

Este examen tiene un valor de 10 puntos, y consta de 32 cuestiones. Cada cuestión plantea 4 alternativas y tiene una única respuesta correcta. Cada respuesta correcta aporta 10/32 puntos, y cada error descuenta 10/96 puntos. Debe contestar en la hoja de respuestas.

Hemos desarrollado el siguiente código para generar un sistema al que vamos a llamar ABC:

```
// client.js
const zmq = require("zeromq")
const url = process.argv[2]
let push = zmq.socket('push')
push.connect(url)
push.send('hola 1')
push.send('hola 2')
push.send('hola 3')
```

```
// a.js
const zmq = require("zeromq")
const [port1,port2]= process.argv.slice(2)
let pull = zmq.socket('pull')
let push = zmq.socket('push')
pull.bindSync('tcp://*:'+port1)
push.bindSync('tcp://*:'+port2)
pull.on('message', (m) => {push.send(m)})
```

```
// b.js
const zmq = require("zeromq")
const [url1,url2]= process.argv.slice(2)
let pull = zmq.socket('pull')
let push = zmq.socket('push')
pull.connect(url1)
push.connect(url2)
pull.on('message', (m) => {push.send(m)})
```

```
// c.js
const zmq = require("zeromq")
const port = process.argv[2]
let pull = zmq.socket('pull')
pull.bindSync('tcp://*:'+port)
pull.on('message', (m) => {console.log(m+"")})
```

Realizamos un arranque manual, lanzando las siguientes órdenes en terminales distintos de la misma máquina

```
node c 8002
```

```
node a 8000 8001
```

```
node b tcp://127.0.0.1:8001 tcp://127.0.0.1:8002
```

```
node b tcp://127.0.0.1:8001 tcp://127.0.0.1:8002
```

```
node client tcp://127.0.0.1:8000
```

Vemos que en el primer terminal (el de la orden node c 8002) el programa 'c' escribe

```
hola 1
hola 2
hola 3
```

Queremos automatizar el despliegue usando contenedores de forma que los clientes accedan de forma remota a través del puerto 80 del anfitrión. Para ello hemos preparado una imagen base denominada `tsr2021/ubuntu-zmq` (con la distribución `ubuntu` y soporte para `NodeJS` y `ZMQ`) y la siguiente estructura de directorios:

```
p2/docker-compose.yml
p2/a/Dockerfile
p2/a/a.js
p2/b/Dockerfile
p2/b/b.js
p2/c/Dockerfile
p2/c/c.js
```

En el Dockerfile para a.js habrá una línea con este contenido:

```
CMD node a 8000 8001
```

Por su parte, en el Dockerfile para el programa c.js tendremos esta instrucción:

```
CMD node c 8002
```

Las preguntas que hacen referencia al sistema ABC asumen dicho escenario, EN EL QUE EL CLIENTE NO ES UN ELEMENTO A CONSIDERAR DURANTE EL DESPLIEGUE AUTOMATIZADO

- 1 En relación con la ejecución manual del sistema ABC:
- a Aparecen 5 componentes distintos, con una instancia cada uno
 - b Aparecen 4 componentes distintos, uno de ellos con 2 instancias
 - c Usamos un único componente, ya que desplegamos todos los programas en la misma máquina
 - d Usamos un único componente, ya que todos los programas utilizan el mismo soporte para ejecución (NodeJS + ZMQ)
- 2 En el sistema ABC CON DESPLIEGUE AUTOMATIZADO:
- a No aparecen dependencias
 - b Únicamente hay dependencias de los clientes respecto de 'a'
 - c Hay dependencias de 'b' respecto de 'a' y 'c', y de clientes respecto de 'a'
 - d Únicamente hay dependencias de 'b' respecto de 'a' y 'c'
- 3 En cuanto a la automatización del despliegue del sistema ABC para ejecutar 'a', 'b' y 'c' en la misma máquina:
- a El uso de contenedores no es adecuado, ya que dentro de una misma máquina la única alternativa es usar máquinas virtuales
 - b Las partes 'b' y 'c' pueden utilizar contenedores, pero la parte a debe atender mensajes de clientes remotos, y por lo tanto únicamente puede implementarse mediante una máquina virtual
 - c Podemos usar contenedores para las partes 'a', 'b' y 'c'
 - d Si lanzamos 'a', 'b', 'c' en la misma máquina, no podemos lanzar clientes de forma remota: deben lanzarse también en el anfitrión

