Capacitación Kubernetes

Servicio de capacitación y entrenamiento ARIU

Clase 5
Almacenamiento

Agenda

- **Introducción**
- Objetos
- Selección de almacenamiento
- Práctica
- Benchmark
- Hardware necesario

Introducción

- Kubernetes no se encarga de garantizar el almacenamiento, pero provee objetos para facilitar la conexión.

Almacenamiento Temporal (Ephemeral):

- Se utiliza para almacenar datos efímeros o temporales.
- No persiste los datos entre reinicios de pod.

Almacenamiento Persistente (Persistent):

- Permite que los datos persistan a lo largo del tiempo y reinicios de pod.
- Adecuado para bases de datos u otros datos críticos.

El almacenamiento persistente puede ser

- Local en el Servidor
  - Datos almacenados directamente en los nodos del clúster.
  - Rápido acceso, pero no tolerante a fallos.
- Remoto en la Propia Organización
  - Utiliza almacenamiento compartido dentro de la organización.
  - Mayor disponibilidad y redundancia.
- En la Nube
  - Utiliza servicios de almacenamiento ofrecidos por proveedores de la nube (por ejemplo, Amazon EBS, Google Persistent Disk).
  - Altamente escalable y tolerante a fallos.

Agenda

- Introducción
- **Objetos**
- Selección de almacenamiento
- Práctica
- Benchmark
- Hardware necesario

Almacenamiento - PV

Objeto que abstrae los detalles de cómo se proporciona el almacenamiento a partir de cómo se consume.

- Los PV no pertenecen a ningún NS, deben ser manipulados por el administrador
- Los tipos de PV están implementados como Plugins, actualmente se puede usar: Aws, Azure, CephFS, ~~GlusterFs~~, iscsi, nfs, etc
- El ciclo de vida es independiente de cualquier Pod.

