

Best Data Practices

Arquitectura MongoDB

Guía Completa de Alta Disponibilidad, Multi-Región y
Recuperación ante Desastres

Explorar Arquitecturas

Ver Estrategias de Backup

 Descargar como PDF

99.999%

Disponibilidad

<20ms

Latencia Global

0 min

RPO con PITR

3+

Proveedores Cloud

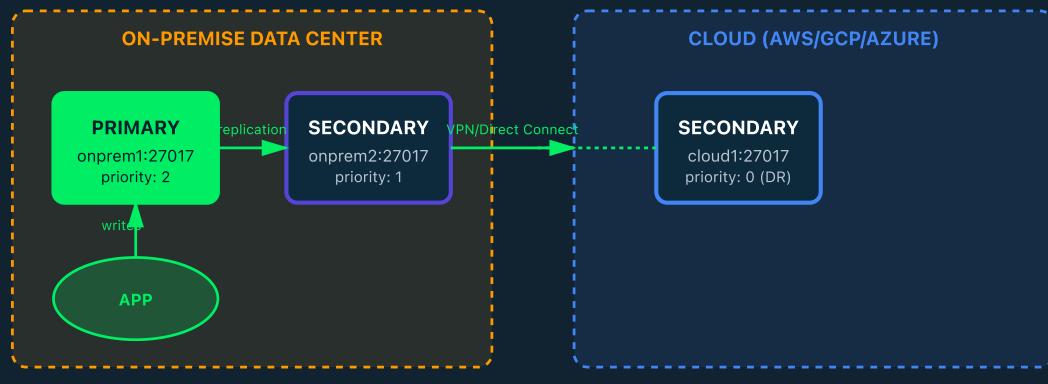
Arquitectura Híbrida

ReplicaSet Híbrido: On-Premise y Nube

Una estrategia robusta para alta disponibilidad y recuperación ante desastres extendiendo tu infraestructura entre centros de datos y la nube.



Arquitectura: ReplicaSet Híbrido



Latencia: < 20-30ms entre nodos

WriteConcern: w: "majority"

Failover: Automático a cloud si on-prem cae



Caso 1: Primario On-Premise

Topología: 2 nodos on-premise (primario + secundario) + 1 nodo en nube (DR)

Ventajas

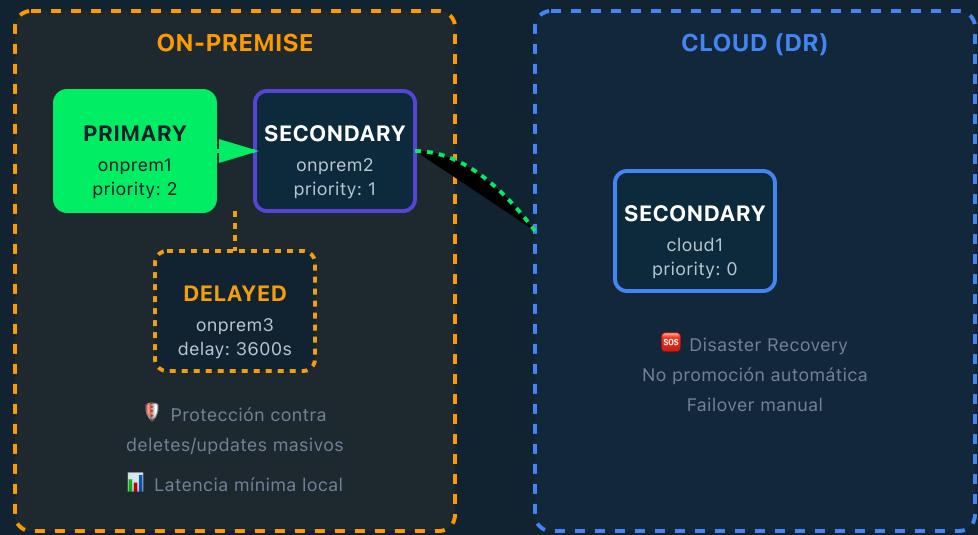
- ✓ Latencia mínima para operaciones locales
- ✓ Control total del entorno principal
- ✓ Protección contra fallos del data center

```
rs.conf().members = [ { _id:0, host:"onprem1:27017", priority:2 },
{ _id:1, host:"onprem2:27017", priority:1 }, { _id:2,
host:"cloud1:27017", priority:0 } ]
```



