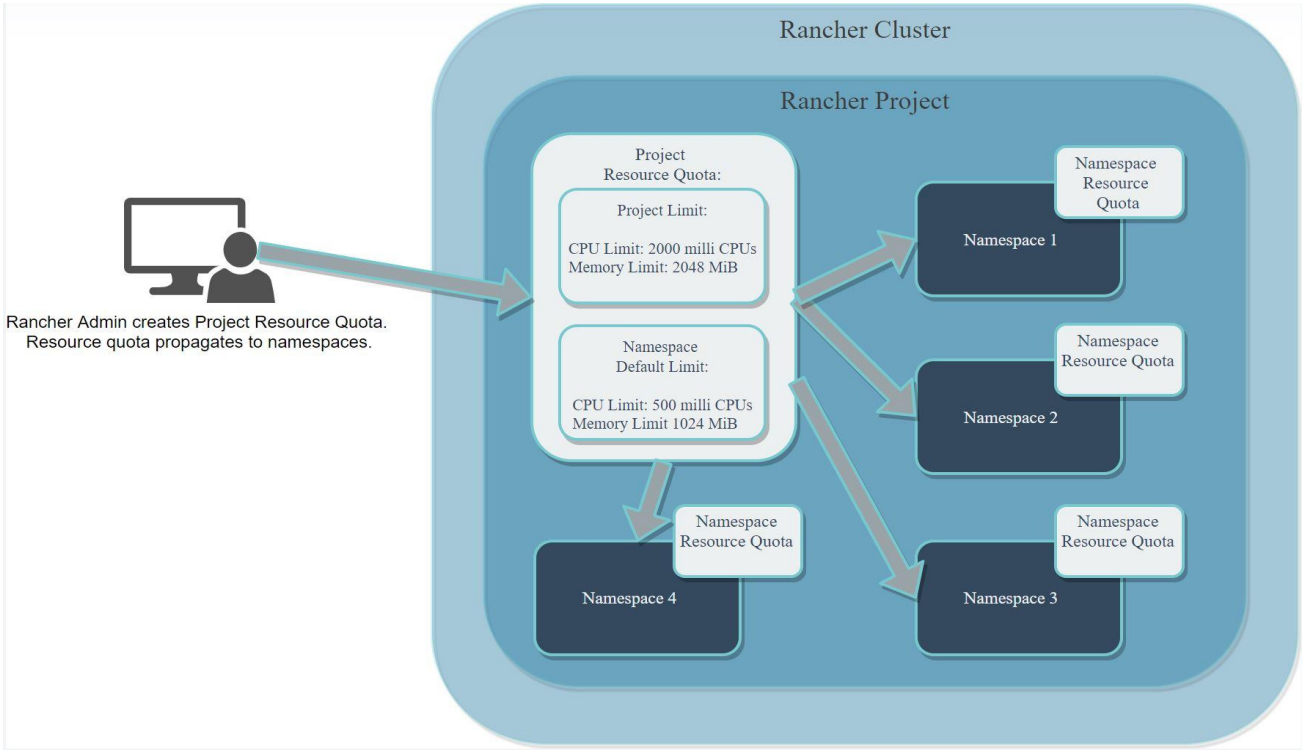
Resource Quota in Rancher & our Approach in IOTCLOUD

Topics

- How Resource Quota work on Rancher ?
- Config Quota in Rancher
- Parameters
- Quota Management approach
- How to assign Quota : Name Space level , Container level
- PODS Quota
- PVC quota
- CPU & MEM QUOTAs
  - Request and limit concept
  - Example

