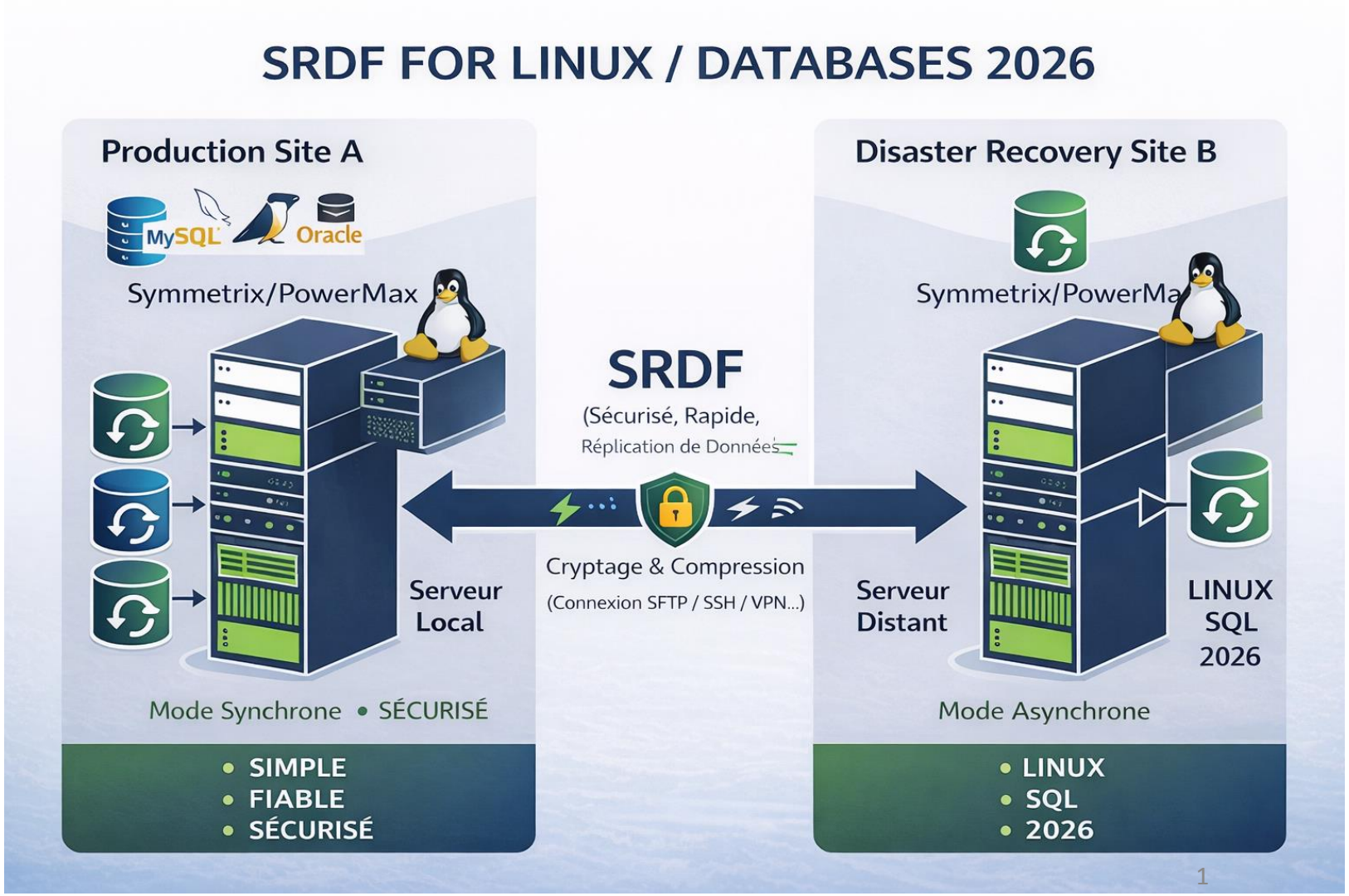
IDEO-LAB

May 2026

v 2.1

SRDF FOR STARTUPS



ANALYSE DE L’EXISTANT - EUROPE - USA

ANALYSE ET ESTIMATION DU POTENTIEL

Région	Nombre de Startups Tech (Est. 2026)	Commentaire
USA	**90 000 – 110 000**	Le plus gros vivier, avec une adoption cloud/DR (Disaster Recovery) native.
Europe (UE)	**45 000 – 55 000**	Écosystèmes forts en France, Allemagne, Pays-Bas et pays Nordiques.
Royaume-Uni	**10 000 – 12 000**	Très forte concentration en Fintech et SaaS B2B.
Suisse	**3 500 – 5 000**	Forte densité en DeepTech, HealthTech et Crypto/Finance.
TOTAL	**~150 000 – 180 000**	**Marché adressable théorique.**

Le "Paper Plan" vs La Réalité Opérationnelle

Si l'on interroge les fondateurs ou CTO de startups matures :

- **75%** affirment avoir une stratégie de Disaster Recovery (DR).
- **Seulement 30% à 35%** disposent d'un plan **réellement opérationnel** (testé au moins une fois par an avec un succès total du basculement).
- **Moins de 10%** des startups ont une solution de **réplication synchrone ou quasi-synchrone** (type SRDF). La majorité se contente de snapshots cloud (AWS EBS/RDS) déportés sur une autre région, ce qui n'est pas de la haute disponibilité "active".

Statistiques de déploiement par typologie

Niveau de Protection	Part des Startups (Core Target)	Mécanisme technique habituel
Basique (Backup)	~50%	Snapshots quotidiens vers S3/Azure Blob. RTO (reprise) > 4h.
Intermédiaire (DRaaS)	~25%	Réplication asynchrone via outils cloud natifs. RTO ~ 1h.
Avancé (SRDF-like)	**~5% à 8%**	**Réplication temps réel / Multi-site actif. RTO < 15 min.**
Néant	~17%	Aucune redondance hors-site formalisée.

Estimation chiffrée (Monde Occidental)

- **USA :** Environ **9 000 à 11 000** startups (principalement des Series B+ ou Fintechs).
- **Europe + UK :** Environ **4 000 à 5 000** startups.
- **Suisse :** Environ **400 à 600** (très forte concentration due aux exigences de la FINMA dans la Crypto/Finance).

  Total "Équipés" : Environ 15 000 entreprises. > Cela signifie que votre **marché de conquête** (ceux qui devraient l'avoir mais ne l'ont pas car c'est trop complexe ou cher) est de plus de **100 000 startups**.

Les "Bloqueurs" qui expliquent ces chiffres

Pourquoi si peu de startups ont un vrai "SRDF" ?

1. **Le Coût :** Le SRDF traditionnel demande du stockage haut de gamme. Dans le Cloud, la réplication multi-région coûte souvent le double de l'infra nominale.

2. **La Complexité :** Gérer la cohérence des données (Data Consistency) lors d'un failover sur une application distribuée est un enfer technique pour une petite équipe.

3. **L'Illusion du Cloud :** Beaucoup de startups pensent que "parce qu'elles sont sur AWS, elles sont protégées", oubliant qu'une panne de zone ou de région (ou une erreur humaine de suppression) nécessite un plan de secours externe.

L'opportunité pour notre projet « SRDF »

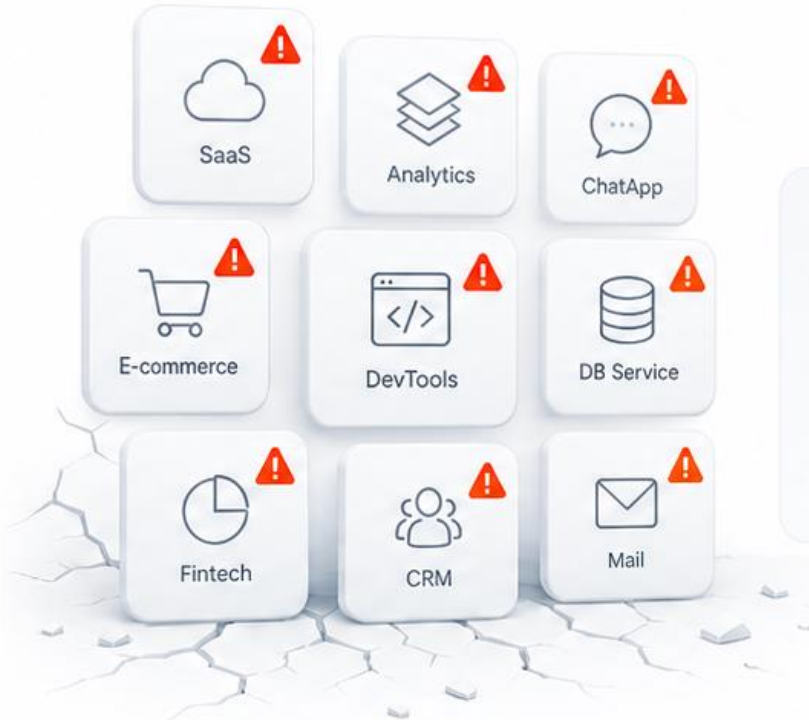
- Si on arrive à offrir la **fiabilité du SRDF** avec la **simplicité d'un plugin SaaS**, vous ciblez les **~65%** de startups qui savent qu'elles sont vulnérables mais qui reculent devant la complexité technique des solutions actuelles.

IDEO·LAB

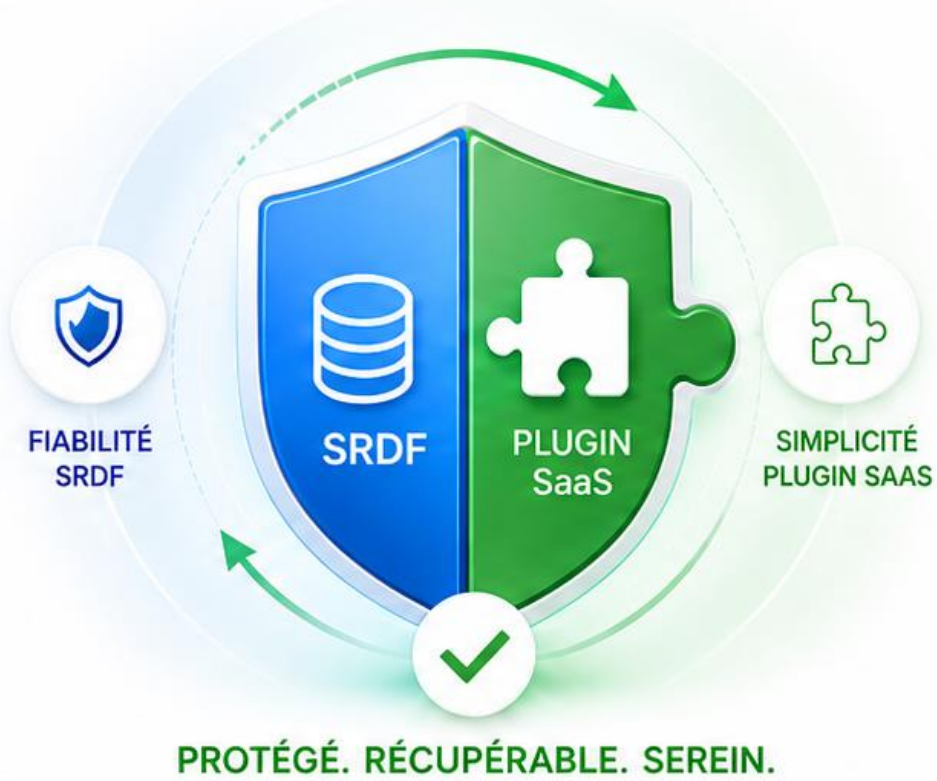
L’opportunité pour notre projet

Fiabilité SRDF + simplicité d’un plugin SaaS

Cibler les startups conscientes de leur vulnérabilité, mais encore mal équipées.



≈ 65%
des startups savent qu’elles sont vulnérables



Transformer une inquiétude technique en *solution* simple, accessible et fiable.

ANALYSE D’UN CAS D’ECOLE

Crash serveur, Cyber Attaques, Incendie

IDEO-LAB

IDEO-LAB

IDEO·LAB

“le site tombe, l’entreprise continue.”

- Le miracle SRDF : quand le site **A tombe**, le site **B reprend**

Une continuité d’activité simple, automatisée et pragmatique pour les bases SQL, Linux, applications et services critiques.

Imaginez une entreprise dont le site principal tombe brutalement : panne serveur, cyberattaque, corruption de données ou incident cloud.
Dans un modèle classique, on cherche la dernière sauvegarde, on restaure, on reconfigure, on teste, on perd du temps, parfois des données.
Avec SRDF, le site B est déjà préparé. Les bases SQL, le socle Linux, le code applicatif et les éléments critiques ont été répliqués au fil de l’eau.
On redémarre sur le site distant, on rebascule le DNS, et l’activité reprend.
C’est cela la promesse SRDF : transformer une catastrophe technique en procédure de reprise maîtrisée.

Production normale → Incident → Site B prêt → Bascule DNS → Site relancé

Un Scénario en 5 actes

Avec SRDF, la catastrophe devient un scénario piloté :détection, réplication, reprise, bascule DNS, retour en ligne.

1. Fonctionnement normal

Le site A est en production.

SRDF réplique au fil de l'eau vers le site B :

SQL, code applicatif, configuration Linux, services, fichiers critiques, dépendances.

2. Incident majeur

Crash serveur, cyberattaque, corruption, panne cloud, erreur humaine.

Le site A devient indisponible.

L'activité est stoppée.

3. Reprise sur le site B

Le site B est déjà synchronisé jusqu'au dernier point de réplication.

RPO cible : environ **5 minutes** selon la configuration et la charge.

4. Bascule DNS

Le domaine est réaffecté vers l'adresse IP du site B.

Temps cible : environ **2 minutes** selon TTL DNS et configuration.

5. Retour en ligne

Le site redémarre sur l'infrastructure distante.

Les données sont récupérées, l'application repart, les utilisateurs reviennent.

1. Réplication continue entre le site A et le site B

IDEO·LAB

En fonctionnement normal, SRDF réplique au fil de l'eau le site de production vers un site distant.

SQL | Code applicatif | Socle Linux | Fichiers critiques | Services

Tant que le site A fonctionne, le site B reste synchronisé et prêt.

2. Incident majeur sur le site A

IDEO·LAB

Crash, cyberattaque, corruption ou panne : le site principal devient indisponible.

CRASH | CYBERATTAQUE | CORRUPTION | PANNE INFRA

Le site A tombe. L'activité est menacée. SRDF garde un site B prêt à reprendre.

3. Reprise d'activité sur le site B

IDEO·LAB

Le site B, déjà synchronisé, redémarre avec les données et services critiques disponibles.

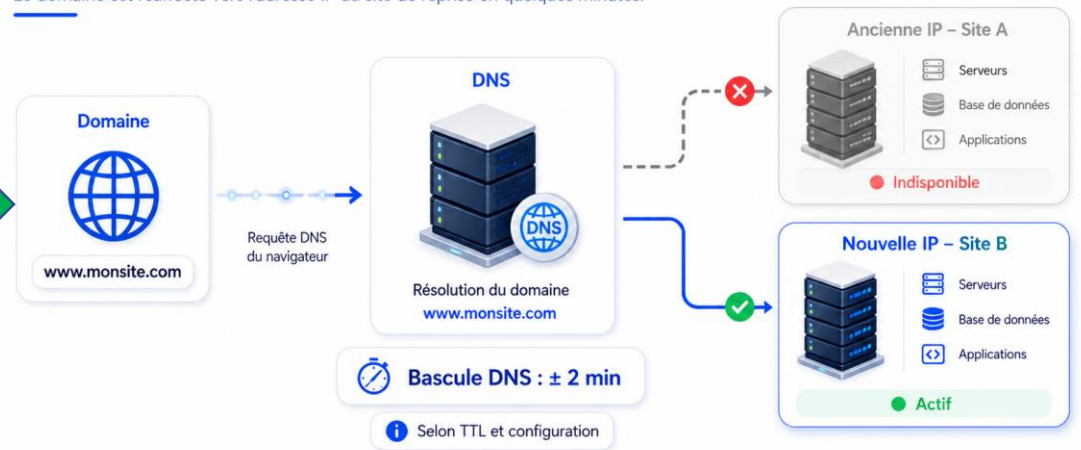
SQL restauré | Application relancée | Linux prêt | Services OK

Le site B reprend rapidement l'activité avec une perte de données minimale.

4. Bascule DNS vers le site B

IDEO·LAB

Le domaine est réaffecté vers l'adresse IP du site de reprise en quelques minutes.



Le trafic utilisateur est redirigé vers le site B pour rétablir l'accès au service.

5. Le site web est de nouveau opérationnel

IDEO·LAB

Les utilisateurs retrouvent le service. Rien n'a été perdu, ou presque.