Patrón Recomendado con Prioridades

Configuración Óptima: Primario on-premise + DR en nube



Distribución Inteligente Híbrida

- **On-Premise:** PRIMARY priority: 2 + SECONDARY priority: 1 (workload principal local)
- **On-Premise Opcional:** 1 nodo hidden delayed (1-2h) para protección lógica
- **Nube (DR):** SECONDARY con priority: 0 (no se promueve automáticamente)

- **Failover:** Si data center cae → promover manualmente nodo de nube

⚠ Latencia Crítica: Inter-electores < 20-30 ms. Si supera, el primario y votos deben estar en nube con on-prem sin voto.

⚡ Puntos Críticos a Considerar

1. Latencia de Red

La latencia alta retrasa la replicación (replication lag) y afecta la consistencia de lecturas.

Solución: Conexiones dedicadas como AWS Direct Connect, Azure ExpressRoute o Google Cloud Interconnect. Evita VPN sobre internet público.

2. Costos de Transferencia

La replicación on-premise → nube (egress) genera costos significativos.

Monitoreo: Vigila el tráfico de replicación y optimiza la frecuencia de sincronización según tus necesidades.

3. Votos y Prioridades

Configuración correcta para elecciones predecibles.

```
// Nodo en nube para DR { priority: 0, hidden: true, tags: {role: "DR"} } //
Árbitro para desempate { arbiterOnly: true }
```

🔒 Seguridad y Conectividad

Networking

- TLS mTLS obligatorio entre nodos
- VPN o conexión dedicada (Direct Connect/ExpressRoute)
- VPC peering para tráfico privado
- Firewalls con reglas de least privilege

Configuración de Escritura

- **writeConcern**: { w: "majority", j: true }
- **readPreference**: nearest con hedged reads
- Retryable writes habilitado en drivers
- Cifrado en reposo con KMS/KMIP

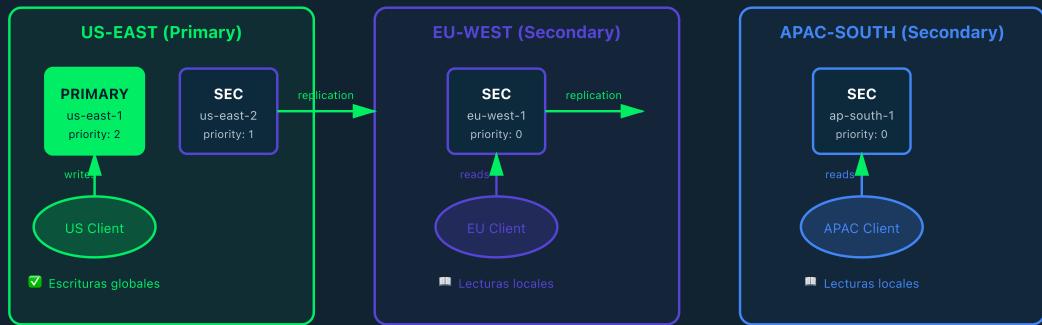
Arquitectura Global

Multi-Región y Multi-Nube

Arquitectura avanzada para aplicaciones globales con baja latencia mundial y resiliencia contra fallos regionales o de proveedor cloud.