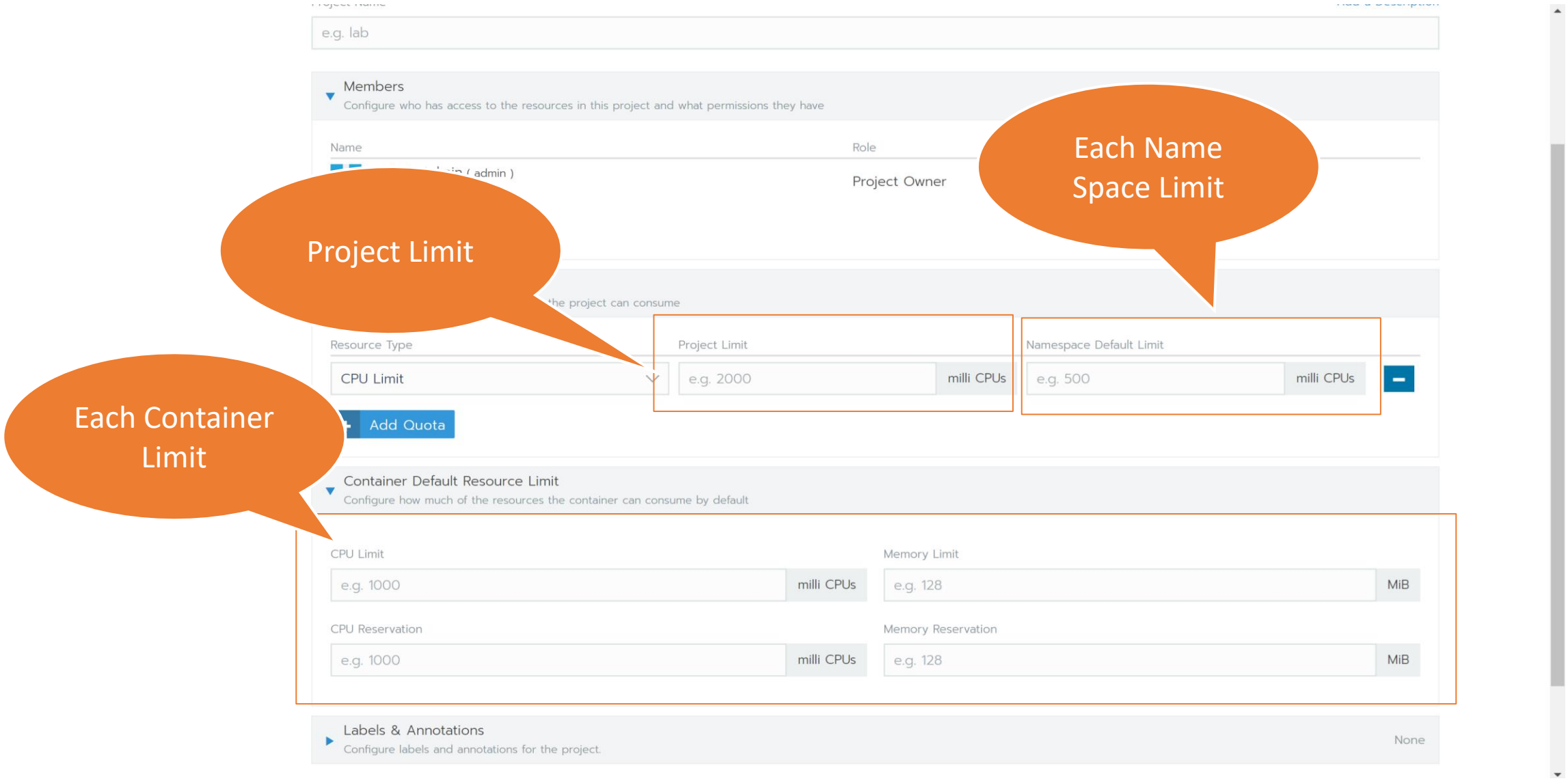
How Resource Quota work on Rancher ?

- In Rancher we can Assign Quota to ***Project*** , Namespace and Container
- In Rancher, you apply a resource quota to the project, and then the quota propagates to each namespace

