Guia para la demo en vivo

Pasos cortos para mostrar tolerancia a fallos. Comandos en bloques y sin tildes para evitar lios de consola. Usa dos terminales si puedes (una para logs, otra para comandos).

0) Previa

- Docker y Docker Compose corriendo.
- Ubicate en la raiz del repo (`Proy_Unidad3_SistDistribuidos`).
- Puertos libres: 8000, 8003 y 8080.

1) Arranque limpio

Sube todo con build fresco (primer terminal):

```
docker compose up -d --build
```

Espera unos segundos y verifica contenedores/health:

```
docker compose ps
curl.exe http://localhost:8000/health
curl.exe http://localhost:8080/app1/health
curl.exe http://localhost:8080/app2/health
curl.exe http://localhost:8080/app3/health
```

Si app2 da 502, reiniciar load-balancer que para que tome las nuevas IP.

2) Escenario base (sin fallas)

Revisa stock inicial (producto 1):

```
curl.exe http://localhost:8080/app1/products
```

Crea una orden via middleware (PowerShell):

```
$body = @{product_id=1; qty=2} | ConvertTo-Json
Invoke-RestMethod -Method Post -Uri http://localhost:8000/orders -Body $body -
ContentType 'application/json'
```

Valida que se refleje en todos los servicios:

```
curl.exe http://localhost:8080/app1/products   # stock decremento
curl.exe http://localhost:8080/app2/orders     # si da 502, revisa con psql
(abajo)
docker compose exec postgres-primary psql -U orders -d ordersdb -c "select
id,product_id,qty,status from orders;"
docker compose exec mysql-primary mysql -uroot -proot -e "select
id,order_id,status,product_id,qty from reportsdb.reports;"
```

3) Caida de instancias de aplicacion (fallback a replicas B)

Simular que se caen las instancias A de App1 y App2:

```
docker compose stop app1-instance-a app2-instance-a
```

Enviar otra orden (uso ConvertTo-Json para evitar errores de comillas):

```
$body = @{product_id=1; qty=1} | ConvertTo-Json
Invoke-RestMethod -Method Post -Uri http://localhost:8000/orders -Body $body -
ContentType 'application/json' -TimeoutSec 20
```

Muestra consistencia de datos:

```
curl.exe http://localhost:8080/app1/products
docker compose exec postgres-primary psql -U orders -d ordersdb -c "select
id,product_id,qty,status from orders;"
docker compose exec mysql-primary mysql -uroot -proot -e "select
id,order_id,status,product_id,qty from reportsdb.reports;"
```

Opcional: levanta de nuevo las instancias A para quedar estable:

```
docker compose start app1-instance-a app2-instance-a
```

4) Caida de base de datos primaria (failover MariaDB)

Tiraré abajo el primario de MariaDB (App1):

```
docker compose stop mariadb-primary
```

Logs de promocion de replica/HAProxy (se mostrará en segunda terminal):

```
docker compose logs --tail=80 mariadb-watchdog mariadb-haproxy
```

Lanzar otra orden; debe responder 200 usando la replica promovida:

```
$body = @{product_id=1; qty=1} | ConvertTo-Json
Invoke-RestMethod -Method Post -Uri http://localhost:8000/orders -Body $body -
ContentType 'application/json' -TimeoutSec 20
```

Verifica datos y que el servicio siguio operando:

```
curl.exe http://localhost:8080/app1/products
docker compose exec postgres-primary psql -U orders -d ordersdb -c "select
id,product_id,qty,status from orders;"
docker compose exec mysql-primary mysql -uroot -proot -e "select
id,order_id,status,product_id,qty from reportsdb.reports;"
```

5) Latencia

```
docker run --rm --network proy_unidad3_sistdistribuidos_backend-net williamyeh/hey
`
  -H "Content-Type: application/json" -z 15s -c 10 `
  -m POST -d '{"product_id":1,"qty":1}' http://middleware:8000/orders
```

Revisa p95/p99 y compara con el SLO.

6) Guía rápida para prueba de fallo MySQL (App3)

- Salud previa (confirma servicios arriba): `curl http://localhost:8000/health` y `curl http://localhost:8003/health`.
- Caída del primario (simula fallo de DB): `docker compose stop mysql-primary`. -> Esperar 20-30s.
- Luego `docker compose exec mysql-replica mysqladmin ping -uroot -proot` hasta OK
- Solicitud de orden (valida continuidad): `Invoke-RestMethod -Method Post -Uri http://localhost:8000/orders -Body '{"product_id":1,"qty":1}' -ContentType 'application/json'` → 200 esperado usando la réplica promovida.
- Logs de failover (evidencia del cambio): `docker compose logs --tail=200 mysql-watchdog mysql-haproxy`.
- Volver a estado sano (recrear primario/réplica limpios): Para la prueba de fallo MySQL ahora:

```
docker compose stop mysql-replica mysql-primary mysql-watchdog mysql-haproxy

docker compose rm -f mysql-primary mysql-replica mysql-haproxy mysql-watchdog

docker volume rm proy_unidad3_sistdistribuidos_mysql_primary_data
proy_unidad3_sistdistribuidos_mysql_replica_data
```

```
docker compose up -d mysql-primary mysql-replica mysql-haproxy mysql-watchdog
```

7) Guía rápida para prueba de fallo MariaDB (App1)

- Salud previa: `curl http://localhost:8000/health` y `curl http://localhost:8080/app1/health` (vía LB).
- Caída del primario: `docker compose stop mariadb-primary`. -> Esperar 20-30s.
- Luego `docker compose exec mariadb-replica mysqladmin ping -uroot -proot` hasta OK
- Solicitud de orden (pasa por App1 para stock): mismo `Invoke-RestMethod` de arriba → debe responder 200 usando la réplica (HAProxy + watchdog).
- Logs de failover: `docker compose logs --tail=200 mariadb-watchdog mariadb-haproxy`.
- Volver a estado sano:

```
docker compose stop mariadb-replica mariadb-primary mariadb-watchdog mariadb-haproxy

docker compose rm -f mariadb-primary mariadb-replica mariadb-haproxy mariadb-watchdog

docker volume rm proy_unidad3_sistdistribuidos_mariadb_primary_data
proy_unidad3_sistdistribuidos_mariadb_replica_data

docker compose up -d mariadb-primary mariadb-replica mariadb-haproxy mariadb-watchdog
```

8) Guía rápida para prueba de fallo PostgreSQL (App2)

- Salud previa: `curl http://localhost:8000/health` y `curl http://localhost:8080/app2/health`.
- Caída del primario: `docker compose stop postgres-primary`. -> Esperar 20-30S
- Levantar App1 / App2 porque no tiene restart: `docker compose exec postgres-replica pg_isready -h postgres-replica -U orders -d ordersdb<`
- Luego `docker compose exec postgres-replica pg_isready -h postgres-replica -U orders -d ordersdb` hasta OK
- Solicitud de orden (toca App2/DB): usar el mismo `Invoke-RestMethod` → debe responder 200 con orden creada en la réplica promovida.
- Logs de failover: `docker compose logs --tail=200 postgres-watchdog postgres-haproxy`.
- Volver a estado sano:

```
docker compose stop postgres-replica postgres-primary postgres-watchdog postgres-haproxy

docker compose rm -f postgres-primary postgres-replica postgres-haproxy postgres-
watchdog

docker volume rm proy_unidad3_sistdistribuidos_postgres_primary_data
proy_unidad3_sistdistribuidos_postgres_replica_data

docker compose up -d postgres-primary postgres-replica postgres-haproxy postgres-
watchdog
```