SITE A – PRODUCTION
INCIDENT MAJEUR

- Site web de production
- Serveur Linux
- Base de données SQL
- Applications & code
- Fichiers
- Services

Indisponible

LES UTILISATEURS RETROUVENT LE SERVICE

Service accessible
Performant & sécurisé

SITE B – PRODUCTION TEMPORAIRE
EN SERVICE

- Site web de production
- Serveur Linux
- Base de données SQL
- Applications & code
- Fichiers
- Services

Opérationnel

SERVICE RÉTABLI
Le site est de nouveau accessible pour tous les utilisateurs.

PERTE MINIMALE
Dernières données répliquées en quelques minutes.

CONTINUITÉ D'ACTIVITÉ
Les opérations métiers continuent sans interruption.

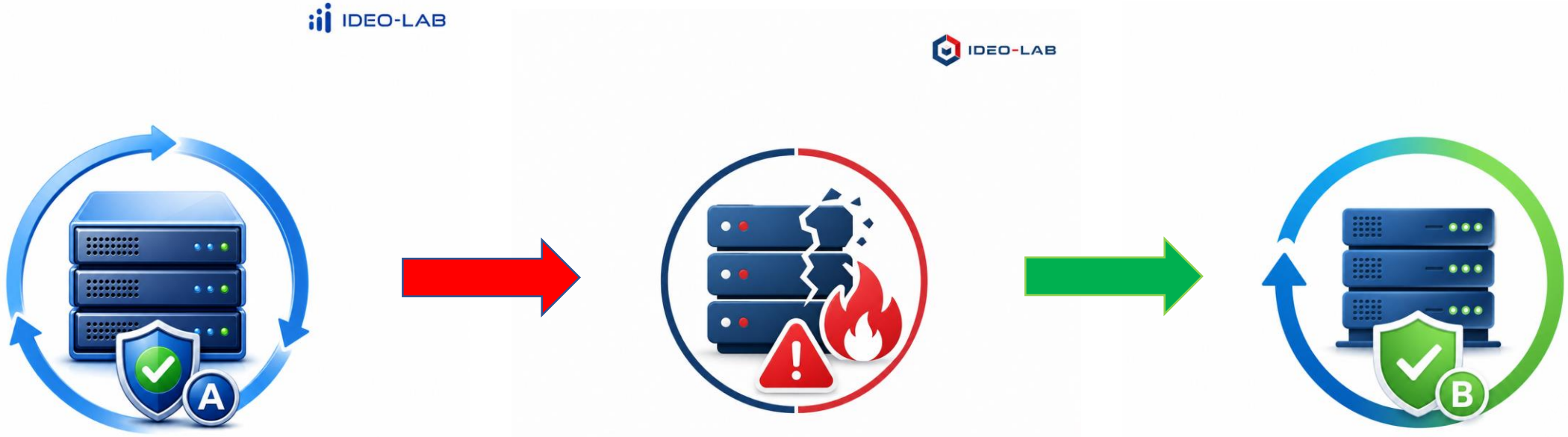
RPO ± 5 min

DNS ± 2 min

Le miracle SRDF : la catastrophe devient une reprise maîtrisée.

Avec SRDF, l'entreprise continue à fonctionner malgré l'incident majeur.

“Le miracle SRDF : presque rien perdu, activité restaurée.”



“Le miracle SRDF : presque rien perdu, activité restaurée.”

SRDF PAR IDEO-LAB EN 2026

SRDF by IDEO-LAB

IDEO·LAB

Réplication, synchronisation et reprise d'activité pour infrastructures Linux et bases SQL

SITE SOURCE

- Serveurs Linux
- Bases de données SQL
- Applications & Services

RÉPLICATION SÉCURISÉE & AUTOMATISÉE

Contrôle | Automatisation | Sécurité

SITE DISTANT

- Serveurs Linux
- Bases de données SQL
- Applications & Services

ENVIRONNEMENTS SUPPORTÉS

AWS | GCP | Azure | OVH | Bare Metal | VM Linux | SQL

Une alternative logicielle ouverte, plus accessible et plus flexible que les solutions propriétaires traditionnelles.

1 Coût réduit

Coût fortement réduit par rapport aux solutions propriétaires.

2 Déploiement simple

Déploiement rapide, sans dépendance à une baie de stockage spécialisée.

3 Portabilité multi-infrastructure

Fonctionne sur AWS, GCP, Azure, OVH, bare metal et VM Linux.

4 Reprise d'activité pragmatique

Conçu pour les équipes techniques, les startups et les PME.

SÉCURISÉE · AUTOMATISÉE · CONTRÔLABLE · ACCESSIBLE

SRDF PAR IDEO-LAB

"10 fois plus accessible, 10 fois plus flexible, 10 à 20 fois moins cher."

- coût
- simplicité
- vitesse de déploiemen
- flexibilité infra
- indépendance vis-à-vis du hardware propriétaire

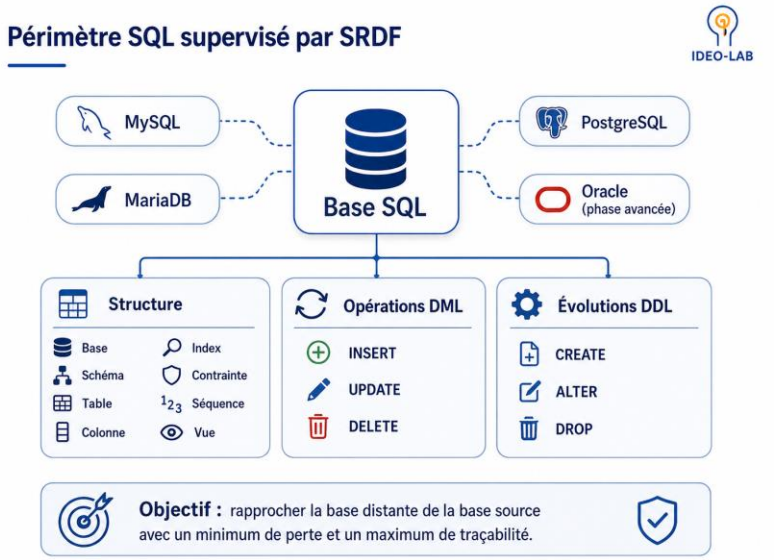
IDEO·LAB

SRDF by IDEO-LAB

Périmètre supervisé et sécurisé par SRDF

- 1. Les bases de données SQL
- 2. Le socle système Linux
- 3. Le code applicatif et ses dépendances
- 4. Les fichiers critiques et volumes applicatifs
- 5. Les services externes et points de dépendance
- 6. Vision cible
- 7. Sécurisation et chiffrement des données SRDF

1. Les bases de données SQL



Le premier contenant naturel de SRDF est la base SQL. C'est le cœur transactionnel de nombreuses applications métier.

SRDF doit pouvoir superviser et sécuriser progressivement :

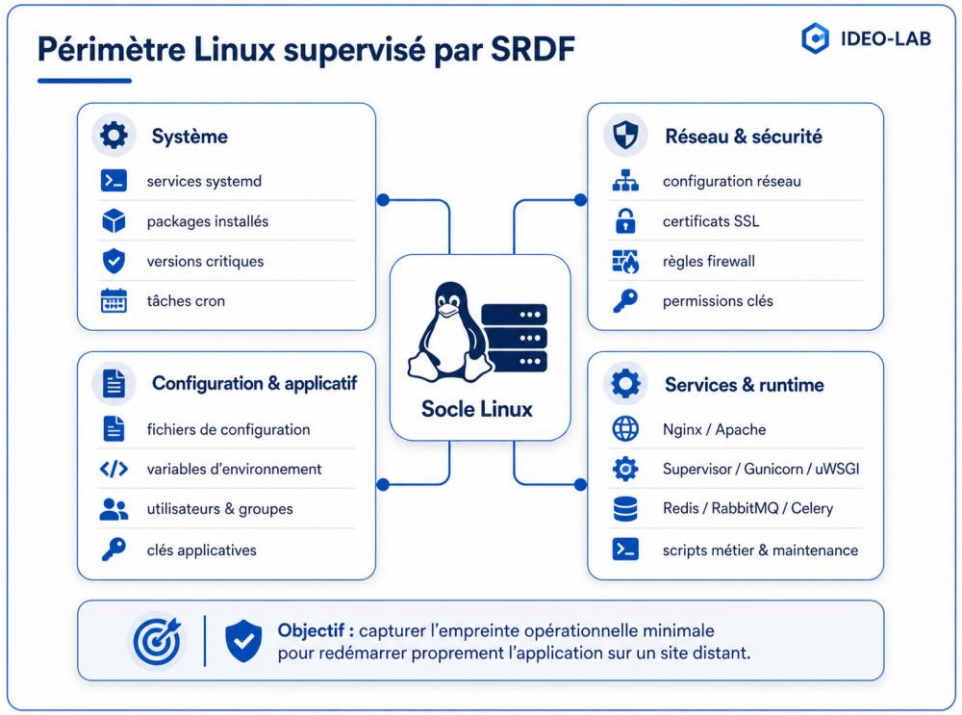
- **MySQL**
- **MariaDB**
- **PostgreSQL**
- **Oracle**, dans une phase plus avancée

Le périmètre ne se limite pas aux données. Il doit couvrir :

- les bases ;
- les schémas ;
- les tables ;
- les colonnes ;
- les index ;
- les contraintes ;
- les séquences ;
- les vues ;
- les procédures et fonctions stockées, selon moteur ;
- les opérations DML : `INSERT`, `UPDATE`, `DELETE` ;
- les opérations DDL : `CREATE`, `ALTER`, `DROP`.

L'objectif est de permettre à une base distante d'être suffisamment proche de la base source pour reprendre l'activité avec un minimum de perte, un minimum de délai et un maximum de traçabilité.

2. Le socle système Linux



Le deuxième contenant est le serveur Linux lui-même. C'est une partie stratégique du projet, car une application ne redémarre pas uniquement avec ses données.

SRDF doit pouvoir inventorier, superviser et reconstruire les éléments indispensables du système :

- services `systemd` ;
- packages installés ;
- versions logicielles critiques ;
- configuration réseau ;
- utilisateurs et groupes techniques ;
- permissions importantes ;
- tâches cron ;
- fichiers de configuration ;
- variables d'environnement ;
- certificats SSL ;
- clés applicatives ;
- règles firewall ;
- configuration Nginx / Apache ;
- configuration Supervisor / Gunicorn / uWSGI ;
- services Redis, RabbitMQ, Celery, Elasticsearch, Memcached, etc. ;
- scripts de démarrage, scripts de maintenance et scripts métier.

L'objectif n'est pas de cloner aveuglément tout le serveur. L'objectif est de **capturer l'empreinte opérationnelle minimale permettant de redémarrer proprement l'application sur un site distant.**

IDEO·LAB

3. Le code applicatif et ses dépendances

Un autre contenant important est la couche applicative.

Pour une application web moderne, il faut pouvoir sécuriser :

- le dépôt Git ou l'artefact déployé ;
- la version exacte du code ;
- les fichiers `.env` ou paramètres équivalents ;
- les dépendances Python, Node.js, Java ou autres ;
- les fichiers statiques nécessaires ;
- les scripts de migration ;
- les modules compilés éventuels ;
- les workers et services applicatifs ;
- les paramètres de lancement.

Dans le cas d'un projet Django/Python, cela inclut notamment :

- le projet Django ;
- les applications installées ;
- les migrations ;
- les settings critiques ;
- les workers Celery ;
- les commandes de management ;
- les fichiers statiques et médias si nécessaire.

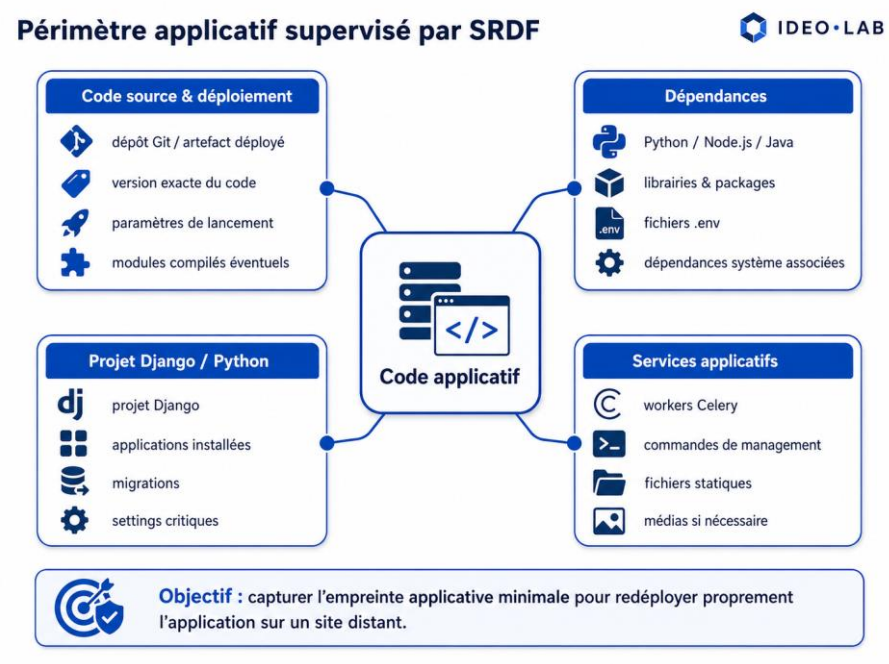
4. Les fichiers critiques et volumes applicatifs

SRDF doit aussi pouvoir identifier les fichiers qui ne sont pas en base SQL mais qui sont indispensables à l'activité.

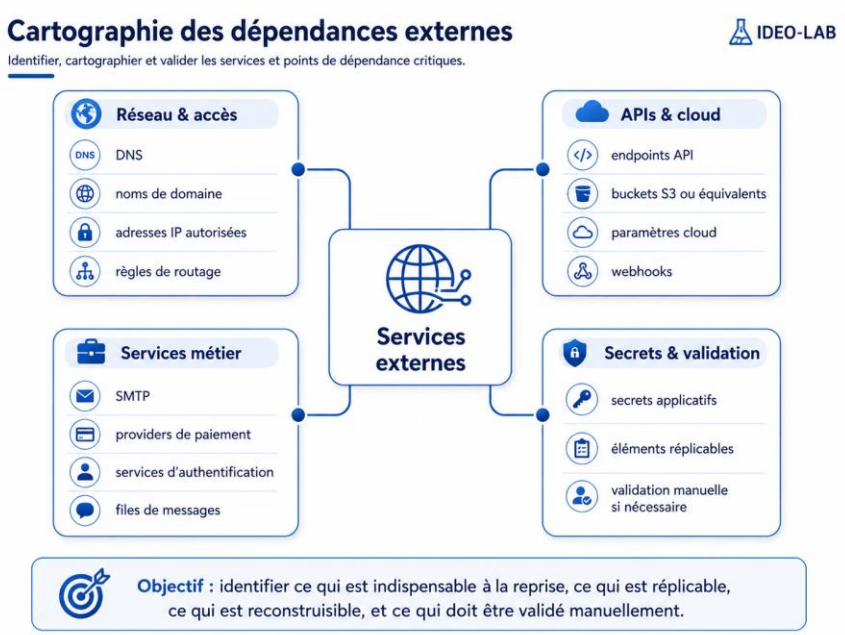
Exemples :

- médias utilisateurs ;
- documents générés ;
- exports ;
- imports ;
- répertoires applicatifs ;
- logs critiques ;
- fichiers de licences ;
- fichiers de configuration métier ;
- clés privées ou certificats, avec règles de sécurité adaptées ;
- volumes Docker ou volumes applicatifs.

Cette couche peut être traitée différemment de la réplication SQL : snapshot, synchronisation incrémentale, checksum, archivage, ou simple reconstruction depuis un stockage objet selon les cas.



5. Les services externes et points de dépendance



Enfin, SRDF doit garder une cartographie des dépendances externes, car un serveur redémarré sur un site distant doit savoir à quoi il doit se reconnecter.

Cela peut inclure :

- DNS ;
- noms de domaine ;
- endpoints API ;
- buckets S3 ou équivalents ;
- SMTP ;
- providers de paiement ;
- services d'authentification ;
- webhooks ;
- files de messages ;
- secrets applicatifs ;
- paramètres cloud ;
- adresses IP autorisées ;
- règles de routage.

Cette cartographie ne signifie pas que SRDF réplique tout. Elle signifie que SRDF doit savoir **ce qui est indispensable à la reprise**, ce qui est réplicable, ce qui est reconstruisible, et ce qui doit être validé manuellement.

Vision cible

Vision cible

SRDF doit donc être vu comme une plateforme de **supervision, sécurisation et reconstruction d'environnement critique**, organisée autour de plusieurs contenants :

1. **Données SQL**
2. **Système Linux**
3. **Code applicatif**
4. **Fichiers et volumes critiques**
5. **Services et dépendances externes**

Cette vision est beaucoup plus forte qu'une simple réplication de base de données. Elle positionne SRDF comme un outil de continuité d'activité pragmatique : capable de préparer un site distant, d'y maintenir les données critiques, d'y reconstruire l'environnement système, et de rendre possible un redémarrage réel de l'application en cas d'incident majeur.