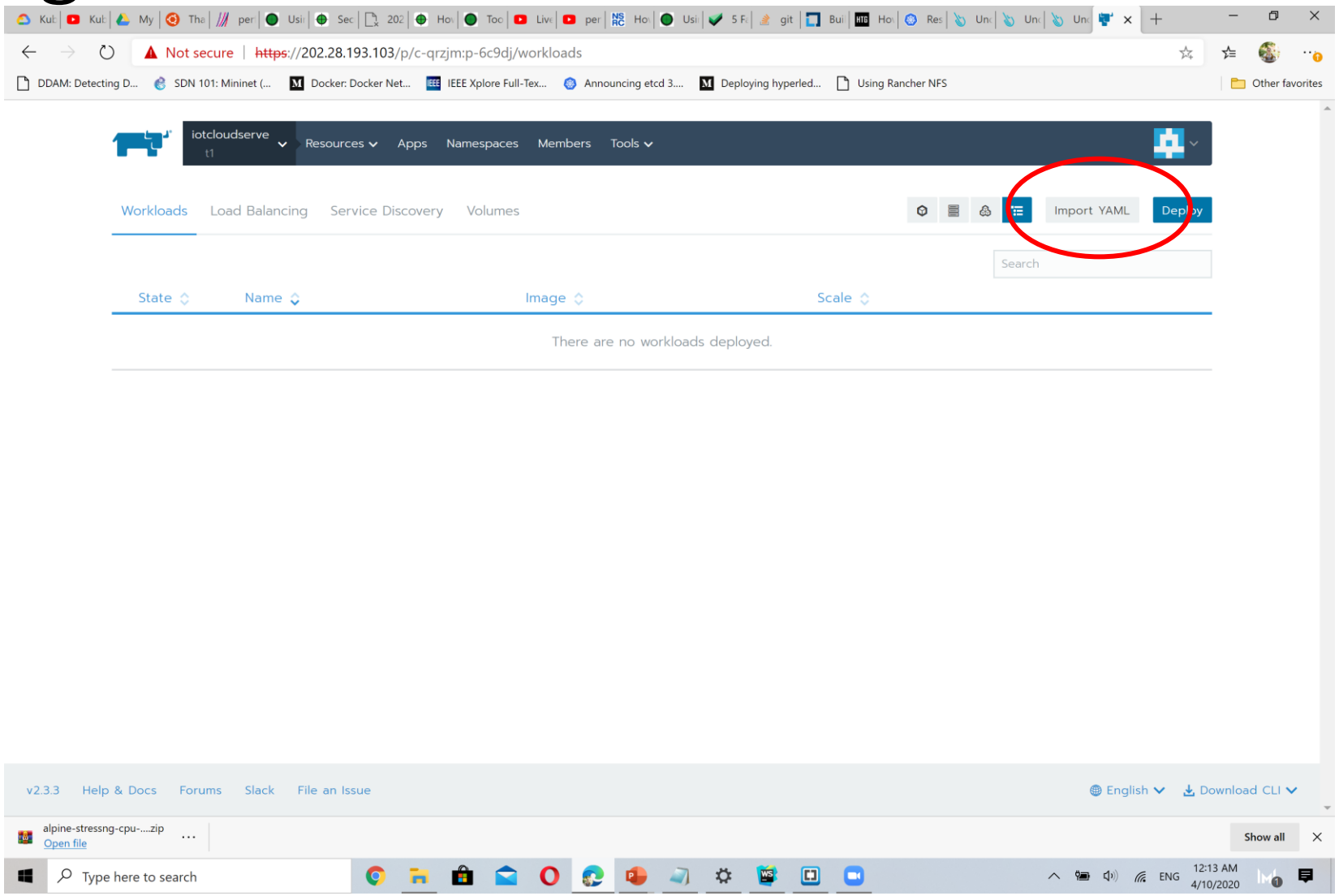


Ref. :https://rancher.com/docs/rancher/v2.x/en/project-admin/resource-quotas/quotas-for-projects/

Config Quota in Rancher (1)



Config Quota in Rancher



Parameters that Can be limited

RESOURCE TYPE	DESCRIPTION
CPU Limit*	The maximum amount of CPU (in millicores) allocated to the project/namespace.[1]
CPU Reservation*	The minimum amount of CPU (in millicores) guaranteed to the project/namespace.[1]
Memory Limit*	The maximum amount of memory (in bytes) allocated to the project/namespace.[1]
Memory Reservation*	The minimum amount of memory (in bytes) guaranteed to the project/namespace.[1]
Storage Reservation	The minimum amount of storage (in gigabytes) guaranteed to the project/namespace.
Services Load Balancers	s services that can exist in the
Services Node Ports	vices that can exist in the project/namespace.
Pods	The maximum number of pods that can exist in the project/namespace in a non-terminal state (i.e., pods with a state of .status.phase in (Failed, Succeeded) equal to true).
Services	The maximum number of services that can exist in the project/namespace.
ConfigMaps	The maximum number of ConfigMaps that can exist in the project/namespace.
Persistent Volume Claims	The maximum number of persistent volume claims that can exist in the project/namespace.
Replications Controllers	The maximum number of replication controllers that can exist in the project/namespace.
Secrets	The maximum number of secrets that can exist in the project/namespace.

