

EXAMEN FINAL "A DOMICILIO": KUBERNETES

Kubernetes es un servicio de código abierto que sirve para desplegar, gestionar y hacer fácilmente escalables nuestras aplicaciones, ejecutándolas en contenedores (muy similar a lo que hacemos con docker).

En este laboratorio de Kubernetes voy a crear un clúster de contenedores usando para ello la aplicación de minikube.

En kubernetes, distribuimos nuestros contenedores entre diferentes nodos precisamente para hacer nuestra aplicación escalable, en este caso, con minikube, voy a generar un clúster en un solo nodo local, esto es útil cuando queremos probar nuestra aplicación en un entorno controlado y queremos realizar pruebas, simulando un entorno "real".

****Importante antes de comenzar la instalación, será necesario tener instalado docker y tener añadido a nuestro usuario administrador al grupo de docker**

Vamos pues con la instalación, descargamos minikube con el siguiente comando:

curl -LO <https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64>

```
davidrl@examenDespliegues: ~  
root@examenDespliegues:/home/davidrl# curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64  
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current  
           Dload  Upload   Total   Spent    Left   Speed  
100  119M  100  119M    0     0  9587k      0  0:00:12  0:00:12 --:--:-- 10.6M
```

Lo instalamos con el siguiente comando:

sudo install minikube-linux-amd64 /usr/local/bin/minikube

```
root@examenDespliegues:/home/davidrl# sudo install minikube-linux-amd64 /usr/local/bin/minikube  
root@examenDespliegues:/home/davidrl#
```

Una vez instalado, podemos comprobar la versión con el siguiente comando:

minikube versión

```
davidrl@examenDespliegues: ~  
root@examenDespliegues:/home/davidrl# minikube version  
minikube version: v1.35.0  
commit: dd5d320e41b5451cdf3c01891bc4e13d189586ed-dirty  
root@examenDespliegues:/home/davidrl#
```

Ahora que ya tenemos minikube instalado, vamos a iniciar nuestro clúster, para ello con nuestro usuario administrador ejecutamos el siguiente comando:

minikube start

```
davidrl@examenDespliegues: ~$ minikube start
minikube v1.35.0 on Debian 12.7 (vbox/amd64)
Automatically selected the docker driver
Using Docker driver with root privileges
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.46 ...
Downloading Kubernetes v1.32.0 preload ...
> preloaded-images-k8s-v18-v1...: 333.57 MiB / 333.57 MiB 100.00% 5.04 Mi
> gcr.io/k8s-minikube/kicbase...: 500.31 MiB / 500.31 MiB 100.00% 5.03 Mi
Creating docker container (CPUs=2, Memory=2200MB) ...
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
  • Generating certificates and keys ...
  • Booting up control plane ...
  • Configuring RBAC rules ...
Configuring bridge CNI (Container Networking Interface) ...
Verifying Kubernetes components...
  • Using image gcr.io/k8s-minikube/storage-provisioner:v5
E0220 21:09:22.339424 5235 start.go:160] Unable to scale down deployment "coredns" in namespace "kube-system" to 1 replica: non-retryable failure while rescaling coredns deployment: Operation cannot be fulfilled on deployments.apps "coredns": the object has been modified; please apply your changes to the latest version and try again
Enabled addons: storage-provisioner, default-storageclass
kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
davidrl@examenDespliegues: ~$
```

A continuación, vamos a descargar la herramienta de kubectl de minikube para poder gestionar nuestro clúster, para ello ejecutamos el siguiente comando:

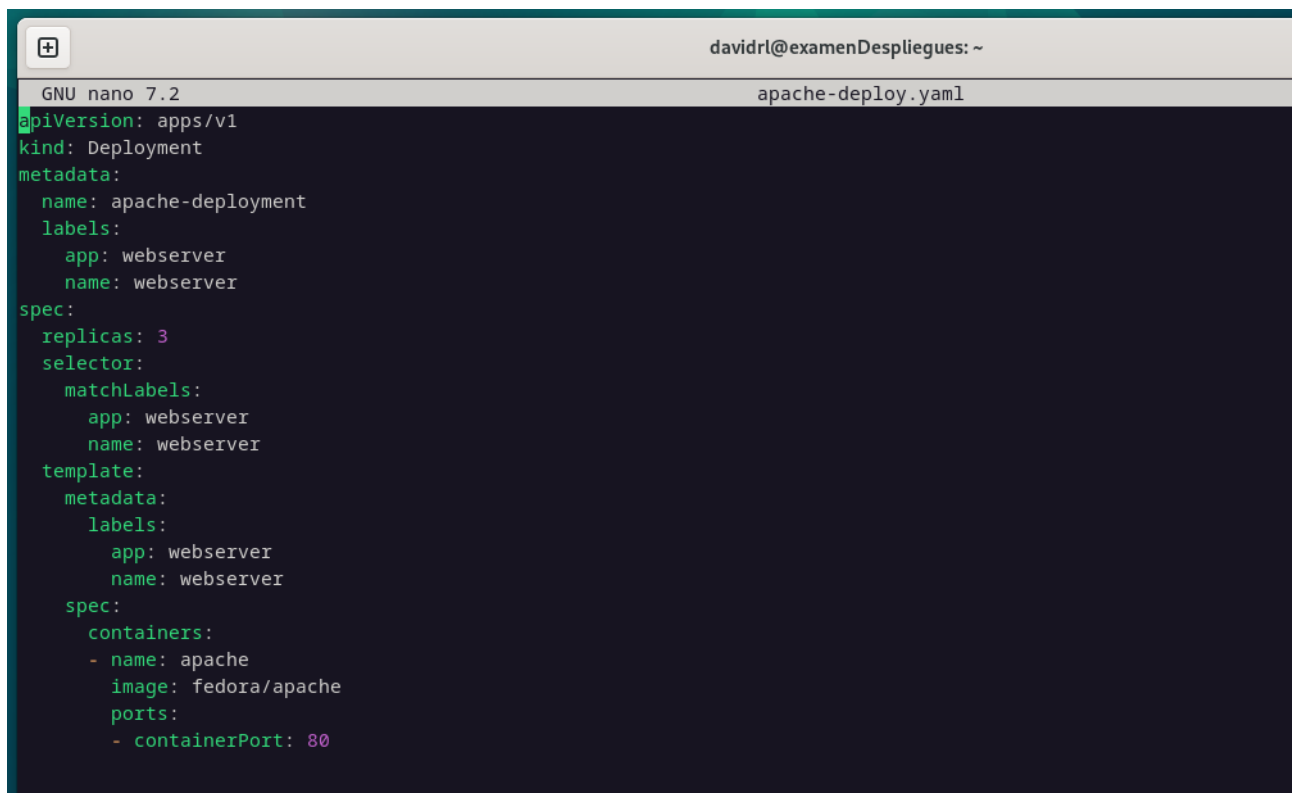
minikube kubectl -- get pods -A

```
davidrl@examenDespliegues: ~$ minikube kubectl -- get pods -A
> kubectl.sha256: 64 B / 64 B [-----] 100.00% ? p/s 0s
> kubectl: 54.67 MiB / 54.67 MiB [-----] 100.00% 9.72 MiB p/s 5.8s
NAMESPACE      NAME                                READY   STATUS    RESTARTS      AGE
kube-system    coredns-668d6bf9bc-bmbvg           1/1     Running   1 (66s ago)    12m
kube-system    coredns-668d6bf9bc-s2lkn           1/1     Running   1 (66s ago)    12m
kube-system    etcd-minikube                      1/1     Running   1 (71s ago)    12m
kube-system    kube-apiserver-minikube             1/1     Running   1 (61s ago)    12m
kube-system    kube-controller-manager-minikube    1/1     Running   1 (71s ago)    12m
kube-system    kube-proxy-zb248                   1/1     Running   1 (71s ago)    12m
kube-system    kube-scheduler-minikube            1/1     Running   1 (71s ago)    12m
kube-system    storage-provisioner                 1/1     Running   3 (53s ago)    12m
```

Bien, ya tenemos nuestro clúster, ahora vamos a desplegar un servicio apache en él, para ello debemos crear un archivo de configuración YAML que nos sirva para el despliegue.

