Kubernetes Taller 1

1. Primeros pasos.

Para poder comenzar con Kubernetes primero debemos instalar:

- Kubectl (https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/)
- Minikube (https://minikube.sigs.k8s.io/docs/start/)

Una vez instaladas iniciamos minikube.

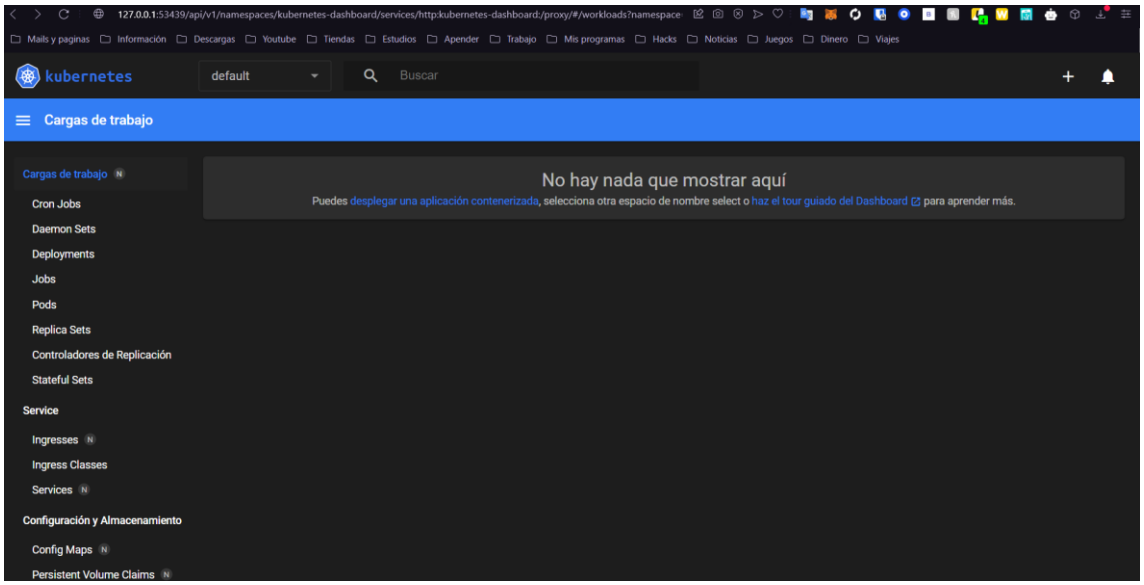
```
C:\Users\josem>minikube start --driver=hyperv
* minikube v1.28.0 en Microsoft Windows 11 Pro For Workstations 10.0.22621 Build 22621
* Using the hyperv driver based on user configuration
* Starting control plane node minikube in cluster minikube
* Creando hyperv VM (CPUs=2, Memory=6000MB, Disk=20000MB) ...
* Preparando Kubernetes v1.25.3 en Docker 20.10.20...
  - Generando certificados y llaves
  - Iniciando plano de control
  - Configurando reglas RBAC...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Complementos habilitados: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Tras su inicialización, podemos comprobar que todo esta correctamente de la siguiente manera:

```
C:\Users\josem>minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured


C:\Users\josem>Kubectl get service
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   22m
```

Si todo está correctamente, podemos entrar en el Dashboard.

```
C:\Users\josem>minikube dashboard
* Habilitando dashboard
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
* Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


* Verifying dashboard health ...
* Launching proxy ...
* Verifying proxy health ...
* Opening http://127.0.0.1:53439/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in yo
ur default browser...
```



2. Creación namespace

Para la creación del namespace usamos el comando create namespace junto al nombre. Si usamos el comando gt ns podemos ver que se ha creado correctamente.

```
C:\Users\josem>Kubectl create namespace bootcamp
namespace/bootcamp created

C:\Users\josem>Kubectl get ns
NAME                   STATUS   AGE
bootcamp               Active   9s
default                Active   109m
kube-node-lease        Active   109m
kube-public            Active   109m
kube-system            Active   109m
kubernetes-dashboard   Active   101m
```

3. Creación ResourceQuota

Para la creación del objeto ResourceQuota primero debemos crear un archivo yalm.

quota-cpu-bootcamp.yaml

C: > Users > josem > Desktop > Desarrollo > NTT DATA > Bootcamp > App > nttdatacenter-kubernetes-t1-jmmd > quota-cpu-bootcamp.yaml

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

Una vez creado, le asignamos esa cuota al namespace creado anteriormente. Para que funcione correctamente debemos situarnos con el cmd en la misma carpeta en la que esta en archivo.

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl apply -f quota-cpu-boot
camp.yaml --namespace=bootcamp
resourcequota/quota-cpu-bootcamp created
```

Comprobamos la información del ResourceQuota creado:

