

## Actividad Nº 1 - MÓDULO 2

Conforme a la documentación del curso y a los enlaces a la documentación oficial del proyecto, junto con el sistema concreto que se vaya a utilizar para realizar el curso; instala y configura minikube y kubectl para poder realizar el resto del curso.

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

1. Pantallazo con la salida de minikube status (pantallazo1.jpg).
2. Pantallazo con la salida de kubectl get nodes -o wide (pantallazo2.jpg).

```
artacho@artacho-VirtualBox:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

```
artacho@artacho-VirtualBox:~$ kubectl get nodes -o wide
NAME                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION
CONTAINER-RUNTIME
minikube            Ready    control-plane,master   40m   v1.23.1   192.168.49.2   <none>        Ubuntu 20.04.2 LTS   5.4.0-42-generic
docker://20.10.12
```

## Actividad Nº1 - MÓDULO 3

Vamos a crear nuestro primer Pod, y para ellos vamos a desplegar una imagen que nos ofrece un servidor web con una página estática. Para ello realiza los siguientes pasos:

1. Crea un fichero yaml con la descripción del recurso Pod, teniendo en cuenta los siguientes aspectos:
  - Indica nombres distintos para el Pod y para el contenedor.
  - La imagen que debes desplegar es iesgn/test\_web:latest.
  - Indica una etiqueta en la descripción del Pod.
2. Crea el Pod.
3. Comprueba que el Pod se ha creado y está corriendo.
4. Obtén información detallada del Pod creado.
5. Accede de forma interactiva al Pod y comprueba los ficheros que están en el DocumentRoot (usr/local/apache2/htdocs/).
6. Crea una redirección con kubectl port-forward utilizando el puerto de localhost 8888 y sabiendo que el Pod ofrece el servicio en el puerto 80. Accede a la aplicación desde un navegador.
7. Muestra los logs del Pod y comprueba que se visualizan los logs de los accesos que hemos realizado en el punto anterior.
8. Elimina el Pod, y comprueba que ha sido eliminado.

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

1. Pantallazo del fichero yaml que has creado con la definición del Pod (pantallazo1.jpg).

```
GNU nano 2.9.3

apiVersion: v1 # required
kind: Pod # required
metadata: # required
  name: pod # required
  labels:
    service: web
spec: # required
  containers:
    - image: iesgn/test_web:latest
      name: web_test
      imagePullPolicy: Always
```

2. Pantallazo donde se comprueba que el Pod ha sido creado (pantallazo2.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl run pod --image=iesgn/test_web:latest
pod/pod created
artacho@artacho-VirtualBox:~$
```

3. Pantallazo donde se ve la información detallada del Pod (pantallazo3.jpg).

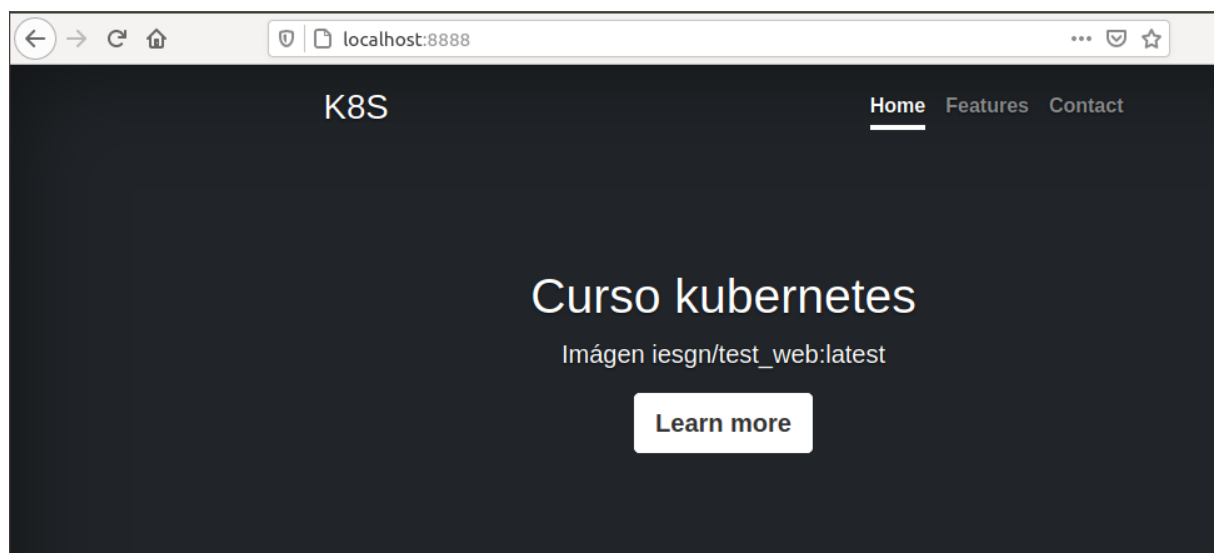
```
artacho@artacho-VirtualBox:~$ kubectl describe pod pod
Name: pod
Namespace: default
Priority: 0
Node: minikube/192.168.49.2
Start Time: Mon, 21 Feb 2022 09:56:02 +0100
Labels: run=pod
Annotations: <none>
Status: Running
IP: 172.17.0.3
IPs:
  IP: 172.17.0.3
Containers:
  pod:
    Container ID: docker://d0e9a9733bc548f7f0601d10c8bd1e5e1f50199afa08e5c6ac615c76a7e4676f
    Image: iesgn/test_web:latest
    Image ID: docker-pullable://iesgn/test_web@sha256:001e1f4d8ab5d7ddf406e481392052769
    Port: <none>
    Host Port: <none>
    State: Running
      Started: Mon, 21 Feb 2022 09:56:16 +0100
    Ready: True
    Restart Count: 0
    Environment: <none>
```