Sécurisation et chiffrement des données SRDF

- TSA l'encryptage du Futur

La sécurité est l'un des piliers majeurs de SRDF. Le projet ne se limite pas à transporter des données d'un serveur source vers un serveur distant : il doit garantir que chaque flux répliqué reste **confidentiel, authentifié, traçable et résistant aux attaques modernes**.

Dans cette logique, SRDF prévoit l'intégration d'un nouveau système de chiffrement **TSA nouvelle génération**, destiné à remplacer l'ancien modèle TSA. Ce nouveau composant est conçu comme une brique cryptographique avancée, pensée pour les besoins de 2026 et au-delà : réplication distante, transport inter-cloud, sauvegarde chiffrée, reprise d'activité, protection contre l'espionnage industriel et anticipation des menaces post-quantiques.

L'objectif est clair : **ne jamais transporter une donnée SRDF exploitable en clair**.

IDEO·LAB

IDEO · LAB

TSA 2026

Cryptographie avancée

Sécurisation et chiffrement des données SRDF

- 1. Chiffrement natif des flux SRDF
- 2. Le nouveau TSA : une rupture cryptographique
- 3. Orientation post-quantique
- 4. Une conception issue d’une expertise scientifique de haut niveau
- 5. Sécurité opérationnelle : au-delà du chiffrement
- 6. Valeur stratégique pour SRDF
- Synthèse

IDEO·LAB

Sécurisation et chiffrement des données SRDF

IDEO·LAB

Chaque flux répliqué reste confidentiel, authentifié, traçable et protégé.

Aucune donnée SRDF n'est transportée en clair.

Réplication distante	Transport inter-cloud	Sauvegarde chiffrée	Reprise d'activité	Protection post-quantique

Algorithmes résistants au quantique (PQ) – Protection durable face aux menaces futures.

1. Chiffrement natif des flux SRDF

Dans SRDF, les données ne sont pas simplement envoyées au serveur distant. Elles suivent une chaîne de sécurisation complète :

1. capture des changements SQL ou système ;
2. constitution d'un batch cohérent ;
3. déduplication et optimisation ;
4. compression ;
5. chiffrement TSA nouvelle génération ;
6. transport sécurisé ;
7. réception côté cible ;
8. contrôle d'intégrité ;
9. déchiffrement contrôlé ;
10. application sur le système distant.

Le chiffrement intervient donc au cœur du pipeline, avant tout transport réseau. Même si un flux était intercepté, le contenu resterait inutilisable sans les mécanismes de déchiffrement associés.

SRDF doit protéger plusieurs types de contenus :

- événements SQL ;
- batches de réplication ;
- fichiers de configuration ;
- secrets applicatifs ;
- métadonnées système sensibles ;
- snapshots logiques ;
- journaux critiques ;
- accusés de réception signés ;
- ordres de reprise ou de resynchronisation.

IDEO·LAB

1. Chiffrement natif des flux SRDF

IDEO·LAB

Dans SRDF, les données ne sont pas simplement envoyées au serveur distant.
Elles suivent une chaîne de sécurisation complète avant tout transport réseau.

1. capture des changements SQL ou système
2. constitution d'un batch cohérent
3. déduplication et optimisation
4. compression
5. chiffrement TSA nouvelle génération
6. transport sécurisé
7. réception côté cible
8. contrôle d'intégrité
9. déchiffrement contrôlé
10. application sur le système distant

Contenus protégés par SRDF

- événements SQL
- batches de réplication
- fichiers de configuration
- secrets applicatifs
- métadonnées système sensibles
- snapshots logiques
- journaux critiques
- accusés de réception signés
- ordres de reprise ou de resynchronisation

Aucune donnée SRDF n'est transportée en clair.

Même en cas d'interception, le contenu reste inutilisable sans les mécanismes de déchiffrement associés.

IDEO·LAB

2. Le nouveau TSA : une rupture cryptographique

Le nouveau système TSA est présenté comme une rupture par rapport aux approches classiques. Il ne s'agit pas seulement d'ajouter une couche de chiffrement standard autour d'un flux réseau, mais de construire une brique cryptographique intégrée à la logique SRDF.

Ce nouveau TSA est conçu autour de plusieurs objectifs :

- résistance aux attaques classiques ;
- anticipation des menaces post-quantiques ;
- protection des clés ;
- contrôle d'intégrité ;
- protection contre la falsification des batches ;
- détection des rejeux ;
- séparation stricte entre transport, stockage et exécution ;
- capacité à sécuriser aussi bien les données SQL que les éléments système.

Le positionnement est volontairement ambitieux : **SRDF doit être conçu comme une solution de réplication sécurisée par défaut**, et non comme une solution de réplication à laquelle on aurait ajouté la sécurité après coup.

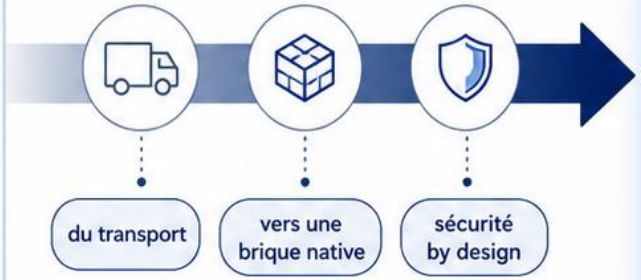
2. Du système RSA historique au nouveau TSA

Une évolution vers une cryptographie native, intégrée à la logique SRDF et pensée pour les menaces de demain.

Ancien système RSA

- Chiffrement classique
- Protection centrée sur le flux
- Gestion des clés conventionnelle
- Sécurité ajoutée autour du transport
- Limites face aux menaces post-quantiques

Évolution cryptographique



Nouveau système TSA

- Brique cryptographique intégrée à SRDF
- Protection des clés
- Contrôle d'intégrité
- Protection contre la falsification des batches
- Détection des rejeux
- Séparation transport / stockage / exécution
- Sécurise données SQL et éléments système
- Anticipation des menaces post-quantiques

RSA : sécurité ajoutée	**TSA : sécurité native**
RSA : approche classique	**TSA : approche nouvelle génération**
RSA : flux réseau	**TSA : flux + batch + stockage + exécution**
RSA : vision court terme	**TSA : résilience long terme**

Avec TSA, SRDF ne chiffre pas seulement le transport : il sécurise l'ensemble du cycle de réplication.

3. Orientation post-quantique

Les systèmes cryptographiques traditionnels sont aujourd'hui remis en question par l'arrivée progressive de capacités de calcul quantique. Même si la menace opérationnelle dépend encore de nombreux facteurs, les données critiques répliquées aujourd'hui peuvent être interceptées, stockées, puis attaquées plus tard.

SRDF doit donc intégrer une logique de sécurité dite **post-quantique** : protéger les données non seulement contre les attaques actuelles, mais aussi contre les capacités de décryptage futures.

Cette approche est particulièrement importante pour :

- les données financières ;
- les données industrielles ;
- les données clients ;
- les données médicales ;
- les données de propriété intellectuelle ;
- les secrets techniques ;
- les environnements SaaS critiques ;
- les réplications inter-cloud ou inter-pays.

Le nouveau TSA doit donc être présenté comme une brique de chiffrement de nouvelle génération, conçue pour résister à des attaquants de très haut niveau, y compris des acteurs disposant de moyens importants.

Il est préférable de formuler la promesse ainsi :

SRDF vise un niveau de sécurité compatible avec les exigences post-quantiques les plus avancées, sous réserve d'audits cryptographiques indépendants et de validations formelles.

Cette formulation reste crédible pour des investisseurs tout en conservant une ambition forte.

4. Une conception issue d’une expertise scientifique de haut niveau

Le projet TSA bénéficie d’un positionnement rare : il repose sur une approche portée par un ancien mathématicien du CERN et un ingénieur formé à CalTech. Cette double culture — mathématiques avancées et ingénierie système — donne au projet une profondeur particulière.

La logique n’est pas de produire un simple module de chiffrement applicatif, mais de concevoir une architecture de sécurité cohérente, intégrée au cycle complet de réplication :

- génération des clés ;
- rotation des clés ;
- segmentation des batches ;
- chiffrement avant transport ;
- contrôle d’intégrité ;
- signature éventuelle des paquets ;
- vérification côté cible ;
- journalisation de sécurité ;
- impossibilité de rejouer un ancien batch ;
- cloisonnement des droits entre source, cible et orchestrateur.

Cette approche permet de positionner TSA comme un élément différenciant majeur de SRDF.

IDEO·LAB

4. Une conception issue d'une expertise scientifique de haut niveau

IDEO·LAB

Le projet TSA repose sur une convergence rare de compétences : mathématiques avancées, ingénierie système et infrastructure critique.

CERN — Genève
Mathématiques avancées

Approche théorique, rigueur scientifique, modélisation cryptographique

Caltech
Ingénierie de pointe

Architecture, systèmes, fiabilité, conception orientée sécurité

EMC Computer / Dell
Infrastructure critique

Expérience terrain en réplication, stockage, continuité d'activité et architectures enterprise

TSA

Brique cryptographique intégrée à SRDF

ARCHITECTURE DE SÉCURITÉ COHÉRENTE

génération des clés	rotation des clés	segmentation des batches	chiffrement avant transport	contrôle d'intégrité
signature des paquets	vérification côté cible	journalisation de sécurité	protection anti-rejeu	cloisonnement des droits

TSA n'est pas un simple module de chiffrement :
c'est une architecture de sécurité complète, conçue pour l'ensemble du cycle de réplication SRDF.

5. Sécurité opérationnelle : au-delà du chiffrement

Le chiffrement seul ne suffit pas. SRDF doit aussi sécuriser l'exploitation quotidienne.

Le système doit donc intégrer :

- authentification forte entre nœuds SRDF ;
- tokens ou certificats par serveur ;
- rotation programmée des secrets ;
- séparation des environnements ;
- contrôle des permissions ;
- logs d'accès ;
- détection d'anomalies ;
- blocage des batches suspects ;
- contrôle de checksum ;
- accusés de réception signés ;
- historique complet des opérations critiques.

Chaque événement transporté doit pouvoir être vérifié :

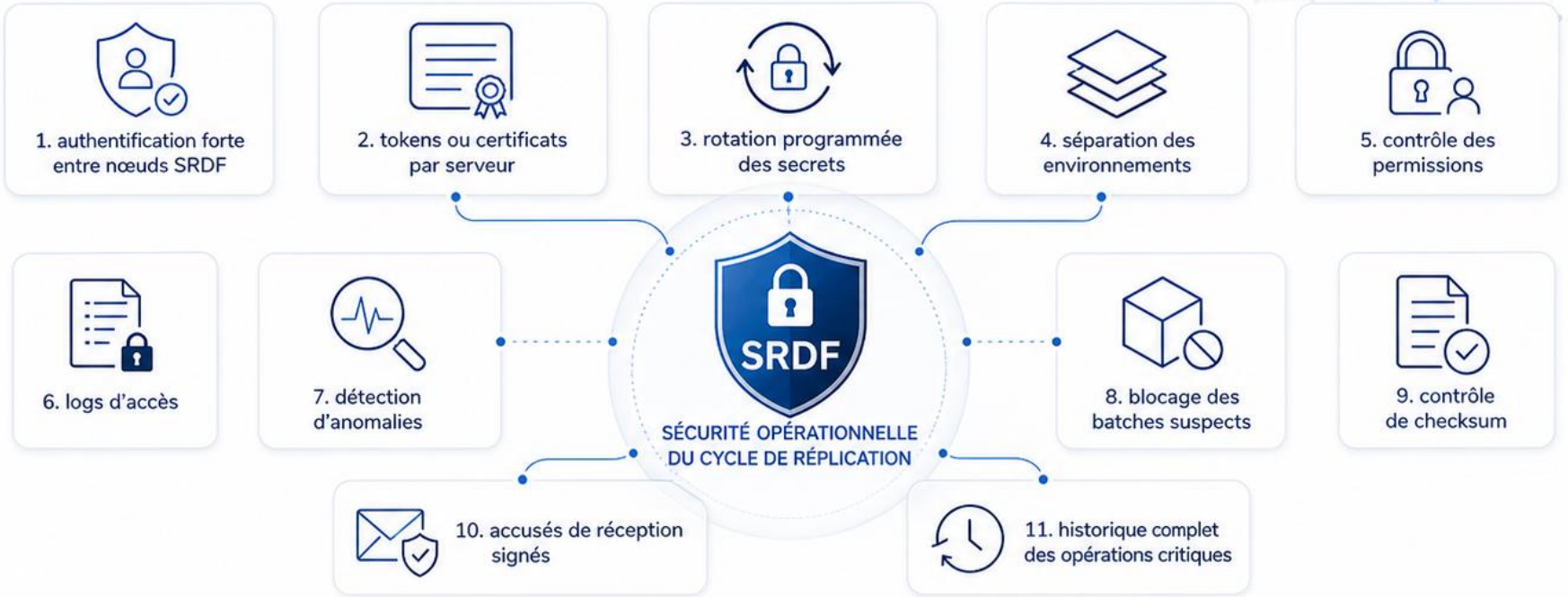
- qui l'a produit ;
- à quelle date ;
- depuis quel serveur ;
- vers quelle cible ;
- dans quel batch ;
- avec quel statut ;
- appliqué ou non ;
- confirmé ou non ;
- rejoué ou non.

IDEO·LAB

5. Sécurité opérationnelle : au-delà du chiffrement

IDEO·LAB

Le chiffrement seul ne suffit pas. SRDF doit aussi sécuriser l'exploitation quotidienne et garantir une traçabilité complète de chaque événement transporté.



Chaque événement transporté doit pouvoir être vérifié

qui l'a produit · à quelle date · depuis quel serveur · vers quelle cible · dans quel batch · avec quel statut · appliqué ou non · confirmé ou non · rejoué ou non

Avec SRDF, chaque flux n'est pas seulement chiffré :
il est authentifié, tracé, vérifiable et gouverné de bout en bout.

6. Valeur stratégique pour SRDF

L'intégration du nouveau TSA donne à SRDF une dimension beaucoup plus forte qu'une simple réplication SQL.

SRDF devient une plateforme capable de transporter des données critiques dans un environnement hostile, en anticipant les exigences futures de cybersécurité.

Pour les investisseurs, cette brique apporte plusieurs avantages :

- différenciation technologique forte ;
- barrière à l'entrée ;
- crédibilité sur les marchés sensibles ;
- extension possible vers d'autres usages que la réplication ;
- valorisation du projet autour d'un actif cryptographique propriétaire ;
- potentiel de licence indépendante du module TSA ;
- compatibilité avec les besoins futurs de cybersécurité post-quantique.

TSA peut donc devenir à terme une brique produit autonome : utilisée dans SRDF, mais aussi intégrable dans d'autres solutions de transport sécurisé, sauvegarde chiffrée, synchronisation inter-sites ou archivage critique.

Synthèse

La sécurisation SRDF repose sur une idée simple : **aucune donnée critique ne doit circuler sans protection cryptographique forte.**

Avec le nouveau système TSA, SRDF ambitionne d'intégrer une couche de chiffrement avancée, orientée post-quantique, conçue pour résister à des attaques de très haut niveau et protéger durablement les données transportées.

La promesse doit être ambitieuse mais crédible :

SRDF ne se contente pas de répliquer les données ; il les protège, les authentifie, les trace et les rend exploitables uniquement par les nœuds autorisés.

Cette capacité de chiffrement avancé, combinée à la réplication SQL, à la supervision système et au redémarrage distant, positionne SRDF comme une solution de continuité d'activité sécurisée, moderne et différenciante.

Synthèse des fonctionnalités maîtresses du projet

- 1. Vision générale du projet
- 2. La fonctionnalité maîtresse : la réplication SQL intelligente
- 3. Le pipeline SRDF : capture, batch, transport, apply
- 4. Orchestration, daemon et automatisation
- 5. Bootstrap initial et reprise d’activité
- 6. Sécurité, fiabilité et contrôle des erreurs
- 7. Supervision, logs et exploitabilité
- 8. Positionnement marché et avantage compétitif
- 9. Évolutions prévues
- Conclusion

1. Vision générale du projet

SRDF by IDEO-LAB est un projet de réplication, de synchronisation et de reprise d'activité destiné aux bases de données et aux infrastructures Linux modernes. L'objectif est de proposer une alternative beaucoup plus accessible, plus légère et plus flexible que les solutions traditionnelles de très haut de gamme, historiquement réservées aux grands groupes et aux infrastructures propriétaires.

Le projet s'inspire du principe des solutions de type SRDF, mais le transpose dans un univers plus ouvert : serveurs Linux standards, bases SQL classiques, clouds publics, VPS, bare metal, environnements hybrides, infrastructures de startups, PME et éditeurs SaaS.

