# *AS2 : Proyecto 2 (Parte 2 de 3)*

## Objetivo

Despliegue de un aplicación web básica y un repositorio privado de contenedores dentro de Kubernetes utilizando un soporte de almacenamiento distribuido de alta disponibilidad.

## Resumen

Realizar el despliegue de una aplicación web y un repositorio privado de contenedores que utilicen un despliegue interno de Ceph, mediante Rook, tanto como dispositivo de bloques como de sistema de ficheros distribuido. El despliegue inicial de Kubernetes será similar a la primera parte, pero con ligeras modificaciones tanto en el Vagrantfile (un disco virtual mayor y diferente – SCSI – y más memoria RAM, por cada VM).

## Evaluación

Para esta segunda parte se deberá **entregar una memoria**, de un **máximo de 6 paginas,** anexos como páginas adicionales, en la que, al menos, se incluya:

1.- Resumen

2.- Arquitectura de elementos relevantes del despliegue Kubernetes/Ceph/Aplicaciones.

3.- Comprehension y explicación de las diferentes aplicaciones desplegadas sobre Kubernetes con la explicación de los conceptos y los recursos Kubernetes utilizados y explicación básica de los conceptos y recursos utilizados por Ceph.

5.- Explicación de los métodos utilizados para la validación de la operativa.

4.- Problemas encontrados y su solución.

**Además**, se deberá mostrar el funcionamiento del despliegue manual, concertando cita previa con el profesor.

**Fecha límite de entrega de memoria : 3 de mayo de 2024**.

## Datos de la configuración y uso de Kubernetes (K3s)

**Requisitos de sistema:**

Linux, VirtualBox, Vagrant, libvirt, Terraform RAM libre > **5.8 GB**, disco libre > 14 **GB**.

**Ficheros a utilizar de la parte 1 :**

- fichero *Vagrantfile*,
- fichero *provision.sh*

- fichero *install.sh* *# Programa shell de configuración de K3s*
- Ejecutable *k3s* *# Versión v1.23.5+k3s1*
- Ejecutable *kubectl* *# v1.23.5*

  **Adaptar** a vuestra situación, en el fichero ***Vagrantfile***, la referencia incluida al PATH completo del fichero ".../.kube/config". **Copiar** el ejecutable ***kubectl*** a un directorio de vuestro HOME que este incluido en variable shell PATH. Todos estos ficheros, salvo Vagrantfile, deben tener el **permiso de ejecución** activado.

En esta segunda parte, seguimos utilizando sólo un maestro y 3 nodos trabajador (worker).

Se definen, en *Vagrantfile* y *provision.sh*, de forma explicita los siguientes datos de **configuración inicial de K3s** (Kubernetes) : IP de cada máquina, dirección MAC, Nombre de cada máquina, token de autentificación entre maestro kubernetes y resto de nodos, interfaz de red donde opera Flannel en cada máquina, inhabilitar Ingress (Traefik) , el maestro no debe ser utilizado para ejecución de Pods.

El sistema operativo Linux que se instala en las VMs, y que da soporte a Kubernetes, debe operar con la herramienta *iptables* clásica (no ngtables). Aqui, utilizamos Ubuntu 18.04 (bionic) AMD64 por ser el mejor validado, y que ya lo incluye por defecto.

Adicionalmente, en el directorio *aplicacionesCephRook* del fichero comprimido *ProyectoParte2.tar.xz*, disponeis de todos los manifiestos Kubernetes (yaml) de despliegue de aplicaciones y recursos Kubernetes que podeis utilizar. Aplicareis dichos manifiestos o las modificaciones y añadidos que realiceis sobre ellos.

# Enunciado

En esta 2ª parte del proyecto se plantean 3 etapas :

- Puesta en marcha de una infraestructura básica de almacenamiento distribuida de alta disponibilidad basada en Ceph, mediante Rook, sobre un cluster Kubernetes con 3 nodos.

- Poner en marcha una aplicación web básica, basada en wordpress y mysql, utilizando dispositivos de bloques Ceph (RBD) creados a partir del despliegue previo de Ceph . Y formateados y montados explicitamente para las aplicaciones.