4. Pantallazo donde se ve el fichero index.html del DocumentRoot (pantallazo4.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl exec -it pod -- ls /usr/local/apache2/htdocs
index.html
artacho@artacho-VirtualBox:~$ kubectl exec -it pod -- cat /usr/local/apache2/htdocs/index.html
<!doctype html>
<html lang="en" class="h-100">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">
    <meta name="generator" content="Hugo 0.80.0">
    <title>Curso K8S</title>

    <link rel="canonical" href="https://getbootstrap.com/docs/5.0/examples/cover/">
```

5. Pantallazo del navegador accediendo a la aplicación con el port-forward (pantallazo5.jpg).



6. Pantallazo donde se ve los logs de acceso del Pod (pantallazo6.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl logs pod
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName'
directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName'
directive globally to suppress this message
[Mon Feb 21 08:56:16.663449 2022] [mpm_event:notice] [pid 1:tid 140629018727552] AH00489: Apache/2.4.46 (Unix) configured --
resuming normal operations
[Mon Feb 21 08:56:16.665338 2022] [core:notice] [pid 1:tid 140629018727552] AH00094: Command line: 'httpd -D FOREGROUND'
127.0.0.1 - - [21/Feb/2022:09:03:22 +0000] "GET / HTTP/1.1" 200 2884
127.0.0.1 - - [21/Feb/2022:09:03:22 +0000] "GET /favicon.ico HTTP/1.1" 404 196
127.0.0.1 - - [21/Feb/2022:09:03:38 +0000] "GET / HTTP/1.1" 200 2884
127.0.0.1 - - [21/Feb/2022:09:03:38 +0000] "GET /favicon.ico HTTP/1.1" 404 196
artacho@artacho-VirtualBox:~$
```

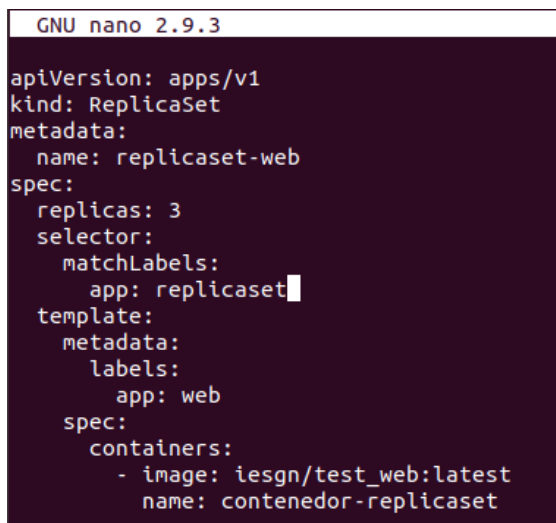
## Actividad Nº1 - MÓDULO 4

Como indicamos en el contenido de este módulo, no se va a trabajar directamente con los Pods (realmente tampoco vamos a trabajar directamente con los ReplicaSet, en el siguiente módulo explicaremos los Deployments que serán el recurso con el que trabajaremos). En este ejercicio vamos a crear un ReplicaSet que va a controlar un conjunto de Pods. Para ello, realiza los siguientes pasos:

1. Crea un fichero yaml con la descripción del recurso ReplicaSet, teniendo en cuenta los siguientes aspectos:
  - Indica nombres distintos para el ReplicaSet y para el contenedor de los Pods que va a controlar.
  - El ReplicaSet va a crear 3 réplicas.
  - La imagen que debes desplegar es `iesgn/test_web:latest`.
  - Indica de manera adecuada una etiqueta en la especificación del Pod que vas a definir que coincida con el *selector* del ReplicaSet.
2. Crea el ReplicaSet.
3. Comprueba que se ha creado el ReplicaSet y los 3 Pods.
4. Obtén información detallada del ReplicaSet creado.
5. Vamos a probar la tolerancia a fallos: Elimina uno de los 3 Pods, y comprueba que inmediatamente se ha vuelto a crear un nuevo Pod.
6. Vamos a comprobar la escalabilidad: escala el ReplicaSet para tener 6 Pods de la aplicación.
7. Elimina el ReplicaSet y comprueba que se han borrado todos los Pods.

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

1. Pantallazo del fichero yaml que has creado con la definición del ReplicaSet (pantallazo1.jpg).



```
GNU nano 2.9.3
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: replicaset-web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: replicaset
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - image: iesgn/test_web:latest
          name: contenedor-replicaset
```

2. Pantallazo donde se comprueba que el ReplicaSet y los 3 Pods se han creado (pantallazo2.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl create -f replicaset.yaml
replicaset.apps/replicaset-web created
```

3. Pantallazo donde se ve la información detallada del ReplicaSet (pantallazo3.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl describe rs replicaset
Name:      replicaset-web
Namespace: default
Selector:  app=replicaset
Labels:    <none>
Annotations: <none>
Replicas:  3 current / 3 desired
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=replicaset
  Containers:
    contenedor-replicaset:
      Image:      iesgn/test_web:latest
      Port:       <none>
      Host Port:  <none>
      Environment: <none>
      Mounts:     <none>
      Volumes:    <none>
Events:
  Type      Reason            Age   From                      Message
  ----      -
  Normal    SuccessfulCreate   3m10s replicaset-controller     Created pod: replicaset-web-rb5fm
  Normal    SuccessfulCreate   3m10s replicaset-controller     Created pod: replicaset-web-rmkg5
  Normal    SuccessfulCreate   3m10s replicaset-controller     Created pod: replicaset-web-wwwvl
```

