ADMINISTRACIÓN DE SISTEMAS 2

Proyecto 2

Pablo Moreno Muñoz 841972

Identificador W->E

ÍNDICE

Resumen

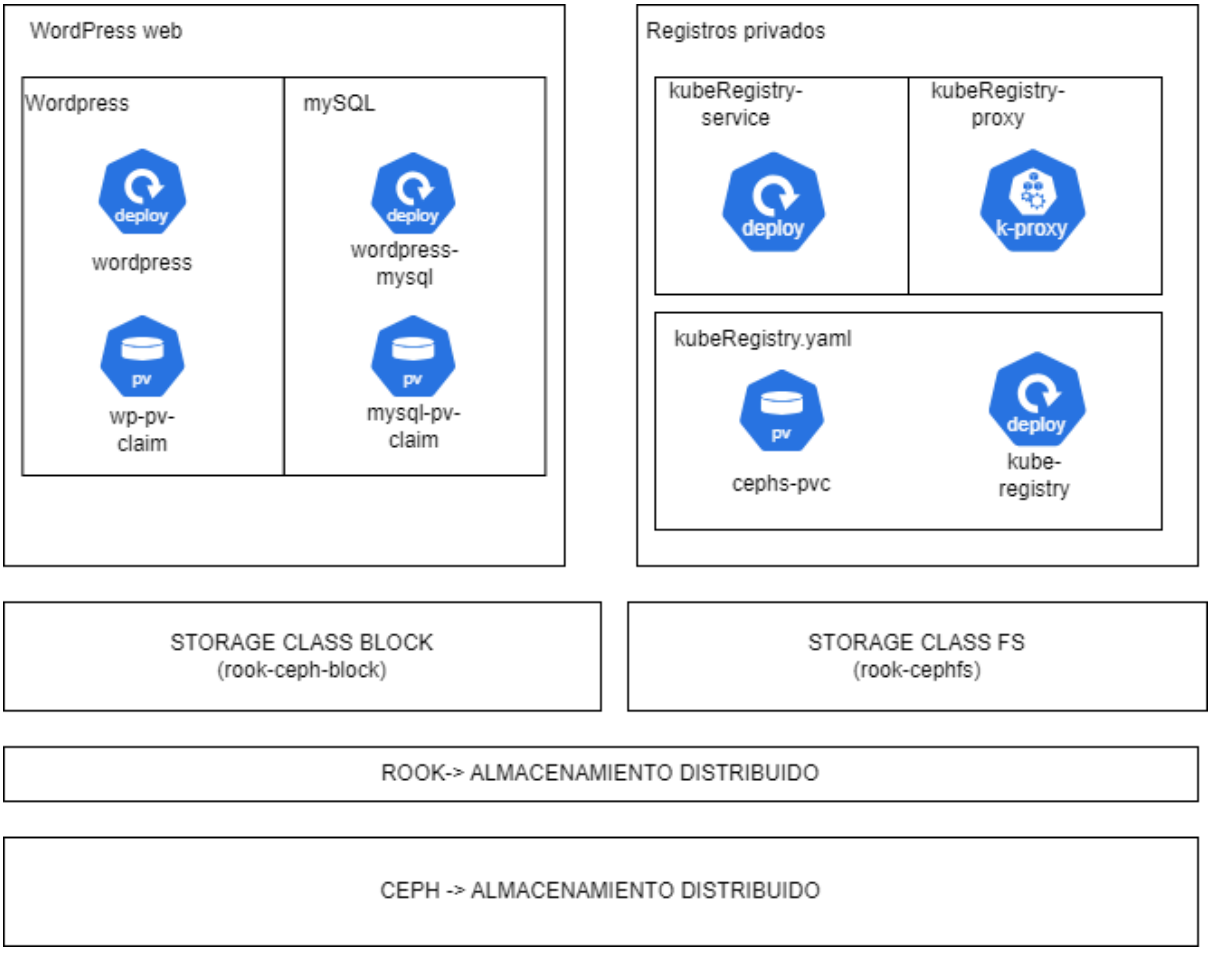
En esta segunda fase del proyecto, se plantea implementar una variedad de servicios en un clúster de Kubernetes utilizando el sistema de almacenamiento distribuido CEPH. Además se ha empleado ROOK, una herramienta que facilita la administración y gestión de servicios de almacenamiento en clústeres de Kubernetes.

El primer despliegue consistirá en una aplicación web basada en WordPress y MySQL, donde los datos van a ser gestionados utilizando el sistema de almacenamiento CEPH.

Por otro lado, como segundo despliegue, se propone la creación de un registro de contenedores privado replicado en Kubernetes, gestionando los datos y metadatos sobre un sistema de archivos distribuido, a su vez basado en el sistema de almacenamiento distribuido CEPH.

En resumen se van a realizar dos despliegues diferentes que harán uso del dispositivos de bloques y del sistemas de archivos distribuidos que ofrece el sistema de almacenamiento distribuido CEPH.

Arquitectura de elementos relevantes



Aplicaciones desplegadas

Etapa 1: Puesta en marcha básica de Ceph en Kubernetes

En esta etapa, implementamos una infraestructura básica de almacenamiento distribuido de alta disponibilidad basada en Ceph, mediante Rook, sobre un clúster Kubernetes con 3 nodos.

Explicación de los conceptos y recursos utilizados por Ceph:
- Ceph: Ceph es una solución de almacenamiento distribuido de alta disponibilidad. En este caso, se despliega sobre Kubernetes con la ayuda de Rook.
Los componentes distribuidos de Ceph, como los OSDs, son puestos en marcha para garantizar el funcionamiento del sistema de almacenamiento.

