LIVRE BLANC

Checklist de reprise d'un projet digital

Changement de prestataire • Reprise d'existant • Projets à risque

Guide opérationnel — Lecture 10 minutes

Ce document contient

- 6 sections opérationnelles structurées
- 3 tableaux de vérification complète des accès
- 1 matrice des risques fréquents
- 1 plan de stabilisation en 4 phases
- 1 checklist finale synthétique de 30 points

Version 1.0 — 2026

Contexte et objectifs

Ce livre blanc s'adresse aux chefs de projet, directeurs techniques et responsables digitaux confrontés à une situation de reprise ou de transition sur un projet existant. Il fournit une méthode structurée pour évaluer, sécuriser et stabiliser un projet avant d'engager de nouvelles évolutions.

> Objectif principal : éviter les situations de blocage, perte de données ou interruption de service lors d'un changement de prestataire, d'une reprise d'existant ou de la gestion d'un projet fragilisé.

01 Quand utiliser cette checklist

Identifier les situations de reprise

Cette checklist s'applique dans quatre situations typiques. Chaque situation présente des risques spécifiques mais nécessite une même rigueur d'inventaire initial.

Situation	Signes déclencheurs	Urgence
Changement d'agence / prestataire	Fin de contrat, rupture de relation, appel d'offres remporté	Haute
Départ d'un développeur clé	Démission, licenciement, absence prolongée	Critique
Projet instable ou en production dégradée	Erreurs fréquentes, indisponibilités, tickets non traités	Critique
Absence de documentation	Onboarding difficile, code non commenté, pas de wiki	Moyenne
Rachat ou fusion d'activité	Intégration d'un SI tiers, consolidation de plateformes	Haute

> ⚠ Signal d'alerte fort : si deux de ces situations se cumulent, la reprise doit être traitée en mode gestion de crise avec gel immédiat des évolutions.

02 Les accès à récupérer

Inventaire complet des droits et identifiants

La première étape d'une reprise est l'obtention de tous les accès nécessaires à la gestion autonome du projet. Un accès manquant peut bloquer une intervention critique en production.

2.1 — Inventaire des accès critiques

Ressource	Ce qu'il faut obtenir	Format idéal	Criticité
Code source	URL du dépôt Git, identifiants, branches actives	Git (GitHub / GitLab / Bitbucket)	🔴 Critique
Hébergement	Accès panel (OVH, AWS, etc.), FTP/SFTP, SSH	Credentials + documentation IP	🔴 Critique
Serveur / VPS	Clé SSH ou mot de passe root, liste des services actifs	Fichier .pem ou credentials	🔴 Critique
Base de données	Host, port, user, password, nom de la BDD	Fichier .env ou credentials séparés	🔴 Critique
DNS	Accès registrar, zones DNS, TTL configurés	Panel registrar (OVH, Gandi...)	🔴 Critique
Emails transactionnels	SMTP ou accès Mailgun / Mailchimp / SendGrid	API Key + domaine vérifié	🟠 Élevé
Comptes tiers (Stripe)	Clés API live et test, webhooks configurés	Dashboard + API keys	🟠 Élevé
Google Analytics / Tag Manager	Accès admin au compte et à la propriété	Email administrateur	🟡 Moyen
CDN / Cloudflare	Accès compte, zones configurées, règles actives	Email + 2FA	🟠 Élevé
Certificat SSL	Expiration, renouvellement automatique ou manuel	Let's Encrypt ou certificat tiers	🟠 Élevé

2.2 — Matrice de transfert

Utiliser ce tableau pour tracer l'état de réception de chaque accès lors du transfert formel.

Ressource	Responsable côté sortant	Date demande	Date réception	Vérifié ?
Code source (Git)				☐
Hébergement / Panel				☐
SSH / Accès serveur				☐
Base de données				☐
DNS / Registrar				☐
Emails (SMTP / API)				☐

Ressource	Responsable côté sortant	Date demande	Date réception	Vérifié ?
Stripe / Paiement				☐
Outils analytics				☐
SSL / CDN				☐

03 Vérifications techniques
Audit de l'environnement et des processus

Une fois les accès obtenus, procéder à l'audit technique de l'existant. L'objectif est d'établir une photographie précise de l'état du projet avant toute intervention.

