

ADMINISTRACIÓN DE SISTEMAS 2

Proyecto 1

Pablo Moreno Muñoz 841972

Identificador W->E

ÍNDICE

ÍNDICE	2
Resumen	2
Introducción y objetivos	3
Aplicaciones desplegadas	4
Problemas encontrados	6
Anexo	7

Resumen

El objetivo de esta práctica es realizar un análisis del despliegue de un cluster Kubernetes utilizando Vagrant sobre VMs de Virtualbox, con el fin de familiarizarse con el entorno y comprender su funcionamiento.

Se implementará un cluster Kubernetes (K3s) y se desplegarán diversos servicios y aplicaciones siguiendo la guía "Practical Introduction to Kubernetes".

Se estudiará el despliegue inicial de las VMs, la red, los servicios y Kubernetes mediante un análisis de los archivos Vagrantfile y provision.sh, además de documentar el proceso de aprendizaje.

Introducción y objetivos

La práctica tiene como objetivo principal realizar un análisis detallado del despliegue de un cluster Kubernetes utilizando herramientas como Vagrant y Kubernetes (K3s) sobre VMs de Virtualbox.

Objetivos

- El despliegue de servicios y aplicaciones sobre Kubernetes, comprendiendo conceptos clave y recursos asociados.
- Comprender las funcionalidades implementadas en los programas provision.sh y Vagrantfile, así como en abordar posibles problemas que puedan surgir durante el proceso de despliegue y encontrar soluciones para los mismos.

El despliegue de Kubernetes va a constar de 4 nodos, 3 workers y un master.

m	->	192.168.1.149
w1	->	192.168.1.141
w2	->	192.168.1.142
w3	->	192.168.1.143

El nodo master será el que realizará comandos e interactuará con el cluster de kubernetes, y sobre los nodos worker se irán desplegando las aplicaciones y servicios necesarios. En cada uno de ellos se instalará k3s como agente a través del script provision.sh pero en el master se realizará como server

Aplicaciones desplegadas

En esta apartado se comentará acerca de las distintas aplicaciones que se han desplegado y en el anexo se explicará la implementación

1 - Pods

Los Pods son la unidad básica de implementación en Kubernetes, que consta de uno o más contenedores. Comparten un espacio de nombres de red y de montaje y se programan en el mismo nodo. Se pueden ejecutar pods de forma individual o a través de un fichero .yaml

Run, para lanzar de forma manual, Apply para actualizar o lanzar con un fichero .yaml . Attach o Exec para abrir terminales del contenedor en ejecución.

2 - Labels

Las etiquetas se utilizan para organizar objetos de Kubernetes y constan de pares de clave-valor. Se utilizan para filtrar y agrupar objetos.

Label para etiquetar, -selector o -l para filtrar objetos por labels.

3 - Nodes

Los nodos son las máquinas físicas o virtuales que ejecutan aplicaciones en Kubernetes. Son la infraestructura subyacente del clúster.

Se puede especificar en qué máquina quieres que se ejecute un pod, etiquetando el Nodo y luego nombrando la etiqueta en el .yaml con nodeSelector.

4 - Logging

Kubernetes proporciona capacidades de registro para monitorear registros de contenedores. Esto ayuda en la depuración y resolución de problemas de aplicaciones que se ejecutan dentro del clúster.

Con logs -p te permite recuperar logs de contenedores o pods que hayan podido re-lanzarse o morir.

5 - API Server access

El servidor de API es un componente de control de Kubernetes. La API de Kubernetes sirve para poder acceder a través del navegador y por lo tanto permite acceder y manipular los datos a través de solicitudes http.

6 - Deployments

Los deployments proporcionan una manera de gestionar el ciclo de vida de los pods, permitiendo actualizaciones y reversiones.

Con rollout history puedes ver los cambios que ha recibido ese deployment y volver a versiones anteriores con `--undo -to-revision='version'`

7 - Services

En Kubernetes, los servicios actúan como una capa de abstracción para la comunicación entre diferentes conjuntos de pods, ofreciendo un punto de acceso estable para acceder a ellos. Como los pods tienen ips variables que van cambiando conforme se eliminan o no, los servicios se encuentran por encima de estos pods y tienen una ip fija y además se encargan de forwardear la información a los pods necesarios.

8 - Service Discovery

El Service Discovery en Kubernetes implica mecanismos dinámicos para encontrar y consumir servicios dentro del clúster, permitiendo que las aplicaciones localicen las IP o nombres DNS de los servicios.

Los recursos o servicios se pueden acceder por nombre dns de la siguiente forma : `'recurso'. [nombre del namespace].svc.cluster.local` .

9 - Port Forward

Permite el acceso local a un servicio o aplicación que se ejecuta dentro del clúster sin exponerlo externamente. Reenvía el tráfico desde un puerto específico en la máquina local del usuario a un puerto en un pod dentro del clúster.