4. Pantallazo donde se ven los Pods que se han creado, después de eliminar uno de ellos (pantallazo4.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl delete pod replicaset-web-75cld
pod "replicaset-web-75cld" deleted
artacho@artacho-VirtualBox:~$ kubectl get pods -o wide
NAME                                READY    STATUS    RESTARTS   AGE    IP            NODE     NOMINATED NODE   READINESS GATES
pod                                 1/1      Running   0          27m    172.17.0.3    minikube <none>          <none>
replicaset-web-cgvdm               1/1      Running   0          6s     172.17.0.4    minikube <none>          <none>
replicaset-web-f8n6m               1/1      Running   0          67s    172.17.0.7    minikube <none>          <none>
replicaset-web-rb5fm               1/1      Running   0          6m23s  172.17.0.5    minikube <none>          <none>
```

5. Pantallazo donde se ven los Pods que se han creado después del escalado (pantallazo5.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl scale rs replicaset-web --replicas=6
replicaset.apps/replicaset-web scaled
artacho@artacho-VirtualBox:~$ kubectl get pods
NAME                                READY    STATUS    RESTARTS   AGE
pod                                 1/1      Running   0          30m
replicaset-web-bgm5m               1/1      Running   0          13s
replicaset-web-cgvdm               1/1      Running   0          2m33s
replicaset-web-f8n6m               1/1      Running   0          3m34s
replicaset-web-ppm8k               1/1      Running   0          13s
replicaset-web-rb5fm               1/1      Running   0          8m50s
replicaset-web-zmsx8               1/1      Running   0          13s
```

## Actividad Nº1 - MÓDULO 5

En esta actividad vamos a crear un Deployment de una aplicación web. Sigamos los siguientes pasos:

1. Crea un fichero yaml con la descripción del recurso Deployment, teniendo en cuenta los siguientes aspectos:
  - Indica nombres distintos para el Deployment y para el contenedor de los Pods que va a controlar.
  - El Deployment va a crear 2 réplicas.
  - La imagen que debes desplegar es `iesgn/test_web:latest`.
  - Indica de manera adecuada una etiqueta en la especificación del Pod que vas a definir que coincida con el *selector* del Deployment.
2. Crea el Deployment.
3. Comprueba los recursos que se han creado: Deployment, ReplicaSet y Pods.
4. Obtén información detallada del Deployment creado.
5. Crea una redirección utilizando el port-forward para acceder a la aplicación, sabiendo que la aplicación ofrece el servicio en el puerto 80, y accede a la aplicación con un navegador web.
6. Accede a los logs del despliegue para comprobar el acceso que has hecho en el punto anterior.
7. Elimina el Deployment y comprueba que se han borrado todos los recursos creados.

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

1. Pantallazo del fichero yaml que has creado con la definición del Deployment (pantallazo1.jpg).



```
GNU nano 2.9.3
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment
  labels:
    app: deployment
spec:
  revisionHistoryLimit: 2
  strategy:
    type: RollingUpdate
  replicas: 2
  selector:
    matchLabels:
      app: deployment
  template:
    metadata:
      labels:
        app: deployment
    spec:
      containers:
      - image: iesgn/test_web:latest
        name: contendor-web
        ports:
        - name: http
          containerPort: 80
```

2. Pantallazo donde se comprueba los recursos que se han creado (pantallazo2.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl apply -f deployment.yaml
deployment.apps/deployment created
artacho@artacho-VirtualBox:~$ kubectl get deploy,rs,pod
```

| NAME                       | READY | UP-TO-DATE | AVAILABLE | AGE |
|----------------------------|-------|------------|-----------|-----|
| deployment.apps/deployment | 2/2   | 2          | 2         | 11s |

| NAME                                 | DESIRED | CURRENT | READY | AGE |
|--------------------------------------|---------|---------|-------|-----|
| replicaset.apps/deployment-88b549bcd | 2       | 2       | 2     | 11s |

| NAME                           | READY | STATUS  | RESTARTS | AGE |
|--------------------------------|-------|---------|----------|-----|
| pod/deployment-88b549bcd-tww6m | 1/1   | Running | 0        | 11s |
| pod/deployment-88b549bcd-z9rgx | 1/1   | Running | 0        | 11s |