3.1 — Environnements

Point de contrôle	Vérification	État	Note
Environnement de développement	Existe et est fonctionnel (accès, URL, données de test)	☐ OK ☐ KO	
Environnement de staging / recette	Distinct de la production, données anonymisées	☐ OK ☐ KO	
Environnement de production	URL, serveur, version applicative documentée	☐ OK ☐ KO	
Variables d'environnement (.env)	Fichier .env documenté, séparé par environnement	☐ OK ☐ KO	
Procédure de déploiement	Documentée ? Automatisée ? CI/CD en place ?	☐ OK ☐ KO	
Rollback possible ?	Procédure de retour arrière définie et testée	☐ OK ☐ KO	

3.2 — Stack technique et dépendances

Composant	À vérifier	Version actuelle	Statut
PHP / Node.js / Python	Version serveur vs version requise par l'application		☐
Gestionnaire de dépendances	composer.json, package.json présents et à jour		☐
Framework principal	Version, compatibilité, fin de support		☐
CMS (WordPress, Drupal...)	Version core, thème actif, plugins actifs		☐

Composant	À vérifier	Version actuelle	Statut
Plugins / modules tiers	Versions, vulnérabilités connues (WPScan...)		☐
Base de données	Version MySQL/PostgreSQL, charset, collation		☐
Serveur web	Apache/Nginx — version, modules actifs, .htaccess		☐
Certificat SSL	Date d'expiration, renouvellement auto ?		☐

3.3 — Sauvegardes

> Règle d'or : ne jamais travailler sur un projet sans avoir vérifié et testé une sauvegarde complète. En l'absence de sauvegarde avérée, la première action est d'en créer une manuellement avant toute intervention.

Point de contrôle	Question clé	Réponse
Sauvegarde base de données	Existe ? Fréquence ? Rétention ?	
Sauvegarde fichiers / médias	Inclut-elle les uploads et assets ?	
Test de restauration	La dernière sauvegarde a-t-elle été testée ?	
Stockage des sauvegardes	Local uniquement ou off-site (S3, distant...) ?	
Accès aux sauvegardes	Qui y a accès ? Procédure documentée ?	
Sauvegarde code source	Dépôt Git à jour avec la production ?	

3.4 — Observabilité et logs

Outil / Ressource	Présent ?	Accessible ?	Action requise
Logs applicatifs (erreurs)	☐ Oui ☐ Non	☐ Oui ☐ Non	
Logs serveur (Apache/Nginx)	☐ Oui ☐ Non	☐ Oui ☐ Non	
Monitoring uptime (Pingdom...)	☐ Oui ☐ Non	☐ Oui ☐ Non	
Alertes email/SMS production	☐ Oui ☐ Non	☐ Oui ☐ Non	
Outil de tracking erreurs (Sentry)	☐ Oui ☐ Non	☐ Oui ☐ Non	

04 Les risques fréquents
Matrice d'impact et de probabilité

Ces risques sont rencontrés dans la grande majorité des reprises de projets. Leur identification précoce permet d'adapter le plan d'action et de définir les priorités de stabilisation.

Risque	Description	Impact	Probabilité
Dépendance personne-clé	Tout le savoir est dans la tête d'une seule personne. Si elle part, le projet devient opaque.	Critique	Élevée
Absence de sauvegarde	Aucune sauvegarde fonctionnelle et testée. Toute intervention peut être irréversible.	Critique	Élevée
Pas de versioning (Git)	Le code est déployé manuellement. Aucun historique, aucune traçabilité des modifications.	Critique	Élevée
Dette technique critique	Code non maintenable, framework obsolète, couplage fort rendant les évolutions dangereuses.	Élevé	Élevée
Plugins/librairies obsolètes	Versions vulnérables exposant le projet à des failles de sécurité (XSS, injection SQL...).	Élevé	Élevée
Absence de documentation	Procédures non documentées, architecture inconnue, onboarding impossible sans accompagnement.	Élevé	Élevée
Variables d'env en dur dans le code	Clés API, mots de passe ou tokens committés en clair dans le dépôt Git.	Critique	Moyenne
Environnements confondus	Modifications directes en production, absence de staging, risque de régression sans filet.	Élevé	Moyenne
Contrats/licences non transférés	Licences de plugins premium ou de services tiers liées à l'ancien prestataire.	Élevé	Moyenne
Infrastructure non documentée	Serveurs, IP, ports, services actifs inconnus rendant toute intervention risquée.	Élevé	Élevée