Verificación de funcionamiento:
- Comprobamos el estado de funcionamiento de Ceph a través del despliegue de un pod de herramientas Ceph (`rook-ceph-tools`).

Etapa 2: Aplicación web y almacenamiento distribuido de dispositivos bloques

En esta etapa, utilizamos dispositivos de bloques para crear una sencilla aplicación web que utilice volúmenes persistentes a partir del despliegue rook-ceph.

Conceptos y recursos Kubernetes utilizados:
- Almacenamiento de bloques: Montamos un almacén de datos a un solo Pod.
- Creamos un pool de bloques en Ceph (nivel de replicación 3) mediante un recurso Kubernetes personalizado `CephBlockPool` y un `StorageClass` asociado a este pool de bloques.

Explicación básica de los conceptos y recursos utilizados por Ceph:
- Dispositivos de bloques (RBD): Utilizamos dispositivos de bloques Ceph (RBD) para proporcionar volúmenes persistentes a la aplicación web.

Etapa 3: Registro de repositorio privado de contenedores y almacenamiento de sistema de ficheros distribuido

En esta etapa, ponemos en funcionamiento el registro de un repositorio privado de contenedores sin seguridad TLS.

Conceptos y recursos Kubernetes utilizados:
- Sistema de fichero distribuido Ceph: Creamos el sistema de fichero Ceph y el StorageClass asociado para que esté disponible para las aplicaciones ejecutadas en Kubernetes.
- Registro de repositorio privado de contenedores: Implementamos un registro de repositorio privado de contenedores utilizando el sistema de ficheros de Ceph.

Para este despliegue de registro de contenedores se han tenido que realizar más configuraciones ya que que se encuentra replicado en 3 réplicas.
Por lo tanto se necesita acceder a estos servicios desde el propio cluster, para que esto funcionara se han tenido que poner unos pods proxy en cada nodo para poder forwardear correctamente las peticiones

Verificación de funcionamiento:

Aplicación web y almacenamiento distribuido de dispositivos bloques:

- Comprobamos el funcionamiento de la aplicación web haciendo un port-forwarding de puerto 80 del servicio wordpress a nuestro puerto 8080 de localhost para corroborar el funcionamiento.
- Verificamos la utilización del nuevo pool de dispositivos de bloques de Ceph (RBD) mediante un pod de creación y uso manual de dispositivos de bloques a partir de ese pool.

Registro de repositorio privado de contenedores y almacenamiento de sistema de ficheros distribuido

- Comprobamos que el sistema de fichero está configurado y que los Pods MDS están en funcionamiento.
- Habilitamos, mediante un "port-forward", una conexión directa desde el host al registro privado de Kubernetes.
- Creamos y subimos a dicho registro un nuevo contenedor para verificar su funcionamiento.
- Comprobamos el acceso al sistema de ficheros Ceph mediante el montaje manual directo, accediendo al pod "rook-direct-mount" puesto en marcha en la etapa previa.

Problemas encontrados:

Un problema encontrado fue a la hora de realizar el vagrant up con las nuevas modificaciones de esta nueva práctica en concreto el rsync del directorio. El problema fue que no me dejaba realizar ninguna función de kubectl ya que me daba problemas del certificado firmado por una entidad desconocida la solución fue comentar la línea nodeconfig.vm.synced_folder ".", "/vagrant", type: "rsync"

Problema que me supuso una gran cantidad de pérdida de tiempo fue que a la hora de lanzar el common.yalm me el siguiente error :
El problema estaba en las versiones del kubectl y del k3s

```
Unset
kubectl create -f common.yaml namespace/rook-ceph created
serviceaccount/rook-ceph-system created serviceaccount/rook-ceph-osd created
serviceaccount/rook-ceph-mgr created serviceaccount/rook-ceph-cmd-reporter
created Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+,
unavailable in v1.25+ podsecuritypolicy.policy/rook-privileged created
clusterrole.rbac.authorization.k8s.io/psp:rook created
clusterrolebinding.rbac.authorization.k8s.io/rook-ceph-system-psp created
rolebinding.rbac.authorization.k8s.io/rook-ceph-default-psp created
rolebinding.rbac.authorization.k8s.io/rook-ceph-osd-psp created
rolebinding.rbac.authorization.k8s.io/rook-ceph-mgr-psp created
rolebinding.rbac.authorization.k8s.io/rook-ceph-cmd-repo
```

Anexo I

Cluster Ceph

kubectl create -f common.yaml

kubectl create -f operator.yaml
kubectl -n rook-ceph get pod

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ kubectl -n rook-ceph get pod
NAME                                READY   STATUS    RESTARTS   AGE
rook-ceph-operator-6d5456dddd-qvzrd   1/1     Running   0          35m
rook-discover-qtfgt                 1/1     Running   0          32m
rook-discover-zcttb                 1/1     Running   0          32m
rook-discover-whskt                 1/1     Running   0          32m
```

kubectl create -f crds.yaml
kubectl create -f cluster.yaml
kubectl -n rook-ceph get pod

