
RA6. Implantació de solucions d'alta disponibilitat

Índex

1. Definició i objectius de l'alta disponibilitat	1
1.1 El concepte dels "nou"	1
1.2 Objectius concrets d'una solució HA	1
2. Anàlisi de configuracions d'alta disponibilitat	2
2.1 Funcionament ininterromput	2
2.2 Integritat de dades i recuperació de servei	2
2.3 Servidors redundants	2
2.4 Sistemes de clúster	3
2.5 Balanceig de càrrega	3
3. Maquinari per a la continuïtat del servei	3
4. Virtualització i alta disponibilitat	4
4.1 Per què virtualitzar per fer HA	4
4.2 Eines de virtualització	4
4.3 Alta disponibilitat de la virtualització vs. alta disponibilitat amb la virtualització	4
4.4 Entorn de pràctiques recomanat	5
5. Pràctica 1 --- Servidor redundant amb Keepalived (VRRP)	5
5.1 Escenari	5
5.2 Instal·lació (a totes dues VM)	6
5.3 Configuració a srv-web1 (MASTER)	6
5.4 Configuració a srv-web2 (BACKUP)	7
5.5 Arrencada i comprovació	7
5.6 Prova de failover	7
6. Pràctica 2 --- Balanceig de càrrega amb HAProxy	8
6.1 Escenari	8
6.2 Instal·lació	8
6.3 Configuració	8
6.4 Comprovació del repartiment	9
6.5 Panell d'estadístiques	9
6.6 Combinant les dues pràctiques	9
7. Pràctica 3 --- Emmagatzematge redundant amb DRBD	9
7.1 Concepte	9
7.2 Preparació (a totes dues VM)	10
7.3 Configuració del recurs	10
7.4 Inicialització (a totes dues VM)	10
7.5 Format i muntatge	11
7.6 Prova de commutació manual	11
8. Pràctica 4 --- Clúster de servidors amb Pacemaker i Corosync	11
8.1 Concepte	11
8.2 Instal·lació (a totes dues VM del clúster)	12
8.3 Autenticació entre nodes	12
8.4 Creació del clúster	12
8.5 Definició dels recursos: IP virtual + Apache	12

8.6 Agrupació de recursos i restriccions	13
8.7 Comprovació de l'estat del clúster	13
8.8 Pacemaker + DRBD (integració, per ampliar)	13
9. Solucions de futur per a sistemes amb demanda creixent	13
10. Activitat final --- Esquematització i documentació	14
10.1 Guió de treball	14
10.2 Supòsits proposats	14
10.3 Rúbrica orientativa de correcció	15
Resum d'ordres	15

1. Definició i objectius de l'alta disponibilitat

L'**alta disponibilitat** (*High Availability*, HA) és el conjunt de tècniques, arquitectures i procediments que permeten que un servei informàtic estigui operatiu el màxim de temps possible, minimitzant el temps d'aturada (*downtime*), tant planificat com no planificat.

L'objectiu no és eliminar les avaries ---cosa impossible---, sinó dissenyar el sistema perquè cap component individual (*single point of failure*, SPOF) pugui aturar el servei complet.

1.1 El concepte dels “nou”

La disponibilitat es mesura en percentatge de temps en funcionament durant un any:

Disponibilitat	Downtime anual	Downtime mensual
99% (“dos nous”)	3,65 dies	7,3 hores
99,9% (“tres nous”)	8,76 hores	43,8 minuts
99,99% (“quatre nous”)	52,6 minuts	4,4 minuts
99,999% (“cinc nous”)	5,26 minuts	26 segons

NOTA

Cada “nou” addicional multiplica per deu el cost d'implementació (maquinari redundant, xarxes duplicades, personal de guàrdia, etc.). Cal triar el nivell de disponibilitat en funció del cost real d'una aturada per a l'organització, no per moda tecnològica.