10 - Namespaces

Los espacios de nombres en Kubernetes proporcionan una manera de dividir los recursos del clúster en subclústeres virtuales dentro del mismo clúster físico,

mejorando la organización y el aislamiento de recursos como pods y servicios. Se utilizan para el control de acceso y la gestión de recursos

11 - Volumnes

En Kubernetes, los volúmenes se utilizan para almacenar datos que necesitan persistir más allá del ciclo de vida de un pod. Permiten compartir datos entre contenedores en un pod, persistencia de datos y desacoplamiento del almacenamiento del ciclo de vida del contenedor.

Se pueden definir de diferentes tipos de uso: local, cloud o compartición en red

12 - Persistent Volumnes

Los volúmenes persistentes son volúmenes pero que no solo se comparten entre contenedores del mismo pod sino que además se comparten entre diferentes pods del mismo cluster.

Además para poder usar este PV primero necesitas reclamarlo usando un (PVC) Persisten aunque el pod ya no exista e incluso persisten aunque se elimine el deployment que lo usa.

Es de gran uso porque si se vuelve a lanzar el mismo deployment el PV se montaría correctamente.

13 - StatefulSet

Los StatefulSets en Kubernetes se utilizan para gestionar aplicaciones con estado. Están diseñados para aplicaciones que requieren identificadores de red estables y únicos y almacenamiento persistente estable. Ejemplos incluyen almacenes de clave-valor y bases de datos.

14 - Jobs

En Kubernetes, los jobs se utilizan para ejecutar pods hasta su finalización, lo que significa que se ejecutan una vez, completan sus tareas y luego se detienen. A menudo se utilizan para procesamiento por lotes o ejecución de tareas de corta duración.

Problemas encontrados

Un problema encontrado fue a la hora de la instalación del vagrant, debido a que la versión la cual se descarga por defecto en mi ordenador no permitía el paso de las claves para realizar la conexión ssh con la la VM, recibiendo el siguiente error:

“**pkeys are immutable on OpenSSL 3.0**” La solución encontrada tras investigar en distintos repositorios fue descargar una versión más actualizada de vagrant la cual

permitía el paso de estas pkeys a la máquina y poder realizar así una conexión ssh para proceder con la configuración de la misma.

Anexo

Pod

```
kubectl run sise --image=quay.io/openshiftlabs/simple-service:0.5.0 --port=9876  
kubectl get pods -o wide
```

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
sise      1/1     Running   0           5m57s  
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

Para conseguir una terminal dentro del propio pod

```
kubectl run -i --tty busybox1 --image=busybox -- sh  
#para conectarse una vez creada:  
#kubectl attach busybox1 -c busybox1 -i -t  
wget -q -O - 10.42.2.2:9876/info
```

```
Error from server (NotFound): pods "busybox1-7c8d571869-6p5wq" not found  
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl top pod  
NAME      CPU(cores)   MEMORY(bytes)  
busybox1   0m           0Mi  
sise       0m           12Mi  
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl attach busybox1 -c busybox1  
-i -t  
If you don't see a command prompt, try pressing enter.  
/#  
/#  
/# wget -q -O - 10.42.2.2:9876/info  
{"host": "10.42.2.2:9876", "version": "0.5.0", "from": "10.42.3.2"}/ #
```

Manifiestos

`kubectl apply -f`

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/pods/pod.yaml>

`kubectl exec twocontainers -c shell -i -t -- bash`

`curl -s localhost:9876/info`

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP            NODE   NOMINATED NODE   READINESS GATES
sise          1/1     Running   0           28m   10.42.2.3     w2     <none>            <none>
busybox1      1/1     Running   1 (17m ago)  19m   10.42.3.3     w3     <none>            <none>
twocontainers 2/2     Running   0           16m   10.42.2.4     w2     <none>            <none>
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl exec twocontainers -c shell
-i -t -- bash
[root@twocontainers /]# curl -s localhost:9876/info
{"host": "localhost:9876", "version": "0.5.0", "from": "127.0.0.1"}[root@twocontainers /]#
```

`kubectl delete pod twocontainers`

`kubectl delete pod sise`

`kubectl delete pod busybox1`

Labels

`kubectl apply -f`

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/labels/pod.yaml>

`kubectl get pods --show-labels`

`kubectl label pods labelex owner=michael`

`kubectl get pods --selector owner=michael`

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods --show-labels
NAME          READY   STATUS    RESTARTS   AGE   LABELS
labelex       1/1     Running   0           2m19s  env=development,owner=michael
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods --selector owner=michael
NAME          READY   STATUS    RESTARTS   AGE
labelex       1/1     Running   0           2m25s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

`kubectl apply -f`

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/labels/anotherpod.yaml>

`kubectl get pods -l 'env in (production, development)'`

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/labels/anotherpod.yaml
pod/labelexother created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -l 'env in (production, development)'
NAME          READY   STATUS    RESTARTS   AGE
labelex       1/1     Running   0           3m31s
labelexother  0/1     ContainerCreating   0           8s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

`kubectl delete pods -l 'env in (production, development)'`

Nodes

`kubectl top nodes`

`kubectl label nodes w1 shouldrun=here`

`kubectl apply -f`

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/nodes/pod.yaml>

kubectl get pods -o wide

