TSR - Rec_Segundo Parcial. 2025-01-30

Este examen consta de 17 cuestiones, con una puntuación total de 10 puntos. Cada cuestión tiene 4 alternativas, de las cuales debe elegirse una sola opción. La nota se calcula de la siguiente forma: tras descartar la peor respuesta, cada acierto suma 10/16 puntos y cada error descuenta 10/48 puntos. Debes contestar en la hoja de respuestas.

D

1. ***Este programa cliente se utilizó en el Tema 3 para ilustrar que el patrón REQ/REP llegaba a bloquear las interacciones cliente/servidor cuando uno de los servidores (por ejemplo, el que utilizase el puerto 8888) abortaba tras recibir una solicitud, pero antes de devolver su correspondiente respuesta. En ese caso, el segundo mensaje no llegaba a enviarse al segundo servidor.***

```
const zmq = require('zeromq')
const rq = zmq.socket('req')
rq.connect('tcp://127.0.0.1:8888')
rq.connect('tcp://127.0.0.1:8889')
rq.send('Hello')
rq.send('Hello again')

rq.on('message', function(msg) {
   console.log('Response: ' + msg)
})
```

¿Qué sucedería si el servidor correspondiente utilizase sockets ROUTER en lugar de sockets REP (junto con el resto de cambios pertinentes para recibir adecuadamente solicitudes y enviar correctamente respuestas) y se diera la misma situación de fallo?

A. El cliente ya no se bloquearía en esa situación
B. Habría ejecuciones en las que el cliente se bloquearía y otras en las que no
C. La situación inicialmente descrita en el enunciado jamás podría darse al utilizar el patrón REQ/REP
D. No se observaría ningún cambio: el cliente seguiría bloqueándose

2. ***Este programa cliente se utilizó en el Tema 3 para ilustrar que el patrón REQ/REP llegaba a bloquear las interacciones cliente/servidor cuando uno de los servidores (por ejemplo, el que utilizase el puerto 8888) abortaba tras recibir una solicitud, pero antes de devolver su correspondiente respuesta. En ese caso, el segundo mensaje no llegaba a enviarse al segundo servidor.***

```
const zmq = require('zeromq')
const rq = zmq.socket('req')
rq.connect('tcp://127.0.0.1:8888')
rq.connect('tcp://127.0.0.1:8889')
rq.send('Hello')
 rq.send('Hello again')

rq.on('message', function(msg) {
   console.log('Response: ' + msg)
})
```

¿Qué sucedería si este cliente utilizase sockets DEALER en lugar de sockets REQ (junto con el resto de cambios pertinentes para enviar adecuadamente las solicitudes y recibir correctamente las respuestas) y se diera la misma situación de fallo?

A. Habría ejecuciones en las que el cliente se bloquearía y otras en las que no
B. La situación inicialmente descrita en el enunciado jamás podría darse al utilizar el patrón REQ/REP
C. No se observaría ningún cambio: el cliente seguiría bloqueándose
D. El cliente ya no se bloquearía en esa situación

3. ***¿Qué ventajas aporta la máquina virtual cuando se la compara con el contenedor a la hora de realizar un despliegue?***
   - **A.** Menor consumo de recursos
   - **B.** Mayor aislamiento, junto con una menor dependencia del sistema operativo anfitrión
   - **C.** Ficheros de configuración más sencillos, proporcionando mayor facilidad en el despliegue
   - **D.** Ninguna; el contenedor siempre será una mejor opción

4. ***¿Cuántos segmentos y con qué contenido añade o espera (y elimina) un socket DEALER al enviar o recibir un mensaje?***
   - **A.** Añade un delimitador inicial al enviar un mensaje, y espera y elimina un delimitador inicial al recibir un mensaje
   - **B.** Añade la identidad del socket emisor al enviar un mensaje, y no espera, ni añade, ni elimina segmentos al recibir un mensaje
   - **C.** Espera y utiliza, eliminándolo, un segmento con la identidad del receptor al enviar un mensaje, y añade la identidad del socket emisor al recibir un mensaje
   - **D.** No añade, espera o elimina ningún segmento tanto en la operación de envío como en la de recepción

5. ***¿Qué etapa, de entre las siguientes, del ciclo de vida del software no está incluida en el despliegue?***
   - **A.** Actualización
   - **B.** Instalación
   - **C.** Activación
   - **D.** Análisis