```
rook-ceph-mgr-a-6459c77f54-579vg                1/1     Running       0          11s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ kubectl -n rook-ceph get pod
NAME                                            READY   STATUS      RESTARTS   AGE
rook-ceph-operator-6d5456dddd-qvzrd             1/1     Running     0          49m
rook-discover-qtfgt                             1/1     Running     0          46m
rook-discover-zcttb                             1/1     Running     0          46m
rook-discover-whskt                             1/1     Running     0          46m
csi-rbdplugin-hd27l                             3/3     Running     0          12m
csi-cephfsplugin-dsfwj                          3/3     Running     0          12m
csi-rbdplugin-xzf88                             3/3     Running     0          12m
csi-cephfsplugin-shrsm                          3/3     Running     0          12m
csi-cephfsplugin-provisioner-64d4468bbf-8p6hj   5/5     Running     0          12m
csi-cephfsplugin-pssw8                          3/3     Running     0          12m
csi-rbdplugin-rs2fs                             3/3     Running     0          12m
csi-rbdplugin-provisioner-76789bdcc7-nl7lw      6/6     Running     0          12m
csi-cephfsplugin-provisioner-64d4468bbf-9dwsd   5/5     Running     0          12m
csi-rbdplugin-provisioner-76789bdcc7-tpl5f      6/6     Running     0          12m
rook-ceph-mon-a-54dbbdbfcb-vwvnw                1/1     Running     0          8m35s
rook-ceph-mon-b-d6d57884-678vl                  1/1     Running     0          6m22s
rook-ceph-mon-c-656f7bf947-5q9p8                1/1     Running     0          4m5s
rook-ceph-crashcollector-w2-564f7c67b5-ghnhg    1/1     Running     0          4m4s
rook-ceph-mgr-a-6459c77f54-579vg                1/1     Running     0          3m49s
rook-ceph-osd-prepare-w2-54dgb                  0/1     Completed   0          3m21s
rook-ceph-osd-prepare-w3-qjgt4                  0/1     Completed   0          3m21s
rook-ceph-osd-prepare-w1-tjckq                  0/1     Completed   0          3m21s
rook-ceph-osd-0-589494c884-6v9lz                1/1     Running     0          2m23s
rook-ceph-osd-1-85d6869dc-s585w                 1/1     Running     0          2m22s
rook-ceph-crashcollector-w1-6965bf654-nhzzt     1/1     Running     0          2m16s
rook-ceph-crashcollector-w3-8659c6997b-7xkcw    1/1     Running     0          6m22s
rook-ceph-osd-2-664db7d449-2cqpx                1/1     Running     0          2m16s
```

El paso anterior puede tardar un poco hasta alcanzar un estado estable, luego de llegar a esto podemos probar las capacidades de almacenamiento del cluster y ver el estado actual ejecutando esto:

kubectl create -f toolbox.yaml
kubectl -n rook-ceph get pod
*kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod -l "app=rook-ceph-tools" -o *
jsonpath='{.items[0].metadata.name}') bash
ceph status
ceph df
rados df

```
[root@rook-ceph-tools-58df894b89-tkvqj /]# ceph status
  cluster:
    id:     4cc92a4b-916b-479b-a60c-246446a9432e
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum a,b,c (age 11m)
    mgr: a(active, since 10m)
    osd: 3 osds: 3 up (since 9m), 3 in (since 9m)

  data:
    pools:   0 pools, 0 pgs
    objects: 0 objects, 0 B
    usage:   3.0 GiB used, 87 GiB / 90 GiB avail
    pgs:

[root@rook-ceph-tools-58df894b89-tkvqj /]# ceph df
RAW STORAGE:
    CLASS     SIZE       AVAIL      USED        RAW USED     %RAW USED
    hdd       90 GiB     87 GiB     5.4 MiB      3.0 GiB          3.34
    TOTAL     90 GiB     87 GiB     5.4 MiB      3.0 GiB          3.34

POOLS:
    POOL     ID     STORED     OBJECTS     USED     %USED     MAX AVAIL
[root@rook-ceph-tools-58df894b89-tkvqj /]# rados df
POOL_NAME USED OBJECTS CLONES COPIES MISSING_ON_PRIMARY UNFOUND DEGRADED RD_OPS RD WR_OPS WR USED COMPR UNDER COMPR

