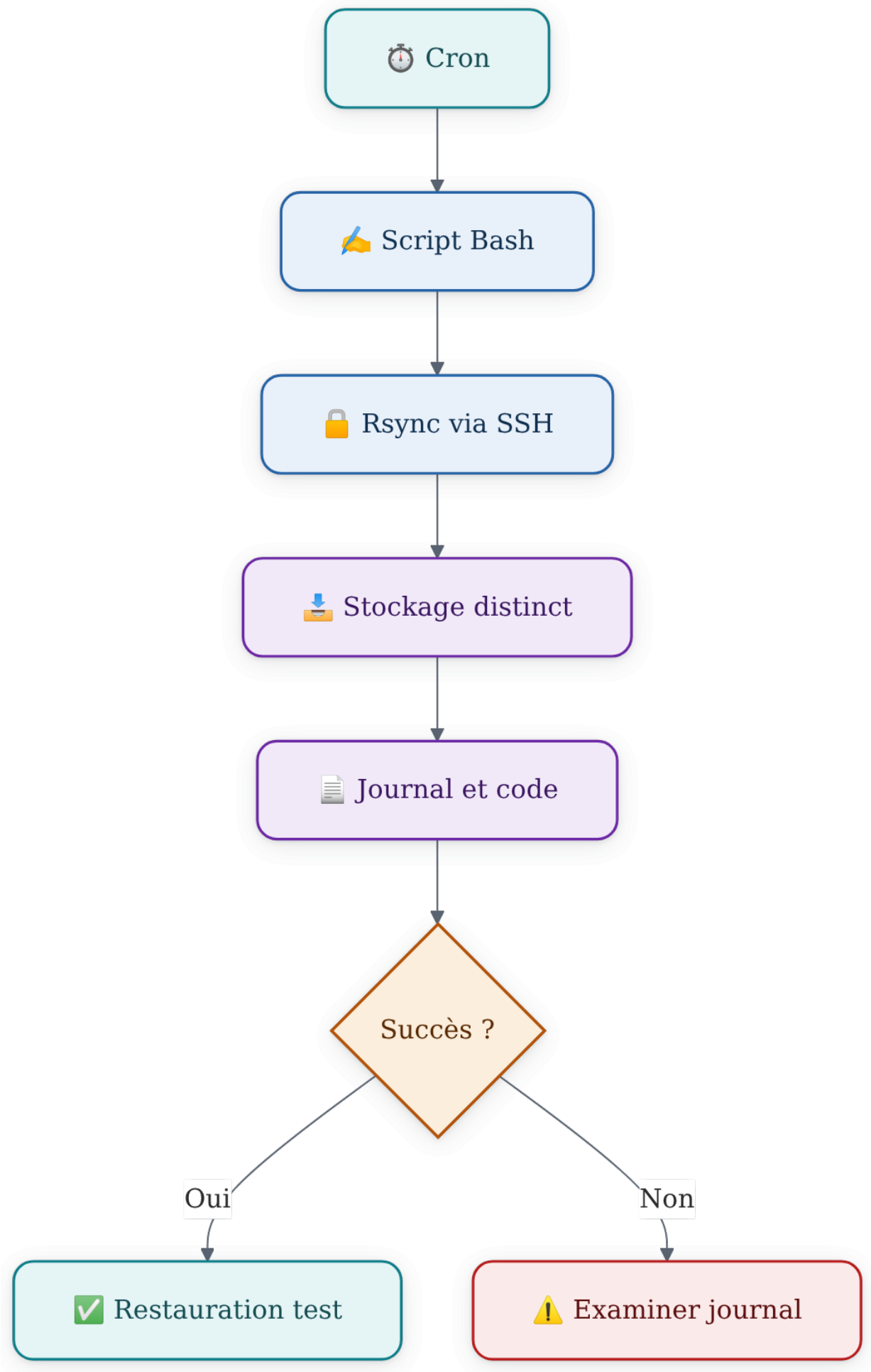
Sauvegarde rsync automatisée : la checklist pour protéger votre serveur Linux

Préparez, testez et automatisez une sauvegarde rsync sans transformer une simple synchronisation en faux sentiment de sécurité. Cochez chaque point avant de considérer votre sauvegarde comme exploitable.

Au sommaire

1. Définir ce qui doit réellement être restauré

- [ ] **Lister les répertoires applicatifs, données utilisateur et fichiers de configuration à protéger.**
  Exemples : répertoires applicatifs, /etc, documents et médias.
- [ ] **Produire des exports cohérents pour les bases de données avant de lancer rsync.**
  La copie de fichiers d'une base en cours d'écriture peut ne pas être restaurable.
- [ ] **Définir le traitement des clés et secrets selon la politique de sécurité.**
  Ne les inclure que si leur sauvegarde est autorisée et protégée.
- [ ] **Écarter les pseudo-systèmes de fichiers de la sélection.**
  Ne pas sauvegarder indistinctement /proc, /sys, /dev et /run.
- [ ] **Noter le chemin de restauration attendu pour chaque source.**
  Le slash final de la source change l'arborescence obtenue.


