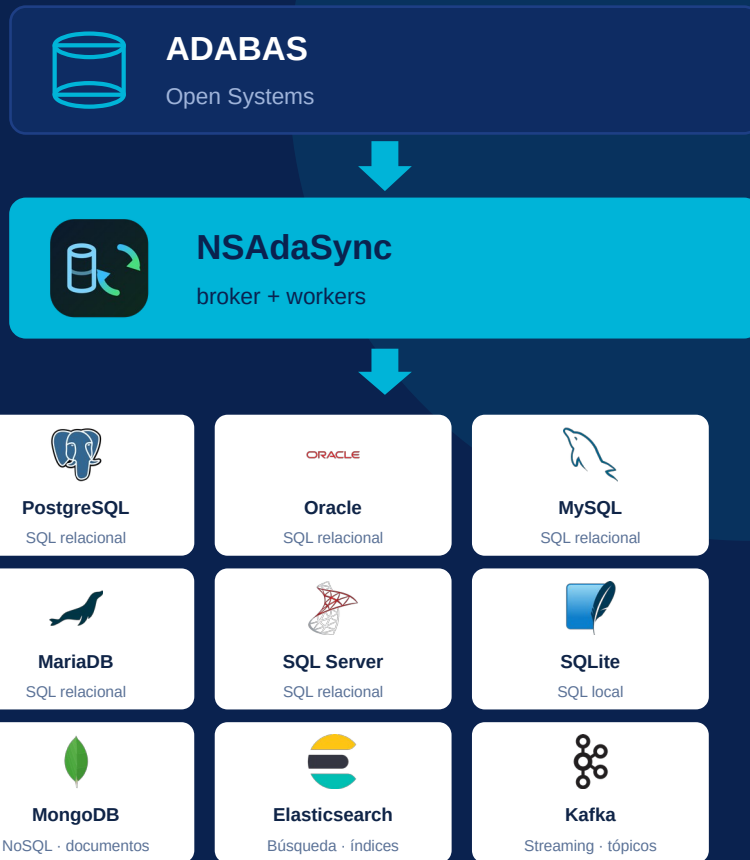


El dato de Adabas, en tiempo real y donde lo necesitas

NSAdaSync replica cada cambio de tus ficheros Adabas a PostgreSQL, Oracle, MySQL, MongoDB, Elasticsearch o Kafka en el momento en que ocurre.

[VER FUNCIONALIDADES →](#)

9 destinos · Del núcleo a tus datos en segundos



...y un destino más cuando lo necesites

Su dato Adabas no llega a donde se consulta

01

Consumo por lotes

Los informes SQL se alimentan de cargas nocturnas: cuando se consulta, el dato ya está obsoleto.

02

Sin streaming

Oracle, MongoDB, Elasticsearch o Kafka no reciben los cambios del núcleo en el momento en que ocurren.

03

Sin control

No sabe qué se ha replicado, qué está en cola ni qué falló hasta que lo consulta a mano, fichero a fichero.

- Horas de retraso en el dato

- Reportes sobre instantáneas

- Cero visibilidad del flujo

QUÉ ES

Réplica de Adabas en tiempo real, para SQL, NoSQL y streaming

NSAdaSync es la réplica de Adabas de Nacarsoft. Un exit en el núcleo publica cada cambio en un broker persistente y los workers de cada suscripción lo aplican en el destino: SQL, NoSQL, búsqueda o streaming.

- ✓ Exit UEX4 mínimo: no bloquea, no reintenta, no toca la red
- ✓ Broker embebido, en proceso, sin puertos TCP
- ✓ Suscripciones fichero → tabla con mapeo del diccionario
- ✓ Consola web con RBAC, auditoría y DLQ con replay