total_objects    0
total_used       3.0 GiB
total_avail      87 GiB
total_space      90 GiB
```

Anexo II

Aplicación web y almacenamiento distribuidos de dispositivos bloques

kubectl create -f storageclassRbdBlock.yaml
kubectl create -f wordpress.yaml
kubectl get pvc --all-namespaces
kubectl create -f mysql.yaml
kubectl get pvc --all-namespaces

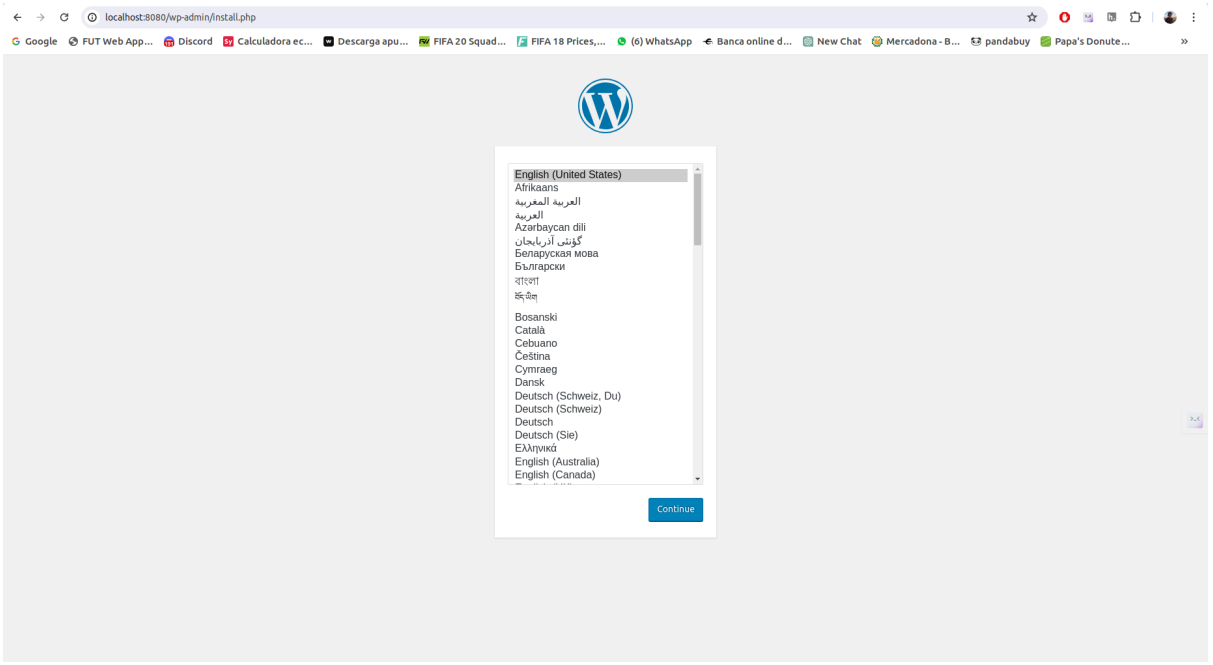
```
deployment.apps/rook-direct-mount created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ kubectl get pvc --all-namespaces
NAMESPACE   NAME             STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS      AGE
default     wp-pv-claim      Bound    pvc-8cb9202b-36bd-4beb-86fe-9bd5c3c368d7   1Gi        RWO            rook-ceph-block   6m25s
default     mysql-pv-claim   Bound    pvc-c97b74ab-72fe-498e-879e-f62c8fb5287a   2Gi        RWO            rook-ceph-block   6m18s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$
```

kubectl get pods

```
default     mysql-pv-claim   Bound    pvc-c97b74ab-72fe-498e-879e-f62c8fb5287a   2Gi        RWO            rook-ceph-block   6m18s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
wordpress-7b989dbf57-f4tn8        1/1     Running   0          7m4s
wordpress-mysql-6965fc8cc8-9265m  1/1     Running   0          6m57s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$
```

#Para comprobar el funcionamiento hacemos un port-forwarding del puerto 80 al 8080 del localhost y desde el navegador hacemos localhost::8080 y aparece la página web wordpress

kubectl port-forward service/wordpress 8080:80



kubectl create -f direct-mount.yaml
kubectl -n rook-ceph get pod -l app=rook-direct-mount; kubectl -n rook-ceph exec -it rook-direct-mount-896d887fb-spkr5 bash

#A continuación podemos crear un sistema RBD llamado réplica pool/ test
y comprobamos información
rbd create replicapool/test --size 10
rbd info replicapool/test

```
[root@w2 /]# rbd create replicapool/test --size 10
[root@w2 /]# rbd info replicapool/test
rbd image 'test':
        size 10 MiB in 3 objects
        order 22 (4 MiB objects)
        snapshot_count: 0
        id: 1b6fef20e5e3
        block_name_prefix: rbd_data.1b6fef20e5e3
        format: 2
        features: layering, exclusive-lock, object-map, fast-diff, deep-flatten
        op_features:
        flags:
        create_timestamp: Tue Apr 23 15:44:14 2024
        access_timestamp: Tue Apr 23 15:44:14 2024
        modify_timestamp: Tue Apr 23 15:44:14 2024
```

#Finalmente comprobamos con el comando lsblk para enumerar los Dispositivos de

#bloques disponibles Y vemos que tras montar el sistema de bloques recién creado
#si volvemos a hacer el lsblk vemos que se ha creado correctamente
lsblk
rbd map replicapool/test

```
[root@w2 /]# rbd feature disable replicapool/test fast-diff deep-flatten object-
map
[root@w2 /]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda      8:0    0  40G  0 disk
`-sda1   8:1    0  40G  0 part /etc/resolv.conf
sdb      8:16   0  10M  0 disk
sdc      8:32   0  30G  0 disk
[root@w2 /]# rbd map replicapool/test
/dev/rbd0
[root@w2 /]# lsblk
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda      8:0    0  40G  0 disk
`-sda1   8:1    0  40G  0 part /etc/resolv.conf
sdb      8:16   0  10M  0 disk
sdc      8:32   0  30G  0 disk
rbd0   252:0    0  10M  0 disk
[root@w2 /]#
```

#Finalmente podemos formatear el volumen y montarlo para comprobar su correcto
#funcionamiento además podemos crear un fichero dentro de este mismo .
mkfs.ext4 /dev/rbd0
mkdir /tmp/test
mount /dev/rbd0 /tmp/test
touch /tmp/test/a.txt
echo "hola mundo" > /tmp/test/hola.txt

```
[root@w2 /]# mkfs.ext4 /dev/rbd0
mke2fs 1.45.6 (20-Mar-2020)
Discarding device blocks: done
Creating filesystem with 10240 1k blocks and 2560 inodes
Filesystem UUID: 0d753960-19c5-4895-94fe-c26ae3e5df71
Superblock backups stored on blocks:
        8193

Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done

[root@w2 /]# mkdir /tmp/test
[root@w2 /]# mount /dev/rbd0 /tmp/test
[root@w2 /]# touch /tmp/test/a.txt
[root@w2 /]# echo "hola mundo" > /tmp/test/hola.txt
[root@w2 /]#
```

#Para comprobar el correcto funcionamiento podemos hacer SSH al nodo del que
#se aloja este pod y apagarlo. Y comprobamos que este pod ha sido desplazado a
#otro nodo

```
rook-ceph     rook-ceph-mon-a-5ff9b76f67-zjxkx              0/2     Pending             0          42s     <none>          <none>
rook-ceph     rook-ceph-osd-2-86b55d7485-ftzsw              0/2     Pending             0          42s     <none>          <none>
rook-ceph     rook-ceph-crashcollector-w2-67c6995fb7-9br7j  0/1     Pending             0          42s     <none>          <none>
rook-ceph     csi-cephfsplugin-provisioner-d5f4bb8c4-gkr7x  0/5     ContainerCreating   0          42s     <none>          w1
rook-ceph     csi-rbdplugin-provisioner-b8f6cd9cf-t9htc     0/5     ContainerCreating   0          42s     <none>          w1
rook-ceph     rook-direct-mount-896d887fb-nd4jl             1/1     Running             0          42s     192.168.1.141   w1
```

#Posteriormente conectarnos al pod y comprobar que la replicación ha funcionado #correctamente y el fichero que hemos creado previamente sigue existiendo

kubectl -n rook-ceph get pod -l app=rook-direct-mount; kubectl -n rook-ceph exec -it rook-direct-mount-896d887fb-nd4jl bash

```
command terminated with exit code 130
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ kubectl -n rook-ceph get pod -l app=rook-direct-mount; kubectl
d4jl bash
NAME                                 READY   STATUS    RESTARTS   AGE
rook-direct-mount-896d887fb-nd4jl    1/1     Running   0          24m
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
[root@w1 /]# cat /tmp/test/hola.txt
hola mundo
```

Anexo III

Registro de repositorio privado de contenedores y almacenamiento de sistema de ficheros distribuido

kubectl create -f filesystem.yaml
kubectl -n rook-ceph get pod -l app=rook-ceph-mds
kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod -l "app=rook-ceph-tools" -o jsonpath='{.items[0].metadata.name}') bash
#Con esto comprobamos que el sistema ceph se ha actualizado correctamente

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod -l "app=rook-ceph-tools" -o jsonpath='{.items[0].metadata.name}')
 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
bash: warning: setlocale: LC_CTYPE: cannot change locale (en_US.UTF-8): No such file or directory
bash: warning: setlocale: LC_COLLATE: cannot change locale (en_US.UTF-8): No such file or directory
bash: warning: setlocale: LC_MESSAGES: cannot change locale (en_US.UTF-8): No such file or directory
bash: warning: setlocale: LC_NUMERIC: cannot change locale (en_US.UTF-8): No such file or directory
bash: warning: setlocale: LC_TIME: cannot change locale (en_US.UTF-8): No such file or directory
[root@rook-ceph-tools-58df894b89-znqxt /]# ceph status
  cluster:
    id:     c262276c-0af5-4426-a594-d44862c67945
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum a,b,c (age 3h)
    mgr: a(active, since 3h)
    mds: myfs:1 {0=myfs-a=up:active} 1 up:standby-replay
    osd: 3 osds: 3 up (since 3h), 3 in (since 17h)

  data:
    pools:   3 pools, 96 pgs
    objects: 99 objects, 210 MiB
    usage:   3.5 GiB used, 86 GiB / 90 GiB avail
    pgs:     96 active+clean

  io:
    client:   853 B/s rd, 1 op/s rd, 0 op/s wr

[root@rook-ceph-tools-58df894b89-znqxt /]#
```

ceph mds stat
##myfs:1 {0=myfs-b=up:active} 1 up:standby-replay

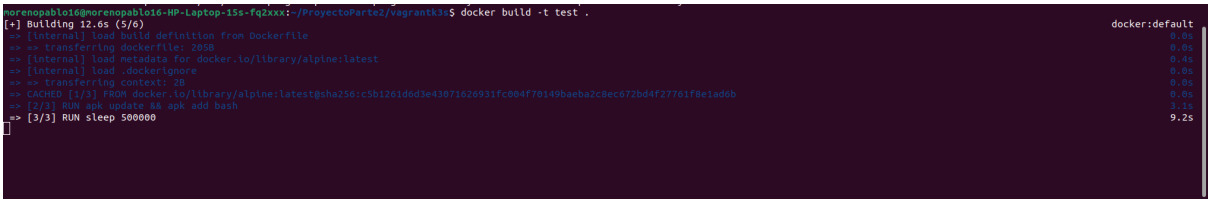
kubectl create -f storageclassCephFS.yaml
kubectl create -f kubeRegistryService.yaml
kubectl create -f kube-registry.yaml
kubectl create -f kubeRegistryProxy.yaml
POD=$(kubectl get pods --namespace kube-system -l k8s-app=kube-registry -o template --template '{{range .items}} {{.metadata.name}} {{.status.phase}}{{"\n"}}{{end}}'| grep Running | head -1 | cut -f1 -d' ')

kubectl port-forward --namespace kube-system $POD 5000:5000 &
kubectl port-forward --namespace kube-system kube-registry-7948766476-vq7c9 5000:5000

#A continuación creamos un contenedor para ser pusheado y pulleado. En este caso usaremos un ALpine Linux
DOCKER FILE #Creamos un docker file muy sencillito para esta prueba

```
FROM alpine:latest
# Update package repositories and install bash
RUN apk update && apk add bash
# Set the default command to run bash
CMD bash -c "sleep 500000"
RUN echo 'hola'
```