```
artacho@artacho-VirtualBox:~$
```

3. Pantallazo donde se ve la información detallada del Deployment (pantallazo3.jpg).

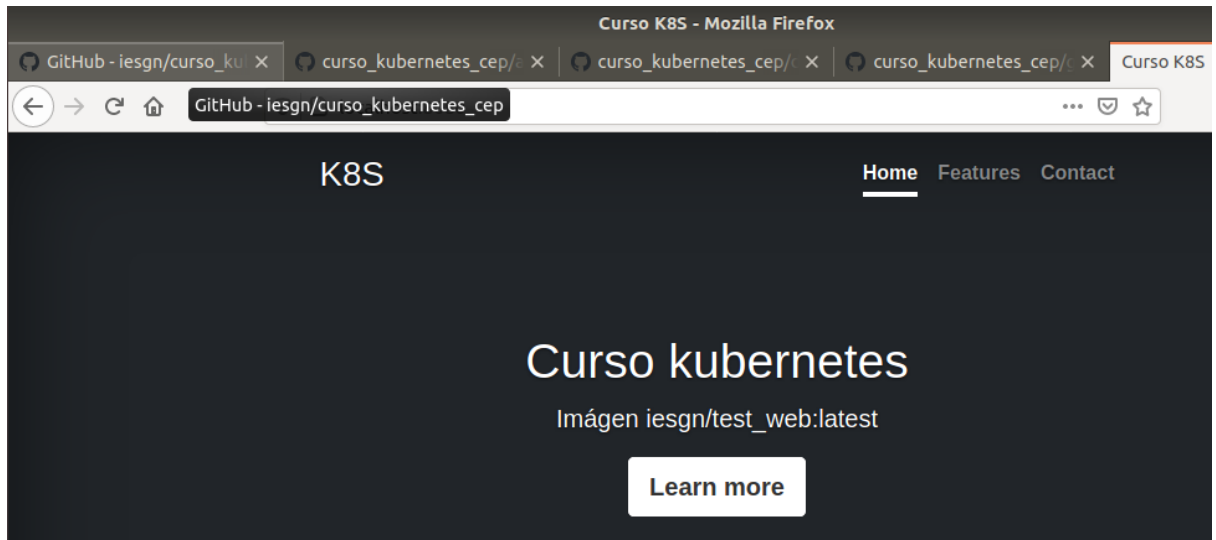
```
artacho@artacho-VirtualBox:~$ kubectl describe deployment
```

```
Name: deployment
Namespace: default
CreationTimestamp: Mon, 21 Feb 2022 12:57:50 +0100
Labels: app=deployment
Annotations: deployment.kubernetes.io/revision: 1
Selector: app=deployment
Replicas: 2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType: RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels: app=deployment
  Containers:
    contendor-web:
      Image: iesgn/test_web:latest
      Port: 80/TCP
      Host Port: 0/TCP
      Environment: <none>
      Mounts: <none>
      Volumes: <none>
Conditions:
  Type           Status  Reason
  ----           -
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets: <none>
```

4. Pantallazo donde se vea el acceso desde un navegador web a la aplicación usando el port-forward (pantallazo4.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl port-forward deployment/deployment 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```





5. Pantallazo donde se vea los logs del despliegue después del acceso (pantallazo5.jpg).

```
artacho@artacho-VirtualBox:~$ kubectl logs deployment/deployment
Found 2 pods, using pod/deployment-88b549bcd-tww6m
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName'
directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName'
directive globally to suppress this message
[Mon Feb 21 11:57:54.327280 2022] [mpm_event:notice] [pid 1:tid 139730528846976] AH00489: Apache/2.4.46 (Unix) configured --
resuming normal operations
[Mon Feb 21 11:57:54.335605 2022] [core:notice] [pid 1:tid 139730528846976] AH00094: Command line: 'httpd -D FOREGROUND'
127.0.0.1 - - [21/Feb/2022:12:04:05 +0000] "GET / HTTP/1.1" 200 2884
127.0.0.1 - - [21/Feb/2022:12:04:05 +0000] "GET /favicon.ico HTTP/1.1" 404 196
artacho@artacho-VirtualBox:~$
```

## Actividad N°3 - MÓDULO 5

En esta tarea vamos a desplegar una aplicación web que requiere de dos servicios para su ejecución. La aplicación se llama GuestBook y necesita los siguientes servicios:

- La aplicación Guestbook es una aplicación web desarrollada en python que es servida en el puerto 5000/tcp. Utilizaremos la imagen iesgn/guestbook.
- Esta aplicación guarda la información en una base de datos no relacional redis, que utiliza el puerto 6379/tcp para recibir las conexiones. Usaremos la imagen redis.

Por lo tanto si tenemos dos servicios distintos, tendremos dos ficheros yaml para crear dos recursos Deployment, uno para cada servicio. Con esta manera de trabajar podemos obtener las siguientes características:

1. Cada conjunto de Pods creado en cada despliegue ejecutarán un solo proceso para ofrecer el servicio.
2. Cada conjunto de Pods se puede escalar de manera independiente. Esto es importante, si identificamos que al acceder a alguno de los servicios se crea un cuello de botella, podemos escalarlo para tener más Pods ejecutando el servicio.
3. Las actualizaciones de los distintos servicios no interfieren en el resto.



- Lo estudiaremos en un módulo posterior, pero podremos gestionar el almacenamiento de cada servicio de forma independiente.

Por lo tanto para desplegar la aplicaciones tendremos dos ficheros.yaml:

- [questbook-deployment.yaml](#)
- [redis-deployment.yaml](#)

Para realizar el despliegue realiza los siguientes pasos:

- Usando los ficheros anteriores crea los dos Deployments.
- Comprueba que los recursos que se han creado: Deployment, ReplicaSet y Pods.
- Crea una redirección utilizando el port-forward para acceder a la aplicación, sabiendo que la aplicación ofrece el servicio en el puerto 5000, y accede a la aplicación con un navegador web.

¿Qué aparece en la página principal de la aplicación?. Aparece el siguiente mensaje: Waiting for database connection.... Por lo tanto podemos indicar varias conclusiones:

- Hasta ahora no estamos accediendo de forma "normal" a las aplicaciones. El uso de la opción port-forward es un mecanismo que realmente nos posibilita acceder a la aplicación, pero utilizando un proxy. Deberíamos acceder a las aplicaciones usando una ip y un puerto determinado.
- Parece que tampoco hay acceso entre los Pods de los distintos despliegues. Parece que los Pods de la aplicación questbook no pueden acceder al Pod donde se está ejecutando la base de datos redis.

En el siguiente módulo estudiaremos los recursos que nos ofrece la API de Kubernetes para permitirnos el acceso a las aplicaciones desde el exterior, y para que los distintos Pods de los despliegues puedan acceder entre ellos.

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

- Pantallazo donde se comprueba los recursos que se han creado (pantallazo1.jpg).