```
error: all resources must be specified before label changes: node/crc-k2rc-master-0
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl label nodes w1 shouldrun=here
node/w1 labeled
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/nodes/pod.yaml
pod/onspecificnode created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -o wide
NAME                READY   STATUS             RESTARTS   AGE   IP        NODE   NOMINATED NODE   READINESS GATES
onspecificnode       0/1     ContainerCreating   0          16s   <none>    w1     <none>            <none>
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

kubectl describe node w1

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl describe node w1
Name:                w1
Roles:               <none>
Labels:              beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/instance-type=k3s
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=w1
                    kubernetes.io/os=linux
                    node.kubernetes.io/instance-type=k3s
                    shouldrun=here
Annotations:         alpha.kubernetes.io/provided-node-ip: 192.168.1.141
```

Logs

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/logging/gen/pod.yaml>

kubectl logs --tail=5 logme -c gen

#Para logear en stream :

kubectl logs -f --since=10s logme -c gen

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl logs --tail=5 logme -c gen
Tue Apr 2 11:08:49 UTC 2024
Tue Apr 2 11:08:50 UTC 2024
Tue Apr 2 11:08:50 UTC 2024
Tue Apr 2 11:08:51 UTC 2024
Tue Apr 2 11:08:51 UTC 2024
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl logs -f --since=10s logme -c gen
Tue Apr 2 11:08:52 UTC 2024
Tue Apr 2 11:08:52 UTC 2024
Tue Apr 2 11:08:53 UTC 2024
Tue Apr 2 11:08:53 UTC 2024
Tue Apr 2 11:08:54 UTC 2024
Tue Apr 2 11:08:54 UTC 2024
```

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/logging/oneshotpod.yaml>

#La opcion -p sirve para recuperar el log de instancias previas

kubectl logs -p oneshot -c gen

```
^C
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/logging/oneshotpod.yaml
pod/oneshot created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl logs -p oneshot -c gen
9
8
7
6
5
4
3
2
1
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

kubectl delete pod/logme pod/oneshot

Proxy

`kubect proxy --port=8080`

#Ahora desde otra terminal puedes hacerle queries

`curl http://localhost:8080/api/v1`



```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ curl http://localhost:8080/api/v1
{
  "kind": "APIResourceList",
  "groupVersion": "v1",
  "resources": [
    {
      "name": "bindings",
      "singularName": "",
      "namespaced": true,
      "kind": "Binding",
      "verbs": [
        "create"
      ]
    },
    {
      "name": "componentstatuses",
      "singularName": "",
      "namespaced": false,
      "kind": "ComponentStatus"
```

#Para ver las versiones recursos de la api

`kubect api-versions`

`kubect api-resources`

#Raw API

`kubect get --raw /api/v1`

Deployments

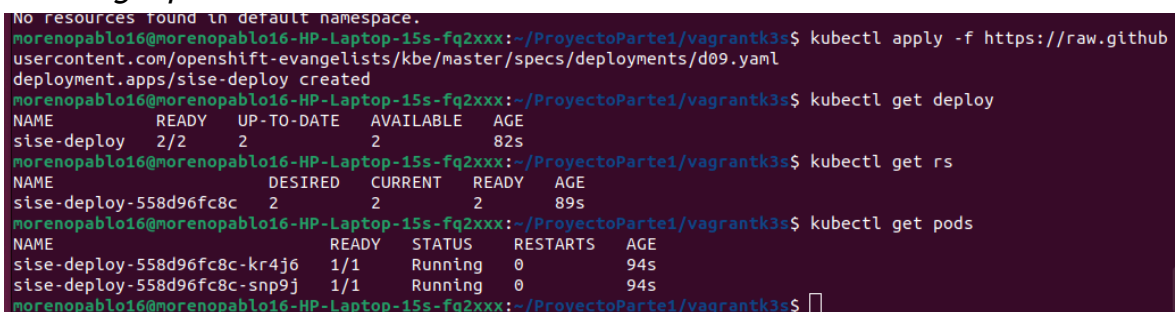
`kubect apply -f`

`https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/deployments/d09.yaml`

`kubect get deploy`

`kubect get rs`

`kubect get pods`



```
No resources found in default namespace.
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubect apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/deployments/d09.yaml
deployment.apps/sise-deploy created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubect get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
sise-deploy   2/2     2            2           82s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubect get rs
NAME          DESIRED   CURRENT   READY   AGE
sise-deploy-558d96fc8c   2         2         2       89s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubect get pods
NAME          READY   STATUS    RESTARTS   AGE
sise-deploy-558d96fc8c-kr4j6   1/1     Running   0          94s
sise-deploy-558d96fc8c-snp9j   1/1     Running   0          94s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