Arquitectura Multi-Región



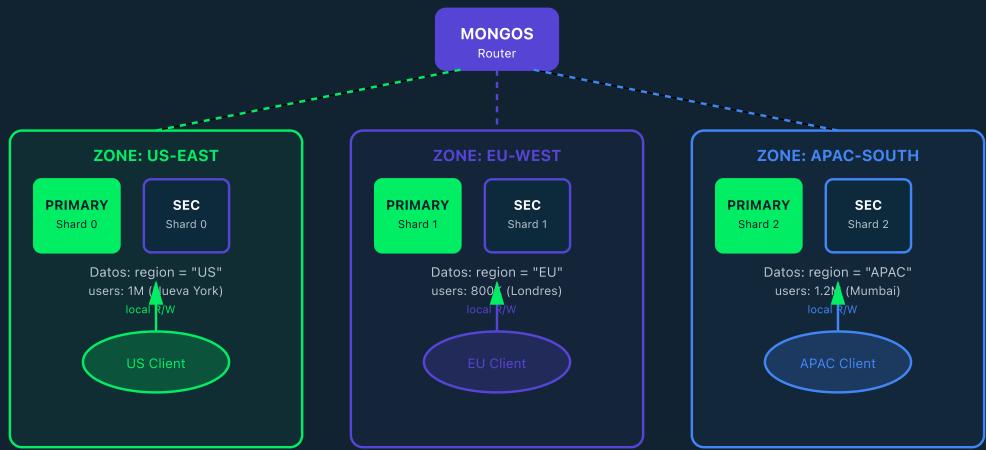
readPreference: nearest

Escrituras centralizadas

Lecturas distribuidas



Zone Sharding (Geo-Sharding)



Shard Key: {region: 1, customerId: 1}

Escrituras y lecturas LOCALES

Cumplimiento GDPR



Estrategias de Distribución Global

A) Multi-Región (Single ReplicaSet)

Use Case: Baja latencia de lectura global con escrituras centralizadas

Setup

- Región primaria: 3 electores
- Regiones secundarias: réplicas hidden o baja prioridad
- Clientes: `readPreference: nearest + tagSets` por región

Pros:

- ✓ Simplicidad operativa
- ✓ RPO ≈ 0, RTO bajo
- ✓ Consistencia fuerte garantizada

Contras:

- ⚠ Escrituras viajan a región primaria
- ⚠ Latencia de write = RTT a primario

B) Sharding Geográfico (Zone Sharding)

Use Case: Escrituras locales por geografía (GDPR, latencias mínimas, escalado horizontal)

Strategy

- Shard-key con dimensión geográfica: {"region": 1, "customerId": 1}
- Zonas definidas: us-east, eu-west, apac-south
- Cada zona vive en su región física
- Lecturas y escrituras quedan locales

```
// Definir zonas sh.addShardTag("shard0", "US-EAST")
sh.addShardTag("shard1", "EU-WEST") // Asignar rangos sh.addTagRange(
"mydb.users", {region: "US", userId: MinKey}, {region: "US", userId:
MaxKey}, "US-EAST" )
```

Pros:

- ✓ Latencia de write local
- ✓ Cumplimiento de residencia de datos
- ✓ Escalado horizontal ilimitado



MongoDB Atlas Global Clusters

Arquitectura más sofisticada basada en **geo-sharding** automático para mantener datos cerca de usuarios globalmente.



Zonas Geográficas

Cada zona representa una región geográfica asociada a un shard:

- **Europa Occidental:** UK, Alemania, Francia
- **US-EAST:** Virginia, Ohio
- **APAC-SOUTH:** Mumbai, Singapur

Mapeo Inteligente: Los datos de usuarios españoles van a la zona Europa Occidental; mexicanos a Norteamérica Central.

🔑 La Clave del Éxito: Shard Key

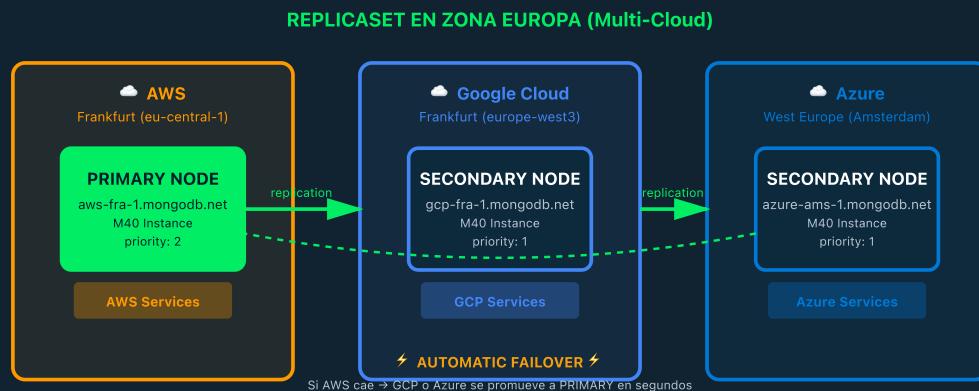
Shard key con dimensión geográfica para enrutamiento inteligente:

```
{ location: "madrid", country_code: "ES", user_region: "eu-west" } // El router (mongos) envía // escrituras directamente al // shard geográficamente cercano
```

Resultado: Usuario en Londres escribe en datacenter de Londres/Dublín, no en Virginia. Latencia mínima.

