| TSR | Recuperación | Práctica 3 | 2019-01-25 | A |
|---|---|---|---|---|

*Esta prueba tiene un valor de 1 punto, y consta de 6 cuestiones tipo test. Cada cuestión plantea 4 alternativas y tiene una única respuesta correcta. Cada respuesta correcta aporta 1/6 puntos, y cada error descuenta 1/18 puntos. Debe contestar en la hoja de respuestas.*

**1** *Con el objetivo de permitir el acceso por parte de clientes externos a la aplicación que desplegamos, en prácticas hemos abordado el problema de cómo conectar clientes a nuestra aplicación:*

**a** A1: Para ello, hemos incluido la sentencia ports en el fichero de configuración de docker

**b** A2: Para ello, hemos utilizado la dirección IP del anfitrión en la orden para lanzar los clientes externos.

**c** A3: Las afirmaciones A1 y A2 son ciertas.

**d** A4: Ninguna de las restantes afirmaciones es cierta.

**2** *Sobre la componente worcli:*

**a** Se trata de un worker idéntico a los demás worker, con la única diferencia que lo ejecutamos de forma externa, igual que el cliente externo.

**b** Se trata de una componente que podemos desplegar junto a un broker y otros worker.

**c** Se trata de un cliente externo respecto a los demás workers

**d** Debe configurarse indicando la dirección IP del cliente externo.

**3** *Al tratar de conectar un cliente externo a nuestra aplicación 'cbw':*

**a** Si el cliente no se encuentra en la misma VPN que el anfitrión, el cliente no logrará conectar.

**b** Si el anfitrión no dispone de un cortafuegos, o dispone de cortafuegos con todos los puertos abiertos, el cliente no podrá conectar.

**c** Este cliente deberá obligatoriamente ejecutarse en el mismo ordenador que la aplicación, pues forma parte de ella.

**d** Deberemos configurar el cliente externo en el fichero de configuración de docker-compose

**4** *Suponiendo que disponemos del siguiente fichero Dockerfile, si ejecutamos la orden* `docker build -t myapp .`

```
FROM centos:7.4.1708
RUN curl --silent --location
https://rpm.nodesource.com/setup_8.x | bash -
RUN yum install -y nodejs
RUN yum install -y epel-release
RUN yum install -y zeromq-devel make python
gcc-c++
RUN npm install zmq
```

**a** Habremos creado nuestra aplicación distribuida, formada por un broker y varios workers.

**b** Será necesario que ejecutemos dicha orden en el mismo directorio donde se encuentre el Dockerfile anterior, pues el . que observamos en la orden docker build precisamente indica este detalle.

**c** Al ejecutar la orden tendremos un fallo, pues el argumento myapp no será reconocido por la orden docker.

**d** Se ejecutará prácticamente al instante, pues toda la información necesaria para crear la imagen docker estará seguro en el ordenador local donde ejecutamos la orden.

**5** *Considere worcli, que conecta dos brokers*

**a** Se despliega como cliente externo para uno de los brokers

**b** Su despliegue se especifica en el docker-compose.yml de ambos brokers

**c** Se despliega de forma independiente a los dos brokers

**d** Se despliega como worker externo a uno de los brokers

**6** *Consideremos como punto de partida el sistema 4_CBW_TFCL, y el docker-compose.yml proporcionado. Si deseamos permitir a worcli añadir logs en el logger*

**a** Necesitamos añadir una sección link a la descripción para el servicio worcli

**b** No necesitamos modificar nada en el docker-compose.yml, únicamente el código en worcli

**c** Necesitamos eliminar la descripción para worcli del fichero docker-compose.yml

**d** Ninguna de las restantes afirmaciones es cierta