Super idée de faire un mémo pour demain. Voilà un **résumé clair et actionnable** de tout ce qu’on a mis en place + la suite à faire pour la gestion des versions GitHub et les points d’attention côté Google Sheets.

**Résumé de la session (sauvegarde totale GAS + CSV)**

**Ce qui fonctionne maintenant**

* **Pipeline complet snapshot\_rk.ps1** (Tools\snapshot\_rk.ps1) :
  1. clasp pull sur 4 projets (01\_Moteur, 02\_configuration, 03\_BaseDeDonnees/03\_BaseDeDonnées, 04\_Templates)
  2. **Concat** des scripts GAS par projet → scripts\_\*.txt
  3. **Export CSV** des 4 Google Sheets par IDs → sous export-onglets-csv/SNAPSHOT\_\*
  4. **Génération auto** : manifest.json, brief.md, diff.md (via snapshot\_helpers.ps1)
  5. **Zip** du snapshot  
     ✅ Dernier snapshot OK : SNAPSHOT\_20250828\_210722 (manifest/brief/diff/zip produits)
* **Helpers** (Tools\snapshot\_helpers.ps1) :
  1. Write-Manifest, Write-BriefMd, Write-DiffMd (ASCII, compatibles PowerShell 5.1).
  2. Test manuel réussi (rechargé via . .\Tools\snapshot\_helpers.ps1).
* **Export CSV simple** (Tools\export\_csv\_only.ps1) :
  1. Permet de relancer uniquement l’export CSV (avec retries et réauth Google si besoin).
* **Nettoyage caractères invisibles** (Tools\fix\_nbsp.ps1) :
  1. Retire NBSP/zero-width/backticks parasites si le parseur PowerShell râle.
* **Git & sécurité**
  1. .gitignore : ignore export-onglets-csv/ et **export-onglets-csv/token.json**.
  2. **Purge historique du token** : git-filter-repo exécuté → token supprimé des commits passés, remote réajouté, push OK.
  3. .gitattributes : normalisation des fins de lignes (limite les warnings CRLF/LF).
  4. VS Code : .vscode/settings.json pour **UTF-8 (BOM) + CRLF sur .ps1**, Shell intégré = **Windows PowerShell** (5.1).
* **Google OAuth**
  1. Token révoqué côté Google et régénéré **localement seulement** (et ignoré par Git) lors du premier export (flow “copie/colle code d’autorisation”).

**Commandes utiles (rappel)**

* **Snapshot complet (fin de session)**
* powershell -NoProfile -ExecutionPolicy Bypass -File .\Tools\snapshot\_rk.ps1
* **Export CSV uniquement**
* powershell -NoProfile -ExecutionPolicy Bypass -File .\Tools\export\_csv\_only.ps1
* **Helpers (manifest/brief/diff) sur le dernier snapshot**
* . .\Tools\snapshot\_helpers.ps1
* $repo = Get-Location
* $snap = Get-ChildItem -LiteralPath (Join-Path $repo 'export-onglets-csv') -Directory |
* Sort-Object LastWriteTime -Descending | Select-Object -First 1
* $man = Write-Manifest -SnapshotDir $snap.FullName -RepoRoot $repo
* Write-BriefMd -SnapshotDir $snap.FullName -Manifest $man | Out-Null
* $prev = Get-ChildItem -LiteralPath (Join-Path $repo 'export-onglets-csv') -Directory |
* Where-Object { $\_.FullName -ne $snap.FullName } |
* Sort-Object LastWriteTime -Descending | Select-Object -First 1
* if ($prev) {
* $pm = Join-Path $prev.FullName 'manifest.json'
* if (Test-Path $pm) {
* Write-DiffMd -PrevManifestPath $pm `
* -CurrManifestPath (Join-Path $snap.FullName 'manifest.json') `
* -OutPath (Join-Path $snap.FullName 'diff.md') | Out-Null
* }
* }
* **Si parse error PowerShell**
* powershell -NoProfile -ExecutionPolicy Bypass -File .\Tools\fix\_nbsp.ps1

**Demain — plan pour la gestion de versions GitHub (code + “rollbacks”)**

**Objectif**

Pouvoir garder **plusieurs versions** de l’appli (scripts GAS + config) et **remettre en place** une version antérieure quand nécessaire.