#Seguido hacemos build del docker
docker build -t test .
docker tag test localhost:5000/test



docker push localhost:5000/test

```
 => => naming to docker.io/library/test
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ docker tag test:latest localhost:5000/test
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ docker push localhost:5000/test
Using default tag: latest
The push refers to repository [localhost:5000/test]
edcfe24bf32c: Pushed
d4fc045c9e3a: Pushed
latest: digest: sha256:e7ac4b33a38f4770ce922228ba3918ddd849f298f9bc35d48eb818baa7f39db1 size: 739
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$
```

docker pull localhost:5000/test

```
latest: digest: sha256:e7ac4b33a38f4770ce922228ba3918ddd849f298f9bc35d48eb818baa7f39db1 size: 739
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ docker pull localhost:5000/test
Using default tag: latest
latest: Pulling from test
Digest: sha256:e7ac4b33a38f4770ce922228ba3918ddd849f298f9bc35d48eb818baa7f39db1
Status: Image is up to date for localhost:5000/test:latest
localhost:5000/test:latest
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$
```

#Con podman
podman build -t prueba .

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ podman build -t prueba .
STEP 1/4: FROM alpine:latest
Resolved "alpine" as an alias (/etc/containers/registries.conf.d/shortnames.conf)
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Copying blob 4abcf2066143 done
Copying config 05455a0888 done
Writing manifest to image destination
Storing signatures
STEP 2/4: RUN apk update && apk add bash
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.19/community/x86_64/APKINDEX.tar.gz
v3.19.1-440-gbcf32c0243b [https://dl-cdn.alpinelinux.org/alpine/v3.19/main]
v3.19.1-443-g7e4aa1a08e2 [https://dl-cdn.alpinelinux.org/alpine/v3.19/community]
OK: 22993 distinct packages available
(1/4) Installing ncurses-terminfo-base (6.4_p20231125-r0)
(2/4) Installing libncursesw (6.4_p20231125-r0)
(3/4) Installing readline (8.2.1-r2)
(4/4) Installing bash (5.2.21-r0)
Executing bash-5.2.21-r0.post-install
Executing busybox-1.36.1-r15.trigger
OK: 10 MiB in 19 packages
--> bd54eb22650
STEP 3/4: CMD [ "bash" ]
--> 48b9de06d14
STEP 4/4: RUN echo "Hello World"
Hello World
COMMIT prueba
--> 1f5450627e9
Successfully tagged localhost/prueba:latest
1f5450627e95a844327b285eca70733db4761d43e8b40a427873f84d2677d158
```

podman run --name prueba_run -d prueba
podman push --tls-verify=false prueba localhost:5000/user/prueba:latest

```
a9ee56070d750f143171d26be8a5241521a8b443a8d2c8030b3ed28cc6866884
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ podman push --tls-verify=false prueba loc
alhost:5000/user/prueba:latest
Getting image source signatures
Copying blob eaa90c2c54ad done
Copying blob 2049a9724619 done
Copying blob d4fc045c9e3a done
Copying config 1f5450627e done
Writing manifest to image destination
Storing signatures
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$
```

docker pull localhost:5000/user/prueba

```
Storing signatures
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ docker pull localhost:5000/user/prueba
Using default tag: latest
latest: Pulling from user/prueba
fad8106c0d2f: Pull complete
aabf1170d752: Pull complete
159fd50d9ba3: Pull complete
Digest: sha256:1de9d8267fb10ee6968b261cb504e259bf5c825351e4e690b4d9553c4475fc11
Status: Downloaded newer image for localhost:5000/user/prueba:latest
localhost:5000/user/prueba:latest
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$
```

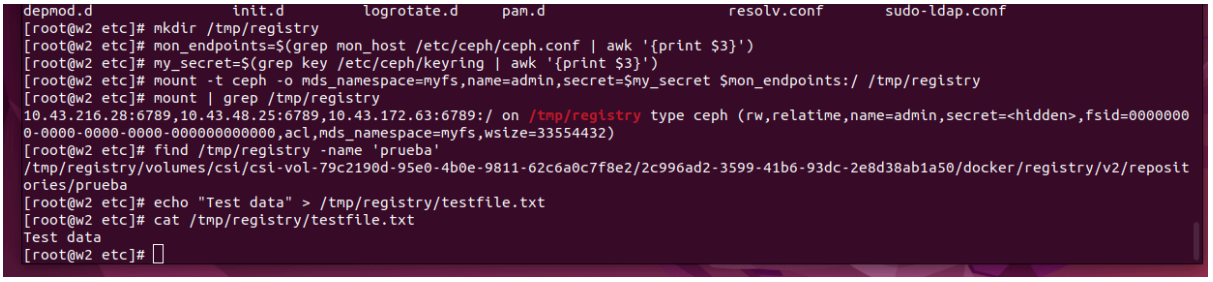
*#*Se puede comprobar si se ha subido correctamente con el siguiente comando:
curl localhost:5000/v2/_catalog

