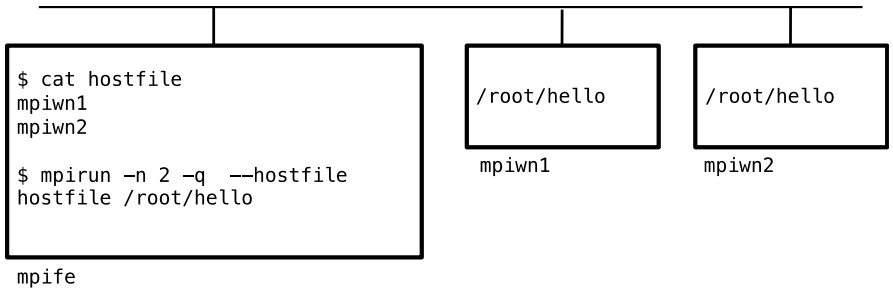
Plataformas de Gestión de Contenedores

Máster Universitario en Computación en la Nube y de Altas Prestaciones
Departamento de Sistemas Informáticos y Computación (DSIC)
2022-23 – Examen Práctico – 40% de la calificación total
Duración 2:30h

Nombre: ___Javier Meliá Sevilla_____

Queremos desplegar un clúster MPI sobre contenedores en Docker y Kubernetes, de forma que desde un contenedor podamos lanzar un programa MPI y que cada uno de los procesos MPI se ejecute en un contenedor diferente, según el esquema siguiente:



Para ello, se deben realizar los siguientes pasos:

1) Crear las Imágenes de Docker apropiadas.
2) Ejecutar una prueba con varios contenedores Docker en local.
3) Crear los recursos Kubernetes apropiados para ejecutarlo.

1. Crear las imágenes de Docker (25%)

Para instalar MPI, es necesario: seguir los siguientes pasos

- Necesitamos una imagen de Docker con un servidor sshd instalado (p.e. la imagen rastasheep/ubuntu-sshd:18.04). El servidor `sshd` se puede arrancar mediante el comando `/usr/sbin/sshd` utilizando la opción `-D` para que funcione en primer plano si fuera necesario), y configurar la conexión ssh sin contraseña con un par de claves, según los pasos siguientes (desde dentro del contenedor):
  - `ssh-keygen -q -N '' -f ~/.ssh/id_rsa`
  - `cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys`
  - Comprobar: `ssh -i /root/.ssh/id_rsa localhost ls -l /`
- Después deberemos Instalar MPI en el contenedor con los siguientes comandos
  - `apt-get update`
  - `apt-get install -y openmpi-bin libopenmpi-dev`
- Finalmente registraremos la imagen del contenedor modificada como una imagen propia y la subiremos a Docker Hub

Referencia de la imagen modificada en Docker Hub The push refers to repository [docker.io/meliatus80/examen]
Contenido de la clave privada utilizada (`/root/.ssh/id_rsa`): -----BEGIN RSA PRIVATE KEY----- MIIEpAIBAAKCAQEAomhB3lR2x7kR6AVfTvSdlz2NPLmztWGpRn7eQKrPQO1sclgi zj0uB74iFVCqnjmv+guBhiDpRHPbsbn4WOhvrqt6/9iMihC+Jz01L22g5dS+vZXP dfRddZUJiBzjltl6PuQd3bqRLUTnWHX9Uu7NVjt7SBn4EGeS23w5IryHv904pWXu