9

destinos soportados

1

exit en el núcleo

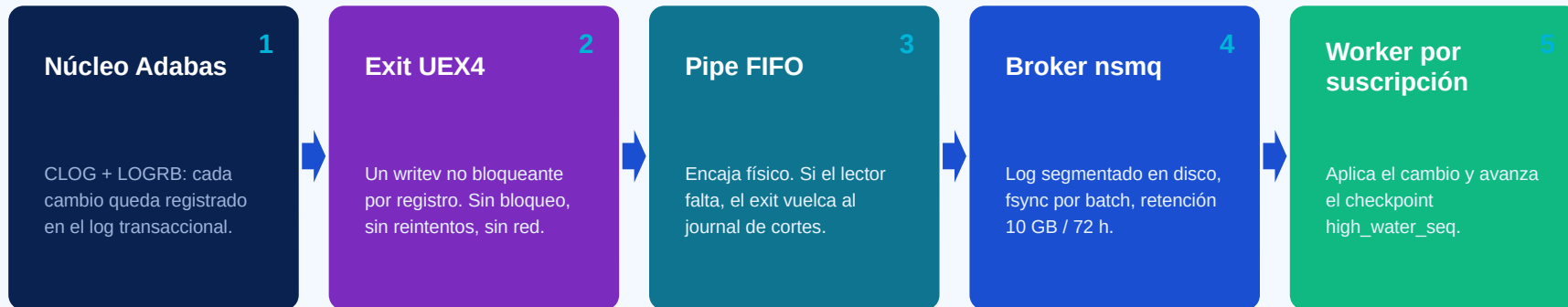
0

cambios perdidos

1

consola web

Cómo viaja un cambio, paso a paso



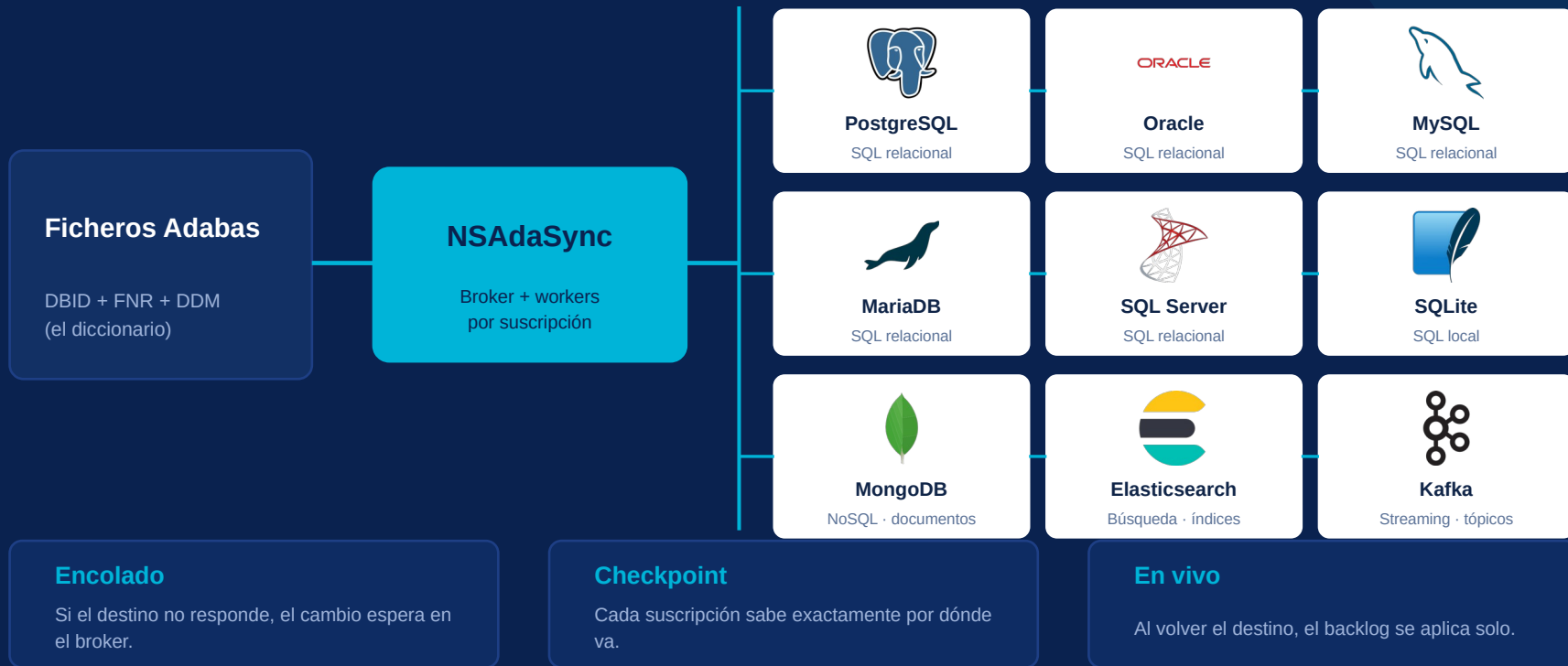
⚠ Sin pipe configurado el exit queda inerte (fail-closed): el núcleo nunca se queda bloqueado esperando a nadie.

