

PTI práctica 4: Microservices

Joan Llonch Majó

Luis Jesús Valverde Zavaleta

Entorno y configuración de la práctica

El entorno que hemos utilizado para la realización de la práctica es una imagen proporcionada la UPC que es la Ubuntu22v3r1 y que la hemos corrido en la máquina virtual VirtualBox.

Para configurar la maquina y tenerla preparada para la realización de la práctica, seguimos los pasos especificados en el enunciado de la práctica. Para esta práctica haremos uso de las siguientes herramientas:

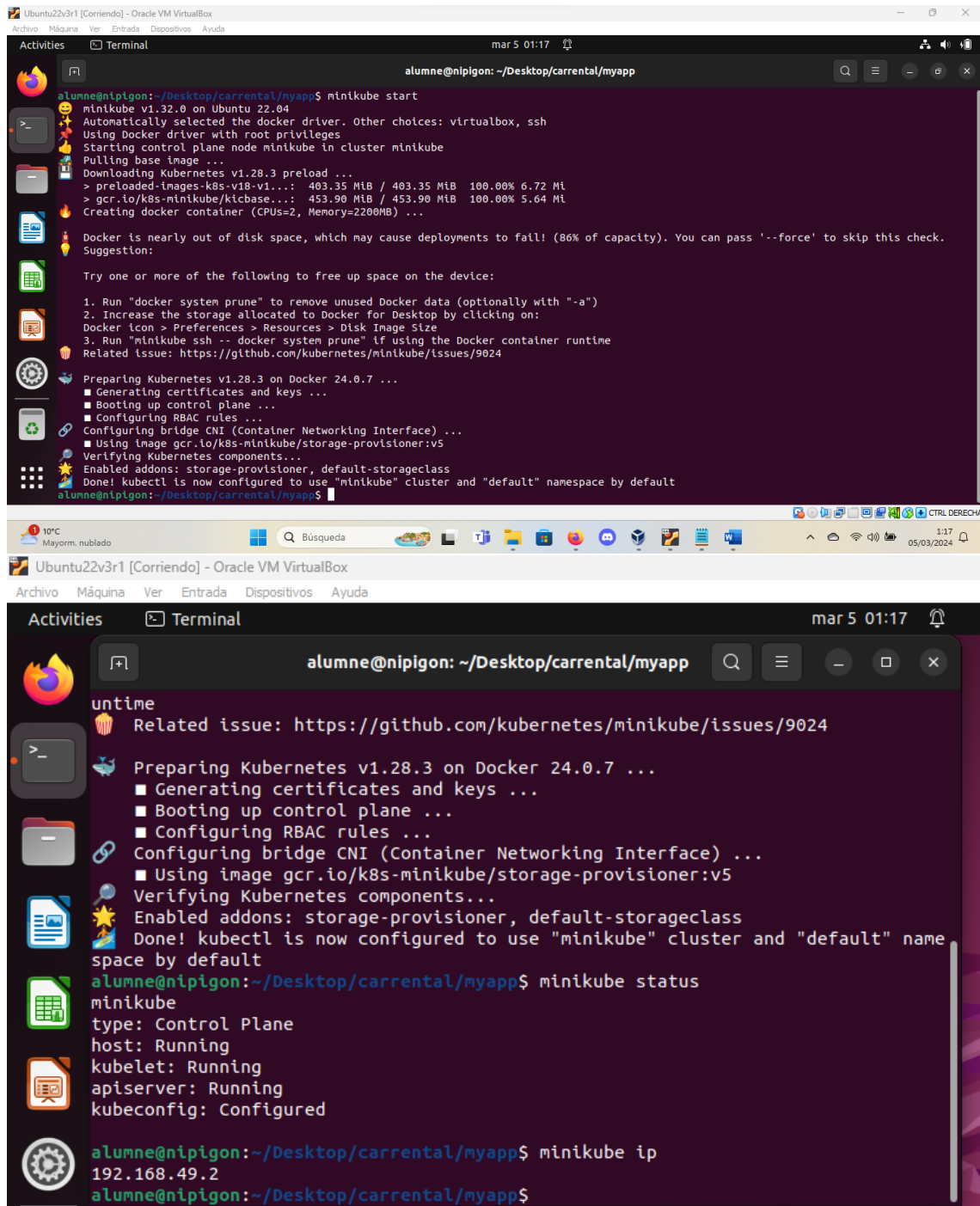
- Minikube que es una herramienta que se encargará de correr un single-node Kubernetes cluster, no estaba instalada en nuestro entorno así que la instalamos siguiendo los siguientes comandos:
 - `curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64`
 - `chmod +x minikube`
 - `sudo mv minikube /usr/local/bin`
- Docker, que ya lo teníamos en nuestra maquina de la práctica anterior y que nos servirá para empaquetar nuestra aplicación de renta de coches en un contenedor, nos permitirá crear la imagen y hacerla correr en un contenedor básicamente.
- Kubectl, que es una herramienta de comando de línea que nos sirve para controlar el cluster de Kubernete, esta herramienta si que no la teníamos en nuestro entorno y la instalamos con los siguientes comandos:
 - `curl -LO https://storage.googleapis.com/kubernetes-release/release/\$\(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt\)/bin/linux/amd64/kubectl`
 - `chmod +x ./kubectl`
 - `sudo mv ./kubectl /usr/local/bin/kubectl`