```
GNU nano 2.9.3
apiVersion: apps/v1
kind: Deployment
metadata:
  name: questbook
  labels:
    app: questbook
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: questbook
      tier: frontend
  template:
    metadata:
      labels:
        app: questbook
        tier: frontend
    spec:
      containers:
        - name: contenedor-questbook
          image: tesgn/questbook
          ports:
            - name: http-server
              containerPort: 5000
```

```
GNU nano 2.9.3
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
  labels:
    app: redis
    tier: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
      tier: backend
  template:
    metadata:
      labels:
        app: redis
        tier: backend
    spec:
      containers:
        - name: contenedor-redis
          image: redis
          ports:
            - name: redis-server
              containerPort: 6379
```

```

artacho@artacho-VirtualBox:~$ kubectl apply -f guestbook-deployment.yaml
deployment.apps/guestbook created
artacho@artacho-VirtualBox:~$ kubectl apply -f redis-deployment.yaml
deployment.apps/redis created
artacho@artacho-VirtualBox:~$ kubectl get deploy,rs,pods
NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/deployment             2/2     2            2           19m
deployment.apps/guestbook              0/3     3            0           30s
deployment.apps/redis                  0/1     1            0           26s

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/deployment-88b549bcd    2         2         2       19m
replicaset.apps/guestbook-7cfcc5ff8d    3         3         0       30s
replicaset.apps/redis-5d96fc576         1         1         0       26s

NAME                                     READY   STATUS              RESTARTS   AGE
pod/deployment-88b549bcd-tww6m          1/1     Running             0          19m
pod/deployment-88b549bcd-z9rgx          1/1     Running             0          19m
pod/guestbook-7cfcc5ff8d-bpfzg          0/1     ContainerCreating   0          30s
pod/guestbook-7cfcc5ff8d-lrhvp          0/1     ContainerCreating   0          30s
pod/guestbook-7cfcc5ff8d-xlg5z          0/1     ContainerCreating   0          30s
pod/redis-5d96fc576-78gpz               0/1     ContainerCreating   0          26s
artacho@artacho-VirtualBox:~$

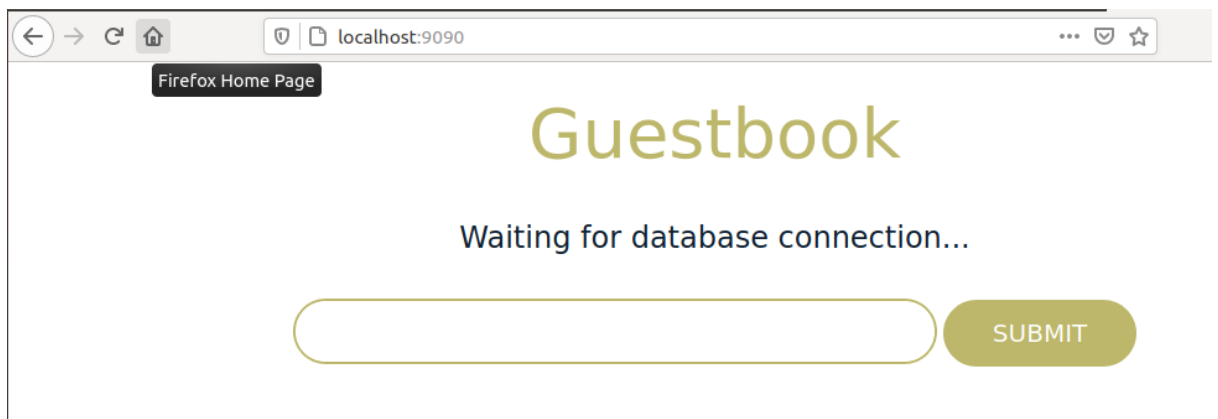
```

2. Pantallazo donde se vea el acceso desde un navegador web a la aplicación usando el port-forward, y se vea el mensaje de error al no poder acceder a la base de datos (pantallazo2.jpg).