El núcleo no sale a la red

Sin clientes que instalar

Todo en un solo servidor

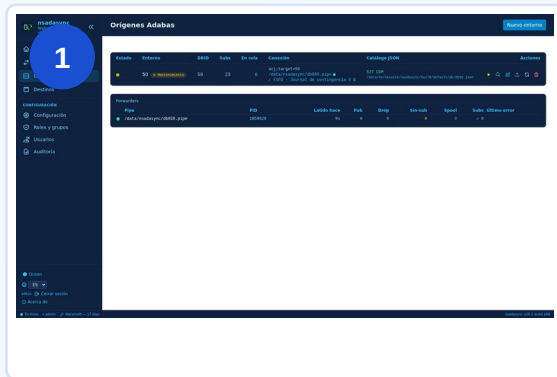
Un origen, nueve destinos, un solo control



→ La misma suscripción se duplica sobre cada destino: un solo origen, muchos consumidores.

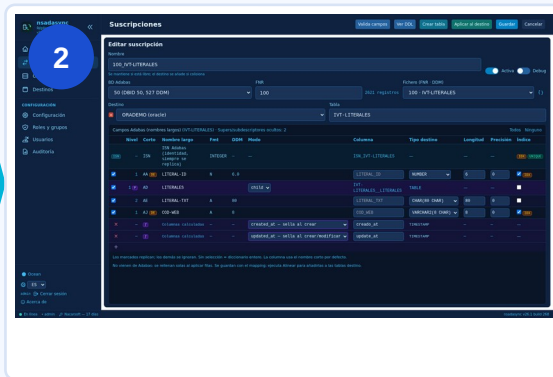
CÓMO FUNCIONA

Del exit en el núcleo al dato en destino, en tres pasos



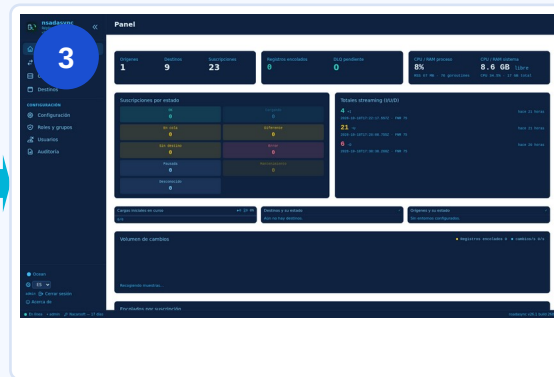
Origen

El fichero Adabas, su diccionario y el pipe por el que salen los cambios.



Suscripción

El mapeo fichero → tabla, con sus campos propios y columnas calculadas.

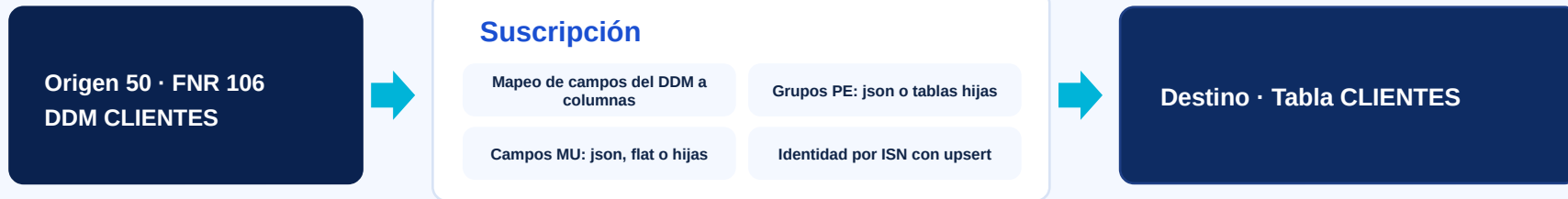


Panel

Estado, encolados, latencia y errores de todo el flujo, en vivo.

Nada de scripts nocturnos ni extractores: un exit en el núcleo, un broker persistente y el resto lo hace la consola.