Después de instalar todo y tener todas las herramientas listas, hicimos el tutorial que nos ponía de ejemplo el enunciado de la práctica para la familiarización con estas tecnologías, teniendo un programa muy simple que solo mostraba "HELLO WORLD!", para hacer esto seguimos los pasos marcados que eran los de iniciar el minikube, crear la imagen, ponerla a correr, etc. Todos estos pasamos que hicimos en el tutorial los aplicaremos a nuestra API creada en el laboratorio anterior en el que consistía en una API de alquiler de coches.

Realización de la práctica

Para la realización de esta práctica utilizamos la API creada en el laboratorio anterior, con lo que el código y el Dockerfile no serán mostrados en esta práctica de forma explícita (para su consulta de estos revisar la entrega de la práctica REST API), tampoco hubo ninguna modificación de estos en esta práctica.

Lo primero que hicimos fue iniciar el Minikube con el comando “start minikube”, y esto nos generó una MV para ejecutar un único nodo de Kubernetes en nuestro sistema local. En la siguiente imagen podréis observar el estado de minikube y la ip de la MV de minikube.



```
alumno@nipigon: ~/Desktop/carrental/myapp
minikube start
minikube v1.32.0 on Ubuntu 22.04
Automatically selected the docker driver. Other choices: virtualbox, ssh
Using Docker driver with root privileges
Starting control plane node minikube in cluster minikube
Pulling base image ...
Downloading Kubernetes v1.28.3 preload ...
> preloaded-images-k8s-v18-v1...: 403.35 MiB / 403.35 MiB 100.00% 6.72 Mi
> gcr.io/k8s-minikube/kicbase...: 453.90 MiB / 453.90 MiB 100.00% 5.64 Mi
Creating docker container (CPUs=2, Memory=2200MB) ...
Docker is nearly out of disk space, which may cause deployments to fail! (86% of capacity). You can pass '--force' to skip this check.
Suggestion:
Try one or more of the following to free up space on the device:
1. Run "docker system prune" to remove unused Docker data (optionally with "-a")
2. Increase the storage allocated to Docker for Desktop by clicking on:
Docker icon > Preferences > Resources > Disk Image Size
3. Run "minikube ssh -- docker system prune" if using the Docker container runtime
Related issue: https://github.com/kubernetes/minikube/issues/9024
Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
■ Generating certificates and keys ...
■ Booting up control plane ...
■ Configuring RBAC rules ...
■ Configuring bridge CNI (Container Networking Interface) ...
■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
Verifying Kubernetes components...
Enabled addons: storage-provisioner, default-storageclass
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
alumno@nipigon:~/Desktop/carrental/myapp$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
alumno@nipigon:~/Desktop/carrental/myapp$ minikube ip
192.168.49.2
alumno@nipigon:~/Desktop/carrental/myapp$
```

Como hemos dicho el código ya lo teníamos que estaba en `assignment.js` y el `Dockerfile` también. Para hacer que la imagen sea accesible desde Minikube directamente ejecutamos el comando **“eval \$(minikube Docker-env)”**, así conseguimos que el entorno Docker apunte al Daemon Docker integrado en Minikube.

Hecho esto procedemos a la creación de la imagen con el siguiente comando: **“docker build -f Dockerfile -t carrental:1.0 .”**, luego de la creación de la imagen, la pusimos a correr el contenedor donde estaba el microservicio con el comando: **“docker run --name carrental -d -p 8080:8080 carrental:1.0”**.