1.2 Objectius concrets d'una solució HA

- **Continuïtat de servei:** que l'usuari final no percebi (o percebi mínimament) una caiguda.
- **Tolerància a fallades:** el sistema absorbeix l'avaria d'un component sense aturar-se.
- **Recuperació ràpida:** si hi ha aturada, el temps de recuperació (RTO) és curt.
- **Integritat de dades:** cap transacció es perd (RPO baix o zero).
- **Escalabilitat:** el sistema pot créixer per absorbir més càrrega sense rediseny complet.

2. Anàlisi de configuracions d'alta disponibilitat

2.1 Funcionament ininterromput

Un servei “ininterromput” real no existeix; el que es persegueix és que les interrupcions siguin:

- **Transparents** per a l'usuari (failover automàtic).
- **De durada mínima** (segons, no minuts).
- **Sense pèrdua de dades.**

Per aconseguir-ho es combinen mecanismes de detecció de fallada (*health checks*, *heartbeat*) amb mecanismes de commutació automàtica (*failover*).

2.2 Integritat de dades i recuperació de servei

Dos indicadors clau que cal conèixer abans de dissenyar qualsevol arquitectura HA:

- **RPO (Recovery Point Objective):** quantitat màxima de dades que es poden perdre, mesurada en temps. Un RPO de 15 minuts significa que, en el pitjor cas, es perden les dades dels darrers 15 minuts.
- **RTO (Recovery Time Objective):** temps màxim acceptable per restablir el servei després d'una fallada.

La integritat de dades es garanteix amb rèpliques síncrones (sense pèrdua, però amb més latència) o asíncrones (menys latència, però amb risc de pèrdua de les últimes escriptures).

```
Esriptura síncrona:  Client → Node A → (espera confirmació) → Node B
                    ↪ → ACK → Client
Esriptura asíncrona: Client → Node A → ACK immediat → (rèplica en
                    ↪ segon pla) → Node B
```

2.3 Servidors redundants

Existeixen dos models bàsics de redundància de servidors:

- **Actiu-passiu:** un node atén tot el trànsit (actiu) i un altre roman en espera (passiu), sincronitzat i llest per assumir el servei si el primer falla. Senzill d'implementar, però desaprofita recursos del node passiu.
- **Actiu-actiu:** tots els nodes atenen trànsit simultàniament i es reparteixen la càrrega. Millor aprofitament de recursos, però més complexitat (cal repartiment de càrrega i, sovint, sincronització d'estat entre nodes).

A la pràctica 1 d'aquest tutorial implementarem un esquema actiu-passiu amb **Keepalived** i el protocol **VRRP**.

2.4 Sistemes de clúster

Un **clúster d'alta disponibilitat** és un conjunt de servidors (nodes) que treballen coordinats i es vigilen mútuament per garantir que un servei sempre s'executa en algun node del conjunt. Els components típics són:

- **Gestor de membres/comunicació** (p. ex. *Corosync*): manté els nodes informats de qui està viu.
- **Gestor de recursos** (p. ex. *Pacemaker*): decideix on s'executa cada servei i el mou (*failover*) si un node cau.
- **Fencing / STONITH** (*Shoot The Other Node In The Head*): mecanisme per aïllar o apagar forçosament un node que sembla mort, però que podria seguir escrivint dades (evita el problema de *split-brain*).

2.5 Balanceig de càrrega

El **balanceig de càrrega** (*load balancing*) reparteix les peticions entrants entre diversos servidors backend, aportant dos beneficis simultanis: repartiment de la càrrega de treball i tolerància a fallades (si un backend cau, el balancejador deixa d'enviar-hi trànsit).

Algorismes habituals: *round robin*, *least connections*, *source IP hash*, ponderat per capacitat (*weighted*).

A la pràctica 2 implementarem balanceig de càrrega HTTP amb **HAProxy**.