La estructura de nuestro YAML será la siguiente:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: webserver
    name: webserver
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webserver
      name: webserver
  template:
    metadata:
      labels:
        app: webserver
        name: webserver
    spec:
      containers:
        - name: apache
          image: fedora/apache
          ports:
            - containerPort: 80
```



```
GNU nano 7.2 apache-deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: webserver
    name: webserver
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webserver
      name: webserver
  template:
    metadata:
      labels:
        app: webserver
        name: webserver
    spec:
      containers:
        - name: apache
          image: fedora/apache
          ports:
            - containerPort: 80
```

Ahora que tenemos nuestro archivo de configuración YAML, vamos a desplegarlo utilizando el siguiente comando:

minikube kubectl -- apply -f apache-deploy.yaml

```
davidrl@examenDespliegues:~$ minikube kubectl -- apply -f apache-deploy.yaml
deployment.apps/apache-deployment created
davidrl@examenDespliegues:~$
```

Ahora vamos a comprobar el estado del despliegue, ejecutamos el siguiente comando:

minikube kubectl -- get deployments

```
davidrl@examenDespliegues:~$ minikube kubectl -- get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
apache-deployment   3/3     3             3           109s
davidrl@examenDespliegues:~$
```

Ahora que está desplegado, tenemos varias maneras de comprobar su correcto funcionamiento:

minikube kubectl -- get nodes

```
davidrl@examenDespliegues:~$ minikube kubectl -- get nodes
NAME     STATUS    ROLES    AGE   VERSION
minikube Ready    control-plane  49m   v1.32.0
davidrl@examenDespliegues:~$
```

Este comando nos da información relevante sobre los nodos en nuestro clúster.

minikube kubectl get pods

```

davidrl@debian12:~$ minikube kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
apache-deployment-748cb5587-9ptfh  1/1     Running   0           6m5s
apache-deployment-748cb5587-j2qzf  1/1     Running   0           6m5s
apache-deployment-748cb5587-s6tqh  1/1     Running   0           6m5s

```

Este comando nos da información sobre los pods en ejecución, un pod es básicamente uno o más contenedores que se ejecutan a través de kubernetes, estos son "efímeros" y kubernetes puede redespugarlos para evitar errores.

Ahora debemos crear un servicio para exponer nuestro apache, creare un apache-service.yaml que usare para publicar el servicio, aquí pongo el contenido de dicho yaml:

```

apiVersion: v1
kind: Service
metadata:
  name: apache-service
spec:
  selector:
    app: webserver
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30080

```

```

GNU nano 7.2                                davidrl@debian12: ~
                                           apache-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: apache-service
spec:
  selector:
    app: webserver
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30080

```

Ahora voy a iniciar el servicio, lo hago con el siguiente comando:

minikube kubectl -- apply -f apache-service.yaml