```
enopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ curl localhost:5000/v2/_catalog
epositories":["cliente","server","servidor","test","test2","user/prueba"]}
enopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ kubectl port-forward service/wordpre
80
warding from 127.0.0.1:8080 -> 80
warding from [::1]:8080 -> 80
dling connection for 8080
orenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte2/vagrantk3s$ vagrant halt
 w3: Attempting graceful shutdown of VM...
```

Vamos a montar manualmente el sistema de archivos Ceph en el pod
rook-direct-mount. Los pasos consisten en la creación de un directorio de montaje,
el montaje del sistema de archivos Ceph, la verificación de que el sistema de
archivos está montado y la búsqueda de la imagen del contenedor en el sistema
de archivos.

kubectl -n rook-ceph get pod -l app=rook-direct-mount; kubectl -n rook-ceph exec -it rook-direct-mount-896d887fb-nd4jl bash

mkdir /tmp/registry
mon_endpoints=$(grep mon_host /etc/ceph/ceph.conf | awk '{print $3}')
my_secret=$(grep key /etc/ceph/keyring | awk '{print $3}')
mount -t ceph -o mds_namespace=myfs,name=admin,secret=$my_secret $mon_endpoints:/ /tmp/registry
mount | grep /tmp/registry
find /tmp/registry -name 'prueba'
echo "Test data" > /tmp/registry/testfile.txt
cat /tmp/registry/testfile.txt



#Para comprobar el correcto funcionamiento podemos hacer SSH al nodo del que
#se aloja este pod y apagarlo. Y comprobamos que este pod ha sido desplazado a
#otro nodo
#Posteriormente conectarnos al pod y comprobar que la replicación ha funcionado
#correctamente y el fichero que hemos creado previamente sigue existiendo

Anexo IV

Modificaciones realizadas sobre el vagrant para usar puppet

Primero hemos modificado el Vagrantfile para cambiar el provider para que use libvirt, además hemos modificado el apartado de network.

```
Unset
nodeconfig.vm.provision "shell",
                path: 'puppet.sh'
            nodeconfig.vm.provision "puppet" do |puppet|
                puppet.options = "--verbose"
                puppet.facter = {
                    "hostname" => node[:hostname],
                    "ip" => node[:ip],
                    "m" => node[:m],
                    "type" => node[:type]
```

```
        }
        puppet.manifests_path = "manifests/"
        puppet.manifest_file = "provision.pp"
    end
```

```
Unset
    nodeconfig.vm.network :public_network,
            bridge: "br1", # Nombre de la interfaz de red
            ip: node[:ip], # Dirección IP
            # virtualbox__intnet: true,
            nic_type: "virtio", #Tipo de interfaz de red virtual
            :dev => "br1"
```

A continuación ha sido necesario hacer un storage pool ya que sino no permitía lanzar el comando vagrant up.

Por lo tanto lo hemos creado de la siguiente manera:

virsh pool-define-as --name a841972remote --type dir --target /misc/alumnos/as2/as22023/a841972/ProyectoParte1/vagrantk3s
virsh build a841972remote
virsh pool-start a841972remote
virsh pool-autostart a841972remote
virsh pool-info a841972remote

```
.ab102-201:/misc/alumnos/as2/as22023/a841972/ProyectoParte1/vagrantk3s/ virsh pool-info a841972remote
lombre:         a841972remote
UUID:           fca35200-774a-4e62-88ef-c0811b73a7a3
Estado:         ejecutando
Persistente:    si
Autoinicio:     si
Capacidad:      1,73 TiB
Ubicaci◆n:      575,98 GiB
Disponible:     1,17 TiB

.ab102-201:/misc/alumnos/as2/as22023/a841972/ProyectoParte1/vagrantk3s/
```

Anexo V

Vagrantfile modificado para usar puppet

```
#boxes
# Definición de la referencia al box de Ubuntu 18.04
Ubu = 'ubuntu/bionic64'
# Deb = 'debian/buster64'
#
# @ IP Master = 192.168.1.W9
# @ IP worker w1 = 192.168.1.W1
# ... etc
# Valor ejemplo atribuido a alumnos al principio de asignatura, W = E (14)

MASTER = '192.168.1.149'
# Lista de nodos trabajadores y master con su respectiva información y direcciones Ips
#ademas de la memoria que se le asigna a cada uno y ademas de definirles la ip de su master
NODES = [
 { hostname: 'm', type: "master", ip: MASTER, mem: 1800, m: MASTER },
 { hostname: 'w1', type: "worker", ip: '192.168.1.141', mem: 1800, m: MASTER },
 { hostname: 'w2', type: "worker", ip: '192.168.1.142', mem: 1800, m: MASTER },
 { hostname: 'w3', type: "worker", ip: '192.168.1.143', mem: 1800, m: MASTER },
]
# Configuración de Vagrant
Vagrant.configure("2") do |config| #Crea una instancia de configuración de Vagrant
   NODES.each do |node| # Itera sobre la lista de nodos
        config.vm.define node[:hostname] do |nodeconfig| # Definición de la configuración de
la máquina
           nodeconfig.vm.box = Ubu # Asignación de la imagen a utilizar
           nodeconfig.vm.hostname = node[:hostname] # Asignación del nombre de la máquina

           # Config red
           # Tipo de red en este caso pública, para que desde la red del host sea visible
usando el
           # interfaz br1 de la maquina host
           nodeconfig.vm.network :public_network,
                   bridge: "br1", # Nombre de la interfaz de red
                   ip: node[:ip], # Dirección IP
                   # virtualbox__intnet: true,
                   nic_type: "virtio", #Tipo de interfaz de red virtual
                   :dev => "br1"
           #:mode => "bridge",
           #:type => "bridge"
# Configuración de recursos de la máquina virtual
           config.vm.provider :libvirt do |lib|
               lib.uri = "qemu+ssh://a841972@155.210.154.201/system"
               lib.username = "a841972"
               lib.memory = node[:mem]
               lib.nic_model_type = "virtio"
```