3. Maquinari per a la continuïtat del servei

Abans d'arribar a la capa de software, cal identificar quins components de maquinari eliminen punts únics de fallada:

Component	Solució de redundància
Font d'alimentació	Fonts redundants <i>hot-swap</i> (N+1) al servidor
Disc	RAID 1/5/6/10, controladores RAID redundants
Xarxa	Targetes de xarxa en <i>bonding/teaming</i> , dos commutadors, dos routers (VRRP/HSRP)
Alimentació elèctrica	SAI (UPS) + generador
Refrigeració	Sistemes de climatització redundants al CPD
Servidor complet	Clúster de servidors (nodes redundants)
Emmagatzematge centralitzat	Cabines SAN/NAS amb controladores dobles, RAID i rèplica

NOTA

La redundància de maquinari resol fallades físiques, però no protegeix de fallades lògiques (errors humans, corrupció de dades, ransomware). Per això la HA es complementa sempre amb còpies de seguretat, no la substitueix.

4. Virtualització i alta disponibilitat

4.1 Per què virtualitzar per fer HA

La virtualització desacobla el servei (màquina virtual) del maquinari físic que l'executa. Aquesta separació és clau per a la HA perquè permet:

- **Migrar en calent** (*live migration*) una màquina virtual d'un host físic a un altre sense aturar-la.
- **Reiniciar automàticament** una VM en un altre host si el host original cau.
- **Provar entorns complets** (clústers, balancejadors, servidors redundants) amb pocs recursos físics reals, cosa fonamental per als entorns de pràctiques d'aquest RA.
- **Encapsular l'estat** d'un servidor en fitxers (discs virtuals, snapshots) que es poden replicar fàcilment.

4.2 Eines de virtualització

Eina	Tipus	Ús típic en HA
Proxmox VE	Hipervisor tipus 1 (bare metal)	Clústers HA de producció i laboratori, ja treballat en tutorials anteriors
VMware vSphere/ESXi	Hipervisor tipus 1	HA i DRS en entorns empresarials
VirtualBox	Hipervisor tipus 2	Entorns de proves en un sol PC, ideal per practicar aquest RA sense maquinari dedicat
KVM/QEMU + libvirt	Hipervisor tipus 1 (Linux)	Base de moltes solucions HA de codi obert
LXC/Docker	Contenidors	Serveis lleugers, escalat horitzontal ràpid

4.3 Alta disponibilitat *de* la virtualització vs. alta disponibilitat *amb* la virtualització

Cal distingir dos nivells, que sovint es confonen:

1. **HA de la infraestructura de virtualització:** el mateix hipervisor (p. ex. un clúster Proxmox VE) detecta que un host físic ha caigut i arrenca automàticament les VM afectades en un altre host del clúster.
2. **HA implementada dins les VM:** independentment de si l'hipervisor té HA, es pot muntar un clúster de servei (p. ex. dos servidors web amb Keepalived) fent servir diverses màquines virtuals com si fossin servidors físics independents.

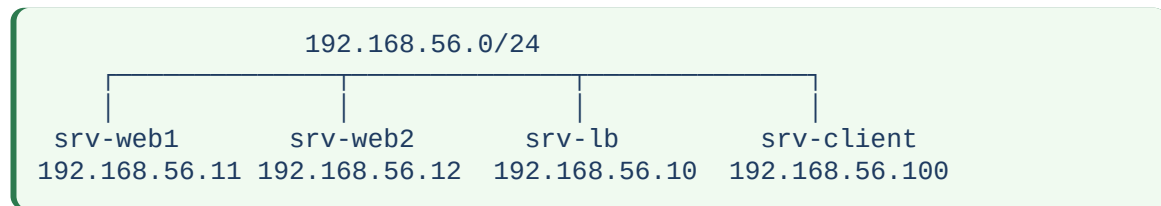
En aquest tutorial ens centrarem sobretot en el punt 2, ja que és el que permet **simular un entorn de producció complet dins d'un sol host físic o portàtil**, tal com demana el contingut 6.9 (*Simulació de serveis mitjançant virtualització*).

4.4 Entorn de pràctiques recomanat

Per seguir les pràctiques d'aquest tutorial necessitem com a mínim:

- Un hipervisor (VirtualBox és suficient per practicar; Proxmox VE si es disposa d'un host dedicat o niat).
- 3-4 màquines virtuals Debian/Ubuntu mínimes (1 vCPU, 512 MB-1 GB RAM, 8 GB disc) amb targeta de xarxa en mode *bridge* o xarxa interna comuna.
- Connectivitat entre totes les VM (mateixa subxarxa).