Cloud Componente Multi-Nube

Cada ReplicaSet distribuido entre diferentes proveedores cloud para máxima resiliencia:



Ejemplo: ReplicaSet en Zona Europa



Nodo Primario

AWS Frankfurt (eu-central-1)



Secundario 1

Google Cloud Frankfurt (europe-west3)



Secundario 2

Azure West Europe (Amsterdam)

Beneficios Multi-Nube

- ✓ **Resiliencia Máxima:** Failover automático si AWS cae completo en Frankfurt
- ✓ **Sin Vendor Lock-in:** No atado a un solo proveedor
- ✓ **Mejor Red:** Aprovecha la conectividad óptima de cada cloud



Resumen de Beneficios Clave en Atlas

Beneficio	Cómo lo logra Atlas	Impacto
Baja Latencia Global	Geo-Sharding + Zonas + Read Preference nearest	Experiencia de usuario super rápida sin importar ubicación
Alta Disponibilidad Extrema	ReplicaSets Multi-AZ + Failover inter-regional y multi-nube	SLAs de 99.999% y supervivencia a desastres mayores

Beneficio	Cómo lo logra Atlas	Impacto
Soberanía de Datos	Control granular sobre ubicación de datos por zona	Cumplimiento simplificado de GDPR y regulaciones
Simplicidad Operacional	Automatización completa de provisión, red y monitoreo	Ahorro de semanas/meses de trabajo de ingeniería

Consideraciones Clave para Arquitectura Global

Consistencia vs Latencia

Define prioridades con `writeConcern`:

- `w: "majority"` en clúster geodistribuido será lento
- Usa `w: 1` o `majority local` para escrituras rápidas
- Acepta pequeña ventana de posible pérdida si región cae completa

Complejidad de Red

Conexión entre nubes requiere:

- VPC/VNet peering entre proveedores
- Soluciones de red multi-nube
- Comunicación segura y baja latencia
- Configuración de rutas y security groups

Shard Key Critical

Decisión más importante del diseño:

- Mala shard key causa "hotspots"
- Requiere alta cardinalidad
- Debe alinearse con patrón de acceso geográfico
- Plan de re-shard si patrones cambian

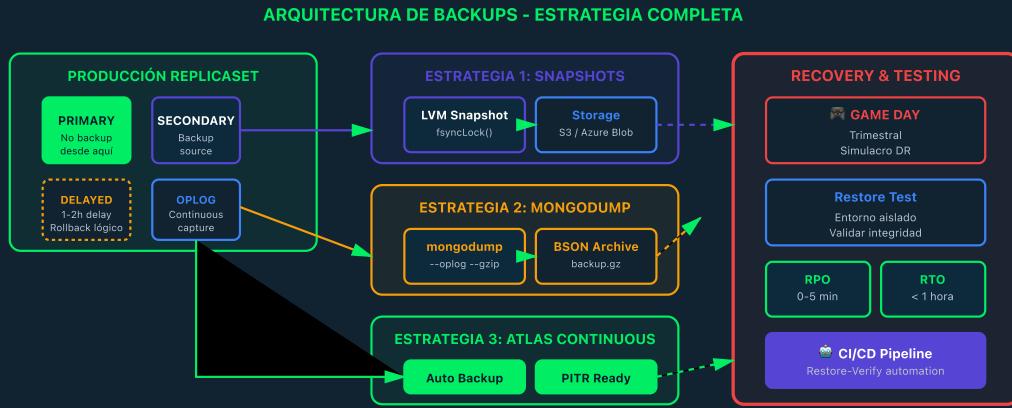
Recuperación de Datos

Backups y Pruebas de Restauración

"Un backup que no ha sido probado, no es un backup. Es solo una esperanza."



