# Taller - Kafka (3 Brokers) + Zookeeper

## Captura de estado inicial (podman ps)

Comando ejecutado:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Estado inicial del clúster con 4 contenedores en ejecución (zookeeper, kafka1, kafka2, kafka3).

## Topología del tópico (líder, réplicas, ISR)

Comandos ejecutados:

*podman exec -it kafka1 bash -lc '/opt/bitnami/kafka/bin/kafka-topics.sh --create --topic demo --bootstrap-server kafka1:29092 --partitions 3 --replication-factor 3'*

podman exec -it kafka1 bash -lc '/opt/bitnami/kafka/bin/kafka-topics.sh --describe --topic demo --bootstrap-server kafka1:29092'

Salida (captura del describe inicial):

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

La captura anterior también muestra el entorno, y a continuación se complementa con describe en la siguiente evidencia.

## Evidencia de producción/consumo

Comandos ejecutados:

# Productor (Terminal A)  
*podman exec -it kafka1 bash -lc '/opt/bitnami/kafka/bin/kafka-console-producer.sh --topic demo --bootstrap-server kafka1:29092'’*

# Consumidor (Terminal B)  
*podman exec -it kafka2 bash -lc '/opt/bitnami/kafka/bin/kafka-console-consumer.sh --topic demo --bootstrap-server kafka1:29092 --from-beginning'*

Texto

Descripción generada automáticamente

Producción y consumo de mensajes (productor en kafka1 y consumidor en kafka2).

## Failover (antes y después de apagar kafka3)

Acción y verificación:

*podman stop kafka3  
podman exec -it kafka1 bash -lc '/opt/bitnami/kafka/bin/kafka-topics.sh --describe --topic demo --bootstrap-server kafka1:29092'*

Resultados y explicación:

• El ISR se reduce (p. ej., de 3 → 2) al salir kafka3.  
• Los líderes permanecen disponibles y el clúster sigue operativo porque conserva quórum.  
• No hay pérdida de disponibilidad para producir/consumir.

Texto

Descripción generada automáticamente

Failover y recuperación: describe antes y después de detener/encender kafka3.

## Recuperación (encender kafka3 y verificar ISR)

Acción y verificación:

*podman start kafka3  
# esperar unos segundos a que sincronice  
podman exec -it kafka1 bash -lc '/opt/bitnami/kafka/bin/kafka-topics.sh --describe --topic demo --bootstrap-server kafka1:29092'*

Nota:

• Tras sincronizar, kafka3 vuelve al ISR y se recupera la redundancia (ISR = 3).  
• Si el describe no lo refleja de inmediato, esperar unos segundos y repetir.

# Conclusiones

Con RF=3 y min ISR=2, el clúster tolera la caída de 1 broker sin perder disponibilidad. La evidencia muestra continuidad de producción/consumo durante failover y la re-entrada al ISR tras la recuperación de kafka3.

**Errores comunes:**  
**Limpieza de contenedores/redes/volúmenes**

Comandos ejecutados:

*podman stop -a && podman rm -a -f && podman pod rm -a -f  
podman network prune -f && podman volume prune -f*

Salida relevante:

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**Interpretación:

• Podman forzó SIGKILL: es normal al detener contenedores “a la fuerza”.

• Los IDs listados confirman eliminación. Se listaron volúmenes del lab (si se prunen, se pierden datos).

**Compose down sin contenedores (esperado)**

Comandos ejecutados:

*cd ~/proyectos/kafka-lab  
podman-compose -f podman-compose-kafka-3brokers.yml down*

Salida relevante:

**Texto

Descripción generada automáticamente**Interpretación:

• Mensaje esperado porque los contenedores ya habían sido eliminados en el paso anterior.

**Verificación de puertos**

Comando ejecutado:

*ss -lntp | grep -E ':(2181|9092|9093|9094)\b' || echo "Puertos libres"*

Salida:

****

Interpretación: Ningún proceso ocupa 2181/9092/9093/9094 — ambiente listo para volver a levantar el clúster.

**Buenas prácticas**

• Ejecutar comandos siempre en Ubuntu (WSL2), no en PowerShell.

• Dentro de contenedores usar kafkaN:2909x; desde host usar localhost:909x.

• Preferir limpieza acotada al proyecto (compose down + network rm del proyecto); usar prune global solo si no importan los datos.

**--PRUEBA CON 5 BROKERS QUE SIGUE CON 3 RF—**

**Contexto**: el clúster tiene 5 brokers arriba, pero demo fue creado con **RF=3** y réplicas en **kafka1–kafka3**; por eso las salidas muestran solo 1–3.