Parameter That We should limit the clients

Ref. :https://rancher.com/docs/rancher/v2.x/en/project-admin/resource-quotas/quota-type-reference/

Our Approach Structure

Admins ***manually*** assign Resource Quota to Project/Namespace

IOT Cloud Cluster

Project : Smart Mobility 01
Quota : CPU,RAM,POD,Storage

Name Space : SDWMN

Project : Smart Mobility 02
Quota : CPU,RAM,POD,Storage

Name Space :Traffic Analysis

Project : Smart Agriculture 01
Quota : CPU,RAM,POD,Storage

Name Space : Smart Agriculture Laos's Team

Project : Smart Energy 01
Quota : CPU,RAM,POD,Storage

Name Space : Cheer's Gateway

Project : Smart Energy 02
Quota : CPU,RAM,POD,Storage

Name Space : CUsmartcampus

Project : Smart Energy 03
Quota : CPU,RAM,POD,Storage

Name Space : CUloadforecasting

Project : Smart Energy 04
Quota : CPU,RAM,POD,Storage

Name Space : ISE2020

= K8S Cluster

= Project

= NameSpace

Note : CPU,RAM not yet Implemented

Our Approach

- PODs
- Persistent Volume Claim (PVC)
- CPU limit
- Memory limit

Can be config in Rancher

have to use .yml file

How to assign Quota : Name Space level , Container level

Kind

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

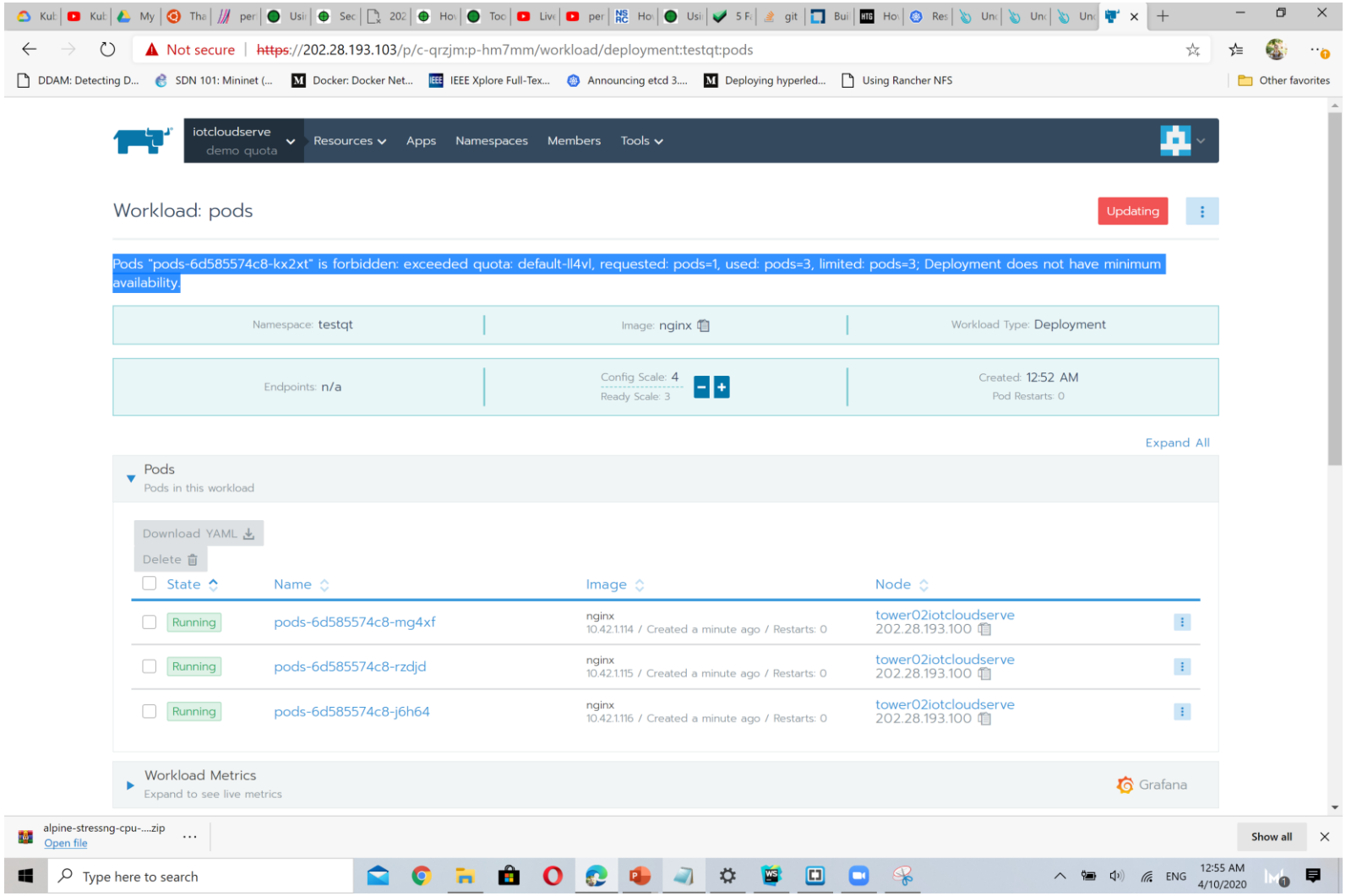
