TSR	Segundo Parcial	Temas 3,4,5,6	2020-01-09	A

Esta prueba tiene un valor de 4 puntos, y consta de 32 cuestiones tipo test. Cada cuestión plantea 4 alternativas y tiene una única respuesta correcta. Cada respuesta correcta aporta 1/8 puntos, y cada error descuenta 1/24 puntos. Debe contestar en la hoja de respuestas.

1 *Supongamos que se ha desarrollado cierta aplicación basada en tres componentes: cliente, broker y trabajador.*

En ella, el cliente utiliza un socket REQ y el trabajador un socket REP.

Sabiendo que el broker no conoce las identidades de los trabajadores, el broker deberá utilizar estos sockets:

a ROUTER para interactuar con clientes y DEALER para interactuar con trabajadores.

b DEALER para interactuar con clientes y ROUTER para interactuar con trabajadores.

c ROUTER para interactuar con clientes y ROUTER para interactuar con trabajadores.

d Todas las demás opciones son válidas.

2 *Se ha desarrollado un programa servidor que utiliza un socket ROUTER para interactuar directamente con múltiples procesos clientes.*

Los clientes no conocen la identidad de dicho servidor, y tras la petición esperan una respuesta.

El programa cliente puede utilizar este tipo de socket:

a REQ.

b ROUTER.

c PUSH.

d Todas las demás opciones son válidas.

3 *Un programa cliente envía un mensaje a un servidor. El mensaje consta de un único segmento, se ha enviado a través de un socket DEALER, y se ha recibido a través de un socket ROUTER.*

¿Cuántos segmentos contiene ese mensaje cuando es procesado por el callback?:

a Ninguno. De hecho, el mensaje no podrá recibirse.

b Uno.

c Dos.

d Depende del transporte utilizado (TCP, IPC…).

4 *En un despliegue de una aplicación distribuida basada en múltiples componentes, ¿cuál de los siguientes mandatos gestiona el orden en que arrancan los componentes?*

a `docker build`

b `docker run`

c `docker commit`

d `docker-compose up`

5 *Si quisiéramos desplegar una aplicación distribuida compuesta por varios componentes en múltiples contenedores ubicados en más de un ordenador, necesitaríamos una herramienta como ésta:*

a `node`

b `kubernetes`

c `git`

d `docker-compose`

6 *El descriptor de despliegue utilizado en Docker para realizar un despliegue de varios componentes sobre un solo anfitrión se llama:*

- **a** Dockerfile
- **b** package.json
- **c** docker-compose.yml
- **d** deployment.js

7 *Respecto a contenedores y máquinas virtuales, ¿Cuál de estas afirmaciones es cierta?*

- **a** Una máquina virtual requiere menos recursos que un contenedor equivalente
- **b** No puedo tener más de una máquina virtual sobre una misma máquina anfitrión
- **c** El anfitrión de un contenedor no puede ser una máquina virtual
- **d** Los contenedores no necesitan mantener y ejecutar un sistema operativo huésped completo.

8 *Si la orden* `docker run mi-nombre` *no genera ningún error, entonces podemos afirmar que* `mi-nombre`*:*

- **a** Es un identificador de contenedor y ese contenedor existe en la máquina local.
- **b** Era un identificador válido de contenedor y ese contenedor ha sido eliminado al ejecutar esa orden.
- **c** Es un nombre de imagen y esa imagen existe en el depósito local o en el Docker Hub.
- **d** Era un nombre de imagen Docker válido y ese nombre ha sido eliminado al ejecutar la orden.

9 *Si se ha generado una imagen a partir de este Dockerfile mediante la orden:* `docker build -t my-worker .`

```
FROM ubuntu-zmq
COPY ./worker.js /worker.js
CMD node worker $BROKER_PORT
```

¿Cuál de las siguientes afirmaciones es FALSA?

- **a** La imagen `my-worker` toma como base la imagen `ubuntu-zmq`.
- **b** En el directorio en el que estaba este `Dockerfile` también había un fichero `worker.js`.
- **c** La imagen `my-worker` no puede utilizarse en el `FROM` de otros `Dockerfile`.
- **d** La orden `docker build` mencionada en esta cuestión podría fallar si la imagen `ubuntu-zmq` no se encontrase en el depósito local.

10 *Sobre la relación entre programa y servicio:*

- **a** Un servicio está formado por un conjunto de programas y las dependencias que existen entre ellos.
- **b** Todo servicio requiere el despliegue de programas.
- **c** Servicio + SLA = programa.
- **d** Todas las opciones mencionadas son válidas.