De fichero Adabas a tabla destino



Campo DDM	Columna destino	Tipo
AA001	nombre	VARCHAR(120)
AA002	apellidos	VARCHAR(120)
MESES (MU)	MESES_1 ... MESES_8	TEXT

- Estado por suscripción: ok, en cola, sin destino, error
- Última carga inicial y su duración
- Contadores de insert, update y delete
- SEQ de control: por dónde va cada flujo

Suscripciones

Replica en pausa por mantenimiento

Nombre	Destino / Tabla	Carga inicial	CRUD	SEQ
100_NF_LITERALES	ORADEMO	2020-10-18 13:00:01	21 insert	0
120_N2-CIS-2	MYDEMO	2020-10-18 17:30:07	71 insert	0
50	mongodemo	2020-10-18 17:37:24	71 insert	0
120_N2-CIS-2-MARIA	MARIADemo	2020-10-12 17:30:09	511 insert	0
50	MSDEMO	2020-10-12 17:30:08	500 insert	0
120_N2-CIS-2-MYSQL	MSDEMO	2020-10-12 18:00:41	500 insert	0

Primero la carga, después el directo

Prepare + Preview DDL

1

Se validan los campos contra el diccionario y se previsualiza el DDL sin ejecutarlo.

Ensure table

2

Crea tabla padre, tablas hijas e índices en el destino.

Carga inicial

3

Modos merge, empty o drop, con capacidad limitada por destino.

Streaming continuo

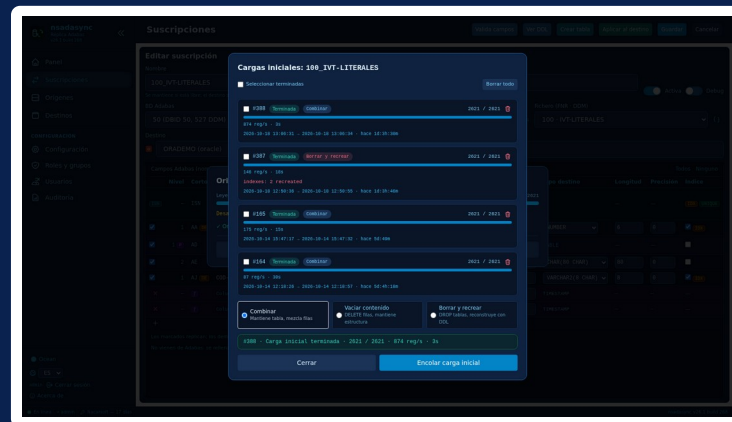
4

Cada cambio posterior se aplica al instante y queda registrado.

Checkpoint `high_water_seq`

Si el proceso se corta, la carga se reanuda desde el checkpoint: no se vuelve a empezar.

✓ Antes de encolar una carga se valida que el destino responde y que el origen está activo: si no, la carga se rechaza en lugar de morir a medias.



Asistente de carga inicial: una tarea con su estado, su duración y su tasa de registros.

La estructura puede cambiar: no hace falta recargar

A · Cambia el Adabas

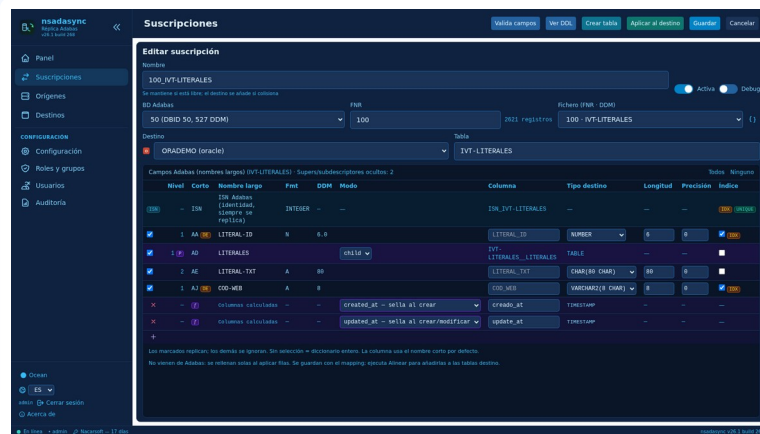
Nuevo campo en el fichero o diccionario actualizado: se recarga el diccionario, se añade el campo al mapeo y «Aplicar al destino» añade la columna a la tabla existente sin tocar datos, contadores ni checkpoint. El streaming sigue.

B · Cambia el destino

Si alguien añade una columna, el upsert la respeta. Si la tabla se borra, la suscripción queda «sin tabla» y el worker se aparca en el broker: al recrearla, reanuda el backlog. Sin recargar.