#Creamos una busy box para hacer wget y conseguir info

`kubect run -i --tty busybox1 --image=busybox -- sh`

`wget -q -O - 10.42.3.7:9876/info`

```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP            NODE   NOMINATED NODE   READINESS GATES
sise-deploy-558d96fc8c-kr4j6        1/1     Running   0           5m24s  10.42.2.9     w2     <none>            <none>
sise-deploy-558d96fc8c-snp9j        1/1     Running   0           5m24s  10.42.3.7     w3     <none>            <none>
busybox1                             1/1     Running   3 (32s ago)  2m52s  10.42.2.10    w2     <none>            <none>
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl attach busybox1 -c busybox1 -i -t
If you don't see a command prompt, try pressing enter.
/ #
/ #
/ # wget -q -O - 10.42.3.7:9876/info
{"host": "10.42.3.7:9876", "version": "0.9", "from": "10.42.2.10"}/ # wget -q -O - 10.42.2.9:9876/info
{"host": "10.42.2.9:9876", "version": "0.9", "from": "10.42.2.10"}/ #

```

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/deployments/d10.yaml>

kubectl get pods

kubectl get rs

```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP            NODE   NOMINATED NODE   READINESS GATES
sise-deploy-6b8d5d46cf-5qk7j        1/1     Running   0           26s    10.42.3.8     w3     <none>            <none>
sise-deploy-558d96fc8c-snp9j        1/1     Terminating   0           7m1s   10.42.3.7     w3     <none>            <none>
sise-deploy-6b8d5d46cf-jb7mf        1/1     Running   0           25s    10.42.2.11    w2     <none>            <none>
sise-deploy-558d96fc8c-kr4j6        1/1     Terminating   0           7m1s   10.42.2.9     w2     <none>            <none>
busybox1                             0/1     CrashLoopBackOff   3 (28s ago)  4m29s  10.42.2.10    w2     <none>            <none>
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get rs
NAME                                DESIRED   CURRENT   READY   AGE
sise-deploy-6b8d5d46cf              2          2          2        65s
sise-deploy-558d96fc8c              0          0          0       7m40s

```

#Ver el historial de los deployments

kubectl rollout history deploy/sise-deploy

kubectl rollout undo deploy/sise-deploy --to-revision=1

```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl rollout history deploy/sise-deploy
deployment.apps/sise-deploy
REVISION  CHANGE-CAUSE
1          <none>
2          <none>

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl rollout undo deploy/sise-deploy --to-revision=1
deployment.apps/sise-deploy rolled back
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl rollout history deploy/sise-deploy
deployment.apps/sise-deploy
REVISION  CHANGE-CAUSE
2          <none>
3          <none>

```

kubectl delete deploy sise-deploy

Services

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/services/rc.yaml>

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/services/svc.yaml>

kubectl get pods -l app=sise

kubectl describe pod rcsise-4v9xm | less

kubectl get svc

kubectl describe svc simpleservice | less

```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -l app=sise
NAME      READY   STATUS    RESTARTS   AGE
rcsise-4v9xm 1/1     Running   0           23s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl describe pod rcsise-4v9xm | less
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get svc
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes ClusterIP  10.43.0.1     <none>        443/TCP    27h
simpleservice ClusterIP  10.43.72.77   <none>        80/TCP     2m26s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl describe svc simpleservice | less
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE   NOMINATED NODE   READINESS GATES
busybox1  1/1     Running   4 (167m ago)  171m  10.42.2.10    w2     <none>            <none>
rcsise-4v9xm 1/1     Running   0           3m14s  10.42.3.10    w3     <none>            <none>

```

`kubectl attach busybox1 -c busybox1 -i -t`

`wget -q -O - 10.42.3.10:9876/info`

`wget -q -O - 10.43.72.77:80/info`

```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl attach busybox1 -c busybox1 -i -t
If you don't see a command prompt, try pressing enter.
/ #
/ #
/ # wget -q -O - 10.43.72.77:80/info
{"host": "10.43.72.77:80", "version": "0.5.0", "from": "10.42.2.10"}/ # wget -q -O - 10.42.3.10:9876/info
{"host": "10.42.3.10:9876", "version": "0.5.0", "from": "10.42.2.10"}/ #

```

`kubectl scale --replicas=2 rc/rcsise`

`kubectl get pods -l app=sise`

```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl scale --replicas=2 rc/rcsise
replicationcontroller/rcsise scaled
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods -l app=sise
NAME      READY   STATUS    RESTARTS   AGE
rcsise-4v9xm 1/1     Running   0           11m
rcsise-kwqwd 1/1     Running   0           7s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$

```

`kubectl delete svc simpleservice`

`kubectl delete rc rcsise`

`kubectl delete pod busybox1`

Service Discovery

#Creamos un servicio y un su RC supervisor

`kubectl apply -f`

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/sd/rc.yaml>

`kubectl apply -f`

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/sd/svc.yaml>

#A continuación creamos el jump pod en el mismo espacio de nombres (default)