11 *Sean dos componentes, A y B, de un mismo servicio a desplegar.*

Para crear la imagen `compa` *se ha utilizado el siguiente Dockerfile*

```
FROM imageBase
COPY ./compa.js compa.js
EXPOSE 8200
CMD node compa
```

Para crear la imagen `compb` *se ha utilizado el siguiente Dockerfile*

```
FROM imageBase
COPY ./compb.js compb.js
CMD node compb $URL
```

Entonces un fragmento de docker-compose.yml válido para su despliegue es:

a
```
services:
      coa:
         image: compa
         endpoint:
           - URL=tcp://localhost:8200
      cob:
         image: compb
         links:
            - coa
```

b
```
services:
      coa:
         image: compa
         expose:
            - '8200'
      cob:
         image: compb
         links:
            - URL=tcp://coa:8200
```

c
```
services:
      coa:
         image: compa
         expose:
            - '8200'
      cob:
         image: compb
         links:
            - coa
         environment:
            - URL=tcp://coa:8200
```

d
```
services:
      coa:
         image: compa
         environment:
            - URL=tcp://cob:8200
      cob:
         image: compb
         url:
            - '8200'
```

12 *Relación entre defectos, errores y fallos:*

a Todo defecto se convierte en error

b Podrá haber fallos que no estén causados por ningún defecto.

c Si no existe redundancia en el componente que sufre un error, dicho error genera un fallo.

d Con un mecanismo de diagnóstico adecuado y un tratamiento correcto se puede evitar que las situaciones de fallo provoquen errores.

13 *Con el modelo activo de replicación…:*

a Ninguna réplica propaga sus resultados a las restantes réplicas

b El rendimiento crece de forma lineal con el número de nodos.

c No existe riesgo de inconsistencia cuando se ejecutan operaciones no deterministas.

d No se necesita ningún tipo de sincronización entre las réplicas para obtener una consistencia secuencial entre ellas.

14 *Objetivos generales de la replicación:*

a Aumentar la disponibilidad.

b Aumentar el rendimiento.

c Reducir el tiempo de servicio de las peticiones.

d Todas las restantes afirmaciones son ciertas

15 *La replicación mejora el rendimiento de un servicio en el cual:*

a Todas las operaciones son de lectura.

b Todas las operaciones implican escrituras.

c Las réplicas están continuamente recuperándose de fallos en sus procesos.

d Se usa un modelo de replicación activa.

16 *En el modelo de partición primaria:*

- **a** Se puede mantener una consistencia secuencial entre los procesos activos.
- **b** Se asume que no hay ni defectos ni errores en el sistema.
- **c** Aplicamos el teorema CAP sacrificando la tolerancia a las particiones.
- **d** Todos los subgrupos de nodos pueden continuar.

17 *La consistencia secuencial es menos escalable que la consistencia FIFO porque:*

- **a** La consistencia secuencial requiere un mayor número de réplicas que la FIFO.
- **b** Para finalizar localmente una escritura en la consistencia secuencial se requiere comunicación con otros nodos y en la consistencia FIFO no es necesario.
- **c** La consistencia secuencial requiere utilizar replicación multi-máster mientras la consistencia FIFO puede generarse con cualquier modelo de replicación.
- **d** Todas las opciones mencionadas son válidas.

18 *¿Cuál de las siguientes afirmaciones sobre la replicación multi-máster es FALSA?*

- **a** Proporciona un modelo de consistencia rápido.
- **b** Respeta la consistencia causal.
- **c** Permite responder al cliente antes de pasar las modificaciones a las demás réplicas.
- **d** No es replicación activa, pero hay más de una réplica primaria.

19 *Sea un conjunto de réplicas de un componente que necesitan propagar las escrituras que aplican sobre un estado compartido.*

Cada réplica aplica sus escrituras localmente y, a continuación, las difunde mediante un socket PUB de ZMQ.

Cada réplica lee las escrituras de las demás al escuchar en un socket SUB de ZMQ, suscrito a todo lo que publican las demás.

Sabiendo que en los sockets utilizamos el protocolo tcp, indica cuál es el modelo de consistencia más fuerte que se cumple:

- **a** Secuencial.
- **b** FIFO.
- **c** Causal.
- **d** Caché.

20 *Para reforzar/relajar el nivel de consistencia entre los procesos* `mongod` *en una base de datos MongoDB podemos utilizar este mecanismo:*

- **a** Ninguno. La consistencia no puede reforzarse ni relajarse.
- **b** Utilizaremos valores bajos en el write concern para relajar la consistencia y valores altos para reforzarla.
- **c** Relajaremos la consistencia si desplegamos MongoDB sobre un IaaS en lugar de sobre máquinas físicas.
- **d** Relajaremos la consistencia si utilizamos pocos servidores de configuración y la reforzaremos si utilizamos muchos.

21 *Dos servicios A y B están basados en los mismos componentes y se han desplegado sobre una misma plataforma en la nube.*

En su despliegue se ha realizado una configuración ligeramente distinta entre ambos a la hora de gestionar su adaptabilidad.