```
davidrl@debian12:~$ minikube kubectl -- apply -f apache-service.yaml
service/apache-service created
```

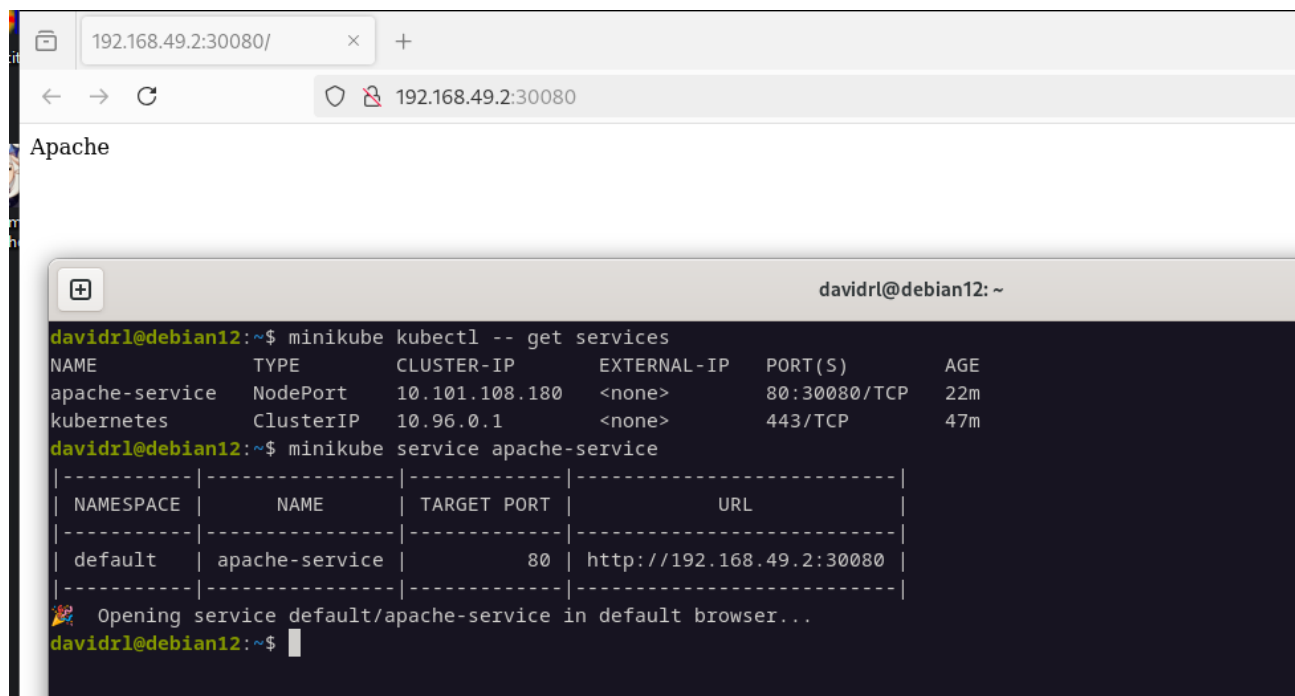
Podemos listar los servicios de nuestro minikube usando el siguiente comando:

minikube kubectl -- get services

```
davidrl@debian12:~$ minikube kubectl -- get services
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
apache-service  NodePort    10.101.108.180 <none>        80:30080/TCP     9s
kubernetes      ClusterIP   10.96.0.1     <none>        443/TCP          24m
```

Ahora podemos hacer la prueba a acceder al apache, para ello usamos el siguiente comando:

minikube kubectl -- get services



192.168.49.2:30080/ x +

192.168.49.2:30080

Apache

```
davidrl@debian12: ~
davidrl@debian12:~$ minikube kubectl -- get services
NAME           TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
apache-service  NodePort    10.101.108.180 <none>        80:30080/TCP     22m
kubernetes      ClusterIP   10.96.0.1     <none>        443/TCP          47m
davidrl@debian12:~$ minikube service apache-service
-----|-----|-----|-----|
| NAMESPACE | NAME       | TARGET PORT | URL                               |
|-----|-----|-----|-----|
| default   | apache-service | 80          | http://192.168.49.2:30080      |
|-----|-----|-----|-----|
🌐 Opening service default/apache-service in default browser...
davidrl@debian12:~$
```

Podemos ver que es bastante cutre de entrada, ahora voy a realizar un cambio en la página que muestra, para realizar ese cambio nuevamente voy a cargar un yaml, en este caso voy a crear algo llamado ConfigMap, este archivo me permite montar archivos dentro del contenedor de mis pod de kubernetes o de minikube en este caso le llamare apache-configmap.yaml y el contenido será el siguiente:

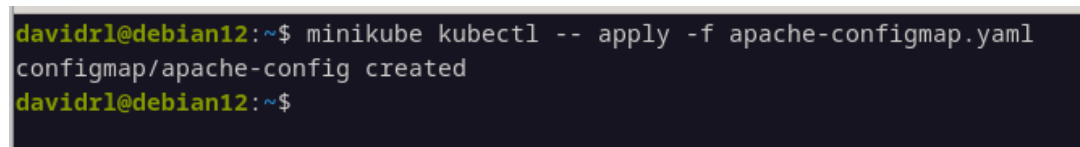
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: apache-config
data:
  index.html: |
    <!DOCTYPE html>
    <html lang="es">
    <head>
      <meta charset="UTF-8">
      <title>Minikube DavidRL</title>
    </head>
    <body>
      <h1>Este es el Apache en Kubernetes de DavidRL</h1>
    </body>
    </html>
```



```
GNU nano 7.2 apache-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: apache-config
data:
  index.html: |
    <!DOCTYPE html>
    <html lang="es">
    <head>
      <meta charset="UTF-8">
      <title>Minikube DavidRL</title>
    </head>
    <body>
      <h1>Este es el Apache en Kubernetes de DavidRL</h1>
    </body>
    </html>
```

