

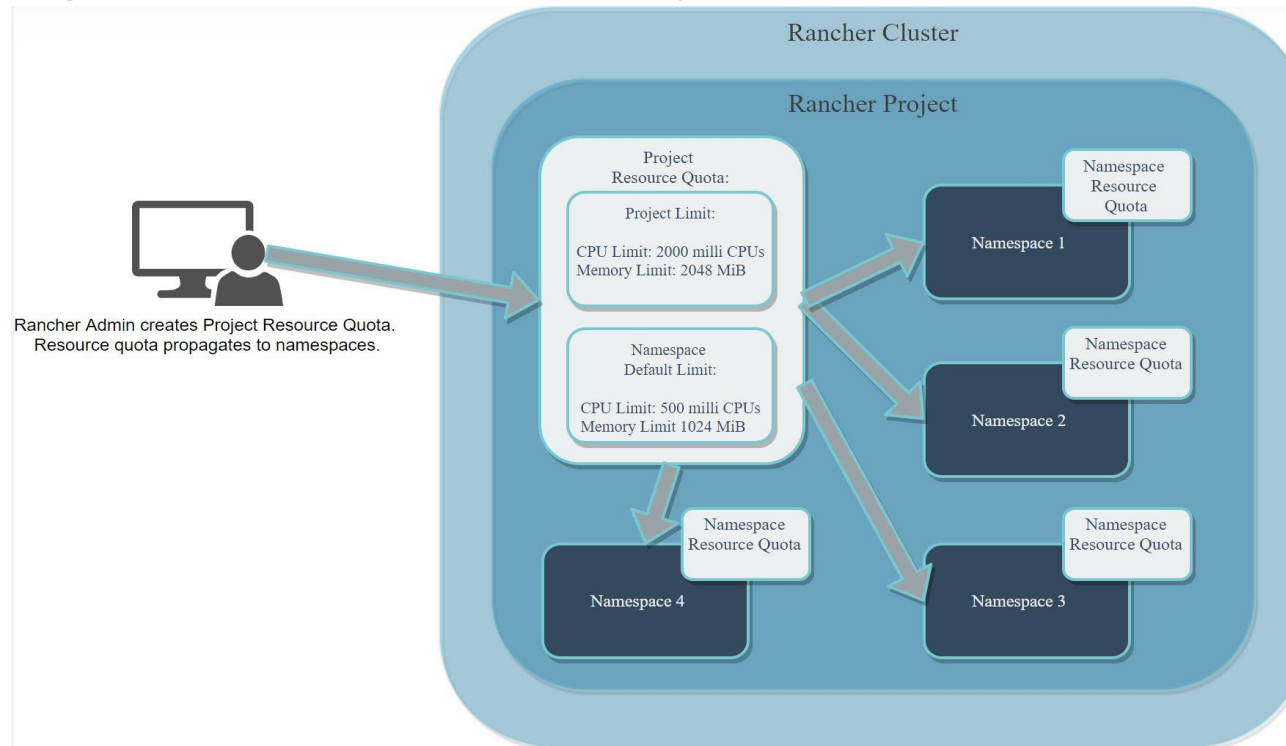
# Resource Quota in Rancher & our Approach in IOTCLOUD

# 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



# Config Quota in Rancher (1)

Project Name  
e.g. lab

**Members**  
Configure who has access to the resources in this project and what permissions they have

Name	Role
admin (admin)	Project Owner

the project can consume

Resource Type

CPU Limit

Project Limit

e.g. 2000

milli CPUs

Namespace Default Limit

e.g. 500

milli CPUs

+ Add Quota

**Container Default Resource Limit**  
Configure how much of the resources the container can consume by default

CPU Limit	e.g. 1000	milli CPUs	Memory Limit	e.g. 128	MiB
CPU Reservation	e.g. 1000	milli CPUs	Memory Reservation	e.g. 128	MiB