+srCF3iq/VDdVBgFzZSOhPOWe6jPPEoU6Y6yuWtj0yADpSimEIhTwtgwGQWNgo97
C8z4+760DNqWGEOl/VoO9+kPfv6NMqb92w5S18+hnbnRWKnxsaGAhsvg+PfZmDxv
3IyZq3GWBiDgSeDQRyC3YPoB9S6zL2DqFPbaFQIDAQABAoIBABfuVu5k8c/CRJI0
4HNo9iPIDRzKnGF4z5G0thk/UiTWF2nN9OsA9txrNuh/J3UDORwWP/K8vMK6c7//
kEnBjltlp0014pehmfN8QG8Jr3DOZYDYJ2bRKEhgOrRU1XBwAWRtBr2adTP/eDt/
v2FOEPHI+ASK1xPLv0caSe0FbNdnjIoxQU1sPB8fj1XicyxK3v1R6/sWmcWPLvLG
3R6D/o/w4V1IB5RGT24fwOx2P+Bsw2kuwlsQnS9vEQ3tp2fhLQfjjeZY1yoq4MXH
Jq680ZdrRGrRBbDDTOmHR0FkoASAqhqlqnL8y94GsskXGn6zSaRslueRuqO9WBGk
HMBrlG0CgYEA0PQ1dNmY8p08h46zzq8/oNDAHuIzCbLjOFFVEsRu9QSItwUWoYjQ
w86ySHDIwzOkhIN0Hss/ttBpXsEW8EzVhw4F//sujfVBd1g+NeFMri46OtpzF3q5
dVp6/DBwwhK/w8jQH1iC2mk/XDstmLKFRwH5B/GE9AfJw3lKI3gG27cCgYEAxvkq
3tjPXftX7zk6wmIzJ50UqPmbi/VPAAcxxTd3I9k3ycZxzpQddxhe6QnRYnyLX/LG
iv4e0HF4XZElkGM08VZT7n1OLPJem6DHuaMVlt8OeTzPQwgW4TZ1oFb+zfaOpfxj
kFv53zT6xfq5E4hAx366HSBjypTTxtWaiwGO0JMCgYBtSIv8xSF9IkdaDoR0AKwi
i4svm9ZdqDW2arM30YzGkMyADCU97rRg9GXCBVq63POaarX8tQhpsK6rZAl/PHAg
5nC05MYAcNkAdiJzQ5Y7Muo5ha7UBNeiTOQtqZgPqIaq+wf+nOgzGwCMWFcMmbGi
5aE9Sk/i4X/gwDrKOig8GQKBgQC/05LYO7BYTCxvzv3uT5w/0boMWnDy0vJx9X2v
RBURs6aMWkx4O+XYXt+JzlRWnfoE69ssP2jXzr2bf4b2mB5/mZnbcGUBAj3hznUf
bLIwvSWIDu9HEbJ1N/rZNWX8jXrjXdtjXrkcz0gBvQHXaeICMxQdZo+uzCMMK/0N
jNfyDwKBgQCvniVG2xbHacpqetAOemqde4TFyZ5IUVmT9Sx8IDINqQF3dCBhODLf
+plDO8dlSTHi774JfyHXh4gHNyt2sEUthu/f/TAKfDyBOLzuJ+otuwI63rHxdHHx
EKx9BtXVvn0fNZbO0aTI/dolzICj51afQcQzbr7poHqWTtbTiOhMEQ==
-----END RSA PRIVATE KEY-----

2. Ejecutar una prueba con varios contenedores Docker en local (30%)

Una vez disponemos de un contenedor Docker preparado para ejecutar MPI, procedemos a hacer una prueba ejecutando directamente tres contenedores en la máquina local y conectándolos de forma apropiada. Para ello, habrá que hacer los siguientes pasos:

- Desplegar un contenedor bajo el nombre "mpife". Entrar en él, compilar el programa siguiente mediante el comando mpicc -o hello hello.c. Si necesitas un editor, lo puedes instalar cono `apt-get install -y vim`.

```
#include <mpi.h>
#include <stdio.h>
#include <unistd.h>
int main(int argc, char **argv) {
  int rank;
  char hostname[256];
  MPI_Init(&argc,&argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  gethostname(hostname,255);
  printf("Hello world!  I am process number: %d on host %s\n",
        rank, hostname);
  MPI_Finalize();
  return 0;
}
```

- Ejecutar el programa mediante `mpirun -n 2 --allow-run-as-root -q /root/hello`. Anotar el resultado en el cuadro que se indica (revisa que el directorio y el nombre del programa "hello" coincidan en tu caso).

```
root@8bfea7ef22ec:~# mpirun -n 4 --allow-run-as-root -q hello
Hello world! I am process number: 0 on host 8bfea7ef22ec
Hello world! I am process number: 1 on host 8bfea7ef22ec
Hello world! I am process number: 2 on host 8bfea7ef22ec
Hello world! I am process number: 3 on host 8bfea7ef22ec
```

Para poder ejecutar en paralelo, deberemos:

- Ejecutar dos contenedores más con los nombres `mpiwn1` y `mpiwn2`.
- Crear un fichero de "`hostfile`" en el contenedor `mpife` con las IPs locales de los contenedores `mpiwn1` y `mpiwn2`. Recuerda que las IPs se pueden obtener ejecutando el comando `ip addr` dentro de un contenedor. Si este comando no está disponible, se puede instalar con `apt-get install -y iproute2`. También las puedes obtener sin instalar ningún paquete inspeccionando la red correspondiente.
- Copiar el ejecutable en ambos contenedores (p.e. mediante `scp` o mediante `docker cp`) en la misma ubicación que en el contenedor `mpife`.
- Comprobar que desde el contenedor `mpife` se puede acceder sin contraseña a los contenedores `mpiwn1` y `mpiwn2` (necesario para que se incluyan sus IPs en el fichero `known_hosts`).
- Ejecutar mediante el comando `mpirun -n 8 -q --allow-run-as-root --machinefile hostfile /root/hello`.