C · Llega un campo sin mapear

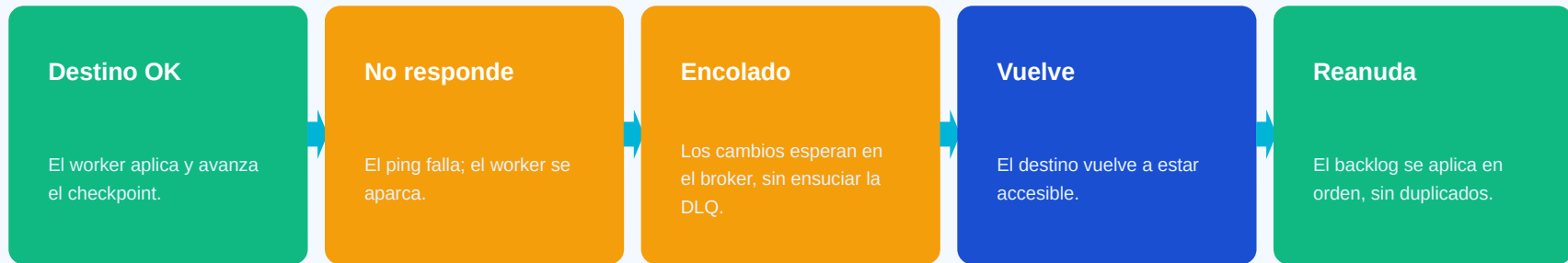
El apply se rechaza con el detalle «unmapped fields... check catalog» y la entrada va a la DLQ. Se corrige el mapeo y se hace replay: tampoco hace falta carga inicial.



El editor de la suscripción: «Valida campos», «Ver DDL», «Crear tabla» y «Aplicar al destino».

La carga inicial es solo el backfill. A partir de ahí, el esquema evoluciona sin volver a cargar.

Si un destino cae, el dato espera. No se pierde



DLQ con replay

Si el apply falla de verdad, la entrada va a la cola de errores con su motivo. Se inspecciona y se reintenta. Ningún evento se pierde.

Lo dice la propia consola

«4 no accesibles – los cambios quedan encolados en el broker: ELADemo, MARIADemo, MSDEMO, kafdemo».

Pensado para producción

1 worker por suscripción

Aislamiento: un destino lento no bloquea a los demás.

Batch 200 con fsync

Persistencia en disco por lote: nada se pierde al reiniciar.

Retención 10 GB / 72 h

El broker acota el disco; se purga por tópicos y por antigüedad.

max_loads por destino

Las cargas se serializan o paralelizan según el destino.

max_records

Límite opcional de registros por carga, para pruebas y migraciones.