<namespace>.yml

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: demo
spec:
  limits:
- default:
    cpu: 600m
    memory: 100Mib
  defaultRequest:
    cpu: 100m
    memory: 50Mib
  max:
    cpu: 1000m
    memory: 200Mib
  min:
    cpu: 10m
    memory: 10Mib
  type: Container
```

<container>.yml

https://cloud.google.com/blog/products/gcp/kubernetes-best-practices-resource-requests-and-limits

Example 1 : PODs



Persistent Volume Claim (PVC) quota config

- The admin can limit 3 scenarios
  1. The number of persistent volume claims in a namespace *on Rancher
  2. The amount of storage each claim can request
  3. The amount of cumulative storage the namespace can have

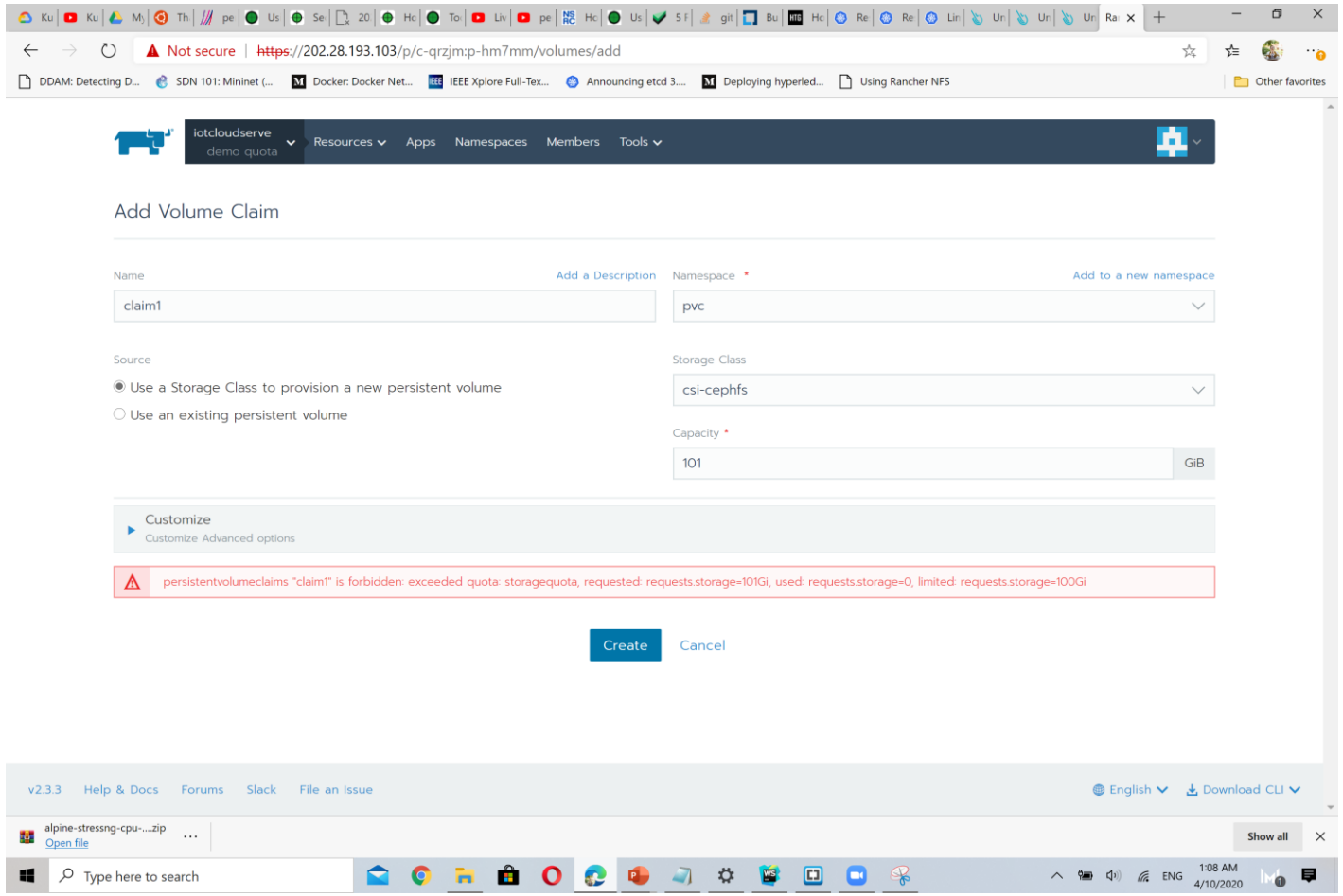
Ref https://kubernetes.io/docs/tasks/administer-cluster/limit-storage-consumption/

PVC scenario 1

Adding a LimitRange to a namespace enforces storage request sizes to a minimum and maximum.
Storage is requested via PersistentVolumeClaim.Limit ranges will reject any PVC that is above or below the values set by the admin.

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: storagelimits
spec:
  limits:
  - type: PersistentVolumeClaim
    max:
      storage: 10Gi
#   min:
#     storage: 0.01Gi
```