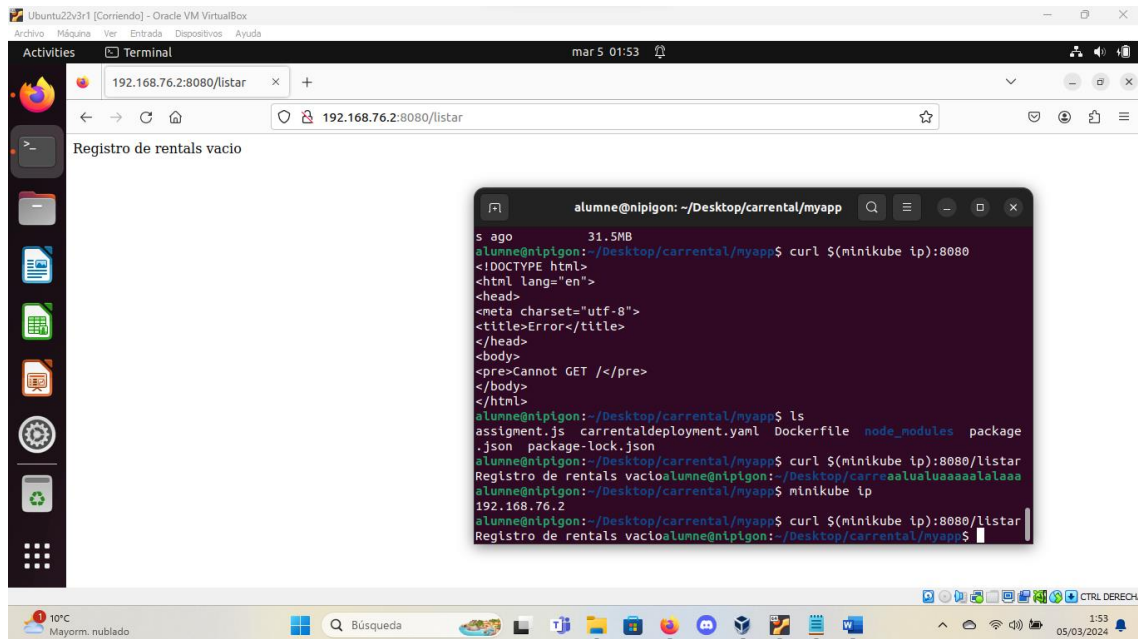
```
Ubuntu22v3r1 [Corriendo] - Oracle VM VirtualBox
Archivos Máquina Ver Entrada Dispositivos Ayuda
mar 5 01:45
Activities Terminal
alumno@nipigon: ~/Desktop/carrental/myapp

alumno@nipigon:~/Desktop/carrental/myapp$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
8170c71eedf937ec2fdfeb54e61fce3079b79090f2a97080718bd7184722dbc  carrental:1.0  "docker-entrypoint.s..."  8 seconds ago  Up 7 seconds  0.0.0.0:8080->8080/tcp    carren
tal
f5c36eb939dc   6e38f40d628d   "/storage-provisioner"   4 minutes ago  Up 4 minutes  -                        k8s_st
orage-provisioner_storage-provisioner_kube-system_1134df07-2781-47aa-8015-0df183b7f584_1  5c1a36afda52   ead0a4a53df8   "/coredns -conf /etc..."  4 minutes ago  Up 4 minutes  -                        k8s_co
redns_coredns-5dd5756b68-jcr5l_kube-system_9563d558-a4ec-4c39-80a3-5c15dd4d1413e_0  eabf6865dade   6e38f40d628d   "/storage-provisioner"   4 minutes ago  Exited (1) 4 minutes ago  -                        k8s_st
orage-provisioner_storage-provisioner_kube-system_1134df07-2781-47aa-8015-0df183b7f584_0  34c89cac9a1f   bfc896cf80fb   "/usr/local/bin/kube..."  4 minutes ago  Up 4 minutes  -                        k8s_ku
be-proxy_kube-proxy-h2tsq_kube-system_4b358fc1-cbdf-485e-90b9-50b548970670_0  77b4c1031bdf   registry.k8s.io/pause:3.9  "/pause"                4 minutes ago  Up 4 minutes  -                        k8s_PO
D_coredns-5dd5756b68-jcr5l_kube-system_9563d558-a4ec-4c39-80a3-5c15dd4d1413e_0  b23436ecbf89   registry.k8s.io/pause:3.9  "/pause"                4 minutes ago  Up 4 minutes  -                        k8s_PO
D_storage-provisioner_kube-system_1134df07-2781-47aa-8015-0df183b7f584_0  b8440322882e   registry.k8s.io/pause:3.9  "/pause"                4 minutes ago  Up 4 minutes  -                        k8s_PO
D_kube-proxy-h2tsq_kube-system_4b358fc1-cbdf-485e-90b9-50b548970670_0  0a7bc43c79d2   10baa1ca1706   "kube-controller-man..."  5 minutes ago  Up 5 minutes  -                        k8s_ku
be-controller-manager_kube-controller-manager-minikube_kube-system_7da72f2c2c7b27aac6cfcfd1c72da80_0  29e8a520b6d7   6d1b4fd1b182   "kube-scheduler -au..."  5 minutes ago  Up 5 minutes  -                        k8s_ku