Priorité absolue : traiter en premier les risques de criticité 'Critique' combinés à une probabilité 'Élevée'. Ces combinaisons représentent un danger immédiat pour la continuité de service.

05 Sécuriser la reprise
Plan de stabilisation en 4 phases

La sécurisation d'une reprise suit une séquence précise. Chaque phase est un prérequis à la suivante. Il est contre-productif de démarrer des évolutions fonctionnelles avant d'avoir stabilisé l'existant.

Phase 1 — Audit rapide (J+1 à J+5)

Objectif : établir un état des lieux factuel sans intervenir. Durée recommandée : 2 à 5 jours selon la complexité du projet.

- Collecter tous les accès listés en section 2
- Cartographier l'architecture technique (serveurs, services, dépendances)
- Identifier les points de défaillance critiques (single points of failure)
- Vérifier l'existence et la validité des sauvegardes
- Analyser les logs récents à la recherche d'erreurs récurrentes
- Documenter la situation initiale dans un rapport d'audit

Phase 2 — Gel des évolutions (J+1 à J+30)

Objectif : ne pas ajouter de complexité à une situation déjà fragilisée. Cette phase est souvent difficile à vendre aux équipes métier mais elle est non négociable.

- Communiquer clairement sur le gel à toutes les parties prenantes
- Mettre en place une procédure de dérogation pour les urgences uniquement
- Continuer les corrections de bugs et de sécurité uniquement
- Proscrire tout déploiement en production sans validation préalable

Phase 3 — Plan de stabilisation (J+5 à J+60)

Objectif : réduire les risques identifiés par ordre de criticité. Chaque action doit être documentée et validée avant passage en production.

Priorité	Action	Délai cible	Validation
🔴 P0	Mettre en place ou vérifier les sauvegardes automatiques	48h	Testée + confirmée
🔴 P0	Sécuriser tous les accès (rotation de mots de passe)	48h	Anciens accès révoqués

Priorité	Action	Délai cible	Validation
🔴 P0	Mettre le code sous Git si absent	5 jours	Dépôt distant créé
🟠 P1	Créer un environnement de staging	2 semaines	Validé fonctionnel
🟠 P1	Documenter l'architecture et les procédures clés	2 semaines	Wiki accessible à l'équipe
🟠 P1	Mettre à jour les plugins/dépendances critiques	2 semaines	Testés sur staging
🟡 P2	Mettre en place un monitoring / alertes	1 mois	Alertes configurées
🟡 P2	Déplacer les variables sensibles hors du code	1 mois	Vérification Git + .gitignore
🟢 P3	Rédiger la documentation technique complète	2 mois	Revue et validée

Phase 4 — Documentation minimale

Objectif : garantir qu'une troisième personne peut reprendre le projet sans aide. La documentation n'est pas une option : c'est une condition de sortie de la phase de stabilisation.

Document	Contenu minimal	Format recommandé
README.md	Installation locale, prérequis, commandes essentielles	Markdown dans le dépôt Git
Architecture technique	Schéma serveurs, services, flux de données	Draw.io ou Confluence
Procédure de déploiement	Étapes précises, environnements, rollback	Markdown ou Notion
Gestion des accès	Liste des comptes, responsables, procédure de rotation	Gestionnaire de mots de passe
Contacts & escalade	Prestataires hébergement, registrar, outils tiers	Document partagé équipe

06 Checklist finale de reprise

30 points de contrôle — à valider avant tout démarrage

Cette checklist de synthèse récapitule les 30 points essentiels à valider lors de toute reprise de projet. Elle est conçue pour être utilisée comme document de suivi partagé entre les équipes entrante et sortante.

