

Este examen vale 10 puntos y consta de 25 preguntas. Cada pregunta plantea 4 alternativas y tiene sólo una respuesta correcta. Cada respuesta correcta proporciona 10/24 puntos, y cada error resta 10/72, salvo el primero, que se descarta. Debes responder en la hoja de respuestas.

Consideremos estos programas JavaScript:

```
1 // Program: client1.js
2 const zmq = require('zeromq')
3 const rq = zmq.socket('req')
4 rq.connect('tcp://127.0.0.1:8888')
5 rq.send('Hello')
6 rq.on('message', (msg) => {
7   console.log('Response: ' + msg)
8   rq.close()
9 })
```

```
1 // Program: server1.js
2 const zmq = require('zeromq')
3 const rp = zmq.socket('rep')
4 rp.bindSync('tcp://127.0.0.1:8888')
5 rp.on('message', (msg) => {
6   console.log('Request: ' + msg)
7   rp.send('World')
8 })
```

Utilizan el patrón de comunicación REQ-REP, y deseamos modificarlos para usar el patrón DEALER-ROUTER manteniendo la misma funcionalidad y usando un **número mínimo de segmentos** en los mensajes.

1 Los cambios a aplicar en client1.js son:

- a const rq = zmq.socket('router') // Line 3
- b const rq = zmq.socket('dealer') // Line 3
rq.send(['', 'Hello']) // Line 5
- c const rq = zmq.socket('dealer') // Line 3
- d Ninguna otra opción es correcta.

2 Los cambios a aplicar en server1.js son:

- a const rp = zmq.socket('dealer') // Line 3
- b const rp = zmq.socket('router') // Line 3
rp.on('message', (who, msg) => { // Line 5
 rp.send([who, 'World']) // Line 7
- c const rp = zmq.socket('router') // Line 3
rp.on('message', (who, sep, msg) => { // Line 5
 rp.send([who, 'World']) // Line 7
- d const rp = zmq.socket('router') // Line 3

3 Consideremos un sistema compuesto por un proceso cliente y dos procesos servidores. Cada servidor usa un socket REP (realizando un bind sobre él), mientras que el cliente usa un socket REQ conectado a ambos sockets REP. El cliente envía una solicitud por segundo y nunca termina.

Supongamos que reemplazamos el socket REQ en el proceso cliente con un socket DEALER, y adaptamos adecuadamente la gestión de segmentos de mensajes para garantizar que todos los mensajes enviados y recibidos se entreguen. Con esta nueva configuración, los tres procesos se reinician. ¿Qué sucede en este nuevo escenario cuando muere un servidor?

- a Lo mismo que en el primer sistema: la comunicación se bloquea y no se entrega ninguna otra solicitud a su destino.
- b Todas las solicitudes de los clientes se entregan al servidor restante y no se pierden mensajes.
- c Cuando se cierra el servidor el cliente genera una excepción y normalmente aborta.
- d La comunicación no se bloquea y el servidor restante seguirá recibiendo la mitad de las solicitudes enviadas por el cliente.

4 ¿Cuál de las siguientes etapas del ciclo de vida del software no forma parte del despliegue?

- a Diseño de software.
- b Eliminación de software.
- c Instalación y configuración del software.
- d Actualización de software.

5 Entre estas opciones, ¿cuál es la mejor para desplegar un servicio en múltiples anfitriones?

- a Kubernetes
- b docker
- c docker-composer
- d Despliegue manual

6 Para ejecutar un contenedor Docker en un anfitrión determinado, necesitamos...

- a Un hipervisor.
- b Un sistema operativo invitado instalado y configurado adecuadamente en la imagen que se va a ejecutar.
- c Un sistema operativo anfitrión diferente al que se supone en la imagen que se va a ejecutar.
- d Ninguna de las restantes opciones es correcta.

Supongamos que la imagen 'tsr-zmq' existe y tiene el contenido y la funcionalidad explicados en el Tema 4 y la Práctica 3. Supongamos también que este Dockerfile (al que nos referiremos como 'Dockerfile A', aunque su nombre real es 'Dockerfile') se ha guardado en el directorio /home/user/docker/config:

```
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
```

7 Consideremos el Dockerfile A. ¿Qué orden debería usarse para crear una imagen llamada 'broker' si el directorio de trabajo actual de nuestro shell es /home/user/docker?