be-scheduler_kube-scheduler-minikube_kube-system_75ac196d3709dde303d8a81c035c2c28_0  b5a91d6bed74   537434729123   "kube-apiserver --ad..."  5 minutes ago  Up 5 minutes  -                        k8s_ku
be-apiserver_kube-apiserver-minikube_kube-system_7b7974b8c1d63ee533f1aaebccfc6b59_0  f581b8587842   73deb9a3f702   "etcd --advertise-cl..."  5 minutes ago  Up 5 minutes  -                        k8s_et
cd_etcd-minikube_kube-system_62bd3844afa3b5066effe7e34214aca9_0  e38172bb37b4   registry.k8s.io/pause:3.9  "/pause"                5 minutes ago  Up 5 minutes  -                        k8s_PO
D_kube-scheduler-minikube_kube-system_75ac196d3709dde303d8a81c035c2c28_0  80599984f73b   registry.k8s.io/pause:3.9  "/pause"                5 minutes ago  Up 5 minutes  -                        k8s_PO
D_etcd-minikube_kube-system_62bd3844afa3b5066effe7e34214aca9_0  35509e4f5888   registry.k8s.io/pause:3.9  "/pause"                5 minutes ago  Up 5 minutes  -                        k8s_PO

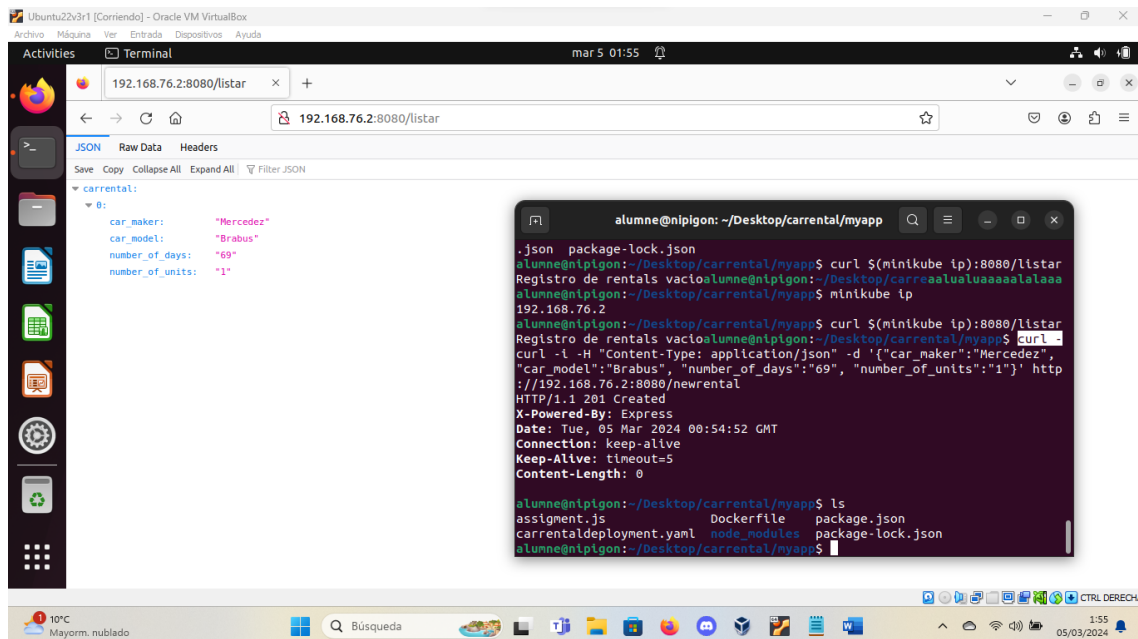
alumno@nipigon:~/Desktop/carrental/myapp$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
carrental           1.0        2e3277a09425  About a minute ago  915MB
registry.k8s.io/kube-apiserver  v1.28.3    537434729123  4 months ago    126MB
registry.k8s.io/kube-scheduler  v1.28.3    6d1b4fd1b182  4 months ago    60.1MB
registry.k8s.io/kube-controller-manager  v1.28.3    10baa1ca1706  4 months ago    122MB
registry.k8s.io/kube-proxy      v1.28.3    bfc896cf80fb  4 months ago    73.1MB
registry.k8s.io/etcd            3.5.9-0    73deb9a3f702  9 months ago    294MB
registry.k8s.io/coredns/coredns v1.10.1    ead0a4a53df8  13 months ago   53.6MB
registry.k8s.io/pause           3.9        e6f181688397  16 months ago   744kB
gcr.io/k8s-minikube/storage-provisioner  v5         6e38f40d628d  2 years ago     31.5MB

alumno@nipigon:~/Desktop/carrental/myapp$ curl $(minikube ip):8080
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET </pre>
</body>
</html>
alumno@nipigon:~/Desktop/carrental/myapp$
```