Podríamos afirmar que A es más elástico que B si sometiendo ambos servicios a los mismos niveles de carga variable, y cumpliendo ambos el mismo SLA, entonces:

a El coste en recursos del servicio A resulta ser menor que el del servicio B.

b El tiempo de respuesta del servicio A es inferior al del servicio B.

c La disponibilidad del servicio A es superior a la del servicio B.

d Todas las opciones mencionadas son válidas.

22 *Considerando una ejecución de este programa en un ordenador con cuatro núcleos de procesamiento en su CPU, ¿Qué afirmación es cierta?*

```
const cluster = require('cluster')
const http = require('http')
const numCPUs = require('os').cpus().length
function processRequest(req,resp) {
  // Some code for processing an HTTP request
  // and generate its response.
}
if (cluster.isMaster) {
  for (var i=0; i < numCPUs; i++) cluster.fork()
  cluster.on('exit', function(worker) {
    cluster.fork()
  })
} else {
  http.createServer(processRequest).listen(8000)
}
```

a El proceso maestro reacciona a la finalización de un proceso trabajador, creando otro que lo reemplace.

b El proceso maestro genera un solo proceso trabajador.

c Si se crease más de un trabajador, el segundo y los siguientes fallarían al tratar de escuchar en el puerto 8000, pues ese puerto ya estaría ocupado.

d Todas las opciones mencionadas son válidas.

23 *Los proxies inversos pueden mejorar la escalabilidad de un servicio porque:*

a Equilibran la carga entre los servidores.

b Pueden mantener una caché y reducir así la carga de los servidores.

c Pueden detectar la caída de un servidor y redirigir sus solicitudes pendientes de respuesta a otro, evitando así que el cliente las reenvíe.

d Todas las opciones mencionadas son válidas.

24 *Considerando el siguiente programa con el que se quiere crear un cluster de procesos trabajadores (servidores web* `"hello world"`*), indica cuál es la afirmación cierta:*

```
const cluster = require('cluster')
let numReqs = 0
if (cluster.isMaster) {
  setInterval(() => {console.log('numReqs = ' +
numReqs); numReqs = 0}, 10000)
  for (let i = 0; i < 4; i++) cluster.fork()
  for (var id in cluster.workers)
    cluster.workers[id].on('message', msg => {num-
Reqs += 1})
} else {
  require('http').createServer((req, res) => {
    res.writeHead(200); res.end('hello world')
    process.send({id:process.pid})
  }).listen(8000)
}
```

a Se instancian tantos procesos trabajadores como procesadores haya en la máquina donde se ejecuta el programa.

b El programa aborta al iniciarse porque los procesos trabajadores no pueden escuchar en un mismo puerto, el 8000.

c El proceso máster muestra en consola, cada 10 seg., el número de peticiones atendidas por los trabajadores durante los últimos 10 seg.

d El formato del mensaje enviado por los procesos trabajadores es incorrecto. El cluster funciona, en cuanto que se atienden correctamente las peticiones de servicio, pero no puede llevar su cuenta y mostrarla periódicamente.

25 *Las diferencias del cliente externo respecto del presentado en el sistema* `3_CBW_FTC` *incluyen*

a Que el cliente externo no puede emplear 127.0.0.1 como IP del broker, pero en `3_CBW_FTC` sí

b Que el cliente externo debe enviar un segmento adicional para informar al broker sobre su dirección externa de respuesta

c Que el cliente externo necesita conocer la IP del anfitrión del contenedor donde se despliega el broker

d Que el cliente externo necesita conocer la IP del broker cuando se despliegue en el anfitrión

26 *En el ejemplo* `1_A_MANO` *de la Práctica 3 se debía averiguar la dirección IP del contenedor en el que se ejecutara el broker. Esa información se necesitaba para configurar manualmente el URL que se recibía como argumento en el cliente y en el trabajador.*

¿Qué orden docker se necesitaba para obtener, entre otras informaciones, esa dirección IP?

a `docker images`

b `docker logs`

c `docker-compose up`

d `docker inspect`

27 *En el ejemplo* `1_A_MANO` *de la Práctica 3 se debía averiguar la dirección IP del contenedor en el que se ejecutara el broker. Esa información se necesitaba para configurar manualmente el URL que se recibía como argumento en el cliente y en el trabajador.*

Supongamos que ya hemos obtenido esa información y hemos modificado adecuadamente el `Dockerfile` *del cliente.*

¿Qué secuencia de órdenes deberíamos utilizar posteriormente para ejecutar ese cliente en un contenedor?

a `docker up imclient`

b `docker commit imclient`, `docker run imclient`

c `docker build -t imclient .` (en el directorio de ese Dockerfile), `docker run imclient`

d `docker images`, `docker ps`, `docker stop imbroker`, `docker run imclient`

28 *En el ejemplo* `2_CBW` *de la Práctica 3 se automatizaba la resolución de dependencias entre los componentes* `cli`, `bro` *y* `wor` *que habíamos tenido que resolver manualmente en el ejemplo* `1_A_MANO`.