Esquema de xarxa que farem servir a totes les pràctiques (adaptable a la vostra xarxa d'aula):



5. Pràctica 1 --- Servidor redundant amb Keepalived (VRRP)

Objectiu: implantar un servidor redundant en esquema actiu-passiu, de manera que si el servidor principal cau, un segon servidor assumeixi automàticament la seva IP i el servei, sense intervenció manual.

5.1 Escenari

Dues VM (srv-web1 i srv-web2) amb Apache instal·lat, que comparteixen una **IP virtual** (VIP) 192.168.56.20. El protocol **VRRP** (*Virtual Router Redundancy Protocol*), implementat via **Keepalived**, decideix en cada moment quin dels dos nodes “posseeix” la VIP.

Màquina	IP real	Rol inicial
srv-web1	192.168.56.11	MASTER
srv-web2	192.168.56.12	BACKUP
VIP compartida	192.168.56.20	---

5.2 Instal·lació (a totes dues VM)

```
sudo apt update
sudo apt install -y keepalived apache2
```

Personalitzem la pàgina d'inici de cada node per poder identificar quin respon:

```
echo "Servidor: $(hostname)" | sudo tee /var/www/html/index.html
```

5.3 Configuració a srv-web1 (MASTER)

```
sudo nano /etc/keepalived/keepalived.conf
```

```
vrp_script chk_apache {
    script "/usr/bin/pgrep apache2"
    interval 2
    weight 2
}

vrp_instance VI_1 {
    state MASTER
    interface enp0s8
    virtual_router_id 51
    priority 150
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass thos2024
    }
    virtual_ipaddress {
        192.168.56.20/24
    }
    track_script {
        chk_apache
    }
}
```

5.4 Configuració a srv-web2 (BACKUP)

Mateix fitxer, amb dues diferències: state BACKUP i una priority més baixa.

```
vrrp_instance VI_1 {
    state BACKUP
    interface enp0s8
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass thos2024
    }
    virtual_ipaddress {
        192.168.56.20/24
    }
    track_script {
        chk_apache
    }
}
```

NOTA

El `virtual_router_id` ha de ser **idèntic** als dos nodes (identifica la instància VRRP); la `priority` és el que decideix qui guanya l'elecció a MASTER quan ambdós estan vius.

5.5 Arrencada i comprovació

```
sudo systemctl enable --now keepalived
ip addr show enp0s8 # comprovem que la VIP apareix només al node
↪ MASTER
```

Des de srv-client:

```
curl http://192.168.56.20
```

5.6 Prova de failover

El `vrrp_script` és clau: sense ell, Keepalived només detecta que la interfície de xarxa és viva, no que el servei Apache respon. Un servidor amb la xarxa activa, però Apache caigut seguiria sent “MASTER” sense aquest control addicional.

6. Pràctica 2 --- Balanceig de càrrega amb HAProxy

Objectiu: implantar un balancejador de càrrega a l'entrada de la xarxa interna que reparteixi peticions entre diversos servidors backend i n'exclougui automàticament els que no responen.

NOTA

Si ja heu seguit el tutorial específic d'HAProxy d'aquest curs, aquesta secció en repassa els conceptes aplicant-los directament al context d'aquest RA6; podeu ampliar-la amb la configuració SSL/TLS ja treballada allà.