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl get resourcequota quota-cpu-bootcamp --namespace=bootcamp -
-output=yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"ResourceQuota","metadata":{"annotations":{},"name":"quota-cpu-bootcamp","namespace":"bootcamp"},"spec":{"hard":{"limits.cpu
":"2","limits.memory":"2Gi","requests.cpu":"1","requests.memory":"1Gi"}}}
  creationTimestamp: "2022-11-17T11:32:03Z"
  name: quota-cpu-bootcamp
  namespace: bootcamp
  resourceVersion: "5641"
  uid: 7f9a04be-b570-4a79-8e04-4ac87c8cc405
spec:
  hard:
    limits.cpu: "2"
    limits.memory: 2Gi
    requests.cpu: "1"
    requests.memory: 1Gi
status:
  hard:
    limits.cpu: "2"
    limits.memory: 2Gi
    requests.cpu: "1"
    requests.memory: 1Gi
  used:
    limits.cpu: "0"
    limits.memory: "0"
    requests.cpu: "0"
    requests.memory: "0"
```

4. Creación del Pod

Al igual que en el paso anterior, debemos crear un archivo yalm.

pod-bootcamp.yaml

C: > Users > josem > Desktop > Desarrollo > NTT DATA > Bootcamp > App > nttdatacenter-kubernetes-t1-jmmd > pod-bootcamp.yaml

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

Una vez hecho, aplicamos el pod al namescape creado.

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl apply -f pod-bootcamp.yaml --namespace=bootcamp
pod/nginx created
```

Comprobamos su estado.

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl get pod nginx --namespace=bootcamp
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          46s
```

5. Creación de Deployment

Creamos de nuevo un archivo yalm

deployment-bootcamp.yaml

C: > Users > josem > Desktop > Desarrollo > NTT DATA > Bootcamp > App > nttdatacenter-kubernetes-t1-jmmd > deployment-bootcamp.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-jmmd
  labels:
    app: nginx-jmmd
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx-jmmd
  template:
    metadata:
      labels:
        app: nginx-jmmd
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Una vez hecho, aplicamos el deployment al namescape creado y comprobamos si se ha creado. Aún así no se ejecutan, problema seguramente debido a las limitaciones. Si se hace pero fuera de este namespace sí que se inician los 3.

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl apply -f deployment-bootcamp.yaml --namespace=bootcamp
deployment.apps/deployment-jmmd created

C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl get deployment --namespace=bootcamp
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment-jmmd   0/3     0            0           21s
```

Si no se mete dentro del namespace:

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl delete -f deployment-bootcamp.yaml --namespace=bootcamp
deployment.apps "deployment-jmmd" deleted

C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl apply -f deployment-bootcamp.yaml
deployment.apps/deployment-jmmd created

C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/deployment-jmmd-74ff6b46f-cwd48   1/1     Running   0          8s
pod/deployment-jmmd-74ff6b46f-m7g4b   1/1     Running   0          8s
pod/deployment-jmmd-74ff6b46f-vgvzh   1/1     Running   0          8s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   176m

NAME                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/deployment-jmmd   3/3     3            3           8s

NAME                                       DESIRED   CURRENT   READY   AGE
replicaset.apps/deployment-jmmd-74ff6b46f   3         3         3       8s
```

NOTA: Al quitar la cuota ya funciona correctamente, así que a partir de este punto seguiré haciendo el ejercicio sin la cuota puesta en el namespace.

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl get pods --all-namespaces
NAMESPACE              NAME                                  READY   STATUS    RESTARTS   AGE
bootcamp               deployment-jmmd-74ff6b46f-h6bwj       1/1     Running   0          3m39s
bootcamp               deployment-jmmd-74ff6b46f-sbkbl       1/1     Running   0          3m39s
bootcamp               deployment-jmmd-74ff6b46f-xfrrc       1/1     Running   0          3m39s
```

6. Creación del Servicio

De nuevo creamos el archivo yalm. No le he puesto un nodeport por defecto en ports.

service-bootcamp.yaml | deployment-bootcamp.yaml

C: > Users > josem > Desktop > Desarrollo > NTT DATA > Bootcamp > App > nttdatacenter-kubernetes-t1-jmmd > service-bootcamp.yaml

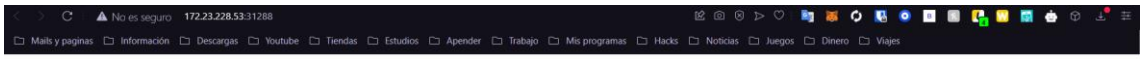
```yaml
apiVersion: v1
kind: Service
metadata:
  name: service-jmmd
  namespace: bootcamp
spec:
  type: NodePort
  selector:
    app: nginx-jmmd
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Aplicamos el servicio.

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl apply -f service-bootcamp.yaml
service/service-jmmd created

C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl get services -n bootcamp
NAME          TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service-jmmd   NodePort   10.101.133.101   <none>        80:31288/TCP   12s

C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>kubectl describe service service-jmmd -n bootcamp
Name:                     service-jmmd
Namespace:                bootcamp
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx-jmmd
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.101.133.101
IPs:                      10.101.133.101
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31288/TCP
Endpoints:                172.17.0.6:80,172.17.0.7:80,172.17.0.8:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Para poder ver la web, debemos entrar con la ip de minikube. Para ello debemos ejecutar el siguiente comando:

```
C:\Users\josem\Desktop\Desarrollo\NTT DATA\Bootcamp\App\nttdatacenter-kubernetes-t1-jmmd>minikube service service-jmmd -n bootcamp --url
http://172.23.228.53:31288
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.