- a docker commit config broker
- b docker run broker
- c docker build -t broker config
- d docker build -t broker

8 Consideremos el Dockerfile A. ¿Dónde debe colocarse la imagen 'tsr-zmq' y los archivos 'tsr.js' y 'broker.js' para poder utilizar con éxito ese Dockerfile para crear una nueva imagen?

- a Todos los elementos necesarios (tsr-zmq, tsr.js y broker.js) deben estar en /home/user/docker/config
- b La imagen tsr-zmq debe estar en el repositorio local o estar presente en hub.docker.com, mientras que los archivos deben estar en /home/user/docker/config
- c Todos los elementos requeridos (tsr-zmq, tsr.js y broker.js) deben estar en algún lugar del anfitrión local, ya que el gestor Docker los buscará y encontrará sin ningún problema.
- d Ninguna de las demás opciones es correcta.

9 Consideremos el Dockerfile A. ¿Qué orden ejecutará el intérprete 'bash' en un contenedor que usa la imagen 'broker1' generada usando ese Dockerfile A?

- a Ninguna orden de Docker puede lograr ese objetivo.
- b docker run bash
- c docker run -i -t broker1 bash
- d docker-compose bash

- 10** Consideremos que este Dockerfile se ha utilizado para crear una imagen llamada 'broker2':

```
FROM tsr-zmq
COPY ./tsr.js tsr.js
RUN mkdir broker
WORKDIR broker
COPY ./broker.js mybroker.js
EXPOSE 9998 9999
ENTRYPOINT [ "/usr/bin/node", "mybroker" ]
CMD [ "9998", "9999" ]
```

¿Podemos usar 'broker2' para ejecutar el shell 'bash' en un contenedor?

- a No, ya que la imagen de broker2 no se puede crear porque un Dockerfile no puede combinar ENTRYPOINT y CMD.
- b Por omisión, no, ya que el programa que se ejecuta en contenedores generados a partir de la imagen de broker2 es node mybroker
- c Sí, con esta orden:
docker run -i -t broker2 bash
- d No, ya que cuando combinamos ENTRYPOINT y CMD en el mismo Dockerfile solo se considera el último de ellos en el archivo, y en este ejemplo CMD tiene valores no válidos.

- 11** La primera sesión del Laboratorio 3 comienza con un despliegue manual de un sistema client-broker-worker. En ese sistema, el broker se inicia primero y los otros dos componentes necesitan conocer la dirección IP del contenedor del broker. La orden (o conjunto de órdenes) que proporciona esa información es...

- a docker images
- b docker inspect ID
Suponiendo que el ID del contenedor del broker se haya encontrado previamente.
- c docker-compose
- d Ninguna otra opción es correcta.

- 12** Consideremos que este docker-compose.yml está en el directorio /home/user/docker

```
version: '2'
services:
  svca:
    image: imga
    links:
      - svcb
    environment:
      - B_HOST=svcb
  svcb:
    image: imgb
    links:
      - svcc
    environment:
      - C_HOST=svcc
  expose:
    - "9999"
  svcc:
    image: imgc
    expose:
      - "9999"
```

Seleccione la oración FALSA sobre el servicio que se desplegará usando ese archivo.

- a Los componentes de ese servicio se iniciarán en este orden: svcc, svcb, svca.
- b Ese servicio se puede desplegar, pero sus componentes svcc y svcb entran en conflicto al escuchar en el mismo puerto del anfitrión (9999).
- c Podemos desplegar una instancia de los componentes svcc y svcb y seis instancias del componente svca usando esta orden en /home/user/docker:
docker-compose up -d --scale svca=6
- d Una vez que se haya desplegado el servicio, podemos detener y eliminar todos sus contenedores usando esta orden en /home/user/docker:
docker-compose down