- 4 El fichero p2/b/Dockerfile del sistema ABC tiene el siguiente contenido

```
FROM tsr2021/ubuntu-zmq
COPY ./b.js b.js
CMD node b ...
```

Indica qué debemos añadir en los puntos suspensivos de la tercera línea

- a tcp://127.0.0.1:8001 tcp://127.0.0.1:8002
- b tcp://127.0.0.1:8001 \$URL
- c \$URL tcp://127.0.0.1:8002
- d \$URL1 \$URL2

- 5 Asumiendo que el fichero p2/b/Dockerfile del sistema ABC tiene el contenido correcto y que estamos situados en el directorio p2/, para crear de forma automática la imagen correspondiente a 'b' (la llamamos imb) debemos ejecutar:

- a docker build -t imb .
- b docker commit -t imb .
- c docker build -t imb ./b
- d docker run -t imb .

- 6 Asumiendo que los puntos suspensivos podrán representar, si fuese necesario, la lista de argumentos o valores para las variables de entorno a utilizar por el programa 'b' del sistema ABC y que se ha creado correctamente la imagen imb correspondiente a ese programa, indica la orden necesaria para lanzar una instancia de la misma sobre un contenedor

- a docker build imb ...
- b docker commit imb ...
- c docker start imb ...
- d docker run imb ...

- 7** En el fichero `p2/a/Dockerfile` del sistema ABC conviene utilizar la siguiente línea
- a** `EXPOSE 8000 8001`
 - b** `PORTS 8000 8001`
 - c** Tanto `EXPOSE 8000 8001` como `PORTS 8000 8001`
 - d** Ni `EXPOSE 8000 8001` ni `PORTS 8000 8001`
- 8** En relación con el servicio 'a' del fichero `docker-compose.yml` del sistema ABC
- a** Requiere parte `links` y parte `environment`
 - b** Requiere parte `links`, pero no `environment`
 - c** Requiere parte `environment`, pero no `links`
 - d** No requiere parte `environment` ni parte `links`
- 9** En relación con el servicio 'a' del fichero `docker-compose.yml` del sistema ABC:
- a** Requiere parte `ports`
 - b** Ninguna de las restantes afirmaciones es cierta
 - c** Requiere parte `expose`, pero no `ports`
 - d** No requiere parte `ports` ni parte `expose`
- 10** En el fichero `docker-compose.yml` del sistema ABC:
- a** Únicamente necesita una sección `environment` el servicio 'b'
 - b** Únicamente necesitan una sección `environment` los servicios 'a' y 'c'
 - c** Únicamente necesita una sección `environment` el servicio 'a'
 - d** Ningún servicio necesita una sección `environment`
- 11** Si en el sistema ABC todos los ficheros son correctos, una orden correcta para lanzar una instancia de 'a', una de 'b' y una de 'c' es
- a** `docker run ima & docker run imb & docker run imc &`
 - b** `docker-compose up`
 - c** `docker build --scale ima=1 --scale imb=1 --scale imc=1`
 - d** No se puede completar con una única orden: hay que realizar parte del trabajo a mano (averiguar la ip del contenedor, editar los Dockerfiles, etc.)
- 12** Si en el sistema ABC hemos desplegado correctamente el servicio compuesto por los programas 'a', 'b' y 'c' de forma automática, la forma correcta de lanzar un cliente en la misma máquina es:
- a** Ejecutando desde otro terminal la orden `node client tcp://127.0.0.1:80`
 - b** Ejecutando desde otro terminal la orden `node client tcp://127.0.0.1:8000`
 - c** No se puede lanzar el cliente en la misma máquina (debe ser remoto)
 - d** Ejecutando desde otro terminal la orden `docker run client tcp://127.0.0.1:80`
- 13** Con relación a la interoperabilidad de sockets DEALERS y REPs ¿cuál de los siguientes enunciados es FALSO?
- a** Todo mensaje enviado por un DEALER a un REP debe incluir, al menos, un segmento vacío.
 - b** El DEALER debe operar de modo sincrónico ya que el REP así lo requiere.
 - c** Varios DEALERS pueden estar conectados y operando con varios REPs.
 - d** Un REP puede estar conectado y operando con un conjunto de sockets en el que haya tanto DEALERS como REPs.

