TSR - Rec_Segon Parcial. 2025-01-30

Aquest examen consta de 17 qüestions, amb una puntuació total de 10 punts. Cada qüestió té 4 alternatives, de les quals únicament una és certa. La nota es calcula de la següent manera: després de descartar la pitjor qüestió, cada encert suma 10/16 punts, i cada error descompta 10/48 punts. Has de contestar en el full de respostes.

D

1. ***Aquest programa client es va utilitzar al Tema 3 per mostrar que el patró REQ/REP arribava a bloquejar les interaccions client/servidor quan un dels servidors (per exemple, el que utilitzara el port 8888) avortava després de rebre una petició, però abans de tornar la resposta corresponent. En aquest cas, el segon missatge no arribava a enviar-se al segon 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)
})
```

Què passaria si el servidor corresponent utilitzara sockets ROUTER en lloc de sockets REP (juntament amb la resta de canvis pertinents per rebre adequadament peticions i enviar correctament respostes) i es donara la mateixa situació de fallada?

A. La situació inicialment descrita a l'enunciat no es podria donar mai en utilitzar el patró REQ/REP
B. No s'observaria cap canvi: el client continuaria bloquejant-se
C. El client ja no es bloquejaria en aquesta situació
D. Hi hauria execucions on el client es bloquejaria i d'altres en què no

2. ***Aquest programa client es va utilitzar al Tema 3 per mostrar que el patró REQ/REP arribava a bloquejar les interaccions client/servidor quan un dels servidors (per exemple, el que utilitzara el port 8888) avortava després de rebre una petició, però abans de tornar la resposta corresponent. En aquest cas, el segon missatge no arribava a enviar-se al segon 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)
})
```

Què passaria si aquest client utilitzara sockets DEALER en lloc de sockets REQ (juntament amb la resta de canvis pertinents per enviar adequadament les peticions i rebre correctament les respostes) i es donara la mateixa situació de fallada?

A. El client ja no es bloquejaria en aquesta situació
B. Hi hauria execucions on el client es bloquejaria i d'altres en què no
C. La situació inicialment descrita a l'enunciat no es podria donar mai en utilitzar el patró REQ/REP
D. No s'observaria cap canvi: el client continuaria bloquejant-se

3. ***Quins avantatges aporta la màquina virtual quan se la compara amb el contenidor a l'hora de realitzar un desplegament?***
   - **A.** Menor consum de recursos
   - **B.** Més aïllament, juntament amb una menor dependència del sistema operatiu amfitrió
   - **C.** Fitxers de configuració més senzills, proporcionant major facilitat en el desplegament
   - **D.** Cap; el contenidor sempre serà una millor opció

4. ***Quants segments, i amb quin contingut, afegeix o espera (i elimina) un socket DEALER en enviar o rebre un missatge?***
   - **A.** Afegeix la identitat del socket emissor en enviar un missatge, i no espera, ni afegeix, ni elimina segments en rebre un missatge
   - **B.** Espera i utilitza, eliminant-lo, un segment amb la identitat del receptor en enviar un missatge, i afegeix la identitat del socket emissor en rebre un missatge
   - **C.** No afegeix, espera o elimina cap segment tant a l'operació d'enviament com a la de recepció
   - **D.** Afegeix un delimitador inicial en enviar un missatge, i espera i elimina un delimitador inicial en rebre un missatge

5. ***Quina etapa, d'entre les següents, del cicle de vida del programari no està inclosa en el desplegament?***
   - **A.** Anàlisi
   - **B.** Actualització
   - **C.** Configuració
   - **D.** Activació

6. ***Aquest és un exemple de Dockerfile utilitzat tant a la Pràctica 3 com al 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
```

   Seleccioneu l'afirmació veritable sobre el contingut d'aquest fitxer:
   - **A.** La primera línia indica que la imatge a generar es guardarà al directori `tsr-zmq`
   - **B.** Les instruccions `RUN` i `WORKDIR` utilitzades en aquest fitxer permeten que el fitxer `mybroker.js` localitze al lloc esperat el mòdul `tsr .js`
   - **C.** Allà on estiga aquest fitxer també hauran d'estar els fitxers `tsr.js` i `mybroker.js` perquè no hi haja errors en utilitzar aquest Dockerfile
   - **D.** Totes les afirmacions són certes

7. ***Un fitxer Dockerfile permet:***
   - **A.** Especificar quines instruccions s'han d'utilitzar per a construir una imatge que permetrà iniciar contenidors
   - **B.** Especificar quines operacions s'han d'utilitzar per desactivar, aturar i eliminar els contenidors usats al desplegament d'algun servei
   - **C.** Desplegar múltiples instàncies de diversos components relacionats en un mateix ordinador amfitrió
   - **D.** Especificar quines instruccions s'han d'utilitzar per generar una màquina virtual

8. ***Seleccioneu l'afirmació veritable sobre l'ordre*** `docker compose up`
   - **A.** Totes les afirmacions són certes
   - **B.** Pressuposa que hi haurà un fitxer `docker-compose.yml` al directori on siga utilitzada
   - **C.** Desplega un servei distribuït, iniciant una instància (és a dir, un contenidor) de cadascun dels seus components, en un determinat ordre
   - **D.** Permet utilitzar l'opció `--scale compo=X` per executar `X` instàncies del component `compo`

9. ***Considereu el següent exemple de fitxer `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"
```

Seleccioneu l'afirmació veritable sobre aquest fitxer:

A. Quan es desplega el servei descrit en aquest fitxer, el primer component que s'iniciarà serà `cc`

B. Quan es desplega el servei descrit en aquest fitxer, el primer component a iniciar serà `ca`, ja que s'ha especificat en primer lloc

C. El Dockerfile present al subdirectori `dirc` utilitza dues variables d'entorn anomenades `C_HOST` i `C_PORT`

D. Totes les altres afirmacions són falses

10. ***Seleccioneu l'afirmació veritable sobre la replicació activa:***

A. Pot oferir millor rendiment que la replicació passiva, especialment quan les operacions a gestionar generen modificacions de gran volum

B. Utilitza una rèplica primària i diverses secundàries

C. És el model de replicació utilitzat en el component que gestiona les dades persistents de la Wikipedia

D. Gestiona les operacions no deterministes sense generar inconsistència entre rèpliques

11. ***Seleccioneu l'afirmació veritable sobre la replicació passiva:***

A. Una vegada rebuda la petició a gestionar, genera menys interaccions entre les rèpliques que la replicació activa

B. Recuperar-se d'una fallada en la seua rèplica primària és més complex que quan falla una rèplica secundària

C. Ofereix millor rendiment que la replicació activa, especialment quan les seues operacions generen moltes modificacions de gran volum.

D. Ofereix millor rendiment que la replicació multimàster.

12. ***Suposeu un sistema format per tres processos P1, P2 i P3, on s'ha donat la següent execució: `W1(x)2, R2(x)2, W2(y)1, R1(y)1, R3(y)1, R3(x)2`. Aquesta execució respecta, entre altres, la consistència:***

A. Causal

B. 'Cache'

C. Estricta

D. Seqüencial

13. ***Seleccioneu l'afirmació veritable sobre les replicacions activa o multimàster:***

A. Resulta més adequat emprar replicació multimàster que replicació activa si volem tolerar fallades bizantines

B. La replicació multimàster és generalment més ràpida en recuperar els serveis després d'una fallada que la replicació activa

C. Resulta adequat emprar replicació multimàster si volem replicar serveis no deterministes

D. La replicació multimàster resulta menys adequada que la replicació activa per a entorns altament escalables

14. ***Seleccioneu l'afirmació veritable sobre el teorema CAP:***

A. Si tenim un sistema on s'adopta el model de partició primària i tenim alta consistència, estarem sacrificant la disponibilitat

B. Si tenim un sistema que segueix el model causal, estarem garantint consistència forta, per tant haurem de sacrificar suport a particions o disponibilitat

C. Tolerància a particions, tal com s'esmenta al teorema CAP, significa que tolerar particions és equivalent a emprar el model de partició primària

D. El teorema CAP indica que el més freqüent serà sacrificar la disponibilitat en sistemes altament escalables

15. ***L'ordre per esbrinar l'adreça IP d'un contenidor que proporciona un servei, després que s'haja iniciat és (suposant que ja coneixem l'ID o el nom del contenidor):***
    A. Totes les altres afirmacions són falses
    B. `docker ps`
    C. `docker inspect <ID>`
    D. `docker logs <ID>`

16. ***Suposem un sistema CBW al qual afegirem dos components `logger`, per recollir els missatges de traça generats al socket frontend i backend del broker respectivament. Per a fer això, el broker utilitzarà un socket PUB addicional, sobre el qual aplicarà un `bind` i amb el qual emetrà missatges amb dos segments, amb "frontend" o "backend" al primer. Els ports que ha d'utilitzar aquest broker es rebran des de la línia d'ordres: primer el del PUB, segon el del ROUTER de clients i tercer el del ROUTER per a treballadors. Si el Dockerfile inicial del broker té aquest contingut:***

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

Quines línies s'haurien d'afegir perquè gestionara adequadament aquesta nova configuració? (per maquetació, alguna línia es pot haver partit en dos)

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

17. ***Suposem un sistema CBW al qual afegirem dos components `logger`, per recollir els missatges de traça generats al socket frontend i backend del broker respectivament. Per a fer això, el broker utilitzarà un socket PUB addicional, sobre el qual aplicarà un `bind` i amb el qual emetrà missatges amb dos segments, amb "frontend" o "backend" al primer. Per la seua banda, els dos components `logger` comparteixen el mateix programa i la mateixa imatge. Cadascun se subscriurà a un prefix diferent. En base a aquest prefix, es generarà un nom de fitxer de log diferent. El directori on es guardaran aquests fitxers als seus contenidors serà el mateix per ambdòs: /tmp/cbwlog. El programa `logger.js` necessitarà rebre, en aquesta seqüència, aquests tres arguments des de la línia d'ordres:***
`hostDelBroker portDelBroker prefixAlQueSubscriure`***.***
Si el Dockerfile inicial del `logger` té aquest contingut:

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

Quines serien les línies que s'haurien d'afegir a aquest Dockerfile per a un correcte funcionament? (per maquetació, alguna línia es pot haver partit en dos)

A.
```
VOLUME /tmp/cbwlog
CMD node mylogger $BROKER_HOST $BROKER_PORT
$PREFIX
```
B.
```
VOLUME /tmp/cbwlog
EXPOSE 9997
CMD node mylogger 9997 /tmp/cbwlog/logs
```
C.
```
VOLUME /tmp/cbwlog
CMD node logger $BROKER_HOST $BROKER_PORT
$PREFIX
```
D.
```
VOLUME /tmp/cbwlog
CMD node logger localhost $BROKER_PORT
$PREFIX
```