Podemos observar que tanto las imágenes y los contenedores necesarios para el funcionamiento están presentes y en funcionamiento. Después podemos ver que al ejecutar `curl $(minikube ip):8080` nos sale ese texto y es porque en nuestro código `assignment.js` no hay nada en esa dirección pero si le añadimos al comando anterior `/listar`, podemos observar esto:



Así que con esto comprobamos su funcionamiento, ya que al principio no hay ningún alquiler de coche, ahora hacemos una petición para añadir una instancia de alquiler de coche y podemos observar que al refrescar y volver a `/listar` sale la nueva instancia. También se crea el fichero json que hace de BD en este caso, pero se crea dentro de la MV de minikube suponemos.



Ahora vamos a desplegar el microservicio, y esto lo haremos siguiendo los comandos especificados en el enunciado. Vamos a crear un despliegue por defecto, y para esto necesitamos darle el nombre y la imagen del microservicio que en nuestro caso es `carrentals:1.0`, queremos correr el microservicio en un puerto correspondiente así que especificamos el 8080, el comando que ejecutamos es el siguiente: **`"kubectl create deployment carrental --image=carrental:1.0 --port=8080 --replicas=2"`**, después comprobamos que esta todo correcto listando los despliegues con el comando **`"kubectl get deployments"`**, y como podemos observar todo esta correcto y la columna `READY` nos muestra 2/2 PODS.

```
alumne@nipigon:~/Desktop/carrental/myapp$ kubectl get nodes
NAME        STATUS    ROLES    AGE   VERSION
minikube    Ready     control-plane  31m   v1.28.3
alumne@nipigon:~/Desktop/carrental/myapp$ kubectl create deployment carrental --image=carrental:1.0 --port=8080 --replicas=2
deployment.apps/carrental created
alumne@nipigon:~/Desktop/carrental/myapp$ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
carrental   2/2     2            2           7s
alumne@nipigon:~/Desktop/carrental/myapp$
```

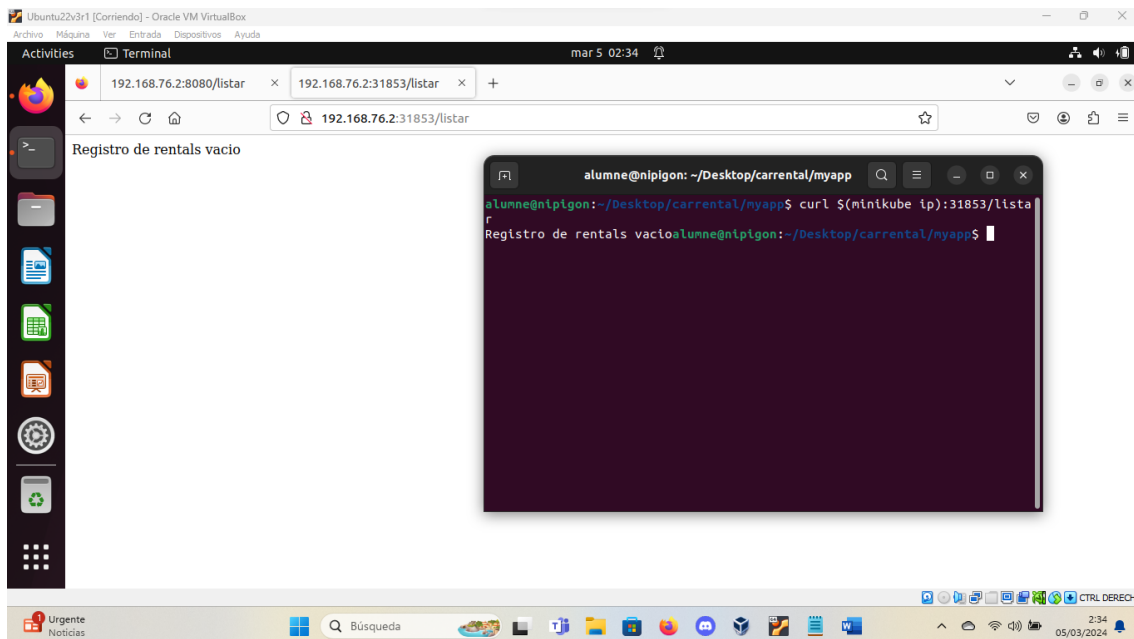
Después pasamos a la parte de los pods y todo lo relacionado con esto, lo primero la definición de pod es una instancia de nuestro microservicio, y podemos escalar multiples pods para escalar el microservicio. Pasamos a ver todos nuestros pods y darle el nombre de un pod a la variable `POD_NAME` para trabajar con este con más comodidad.