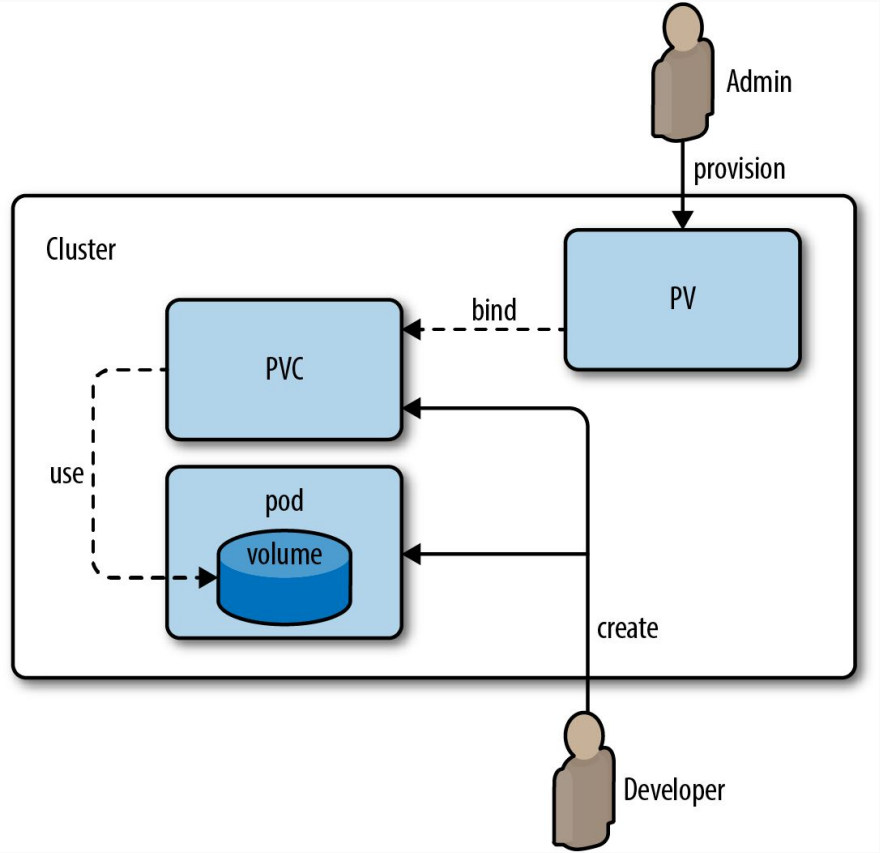
Almacenamiento - PVC

Es una solicitud de almacenamiento por parte de un usuario

- Los PVC pertenecen a un NameSpace
- Al vincularlos con un PV se realiza una conexión uno a uno
- Un PVC vinculado a un Pod no se borra de inmediato, de igual forma el PV respecto del PVC. Queda pendiente de eliminación hasta que se libera el recurso

Almacenamiento - PVC

Estático



Se va a usar para el ejemplo un servidor NFS (Almacenamiento remoto)

Preparar servidor NFS

- mkdir /app/k8s
- sudo apt install nfs-kernel-server rpcbind
- systemctl start rpcbind
- sudo systemctl enable rpcbind
- sudo systemctl enable nfs-server
- echo "/app/k8s 172.27.101.0/24(rw,no_root_squash,sync)" >> /etc/exports
- exportfs -r

Eliminar namespace completo

- k delete ns prueba
- k create ns prueba

Crear PV

- k apply -f 01-pv.yaml
- k get pv

Crear PVC

- k apply -f 02-pvc.yaml
- k get pv
- k get pvc

Iniciar pod con volumen

- k apply -f 03-pod.yaml

Port forward para acceder al pod por web (falla por falta de index)

- k port-forward pod/nginx 8080:80
- > localhost:8080

Crear index.html entrando al pod

- k exec -it nginx -- bash
- echo "Nuevo index" >> /usr/share/nginx/html/index.html
- k port-forward pod/nginx 8080:80
- > localhost:8080

Eliminar y levantar nuevamente pod para comprobar persistencia

- k delete pod nginx
- k apply -f 03-pod.yaml
- > localhost:8080

Modificar index.html desde el servidor NFS

- nano /app/k8s/nginx/index.html
- > localhost:8080

Storage Class

- Define una clase de almacenamiento con propiedades específicas y capacidades de almacenamiento que pueden ser solicitadas por las PVCs.
- Cada SC está asociada a un proveedor de almacenamiento y contiene información sobre cómo provisionar y configurar los Persistent Volumes según los requisitos definidos.
- Cuando se crea una PVC sin especificar una clase de almacenamiento, Kubernetes utiliza el StorageClass predeterminado (si está configurada)

Storage Class

- Permite a los usuarios solicitar diferentes niveles de almacenamiento sin preocuparse por la infraestructura subyacente.
- Definir diferentes StorageClasses según las políticas de almacenamiento necesarias (por ejemplo, almacenamiento con replicación para alta disponibilidad o almacenamiento local para mayor rendimiento).

Política de Reclamación en Kubernetes

- Configuración aplicada a PV una vez que ya no son necesarios por PVC.
- Define qué ocurre con los recursos cuando un PV queda liberado.