**Recommandation de workflow Git**

1. **Branches**
   * main = prod stable
   * dev = intégration/expériences
2. **Tags**
   * Taguer chaque état déployé GAS : vYYYY.MM.DD-HHMM (ex : v2025.08.28-2107)
   * Ajouter un **annotated tag** avec message (lien vers SNAPSHOT\_\*).
3. **Mapping snapshot ↔ tag**
   * À la fin d’un snapshot + déploiement, créer un tag Git du code **exact** utilisé.

**Rituels (blocs prêts à coller)**

* **Créer un tag à partir de HEAD (état prêt à pousser vers GAS)**
* $tag = "v$(Get-Date -Format 'yyyy.MM.dd-HHmm')"
* git tag -a $tag -m "Release $tag (voir export-onglets-csv\SNAPSHOT\_...)"
* git push origin $tag
* **Lister les versions**
* git tag --sort=-creatordate
* **Basculer vers une version (checkout lecture)**
* git checkout tags/v2025.08.28-2107
* **Créer une branche à partir d’une version**
* git switch -c hotfix/from-v2025.08.28-2107 v2025.08.28-2107

**Déployer une ancienne version vers Google Apps Script**

⚠️ On parle ici **du code** GAS. Les CSV/Google Sheets sont indépendants (voir “Incidences” plus bas).

1. **Revenir au commit/tag voulu** (en détaché ou via une branche)
2. git checkout tags/v2025.08.28-2107
3. **Pousser vers chaque projet** (dans chacun des sous-dossiers clasp)
4. # Exemple pour 01\_Moteur
5. cd .\01\_Moteur
6. clasp push
7. # Refaire pour 02\_configuration, 03\_BaseDeDonnees (ou 03\_BaseDeDonnées), 04\_Templates
8. **Vérifier/rafraîchir les triggers/autorisation** si le code a changé des scopes (GAS peut redemander des consentements au premier run).

Bonus (option à mettre en place) : un **script PowerShell “rollback\_gas.ps1”** qui itère automatiquement sur les 4 dossiers et fait les clasp push. Dispo si tu veux.

**Incidences quand on remet une ancienne version**

**1) Scripts Google Apps Script**

* **Triggers** : si la version antérieure n’a pas les mêmes triggers, il faudra les **recréer** (ou les scripts “GestionTriggers.js” doivent le faire).
* **Scopes/autorisation** : si la version antérieure demande moins/plus de scopes, GAS peut **redemander l’autorisation** au premier run.
* **Déploiements Web/Executions** : si tu utilises des **versions de déploiement** (Apps Script “déployer”), il faut republier ou réassigner le déploiement vers la version.

**2) Données Google Sheets**

* **Les autorisations d’accès (partage Drive/Sheets) ne changent pas** tant que tu **ne remplaces pas l’ID du fichier**. Écraser le contenu ou réimporter des CSV **ne modifie pas** les partages existants.
* **En revanche** :
  + Si tu **supprimes et recrées** un Spreadsheet (nouvel ID), tu perds les autorisations : il faudra **repartager**.
  + Restaurer des **données** (réimport CSV) peut casser des **références** (formules vers des onglets renommés/supprimés) ou des **nommages** (Named ranges) si les onglets diffèrent.
  + Un **importeur CSV** (TODO) doit écraser proprement chaque onglet cible, sans toucher aux paramètres/partages.

**Rappel à me faire demain** :  
“Quand on remet une ancienne version, vérifier s’il y a une incidence sur les fichiers Google Sheets et leurs autorisations.”  
→ Réponse courte : *Non, pas d’impact sur les autorisations tant que l’ID reste le même ; oui, attention aux triggers/scopes côté Apps Script et à la cohérence des onglets si on restaure des CSV.*

**À faire demain (proposition)**

1. **Mettre en place les tags release** (blocs prêts à coller) + un petit script Tools\tag\_release.ps1.
2. **Créer Tools\rollback\_gas.ps1** : sélection d’un **tag Git**, git checkout, puis clasp push sur les 4 projets.
3. **Spécifier l’importeur CSV “restore”** (optionnel) : script qui prend un dossier SNAPSHOT\_\* et écrase les onglets correspondants (avec un --dry-run).

Si tu veux, je te fournis directement les scripts tag\_release.ps1 et rollback\_gas.ps1 au prochain message.

Haut du formulaire

Bas du formulaire

ChatGPT peut commettre des erreurs. Il est re