Estrategias de Backup



3 Estrategias Complementarias

Testing Obligatorio

Automatización CI/CD

1

Snapshots del Sistema de Archivos

Método: LVM en Linux o tecnologías similares para snapshots atómicos

Proceso

1. Detener balanceador (si sharded)
2. Ejecutar `fsyncLock()` en secundario
3. Tomar snapshot a nivel de filesystem
4. Ejecutar `fsyncUnlock()`
5. Copiar a almacenamiento (S3, Azure Blob)

Ventaja: Rápido y consistente

2

mongodump

Método: Utilidad nativa de MongoDB para volcado lógico en BSON

```
mongodump \ --uri="mongodb://..." \ --oplog \ --gzip \ --
archive="backup.gz"
```

Uso Ideal

- Backups pequeños o migraciones
- Colecciones específicas
- Siempre desde nodo secundario

⚠ Nota: --oplog captura operaciones durante dump para Point-in-Time Recovery (PITR)

3

Soluciones Gestionadas

Método: MongoDB Atlas / Ops Manager

Características Atlas

- **Backups continuos:** Snapshots + oplogs
- **PITR:** Restaurar a cualquier minuto específico
- **Automatización:** Sin intervención manual
- **Retención:** Configurable (14-35+ días)

🏆 **Mejor opción:** Para producción, las soluciones gestionadas ofrecen el mejor balance de seguridad y facilidad.

🎯 Políticas de Recuperación (RPO/RTO)

RPO (Recovery Point Objective)

Pérdida de datos aceptable

- **Productivo:** 0-5 minutos con PITR
- **Analytics:** Puede tolerar horas
- **Dev/Test:** 24 horas o más

0-5 min

RPO Objetivo Producción

RTO (Recovery Time Objective)

Tiempo de vuelta aceptable

- **Core:** < 1 hora
- **Sistemas periféricos:** 2-4 horas
- **Reportes:** 24 horas

<1h

RTO Objetivo Core

Retención Recomendada

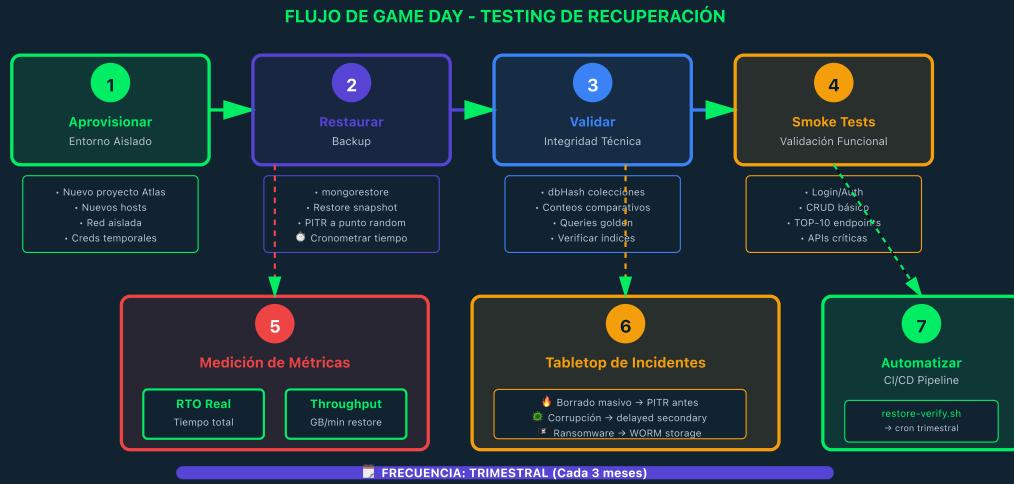
Define retención según compliance y legal:

- **Productivo:** 30-90 días mínimo
- **Compliance/Legal:** 365 días o más
- **Evita:** Retenciones infinitas sin costo-beneficio



Backup Testing - Game Days Trimestrales

El músculo que casi nadie entrena. La copia es el 50% del problema; el otro 50% es poder restaurar a tiempo lo que necesitas.