```
            lib.driver = "kvm"
            lib.cpus = 1
            lib.keymap = 'es'
            lib.storage_pool_name = "a841972remote"
        end
        #nodeconfig.vm.synced_folder ".", "/vagrant", type: "rsync"
        if node[:type] == "worker"
            nodeconfig.vm.customize [ "createmedium",
                "--filename", "disk-#{node[:hostname]}.vdi",
                "--size", 30*1024 ] # Que tamaño es este ??
            nodeconfig.vm.customize [ "storageattach", :id,
                "--storagectl", "SCSI",
                "--port", 2, "--device", 0,"--type", "hdd",
                "--medium", "disk-#{node[:hostname]}.vdi" ]
        end

        #Este fragmento de código si no estuviera comentado
        # se encargaría de mirar cada nodo worker  y comprobar si exite el fichero de
disco
        # y si no crearía un controlador SATA para cada nodo y posteriormente un disco
virtual de 20GB

=begin
        if node[:type] == "worker"
            nm = node[:hostname]
            unless File.exist?("disk-#{nm}.vdi")
                v.customize ["storagectl", :id,"--name", "VboxSata",
                                                    "--add", "sata"]
            end
            unless File.exist?("disk-#{nm}.vdi")
                v.customize [ "createmedium", "--filename",
                    "disk-#{nm}.vdi","--size", 20*1024 ]
            end
        end
=end
        #Ahora se bootea la maquina recien creada
        nodeconfig.vm.boot_timeout = 400
        # A continuación es el aprovisionamiento, se establece que se usara un script de
shell llamado
        # provision.sh y se le pasan los argumentos necesarios :(nombre de la máquina,
dirección IP,
        # su master y tipo si es master o worker)
        nodeconfig.vm.provision "shell",
            path: 'puppet.sh'
        nodeconfig.vm.provision "puppet" do |puppet|
            puppet.options = "--verbose"
            puppet.facter = {
                "hostname" => node[:hostname],
                "ip" => node[:ip],
                "m" => node[:m],
                "type" => node[:type]
            }
            puppet.manifests_path = "manifests/"
```

```
            puppet.manifest_file = "provision.pp"
          end
          #args: [ node[:hostname], node[:ip], node[:m], node[:type] ]
          #Finalmente tras realizar el aprovisionamineto se define un trigger
          #que se ejecutará después de que la máquina termine de arrancar, entonces se
copia el fichero k3s.yaml
          #en el directorio /home/morenopablo16/.kube/config
          # No hace caso al if ?
          if node[:type] == "master"
            nodeconfig.trigger.after :up do |trigger|
              trigger.run = \
                      {inline: "sh -c 'cp k3s.yaml /home/morenopablo16/.kube/config'"}
            end
          end
      end
  end
end
```

Provision.pp

```
node default {
 class { 'vagrant_vm':
   hostname => $::hostname,
   nodeip   => $::nodeip,
   masterip => $::masterip,
   nodetype => $::nodetype,
 }
}

class vagrant_vm (
 String $hostname,
 String $nodeip,
 String $masterip,
 String $nodetype,
) {
 exec { 'set_timezone':
   command => 'timedatectl set-timezone Europe/Madrid',
   path    => '/bin:/usr/bin',
 }

 file { '/vagrant':
   ensure => 'directory',
   path   => '/vagrant',
 }
```

```
 host { 'hostname_entry':
   ensure       => present,
   name         => $nodeip,
   ip           => $nodeip,
   host_aliases => $hostname,
 }

 file { '/etc/hosts':
   ensure  => present,
   path    => '/etc/hosts',
   content => "192.168.0.149 m\n192.168.0.141 w1\n192.168.0.142 w2\n192.168.0.143 w3\n",
 }

 file { '/usr/local/bin/k3s':
   ensure  => present,
   source  => '/vagrant/k3s',
   mode    => '0755',
   require => File['/vagrant'],
 }

 if $nodetype == 'master' {
   exec { 'install_k3s_master':
     command => "env INSTALL_K3S_SKIP_DOWNLOAD=true /vagrant/install.sh server --token
'wCdC16AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2d1oDJQQOsBkIwy5OZ/5' --flannel-iface enp0s8
--bind-address $nodeip --node-ip $nodeip --node-name $hostname --disable traefik
--node-taint k3s-controlplane=true:NoExecute",
     path    => '/bin:/usr/bin',
     require => File['/usr/local/bin/k3s'],
   }

   exec { 'copy_k3s_yaml':
     command     => 'cp /etc/rancher/k3s/k3s.yaml /vagrant',
     path        => '/bin:/usr/bin',
     refreshonly => true,
     subscribe   => Exec['install_k3s_master'],
   }
 } else {
   exec { 'install_k3s_agent':
     command => "env INSTALL_K3S_SKIP_DOWNLOAD=true /vagrant/install.sh agent --server
https://$masterip:6443 --token
'wCdC16AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2d1oDJQQOsBkIwy5OZ/5' --node-ip $nodeip
--node-name $hostname --flannel-iface enp0s8",
     path    => '/bin:/usr/bin',
     require => File['/usr/local/bin/k3s'],
   }
 }
}
```

Puppet.sh

```
#!/bin/sh
command -v puppet > /dev/null && { echo "Puppet is installed! skipping" ; exit
0; }

# Install Puppet
apt-get update
apt-get install -y puppet

# Verify installation
command -v puppet > /dev/null && echo "Puppet installed successfully!" || echo
"Failed to install Puppet"
```