- 13** *Este es un esqueleto del archivo docker-compose.yml a utilizar en la segunda mitad de la primera sesión del Laboratorio 3 para automatizar el despliegue de un sistema CBW:*

```
version: '2'
services:
  cli:
    image: client
    build: ./client/
    links:
      - W
    environment:
      - BROKER_HOST=X
      - BROKER_PORT=9998
  wor:
    image: worker
    build: ./worker/
    links:
      - Y
    environment:
      - BROKER_HOST=Z
      - BROKER_PORT=9999
  bro:
    image: broker
    build: ./broker/
    expose:
      - "9998"
      - "9999"
```

Los valores necesarios de W, X, Y y Z para completar dicho archivo docker-compose.yml con el contenido mínimo para administrar correctamente ese despliegue (es decir, para garantizar un orden de inicio apropiado y una resolución de dependencias adecuada) son :

- a** X=bro, Z=bro, y las cláusulas links no son necesarias y deben eliminarse, por lo que W e Y no necesitan ningún valor.
- b** W=wor, X=wor, mientras que las cláusulas links y environment en wor deben eliminarse, por lo que Y y Z no necesitan ningún valor.
- c** No se necesitan cláusulas links ni environment para automatizar este despliegue. Todas esas secciones pueden eliminarse y la implementación resultante se comportará correctamente. Por lo tanto, no se necesita ningún valor para W, X, Y y Z.
- d** W=bro, X=bro, Y=bro, Z=bro.

- 14** *La segunda sesión del Laboratorio 3 introduce un nuevo componente logger en el sistema CBW. ¿Qué otros componentes CBW interactúan con este nuevo logger?*

- a** Los clientes y trabajadores envían sus mensajes de traza al logger.
- b** Sólo el broker envía sus mensajes de traza al logger.
- c** Los clientes, workers y broker envían sus mensajes de traza al logger.
- d** Sólo los workers envían sus mensajes de traza al logger.

- 15** *La segunda sesión del Lab 3 propone el despliegue de otro tipo de cliente (un cliente externo) que se ejecutará en otro ordenador, diferente al anfitrión donde docker y docker-compose administran los contenedores CBW. ¿Qué se necesita en la configuración del componente broker para habilitar ese cliente externo?*

- a** Se necesita una cláusula 'ports:' en la sección 'bro' del archivo docker-compose.yml para asignar el puerto 9998 del contenedor del broker al puerto 9998 del anfitrión.
- b** Se necesita una cláusula 'ports:' en el Dockerfile de la imagen del broker para asignar el puerto 9998 del contenedor del broker al puerto 9998 del anfitrión.
- c** No se necesita nada especial, sólo necesitamos averiguar la IP del contenedor del broker, usando docker ps y docker inspect para ello.
- d** Se necesita una instrucción 'EXPOSE' en el Dockerfile de la imagen del broker, para indicar qué puerto del contenedor del broker deben utilizar los clientes externos.

- 16** El `docker-compose.yml` utilizado en la tercera sesión del Laboratorio 3 contiene una línea como esta:

```
image: docker.io/bitnami/mariadb:11.1
```

¿Cuál es la consecuencia de reemplazar la parte '11.1' de esa línea con 'latest' en ese despliegue?

- a** Que, tal vez, en dos años el archivo 'docker-compose.yml' resultante se vuelva inútil.
 - b** Ninguna. El despliegue resultante se comporta y se comportará correctamente, independientemente de ese cambio.
 - c** Un error, pues la palabra 'latest' no puede formar parte del nombre de una imagen docker.
 - d** Un error, pues 'latest' no puede ser parte de un nombre de ruta o directorio.
- 17** Si comparamos la replicación activa y pasiva, el modelo pasivo es el modelo de replicación preferido cuando las operaciones solo modifican una pequeña parte del estado del servicio porque...:
- a** El modelo pasivo debe enviar esas modificaciones a las réplicas secundarias y estas deben aplicarlas, mientras que en el modelo activo no se necesita transferencia de modificaciones.
 - b** Cuando esas modificaciones son pequeñas, las réplicas secundarias no las necesitan.
 - c** Cuando esas modificaciones son pequeñas, pueden transferirse de forma asincrónica y sin ningún efecto en la consistencia resultante entre réplicas.
 - d** Todas las demás opciones son correctas.
- 18** Con respecto a los fallos de conectividad, si se asume un modelo de partición primaria:
- a** Todos los nodos del sistema pueden continuar y la consistencia del sistema resultante es muy relajada.
 - b** La disponibilidad del sistema está asegurada.
 - c** Los procesos en subgrupos menores se detienen.
 - d** Los servicios utilizan replicación pasiva.