- 14** Se modifica el sistema clientes-broker-workers descrito en teoría del modo siguiente:
- 1) El broker en lugar de disponer de dos sockets de tipo ROUTER dispone de solo uno que presta servicio tanto a los clientes como a los workers.
 - 2) En consecuencia los clientes y los workers han sido modificados de modo que comparten el mismo endpoint (del ROUTER) y además son anónimos, esto es, no se ha asignado ningún identificador a su atributo identity.

Elige la afirmación correcta:

- a** Esta modificación se puede llevar a cabo, ya que el broker puede distinguir entre los clientes y los workers.
- b** Esta modificación no se puede llevar a cabo, ya que el broker no podría distinguir entre los clientes y los workers.
- c** Esta modificación solo se podría implantar si los workers utilizaran sockets de tipo REP.
- d** Ninguno de los restantes enunciados es válido.
- 15** Seleccione cuál de los siguientes enunciados es FALSO:
- a** Los mensajes enviados por los sockets DEALERS siempre deben incluir algún segmento vacío.
- b** Los sockets ROUTERS son asincrónicos.
- c** Los sockets ROUTERS utilizan identidades para manejar los mensajes que envían y reciben.
- d** Los sockets DEALERS pueden interoperar entre ellos.
- 16** Seleccione cuál de los siguientes enunciados es FALSO:
- a** Los sockets DEALERS son sincrónicos.
- b** Los sockets DEALERS encolan los mensajes que no han podido todavía enviar.
- c** Los sockets DEALERS encolan los mensajes recibidos todavía no procesados.
- d** Los sockets DEALERS pueden interoperar con sockets del tipo ROUTER.

- 17** La siguiente traza, en la que:
- un elemento de la forma $1xW2$ indica que el proceso (1) aplica sobre la variable (x) la operación de escritura (W) generando el valor (2)
 - un elemento de la forma $3yR1$ indica que el proceso (3) aplica sobre la variable (y) la operación de lectura (R), obteniendo el valor (1)

1xW1	2xR1	4xR1	3xR1	2xW2	1xW3	3xR3	4xR2	3xR2	4xR3
------	------	------	------	------	------	------	------	------	------

- a** Cumple consistencia estricta
- b** Cumple consistencia secuencial
- c** Cumple consistencia causal
- d** Todas las restantes son falsas
- 18** Si una determinada traza cumple consistencia secuencial
- a** No podemos asegurar que cumpla consistencia procesador
- b** No podemos asegurar que cumpla consistencia causal
- c** No podemos asegurar que cumpla consistencia caché
- d** Todas las restantes afirmaciones son falsas
- 19** En un sistema con operaciones no deterministas y sin errores bizantinos, y con duración de cada operación 10 veces el coste de propagar y aplicar modificaciones
- a** Debemos escoger replicación activa si desplegamos un único servicio
- b** Debemos escoger replicación pasiva únicamente si desplegamos más de un servicio
- c** Tanto la activa como la pasiva son válidas
- d** Debemos escoger replicación pasiva con independencia del número de servicios desplegados