1

Restauración Real a Entorno Aislado

Nunca restaures sobre producción. Aprovisiona:

- Nuevo proyecto Atlas o nuevos hosts
- Configuración de red aislada
- Credenciales temporales separadas

2

Validación Técnica

```
// dbHash por colección crítica db.runCommand({dbHash: 1}) //
Conteos comparativos db.users.countDocuments()
```

```
db.orders.countDocuments() // Verificar índices  
db.collection.getIndexes()
```

Validaciones clave:

- dbHash por colección crítica
- Conteos comparativos
- Queries "golden" dan mismos resultados
- Integridad de índices (rebuild si es necesario)

3

Validación Funcional (Smoke Tests)

- Pruebas de login y autenticación
- Transacciones simples (CRUD básico)
- Endpoints TOP-10 de negocio
- Integraciones críticas (APIs externas)

4

Medición de Métricas

RTO Real

Tiempo total de restauración

Throughput

GB/min de restauración

Cuellos de Botella

Red / IO / CPU limitantes

Errores

Problemas encontrados

5

Tabletop de Incidentes

Simula escenarios reales:

Escenario 1: Borrado Lógico Masivo

Restaura desde PITR minutos antes del evento

Escenario 2: Corrupción de App

Usa delayed secondary para capturar el "antes"

Escenario 3: Ransomware

Restaura desde snapshot en almacenamiento WORM

Escenario 4: Pérdida Regional

Failover a región secundaria y verificación

6

Automatizar Restore-Verify (CI)

```
#!/bin/bash # Script automatizado de restore-verify # 1. Crear  
cluster temporal create_temp_cluster() # 2. Restaurar  
snapshot/PITR restore_backup() # 3. Correr suite de checks
```

```
run_validation_suite() # 4. Generar reporte generate_report() #  
5. Destruir recursos temporales cleanup()
```



Controles de Prevención

Prevención de Pérdida de Datos

- TTL con expireAfterSeconds en vez de cronos peligrosos
- Change Streams + auditoría para anomalías tempranas
- Privilegios mínimos; producción sin dropCollection para apps
- Validadores de esquema para integridad de datos

Monitoreo y Alertas

- Alertas de replication lag
- Monitoreo de tamaño de oplog
- Métricas de backup success/failure
- Alertas de operaciones masivas de delete/update

Atlas vs Self-Managed

¿Por Qué No Puedes Mezclar Atlas con On-Premise?

⚠ Limitación Importante

MongoDB Atlas no permite añadir nodos on-premise directamente a sus ReplicaSets o clústeres sharded.

¿Por Qué No Se Puede?

Control de Automatización

Atlas depende de tener control total sobre todos los nodos:

- No podría reemplazar automáticamente un nodo externo si falla
- No puede aplicar parches de seguridad coordinados
- Pierde capacidad de garantizar salud y consistencia del clúster
- Rompe el modelo de automatización end-to-end

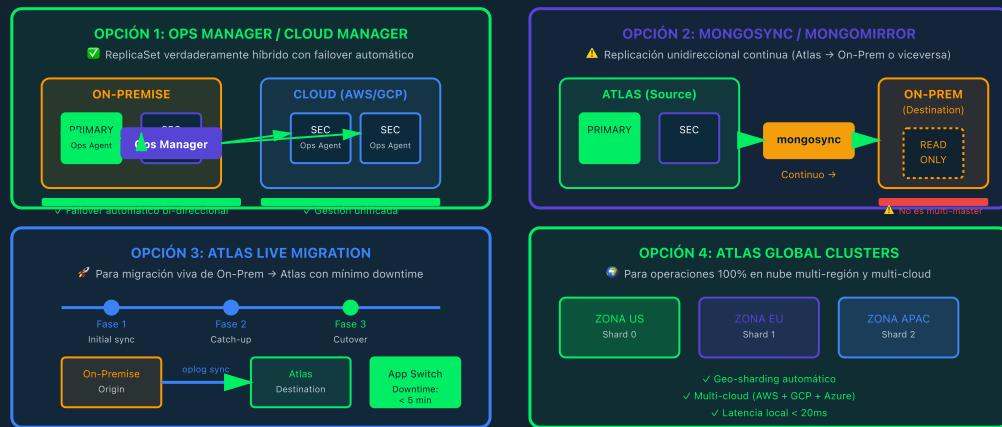
Seguridad y Red

Atlas opera en red virtual segura preconfigurada:

- VPC/VNet gestionada y aislada
- Integrar nodo externo rompería modelo de seguridad
- Complejidad de conectividad no gestionable automáticamente
- Imposible garantizar SLAs de disponibilidad

✓ Alternativas y Soluciones Correctas

COMPARACIÓN: SOLUCIONES HÍBRIDAS ON-PREMISE ⇔ ATLAS



1. MongoDB Ops Manager / Cloud Manager

La solución diseñada para clústeres híbridos

¿Qué es?