6. ***Este es un ejemplo de Dockerfile utilizado tanto en la Práctica 3 como en el Tema 4:***

```
FROM tsr-zmq
COPY ./tsr.js tsr.js
RUN mkdir broker
WORKDIR broker
COPY ./broker.js mybroker.js
EXPOSE 9998 9999
CMD node mybroker 9998 9999
```

   Seleccione la afirmación verdadera sobre el contenido de ese fichero:
   - **A.** Las instrucciones `RUN` y `WORKDIR` utilizadas en ese fichero permiten que el fichero `mybroker.js` localice en el lugar esperado el módulo `tsr.js`
   - **B.** Allí donde resida este fichero también deberemos tener los ficheros `tsr.js` y `mybroker.js` para que no haya errores al utilizar este Dockerfile
   - **C.** Todas las demás afirmaciones son ciertas
   - **D.** La primera línea indica que la imagen a generar se guardará en el directorio `tsr-zmq`

7. ***Un fichero Dockerfile permite:***
   - **A.** Especificar qué instrucciones deben utilizarse para construir una imagen que permitirá iniciar contenedores
   - **B.** Especificar qué instrucciones deben utilizarse para desactivar, parar y eliminar los contenedores usados en el despliegue de algún servicio
   - **C.** Desplegar múltiples instancias de varios componentes relacionados en un mismo ordenador anfitrión
   - **D.** Especificar qué instrucciones deben utilizarse para generar una máquina virtual

8. ***Seleccione la afirmación verdadera sobre la orden*** `docker compose up`
   - **A.** Presupone que habrá un fichero `docker-compose.yml` en el directorio donde sea utilizada
   - **B.** Despliega un servicio distribuido, iniciando una instancia (es decir, un contenedor) de cada uno de sus componentes, en un determinado orden
   - **C.** Permite utilizar la opción `--scale compo=X` para ejecutar `X` instancias del componente `compo`
   - **D.** Todas las demás afirmaciones son ciertas

9. ***Considere el siguiente ejemplo de fichero `docker-compose.yml`:***

```
version: '2'
services:
  ca:
    image: ima
    build: ./dira/
    links:
      - cc
    environment:
      - C_HOST=cc
      - C_PORT=9998
  cb:
    image: imb
    build: ./dirb/
    links:
      - cc
    environment:
      - C_HOST=cc
      - C_PORT=9999
  cc:
    image: imc
    build: ./dirc/
    expose:
      - "9998"
      - "9999"
```

Seleccione la afirmación verdadera sobre ese fichero:

A. Cuando se despliegue el servicio descrito en ese fichero, el primer componente a iniciar será `ca`, pues se ha especificado en primer lugar

B. El Dockerfile presente en el subdirectorio `dirc` utiliza dos variables de entorno llamadas `C_HOST` y `C_PORT`

C. Todas las demás afirmaciones son falsas

D. Cuando se despliegue el servicio descrito en ese fichero, el primer componente que se iniciará será `cc`

10. ***Seleccione la afirmación verdadera sobre la replicación activa:***

A. Es el modelo de replicación utilizado en el componente que gestiona los datos persistentes de la Wikipedia

B. Gestiona las operaciones no deterministas sin generar inconsistencia entre réplicas

C. Puede ofrecer mejor rendimiento que la replicación pasiva, especialmente cuando las operaciones a gestionar generen modificaciones de gran volumen

D. Utiliza una réplica primaria y varias secundarias

11. ***Seleccione la afirmación verdadera sobre la replicación pasiva:***

A. Recuperarse de un fallo en su réplica primaria es más complejo que ante el fallo de una réplica secundaria

B. Ofrece mejor rendimiento que la replicación activa, especialmente cuando sus operaciones generen muchas modificaciones de gran volumen

C. Ofrece mejor rendimiento que la replicación multi-máster

D. Una vez recibida la petición a gestionar, necesita menos interacciones entre las réplicas que la replicación activa

12. ***Suponga un sistema formado por tres procesos P1, P2 y P3, donde se ha dado la siguiente ejecución: `W1(x)2, R2(x)2, W2(y)1, R1(y)1, R3(y)1, R3(x)2`. Esa ejecución respeta, entre otras, la consistencia:***

A. Caché

B. Estricta

C. Secuencial

D. Causal

13. ***Seleccione la afirmación verdadera sobre las replicaciones activa o multi-máster:***

A. Resulta más adecuado emplear replicación multi-máster frente a replicación activa si pretendemos tolerar fallos bizantinos

B. La replicación multi-máster es generalmente más rápida en recuperar los servicios tras un fallo que la replicación activa