6.1 Escenari

Una tercera VM, `srv-lb` (192.168.56.10), rep tot el trànsit HTTP dels clients i el reparteix entre `srv-web1` i `srv-web2`.

```
srv-client → srv-lb (192.168.56.10:80) → { srv-web1:80, srv-web2:80 }
```

6.2 Instal·lació

```
sudo apt update
sudo apt install -y haproxy
```

6.3 Configuració

```
sudo nano /etc/haproxy/haproxy.cfg
```

```
frontend web_frontend
    bind *:80
    default_backend web_backend

backend web_backend
    balance roundrobin
    option httpchk GET /
    http-check expect status 200
    server web1 192.168.56.11:80 check
    server web2 192.168.56.12:80 check

listen stats
    bind *:8404
    stats enable
    stats uri /stats
    stats refresh 10s
```

```
sudo systemctl restart haproxy
```

6.4 Comprovació del repartiment

```
for i in $(seq 1 6); do curl -s http://192.168.56.10; done
```

Hauríeu de veure alternar les respostes de `srv-web1` i `srv-web2` (algorisme *round robin*).

6.5 Panell d'estadístiques

Obriu `http://192.168.56.10:8404/stats` des del navegador: mostra l'estat (UP/DOWN) de cada backend en temps real, connexions actives i comptador d'errors.

6.6 Combinant les dues pràctiques

Noteu la diferència de rol respecte a la pràctica 1: **Keepalived/VRRP** proporciona redundància pel que fa a la IP (útil per a serveis que no admeten repartiment senzill, o per fer HA del propi balancejador), mentre que **HAProxy** proporciona repartiment de càrrega actiu-actiu real entre backends. En entorns de producció ambdues tècniques es combinen: dos HAProxy en actiu-passiu amb Keepalived, cadascun balancejant cap al mateix conjunt de servidors web.

7. Pràctica 3 --- Emmagatzematge redundant amb DRBD

Objectiu: implantar un sistema d'emmagatzematge redundant sobre servidors, de manera que les dades es mantinguin sincronitzades entre dos nodes sense necessitat d'una cabina SAN física.

7.1 Concepte

DRBD (*Distributed Replicated Block Device*) replica en temps real, a escala de bloc, el contingut d'un disc entre dos servidors, com un "RAID 1 per xarxa". El node primari (*Primary*) llegeix i escriu normalment; el secundari (*Secondary*) rep totes les escriptures i les aplica al seu propi disc.

```
srv-storage1 (Primary) <---- xarxa dedicada ----> srv-storage2
↔ (Secondary)
/dev/sdb1 ————— replicació síncrona ————— /dev/sdb1
```

7.2 Preparació (a totes dues VM)

Cada VM necessita un segon disc sense format (p. ex. /dev/sdb, 2 GB) i el paquet DRBD:

```
sudo apt update
sudo apt install -y drbd-utils
```

7.3 Configuració del recurs

```
sudo nano /etc/drbd.d/r0.res
```

```
resource r0 {
    protocol C;
    on srv-storage1 {
        device /dev/drbd0;
        disk /dev/sdb;
        address 192.168.56.21:7789;
        meta-disk internal;
    }
    on srv-storage2 {
        device /dev/drbd0;
        disk /dev/sdb;
        address 192.168.56.22:7789;
        meta-disk internal;
    }
}
```

NOTA

`protocol C` és una replicació **síncrona**: l'escriptura no es confirma a l'aplicació fins que ambdós nodes l'han desada. És la configuració amb RPO més baix (idealment zero), a costa de latència addicional en cada escriptura.

7.4 Inicialització (a totes dues VM)

```
sudo drbdadm create-md r0
sudo drbdadm up r0
```

Al node que farà de primari:

```
sudo drbdadm primary --force r0
```

Comprovem l'estat de sincronització:

```
sudo drbdadm status r0
# o bé
```

```
cat /proc/drbd
```

7.5 Format i muntatge

```
sudo mkfs.ext4 /dev/drbd0  
sudo mkdir -p /mnt/dades  
sudo mount /dev/drbd0 /mnt/dades  
echo "prova alta disponibilitat" | sudo tee /mnt/dades/prova.txt
```

7.6 Prova de commutació manual

DRBD per si sol només replica el bloc de dades; **no decideix automàticament** quin node ha de ser primari en cas de fallada. Per això, en entorns reals, DRBD es combina amb un gestor de clúster (Pacemaker) que automatitza aquesta commutació --- el que veurem tot seguit.