Tipos de Reclam Policy

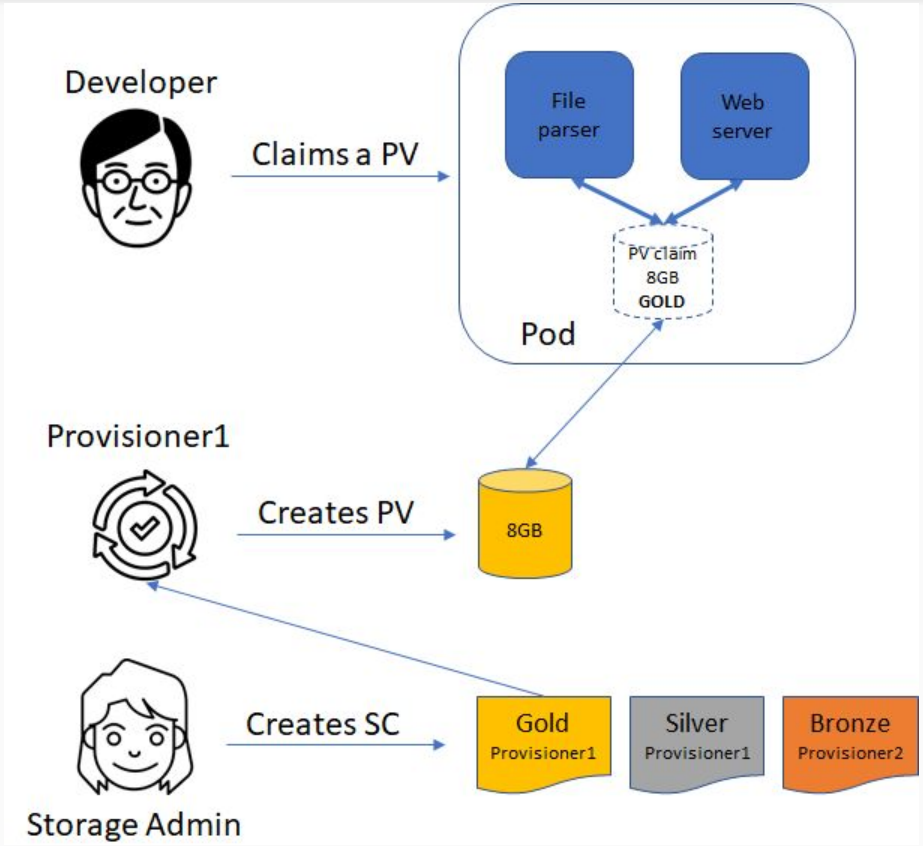
Retain

- Conserva el PV y sus recursos después de ser liberado.
- Requiere intervención manual para liberar recursos.
- Útil para datos críticos o valiosos.

Delete

- Elimina automáticamente el PV y sus recursos al liberarse.
- Adecuado para almacenamiento temporal o de pruebas.