La promesse centrale est simple :

permettre à une entreprise de répliquer ses données critiques vers un site distant, de manière sécurisée, automatisée, contrôlable et économiquement accessible.

SRDF by IDEO-LAB vise donc un marché très large : toutes les structures qui ne peuvent pas investir dans des baies propriétaires très coûteuses, mais qui ont pourtant besoin d'une vraie stratégie de continuité d'activité.

Le positionnement du projet repose sur quatre piliers :

1. **Coût fortement réduit** par rapport aux solutions propriétaires.
2. **Simplicité de déploiement**, sans dépendance à une baie de stockage spécialisée.
3. **Portabilité multi-infrastructure** : AWS, GCP, Azure, OVH, bare metal, VM Linux.
4. **Reprise d'activité pragmatique**, pensée pour les petites équipes techniques, les startups et les PME.

L'ambition n'est pas de copier les solutions historiques de datacenter, mais de créer une solution moderne, logicielle, plus modulaire, mieux adaptée aux besoins de 2026.

IDEO·LAB

2. La fonctionnalité maîtresse : la réplication SQL intelligente

Le cœur fonctionnel de SRDF est la réplication des changements SQL entre un serveur source et un serveur cible.

La première version vise principalement les moteurs **MariaDB / MySQL**, avec une ouverture future vers **PostgreSQL**. Le système est conçu pour capturer les opérations effectuées sur la base source, les transformer en événements internes, les regrouper, les optimiser, puis les appliquer sur une base distante.

Les objets concernés sont à la fois les données et la structure :

Données :

- INSERT
- UPDATE
- DELETE

Structure :

- DATABASE
- TABLE
- COLUMN
- INDEX
- CREATE
- ALTER
- DROP

IDEO·LAB

IDEO·LAB

2. La fonctionnalité maîtresse : la réplication SQL intelligente

Le cœur fonctionnel de SRDF est la réplication des changements SQL entre un serveur source et un serveur cible.

MOTEURS SQL

MariaDB / MySQL — PRINCIPAL

PostgreSQL — OUVERTURE FUTURE

DONNÉES

- INSERT
- UPDATE
- DELETE

STRUCTURE

- DATABASE
- TABLE
- COLUMN
- INDEX
- CREATE
- ALTER
- DROP

SRDF couvre à la fois les modifications de données et une part importante des évolutions de schéma.

Une approche essentielle pour les applications web modernes, où les bases évoluent fréquemment avec les migrations applicatives.

IDEO·LAB

2b – Suivi et évolution des schémas SQL

Cela permet à SRDF de couvrir non seulement les modifications de lignes, mais aussi une partie importante des évolutions de schéma. Cette approche est essentielle pour les applications web modernes, où les bases évoluent fréquemment avec les migrations applicatives.

Le système repose sur une logique de **Log-Based CDC** — Change Data Capture — c'est-à-dire l'analyse des journaux de transaction, notamment les binlogs MariaDB/MySQL. Plutôt que de scanner les tables en permanence, SRDF lit les changements produits par le moteur SQL. Cette méthode est beaucoup plus efficace, plus fiable et plus adaptée à une réplication continue.

Le flux général est le suivant :

1. Une modification est réalisée sur la base source.
2. Le binlog est lu de manière asynchrone.
3. L'opération SQL est parsée et convertie au format interne SRDF.
4. L'événement est stocké dans une table de capture.
5. Les événements sont regroupés dans un batch.
6. Le batch est compressé, chiffré et envoyé au serveur distant.
7. Le serveur cible reçoit le batch.
8. Les opérations sont appliquées sur la base cible.
9. Un accusé de réception est renvoyé à la source.
10. La source marque les événements comme synchronisés.

Cette logique transforme SRDF en moteur de réplication applicative indépendant du matériel.

IDEO·LAB

3. Le pipeline SRDF : capture, batch, transport, apply

3. Le pipeline SRDF : capture, batch, transport, apply

SRDF est structuré autour d'un pipeline clair, découpé en phases indépendantes. Cette séparation est importante pour la fiabilité, la supervision et l'évolution future du produit.

Le pipeline complet est le suivant :

1. Capture côté source

Le service de capture lit les changements SQL depuis le binlog MariaDB/MySQL. Les événements sont convertis dans un format interne standardisé, puis stockés dans une table de type `OutboundChangeEvent`.

2. Construction du batch

Les changements capturés ne sont pas envoyés un par un. SRDF construit des groupes cohérents d'événements, appelés batches ou consistency groups. Cela permet de limiter le coût réseau, d'améliorer les performances et de garantir une meilleure cohérence logique.

3. Déduplication

Lorsque plusieurs modifications concernent le même objet ou la même ligne, SRDF peut éliminer les opérations redondantes et conserver uniquement l'état final utile. Par exemple, plusieurs updates successifs sur une même ligne peuvent être réduits à une seule opération pertinente. Cette déduplication est l'un des leviers majeurs d'optimisation.

4. Compression et chiffrement

Avant transport, les batches peuvent être compressés puis chiffrés. Cela réduit le volume réseau et sécurise les données en transit. La solution est donc conçue pour des environnements distants, cloud, VPS ou datacenters.

IDEO·LAB

3b – Transport et réception sécurisés

5. Transport vers la cible

Le serveur source vérifie que le serveur cible est joignable et disponible. Le batch est ensuite envoyé via une API ou un canal sécurisé vers le serveur distant.

6. Réception côté cible

Le serveur cible reçoit les batches, les inscrit dans une file d'attente locale ou une table d'inbox, vérifie leur intégrité et prépare leur application.

7. Application SQL

Les opérations sont appliquées sur la base distante. Le système doit éviter les doubles exécutions grâce à des mécanismes d'idempotence, de contrôle d'état et d'identifiants uniques.

8. Accusé de réception

Une fois le batch appliqué, le serveur cible renvoie un accusé de réception au serveur source. La source peut alors marquer les événements comme confirmés et libérer les verrous éventuels.

Cette architecture permet de rendre le système robuste, observable et industrialisable.

IDEO·LAB

IDEO·LAB

3. Le pipeline SRDF : capture, batch, transport, apply

Un pipeline modulaire, fiable et industrialisable pour la réplication sécurisée des données.

Fiabilité
séparation claire des phases

Performance
batching et déduplication

Sécurité
compression, chiffrement, traçabilité

SRDF transforme les changements SQL en flux répliqués, sécurisés, optimisés et supervisés de bout en bout.

IDEO·LAB

4. Orchestration, daemon et automatisation

Une autre fonctionnalité maîtresse du projet est son système d'orchestration.

SRDF n'est pas seulement une commande ponctuelle. Le projet prévoit un fonctionnement en mode service, via daemon ou systemd, capable de se réveiller régulièrement, par exemple toutes les 30 secondes, pour piloter les différentes phases.

L'orchestrateur SRDF doit gérer :

- le lancement de la capture ;
- la détection des événements en attente ;
- la construction des batches ;
- l'envoi des batches prêts ;
- le suivi des accusés de réception ;
- la relance en cas d'échec ;
- la mise à jour des états internes ;
- la supervision globale du pipeline.

Le daemon devient donc un **orchestrateur d'états**. Chaque phase retourne un résultat structuré, ce qui permet de savoir précisément si une étape a réussi, échoué, été ignorée ou doit être rejouée.

Cette logique est essentielle pour passer d'un prototype technique à un produit exploitable en production.

IDEO·LAB

4b – Orchestration de la capture - MariaDB / PostgreSQL - Oracle

Le projet prévoit notamment des commandes opérationnelles comme :

- `srdf_capture_mariadb_binlog`
- `srdf_transport`
- `srdf_apply_inbound`
- `srdf_run_bootstrap_plan`
- `srdf_monitor_status`
- `srdf_daemon`

Cette approche par commandes spécialisées permet de tester chaque phase séparément, puis de les intégrer progressivement dans un service automatisé.

Pour un investisseur, ce point est important : SRDF n'est pas seulement un script de réplication. C'est une future plateforme logicielle de réplication, pilotable, observable et automatisable.

IDEO·LAB

4. Orchestration, daemon et automatisation

IDEO · LAB

Un système d'orchestration qui pilote, supervise et relance l'ensemble du pipeline SRDF.

Commandes opérationnelles

```
> srdf_capture_mariadb_binlog
> srdf_transport
> srdf_apply_inbound
> srdf_run_bootstrap_plan
> srdf_monitor_status
> srdf_daemon
```

Le daemon devient un orchestrateur d'états : chaque phase retourne un résultat structuré, pilotable et rejouable.

Fiabilité
pilotage des phases

Observabilité
états, logs, supervision

Automatisation
service continu et relance

5. Bootstrap initial et reprise d'activité

Un système de réplication ne peut pas se limiter aux deltas. Il doit aussi être capable d'initialiser correctement un serveur cible.

SRDF prévoit donc un module de **bootstrap initial** permettant de préparer une base distante avant de démarrer la réplication continue.

Le bootstrap doit permettre :

- de sélectionner une base complète ;
- de sélectionner certaines tables critiques ;
- de créer les structures nécessaires sur le serveur cible ;
- de charger les données initiales ;
- d'exclure certaines tables techniques, Django ou SRDF ;
- de journaliser chaque étape ;
- de rejouer une opération en cas d'échec ;
- de consolider le résultat final.

Cette fonctionnalité est stratégique, car elle permet de répondre à un cas réel très fréquent : une entreprise possède déjà une base de production ancienne et veut mettre en place une réplication distante sans repartir de zéro.

Le scénario cible est le suivant :

1. Préparer le serveur distant.
2. Créer ou synchroniser les schémas.
3. Charger les tables critiques.
4. Placer un checkpoint de capture au bon endroit.
5. Démarrer la réplication continue des deltas.

IDEO·LAB

5. BOOTSTRAP INITIAL ET REPRISE D'ACTIVITÉ

Préparer, charger et sécuriser la base cible avant la réplication continue

1 **SÉLECTION DES DONNÉES**
Sélectionner une base complète ou des tables critiques.

2 **FILTRAGE INTELLIGENT**
Inclure les données necessaires, exclure les tables techniques, Django ou SRDF.

3 **CRÉATION DES STRUCTURES**
Créer les bases, tables, index et contraintes sur le serveur cible.

4 **CHARGEMENT DES DONNÉES**
Charger les données initiales de manière sécurisée et optimisée.

5 **JOURNALISATION DE CHAQUE ÉTAPE**
Tracer et enregistrer chaque action pour assurer la traçabilité.

6 **REJOUABILITÉ**
Permettre de rejouer une opération en cas d'échec ou d'interruption.

7 **CONSOLIDATION**
Vérifier l'intégrité et consolider le résultat final.

8 **PRÊT POUR LA RÉPLICATION**
La base cible est prête pour basculer en mode réplication continue.

POURQUOI C'EST STRATÉGIQUE ?

Permettre à une entreprise de répliquer une base existante vers un site distant sans repartir de zéro.

SÉCURISÉ | RAPIDE | EFFICACE | FIABLE

6. Sécurité, fiabilité et contrôle des erreurs

SRDF intègre plusieurs mécanismes indispensables à une solution de réplication sérieuse.

La sécurité repose notamment sur :

- le chiffrement des batches ;
- les tokens API ;
- la communication contrôlée entre nœuds ;
- la séparation source / cible ;
- la journalisation des actions critiques.

La fiabilité repose sur :

- les checkpoints ;
- les accusés de réception ;
- l'idempotence ;
- les retries ;
- les mécanismes de backoff ;
- les files d'attente ;
- les dead letters pour les erreurs non récupérables ;
- les journaux d'exécution ;
- les tables d'audit.

L'objectif est d'éviter les deux grands dangers de la réplication :

1. **perdre une modification** ;
2. **appliquer deux fois la même modification.**

SRDF doit donc savoir exactement quels événements ont été capturés, envoyés, reçus, appliqués et confirmés.

IDEO·LAB

6. SÉCURITÉ, FIABILITÉ ET CONTRÔLE DES ERREURS

Des mécanismes robustes pour une réplication sécurisée, fiable et maîtrisée

SÉCURITÉ

Protéger les données et les échanges

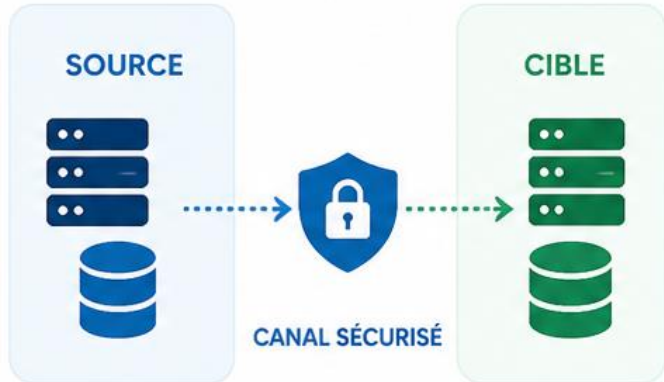
CHIFFREMENT DES BATCHES
Les données sont chiffrées avant transport.

TOKENS API
Authentification forte et sécurisée des composants.

COMMUNICATION CONTRÔLÉE ENTRE NŒUDS
Échanges sécurisés et authentifiés de bout en bout.

SÉPARATION SOURCE / CIBLE
Isolation stricte des environnements.

JOURNALISATION DES ACTIONS CRITIQUES
Traçabilité complète et infalsifiable des opérations.



Données chiffrées et authentifiées

TRAÇABILITÉ & AUDIT
Toutes les actions critiques sont journalisées et archivées.

FIABILITÉ

Assurer la continuité et la résilience

CHECKPOINTS
Points de reprise pour garantir la continuité.

ACCUSÉS DE RÉCEPTION
Confirmation systématique des données reçues.

IDEMPOTENCE
Ré-exécution sans effet de bord.

RETRIES
Nouvelles tentatives automatiques en cas d'échec.

MÉCANISMES DE BACKOFF
Gestion intelligente des temporisations.

FILES D'ATTENTE
Ordonnancement et gestion du flux.

DEAD LETTERS
Isolation des erreurs non récupérables.

JOURNAUX D'EXÉCUTION
Suivi détaillé de toutes les opérations.

TABLES D'AUDIT
Historique fiable pour contrôle et conformité.

L'OBJECTIF : ÉVITER LES 2 GRANDS DANGERS DE LA RÉPLICATION

PERTE DE DONNÉES
Aucune donnée ne doit être perdue entre la source et la cible.

CORRUPTION SILENCIEUSE
Aucune donnée ne doit être altérée sans détection.

7. Supervision, logs et exploitabilité

Un autre axe fort du projet est l'exploitabilité.

SRDF doit fournir des logs lisibles, des compte-rendus d'exécution, des compteurs et des états clairs. L'objectif est qu'une petite équipe technique puisse comprendre rapidement :

- combien d'événements sont en attente ;
- combien ont été capturés ;
- combien sont dans un batch ;
- combien ont été envoyés ;
- combien ont été appliqués ;
- quels batches sont en erreur ;
- quel est le dernier checkpoint connu ;
- quel serveur est disponible ou indisponible ;
- quelle phase du pipeline bloque.

Le projet prévoit donc une logique de tables de suivi, d'audit et de monitoring, par exemple autour des entités :

- `OutboundChangeEvent`
- `TransportBatch`
- `AuditEvent`
- `ReplicationCheckpoint`
- tables de bootstrap ;
- tables de runtime ;
- tables d'état des services.

IDEO·LAB

8. Positionnement marché et avantage compétitif

Le positionnement de SRDF by IDEO-LAB est clair : rendre accessible une classe de fonctionnalités historiquement réservées aux grands comptes.

Les solutions propriétaires de réplication et de continuité d'activité sont puissantes, mais souvent :

- coûteuses ;
- complexes ;
- liées à un matériel spécifique ;
- difficiles à déployer ;
- pensées pour de grandes équipes d'infrastructure ;
- peu adaptées aux startups, PME et petits éditeurs SaaS.

SRDF vise une approche différente :

- solution logicielle ;
- indépendante du hardware ;
- compatible infrastructure standard ;
- orientée SQL et Linux ;
- portable sur cloud ou bare metal ;
- progressive ;
- automatisable ;
- économiquement beaucoup plus accessible.

IDEO·LAB

9b – Un positionnement stratégique

La proposition de valeur est donc forte :

offrir une réplication sécurisée, pilotable et industrialisable à des entreprises qui n'ont ni le budget, ni les équipes, ni l'envie de déployer des solutions propriétaires lourdes.

C'est un positionnement particulièrement pertinent pour :

- SaaS B2B ;
- PME critiques ;
- hébergeurs spécialisés ;
- éditeurs Django / Python ;
- plateformes e-commerce ;
- startups manipulant des données clients ;
- entreprises souhaitant une PRA simple ;
- structures voulant répliquer vers un cloud secondaire.