`kubectl apply -f`

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/sd/jumpod.yaml>

`kubectl exec -it jumpod -c shell -- curl http://thesvc/info`

`##{"host": "thesvc", "version": "0.5.0", "from": "10.42.3.12"}`

#A continuación vamos a probar el dns creando unos pods como hemos

previamente pero en otro namespace y comprobar el correcto funcionamiento del jumpod

#Creamos el namespace "other"

kubectrl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/sd/other-ns.yaml>

#Creamos el servicio y el supervisor

kubectrl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/sd/other-rc.yaml>

kubectrl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/sd/other-svc.yaml>

#Ahora podemos consumir el servicio thesvc del namespace other

kubectrl exec -it jumpod -c shell -- curl http://thesvc.other/info

##{"host": "thesvc.other", "version": "0.5.0", "from": "10.42.3.12"}

kubectrl delete pods jumpod

kubectrl delete svc thesvc

kubectrl delete rc rcsise

kubectrl delete ns other

Port-forwarding

kubectrl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/pf/app.yaml>

kubectrl port-forward service/simpleservice 8080:80

#Una vez redirigido podemos abrir una nueva terminal y comprobar el funcionamiento

curl localhost:8080/info

##{"host": "localhost:8080", "version": "0.5.0", "from": "127.0.0.1"}

kubectrl delete service/simpleservice

kubectrl delete deployment sise-deploy

Namespaces

kubectl describe ns default

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl describe ns default
Name:         default
Labels:       kubernetes.io/metadata.name=default
Annotations:  <none>
Status:       Active

No resource quota.

No LimitRange resource.
```

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/ns/ns.yaml>

kubectl get ns

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/ns/ns.yaml
namespace/test created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get ns
NAME                STATUS    AGE
default             Active    28h
kube-system         Active    28h
kube-public         Active    28h
kube-node-lease     Active    28h
test                Active    15s
```

kubectl apply --namespace=test -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/ns/pod.yaml>

kubectl get pods --namespace=test

```
bash: https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/ns/pod.yaml: No existe el archivo o el directorio
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl apply --namespace=test -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/ns/pod.yaml
pod/podintest created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pods --namespace=test
NAME      READY   STATUS    RESTARTS   AGE
podintest 1/1     Running   0           6s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

kubectl delete ns test

Volumes

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/volumes/pod.yaml>

kubectl describe pod sharevol

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/volumes/pod.yaml
pod/sharevol created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl describe pod sharevol
Name:         sharevol
Namespace:    default
Priority:      0
Service Account: default
Node:         w3/192.168.1.143
Start Time:   Tue, 02 Apr 2024 17:25:54 +0200
Labels:       <none>
Annotations:  <none>
Status:       Running
IP:           10.42.3.16
```

#Ejecutamos uno de los contenedores en el pod y generamos algo de datos

kubectl exec -it sharevol -c c1 -- bash

[root@sharevol /] mount | grep xchange

#/dev/sda1 on /tmp/xchange type ext4 (rw,relatime,data=ordered)

[root@sharevol /] echo 'some data' > /tmp/xchange/data

#A continuación ejecutamos el otro pod y podemos comprobar que se han compartido los datos generado por el otro contenedor

kubectl exec -it sharevol -c c2 -- bash

[root@sharevol /] cat /tmp/data/data

```
Normal Created 5s kubelet Created container c2
Normal Started 5s kubelet Started container c2
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl exec -it sharevol -c c1 -- bash
[root@sharevol /]# mount | grep xchange
/dev/sda1 on /tmp/xchange type ext4 (rw,relatime,data=ordered)
[root@sharevol /]# echo 'some data' > /tmp/xchange/data
[root@sharevol /]# ^C
[root@sharevol /]# exit
exit
command terminated with exit code 130
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl exec -it sharevol -c c2 -- bash
[root@sharevol /]# cat /tmp/data/data
some data
[root@sharevol /]#
```

kubectl delete pod/sharevol

Persistent volume

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/pv/pv.yaml>

#Una vez creado lo necesitamos reclamar

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/pv/pvc.yaml>

kubectl get pvc

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl apply -f https://raw.githubusercontent.com/opensh
ift-evangelists/kbe/master/specs/pv/pv.yaml
persistentvolume/pv0001 created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl apply -f https://raw.githubusercontent.com/opensh
ift-evangelists/kbe/master/specs/pv/pvc.yaml
persistentvolumeclaim/myclaim created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl get pvc
NAME      STATUS  VOLUME  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
myclaim   Bound   pv0001  5Gi       RWO           local-path    6s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$
```

#Vamos a crear un deployment que usa el PVC

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/pv/deploy.yaml>

#Ahora queremos probar si los datos del volumen realmente persisten. Para ello

#encontramos el pod gestionado mediante el deployment anterior, ejecutamos en

#su contenedor principal y creamos un archivo en el

#Directorio /tmp/persistent (donde decidimos montar el PV):

kubectl get po