```

artacho@artacho-VirtualBox:~$ kubectl port-forward deployment/guestbook 9090:5000
0
Forwarding from 127.0.0.1:9090 -> 5000
Forwarding from [::1]:9090 -> 5000
Handling connection for 9090
Handling connection for 9090
Handling connection for 9090

```



## Actividad Nº1 - MÓDULO 6

Realiza los siguientes pasos:

1. Activa el *addon* ingress en minikube para instalar el Ingress Controller.
2. Crea La definición del recurso Ingress con los datos sugeridos, y crea el recurso Ingress.
3. Modifica el fichero `/etc/hosts` de tu ordenador para configurar la resolución estática.
4. Accede a la aplicación usando el nombre que has asignado.

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

1. Pantallazo donde se vea el acceso desde un navegador web a la aplicación cuando sólo tenemos el servicio para acceder a la aplicación (tiene que aparecer el mensaje de error) (pantallazo1.jpg).

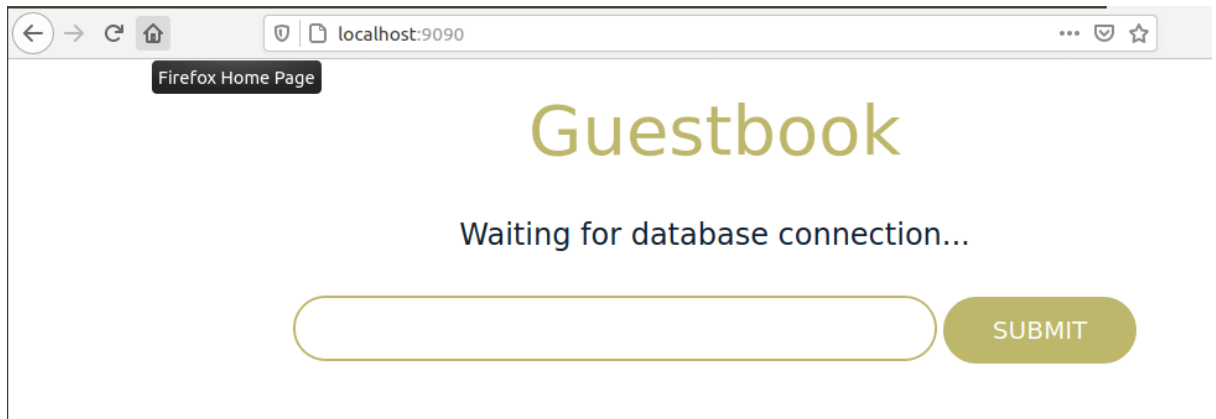
```
artacho@artacho-VirtualBox:~$ kubectl describe Service/guestbook
Name:                guestbook
Namespace:            default
Labels:              app=guestbook
                    tier=frontend
Annotations:          <none>
Selector:             app=guestbook,tier=frontend
Type:                NodePort
IP Family Policy:    SingleStack
IP Families:         IPv4
IP:                  10.96.47.99
IPs:                 10.96.47.99
Port:                <unset> 80/TCP
TargetPort:          http-server/TCP
NodePort:            <unset> 31869/TCP
Endpoints:           172.17.0.5:5000,172.17.0.6:5000,172.17.0.7:5000
Session Affinity:    None
External Traffic Policy: Cluster
Events:              <none>
```

Vemos que el puerto asignado es el 31869. Y que nuestra ip de minikube es

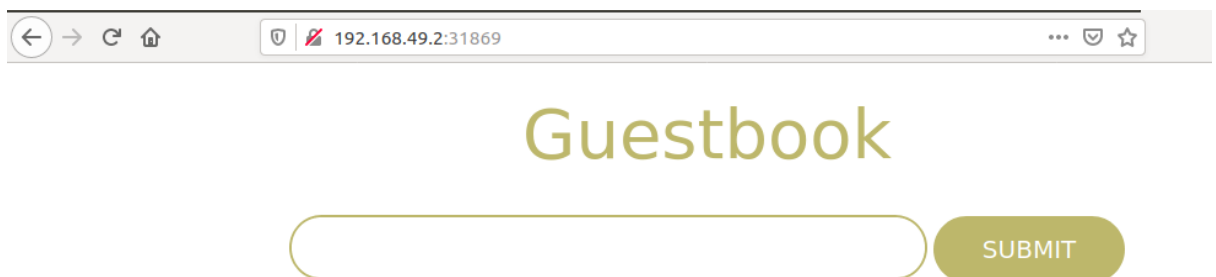
```
artacho@artacho-VirtualBox:~$ minikube ip
192.168.49.2
```

```
artacho@artacho-VirtualBox:~$ kubectl get services
NAME          TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
guestbook     NodePort      10.96.47.99    <none>         80:31869/TCP     17m
kubernetes    ClusterIP     10.96.0.1      <none>         443/TCP          73m
redis         NodePort      10.97.124.234  <none>         6379:31177/TCP   9m56s
```

Accedemos en el equipo local y debe salir “esperando conexión con la base de datos”.



2. Pantallazo donde se vea el acceso desde un navegador web a la aplicación usando la ip del nodo master y el puerto asignado al Service (pantallazo2.jpg). Ahora que hemos configurado los dos servicios nos permite el acceso.



3. Pantallazo donde se vea el acceso desde un navegador web a la aplicación usando el nombre que hemos configurado en el recurso Ingress (pantallazo3.jpg).

Para hacer esto hemos instalado los addons de minikube. Luego hay que acceder a /etc/hosts en modo root y hay que introducir la ip de minikube y la dirección de host del ingress.yaml.

```
GNU nano 2.9.3
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: guestbook
spec:
  rules:
  - host: www.artacho.org
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: guestbook
            port:
              number: 80
```

```
artacho@artacho-VirtualBox:~$ nano ingress.yaml
artacho@artacho-VirtualBox:~$ minikube addons enable ingress
  ■ Using image k8s.gcr.io/ingress-nginx/controller:v1.1.0
  ■ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  ■ Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  ☀ Verifying ingress addon...
  ☀ The 'ingress' addon is enabled
artacho@artacho-VirtualBox:~$ kubectl get pods -n ingress-guestbook
No resources found in ingress-guestbook namespace.
artacho@artacho-VirtualBox:~$ kubectl get pods -n guestbook
No resources found in guestbook namespace.
artacho@artacho-VirtualBox:~$ kubectl get pods -n ingress-nginx
NAME                                READY   STATUS    RESTARTS   AGE
ingress-nginx-admission-create-bdkxm 0/1     Completed 0           98s
ingress-nginx-admission-patch-jmw6k  0/1     Completed 0           98s
ingress-nginx-controller-6d5f55986b-wdn65 1/1     Running   0           98s
artacho@artacho-VirtualBox:~$
```

```
GNU nano 2.9.3 /etc/hosts
127.0.0.1    localhost
127.0.1.1    artacho-VirtualBox
192.168.49.2 www.artacho.org
# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