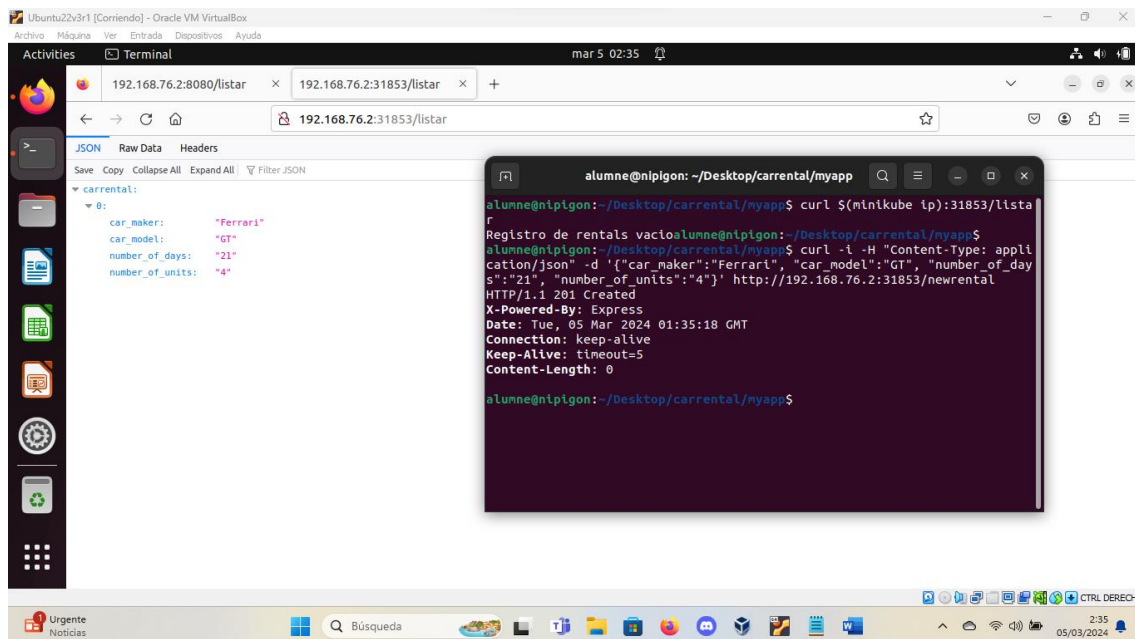
```
alumne@nipigon:~/Desktop/carrental/myapp$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
carrental-86c84d86cc-c9ff9         1/1     Running   0           3m24s
carrental-86c84d86cc-hpb8c         1/1     Running   0           3m24s
alumne@nipigon:~/Desktop/carrental/myapp$ export MYPOD=carrental-86c84d86cc-c9ff9
alumne@nipigon:~/Desktop/carrental/myapp$ echo $MYPOD
carrental-86c84d86cc-c9ff9
```

Después de esto pasamos a la parte de servicios, para aislar el microservicio de su complejidad en Kubernetes definimos una configuración de servicio. Los servicios nos permiten recibir tráfico. Hay diferentes tipos de servicios, la de por defecto es la de ClusterIP, que expone el servicio en una IP interna en el cluster (no es visible desde el exterior), nosotros nos encargaremos de crear un servicio del tipo NodePort que nos permite ser accesibles desde fuera del cluster. Nuestra ejecución de comandos fue el siguiente y obtuvimos el siguiente resultado.

```
alunne@nipigon:~/Desktop/carrental/myapp$ kubectl logs pods/$MYPOD carrental
PTI HTTP Server listening at http://localhost:8080
alunne@nipigon:~/Desktop/carrental/myapp$ kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes ClusterIP   10.96.0.1     <none>        443/TCP   46m
alunne@nipigon:~/Desktop/carrental/myapp$ kubectl expose deployment/carrental --type="NodePort" --port 8080
service/carrental exposed
alunne@nipigon:~/Desktop/carrental/myapp$ kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
carrental NodePort    10.111.185.243 <none>        8080:31853/TCP 4s
kubernetes ClusterIP   10.96.0.1     <none>        443/TCP   47m
alunne@nipigon:~/Desktop/carrental/myapp$ kubectl describe service carrental
Name:      carrental
Namespace: default
Labels:    app=carrental
Annotations: <none>
Selector:  app=carrental
Type:      NodePort
IP Family Policy: SingleStack
IP Families: IPv4
IP: 10.111.185.243
IPs: 10.111.185.243
Port: <unset> 8080/TCP
TargetPort: NodePort: <unset> 31853/TCP
Endpoints: 10.244.0.3:8080,10.244.0.4:8080
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
```