⏱️ Cron
✍️ Script Bash
🔒 Rsync via SSH
📥 Stockage distinct
📄 Journal et code
Succès ?
Oui
Non
✅ Restauration test
⚠️ Examiner journal

Repères rsync à valider avant l'automatisation

Option ou point	Rôle	Contrôle indispensable
-a	Copie récursivement en préservant des métadonnées importantes.	Vérifier que le compte dispose des droits nécessaires.
--dry-run ou -n	Simule les changements sans écrire sur la destination.	L'utiliser avec les chemins, exclusions et options définitifs.
--delete	Supprime dans la destination les éléments absents de la source.	Ne l'activer qu'après simulation et compréhension du comportement miroir.
--partial	Conserve les parties déjà transférées après une interruption.	À envisager pour les fichiers volumineux ou une liaison instable.
--exclude	Ignore des chemins lors de la synchronisation.	Exclure uniquement ce qui n'est pas nécessaire à la restauration.

2. Sécuriser le chemin de sauvegarde

- **Choisir une destination distincte du serveur et de son disque source.**
  Un second serveur, un volume séparé ou un stockage sur une autre machine réduit le risque de perte matérielle.
- **Créer un compte dédié à la sauvegarde sur la destination.**
  Un compte séparé limite la portée de l'accès utilisé par le script.
- **Configurer l'accès SSH par clé, sans mot de passe inscrit dans le script ou la crontab.**
  La clé publique est placée dans authorized_keys sur la machine distante.
- **Restreindre les permissions de la clé privée et protéger le serveur qui l'héberge.**
  Une exécution non interactive renforce l'importance de cette protection.
- **Vérifier que rsync, ssh, Bash et le répertoire de destination sont disponibles.**
  Contrôler aussi la présence de flock si un verrouillage est prévu.

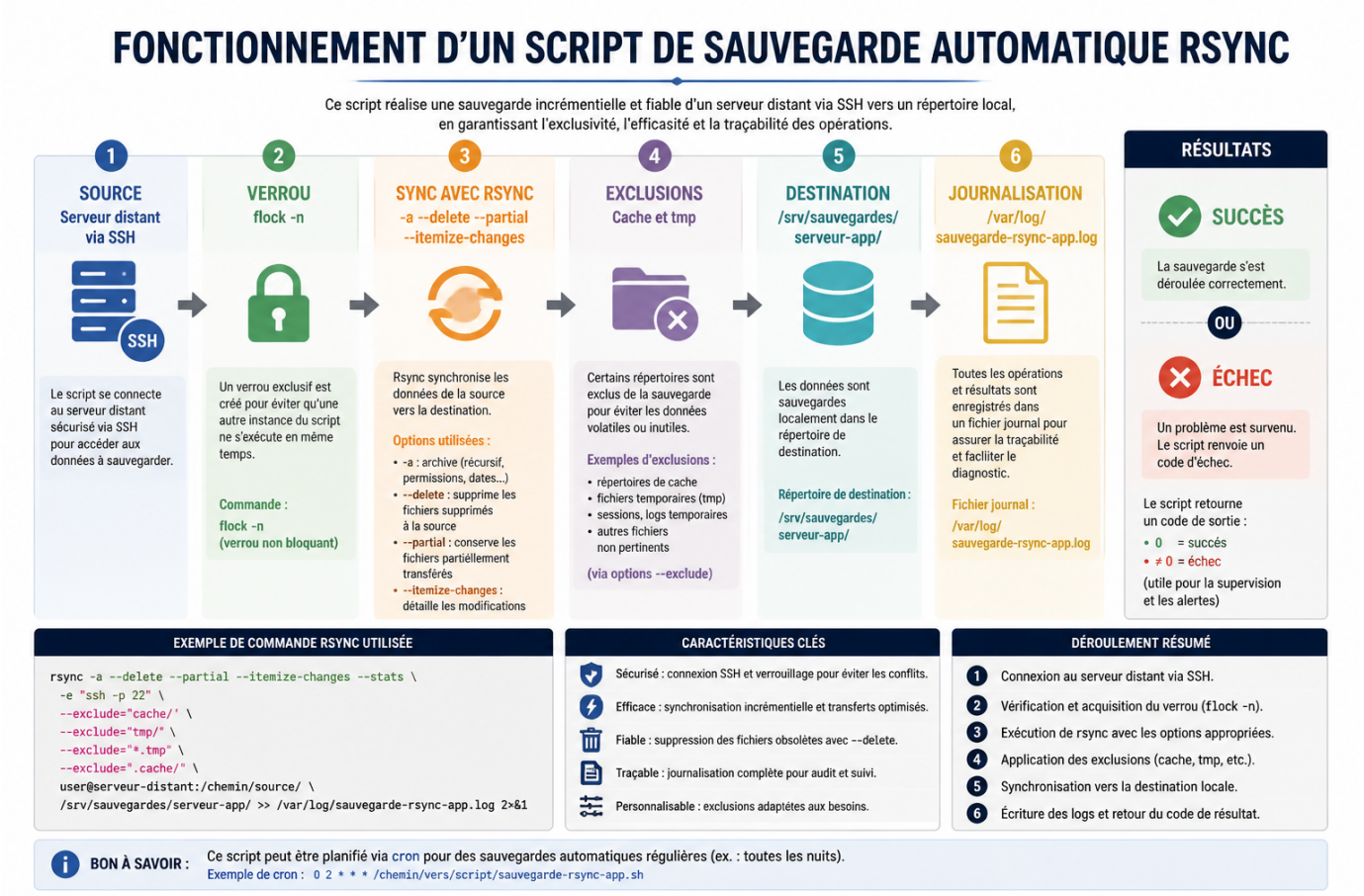
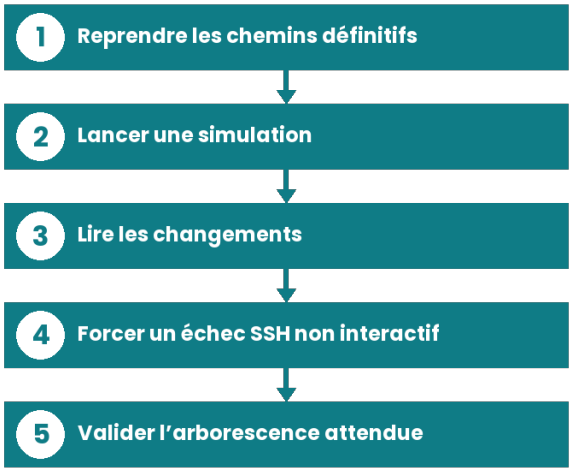