← → ↻ 🏠

🔒 <https://www.artacho.org>

⋮ 📄 ☆

Guestbook

SUBMIT

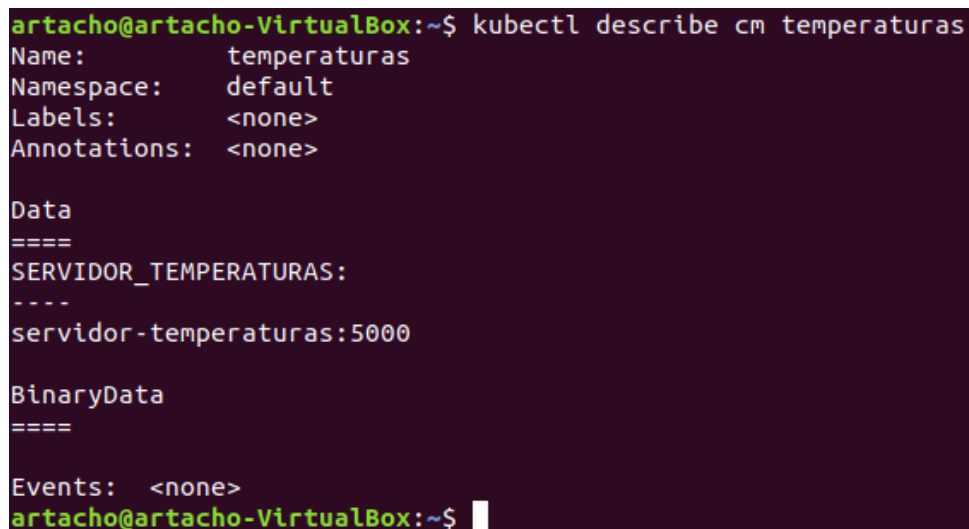
## Actividad Nº1 - MÓDULO 7

Vamos a modificar esta variable en el despliegue del frontend y cambiaremos el nombre del Service del backend para que coincidan, para ello realiza los siguientes pasos:

1. Crea un recurso ConfigMap con un dato que tenga como clave `SERVIDOR_TEMPERATURAS` y como contenido `servidor-temperaturas:5000`.
2. Modifica el fichero de despliegue del frontend: [frontend-deployment.yaml](#) para añadir la modificación de la variable `TEMP_SERVER` con el valor que hemos guardado en el ConfigMap.
3. Realiza el despliegue y crea el Service para acceder al frontend.
4. Despliega el microservicio backend.
5. Modifica el fichero [backend-srv.yaml](#) para cambiar el nombre del Service por `servidor-temperaturas` y crea el Service.
6. Accede a la aplicación usando el puerto asignado al Service NodePort del frontend o creando el recurso Ingress.

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

1. Pantallazo donde se vea la definición del recurso ConfigMap (pantallazo1.jpg).



```
artacho@artacho-VirtualBox:~$ kubectl describe cm temperaturas
Name:         temperaturas
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
SERVIDOR_TEMPERATURAS:
----
servidor-temperaturas:5000

BinaryData
=====

Events:      <none>
artacho@artacho-VirtualBox:~$
```

2. Pantallazo donde se vea la modificación del fichero `frontend-deployment.yaml` (pantallazo2.jpg).

```
GNU nano 2.9.3 frontend-deployment-temperaturas.yaml

template:
  metadata:
    labels:
      app: temperaturas
      tier: frontend
  spec:
    containers:
      - name: contenedor-temperaturas
        image: iesgn/temperaturas_frontend
        ports:
          - name: http-server
            containerPort: 3000
        env:
          - name: SERVIDOR_TEMPERATURAS
            valueFrom:
              configMapKeyRef:
                name: temperaturas
                key: servidor-temperaturas:5000
```

3. Pantallazo donde se vea la modificación del fichero backend-srv.yaml (pantallazo3.jpg).

```
GNU nano 2.9.3 backend-srv-temperaturas.yaml

apiVersion: v1
kind: Service
metadata:
  name: servidor-temperaturas
  labels:
    app: temperaturas
    tier: backend
spec:
  type: ClusterIP
  ports:
    - port: 5000
      targetPort: api-server
  selector:
    app: temperaturas
    tier: backend
```