IDEO·LAB

9. Évolutions prévues

Le projet est conçu pour évoluer progressivement.

La première version doit se concentrer sur un périmètre réaliste :

- MariaDB / MySQL ;
- capture binlog ;
- construction de batches ;
- transport vers cible ;
- application SQL distante ;
- accusés de réception ;
- bootstrap initial ;
- daemon d'orchestration ;
- logs et supervision.

Les versions suivantes peuvent ouvrir des axes plus ambitieux :

- réplication parallèle ;
- support PostgreSQL ;
- installation automatique du serveur distant ;
- API standalone sur le serveur remote ;
- auto-diagnostic ;
- intégration cloud AWS, puis GCP et Azure ;
- moteur de comparaison source / cible ;
- dashboard d'administration avancé ;
- gestion multi-tenant ;
- politique de priorité métier des données ;

IDEO·LAB

Conclusion

SRDF by IDEO-LAB est un projet de réplication et de reprise d'activité pensé pour les infrastructures modernes. Sa valeur repose sur une idée simple : rendre accessibles des mécanismes de synchronisation avancés sans imposer de matériel propriétaire, de budgets massifs ou d'architectures datacenter complexes.

Ses fonctionnalités maîtresses sont :

- la capture intelligente des changements SQL ;
- la réplication des données et des structures ;
- le batching et la déduplication ;
- la compression et le chiffrement ;
- le transport sécurisé vers un serveur distant ;
- l'application contrôlée côté cible ;
- les accusés de réception ;
- le bootstrap initial ;
- l'orchestration par daemon ;
- la supervision complète du pipeline.

Pour un investisseur, SRDF représente une opportunité produit sur un besoin réel : la continuité d'activité abordable pour les PME, startups, éditeurs SaaS et infrastructures hybrides.
Le projet combine une vision technique forte, une cible marché claire et un différenciateur économique évident : **faire beaucoup plus simple, beaucoup plus portable et beaucoup moins cher que les solutions propriétaires historiques.**

POURQUOI SRDF

IDEO·LAB

POSITIONNEMENT DU SRDF PAR IDEO-LAB

Promesse produit

- 10 à 20 fois moins cher
- beaucoup plus simple à déployer
- agnostique infra : AWS, GCP, Azure, OVH, bare metal, VM Linux, et même Windows si besoin
- pas lié à une baie propriétaire
- orienté reprise d'activité réelle, pas luxe datacenter réservé aux grands groupes

Là où nous sommesvéritablement meilleurs

- Installation légère, autonome, industrialisée
- Réplication multi-moteurs SQL standards : MySQL, MariaDB, puis PostgreSQL
- Compression + chiffrement natifs
- File d'attente / batching intelligent pour limiter le coût réseau
- Priorisation métier des données critiques
- Dashboard clair, logs lisibles, exploitabilité simple
- Fonctionnement sur machines ordinaires, pas sur matériel EMC hors de prix
- Mode "disaster recovery pragmatique" pour petites équipes techniques

SRDF IDEO-LAB VS SRDF Dell

La vraie rupture

Dell vend une solution de très haut de gamme, très robuste, mais lourde, chère, fermée et pensée pour de grosses entreprises.

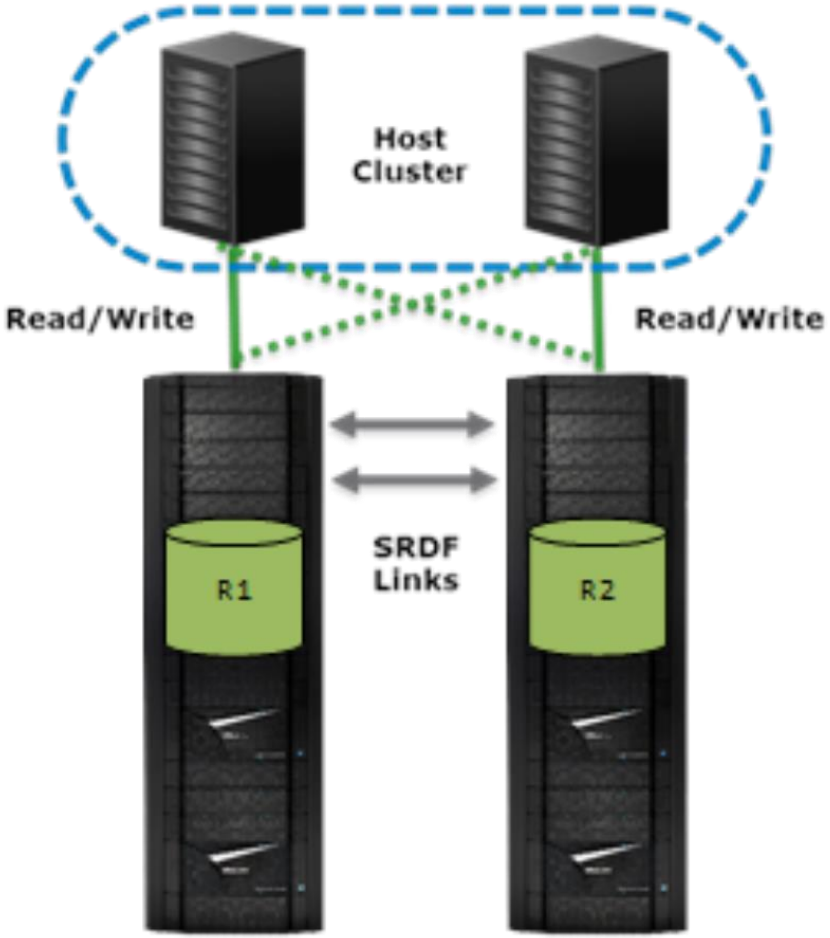
Toi, tu peux construire une solution :

- ouverte
- portable
- déployable partout
- beaucoup plus accessible économiquement
- adaptée aux infrastructures modernes hybrides et cloud

DELL SRDF EN QUELQUES MOTS ET CHIFFRES

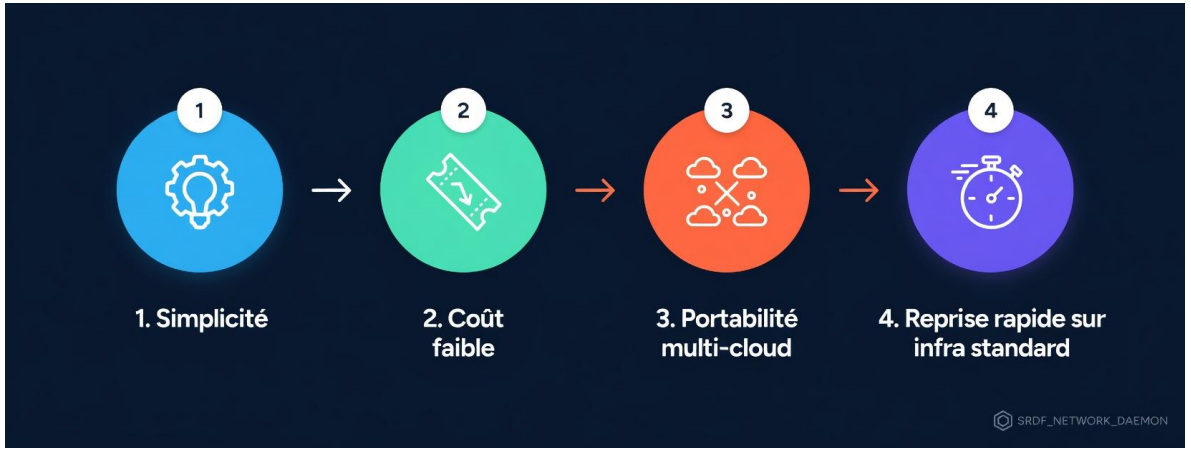
- **Petit périmètre** : quelques To protégés → on peut vite être dans une logique de **quelques milliers à quelques dizaines de milliers d'euros** rien que pour la couche SRDF/licences.
- **Projet entreprise réaliste** avec 2 baies, support, services, réseau, installation → on bascule souvent sur des **budgets de plusieurs dizaines de milliers à plusieurs centaines de milliers d'euros**, voire davantage selon la taille et la haute dispo visée. Cette estimation est une **inférence** à partir du fait que SRDF s'appuie sur des plateformes PowerMax mission-critical et sur une tarification à la capacité/licence, pas un prix public Dell affiché. Dell +2

SRDF / METRO DELL



SRDF IDEO-LAB

1. Simplicité
2. Coût faible
3. Portabilité multi-cloud
4. Reprise rapide sur infra standard



PLAN GENERAL DU DOCUMENT SRDF

IDEO·LAB

Table des Matières

SRDF
PLAN GÈNERAL DU DOCUMENT

TABLE DES MATIERES

PROJET SRDF

TABLE DES MATIERES - 1-3

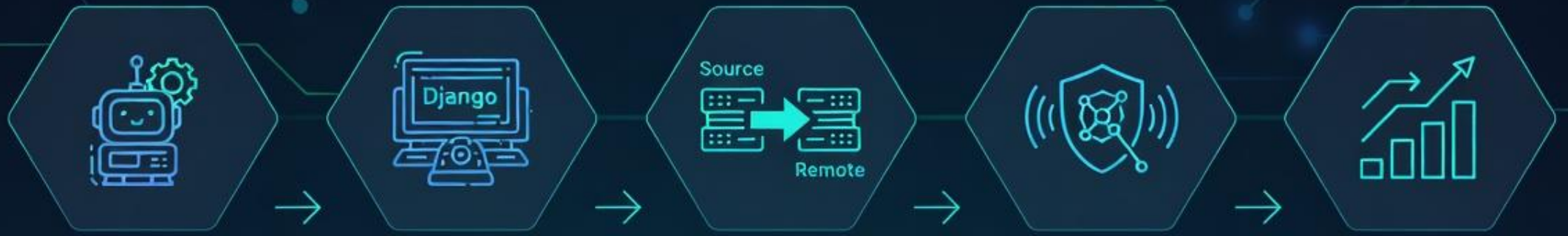
TABLE DES MATIERES - 4 - 8

TABLE DES MATIERES - 9-10

==========

SRDF LES OBJECTIFS POURSUIVIS

Architecture SRDF for Startups



Local Agent

Local agent
deamon par moteur

Control Plane

Control plane
Django central

Data Plane

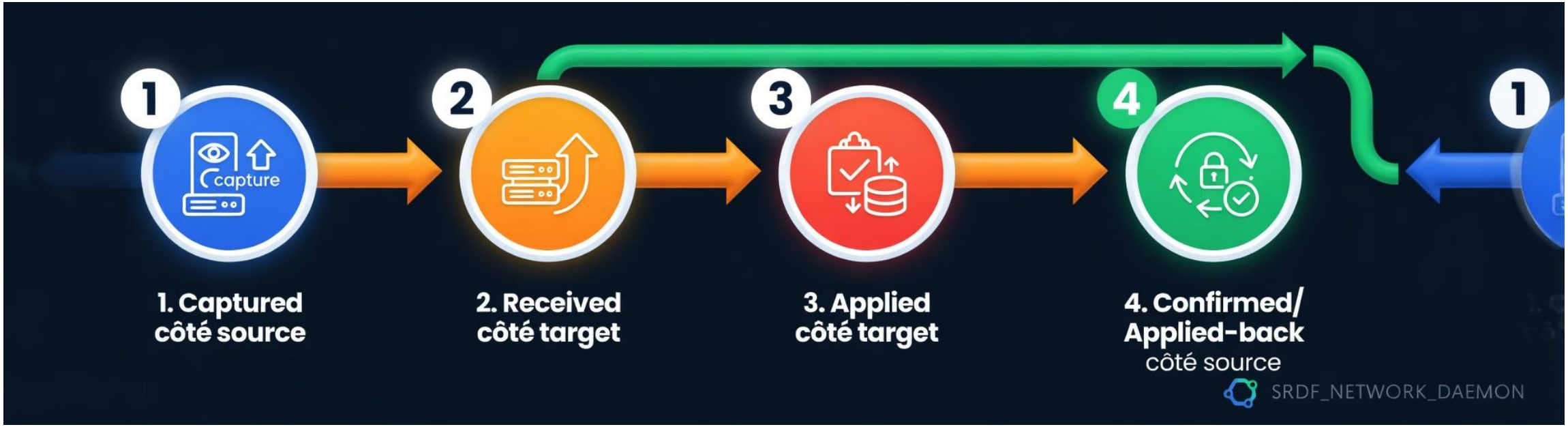
Data plane dédié
source → remote

Meilleur Mode

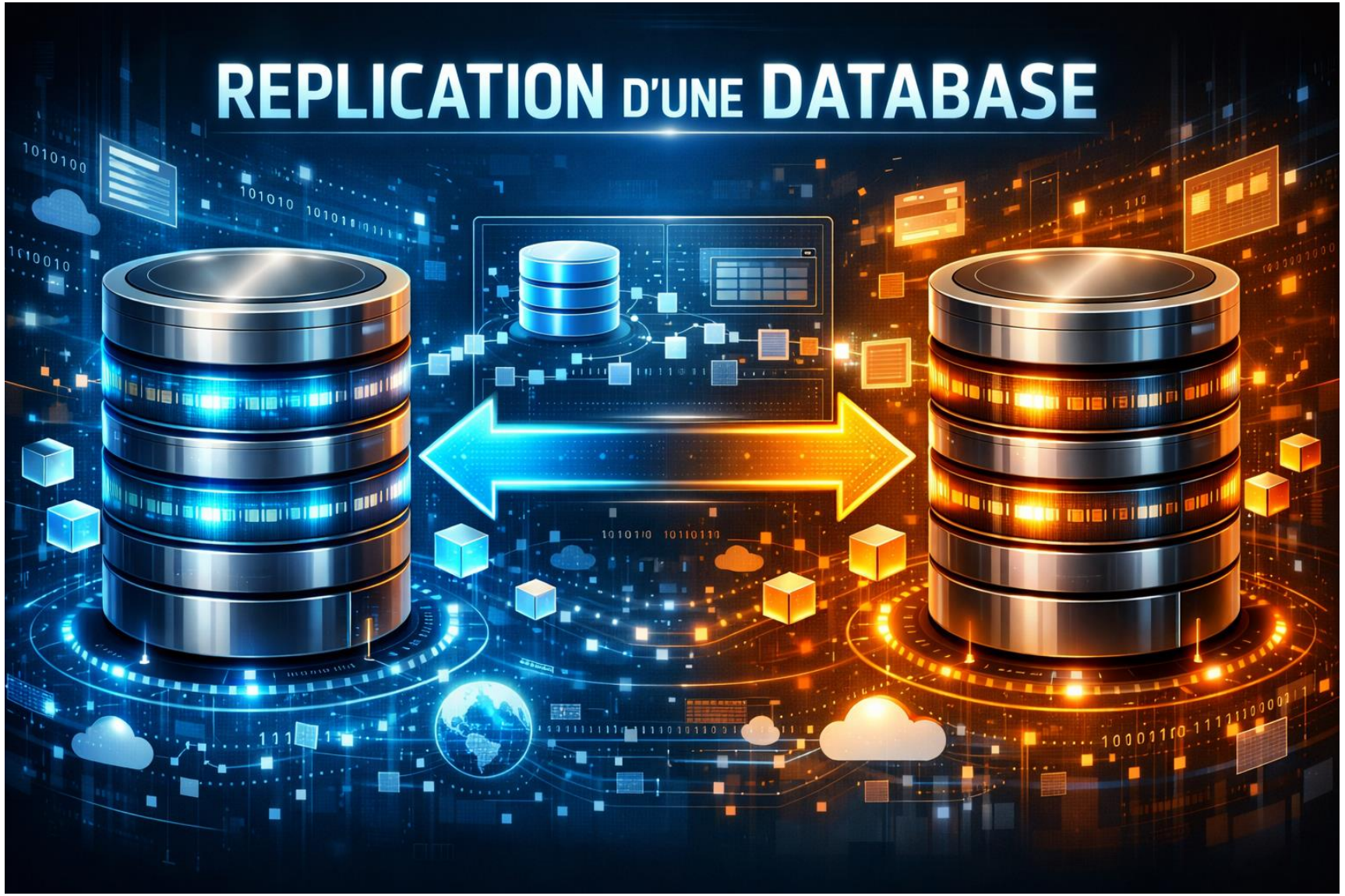
Meilleur mode de
communication

Plan de Dev

Plan de dev
étape par étape

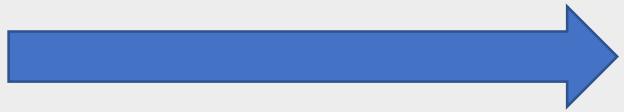

1
capture
2
3
4
1
1. Captured
côté source
2. Received
côté target
3. Applied
côté target
4. Confirmed/
Applied-back
côté source
SRDF_NETWORK_DAEMON

LA REPLICATION SRDF : PERIMETRE



OBJETS REPLIQUES SOUS MARIADB/MYSQL

- DATABASE
- TABLE
- INDEX
- COLONNE



- CREATE
- ALTER
- DROP

DDL

- UPDATE OF TABLE’S ROW
- INSERT VALUE INTO TABLE
- DELETE OF TABLE’S ROW

DML

OBJETS SQL CONCERNES

OBJETS REPLIQUES

OBJETS SOURCE

- DATABASE
- TABLE
- INDEX
- COLONNE

OBJETS REPLIQUES

- DATABASE
- TABLE
- INDEX
- COLONNE

COMMANDES DDL

- CREATE
- ALTER
- DROP

OPERATIONS SUR LES DONNÉES (DML)

UPDATE OF TABLE

Mettre à jour une table

INSERT VALUE INTO TABLE

Insérer des valeurs dans une table

DELETE OF TABLE'S ROW

Supprimer une ligne d'une table

Réplication synchronisée

REPLICATION D'UNE DATABASE

DATABASE SOURCE

PROCESSUS DE REPLICATION

CAPTURE
Lecture des modifications sur la source (Log, Binlog, ...)