- Poner en marcha un registro de repositorio privado de contenedores (sin seguridad TLS), utilizando un sistema de ficheros distribuido Ceph creado a partir del despliegue Ceph previo y montados explicitamente para las aplicaciones.

La puesta en marcha inicial de las de 3 VMs se realizan también mediante la herramienta *vagrant* como en la 1ª parte. Como aspectos específicos, incrementar la memoria de todos los **workers** a **1800M**, e incluir discos duros adicionales para los workers mediante un código Vagrant file similar al del anexo A.

*¿ Qué tamaño de disco se está asignando a cada VM ?*

En la provisión de las VMs, recordad que **NO** debeis deshabilitar el balanceador de carga en la opciones de instalación de K3s.

Para utilizar adecuadamente la herramienta *kubectl*, debeis copiarlo en un directorio incluido en la variable shell *PATH*, y crear, con *mkdir*, el directorio de configuración "**.***kube*" en vuestro HOME. Una vez puesta en marcha el Cluster de 3 VMs con *K3s*, comprobar el funcionammiento de *K3s* y *kubectl* ejecutando en línea de comandos del host "*kubectl get nodes*".

# Etapa inicial : puesta en marcha básica de Ceph en Kubernetes

Una vez que el cluster Kubernetes esté en funcionamiento, efectuar el despliegue de Ceph sobre Kubernetes con Rook utilizando los manifiestos de Kubernetes del directorio "*aplicacionesCephRook/ceph*", y empezando por poner en marcha el *operador rook* (como recurso Kubernetes personalizado) :

- *kubectl create -f common.yaml*
- *kubectl create -f operator.yaml*

Esperar que estén en marcha los pods del deployment del operador Rook: *kubectl -n rook-ceph get pod*. Tarda un poquito. Deberían verse 4 pods :

- rook-ceph-operator-.....
- rook-discover-.....
- rook-discover-.....
- rook-discover-.....

Una vez estén en funcionamienot los pods del operador *rook*, poner en marcha la infraestructura básica del sistema de almacenamiento Ceph con la ayuda de dicho operador *rook* mediante el manifiesto *aplicacionesCephRook/ceph/cluster.yaml, con* el comando "*kubectl create -f ....*".

Comprobar la puesta en funcionamiento de los diferentes componentes distriabuidos de Ceph (descubrimiento, plugins de sistemas de fichero y RBD, monitores, gestor -mgr- , OSDs, ... entre otros) mediante :

- *kubectl -n rook-ceph get pod*

Esta fase dura más tiempo, y podeis observar como se ponen en marcha dichos componentes distribuidos de Ceph. En particular, explorará los 3 nodos Kubernetes para ver que discos duros están vacios (ningun formato), sin utilización y formateará con su formato base de bajo nivel (*BlueStore*) :

- https://ceph.io/community/new-luminous-bluestore/
- https://docs.ceph.com/docs/master/rados/configuration/bluestore-config-ref/

Ahora, comprobar el estado de funcionamiento de Ceph a través del despliegue de un pod de herramientas Ceph (*rook-ceph-tools*)). El manifiesto *aplicacionesCephRook/ceph/toolbox.yaml*, permitirá crear dicho pod con "*kubectl create -f ....*".

Comprobar su puesta en funcionamiento completa con :

- *kubectl -n rook-ceph get pod -l "app=rook-ceph-tools"*

Y finalmente acceder a la caja de herramientas de ceph, dentro del pod, mediante :

- *kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get pod -l "app=rook-ceph-tools" -o \ jsonpath='{.items[0].metadata.name}') bash*

Ya dentro del pod de herramientas Ceph, podeis utilizar diferentes comandos cliente de gestión de Ceph. Por ejemplo :

- *ceph status*
- *ceph df*
- *rados df*

Los 3 demonios de monitorización deben ser mostrados junto con el quorum a, b, d. También el gestor (mgr:a (active ...) y los 3 osds ( osd: 3 osds: 3 up ....) correspondientes a los 3 discos duros puestos a disposición en las VMs.

Si todo es correcto , teneis funcionando un sistema de almacenamiento de alta disponibilidad Ceph con replicación básica de hasta 3 nodos.