Ahora aplico este configmap:

```
minikube kubectl -- apply -f apache-configmap.yaml
```



```
davidrl@debian12:~$ minikube kubectl -- apply -f apache-configmap.yaml
configmap/apache-config created
davidrl@debian12:~$
```

Una vez hecho esto, debo modificar mi archivo apache-deploy para poder montar el configmap sobre él, solamente tengo que añadir las siguientes entradas a continuación de lo que ya tenía:

volumeMounts:

- *name: apache-html*
 mountPath: /var/www/html

volumes:

- *name: apache-html*
- configMap:*
- *name: apache-config*

quedando así:

```
GNU nano 7.2
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: webserver
    name: webserver
spec:
  replicas: 3
  selector:
    matchLabels:
      app: webserver
      name: webserver
  template:
    metadata:
      labels:
        app: webserver
        name: webserver
    spec:
      containers:
        - name: apache
          image: fedora/apache
          ports:
            - containerPort: 80
          volumeMounts:
            - name: apache-html
              mountPath: /var/www/html
      volumes:
        - name: apache-html
          configMap:
            name: apache-config
```


Ahora volvemos a lanzar el deployment:

```
davidrl@davidrl@debian12:~$ minikube kubectl -- apply -f apache-deploy.yaml
deployment.apps/apache-deployment configured
davidrl@davidrl@debian12:~$
```

Y relanzamos el servicio:

```
davidrl@davidrl@debian12:~$ minikube kubectl -- apply -f apache-service.yaml
service/apache-service unchanged
```

Volvemos a acceder al servicio publicado y vemos que se han efectuado los cambios:



The screenshot shows a web browser window with the title 'Minikube DavidRL' and the address bar displaying '192.168.49.2:30080'. The main content of the browser is a large heading: **Este es el Apache en Kubernetes de DavidRL**.

Below the browser window, a terminal window is open, showing the command `minikube service apache-service` and its output. The output is a table with the following data:

NAMESPACE	NAME	TARGET PORT	URL
default	apache-service	80	http://192.168.49.2:30080

Below the table, the terminal shows the message: `Opening service default/apache-service in default browser...`

Y esto sería a grandes rasgos una demostración muy básica de lo que es kubernetes.

Yo mismo no entendía muy bien de que se trataba hasta que no me hice la siguiente pregunta:

-Por qué algo tan sencillo como cambiar el index.html de apache es tan aparatoso?

Pues bien, en la respuesta a esta pregunta también radica parte del significado de kubernetes, es por la escalabilidad, kubernetes no pretende ser amigable con el desarrollo local, si no que mas bien esta pensado para un despliegue a grande escala, piensa en este laboratorio, en este caso al ser solo un nodo, es mucho más fácil instalarte un apache y configurarlo, pero imagínate que tuvieras que servir esta misma pagina que he mostrado en 5000 nodos simultáneamente, pues en ese caso es mucho más cómodo cargar los yaml que configurar los apache uno a uno.