TRANSFERT
Envoi des changements à la cible

APPLICATION
Rejeu des opérations sur la base cible

REPLICATION EN COURS...

DATABASE CIBLE (REPLIQUE)

TABLE: UTILISATEURS

id	nom	email
1	Alice	alice@mail.com
2	Bob	bob@mail.com

TABLE: PRODUITS

id	nom	prix
1	Clavier	50
2	Souris	30

TABLE: COMMANDES

id	user_id	total
1	1	80
2	2	30

OBJETS REPLIQUES

DDL (Structure)
- DATABASE
- TABLE
- INDEX
- COLONNE

COMMANDES (DML/DCL)
- CREATE
- ALTER
- DROP

OPERATIONS SUR LES DONNEES
- UPDATE OF TABLE
- INSERT VALUE INTO TABLE
- DELETE OF TABLE'S ROW

TABLE: UTILISATEURS

id	nom	email
1	Alice	alice@mail.com
2	Bob	bob@mail.com

TABLE: PRODUITS

id	nom	prix
1	Clavier	50
2	Souris	30

TABLE: COMMANDES

id	user_id	total
1	1	80
2	2	30

OBJECTIF : ASSURER LA CONSISTANCE ET LA DISPONIBILITE DES DONNEES SUR LA BASE CIBLE

SRDF : C’est quoi ?

Notre solution permet une **synchronisation temps réel** des bases de données (MySQL, MariaDB, PostgreSQL, Oracle) entre un serveur d’origine et un serveur distant, avec une latence minimale et une fiabilité maximale.

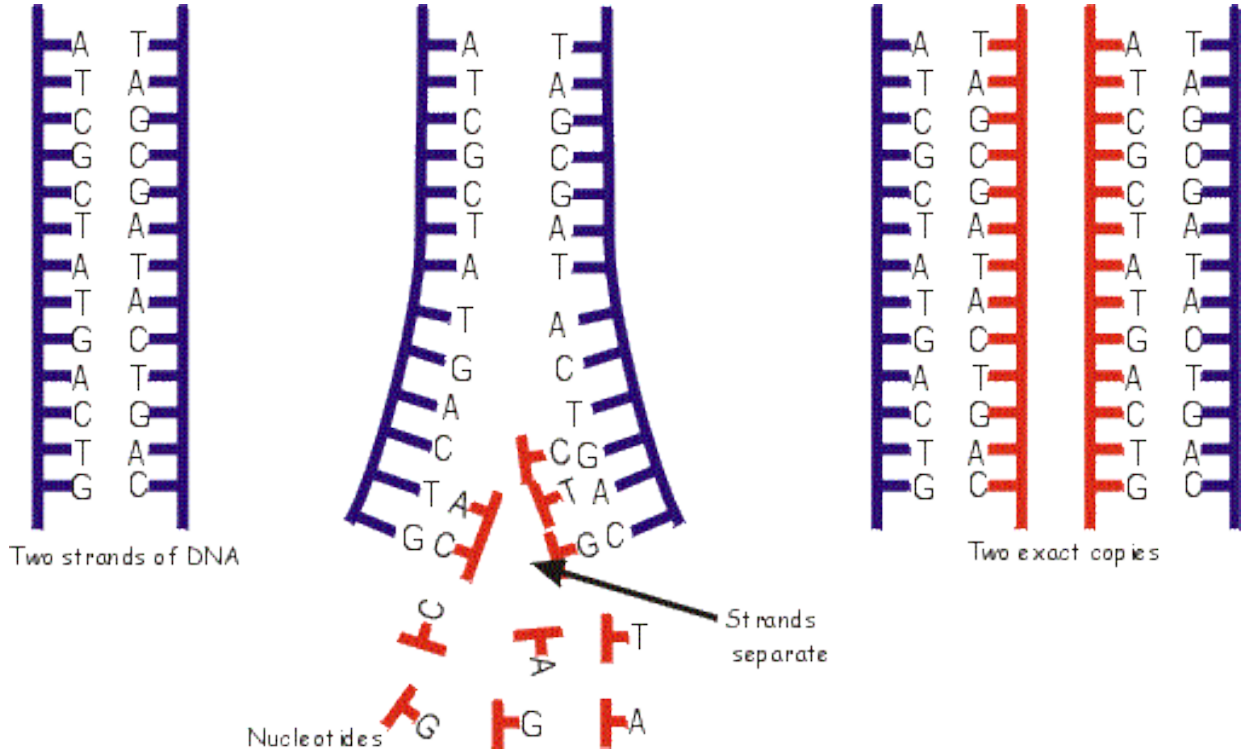
Le processus est le suivant :
Dès qu’une modification SQL (Insert, Update, Delete, Create…) est effectuée sur la base origine, elle est capturée, regroupée par paquets cohérents, dé-dupliquée, compressée et chiffrée.

Un batch s’exécute toutes les 30 secondes pour envoyer uniquement les mises à jour non encore synchronisées. Avant envoi, le système vérifie la disponibilité du serveur distant et gère les erreurs avec reprise automatique.

À réception, les données sont contrôlées pour éviter toute duplication, puis appliquées sur la base cible. Enfin, un accusé de réception est renvoyé pour confirmer la bonne synchronisation.

Grâce à ce mécanisme robuste, nous assurons une **réplication fiable, sécurisée et optimisée** des bases de données, idéale pour les startups qui ont besoin de haute disponibilité, de reprise après sinistre ou de bascule géographique sans perte de données.

LA REPLICATION



LA REPLICATION D'UN SERVEUR - SYSTEM

3. Delta services

État et définition :

- service activé/désactivé
- service ajouté
- override systemd modifié
- timer ajouté
- cron modifié

4. Delta data applicative hors DB

Exemples :

- uploads critiques
- fichiers générés
- cache persistant utile
- répertoires métiers

2. Delta packages / runtimes

Ce qui a été installé, mis à jour, supprimé :

- apt/dpkg
- pip/venv
- éventuellement npm, binaries maison, etc.

3. Delta services

État et définition :

- service activé/désactivé
- service ajouté
- override systemd modifié
- timer ajouté
- cron modifié

1. Delta filesystem

Tout ce qui change dans les zones utiles :

- `/etc/nginx/`
- `/etc/systemd/`
- `/etc/mysql/`
- `/opt/app/`
- `/var/www/...`
- scripts d'exploitation
- certificats
- templates
- fichiers de config métiers

Pas tout le filesystem, seulement les zones pilotées.

LA REPLICATION D'UN SERVEUR - CONFIG

5. Delta configuration système

- firewall
- users/groups techniques
- sudoers ciblés
- variables d'environnement de services
- montages ou points de montage logiques

6. Delta secrets / certificats

À traiter avec précaution extrême :

- renouvellement TLS
- clés privées
- tokens applicatifs
- credentials de réplication

LA REPLICATION : A NE JAMAIS FAIRE !

Ce qu'il ne faut surtout pas faire

Tu as aussi raison implicitement sur un point important : il ne faut pas tomber dans le piège du "**replicate everything**".

Sinon on refait un mauvais clone de disque.

Il ne faut pas répliquer naïvement :

- `/proc`
- `/sys`
- `/dev`
- sockets
- pid files
- caches temporaires
- gros logs volatils
- machine-id tel quel
- clés host SSH si non voulu
- IP / identité réseau source
- artefacts temporaires du noyau

LA REPLICATION : CONCEPTS

A. Une baseline

On capture une photographie de référence du serveur :

- packages
- services
- arbres surveillés
- configs
- users techniques
- métadonnées critiques

B. Un journal de changements

Ensuite, on ne renvoie plus tout.

On envoie uniquement des événements ou mini-deltas :

- fichier X modifié
- fichier Y supprimé
- paquet Z installé
- service nginx restart demandé
- override systemd changé
- cert renouvelé
- venv hash modifié

C. Un apply engine sur la cible

La cible reçoit ces deltas et les applique :

- dans l'ordre,
- avec version,
- avec checksum,
- avec validation,
- avec rollback local si besoin.

LA REPLICATION : ENGINE

Option 2 — Scan incrémental très fréquent

Toutes les N secondes :

- checksum/mtime/size
- diff des services
- diff paquets
- diff configs
- diff users/groups ciblés

Avantages :

- plus robuste
- plus simple à développer
- plus facile à auditer

Inconvénients :

- moins instantané
- plus coûteux si mal conçu

LA REPLICATION : RPO

La notion clé : RPO quasi nul, pas “backup”

Tu décris un objectif de type :

- **RPO très faible**, potentiellement < 1 minute
- **RTO très faible**, quelques minutes
- sans refaire une restauration massive

Donc le produit doit viser :

- serveur cible déjà prêt,
- déjà provisionné,
- déjà compatible,
- déjà “chaud”,
- simplement maintenu en phase.

Le jour du sinistre, on ne “restaure” pas.
On fait plutôt :

- stop final des flux,
- validation dernier delta,
- promotion,
- remapping réseau/DNS,
- reprise de services.

C’est très différent d’un backup.

LA REPLICATION : LES REGLES

1. Ordre des opérations

Exemple :

- copier config avant restart service
- installer package avant déployer override
- restaurer cert avant reload nginx

Donc chaque delta doit être typé et ordonnançable.

2. Idempotence

Si un delta est rejoué deux fois, la cible ne doit pas casser.

3. Vérification

Après application :

- checksum fichier
- état service
- version paquet
- healthcheck applicatif

4. Reconciliation

Si un delta a été raté :

- rescan
- diff
- resync sélectif

5. Cohérence multi-objets

Exemple :

- config nginx + cert + app + socket + service doivent converger ensemble.

LA REPLICATION : BASE ARCHITECTURE

1. Host Agent

Installé sur le primaire :

- observe
- capture
- compacte
- signe/chiffre
- pousse les deltas

2. SRDF Control Plane

Dans Django :

- inventaire
- policies
- filesets
- snapshots
- delta logs
- état des nœuds
- supervision
- commandes de failover

3. Target Apply Agent

Sur le secondaire :

- reçoit
- met en file
- applique
- vérifie
- expose son état

4. Promotion Engine

En cas d'incident :

- gèle la file
- valide dernier état cohérent
- démarre les services dans l'ordre
- bascule IP / DNS / rôle

LA REPLICATION : Déploiement

V1 réaliste

Pour rester sérieux techniquement, la V1 pourrait se limiter à :

- réplication de fichiers ciblés
- réplication paquets apt/dpkg
- réplication services systemd
- réplication cron/timers
- réplication certificats/configs
- snapshots de cohérence
- commande de promotion manuelle

Sans toucher encore à :

- réseau complexe
- kernel tuning avancé
- containers
- block devices
- SELinux/AppArmor avancé
- identité machine complète

LA REPLICATION : Déploiement

V2

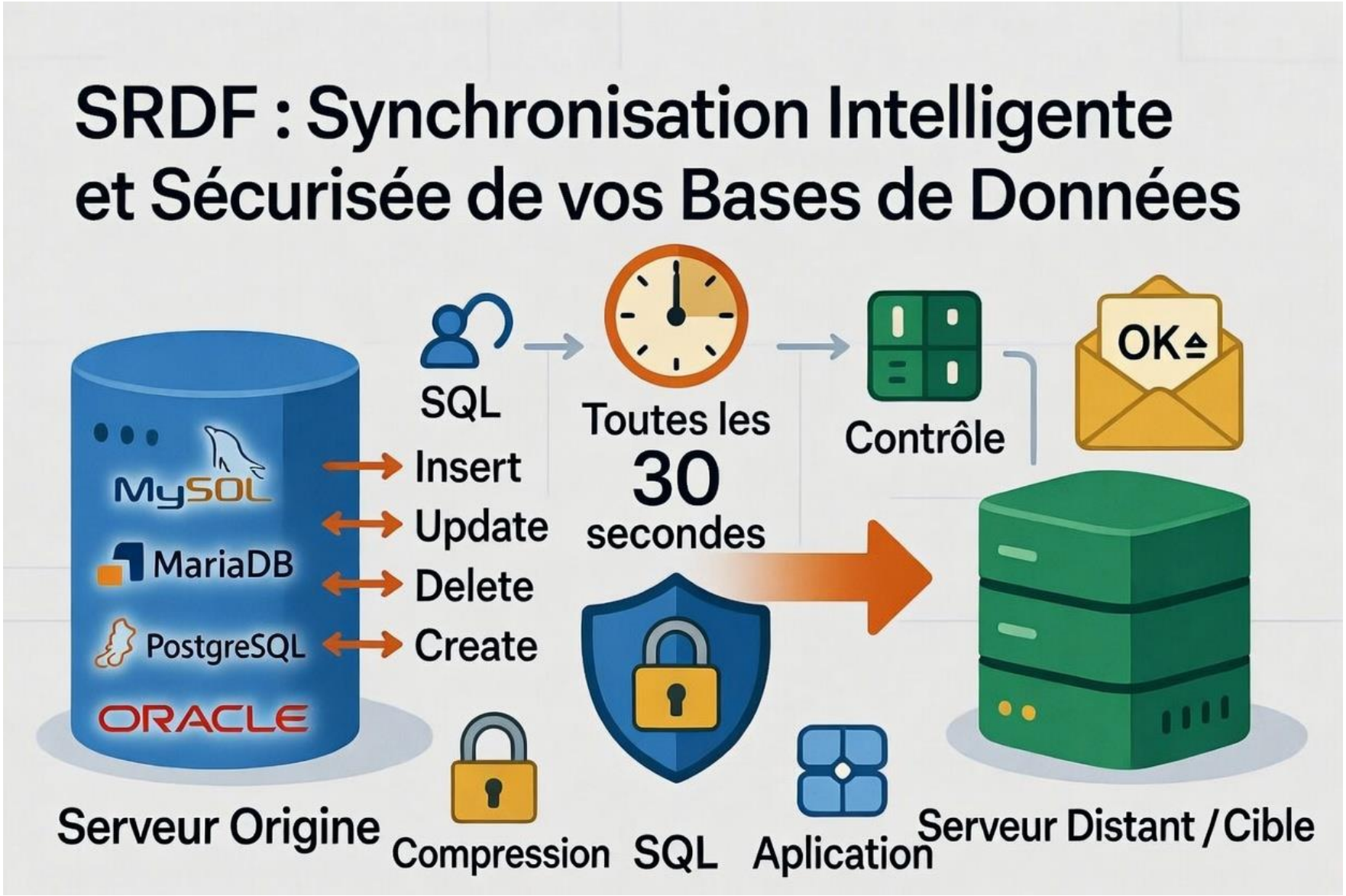
Ensuite :