Esto correponde a un despliegue de pruebas automático. En un entorno de producción habría que asociar el despliegue a determinados nodos de Kubernetes, por ejemplo mediante labels, para controlar de forma más precisa sus características de funcionamiento (ubicación de discos, recursos de red, alta disponibilidad, prestaciones, etc).

# Etapa 2 : Aplicación web y almacenamiento distribuidos de dispositivos bloques

El almacenamiento de bloques permite montar un almacen de datos a un solo Pod. En esta etapa ,vamos a utilizarlo para crear una sencilla aplicación web que utilice volumenes persistentes a partir del despliegue rook-ceph.

Ahora que Ceph está ya en funcionamiento, como administradores del Cluster de Kubernetes, ponemos en marcha 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 mediante el aprovisionador ligado al operador rook-ceph puesto en marcha en la etapa inicial.

Para ello, se pueden crear los recursos Kubernetes asociados aplicando el manifiesto s*torageclassRbdBlock.yaml.*

*A* continuación crear los recursos asociados a la aplicación web. En el directorio *aplicacionesCephRook, d*isponeis de 2 manifiestos, *mysql.yaml y wordpress.yaml*, incompletos, como una sugerencia para tal labor.

Completarlos y poner en funcionamiento una aplicación *wordpress*.

Comprobar su funcionamiento mediante un acesso *http* mediante *curl* o *wget*, desde otro pod del mismo cluster Kubernetes.

También, se puede comprobar la utilización del nuevo pool de dispositivos de bloques de *Ceph* (RBD), mediante un pod de creación y uso manual de dispisitovs de bloques a partir de ese pool. La creación de este pod la teneis disponible en el manifiesto *aplicacionesCephRook/ceph/direct-mount.yaml*. Una vez puesto en marcha este pod, su acceso lo podeis hacer con :

- *kubectl -n rook-ceph get pod -l app=rook-direct-mount; kubectl -n rook-ceph exec -it <pod> bash*

Por ejemplo, centro de este pod, algunas operaciones manuales que podeis hacer son :

```
$ rbd create replicapool/test --size 10
$
$ rbd info replicapool/test
$
$ # Disable the rbd features that are not in the kernel module
$ rbd feature disable replicapool/test fast-diff deep-flatten object-map
$
$ #Map the block volume and format it and mount it:
$
$ # Map the rbd device. If the Direct Mount Pod was started with "hostNetwork: false" this hangs
  and you have to stop it with Ctrl-C,
$ # however the command still succeeds; see https://github.com/rook/rook/issues/2021
$ rbd map replicapool/test
$
$ # Find the device name, such as rbd0 or ...
$ lsblk | grep rbd
$
$ # Format the volume (only do this the first time or you will lose data)
$ mkfs.ext4 -m0 /dev/rbd0 (o rbd2 ?)
```

Crear un directorio, por ejemplo en /tmp, y montar este dispositivo. Comprobar que se puede escribir.

Para comprobar el funcionamiento tolerante a fallos, parar uno de los nodos workerKubernetes. Seleccionar aquel que no tenga ni al pod DNS, ni el de metricas, ni el ceph-mgr (comprobarlo con "kubectl get pods --all-namespaces -o wide"). Y comprobar que, tanto la aplicación web como el fichero manualmente creado, siguen funcionando.

Y finalmente desmontar el dispositivo manual normalmente. Y quitarle del mapa de RBD con "rbd unmap /dev/rbd0".

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

Un sistema de fichero distribuido y compartido puede ser montado con permisos de lectura/escritura y utilizado desde múltiples Pods, para aplicaciones distribuidas que requieran un sistema de ficheros compartido. En esta etapa puede ser utilizado para poner en funcionamiento el registro de un repositorio privado de contenedores sin seguridad TLS.