Dinámico

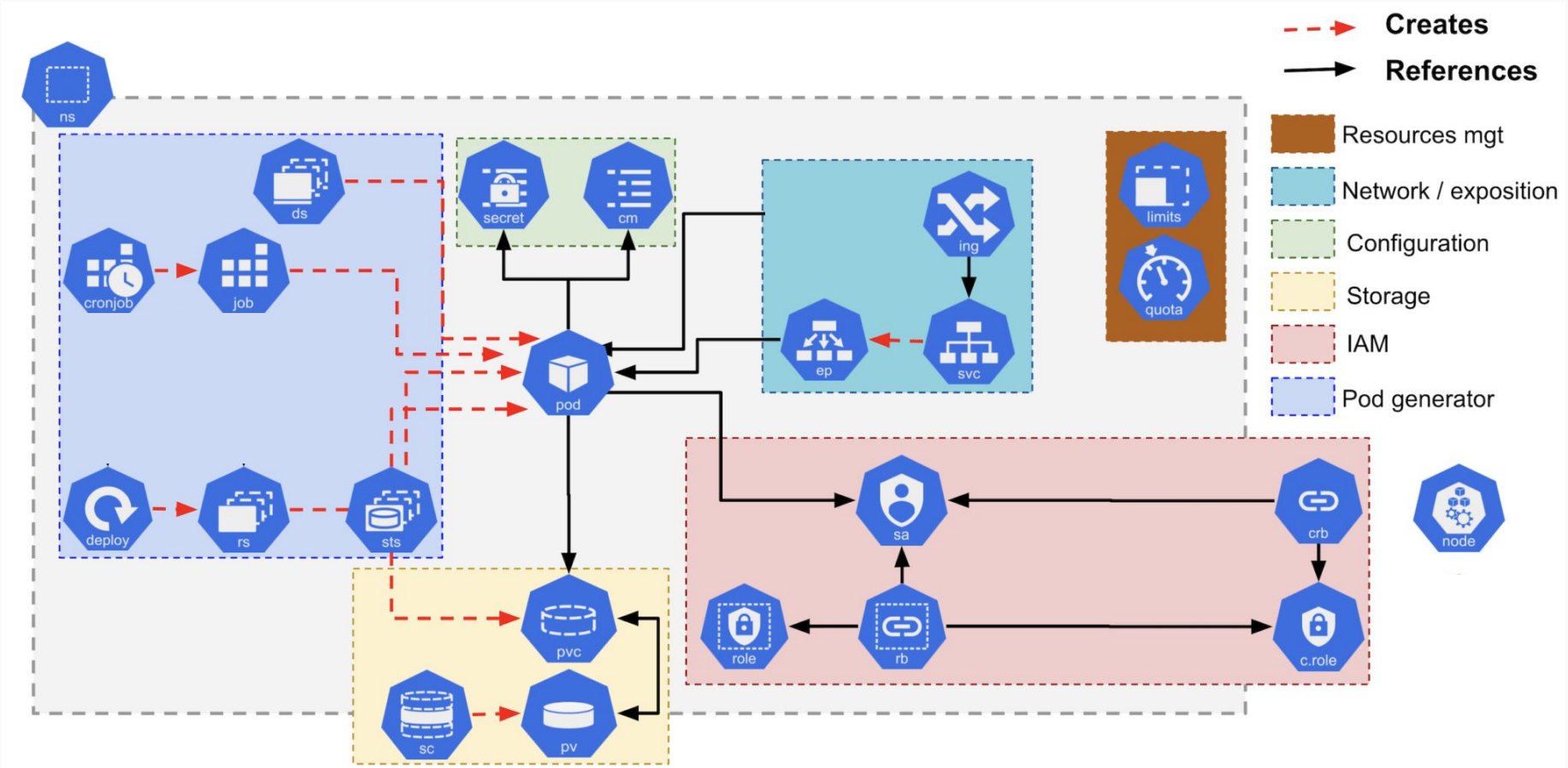

Developer
Claims a PV
File parser
Web server
PV claim
8GB
GOLD
Pod
Provisioner1
Creates PV
8GB
Creates SC
Gold
Provisioner1
Silver
Provisioner1
Bronze
Provisioner2
Storage Admin

- Eliminar namespace completo
  - k delete ns prueba
  - k create ns prueba
  - k delete pv nginx-volum
- Instalar Provisioner con su StorageClass
  - helm repo add nfs-subdir-external-provisioner https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/
  - helm install nfs-subdir-external-provisioner \
  - nfs-subdir-external-provisioner/nfs-subdir-external-provisioner \
  - --set nfs.server=172.27.101.231 \
  - --set nfs.path=/app/k8s \
  - --set storageClass.onDelete=true \
  - --create-namespace \
  - --namespace nfs-provisioner-system

Ejemplo práctico - SC

- Crear PVC
  - k apply -f 04-pvc-storageclass.yaml
  - k get pvc
  - k get pv
  - k get pod
  - > SSH al server NFS y mostrar directorio creado en /app/k8s
- Eliminar pod, pvc y chequear
  - k delete pod nginx
  - k delete pvc nginx-volum-sc
  - > Chequear en el server NFS que el provisioner elimina el directorio pero mantiene un backup
  - ls -l /app/k8s
  - helm show values nfs-subdir-external-provisioner/nfs-subdir-external-provisioner | grep -5 archived

Grupos de Objetos - Resumen



Agenda

- Introducción
- Objetos
- **Selección de almacenamiento**
- Práctica
- Benchmark
- Hardware necesario

Almacenamiento - Selección

- Es útil tener más de un medio de almacenamiento, dado que las aplicaciones desplegadas pueden tener distintos requerimientos
- La primera restricción es el equipo de almacenamiento que cada universidad ya posee.

Análisis de las necesidades específicas

- Tipo de aplicaciones y sus requisitos de almacenamiento.
- Nivel de disponibilidad y tolerancia a fallos requeridos.
- Escalabilidad y capacidad de expansión.
- Rendimiento y velocidad de acceso.