8. Pràctica 4 --- Clúster de servidors amb Pacemaker i Corosync

Objectiu: avaluar i implantar un sistema de clúster que augmenti la fiabilitat automatitzant la detecció de fallades i la migració de serveis.

8.1 Concepte

Un clúster Pacemaker/Corosync gestiona **recursos** (una IP virtual, un servei Apache, un dispositiu DRBD...) i els manté sempre actius en algun node del clúster, movent-los automàticament si el node que els executa falla.

- **Corosync:** capa de comunicació i pertinença al clúster (qui està viu).
- **Pacemaker:** gestor de recursos (què s'executa i on).
- **crmsh / pcs:** eines de línia d'ordres per administrar el clúster.

8.2 Instal·lació (a totes dues VM del clúster)

```
sudo apt update
sudo apt install -y pacemaker corosync pcs
sudo systemctl enable --now pcsd
```

8.3 Autenticació entre nodes

```
sudo passwd hacluster # mateixa contrasenya a tots dos nodes
sudo pcs host auth srv-web1 srv-web2 -u hacluster
```

8.4 Creació del clúster

Des de qualsevol dels dos nodes:

```
sudo pcs cluster setup cluster_thos srv-web1 srv-web2
sudo pcs cluster start --all
sudo pcs cluster enable --all
```

Com que només tenim dos nodes, desactivem el *quorum* estricte (en producció, amb 3+ nodes, no cal aquest pas):

```
sudo pcs property set no-quorum-policy=ignore
sudo pcs property set stonith-enabled=false
```

NOTA

Desactivar STONITH (*fencing*) només és acceptable en un **entorn de pràctiques**. En producció, l'absència de *fencing* pot provocar *split-brain*: dos nodes creient-se cadascun "el primari" i escrivint dades de manera incoherent. En un clúster real cal configurar un dispositiu de *fencing* (IPMI, controladora de la VM, PDU intel·ligent...).

8.5 Definició dels recursos: IP virtual + Apache

```
sudo pcs resource create vip ocf:heartbeat:IPaddr2 \
    ip=192.168.56.30 cidr_netmask=24 op monitor interval=5s

sudo pcs resource create web_service ocf:heartbeat:apache \
    configfile=/etc/apache2/apache2.conf \
    statusurl="http://localhost/server-status" op monitor interval=10s
```

8.6 Agrupació de recursos i restriccions

Volem que la IP i Apache viatgin sempre junts, en aquest ordre:

```
sudo pcs resource group add grup_web vip web_service
```

8.7 Comprovació de l'estat del clúster

```
sudo pcs status
```

Aquesta ordre mostra els nodes, el seu estat (Online/Offline), i en quin node s'està executant grup_web en aquest moment.

8.8 Pacemaker + DRBD (integració, per ampliar)

En una implantació completa, el recurs DRBD de la pràctica 3 també es gestionaria com a recurs de Pacemaker (ocf:linbit:drbd), de manera que la promoció a *Primary* i el muntatge del sistema de fitxers siguin automàtics i estiguin sincronitzats amb el moviment del servei Apache. És l'exercici natural d'ampliació d'aquest RA per a l'alumnat que vulgui aprofundir-hi.

9. Solucions de futur per a sistemes amb demanda creixent

Objectiu: analitzar com evolucionar una solució HA quan la demanda creix per sobre de la capacitat prevista inicialment.

Estratègia	Descripció	Quan aplicar-la
Escalat vertical (<i>scale up</i>)	Augmentar CPU/RAM/disc d'un node existent	Solució ràpida a curt termini; té un límit físic
Escalat horitzontal (<i>scale out</i>)	Afegir més nodes al clúster o al pool de backend	Solució que escala millor a llarg termini; requereix arquitectura <i>stateless</i> o amb estat compartit
Autoescalat	Afegir/treure nodes automàticament segons mètriques de càrrega (CPU, connexions)	Entorns amb demanda molt variable (pics estacionals, campanyes)
Migració a núvol públic o híbrid	Desplaçar part de la infraestructura a IaaS/PaaS	Quan el CPD propi no pot créixer més o cal disponibilitat geogràfica
CDN i càtcing	Reduir la càrrega real sobre els servidors d'origen	Contingut estàtic o poc canviant, trànsit d'usuaris geogràficament dispers
Bases de dades distribuïdes / replicades	Evitar que la BD esdevingui el nou coll d'ampolla	Quan els servidors d'aplicació ja escalen però la BD no
Contenidors i orquestració (Kubernetes)	Escalat horitzontal molt granular i automatitzat	Arquitectures de microserveis