En primer lugar, crear el sistema de ficheros Ceph, definiendo la configuración para el pool de metadatos, los pools de datos y el servidor de metadatos en Ceph. Para ello, podeis utilizar el manifiesto *aplicacionesCephRook/ceph/filesystem.yaml.*

Comprobar que el sistema de fichero está configurado y los Pods MDS puestos en marcha con "*kubectl -n rook-ceph get pod -l app=rook-ceph-mds*". Adicionalmente, entrar en el pod *toolbox* de la etapa inicial y comprobar con el sistema de ficheros está funcionando correctamente en Ceph, mediante "*ceph status*" y su salida con un par de lineas adicionales del estilo de :

```
 ...
  services:
          ....
    mds: myfs:1 {0=myfs-a=up:active} 1 up:standby-replay
```

Ahora, para crear volumenes persistentes mediante el sistema de ficheros Ceph, se necesita crear el StorageClass asociado. Esto lo podeis crear con el manifiesto *plicacionesCephRook/ceph/storageclassCephFS.yaml.*

En este punto, el sistema de ficheros de Ceph debería estar disponible para aplicaciones ejecutadas en Kubernetes. Y va a ser aprovechado para poner en funcionamiento un registro de repositorio privado de contenedores.

Teneis a disposición 2 manifiestos, *kube-registry.yaml*, *kubeRegistryService.yaml* para poner en funcionamiento el registro con almacen sobre sistema de ficheros de Ceph. Completar el enlace con los volumenes para su funcionamiento.

Y el manifiesto *kubeRegistryProxy.yaml* como método de acceso local en las VMs al registro desplegado con un "curl localhost:5000" que no debe devolver por ahora nada efectivo.

Para comprobar su funcionamiento, se puede habilitar, mediante un "port-forward" una conexión directa desde el host al registo privado de Kubernetes de la siguiente forma :

```
$ 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 &
```

Para comprobar su funcionamiento se puede crear y subir a dichon registro un nuevo contenedor. Para esta labor se puede utilizar "podman" (*https://podman.io/getting-started/installation*), con un Dockerfile mínimo (FROM) a partir de un contenedor mínimo (únicamente con un shell, estilo *busybox* o *bash*), y mediante subcomandos estilo "*build*", "*run*" y "*push*" (*sudo podman push --tls-verify=false 330fdabba8e4 localhost:5000/me/prueba:latest*). Aunque, de error al conectarse con el registro, insistir con el push (pero no demasiado) hasta que veais aparecer, entre líneas, algo del estilo de "*copying blob <XXXXXXXXXX> skipped: already exists*".

Comprobarlo con un manifiesto que utilice esa nueva imagen que tenga una línea de definition de imagen del estilo de :

```
"image: localhost:5000/user/container"
```

O ejecutando en línea de comandos algo del estilo de :

```
kubectl run --generator=run-pod/v1 -i --tty p --image=localhost:5000/user/container -- sh
```

Tambien se puede comprobar 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. Aunque el montaje es un modo más elaborado. Se crea un directorio de montaje (en */tmp?*) y se monta como sigue :

```
$ # Detect the mon endpoints and the user secret for the connection
$ mon_endpoints=$(grep mon_host /etc/ceph/ceph.conf | awk '{print $3}')
$ my_secret=$(grep key /etc/ceph/keyring | awk '{print $3}')
$
$ # Mount the filesystem
$ mount -t ceph -o mds_namespace=myfs,name=admin,secret=$my_secret $mon_endpoints:/
  /tmp/registry
```

Comprobar que está montado y mirar su contenido con "find" para buscar la imagen de contenedor que habeis subido al registro privado. Adicionalmente podeis crear un fichero, escribir en él y ver si perdura entre montajes y desmontajes.

Para comprobar el funcionamiento tolerante a fallos, parar uno de los nodos workerKubernetes. Seleccionar aquel que no tenga ni al pod DNS, ni el de metricas, ni el ceph-mgr (comprobarlo con "kubectl get pods --all-namespaces -o wide"); el resto ya se regenerará o tiene más réplicas funcionando. Y comprobar que, tanto la aplicación web como el fichero manualmente creado, siguen funcionando.

## Eliminación de recursos

Para eliminar ceph de Kubernetes consultar :

https://rook.io/docs/rook/v1.3/ceph-teardown.html

Para eliminar otros recursos de Kubernetes utilizar el subcomando "delete" de kubectl, como por ejemplo :