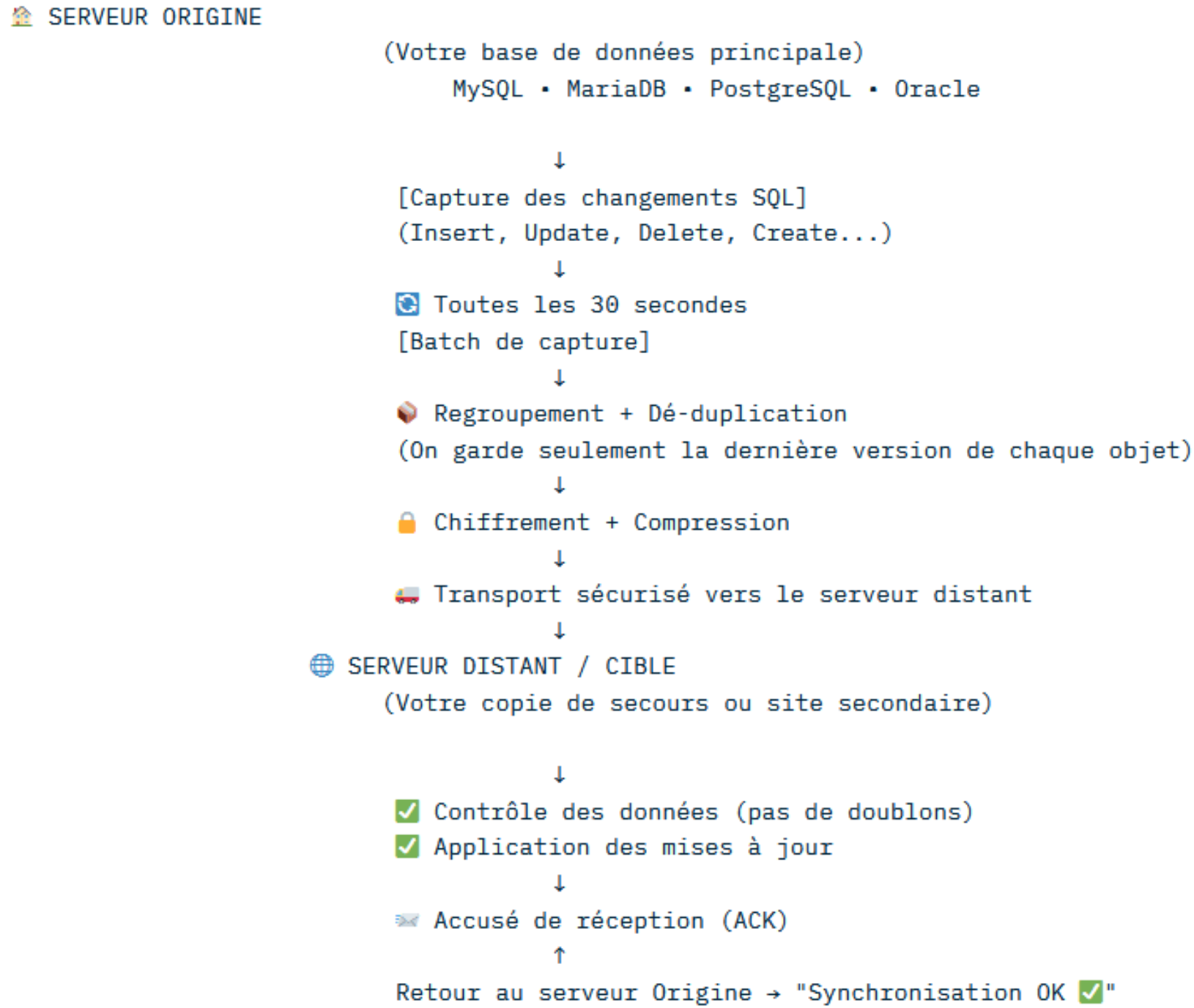
- users/groups techniques
- firewall
- Python venv / dépendances applicatives
- secrets via coffre
- replay plus rapide
- bascule semi-automatique
- tests de PRA

V3

Puis :

- multi-nodes
- dépendances inter-serveurs
- orchestration de rôle
- clusters applicatifs
- promotion coordonnée
- règles métier de bascule


SRDF : Synchronisation Intelligente
et Sécurisée de vos Bases de Données
SQL
Toutes les
30
secondes
Contrôle
OK
MySQL
MariaDB
PostgreSQL
ORACLE
Insert
Update
Delete
Create
Serveur Origine
Compression
SQL
Aplication
Serveur Distant / Cible


🏠 SERVEUR ORIGINE
(Votre base de données principale)
MySQL • MariaDB • PostgreSQL • Oracle
↓
[Capture des changements SQL]
(Insert, Update, Delete, Create...)
↓
🔁 Toutes les 30 secondes
[Batch de capture]
↓
📦 Regroupement + Dé-duplication
(On garde seulement la dernière version de chaque objet)
↓
🔒 Chiffrement + Compression
↓
🚚 Transport sécurisé vers le serveur distant
↓
🌐 SERVEUR DISTANT / CIBLE
(Votre copie de secours ou site secondaire)
↓
✅ Contrôle des données (pas de doublons)
✅ Application des mises à jour
↓
📨 Accusé de réception (ACK)
↑
Retour au serveur Origine → "Synchronisation OK ✅"

Un résumé ultra-simple

Imaginez que votre base de données principale (à gauche) est comme une **cuisine centrale** où toutes les commandes arrivent.

Dès qu'un plat (une modification) est préparé, notre système le **capture**, le met dans une **boîte bien organisée** (paquet), le rend plus léger et sécurisé, puis l'envoie rapidement à votre **deuxième cuisine** (serveur distant).

Une fois arrivé, on vérifie que tout est bon, on le sert, et on renvoie un petit mot : « Tout est bien reçu ! ».

Tout cela se fait **automatiquement toutes les 30 secondes**, sans doublons, sans erreurs, et en toute sécurité.

Pourquoi SRDF est largement supérieur aux solutions classiques ?

Contrairement aux outils traditionnels (réplication native MySQL/PostgreSQL ou solutions lourdes type Oracle GoldenGate), SRDF offre une **synchronisation intelligente toutes les 30 secondes** avec un contrôle total : groupement intelligent, dé-duplication automatique, chiffrement et compression intégrés.

Il consomme très peu de ressources, évite les doublons et les mises à jour inutiles, et gère les erreurs avec reprise automatique.

Résultat : une réplication **fiable, sécurisée, ultra-rapide et légère**, idéale pour les startups qui veulent une haute disponibilité et une reprise après sinistre sans complexité ni coût exorbitant.

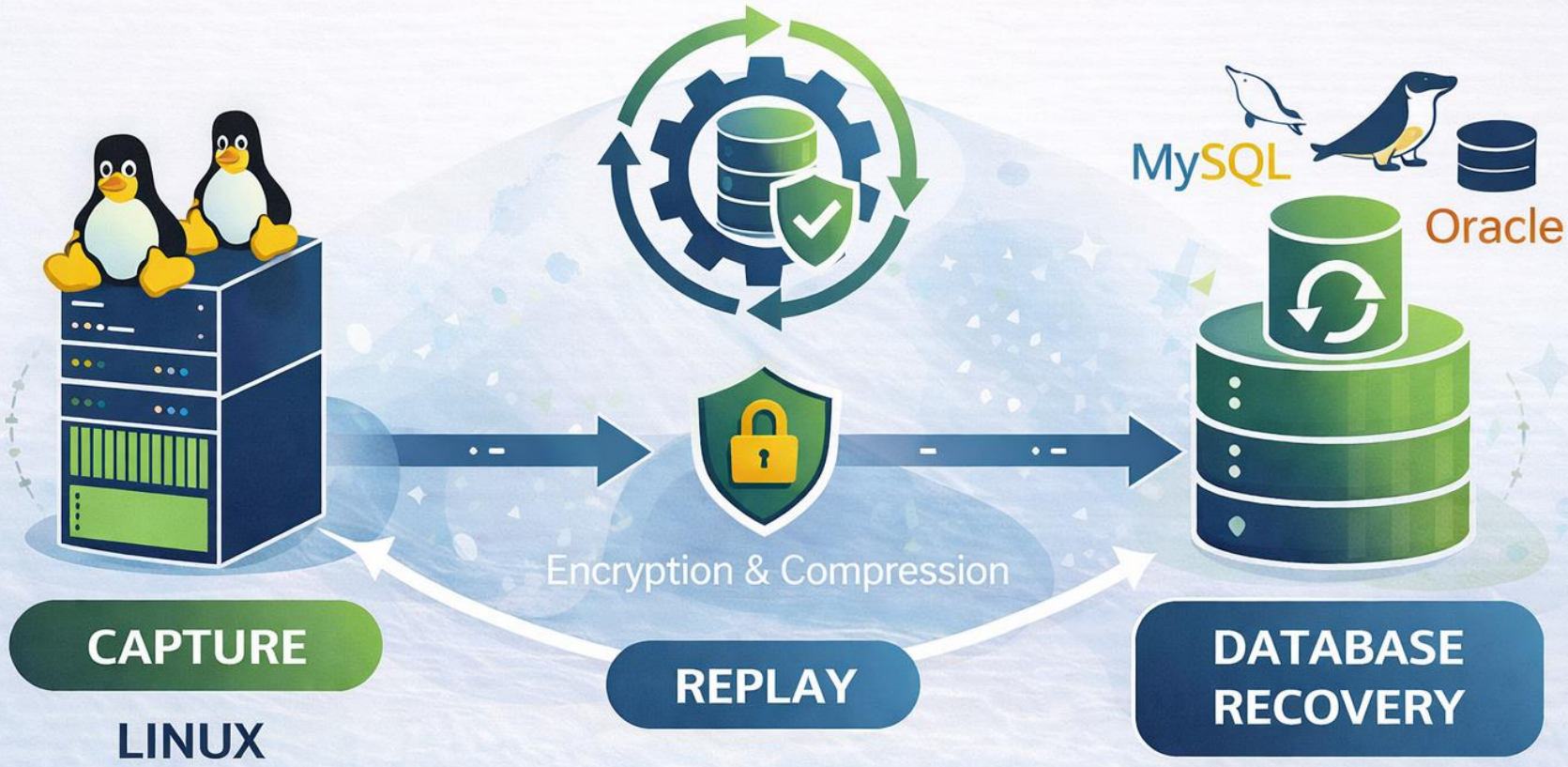
SRDF transforme une contrainte technique en **avantage compétitif** : vos données sont toujours à jour, partout, en toute sécurité, sans ralentir vos applications.

Solution	⚡ Points forts	⚠️ Limites	🎯 Cible
SRDF	• Batch optimisé + dédup • Compression + chiffrement • Faible consommation • Reprise automatique	• Moins mature	Startups / scale-ups solution moderne & optimisée
Native	• Gratuit / intégré • Latence temps réel • Setup simple	• Pas de dédup • Pas de compression native • Consomme bande passante • Gestion manuelle erreurs	Environnements simples budget limité
GoldenGate	• Très mature • Latence très faible • Multi-DB robuste	• Très cher • Complexe • Ressources élevées	Grandes entreprises Oracle-centric
AWS DMS	• Managed AWS • Facile à déployer • Multi-sources	• Latence variable • Coûts évolutifs • Lock-in AWS	Projets AWS migration simple

SRDF 2026
DOIT ETRE

- SIMPLE
- EFFICACE
- RAPIDE
- FIABLE
- SECURE
- EVOLUTIF
- ABORDABLE

SRDF IDEO-LAB : GENERAL FLOW



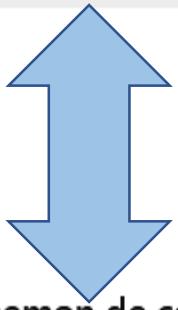
SRDF COMMUNICATIONS

SRDF C'EST 4 BRIQUES DISTINCTES

1. Orchestrator Local

Sur le serveur source, il pilote :

- capture
- dedup
- build wagon / batch
- envoi du wagon vers le remote
- attente ACK / confirmation d'application
- mise à jour des `OutboundChangeEvent`



3. Mini serveur / daemon de communication Local

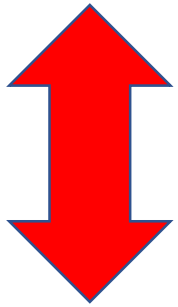
Côté source, exposé en API, pour recevoir les callbacks du remote :

- ACK batch reçu
- APPLY batch terminé
- APPLY batch failed

2. Orchestrator Remote

Sur le serveur target, il pilote :

- réception du wagon
- enregistrement dans `InboundChangeEvent`
- apply SQL sur la database target
- marquage des inbound en `applied` / `failed`
- retour d'état vers le source

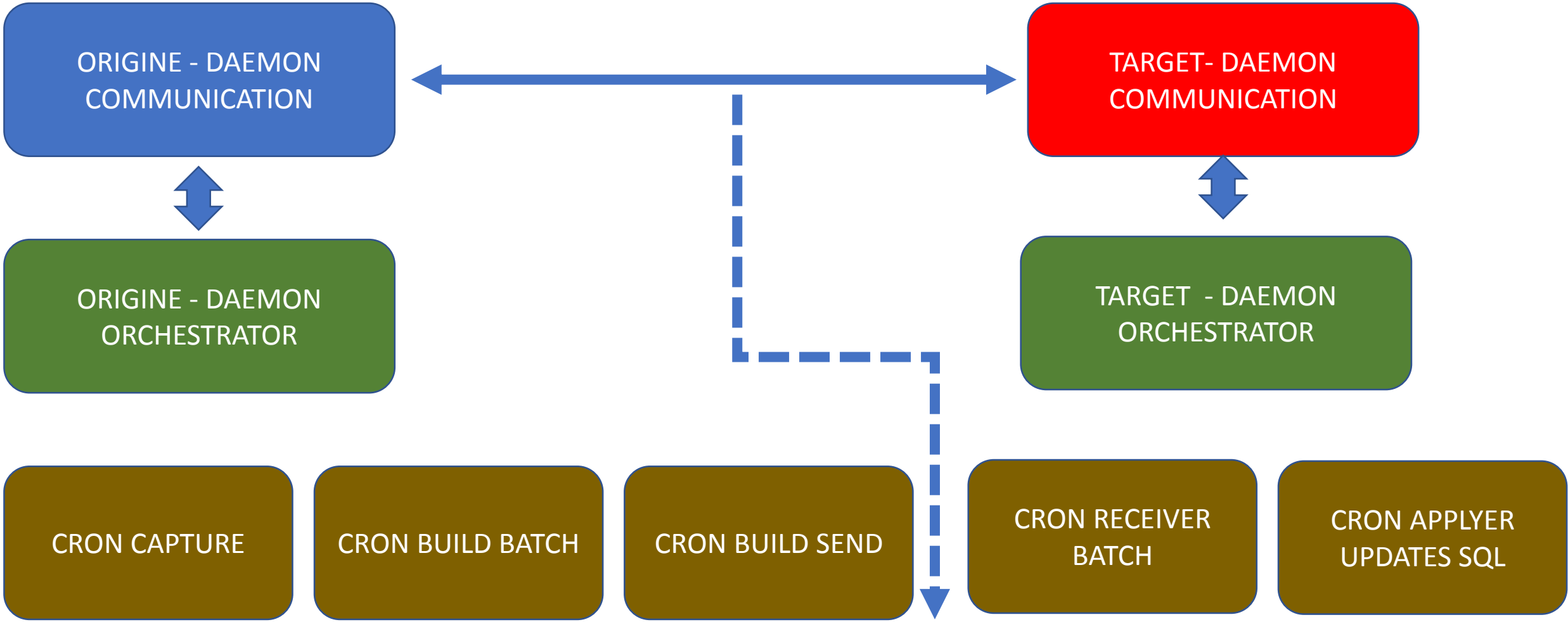


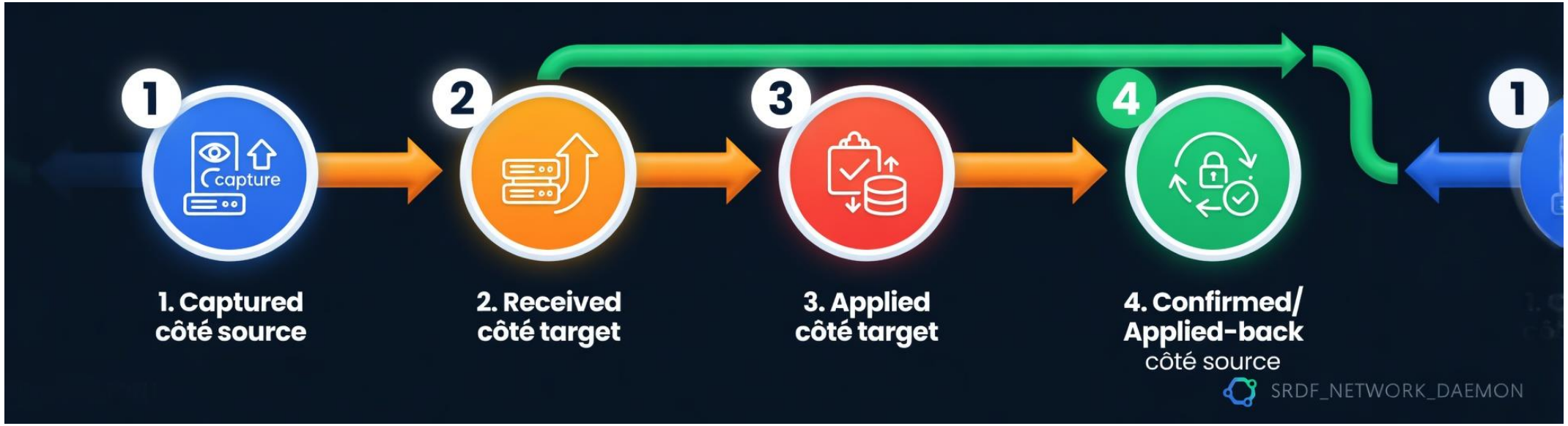
4. Mini serveur / daemon de communication Remote

Côté target, exposé en API, pour recevoir les demandes du source :