BLOC A — Accès et credentials

#	Action / Vérification	Priorité	Statut
A1	Accès au dépôt Git (code source complet, toutes branches)	Critique	☐
A2	Accès SSH / panel hébergement avec droits admin	Critique	☐
A3	Credentials base de données (host, user, password, BDD)	Critique	☐
A4	Accès registrar DNS avec capacité de modification	Critique	☐
A5	Clés API des services tiers (Stripe, Mailgun, etc.)	Élevé	☐
A6	Accès aux outils analytics (Google Analytics, Tag Manager)	Moyen	☐

BLOC B — Infrastructure et environnements

#	Action / Vérification	Priorité	Statut
B1	Architecture serveur documentée (IP, services actifs, ports)	Critique	☐
B2	Variables d'environnement (.env) complètes et séparées par env.	Critique	☐
B3	Environnement de staging distinct et fonctionnel	Élevé	☐
B4	Procédure de déploiement documentée étape par étape	Élevé	☐
B5	Procédure de rollback définie et testée	Élevé	☐
B6	Certificat SSL valide + renouvellement automatique configuré	Élevé	☐

BLOC C — Sauvegardes

#	Action / Vérification	Priorité	Statut
C1	Sauvegarde complète de la base de données créée et vérifiée	Critique	☐
C2	Sauvegarde des fichiers / médias créée et vérifiée	Critique	☐

#	Action / Vérification	Priorité	Statut
C3	Test de restauration effectué avec succès	Critique	☐
C4	Sauvegarde stockée off-site (S3, stockage distant)	Élevé	☐
C5	Sauvegardes automatiques configurées (fréquence + rétention)	Élevé	☐

BLOC D — Stack technique

#	Action / Vérification	Priorité	Statut
D1	Versions PHP / Node.js / Python identifiées et documentées	Élevé	☐
D2	Dépendances (composer.json / package.json) à jour dans Git	Élevé	☐
D3	Plugins et librairies tiers inventoriés avec versions	Élevé	☐
D4	Vulnérabilités critiques identifiées (WPScan, npm audit...)	Critique	☐
D5	Logs applicatifs accessibles et analysés	Moyen	☐

BLOC E — Gouvernance et documentation

#	Action / Vérification	Priorité	Statut
E1	Anciens accès de l'équipe sortante révoqués / modifiés	Critique	☐
E2	Gel des évolutions fonctionnelles communiqué et acté	Élevé	☐
E3	README.md à jour dans le dépôt Git	Élevé	☐
E4	Architecture documentée (schéma + texte)	Élevé	☐
E5	Contacts prestataires tiers identifiés	Moyen	☐
E6	Rapport d'audit initial rédigé et partagé	Élevé	☐
E7	Planning de stabilisation (phases P0 → P3) validé	Élevé	☐
E8	Propriété intellectuelle du code vérifiée (contrats)	Moyen	☐

> Seuil de démarrage recommandé : tous les points 'Critique' (rouge) doivent être à statut OK avant de démarrer le moindre développement. Les points 'Élevé' doivent être couverts dans les 30 premiers jours.

Synthèse et points clés

 **Accès complets** Condition #1 avant tout démarrage	 **Sauvegardes testées** Filet de sécurité indispensable	 **Documentation minimale** Condition de sortie de stabilisation

La reprise d'un projet digital est un exercice de gestion du risque avant d'être un exercice technique. Suivre cette méthodologie permet de transformer une situation potentiellement chaotique en processus maîtrisé, avec des étapes claires et des critères de validation objectifs.

La règle fondamentale reste : ne jamais ajouter de complexité à un système que l'on ne maîtrise pas encore pleinement. Audit, stabilisation, documentation — dans cet ordre, sans exception.