```
deployment.apps/pv-deploy created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl get po
NAME              READY  STATUS   RESTARTS  AGE
pv-deploy-d86794765-f5mzn  1/1    Running  0          31s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ sudo kubectl exec -it pv-deploy-d86794765-f5mzn -- bash
```

kubectl exec -it pv-deploy-d86794765-f5mzn -- bash

[root@pv-deploy-d86794765-f5mzn/] touch /tmp/persistent/data

[root@pv-deploy-d86794765-f5mzn /] ls /tmp/persistent/


```
exit
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl exec -it pv-deploy-d86794765-f5mzn -- bash
[root@pv-deploy-d86794765-f5mzn /]# touch /tmp/persistent/data
[root@pv-deploy-d86794765-f5mzn /]# ls /tmp/persistent/
data
[root@pv-deploy-d86794765-f5mzn /]#
```

*#Ahora borramos el pod y dejamos que el deployment genere otro pod nuevo.
#En teoría el PV volverá a estar disponible para el nuevo pod con los cambios
#realizados previamente.*

kubectl delete po pv-deploy-d86794765-f5mzn

kubectl get po

kubectl exec -it pv-deploy-d86794765-cbgkc -- bash

[root@pv-deploy-d86794765-cbgkc/] ls /tmp/persistent/

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get pvc
NAME      STATUS  VOLUME      CAPACITY  ACCESS MODES  STORAGECLASS  AGE
myclaim   Bound   pvc-ca45bb31-8597-4446-8bb3-a829c1f2f9bf  1Gi          RWO           local-path    4m42s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get po
NAME      READY  STATUS   RESTARTS  AGE
pv-deploy-d86794765-cbgkc  1/1    Running  0          2m1s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl exec -it pv-deploy-d86794765-cbgkc -- bash
[root@pv-deploy-d86794765-cbgkc /]# ls /tmp/persistent/
data
[root@pv-deploy-d86794765-cbgkc /]#
```

kubectl delete deployment pv-deploy

kubectl delete pvc myclaim

StatefulSet

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/mehdb/master/app.yaml>

kubectl get sts,po,pvc,svc

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl get sts,po,pvc,svc
NAME                                READY  AGE
statefulset.apps/mehdb              2/2    4m23s

NAME      READY  STATUS   RESTARTS  AGE
pod/mehdb-0  1/1    Running  0          4m23s
pod/mehdb-1  1/1    Running  0          2m12s

NAME                                STATUS  VOLUME      CAPACITY  ACCESS MODES  STORAGECLASS  AGE
persistentvolumeclaim/data-mehdb-0  Bound   pvc-f25258d6-1d0b-46bd-81e6-9e239b7cee6d  1Gi          RWO           local-path    4m23s
persistentvolumeclaim/data-mehdb-1  Bound   pvc-58cd577b-18a4-44fa-a3c9-a6147f846517  1Gi          RWO           local-path    2m12s

NAME      TYPE      CLUSTER-IP  EXTERNAL-IP  PORT(S)  AGE
service/kubernetes  ClusterIP  10.43.0.1   <none>        443/TCP  29h
service/mehdb      ClusterIP  None        <none>        9876/TCP  4m23s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

kubectl run -it --rm jumpod --restart=Never --image=quay.io/openshiftlabs/jump:0.2

-- curl mehdb:9876/status? level=full

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$ kubectl run -it --rm jumpod --restart=Never --image=quay.io/openshiftlabs/jump:0.2 -- curl mehdb:9876/status?level=full
0
pod "jumpod" deleted
```

kubectl delete sts mehdb

Jobs

#Este job supervisa un pod que cuenta de 9 a 1

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/jobs/job.yaml>

kubectl get jobs

kubectl get pods


```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl apply -f https://raw.githubusercontent.com/opensh
ift-evangelists/k8e/master/specs/jobs/job.yaml
job.batch/countdown created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl get jobs
NAME                COMPLETIONS   DURATION   AGE
countdown           1/1           1s         10s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
countdown-f8q75     0/1     Completed 0           15s

```

#Para más información

kubectl describe jobs/countdown

```

morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl describe jobs/countdown
Name:                countdown
Namespace:           default
Selector:             controller-uid=48096581-7ec8-4680-8be4-5cb810e3a5ed
Labels:               controller-uid=48096581-7ec8-4680-8be4-5cb810e3a5ed
                     job-name=countdown
Annotations:          batch.kubernetes.io/job-tracking:
Parallelism:         1
Completions:         1
Completion Mode:     NonIndexed
Start Time:          Tue, 02 Apr 2024 18:34:58 +0200
Completed At:        Tue, 02 Apr 2024 18:34:59 +0200
Duration:            1s
Pods Statuses:       0 Active / 1 Succeeded / 0 Failed
Pod Template:
  Labels:  controller-uid=48096581-7ec8-4680-8be4-5cb810e3a5ed
          job-name=countdown

```

#Aquí podemos ver que ha funcionado correctamente viendo los logs

kubectl logs countdown-wn5lw