SCHÉMA *Les éléments à relier dans le script : SSH, rsync, verrou flock et journalisation.*

3. Tester la commande avant toute copie réelle



1 **Reprendre les chemins définitifs**
Testez les mêmes sources, destinations, exclusions et slash final que ceux prévus pour l'automatisation.

2 **Lancer une simulation**
Ajoutez --dry-run afin d'afficher les modifications prévues sans copier ni supprimer de fichier.

3 **Lire les changements**
Utilisez --itemize-changes pour identifier clairement les fichiers ajoutés, modifiés ou supprimés.

4 **Forcer un échec SSH non interactif**
Avec BatchMode=yes, SSH échoue au lieu d'attendre un mot de passe, ce qui convient à une tâche planifiée.

5 **Valider l'arborescence attendue**
Confirmez que la destination contient le dossier source ou seulement son contenu, selon le slash final choisi.

4. Rendre l'exécution automatique contrôlable

- [ ] **Placer la commande validée dans un script Bash exécutable.**
Le script devient le point unique à tester et maintenir.
- [ ] **Planifier le script avec cron ou un minuteur systemd.**
L'automatisation apporte de la régularité, pas une preuve de succès.
- [ ] **Ajouter un verrou avec flock pour empêcher deux exécutions simultanées.**
Évitez qu'un nouveau lancement chevauche une sauvegarde encore en cours.
- [ ] **Écrire la sortie et les erreurs dans un journal.**
Le journal doit permettre de comprendre ce qui a été transféré ou échoué.
- [ ] **Contrôler le code de retour du script.**
Un lancement planifié doit signaler un échec, pas seulement démarrer.
- [ ] **Surveiller l'espace disponible sur la destination.**
Une tâche peut être planifiée correctement tout en échouant faute d'espace.

! **Un miroir n'est pas un historique**
Avec --delete, une suppression ou un chiffrement sur la source peut être reproduit sur la destination. Rsync synchronise un état ; il ne crée pas automatiquement des versions antérieures restaurables.

5. Prouver que la sauvegarde est restaurable

- [ ] **Choisir un échantillon de fichiers et de configurations à restaurer.**
Inclure des données représentatives de vos services.
- [ ] **Restaurer vers un répertoire séparé.**
Ne pas écraser les données de production pendant le test.
- [ ] **Vérifier les chemins, contenus et métadonnées restaurés.**
Contrôler notamment ce que les droits du compte de sauvegarde permettent réellement de préserver.
- [ ] **Tester la réutilisation des exports de base de données lorsque cela s'applique.**
Une exportation doit être exploitable, pas seulement présente sur le stockage.
- [ ] **Documenter la procédure et répéter le test régulièrement.**
Une sauvegarde crédible est une sauvegarde dont la restauration a été vérifiée.

Avant toute automatisation, simulez les changements avec --dry-run. Une sauvegarde rsync doit être complétée par des restaurations de test.