Rellenar la siguiente información:

Comandos para la copia del fichero en los contenedores docker cp mpife:/root/hello hello → esto para copiar el fichero al host docker cp hello mpiwn1:/root/hello docker cp hello mpiwn2:/root/hello → esto para copiar el fichero de antes a los anteriores	
Contenido de hostfile 172.17.0.7 mpiwn1 172.17.0.8 mpiwn2	Resultado de la ejecución Hello world! I am process number: 1 on host 6f69a1fdce9a Hello world! I am process number: 3 on host 6f69a1fdce9a Hello world! I am process number: 0 on hst 6f69a1fdce9a Hello world! I am process number: 2 on host 6f69a1fdce9a Hello world! I am process number: 6 on host 420a8b83a40d Hello world! I am process number: 5 on host 420a8b83a40d Hello world! I am process number: 7 on host 420a8b83a40d Hello world! I am process number: 4 on host 420a8b83a40d

3. Crear los recursos Kubernetes apropiados para ejecutarlo (30%)

En este último apartado crearemos los objetos apropiados para desplegar el clúster en un entorno Kubernetes. Por motivos de tiempo, haremos las siguientes simplificaciones:

- La configuración del fichero hostfile se hará a posteriori, una vez desplegado el clúster.
- El lanzamiento de proceso implica que el ejecutable esté en todos los contenedores, para lo que se realizará manualmente la copia del fichero en todos ellos.

Indicar en el siguiente cuadro el (o los) ficheros YAML necesarios para el despliegue:

Necesitaremos primer un pod que haga de front-end:

```
apiVersion: v1
kind: Pod
metadata:
  name: mpife
spec:
  containers:
  - name: mipfe
    image: meliatus80/examen:alumno
    command: [ "usr/bin/tail" ]
    args: [ '-f', '/dev/null' ]
```

Luego necesitaremos un deployment que nos cree dos replicas:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mpi-deployment
spec:
  selector:
    matchLabels:
      app: mpi-aplicacion
  replicas: 2
  template:
```

```
  metadata:
    labels:
      app: mpi-aplicacion
  spec:
    containers:
    - name: mpiwn
      image: meliatus80/examen:alumno
```

Desplegar el cluster, entrar en el contenedor elegido como front-end, compilar y ejecutar el programa como se ha descrito en el paso anterior

Comandos para el lanzamiento de la aplicación en Kubernetes

```
kubectl apply -f pod.yaml
kubectl apply -f deployment.yaml
kubectl cp hello  mpi-deployment-78f7b7fdbc-r297r:/root
kubectl cp hello  mpi-deployment-78f7b7fdbc-s78cb:/root
kubectl cp hello  mpife:/root
kubectl exec -it mpife /bin/bash
vim hostfile
>> mpirun -n 8 -q --allow-run-as-root --machinefile hostfile /root/hello
```

Contenido de hostfile	Resultado de la ejecución
172.17.0.10 172.17.0.11	Hello world! I am process number: 2 on host mpi-deployment-78f7b7fdbc-r297r

	Hello world! I am process number: 3 on host mpi-deployment-78f7b7fdbc-r297r Hello world! I am process number: 6 on host mpi-deployment-78f7b7fdbc-r297r Hello world! I am process number: 5 on host mpi-deployment-78f7b7fdbc-r297r Hello world! I am process number: 0 on host mpi-deployment-78f7b7fdbc-s78cb Hello world! I am process number: 4 on host mpi-deployment-78f7b7fdbc-s78cb Hello world! I am process number: 7 on host mpi-deployment-78f7b7fdbc-s78cb Hello world! I am process number: 1 on host mpi-deployment-78f7b7fdbc-s78cb

4. Mejora de la configuración y acceso a los ejecutables en Kubernetes (15%)

En el apartado anterior se han introducido dos simplificaciones para facilitar su implementación:

- Rellenar el fichero hostfile a posteriori una vez desplegado el clúster.
- Copia manual del ejecutable en todos los contenedores.

Indica (sin que haga falta implementarlo) que soluciones te parecen adecuadas para que no haga falta copiar manualmente el ejecutable y para automatizar la creación del fichero hostfile. Indica los objetos Kubernetes que utilizarías, el orden de creación y qué acciones serían necesarias.

Para rellenar el fichero hostfile a posteriori una vez desplegado el cluster se podrá ejecutar un configmap para compartir la información del ejectuable mas menos la acción necesaria seria la siguiente:

“kubectl create configmap hello --from-file=hello”

Y después en el pod.yaml se le añadiría el configmap como un fichero.

Para la Copia manual del ejecutable en todos los contenedores se utilizaría los volúmenes y los volúmenes persistentes para almacenar y compartir el ejecutable entre los diferentes pods que tendríamos.