Probamos nuestra API y vemos que el comportamiento que obtenemos es el siguiente:





Por último lo que hacemos es escalar el microservicio, y para esto seguimos los siguientes comandos que observamos por la terminal:

```
alumno@nipigon: ~/Desktop/carrental/myapp$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
carrental     2/2     2             2           28m
alumno@nipigon: ~/Desktop/carrental/myapp$ kubectl scale deployments/carrental --replicas=4
deployment.apps/carrental scaled
alumno@nipigon: ~/Desktop/carrental/myapp$ kubectl get deployments
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
carrental     4/4     4             4           29m
alumno@nipigon: ~/Desktop/carrental/myapp$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP            NODE          NOMINATED NODE   READINESS GATES
carrental-86c84d86cc-c9ff9          1/1     Running   0           29m   10.244.0.4    minikube     <none>           <none>
carrental-86c84d86cc-hpb8c          1/1     Running   0           29m   10.244.0.3    minikube     <none>           <none>
carrental-86c84d86cc-jb2ph          1/1     Running   0           18s   10.244.0.5    minikube     <none>           <none>
carrental-86c84d86cc-zftzq          1/1     Running   0           18s   10.244.0.6    minikube     <none>           <none>
```

Opcional Minikube dashboard

Primero de todo chequeamos la lista de complementos disponibles con el comando ***“minikube addons list”***, y vemos que la opción de dashboard esta deshabilitada.

```
alumno@nipigon: ~/Desktop/carrental/myapp$ minikube addons list
```

ADDON NAME	PROFILE	STATUS	MAINTAINER
ambassador	minikube	disabled	3rd party (Ambassador)
auto-pause	minikube	disabled	minikube
cloud-spanner	minikube	disabled	Google
csi-hostpath-driver	minikube	disabled	Kubernetes
dashboard	minikube	disabled	Kubernetes

Después lo habilitamos con el comando “minikube addons enable dashboard”, y observamos que ahora si está habilitado.

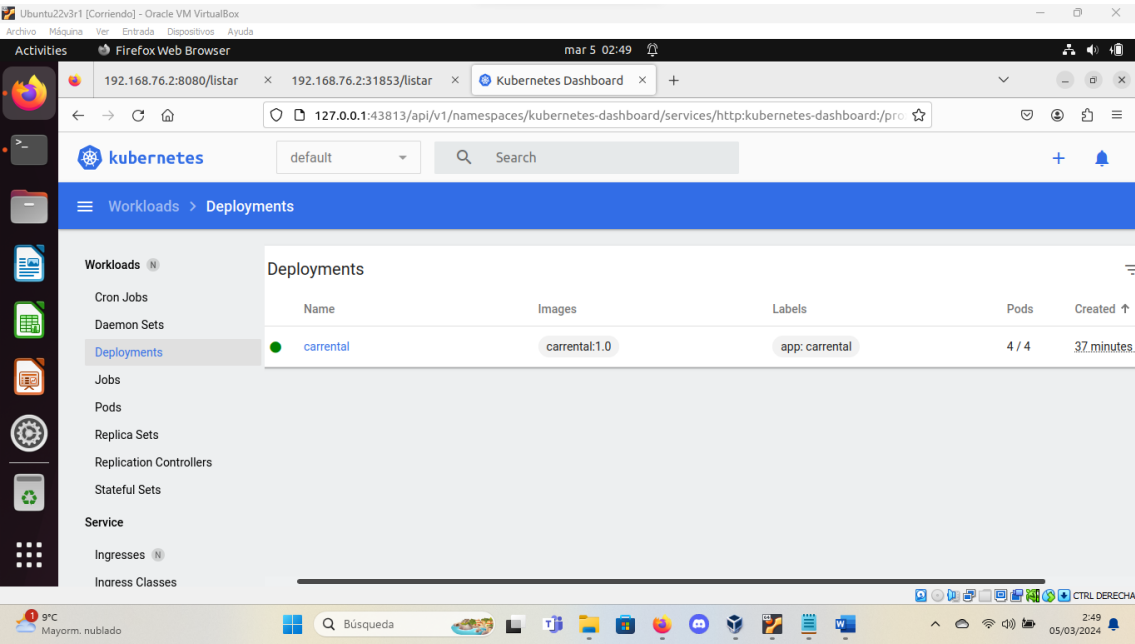
```
alunne@nlpigon:~/Desktop/carrental/myapp$ minikube addons enable dashboard
dashboard is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
■ Using image docker.io/kubernetesui/dashboard:v2.7.0
■ 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

★ The 'dashboard' addon is enabled
alunne@nlpigon:~/Desktop/carrental/myapp$ minikube addons list
```