4. Pantallazo donde se compruebe que la aplicación está funcionando (pantallazo4.jpg).





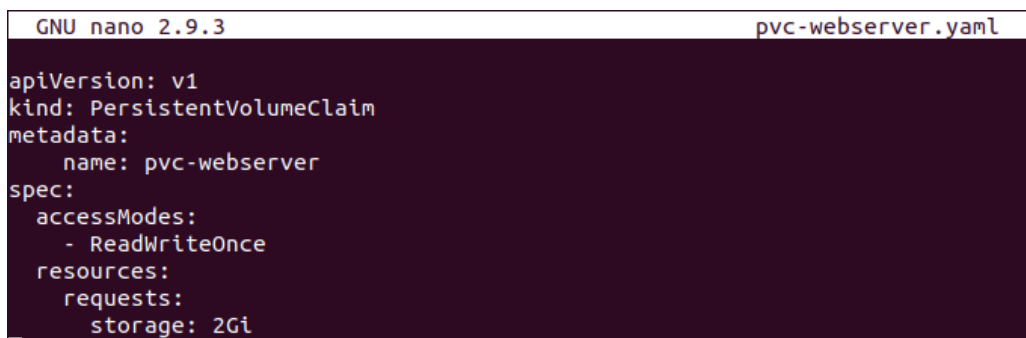
## Actividad Nº1 - MÓDULO 8

### Realiza los siguientes pasos:

1. Crea un fichero yaml para definir un recurso PersistentVolumeClaim que se llame pvc-webserver y para solicitar un volumen de 2Gb.
2. Crea el recurso y comprueba que se ha asociado un volumen de forma dinámica a la solicitud.
3. Crea un fichero yaml para desplegar un servidor web desde la imagen php:7.4-apache, asocia el volumen al Pod que se va a crear e indica el punto de montaje en el *DocumentRoot* del servidor: */var/www/html*.
4. Despliega el servidor y crea un fichero info.php en */var/www/html*, con el siguiente contenido: `<?php phpinfo(); ?>`.
5. Define y crea un Service NodePort, accede desde un navegador al fichero info.php y comprueba que se visualiza de forma correcta.
6. Comprobemos la persistencia: elimina el Deployment, vuelve a crearlo y vuelve a acceder desde el navegador al fichero info.php. ¿Se sigue visualizando?

Para superar la actividad deberás entregar en un fichero comprimido los siguientes pantallazos:

1. Pantallazo con la definición del recurso PersistentVolumeClaim (pantallazo1.jpg).



```
GNU nano 2.9.3 pvc-webserver.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-webserver
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

2. Pantallazo donde se visualice los recursos pv y pvc que se han creado (pantallazo2.jpg).



```
artacho@artacho-VirtualBox:~$ kubectl apply -f pvc-webserver.yaml
persistentvolumeclaim/pvc-webserver created
artacho@artacho-VirtualBox:~$ kubectl get pv, pvc
error: arguments in resource/name form must have a single resource and name
artacho@artacho-VirtualBox:~$ kubectl get pv,pvc
```

| NAME  | STORAGECLASS | REASON | AGE | CAPACITY | ACCESS MODES | RECLAIM POLICY | STATUS | CLAIM         |
|---|--------------|--------|-----|----------|--------------|----------------|--------|---------------|
| persistentvolume/pvc-1f8e9e33-9c8d-4c77-b581-569ae3ea79f2 | standard     |        | 21s | 2Gi      | RWO          | Delete         | Bound  | default/pvc-w |

| NAME  | STATUS | VOLUME                                   | CAPACITY | ACCESS MODES | STORAGECL |
|---|--------|--|----------|--------------|-----------|
| ASS AGE persistentvolumeclaim/pvc-webserver | Bound  | pvc-1f8e9e33-9c8d-4c77-b581-569ae3ea79f2 | 2Gi      | RWO          | standard  |

```
artacho@artacho-VirtualBox:~$
```

3. Pantallazo donde se vea el fichero yaml para el despliegue (pantallazo3.jpg).

```

GNU nano 2.9.3                                deploy-webserver.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-webserver
  labels:
    app: apache
spec:
  replicas: 1
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      volumes:
        - name: apache-webserver
          persistentVolumeClaim:
            claimName: pvc-webserver
      containers:
        - name: contenedor-apache
          image: php:7.4-apache
          ports:
            - name: http-server

```


```

artacho@artacho-VirtualBox:~$ kubectl exec pod/apache-webserver-68579b955c-9mwd4 -- bash -c "echo '<?php phpinfo
(); ?>' > /var/www/html/index.php"

```

4. Pantallazo donde se vea el acceso a info.php (pantallazo4.jpg).

← → ↻
🔒 192.168.49.2:31438
☆

**PHP Version 7.4.28**


|  |   |
|--|---|
| <b>System</b>                            | Linux apache-webserver-68579b955c-9mwd4 5.4.0-100-generic #113~18.04.1-Ubuntu SMP Mon Feb 7 15:02:59 UTC 2022 x86_64  |
| <b>Build Date</b>                        | Feb 17 2022 18:29:15  |
| <b>Configure Command</b>                 | './configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-openssl' '--with-readline' '--with-zlib' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--with-apxs2' 'build_alias=x86_64-linux-gnu' |
| <b>Server API</b>                        | Apache 2.0 Handler  |
| <b>Virtual Directory Support</b>         | disabled  |
| <b>Configuration File (php.ini) Path</b> | /usr/local/etc/php  |

5. Pantallazo donde se vea que se ha eliminado y se ha vuelto a crear el despliegue y se sigue sirviendo el fichero info.php (pantallazo5.jpg).

```

artacho@artacho-VirtualBox:~$ kubectl delete deployment/apache-webserver
deployment.apps "apache-webserver" deleted
artacho@artacho-VirtualBox:~$ kubectl apply -f deploy-webserver.yaml
deployment.apps/apache-webserver created
artacho@artacho-VirtualBox:~$


```

← → ↻

🔒 192.168.49.2:31438

☆

PHP Version 7.4.28



|                                   |   |
|-----------------------------------|---|
| System                            | Linux apache-webserver-68579b955c-b2xdw 5.4.0-100-generic #113~18.04.1-Ubuntu SMP Mon Feb 7 15:02:59 UTC 2022 x86_64  |
| Build Date                        | Feb 17 2022 18:29:15  |
| Configure Command                 | './configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--with-pic' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-openssl' '--with-readline' '--with-zlib' '--with-pear' '--with-libdir=lib/x86_64-linux-gnu' '--disable-cgi' '--with-apxs2' 'build_alias=x86_64-linux-gnu' |
| Server API                        | Apache 2.0 Handler  |
| Virtual Directory Support         | disabled  |
| Configuration File (php.ini) Path | /usr/local/etc/php  |
| Loaded Configuration File         | (none)  |

Como podemos comprobar sigue dejándome entrar.