C. Resulta adecuado emplear replicación multi-máster si pretendemos replicar servicios no deterministas

D. La replicación multi-máster resulta menos adecuada que la replicación activa para entornos altamente escalables

14. ***Seleccione la afirmación verdadera sobre el teorema CAP:***
    - **A.** El teorema CAP viene a indicar que lo más frecuente será sacrificar la disponibilidad en sistemas altamente escalables
    - **B.** Si tenemos un sistema donde se adopta el modelo de partición primaria y tenemos alta consistencia, estaremos sacrificando la disponibilidad
    - **C.** Si tenemos un sistema que sigue el modelo causal, estaremos garantizando consistencia fuerte, por tanto tendremos que sacrificar soporte a particiones o disponibilidad
    - **D.** Tolerancia a particiones, tal y como se menciona en el teorema CAP, significa que tolerar particiones es equivalente a emplear el modelo de partición primaria

15. ***La orden para averiguar la dirección IP de un contenedor que proporciona un servicio, después de que se haya iniciado es (suponiendo que ya conocemos el ID o el nombre del contenedor):***
    - **A.** `docker inspect <ID>`
    - **B.** `docker logs <ID>`
    - **C.** Ninguna de las demás opciones es cierta
    - **D.** `docker ps`

16. ***Supongamos un sistema CBW al que añadiremos dos componentes `logger`, para recoger los mensajes de traza generados en el socket frontend y backend del broker respectivamente. Para ello, el broker utilizará un socket PUB adicional, sobre el que aplicará un `bind` y con el que emitirá mensajes con dos segmentos, con "frontend" o "backend" en el primero. Los puertos a utilizar por este broker se recibirán desde la línea de órdenes: primero el del PUB, segundo el del ROUTER de clientes y tercero el del ROUTER para trabajadores. Si el Dockerfile inicial del broker tiene este contenido:***

```
FROM tsr-zmq
COPY ./tsr.js tsr.js
RUN mkdir broker
WORKDIR broker
COPY ./broker.js mybroker.js
```

¿Qué líneas deberían añadirse para que gestionase adecuadamente esa nueva configuración? (por maquetación, alguna línea puede haberse partido en dos)

- **A.**
```
EXPOSE 9998 9999
CMD node mybroker 9998 9999 $LOGGER_HOST
$LOGGER_PORT
```
- **B.**
```
EXPOSE 9998 9999
CMD node mybroker 9998 9999 localhost 9997
```
- **C.**
```
EXPOSE 9997 9998 9999
CMD node mybroker 9997 9998 9999
$LOGGER1_HOST $LOGGER2_HOST
```
- **D.**
```
EXPOSE 9997 9998 9999
CMD node mybroker 9997 9998 9999
```

17. ***Supongamos un sistema CBW al que añadiremos dos componentes `logger`, para recoger los mensajes de traza generados en el socket frontend y backend del broker respectivamente. Para ello, el broker utilizará un socket PUB adicional, sobre el que aplicará un `bind` y con el que emitirá mensajes con dos segmentos, con "frontend" o "backend" en el primero. Por su parte, los dos componentes `logger` comparten el mismo programa y la misma imagen. Cada uno se suscribirá a un prefijo distinto. En base a ese prefijo, se generará un nombre de fichero de log distinto. El directorio en el que guardarán esos ficheros en sus contenedores será el mismo para ambos: `/tmp/cbwlog`. El programa `logger.js` necesitará recibir, en esta secuencia, estos argumentos desde la línea de órdenes: `hostDelBroker puertoDelBroker prefijoAlQueSuscribirse`. Si el Dockerfile inicial del `logger` tiene este contenido:***

```
FROM tsr-zmq
COPY ./tsr.js tsr.js
RUN mkdir logger
WORKDIR logger
COPY ./logger.js mylogger.js
```

¿Cuáles serían las líneas que se tendrían que añadir a ese Dockerfile para su correcto funcionamiento? (por maquetación, alguna línea puede haberse partido en dos)

A. `VOLUME /tmp/cbwlog`
`CMD node logger localhost $BROKER_PORT $PREFIX`

B. `VOLUME /tmp/cbwlog`
`CMD node mylogger $BROKER_HOST $BROKER_PORT $PREFIX`

C. `VOLUME /tmp/cbwlog`
`EXPOSE 9997`
`CMD node mylogger 9997 /tmp/cbwlog/logs`

D. `VOLUME /tmp/cbwlog`
`CMD node logger $BROKER_HOST $BROKER_PORT $PREFIX`