```

countdown-f8q75     0/1     Completed 0           2m4s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl logs countdown-f8q75
9
8
7
6
5
4
3
2
1

```

kubectl delete job countdown

Health checks

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/main/specs/healthz/pod.yaml>

kubectl describe pod hc

```
IP: 10.42.2.21
Containers:
  hc:
    Container ID: containerd://a3f38a5298519b884e2cc6fe7669ec65f7b9765d3626f013f365ee09eb417582
    Image: quay.io/openshiftlabs/simple-service:0.5.0
    Image ID: quay.io/openshiftlabs/simple-service@sha256:72bfe1acc54829c306dd6683fe28089d222cf50a2df9
    Port: 9876/TCP
    Host Port: 0/TCP
    State: Running
      Started: Sat, 13 Apr 2024 16:47:58 +0200
    Ready: True
    Restart Count: 0
    Liveness: http-get http://:9876/health delay=2s timeout=1s period=5s #success=1 #failure=3
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-xqlq6 (ro)
Conditions:
  Type             Status
  PodReady         True
```

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/main/specs/healthz/badpod.yaml>

kubectl describe pod badpod

```
TokenExpirationSeconds: 3607
ConfigMapName: kube-root-ca.crt
ConfigMapOptional: <nil>
DownwardAPI: true
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age   From          Message
  ----     -
  Normal   Scheduled   8s    default-scheduler   Successfully assigned default/badpod to w3
  Normal   Pulled      8s    kubelet        Container image "quay.io/openshiftlabs/simple-service:0.5.0" already present on machine
  Normal   Created     8s    kubelet        Created container hc
  Normal   Started     8s    kubelet        Started container hc
  Warning  Unhealthy   2s    kubelet        Liveness probe failed: Get "http://10.42.3.28:9876/health": context deadline exceeded (Client.Timeout
t exceeded while awaiting headers)
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

kubectl get pods

```
NAME     READY   STATUS    RESTARTS   AGE
hc       1/1     Running   0           3m46s
badpod   1/1     Running   3 (44s ago)  3m1s
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3s$
```

kubectl apply -f

<https://raw.githubusercontent.com/openshift-evangelists/kbe/main/specs/healthz/ready.yaml>

kubectl delete pod/hc pod/ready pod/badpod

Env variables

kubectl apply -f

<https://github.com/openshift-evangelists/kbe/raw/main/specs/envs/pod.yaml>

#Este pod tiene como environment variable SIMPLE_SERVICE_VERSION = 1.0

#La salida refleja el valor que se estableció el valor en la variable de entorno (el

#valor predeterminado, a menos que la variable lo anule, es 0.5.0):

#kubectl exec envs -t -- curl -s 127.0.0.1:9876/info

```
pod "badpod" deleted
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl apply -f https://github.com/openshift-evangelists/kbe/raw/main/specs/envs/pod.yaml
pod/envs created
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl exec envs -t -- curl -s 127.0.0.1:9876/info
{"host": "127.0.0.1:9876", "version": "1.0", "from": "127.0.0.1"}morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$
```

#Para ver las variables de entorno podemos ejecutar el siguiente comando

kubectl exec envs -t -- curl -s 127.0.0.1:9876/env

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl exec envs -t -- curl -s 127.0.0.1:9876/env
{"host": "127.0.0.1:9876", "version": "1.0", "from": "127.0.0.1"}morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl exec envs -t -- curl -s 127.0.0.1:9876/env
{"version": "1.0", "env": {"LANG": "C.UTF-8", "KUBERNETES_PORT_443_TCP_PROTO": "tcp", "GPG_KEY": "C01E1CAD5EA2C4F0B8E3571504C367C218ADD4FF", "PYTHON_VERSION": "2.7.13", "PYTHON_PIP_VERSION": "9.0.1", "KUBERNETES_SERVICE_HOST": "10.43.0.1", "HOSTNAME": "envs", "KUBERNETES_SERVICE_PORT_HTTPS": "443", "REFRESHED_AT": "2017-04-24T13:50", "KUBERNETES_PORT_443_TCP_ADDR": "10.43.0.1", "KUBERNETES_PORT": "tcp://10.43.0.1:443", "PATH": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", "KUBERNETES_PORT_443_TCP_PORT": "443", "KUBERNETES_PORT_443_TCP": "tcp://10.43.0.1:443", "HOME": "/root", "KUBERNETES_SERVICE_PORT": "443", "SIMPLE_SERVICE_VERSION": "1.0"}}morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$
```

#Alternativamente, también puedes usar el comando para mostrar las variables de entorno dentro del pod:

kubectl exec envs -- printenv