Almacenamiento - Selección

- App Nivel Usuario
  - Longhorn (iSCSI)
  - ~~Portwork~~ $ $
  - OpenEbs
    - Jiva (iSCSI) Archivos dispersos
    - cStore (iSCSI) Archivos replicados
    - Mayastor (NVMe-oF)
- App externa
  - Ceph (rook)
  - ~~Glusterfs (Heketi)~~
- Nas
  - TrueNas
  - Nas propietario
- Proveedores Nube
  - AzureFile
  - VsphereVolume

Experiencias

SIU

- Nas con Samba y NFS (Ventajas, desventajas)
- TrueNas NFS (Ventaja)

UNICEN

- Glusterfs (porque fué elegido)
- OpenEBS
- Longhorn

Agenda

- Introducción
- Objetos
- Selección de almacenamiento
- **Práctica**
- Benchmark
- Hardware necesario

Introducción

- Longhorn es una solución de almacenamiento distribuido y persistente para clústeres de Kubernetes.
- Utiliza contenedores nativos de Kubernetes para implementar su arquitectura.
- Ofrece una interfaz de usuario Web intuitiva que permite a los administradores gestionar fácilmente los volúmenes, snapshots, backup y configuraciones.
- Es posible crear el volumen sin réplica (un solo nodo) o con más de una.
- Si un nodo se cae, las réplicas que se pierden se regeneran en otro nodo.

- Pre-requisitos
- Verificar pre-requisitos
  - apt install jq
  - curl -sSfL https://raw.githubusercontent.com/longhorn/longhorn/v1.5.1/scripts/environment_check.sh | bash
- Instalar en todos los nodos `open-iscsi`
  - apt install open-iscsi
- Verificar que todos los nodos sean scheduleables
  - k taint nodes ed-k8s-m01 node-role.kubernetes.io/control-plane:NoSchedule-
  - k taint nodes ed-k8s-001 node-role.kubernetes.io/control-plane:NoSchedule-
  - k taint nodes ed-k8s-002 node-role.kubernetes.io/control-plane:NoSchedule-
  - > Volver a verificar pre-requisitos

- Instalar
  - kubectl apply -f https://raw.githubusercontent.com/longhorn/longhorn/v1.5.1/deploy/longhorn.yaml
- Crear usuario para autenticar en frontend
  - USER=admin; PASSWORD=<PASSWORD_HERE>; echo "${USER}:$(openssl passwd -stdin -apr1 <<< ${PASSWORD})" >> auth
  - kubectl -n longhorn-system create secret generic basic-auth --from-file=auth

Probar

- Describir storage class
  - k get sc
  - k describe sc longhorn
- Crear pod y pvc para probar
  - k apply -f 05-pvc-pod-longhorn.yaml
  - k get pod
  - k get pvc
- Editar servicio para que sea de tipo `LoadBalancer`
  - k edit service longhorn-frontend
  - > http://<IP>
- Verificar path en algún nodo
  - ls -l /var/lib/longhorn/replicas/pvc-<ID>

Agenda

- Introducción
- Objetos
- Selección de almacenamiento
- Práctica
- **Benchmark**
- Hardware necesario

pgbench

- Se crea una base de datos con una cantidad parametrizable de tuplas.
- Se puede parametrizar en la consulta:
  - **<clientes>** es el número de clientes simultáneos que ejecutarán las transacciones.
  - **<hilos>** es el número de hilos por cliente.
  - **<transacciones>** es el número total de transacciones a ejecutar.