Si asumimos que todavía no se han generado las imágenes `client`, `broker` *y* `worker`, *¿qué efecto tendrá la orden*

```
docker-compose up --scale cli=4 --scale wor=2
```

cuando sea ejecutada en ese directorio `2_CBW`*?*

a Se construirán las imágenes `client`, `broker` y `worker`. Se iniciará un broker, dos trabajadores y cuatro clientes.

b Ninguno, pues si las imágenes todavía no existen, no se podrá lanzar ningún contenedor.

c Se construirán las imágenes `client`, `broker` y `worker`. Se iniciarán dos trabajadores y cuatro clientes.

d Ninguno, pues no se puede utilizar la opción `--scale` si previamente no se ha utilizado un `docker-compose up` sin opciones adicionales.

29 *El ejemplo* `3_CBW_FTC` *de la Práctica 3 utilizaba un broker tolerante a fallos (ya visto en la práctica 2) y era capaz de gestionar tres clases de clientes (A, B y C) y tres clases de trabajadores (A, B y C).*

Para gestionar esas tres clases de trabajadores se emplea este fragmento en el fichero `docker-compose.yml`.

```
worA:
  image: worker
  build: ./worker/
  links:
    - bro
  environment:
    - BROKER_URL=tcp://bro:9999
    - CLASSID=A
worB:
  image: worker
  build: ./worker/
  links:
    - bro
  environment:
    - BROKER_URL=tcp://bro:9999
    - CLASSID=B
worC:
  image: worker
  build: ./worker/
  links:
    - bro
  environment:
    - BROKER_URL=tcp://bro:9999
    - CLASSID=C
```

En base a esa configuración, ¿cuántas imágenes docker se necesitan para gestionar a estos tres tipos de trabajadores?

a Ninguna. Los trabajadores se ejecutan en contenedores y las imágenes no son estrictamente necesarias.

b Una.

c Tres. Una por cada tipo de contenedor.

d No podemos determinarlo con esta información. Necesitaríamos revisar también la información contenida en sus respectivos `Dockerfile`.

30 *En la segunda sesión de la Práctica 3 (apartado* `4_CBW_FTCL`*) se añadía un componente logger que debía desplegarse junto al broker, clientes y trabajadores.*

Aparte del directorio específico para el logger, su fichero fuente, su `Dockerfile` *y la creación de un directorio para guardar el fichero del log, ¿qué otros cambios en la configuración introdujo este componente?*

a Ninguno. La orden `docker-compose` dedujo las dependencias existentes automáticamente.

b Leves modificaciones en el fichero `docker-compose.yml` para especificar la secuencia de inicio apropiada.

c Modificaciones en el `docker-compose.yml` para indicar las nuevas dependencias entre componentes y la gestión de volúmenes, así como algunas modificaciones en uno de los `Dockerfile` utilizados en apartados anteriores.

d Hubo que rehacer por completo el `docker-compose.yml` pues la configuración de todos los elementos de este despliegue se vio alterada por la inclusión del logger. Además, hubo que modificar cada `Dockerfile`.

31 *En la segunda sesión de la Práctica 3 se describe cómo permitir que clientes externos a la máquina virtual puedan interactuar con el broker (apartado* `5_CBW_FTCL_CLEXT`*).*

Para ello se sugiere la utilización de la sección ports: en la configuración del componente broker del fichero `docker-compose.yml`.

Suponga el siguiente contenido para dicha sección, e indique su significado

```
ports:
   - "9998:9998"
```

a Obliga a que la operación `bind()` en el código del broker se realice sobre todas las interfaces de red del contenedor.

b Establece una correspondencia entre el puerto 9998 del contenedor del broker y el puerto 9998 del anfitrión.

c Establece una correspondencia entre el puerto 9998 de los contenedores de los clientes y el puerto 9998 del anfitrión.

d Permite que los clientes accedan al puerto 9998 del contenedor del broker, sin necesidad de utilizar ningún recurso del anfitrión.

32 *En la tercera sesión de la Práctica 3 se presenta un nuevo componente* `"worcli"`. *¿Cuál es el objetivo de ese componente?*

a Es un nuevo tipo de cliente que es capaz de modificar el comportamiento de los trabajadores con los que interactúa, modificando su tiempo de respuesta.

b Es un nuevo tipo de cliente que modifica dinámicamente su clase (entre las A, B y C disponibles) para así poder interactuar con todos los tipos de trabajadores.

c Se comporta como trabajador para un servicio y como cliente para otro, encadenando así dos servicios CBW.

d Ninguna de las opciones mencionadas es correcta.