"Atlas, pero para tus propios servidores". Es el mismo software de automatización que usa Atlas, pero instalado en tu infraestructura.

¿Cómo funciona?

- Instalas agente de MongoDB en tus servidores on-premise y en nube
- Desde consola de Ops Manager: creas, gestionas y monitorean ReplicaSets híbridos
- Nodos distribuidos entre on-premise y nube con gestión unificada
- Misma potencia de automatización que Atlas

Ventaja: Control total y gestión unificada para un verdadero clúster híbrido



2. Replicación Unidireccional con mongomirror

Para copiar datos de Atlas a on-premise (o viceversa)

¿Qué es?

Utilidad que permite replicar datos de un clúster de MongoDB a otro continuamente.

¿Cómo funciona?

- Se conecta a tu clúster Atlas como fuente
- Clúster on-premise como destino
- Sincroniza continuamente los datos

Limitación importante: Replicación en una sola dirección. El clúster on-premise sería copia de solo lectura. No es HA activa-activa.

Casos de uso: Analítica, reportes, backup adicional on-premise



3. Cluster-to-Cluster Sync (mongosync)

Sincronización continua near-real-time

Cuándo usar

- DR activo-pasivo entre on-prem y Atlas
- Feeds analíticos continuos
- Migraciones prolongadas con rollback plan

⚠ Importante: El destino debe ser read-only para colecciones sincronizadas (evita conflictos)



Resumen: ¿Qué Usar y Cuándo?

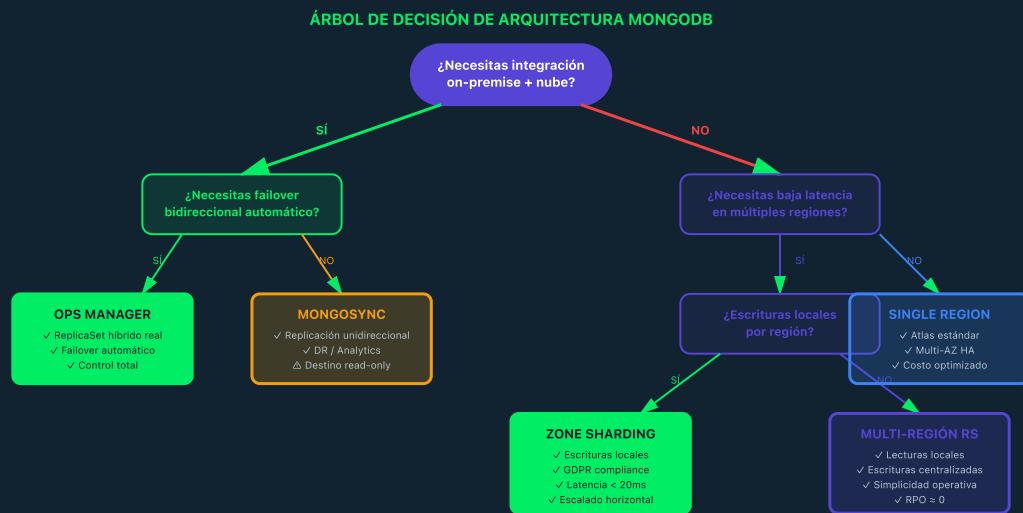
Objetivo	Solución Recomendada	Características
Alta Disponibilidad / DR Híbrido	MongoDB Ops Manager o Cloud Manager	ReplicaSet verdaderamente híbrido con failover automático
Copia de datos Atlas → On-Premise	mongomirror o mongosync	Replicación unidireccional continua, destino read-only
Gestión 100% en nube multi-región/nube	MongoDB Atlas Global Clusters	Geo-sharding automático, multi-cloud, gestión completa
Migración viva on-prem → Atlas	Atlas Live Migration	Migración online con mínimo downtime, cambio controlado

Para Arquitectos

Checklist Operativo

"Para que tu yo del futuro te aplauda"