Ejemplo pgbench

- create database test;
- pgbench -i -s 50 test
- pgbench -c <clientes> -j <hilos> -t <transacciones> <base>
- pgbench -c 4 -j 4 -t 1000 test

dd

- Herramienta simple para copiar archivos

dd if=/dev/zero of=/<path>/test bs=64k count=16k conv=fdatasync

- Si se usa un archivo de entrada el máximo de lectura limita el resultado

Salida

16384+0 records in
16384+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 4.5823 s, **234 MB/s**

- Eliminar recursos anteriores
  - k delete pod nginx-lh
  - k delete pvc nginx-volum-sc-lh
- LONGHORN: Crear postgres para prueba de pgbench
  - k apply -f 06-pgbech-longhorn.yaml
- Ejecutar pgbench
  - k exec -it postgres-<ID> -- bash
  - su postgres
  - pgbench -i -s 50 prueba
  - pgbench -c 4 -j 4 -t 1000 prueba
- Ejecutar dd
  - dd if=/dev/zero of=/var/lib/postgresql/test bs=64k count=16k conv=fdatasync
  - ls -lh /var/lib/postgresql/test

- Eliminar recursos anteriores
  - k delete -f 06-pgbech-longhorn.yaml
- NFS: Crear postgres para prueba de pgbench
  - k apply -f 07-pgbech-nfs.yaml
- Ejecutar pgbench
  - k exec -it postgres-<ID> -- bash
  - su postgres
  - pgbench -i -s 50 prueba
  - pgbench -c 4 -j 4 -t 1000 prueba
- Ejecutar dd
  - dd if=/dev/zero of=/var/lib/postgresql/test bs=64k count=16k conv=fdatasync
  - ls -lh /var/lib/postgresql/test

Agenda

- Introducción
- Objetos
- Selección de almacenamiento
- Práctica
- Benchmark
- **Hardware necesario**

Almacenamiento - Hardware necesario

Introducción:

- En esta sección no se profundizará en exceso, pero se proporcionarán pautas importantes.

Uso de CPU en Almacenamiento:

- El uso de CPU depende de factores como encriptación, compresión y cálculos de dispersión.
- Por lo que hemos trabajo en UNICEN no se realiza un consumo considerable.

Memoria y Caché:

- Los sistemas tienden a cachear datos en memoria para una mayor eficiencia.
- Más memoria permite más caché, pero exceso de memoria puede no generar un gran impacto.
- La memoria disponible influye en la velocidad de operaciones de lectura/escritura.

Almacenamiento - Hardware necesario

Espacio almacenamiento:

- Mayor capacidad de disco según necesidades.
- La replicación duplica el espacio requerido.
- Verificar cantidad de conectores de la placa madre y la necesidad de controladoras adicionales.

Velocidad de los Discos:

- Alinear la velocidad de disco con las demandas de la aplicación y el presupuesto.
- Protocolo de almacenamiento influye en la velocidad real aprovechada.
- RAID o discos más veloces (HDD, SSD, NVMe) como opciones.

Red y Ancho de Banda:

- Relación entre velocidad de red, velocidad de disco y protocolo usado.
- Tanto el acceso remoto como la replicación para alta disponibilidad necesitan de la red.
- Cálculos rápidos para dimensionar la red según velocidad de disco.
- Explorar opciones de bonding entre placas para maximizar la velocidad de red. Además provee redundancia.
- Opciones: UTP, fibra, InfiniBand.

Almacenamiento - Hardware necesario

Cálculos Rápidos:

SSD SATA Estándar:

- Velocidad de lectura/escritura de 500-550 MB/s.
- Ancho de banda de red necesario: 4,400 Mbps.

SSD NVMe de Alta Velocidad:

- Velocidad de lectura/escritura de hasta 3,500 MB/s.
- Ancho de banda de red necesario: 28,000 Mbps.

Disco Duro HDD:

- Velocidad de lectura/escritura de 100-150 MB/s.
- Ancho de banda de red necesario: 1,200 Mbps.

Deploy Aplicación Wordpress

Deploy de Worpress de prueba

Opción de Implementación

- Helm
- YAML

Despliegue Inicial

- Iniciar una aplicación WordPress sin almacenamiento persistente para comprender la estructura y la funcionalidad básica.

Integración de Almacenamiento Persistente:

- Solicitar un volumen mediante un PersistentVolumeClaim (PVC) y un StorageClass.

Ajuste de Configuración

- Abordar cómo cambiar la configuración para ajustar los límites de carga de archivos.

A R I U

Asociación Redes de Interconexión Universitaria