- 20** *En un sistema sin operaciones no deterministas pero con posibilidad de errores bizantinos, y con duración de cada operación 10 veces el coste de propagar y aplicar modificaciones*
- a** Debemos escoger replicación pasiva
 - b** Debemos escoger replicación multi-master
 - c** Debemos escoger replicación activa exclusivamente si desplegamos un único servicio y replicación pasiva si desplegamos más de un servicio
 - d** Debemos escoger replicación activa con independencia del número de servicios desplegados
- 21** *¿Qué ocurre si en el modelo pasivo se permite que las réplicas secundarias respondan directamente a las operaciones de consulta enviadas por los clientes?*
- a** Se tolerarán los fallos arbitrarios.
 - b** Se toleran mejor las particiones en la red de intercomunicación.
 - c** No se podrán gestionar de manera consistente las modificaciones no deterministas.
 - d** Mejora el rendimiento de los servicios replicados de esa manera.
- 22** *Un almacén NoSQL es una base de datos en la que (entre otras posibles características):*
- a** No se utiliza el modelo relacional ni el soporte transaccional.
 - b** Se simplifica el esquema.
 - c** Se permite el particionado y el reparto de los datos entre diferentes nodos.
 - d** Cumple con lo especificado en los restantes enunciados.
- 23** *Según el teorema CAP, cuando ocurra una partición de la red ...*
- a** Todos los subgrupos continuarán respondiendo a sus clientes y los servicios replicados podrán mantener consistencia secuencial.
 - b** Los servicios replicados podrán respetar una consistencia causal si deben mantener su disponibilidad.
 - c** Se utilizará un modelo de partición primaria y todas las réplicas del servicio podrán responder a sus clientes.
 - d** No se permitirá ninguna actividad de los servicios distribuidos en ese sistema durante ese intervalo.
- 24** *¿Cuál de las siguientes afirmaciones sobre la replicación multi-máster es CIERTA?*
- a** Es capaz de gestionar más modelos de fallos que la replicación activa.
 - b** Necesita una difusión de orden total para garantizar la consistencia entre sus réplicas.
 - c** Permite que las instrucciones de cada operación solicitada sean ejecutadas por una sola réplica.
 - d** Requiere que las operaciones de lectura sean contestadas por todas sus réplicas para soportar el modelo de fallos arbitrarios (bizantinos).
- 25** *¿Qué tipo de replicación utiliza MongoDB?*
- a** Ninguna, pues consigue mejorar su escalabilidad mediante particionado/distribución de datos.
 - b** Multi-máster.
 - c** Pasiva mientras la red no esté particionada y multi-máster cuando haya particiones en la red.
 - d** Pasiva, pero admite en su configuración que las réplicas secundarias atiendan consultas.

- 26** *El módulo cluster de Node.js se utiliza para:*
- a** Desplegar un conjunto de programas Node.js en un cluster de ordenadores.
 - b** Gestionar múltiples hilos de ejecución en un proceso Node.js.
 - c** Gestionar un conjunto de procesos Node.js para que puedan compartir el programa a ejecutar y el puerto en el que recibir solicitudes.
 - d** Facilitar el desarrollo de proxies que propaguen peticiones a un cluster de ordenadores donde desplegar múltiples instancias servidoras.
- 27** *Durante el desarrollo de la práctica se crean imágenes con el mismo nombre correspondientes a distintas versiones del programa. Para evitar conflictos se recurre a:*
- a** Dejar las imágenes cuyo nombre coincide en directorios distintos
 - b** Borrar las imágenes anteriores antes de generar las nuevas
 - c** No hay posibilidad de conflicto, porque las imágenes nuevas reemplazan automáticamente a las preexistentes con el mismo nombre
 - d** No hay posibilidad de conflicto, porque el sistema elige siempre la versión con fecha más reciente
- 28** *En la variante CBW que acepta tipos de trabajos asumimos que:*
- el cliente tiene un código único que genera un tipo de trabajo (petición) u otro según un argumento en línea de órdenes
 - el worker tiene un código único que le habilita para un tipo de trabajo u otro según un argumento en línea de órdenes
- En ese escenario, y asumiendo que luego queremos controlar el número de instancias de cada tipo de cliente y de worker:*
- a** Podemos tener el código de los clientes en un único directorio, y el de los workers en un único directorio
 - b** Necesitamos un directorio distinto por tipo de worker, pero para los clientes es suficiente un único directorio
 - c** Necesitamos un directorio distinto por tipo de cliente, pero para los workers es suficiente un único directorio
 - d** Necesitamos un directorio distinto por tipo de cliente, y un directorio distinto por tipo de worker

29 En la variante CBW que acepta tipos de trabajos asumimos que:

- el cliente tiene un código único que genera un tipo de trabajo (petición) u otro según un argumento en línea de órdenes
- el worker tiene un código único que le habilita para un tipo de trabajo u otro según un argumento en línea de órdenes

En ese escenario, y asumiendo que luego queremos controlar el número de instancias de cada tipo de cliente y de worker:

- a** El fichero docker-compose.yml debe incluir un entrada distinta para cada tipo de cliente y una entrada distinta para cada tipo de worker
- b** El fichero docker-compose.yml debe incluir un entrada distinta para cada tipo de cliente, pero es suficiente una entrada única para los workers
- c** El fichero docker-compose.yml debe incluir un entrada distinta para cada tipo de worker, pero es suficiente una entrada única para los clientes
- d** En el fichero docker-compose.yml es suficiente una entrada única para todos los clientes, y otra entrada única para todos los workers

30 En el apartado de despliegue manual de la práctica 3 se tiene el siguiente Dockerfile para los clientes

```
FROM tsr2021/ubuntu-zmq
COPY ./2client.js myclient.js
CMD node myclient NEED_BROKER_URL
```

Sea imclient su imagen correspondiente. Considere el siguiente Dockerfile

```
FROM imclient
ENTRYPOINT [ "node" ]
CMD [ "myclient" ]
```

ubicado en un nuevo directorio, en el que ejecutamos

```
docker build -t nuevocliente .
```

En estas condiciones:

- a** El nuevo Dockerfile no está bien definido.
- b** La imagen nuevocliente no está adecuadamente construida.
- c** Con la imagen nuevocliente pueden construirse y ejecutarse clientes que pueden interoperar con el broker si reciben la información adecuada.
- d** Aunque con la imagen nuevocliente puedan generarse contenedores clientes, estos no tendrán manera de interoperar con el broker.

- 31** En la segunda parte de la tercera práctica se considera el uso de un logger en las condiciones allí presentadas. Considérese la siguiente variante: el logger solo recibirá anotaciones por parte del broker y se ha especificado que en el despliegue el primer servicio desplegado sea el broker. Para ello se han modificado adecuadamente los códigos del broker y del logger. Tomando en cuenta estas condiciones proponemos el siguiente `docker-compose.yml`

```
version: '2'
services:
  ...
  bro:
    image: broker
    build: ./broker/
    expose:
      - "9997"
      - "9998"
      - "9999"
  log:
    image: logger
    build: ./logger/
    links:
      - bro
    volumes:
      - /tmp/logger.log:/tmp/cbwlog
    environment:
      - LOGGER_DIR=/tmp/cbwlog
      # logger connects
      - BROKER_URL=tcp://bro:9997
```

En estas condiciones:

- a** Este despliegue no es factible y reportará un error.
- b** El sistema puede desplegarse pero al desplegar primero el broker el logger podría no recibir las primeras anotaciones.
- c** El sistema así desplegado funcionará adecuadamente.
- d** Para que el sistema funcionara adecuadamente habría que desplegar el logger antes que los clientes y los workers.

- 32** En la tercera sesión de la práctica 3 debe desplegarse un componente worcli que intercomunica dos sistemas cliente-broker-worker. Elija la afirmación correcta:

- a** Worcli debe desplegarse forzosamente en el anfitrión del primer broker.
- b** Worcli debe desplegarse forzosamente en el anfitrión del segundo broker.
- c** Un Worcli únicamente puede comunicar con otros Worcli.
- d** Worcli no es estrictamente necesario. Los brokers de esos dos sistemas, sin modificaciones, podrían comunicarse directamente sin ningún componente intermediario.



Rellena y entrega la siguiente hoja de respuestas. Cada cuestión posee una única respuesta correcta. No olvides cumplimentar correctamente tus datos personales.

No taches una posible respuesta incorrecta: bórrala o cúbrela con Tipp-Ex
Una cuestión con más de una respuesta marcada se considera no contestada

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

DNI: _____

Apellidos: _____

Nombre: _____

1	A	B	C	D
2	A	B	C	D
3	A	B	C	D
4	A	B	C	D
5	A	B	C	D
6	A	B	C	D
7	A	B	C	D
8	A	B	C	D
9	A	B	C	D
10	A	B	C	D
11	A	B	C	D
12	A	B	C	D
13	A	B	C	D
14	A	B	C	D
15	A	B	C	D
16	A	B	C	D
17	A	B	C	D
18	A	B	C	D
19	A	B	C	D
20	A	B	C	D

21	A	B	C	D
22	A	B	C	D
23	A	B	C	D
24	A	B	C	D
25	A	B	C	D
26	A	B	C	D
27	A	B	C	D
28	A	B	C	D
29	A	B	C	D
30	A	B	C	D
31	A	B	C	D
32	A	B	C	D