kubectl delete -f wordpress.yaml

kubectl delete -f mysql.yaml

kubectl delete -n rook-ceph cephblockpools.ceph.rook.io replicapool

kubectl delete storageclass rook-ceph-block

kubectl delete -f ceph/kubeRegistry.yaml

### To delete the filesystem components and backing data, delete the Filesystem CRD.

## WARNING: Data will be deleted if preservePoolsOnDelete=false.

kubectl -n rook-ceph delete cephfilesystem myfs

## Distribución de VMs : Utilización de Libvirt con Vagrant y Puppet para provisión de configuración

Reprogramar Vagrant (su Vagrantfile), para utilizar libvirt, en lugar de Virtualbox, para distribuir las máquinas virtuales en diferentes máquinas físicas en el laboratorio 1.02. Utilizar vuestro identificador W y las indicaciones del anexo A de la 1ª parte del proyecto, tanto para ubicar el **VAGRANT-HOME** (sea en directorio /mnt/scratch/aXXXXXX o el directorio /misc/alumnos/as22020/aXXXXXX) como para diferenciaros de vuestros compañeros tanto a nivel de direcciones **MAC** como direcciones **IP**.

Posteriormente, es necesario que instaleis el plugin libvirt de Vagrant (*vagrant plugin install vagrant-libvirt*).

Finalmente, modificar el fichero Vagrantfile para utilizar Libvirt en lugar de VirtualBox. En el anexo B teneis algunas indicaciones al respecto.

Substituir la utilización del programa shell "provision.sh" con una manifiesto Puppet para poner en marcha k3s en las máquinas virtuales,. Utilizar, en lo posible, recursos Puppet para definir la configuración.

## OPCIONAL : Despliegue de infraestructura con Terraform

Utilizar una cuenta estudiante en Azure, Amzon o Google, y realizar el despliegue de esta parte mediante Terraform en el Cloud público que hayais elegido.

Posterioemtne utilizar el plugin libvirt de Terraform para realizar dicho despliegue sobre libvirt en las máquinas físicas del laboratorio 1.02, como Cloud privado.

## Referencias

**Rook-Ceph** : https://rook.io/docs/rook/master/ceph-storage.html

**Ceph** : https://docs.ceph.com/docs/master/

k3s : https://rancher.com/docs/k3s/latest/en/

**Tema 10** : Kubernetes. Complementos : Validación y pruebas : Vagrant

Vagrant : https://www.vagrantup.com/docs/

Kubernetes :

https://kubernetes.io/docs/home/

shttps://access.redhat.com/documentation/en-us/openshift_container_platform/4.4/

**Conceptos** : https://kubernetes.io/docs/concepts/

**Kubectl** :

https://kubernetes.io/docs/user-guide/kubectl-cheatsheet/

https://kubernetes.io/docs/reference/kubectl/overview/

Referencia completa :

https://kubernetes.io/docs/user-guide/kubectl

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

**gestión de almacenamiento** :

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.3/

# Anexo A

Código Vagrantfile para compatir directorio vagrant remoto y para añadir discos virtuales adicionales a añadir a cada worker (dentro del iterador del proveedor Virtualbox)

```
# Para compartir directorio vagrant pero en remoto con método rsync
# documentaros para saber un poco más sobre rsync
nodeconfig.vm.synced_folder ".", "/vagrant", type: "rsync"

if node[:type] == "worker"
    v.customize [ "createmedium",
                "--filename", "disk-#{node[:hostname]}.vdi",
                "--size", 30*1024 ]  # Que tamaño es este ??

    v.customize [ "storageattach", :id,
                 "--storagectl", "SCSI",
                 "--port", 2, "--device", 0,"--type", "hdd",
                 "--medium", "disk-#{node[:hostname]}.vdi" ]
  end
```

# Anexo B

Teneis información con ejemplos en :

https://github.com/vagrant-libvirt/vagrant-libvirt/blob/master/README.md

https://blogs.oracle.com/linux/getting-started-with-the-vagrant-libvirt-provider-for-oracle-linux

Podeis modificar el Vagrantfile previo con Virtualbox para incluir la utilización de libvirt para despliegue de la VMs, cada una en una máquina física diferente.

Para la configuración de red, realizar en modo bridge con "br1" como "dev".

Para la configuración de los recursos básicos de una VM en libvirt , un ejemplo puede ser :

```
nodeconfig.vm.provider :libvirt do |v|
  v.uri = 'qemu://155.210.154.191/system'
  v.driver = "kvm"
  v.memory = node[:mem]
  v.cpus = 1
  v.keymap = 'es'
end
```