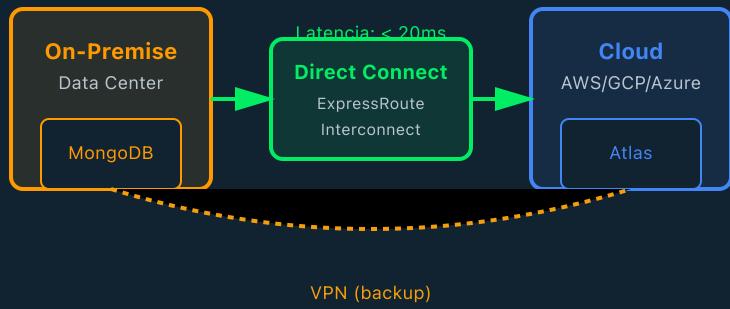
🎯 Árbol de Decisión: ¿Qué Arquitectura Elegir?



✓ Checklist Visual Completo

ARQUITECTURA	BACKUPS	SEGURIDAD	OBSERVABILIDAD
<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Latencia medida (p50/p95/p99) <input checked="" type="checkbox"/> w: majority, j: true <input checked="" type="checkbox"/> Prioridades/tags por región <input checked="" type="checkbox"/> Shard-key validada <input checked="" type="checkbox"/> Plan de re-shard 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> PITR activo <input checked="" type="checkbox"/> Retención documentada <input checked="" type="checkbox"/> Restores trimestrales <input checked="" type="checkbox"/> RPO/RTO medidos <input checked="" type="checkbox"/> Runbooks actualizados 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> TLS mTLS entre nodos <input checked="" type="checkbox"/> KMS/KMIP cifrado <input checked="" type="checkbox"/> Rotación de secretos <input checked="" type="checkbox"/> RBAC mínimo privilegio <input checked="" type="checkbox"/> Auditoría habilitada 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Profiler por muestreo <input checked="" type="checkbox"/> FTDC/metrics centralizadas <input checked="" type="checkbox"/> Alertas replication lag <input checked="" type="checkbox"/> Opcounters, cache hit <input checked="" type="checkbox"/> Dashboard métricas clave
NETWORKING & CONECTIVIDAD		RUNBOOKS & DOCUMENTACIÓN	
<ul style="list-style-type: none"> <input checked="" type="checkbox"/> VPN/Direct Connect configurado <input checked="" type="checkbox"/> VPC Peering activo <input checked="" type="checkbox"/> Private Endpoints configurados <input checked="" type="checkbox"/> Firewall rules least privilege <input checked="" type="checkbox"/> Latencia inter-nodos < 30ms 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> DNS resolution validado <input checked="" type="checkbox"/> IP Access List actualizada <input checked="" type="checkbox"/> BGP routes configuradas <input checked="" type="checkbox"/> NACLs/Security Groups <input checked="" type="checkbox"/> Bandwidth monitoring 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Failover procedure documentado <input checked="" type="checkbox"/> Rollback plan actualizado <input checked="" type="checkbox"/> Incidentes 3 a.m. cubiertos <input checked="" type="checkbox"/> Contactos on-call definidos <input checked="" type="checkbox"/> Escalation path claro 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Arquitectura diagramada <input checked="" type="checkbox"/> Configs versionadas (IaC) <input checked="" type="checkbox"/> Change log mantenido <input checked="" type="checkbox"/> Postmortem template <input checked="" type="checkbox"/> Conocimiento compartido

🌐 Topología de Red



- Conexión dedicada para replicación
- VPN como backup redundante
- Cifrado TLS en tránsito

Métricas Clave a Monitorear

Replication Lag

Alerta: > 10s

Oplog Window

Alerta: < 2h

Cache Hit Ratio

Target: > 95%

Connections

Alerta: > 80%

Disk I/O

Monitor IOPS

CPU Usage

Alerta: > 75%

Frases de Arquitecto



"No es alta disponibilidad si no has medido el RTO."



"La copia que no restauras es un souvenir caro."



"Latencia manda la topología; compliance manda la shard-key."

Best Data Practices

Arquitectura MongoDB para Latinoamérica - Guía completa de patrones de alta disponibilidad, multi-región y recuperación ante desastres.

© 2025 BDP. Todos los derechos reservados.