- `receive_wagon`
- éventuellement `apply_batch`
- éventuellement `health/ping`

ARCHITECTURE DE COMMUNICATION




1
capture
2
3
4
1
1. Captured
côté source
2. Received
côté target
3. Applied
côté target
4. Confirmed/
Applied-back
côté source
SRDF_NETWORK_DAEMON

RAPPEL
PRINCIPE
ARCHITECTURE
GLOBAL

Les 4 états métier à distinguer

Tu veux implicitement 4 niveaux, et c'est très bien :

1. **captured** côté source
2. **received** côté target
3. **applied** côté target
4. **confirmed/applied-back** côté source

C'est seulement au niveau 4 qu'on peut dire :

- "ce wagon est vraiment répliqué"

Le vrai flux correct

Côté source

1. capture
2. dedup
3. build d'un wagon
4. appel API vers le **daemon remote**
5. le wagon passe en `sent`
6. attente du retour du remote
7. quand le remote dit "applied", alors seulement :
   - batch `acked/applied`
   - outbound `acked/applied`

Côté target

1. le daemon remote reçoit le wagon
2. il l'enregistre dans la table `Inbound`
3. il déclenche l'**orchestrator remote**
4. l'orchestrator remote applique les updates SQL
5. il met à jour `InboundChangeEvent`
6. il appelle le **daemon local** pour dire :
   - batch reçu
   - batch appliqué
   - ou batch en erreur

Donc le modèle cible devient

Serveur origine

- `OutboundChangeEvent`
- `TransportBatch`
- `SRDFOrchestrator` local
- `CommunicationDaemon` local

Serveur target

- `InboundChangeEvent`
- `SRDFRemoteOrchestrator`
- `CommunicationDaemon` remote
- applier SQL local

Donc oui, je reformule

- un Orchestrator Local
- un Orchestrator Remote
- un mini serveur de communication Local
- un mini serveur de communication Remote
- un protocole bidirectionnel
- et des statuts cohérents source/target

La cible d'architecture

Orchestrator Local

Responsable de :

1. capture
2. dedup
3. build batch
4. send wagon vers target
5. attendre retour remote
6. mettre à jour `OutboundChangeEvent`

Orchestrator Remote

Responsable de :

1. recevoir wagon
2. créer `InboundChangeEvent`
3. appliquer les updates SQL sur la DB target
4. marquer les inbound `applied` / `failed`
5. notifier le source

Daemon/API Local

Endpoints pour recevoir du remote :

- `batch_received`
- `batch_applied`
- `batch_failed`

Daemon/API Remote

Endpoints pour recevoir du source :

- `receive_batch`
- éventuellement `trigger_apply`
- `health`

Le flux complet voulu

Sens Local → Remote

1. Local orchestrator lance capture
2. Local orchestrator lance dedup
3. Local orchestrator lance build batch
4. Local sender appelle le daemon remote avec :
   - `batch_uid`
   - payload
   - checksum

Sur le target

5. Remote daemon reçoit le wagon
6. appelle `transport_receiver.py`
7. crée les `InboundChangeEvent`
8. déclenche l'orchestrateur remote
9. l'orchestrateur remote lance `transport_applier.py`
10. les inbound passent en `applied` ou `failed`

Sens Remote → Local

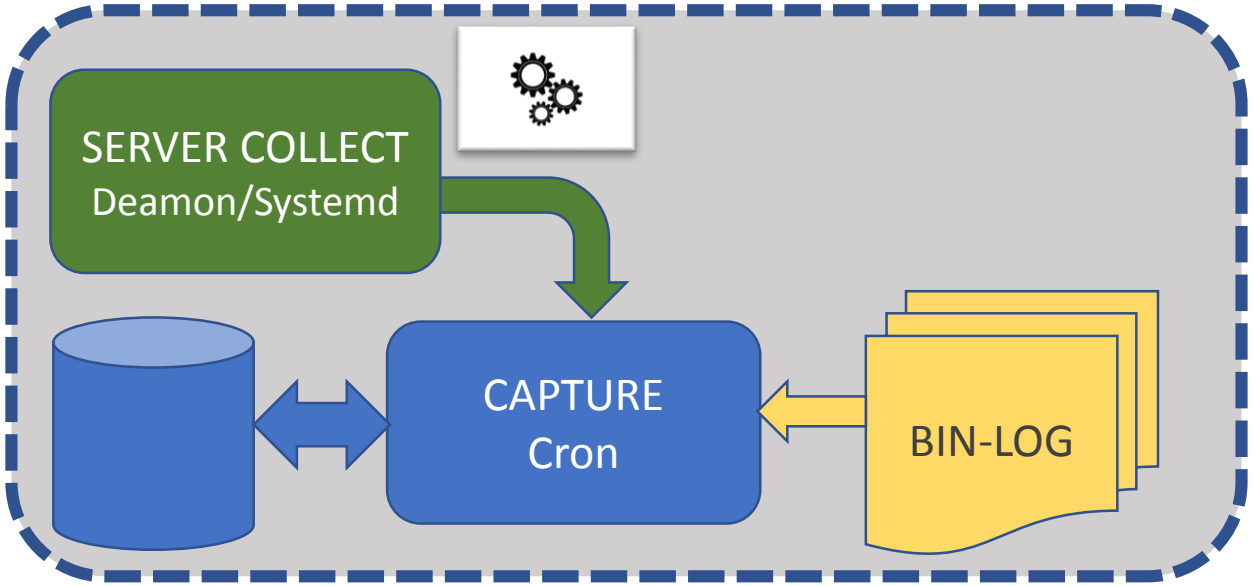
11. Remote daemon appelle le daemon local
12. local met à jour :
    - `TransportBatch`
    - `OutboundChangeEvent`
    - statuts finaux source

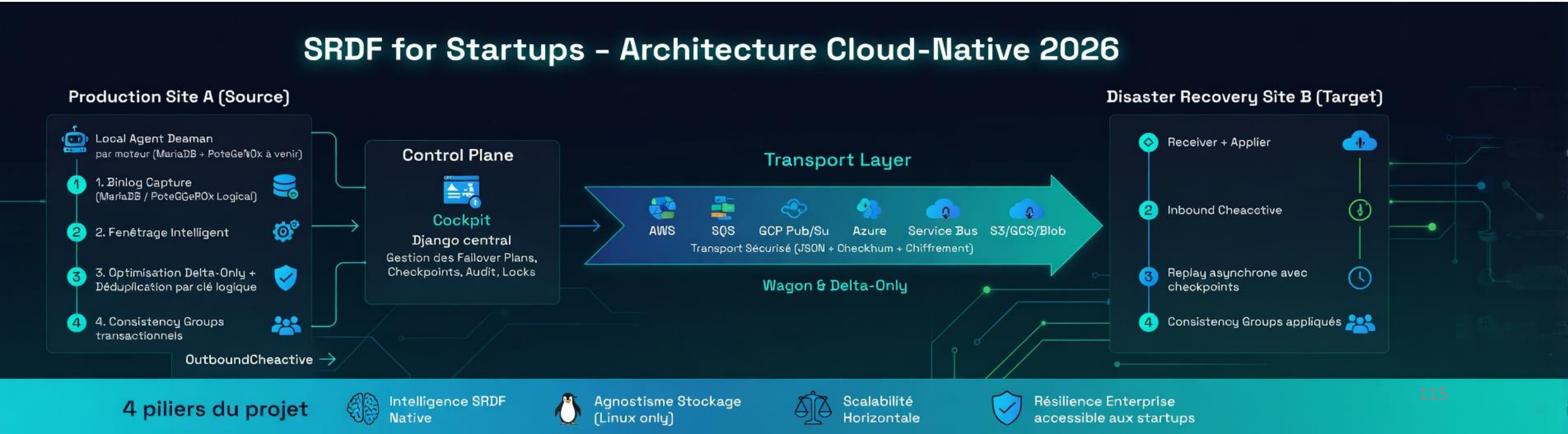
SRDF OBJECTIFS & FLOW

OBJECTIFS POURSUIVIS

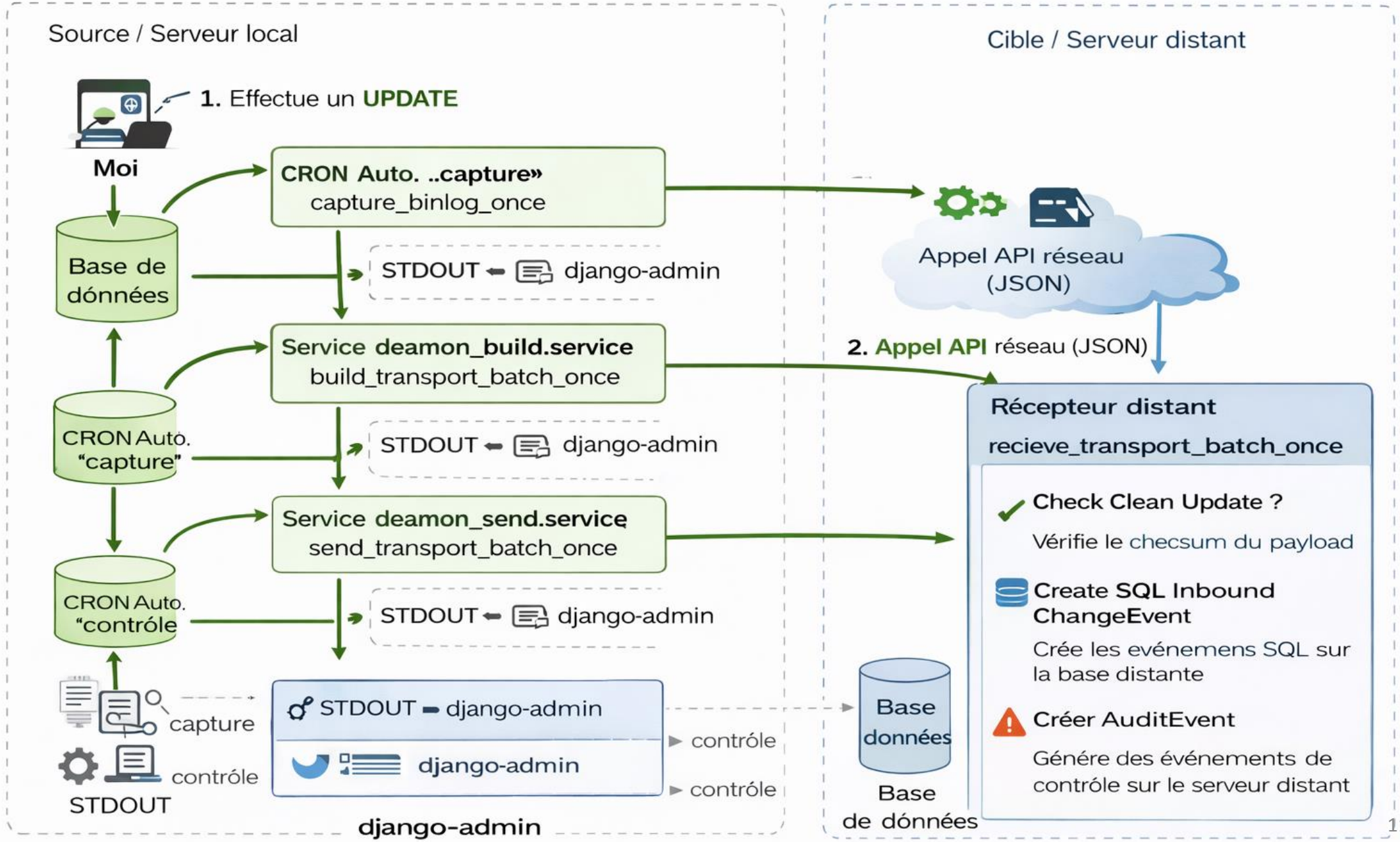
- Mise à jour « SQL » sur la Base de donnée (Origine) , *Insert, Update, Delete, Create, ..*
- Réveil du Daemon toutes les 30 secondes afin de lancer le « Batch de capture »
- Batch de capture des dernières mises à jour « non synchronisées» sur Cible
- Groupement de ces mises à jour par « paquets consistents » seuil à définir..
- De-duplication de ces Maj « On ne garde que la dernière » sur le même objet.
- Vérification si un transport est disponible (Check du serveur distant), Busy ??
- **Encryptage des données** - Compression des données
- Transport du paquet de « mises à jour » vers le serveur distant.
- Réception du paquet de mises à jour sur le serveur distant
- Contrôle des données à mettre à jour (***ne pas refaire un update déjà réalisé***)
- **Controle des erreurs** et retour à l'envoyeur !!
- Préparation du message de retour et envoie « accusé réception » vers le serveur Origine
- Le serveur origine, met à jour la table des captures « **synchro » OK....** »

SRDF : GENERAL FLOW

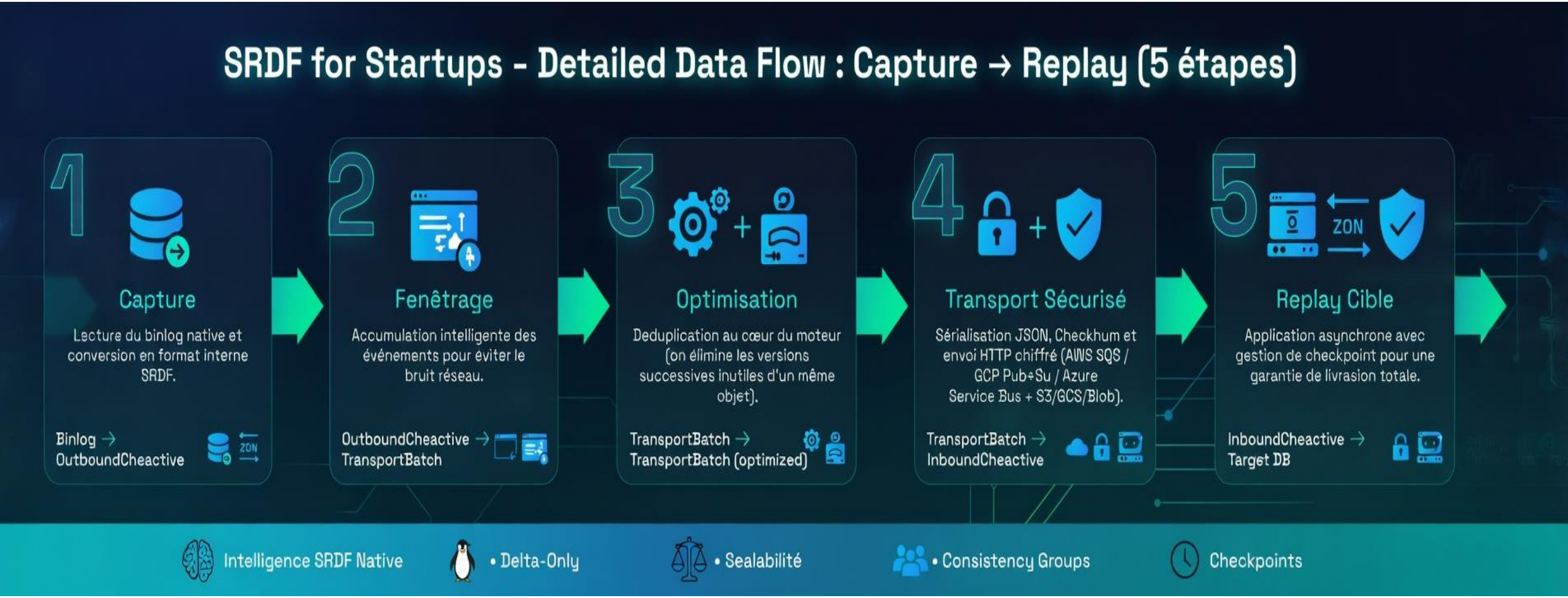




Flux complet du traitement d'un UPDATE local jusque sur le serveur distant



DETAILED DATA FLOW



BRIQUES LOGICIELLES ENVISAGEES

Niveau 1 — briques unitaires

Déjà faites :

- capture
- build batch
- send batch
- receive batch

Niveau 2 — orchestrateur permanent

Pas encore fait :

- daemon
- boucle continue
- systemd
- wakeup interval
- scheduling interne
- supervision complète

AGENT / DAEMON

- gestion des erreurs
- logs
- lock d'exécution
- backoff
- heartbeat
- config par service
- mode dry-run
- mode debug

BOUCLE INTERNE – SERVICE SYSTEMD

Sur le source

service systemd

```
srdf-source-agent.service
```

boucle interne

- wake up toutes les N secondes
- capture binlog
- construit un ou plusieurs batchs
- envoie les batchs
- loggue
- dort

Sur la cible

service systemd

```
srdf-target-agent.service
```

ou plus simplement :

- Django + API déjà exposée
- receiver appelé via `/api/action/`
- plus tard applier SQL en fond

SRDF ARCHITECTURE

COMPOSANTS

FLUX & RÉPLICATION