**Labels & Annotations**  
Configure labels and annotations for the project.

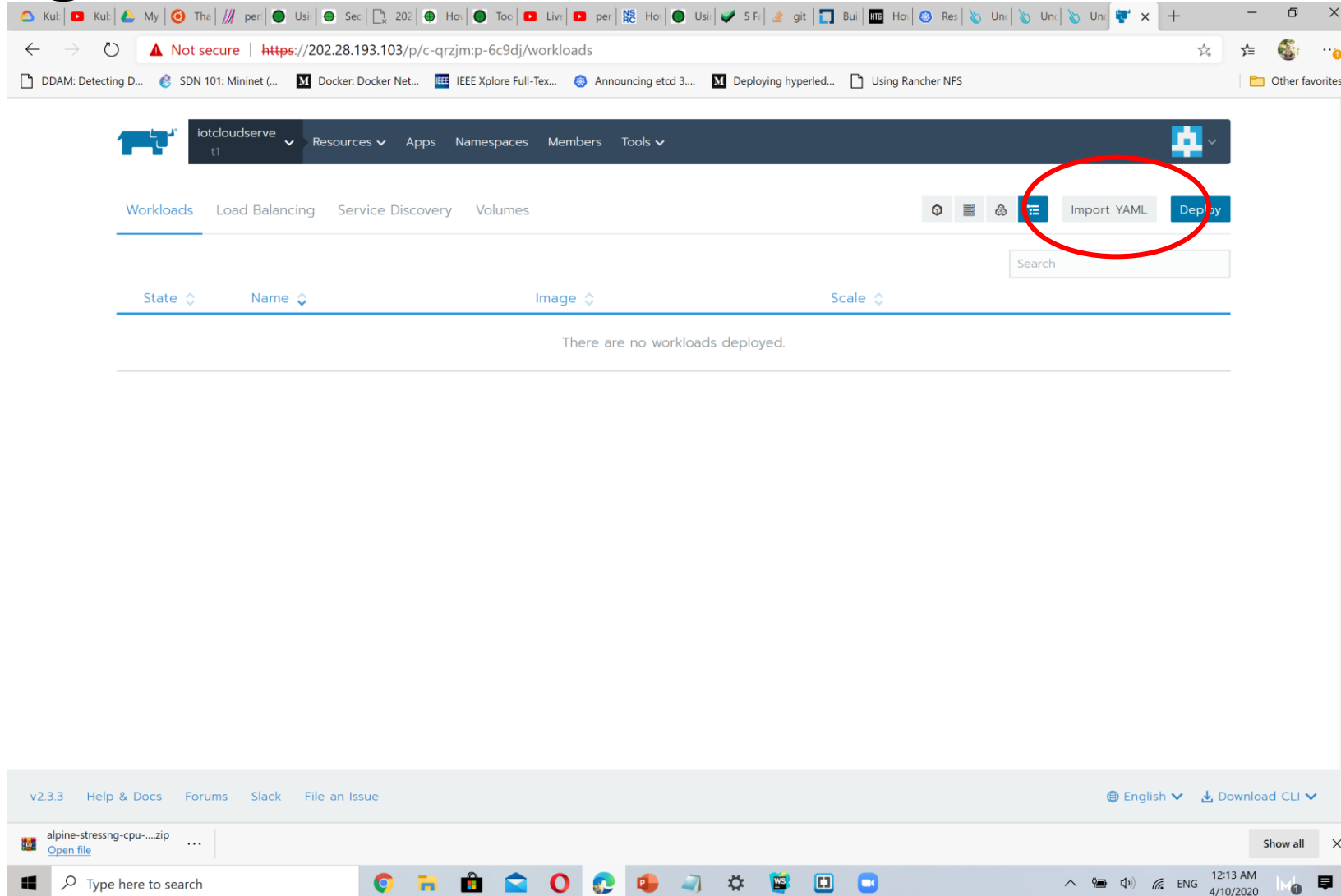
None

Project Limit

Each Name  
Space Limit

Each Container  
Limit

# Config Quota in Rancher



# Parameters that Can be limited

RESOURCE TYPE	DESCRIPTION
CPU Limit*	The maximum amount of CPU (in <a href="#">millicores</a> ) allocated to the project/namespace. <sup>1</sup>
CPU Reservation*	The minimum amount of CPU (in millicores) guaranteed to the project/namespace. <sup>1</sup>
Memory Limit*	The maximum amount of memory (in bytes) allocated to the project/namespace. <sup>1</sup>
Memory Reservation*	The minimum amount of memory (in bytes) guaranteed to the project/namespace. <sup>1</sup>
Storage Reservation	The minimum amount of storage (in gigabytes) guaranteed to the project/namespace.
Services Load Balancers	The maximum number of services that can exist in the project/namespace.
Services Node Ports	The maximum number of services 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