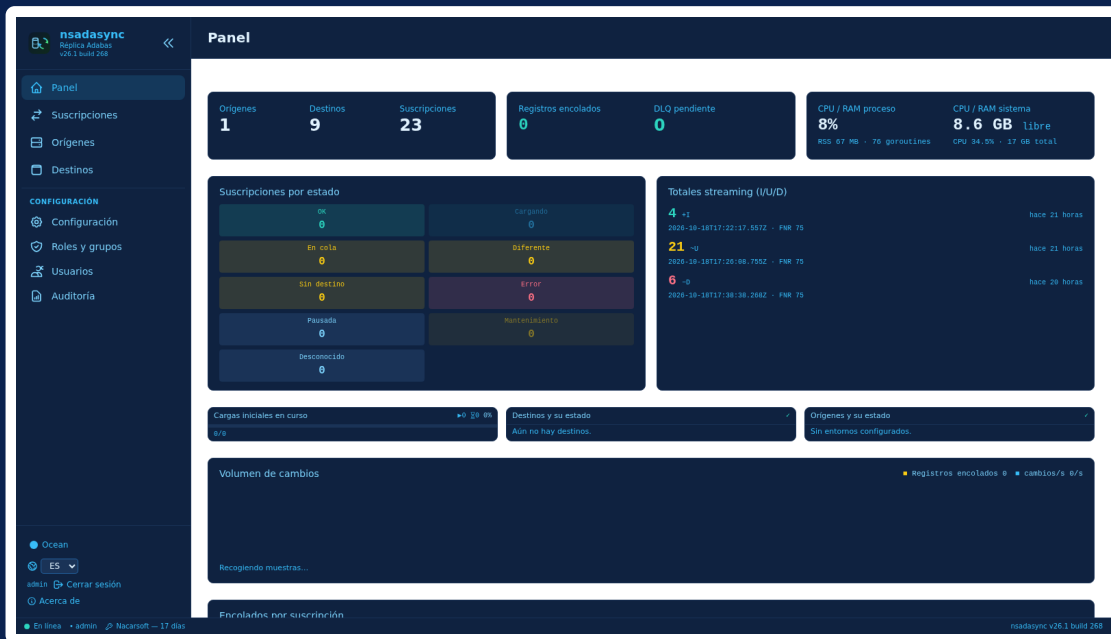
Checkpoints por suscripción

Cada una sabe por dónde va y se reanuda sin repetir.

El exit no bloquea · El broker persiste por lote · Cada suscripción se recupera sola

OPERACIÓN

Todo el estado en una pantalla



- Orígenes y destinos con su estado y su latencia
- Cargas iniciales en curso y encoladas
- Estado de cada suscripción: ok, sin destino, en cola, error
- Totales de insert, update y delete del streaming
- Volumen de cambios y registros encolados

Encolados por suscripción

Volumen de cambios

Salud de la cadena

Latencia por destino

Quién ve qué, y queda registrado

Control de acceso

- RBAC con roles de sistema
- Admin, SyncOperators, Viewers
- Grupos y usuarios
- Permisos por área

Automatización

- Tokens máquina nstm_...
- Scope por suscripción
- Caducidad configurable
- Allowlist de CIDRs

Trazabilidad

- Auditoría de cada acción
- Usuario e IP de cliente
- Log por suscripción
- Retención configurable

config.yaml 0600 · base de datos 0700 · bcrypt · JWT HS256 · cabeceras CSP · proxy inverso con HTTPS

Dónde aporta valor desde el primer día

Analítica y BI

Oracle, PostgreSQL o MySQL alimentan el reporting en directo, sin cargas nocturnas ni ficheros intermedios.

Streaming de eventos

Kafka recibe un tópico por tabla, con la clave ISN y tombstone al borrar: converge con compactación.

Búsqueda y texto

Elasticsearch mantiene un índice por tabla, creado durante la carga, alimentado en vivo.

Aplicaciones modernas

MongoDB y SQL Server reciben documentos y filas que alimentan APIs y aplicaciones sin tocar el núcleo.

→ Un solo origen puede alimentar varios destinos a la vez: la suscripción se duplica sobre el destino que se quiera.

De cero a datos en destino, en tres pasos

1

Activar el exit

Compilar libnsadasync.so y activar ADAUEX_4, USEREXITS=4, LOGGING=(CB) y LOGRB en el núcleo.

2

Dar de alta el origen

Diccionario JSON de nsddm2file y pipe_path de la fila de orígenes.

3

Destinos y suscripciones

Alta del destino con ping y test, ensure-table, carga inicial y streaming.

REQUISITOS

- Linux (RHEL 8 / 9)
- Adabas Open Systems con CLOG y LOGRB
- Diccionario JSON por origen (≤ 8 MiB)
- Proxy inverso con HTTPS en producción
- Consola web en el puerto 11035

```
nsadasync init ./data → nsadasync start ./data → consola web :11035
```

Lo que más se pregunta

¿Hay que parar el núcleo?

No. El exit escribe un frame por registro con un único writev no bloqueante. Si no hay lector, vuelca al journal y sigue.

¿Qué pasa si el destino se cae?

Los cambios se encolan en el broker y se aplican al volver. La DLQ solo registra errores de apply reales.

¿Y los borrados?

Viajan sin after-image: se aplican por ISN, sin decodificar la imagen del registro.

¿Cómo se tratan MU y PE?

Como JSON (por defecto) o como tablas hijas normalizadas, resueltas igual en el DDL, la carga y el apply.

¿Soporta cargas grandes?

Sí: carga inicial por lotes con checkpoints reanudables y capacidad limitada por destino (max_loads).

¿Qué puedo monitorizar?

Origen, destino, estados, encolados, latencia, DLQ y auditoría; el broker expone offsets y checkpoints.

Su dato Adabas, en vivo donde lo necesite

Pruebe NSAdaSync con sus propios ficheros: se lo montamos y lo vemos funcionando con sus datos.

Demo guiada

Pruébalo en su entorno

Hablemos de su caso

nacarsoft.com · soporte@nacarsoft.com