*Texto

Descripción generada automáticamente*

**1) Apago kafka3 (quedan kafka1 y kafka2 vivos)**

Salida relevante:

***Topic: demo ... PartitionCount: 3 ReplicationFactor: 3 Configs: min.insync.replicas=2***

***P0 Leader: 2 Replicas: 3,2,1 Isr: 2,1***

***P1 Leader: 1 Replicas: 1,3,2 Isr: 1,2***

***P2 Leader: 2 Replicas: 2,1,3 Isr: 2,1***

Lectura:

* kafka3 sale del ISR (ya no aparece).
* Se mantiene quórum: **ISR=2** en todas las particiones (combinaciones de 1 y 2).
* La operación sigue **disponible** (lecturas/escrituras con acks=all aún pueden confirmarse porque min.insync.replicas=2).

**2. Apago también kafka2 (queda solo kafka1 vivo)**

Salida relevante:

*Topic: demo ... min.insync.replicas=2*

*P0 Leader: 1 Isr: 1*

*P1 Leader: 1 Isr: 1*

*P2 Leader: 1 Isr: 1*

Lectura:

* El ISR cae a **1** (solo el líder).
* Con min.insync.replicas=2, **no hay quórum de escritura**:
  + con **acks=all** → **fallan** las escrituras (no cumple ISR≥2).
  + con **acks=1** podrían aceptarse en el líder, pero **no es recomendable** (riesgo de pérdida si el líder cae antes de replicar).
* Las **lecturas** de lo ya escrito sí son posibles.

**3) Enciendo kafka2 y kafka3 (recuperación)**

**Salida relevante:**

*Topic: demo ... min.insync.replicas=2*

*P0 Leader: 1 Isr: 1,3*

*P1 Leader: 1 Isr: 1,3*

*P2 Leader: 1 Isr: 1,3*

Lectura:

* El líder permanece en **kafka1** (no hubo reelección necesaria).
* **kafka3** ya reingresó al ISR (sincronizó), pero kafka2 aún no; por eso se ve Isr: 1,3.
* Esto es normal: la re-entrada al ISR es gradual. Tras unos segundos/minutos (o con tráfico), **kafka2** terminará de ponerse al día y verás Isr: 1,2,3.

**Conclusión general:** con RF=3 y min.insync.replicas=2 el clúster tolera la caída de **1 broker** sin perder disponibilidad de escritura con acks=all; con la caída de **2 brokers** se pierde el quórum de escritura (queda sólo 1 en ISR). Al volver los nodos, el ISR **crece** conforme cada broker termina de sincronizar.

**PRUEBA CON 5 BROKERS REASIGNANDO EL MISMO TOPIC A LOS 5 (tópico demo, RF=5, min.insync.replicas=2)**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Estado pleno (5/5 en ISR)**

Comando previo: productor con acks=all y 200 mensajes.

**Describe:**

* **Partición 0** → Líder **1** · Réplicas **1,2,3,4,5** · **ISR: 5,1,2,3,4**
* **Partición 1** → Líder **2** · Réplicas **2,3,4,5,1** · **ISR: 5,1,2,3,4**
* **Partición 2** → Líder **3** · Réplicas **3,4,5,1,2** · **ISR: 5,1,2,3,4**

**Lectura:** RF=5 totalmente saludable; **todos los brokers (1..5) en ISR**. Líderes repartidos (1/2/3).

**Falla de 1 broker (apagado kafka5)**

**Describe:**

* ISR cae a **1,2,3,4** en todas las particiones; **líderes se mantienen** (1/2/3).

**Lectura:** con **ISR=4** y min.insync.replicas=2, las **escrituras con acks=all siguen OK** (quórum suficiente). No hubo reelección porque **no cayó el líder**.

**Falla de 2 brokers (apagado kafka4)**

**Describe:**

* ISR cae a **1,2,3** en todas las particiones; **líderes siguen** (1/2/3).

**Lectura:** con **ISR=3** aún hay más que suficiente quórum (**≥2**), por lo que **acks=all sigue funcionando**. Kafka mantiene los líderes al no fallar 1/2/3.

**Recuperación (encendido de kafka4 y kafka5)**

**Describe final:**

ISR vuelve a 5 miembros: 5,1,2,3,4 en todas las particiones.

Líderes permanecen (1/2/3).

Lectura: la re-entrada al ISR es automática tras sincronizarse las réplicas; no hubo cambio de líderes porque los líderes nunca cayeron.

**Conclusiones**

* Con **RF=5** y min.insync.replicas=2, el clúster **tolera la caída de 1 y 2 brokers** manteniendo escrituras con acks=all.
* La **elección de líder** solo ocurre si cae el propio líder de la partición; por eso los líderes **no cambiaron** en tus pruebas.
* El **orden** en Replicas/ISR no importa; lo clave es **quiénes están dentro** del ISR.
* Si se sube min.insync.replicas a **3**, seguiría escribiendo con 2 caídos (ISR=3), pero **fallará** si caen 3 (ISR<3), como debe ser para proteger consistencia.

NOTA: el error que allí aparece (imagen) probablemente es por la condición de carrera (el controlador aún no había propagado el estado tras start); al repetir el --describe segundos después, todo OK.