- 19** Este es el archivo `docker-compose.yml` utilizado en la última sesión del Laboratorio 3:

```
version: '2'
services:
  mariadb:
    image: docker.io/bitnami/mariadb:11.1
    volumes:
      - 'mariadb_data:/bitnami/mariadb'
    environment:
      - ALLOW_EMPTY_PASSWORD=yes
      - MARIADB_USER=bn_wordpress
      - MARIADB_DATABASE=bitnami_wordpress
  wordpress:
    image: docker.io/bitnami/wordpress:6
    ports:
      - '80:8080'
      - '443:8443'
    volumes:
      - 'wordpress_data:/bitnami/wordpress'
    depends_on:
      - mariadb
    environment:
      - ALLOW_EMPTY_PASSWORD=yes
      - WORDPRESS_DATABASE_HOST=mariadb
      - WORDPRESS_DATABASE_PORT_NUMBER=3306
      - WORDPRESS_DATABASE_USER=bn_wordpress
      - WORDPRESS_DATABASE_NAME=bitnami_wordpress
    volumes:
      mariadb_data:
        driver: local
      wordpress_data:
        driver: local
```

¿Cómo podemos desplegar manualmente, con la ayuda de la orden `docker-compose`, los componentes `mariadb` y `wordpress` en dos anfitriones diferentes?

- a** No es posible desplegar el sistema, ya que no conocemos el contenido de los Dockerfiles que generaron las imágenes que se utilizarán.
- b** No es posible el despliegue en dos ordenadores, ya que este `docker-compose.yml` ya no funcionó como se esperaba con un solo anfitrión.
- c** Ese despliegue es posible sin aplicar ninguna modificación en ese archivo.
- d** Dividir el fichero en dos, uno por componente, y agregar una sección 'ports:' a la parte `mariadb` y un buen valor para `WORDPRESS_DATABASE_HOST` en la otra.

- 20** Si comparamos los modelos de consistencia causal y caché, ¿cuál de ellos es el más relajado?
- a Causal
 - b Caché
 - c En cuanto a su grado de relajación, son equivalentes.
 - d Su grado de relajación no se puede comparar.
- 21** Si consideramos el teorema CAP, ¿Qué modelo de consistencia podemos respetar si se necesita disponibilidad en un sistema particionado?
- a Estricto
 - b Causal
 - c Secuencial
 - d Ninguno
- 22** ¿Cuál de estas alternativas es una diferencia correcta entre los modelos multi-master y de replicación pasiva?
- a El modelo multi-master puede usar una réplica de procesamiento diferente (es decir, la 'master') por solicitud, mientras que el modelo pasivo siempre usa la misma réplica primaria.
 - b En el modelo pasivo, cada solicitud solo se envía a la réplica primaria, mientras que en el modelo multi-master el cliente difunde cada petición a todas las réplicas.
 - c El modelo multi-master puede manejar el modelo de fallos arbitrarios, mientras que el modelo pasivo no puede.
 - d Todas las demás opciones son verdaderas.
- 23** Node.js proporciona su módulo 'cluster' para:
- a Ejecutar un programa determinado en un conjunto de procesos, compartiendo algunos recursos (por ejemplo, el acceso a un socket) para implementar un servicio escalable localmente.
 - b Todas las demás opciones son verdaderas.
 - c Iniciar múltiples hilos de ejecución en un solo proceso.
 - d Implementar un servicio Node.js en un cluster de ordenadores.
- 24** MongoDB usa este modelo de replicación:
- a Replicación activa
 - b Replicación pasiva
 - c Replicación multi-master
 - d Sin replicación
- 25** Para mejorar su escalabilidad, MongoDB utiliza:
- a Réplicas secundarias legibles.
 - b Particionado horizontal de la base de datos.
 - c Todas las demás opciones son correctas.
 - d Una caché de la información de configuración en sus componentes mongos.