10. Activitat final --- Esquematització i documentació

Objectiu: esquematitzar i documentar una solució d'alta disponibilitat per a un supòsit donat, integrant tot el que s'ha treballat en aquest RA.

10.1 Guió de treball

Per a cadascun dels supòsits següents (o un altre de proposat pel professorat), l'alumnat ha de lliurar un document que inclogui:

1. **Anàlisi del supòsit:** quins requisits de disponibilitat té l'organització (RTO/RPO admissibles, pressupost aproximat, horari crític).
2. **Esquema de xarxa i maquinari:** diagrama amb tots els nodes, IP, VIP i connexions, indicant quins components de maquinari són redundants.
3. **Justificació de les tècniques triades:** per què s'ha triat actiu-passiu, actiu-actiu, clúster, balanceig, etc., enfront d'alternatives.
4. **Configuració implantada:** captures o extractes de configuració de l'entorn de proves muntat en una màquina virtual, amb les proves de failover documentades (temps, comandes utilitzades, resultat).
5. **Pla d'evolució futura:** com creixeria la solució si la demanda s'incrementés.

10.2 Supòsits proposats

- **Supòsit A:** un institut vol garantir que el seu servidor intranet (horaris, notes) no caigui durant l'horari lectiu, però pot acceptar aturades curtes fora d'horari. Pressupost limitat.
- **Supòsit B:** una botiga en línia factura les 24 hores; una caiguda de 5 minuts en hora punta té un cost econòmic elevat. Cal RPO pràcticament zero a la base de dades de comandes.
- **Supòsit C:** un servei intern d'arxius (NAS) d'una empresa petita, on la prioritat és no perdre dades encara que el servei quedi aturat uns minuts en cas d'incidència.

NOTA

No hi ha una única solució "correcta" per a cada supòsit: l'objectiu de l'activitat és que l'alumnat justifiqui, amb criteris tècnics i econòmics, per què tria un nivell de redundància i unes tècniques concretes (servidor redundant, clúster, balanceig, emmagatzematge replicat) i no unes altres. Un excés d'HA per a un requisit modest és, tècnicament, tan mal disseny com un dèficit d'HA per a un servei crític.

10.3 Rúbrica orientativa de correcció

Criteri	Insuficient	Suficient	Excel·lent
Anàlisi del supòsit	No identifica requisits de RTO/RPO	Identifica RTO/RPO de manera genèrica	Justifica RTO/RPO amb dades concretes del supòsit
Esquema	Falta o incomplet	Mostra els nodes principals	Inclou xarxa, maquinari redundant i fluxos de dades
Implantació pràctica	No es prova el failover	Es prova el failover una vegada	Es documenten temps i es comparen diverses tècniques
Pla de futur	No es planteja escalat	Proposa una estratègia d'escalat	Prioritza diverses estratègies justificadament

Resum d'ordres

```
# Keepalived
sudo systemctl status keepalived
sudo journalctl -u keepalived -f

# HAProxy
sudo systemctl restart haproxy
sudo haproxy -c -f /etc/haproxy/haproxy.cfg # validar sintaxi

# DRBD
sudo drbdadm status r0
sudo drbdadm primary|secondary r0

# Pacemaker/Corosync
sudo pcs status
sudo pcs resource move <recurs> <node>
sudo pcs node standby|unstandby <node>
```

Versions d'aquest document

- [HTML](#)
- [PDF](#)
- [ODT](#)
- [MD](#)

[Domini Públic \(CC0\)](#)