# Our Approach Structure



Admins manually assign Resource  
Quota to Project/Namespaces

## 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 .yaml file



# How to assign Quota : Name Space level , Container level

Kind

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

<namespace>.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>.yaml

# Example 1 : PODs

Workload: pods Updating

Pods "pods-6d585574c8-kx2xt" is forbidden: exceeded quota: default-ll4vl, requested: pods=1, used: pods=3, limited: pods=3; Deployment does not have minimum availability.

Namespace: testqt	Image: nginx	Workload Type: Deployment
Endpoints: n/a	Config Scale: 4 Ready Scale: 3	Created: 12:52 AM Pod Restarts: 0

Expand All

▼ Pods  
Pods in this workload

Download YAML

Delete

State	Name	Image	Node
<input type="checkbox"/> Running	pods-6d585574c8-mg4xf	nginx 10.42.1.114 / Created a minute ago / Restarts: 0	tower02iotcloudserve 202.28.193.100
<input type="checkbox"/> Running	pods-6d585574c8-rzdjd	nginx 10.42.1.115 / Created a minute ago / Restarts: 0	tower02iotcloudserve 202.28.193.100
<input type="checkbox"/> Running	pods-6d585574c8-j6h64	nginx 10.42.1.116 / Created a minute ago / Restarts: 0	tower02iotcloudserve 202.28.193.100

Workload Metrics  
Expand to see live metrics

alpine-stressng-cpu-....zip [Open file](#)

Type here to search

12:55 AM  
4/10/2020

# 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

# 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.

```
1  apiVersion: v1
2  kind: LimitRange
3  metadata:
4    name: storagelimits
5  spec:
6    limits:
7      - type: PersistentVolumeClaim
8        max:
9          storage: 10Gi
10 #   min:
11 #     storage: 0.01Gi
```

# Example 2 : PVC (1)

The screenshot shows the 'Add Volume Claim' form in the Kubernetes Dashboard. The form fields are as follows:

- Name:** claim1
- Namespace:** pvc
- Source:** ☒ Use a Storage Class to provision a new persistent volume
- Storage Class:** csi-cephfs
- Capacity:** 101 GiB

A red error message is displayed at the bottom of the form:

**Warning:** persistentvolumeclaims "claim1" is forbidden: exceeded quota: storagequota, requested: requests.storage=101Gi, used: requests.storage=0, limited: requests.storage=100Gi

The form has 'Create' and 'Cancel' buttons at the bottom.

# PVC scenario 2

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

```
1  apiVersion: v1
2  kind: ResourceQuota
3  metadata:
4    name: storagequota
5  spec:
6    hard:
7      persistentvolumeclaims: "100"
8      requests.storage: "100Gi"
```

# Example 2 : PVC (2)

Health Check  
Periodically make a request to the container to see if it is alive and responding correctly.

Volumes  
Persist and share data separate from the lifecycle of an individual container.

Volume Name  
claim1

Volume Type  
New Volume Claim [Edit](#)

Remove Volume

Mount Point \*  
/home

Sub Path in Volume

Read-Only ☐

Add Mount

Add Volume... [v](#)

Scaling/Upgrade Policy  
Configure how pods are replaced when performing an upgrade.