Example 2 : PVC (1)

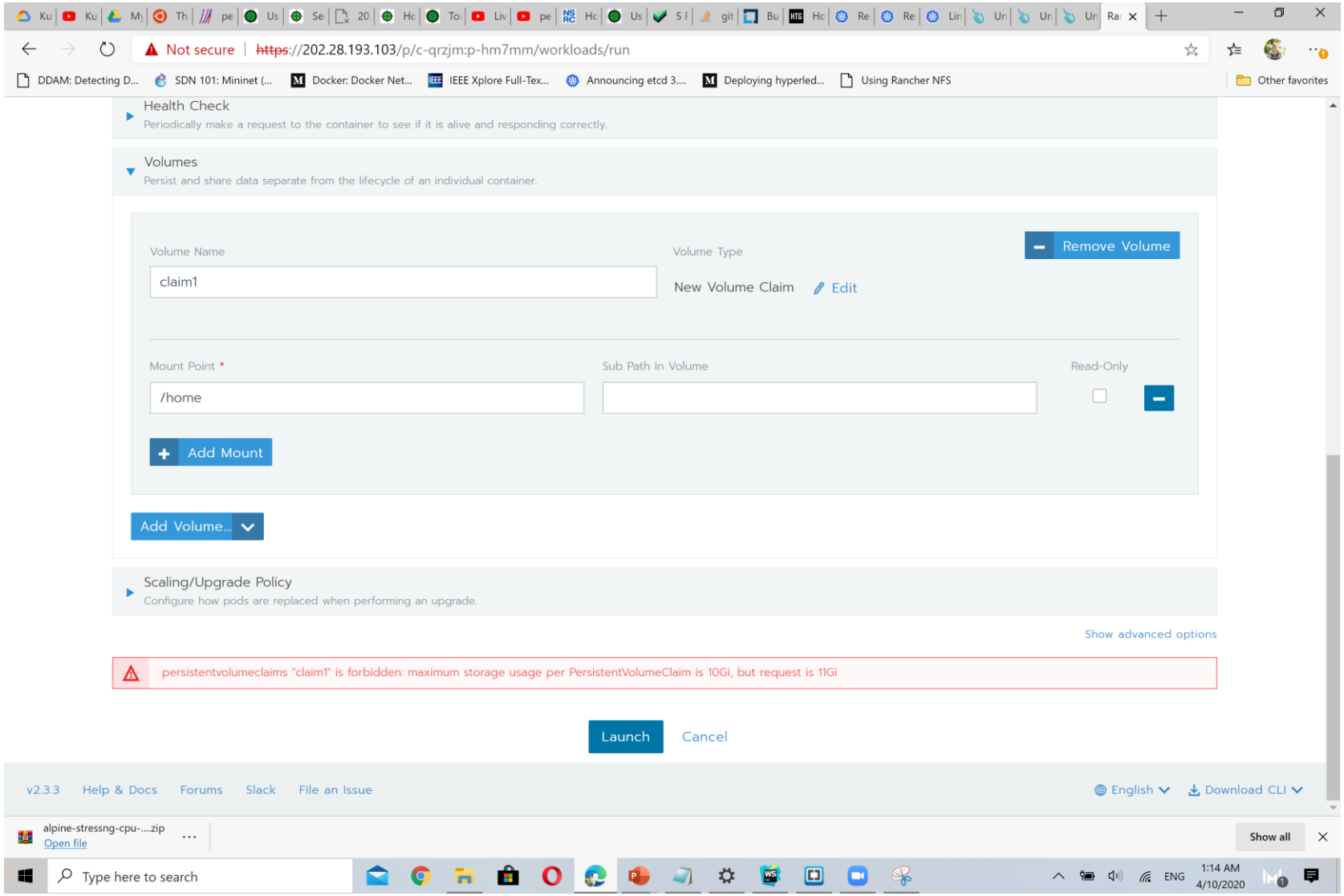


PVC scenario 2

- New PVCs that exceed either maximum cumulative value will be rejected.

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storagequota
spec:
  hard:
    persistentvolumeclaims: "100"
    requests.storage: "100Gi"
```

Example 2 : PVC (2)



Request and limit

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-example
spec:
  hard:
    requests.cpu: 2
    requests.memory: 2Gi
    limits.cpu: 3
    limits.memory: 4Gi
```

What is the difference?

Request and limit

- Requests and limits are the mechanisms Kubernetes uses to control resources such as CPU and memory.

Requests VS Limit ?

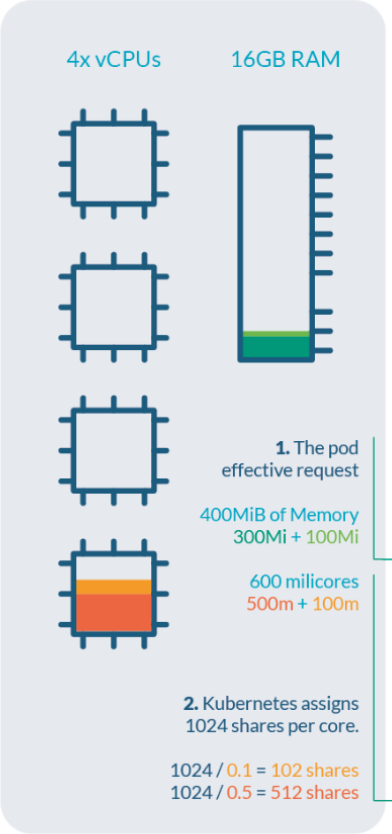
- Requests are what the container is guaranteed to get. Kubernetes will only schedule it on a node that can give it that resource.
- Limits, on the other hand, make sure a container never goes above a certain value. The container is only allowed to go up to the limit, and then it is restricted.

Example : CPU Quota

The pod - Deployment.yaml

```yaml
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
 name: redis
 labels:
   name: redis-deployment
   app: example-voting-app
spec:
 replicas: 1
 selector:
   matchLabels:
    name: redis
    role: redisdb
    app: example-voting-app
 template:
   spec:
     containers:
       - name: redis
         image: redis:5.0.3-alpine
         resources:
           limits:
             memory: 600Mi
             cpu: 1
           requests:
             memory: 300Mi
             cpu: 500m
       - name: busybox
         image: busybox:1.28
         resources:
           limits:
             memory: 200Mi
             cpu: 300m
           requests:
             memory: 100Mi
             cpu: 100m
```

A cluster node:



3. Will be killed if allocates > 600MB.
The whole Pod will fail.

4. Will be throttled if uses more than "1 Core".
1 core = 1000 milicores = 1000m =
100ms of computing time every 100 real ms

Full computing time of the node:
4 vCPUs * 100 real ms =
400ms of computing time = 4000m

5. Killed if allocates > 200MB.

6. Throttled if uses > 30ms of computing time in 100ms

https://sysdig.com/blog/kubernetes-limits-requests/

References

- https://cloud.google.com/blog/products/gcp/kubernetes-best-practices-resource-requests-and-limits
- https://sysdig.com/blog/kubernetes-limits-requests/
- https://kubernetes.io/docs/concepts/policy/resource-quotas/#quota-scopes