ADDON NAME	PROFILE	STATUS	MAINTAINER
ambassador	minikube	disabled	3rd party (Ambassador)
auto-pause	minikube	disabled	minikube
cloud-spanner	minikube	disabled	Google
csi-hostpath-driver	minikube	disabled	Kubernetes
dashboard	minikube	enabled ✓	Kubernetes

En las siguientes imágenes podemos ver una interfaz donde se ven reflejados los 4 pods creados, el único deployment carrental creado y el servicio carrental tipo NodePort más el servicio por defecto kubernetes. Aquí esta la foto de los Deploiments:



Aquí podemos observar los Pods:

The screenshot shows the Kubernetes Dashboard interface. The left sidebar has a menu with 'Workloads' expanded, showing 'Pods' as the selected item. The main content area is titled 'Pods' and contains a table with the following data:

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
carrental-86c84d86cc-jbzbh	carrental:1.0	app: carrental	minikube	Running	0	-	-	8 m...
carrental-86c84d86cc-zftzq	carrental:1.0	app: carrental	minikube	Running	0	-	-	8 m...
carrental-86c84d86cc-c9ff9	carrental:1.0	app: carrental	minikube	Running	0	-	-	37 m...
carrental-86c84d86cc-hpb8c	carrental:1.0	app: carrental	minikube	Running	0	-	-	37 m...

Y por último aquí podemos ver lo Services:

The screenshot shows the Kubernetes Dashboard interface. The left sidebar has a menu with 'Service' expanded, showing 'Services' as the selected item. The main content area is titled 'Services' and contains a table with the following data:

Name	Labels	Type	Cluster IP	Internal Endpoints	External Endpoints	Created
carrental	app: carrental	NodePort	10.111.185.243	carrental:8080 TCP carrental:31853 TCP	-	22 min...
kubernetes	component: apiserver provider: kubernetes	ClusterIP	10.96.0.1	kubernetes:443 TCP kubernetes:0 TCP	-	an h...

Opcional Using a manifest file

Principalmente este apartado opcional se basa en jugar con los archivos de manifiesto, uno para los despliegues y otro para los servicios.

Primero con el deployment, obtenemos el yaml ejecutando ***"kubectl get deployments/carrental -o=yaml"***, el resultado lo metemos en el fichero carrentaldeployment.yaml. Luego eliminamos el deployment y lo volvemos a crear con el fichero carrentaldeployment.yaml, ejecutando el comando ***"kubectl apply -f carrentaldeployment.yaml"***.

```
alunne@nipigon:~/Desktop/carrental/nyapp$ vim carrentaldeployment.yaml
alunne@nipigon:~/Desktop/carrental/nyapp$ rm carrentaldeployment.yaml
alunne@nipigon:~/Desktop/carrental/nyapp$ vim carrentaldeployment.yaml
alunne@nipigon:~/Desktop/carrental/nyapp$ ls
assignment.js carrentaldeployment.yaml Dockerfile node_modules package.json package-lock.json
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl delete deployment carrental
deployment.apps "carrental" deleted
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl get deployment
No resources found in default namespace.
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl get deployments
No resources found in default namespace.
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl apply -f carrentaldeployment.yaml
deployment.apps/carrental created
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
carrental 0/4      4            0           3s
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
carrental 4/4      4            4           8s
alunne@nipigon:~/Desktop/carrental/nyapp$
```

Ahora con el servicio, hicimos lo mismo, solo que ejecutando otros comandos, obtenemos el yaml ejecutando ***"kubectl get services/carrental -o=yaml"***, el resultado lo metemos en el fichero carrentalservice.yaml. Luego eliminamos el servicio y lo volvemos a crear con el fichero carrentalservice.yaml, ejecutando el comando ***"kubectl apply -f carrentalservice.yaml"***.

```
alunne@nipigon:~/Desktop/carrental/nyapp$ vim carrentalservice.yaml
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl delete service carrental
service "carrental" deleted
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes ClusterIP  10.96.0.1     <none>        443/TCP    90m
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl apply -f carrentalservice.yaml
service/carrental created
alunne@nipigon:~/Desktop/carrental/nyapp$ kubectl get services
NAME      TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
carrental NodePort    10.111.185.243 <none>        8080:31853/TCP 3s
kubernetes ClusterIP  10.96.0.1     <none>        443/TCP        90m
alunne@nipigon:~/Desktop/carrental/nyapp$
```

Bueno y al final vemos el contenido de los yaml y vemos que contienen la información del deployment y el servicio correspondiente.