Show advanced options

**Warning:** persistentvolumeclaims "claim1" is forbidden: maximum storage usage per PersistentVolumeClaim is 10Gi, but request is 11Gi

Launch Cancel

v2.3.3 [Help & Docs](#) [Forums](#) [Slack](#) [File an Issue](#) [English](#) [Download CLI](#)

alpine-stressng-cpu-....zip [Open file](#)

Type here to search

1:14 AM 4/10/2020

# Request and limit

```
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

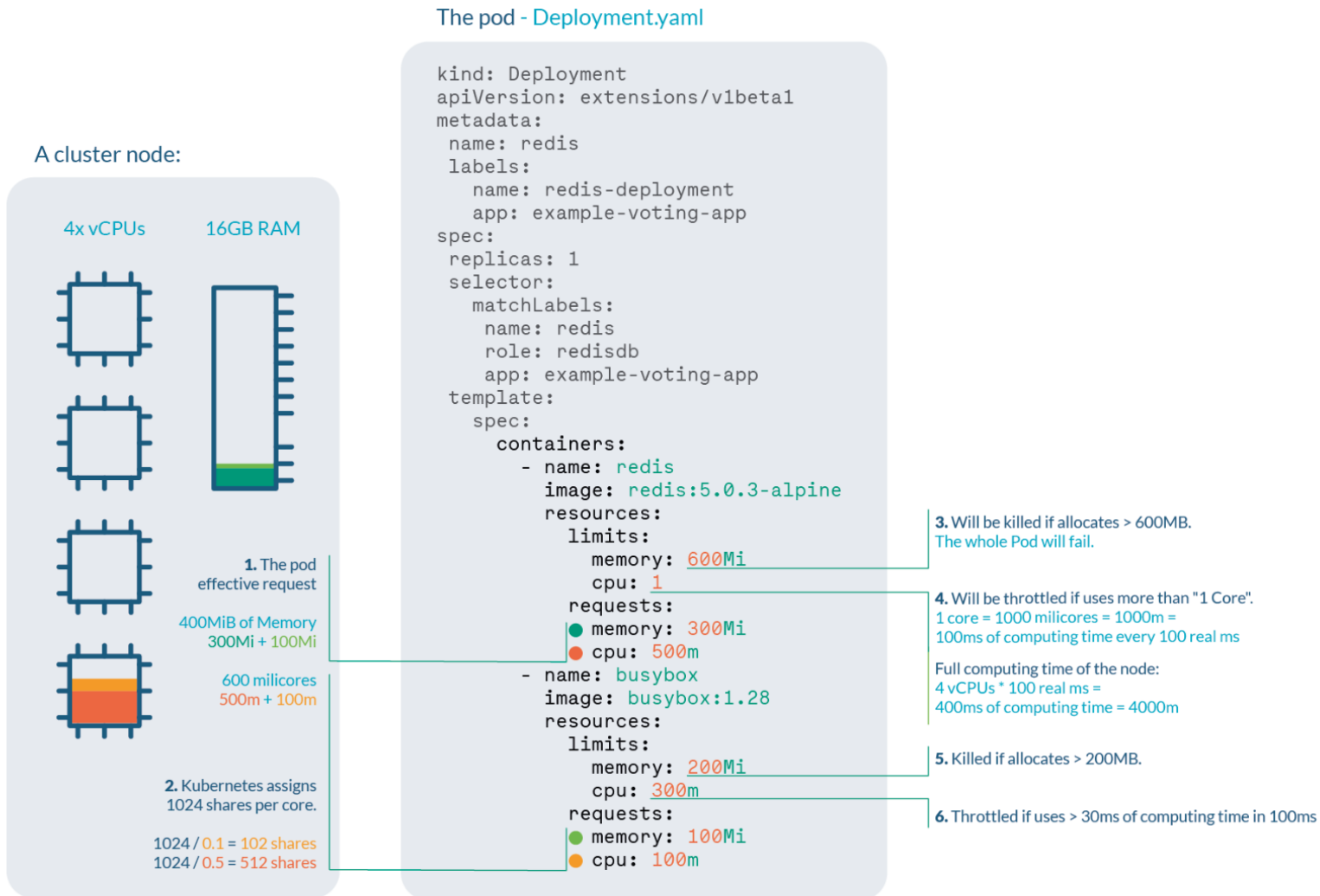
- Requests and limits are the mechanisms Kubernetes uses to control resources such as CPU and memory.



## 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



# 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>