```
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$ kubectl exec envs -- printenv
PATH=/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=envs
LANG=C.UTF-8
GPG_KEY=C01E1CAD5EA2C4F0B8E3571504C367C218ADD4FF
PYTHON_VERSION=2.7.13
PYTHON_PIP_VERSION=9.0.1
REFRESHED_AT=2017-04-24T13:50
SIMPLE_SERVICE_VERSION=1.0
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_ADDR=10.43.0.1
KUBERNETES_SERVICE_HOST=10.43.0.1
KUBERNETES_SERVICE_PORT=443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT=tcp://10.43.0.1:443
KUBERNETES_PORT_443_TCP=tcp://10.43.0.1:443
HOME=/root
morenopablo16@morenopablo16-HP-Laptop-15s-fq2xxx:~/ProyectoParte1/vagrantk3$
```

kubectl delete pod/envs

Anexo II

Vagrantfile

```
#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 además de definirles la
ip de su master
NODES = [
  { hostname: 'm', type: "master", ip: MASTER, mem: 1000, m: MASTER },
  { hostname: 'w1', type: "worker", ip: '192.168.1.141', mem: 1000, m: MASTER
},
  { hostname: 'w2', type: "worker", ip: '192.168.1.142', mem: 1000, m: MASTER
},
  { hostname: 'w3', type: "worker", ip: '192.168.1.143', mem: 1000, 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
      # Configuración de recursos de la máquina virtual
      nodeconfig.vm.provider "virtualbox" do |v| #Usamos virtualbox como
proveedor
        v.customize
["modifyvm", :id, "--memory", node[:mem], "--cpus", "1"] #Asignación de memoria y
CPU
        v.default_nic_type = "virtio" #Tipo de interfaz de red virtual

        #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
=end

    end
    #Ahora se bootea la maquina recién 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: 'provision.sh',
        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
end
end

```

Provision.sh

```
#!/bin/bash -x
#Establece las variables de entorno que se pasan por parámetro
HOSTNAME=$1
NODEIP=$2
MASTERIP=$3
NODETYPE=$4

timedatectl set-timezone Europe/Madrid

cd /vagrant

#Introduce en el /etc/hostname el nombre del nodo (HOSTNAME) que se pasa por parámetro
echo $1 > /etc/hostname
hostname $1

#Introduce en el /etc/hosts la IP y el nombre de los nodos que se pasan por parámetro
{
    echo "192.168.1.141 w1"
    echo "192.168.1.142 w2"
    echo "192.168.1.143 w3"
    cat /etc/hosts
} > /etc/hosts.new
mv /etc/hosts{.new,}

#Se copia el ejecutable k3s de kubernetes en /usr/local/bin
cp k3s /usr/local/bin/

# A continuación se realiza la instalación de k3s en los nodos, es diferente la instalación
en el nodo master y en los nodos worker
# En el nodo master se instala el servidor de k3s el cual se especifica en el parámetro
install.sh
# y en los nodos worker se instala el agente de k3s.
# En caso del máster se especifica el token que se ha generado en el nodo master y se
especifica la interfaz de red que se va a utilizar para la red flannel, la dirección IP del
nodo, el nombre del nodo y se deshabilita el traefik y el servicelb.
if [ $NODETYPE = "master" ]; then
    INSTALL_K3S_SKIP_DOWNLOAD=true \
    ./install.sh server \
    --token "wCdCl6AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2dloDJQQOsBkIwy5OZ/5" \
    --flannel-iface enp0s8 \
    --bind-address $NODEIP \
    --node-ip $NODEIP --node-name $HOSTNAME \
    --disable traefik \
    --disable servicelb \
    --node-taint k3s-controlplane=true:NoExecute
    # El parámetro `--node-taint` se utiliza para aplicar una restricción a un nodo en el
    clúster de k3s.
    # En este caso, se está aplicando la restricción `k3s-controlplane=true:NoExecute` al
    nodo.
    # esta restricción indica que este nodo es parte del plano de control de k3s
```

y que no se deben programar pods regulares en él. Los pods regulares solo se programarán en los nodos que no tengan esta restricción. La opción `NoExecute` en la restricción significa que si un pod ya está en ejecución en este nodo y la restricción se aplica posteriormente, el pod no se eliminará automáticamente, pero no se programarán nuevos pods en este nodo.

```
--advertise-address $NODEIP
--cluster-domain "cluster.local"
--cluster-dns "10.43.0.10"
```

```
#Copia el fichero de configuración yaml de k3s en el directorio /vagrant
cp /etc/rancher/k3s/k3s.yaml /vagrant
```

```
else
```

```
# En caso de los nodos worker se instala el agente de k3s,
# se especifica la dirección IP del nodo maestro y el puerto, y el token que se ha
generado en el nodo master,
# la ip del nodo, el nombre del nodo y la interfaz de red que se va a utilizar para la red
flannel.
```

```
INSTALL_K3S_SKIP_DOWNLOAD=true \
./install.sh agent --server https://{MASTERIP}:6443 \
--token "wCdCl6AlP8qpqqI53DM6ujtrfZ7qsEM7PHLxD+Sw+RNK2dloDJQQOsBkIwy5OZ/5" \
--node-ip $NODEIP --node-name $HOSTNAME --flannel-iface enp0s8
```

```
fi
```