Super, on pose les bases clairement. Voici **la logique Git/GitHub** et **comment l’appliquer à ton projet (GAS + CSV Google Sheets)**, avec des pas-à-pas VS Code et les réponses directes à tes questions.

**1) Git / GitHub en 2 minutes**

* **Git (local)** : ta “base de vérité” sur ton PC.
  + **Working directory** : tes fichiers tels qu’ils sont sur disque.
  + **Index (staging)** : la “sélection” de ce que tu veux figer.
  + **Commit** : un instantané (avec message) dans l’**historique**.
* **Branches** : des “lignes de temps” parallèles (ex. main, dev).
* **Tags** : des étiquettes immuables (ex. v2025.08.28-2107), parfaits pour marquer des versions déployées.
* **GitHub (remote)** : la **copie distante** du dépôt, pour partager, sauvegarder et retrouver l’historique partout.

**2) Architecture recommandée pour ton repo**

/ (racine du repo)

├─ 01\_Moteur/ # projet GAS (clasp)

├─ 02\_Configuration/ # projet GAS

├─ 03\_BaseDeDonnees/ # projet GAS

├─ 04\_Templates/ # projet GAS

├─ export-onglets-csv/

│ ├─ SNAPSHOT\_YYYYMMDD\_HHMMSS/

│ │ ├─ \*.csv # exports d’onglets (données)

│ │ ├─ manifest.json # inventaire du snapshot

│ │ ├─ brief.md # résumé

│ │ └─ diff.md # différences avec snapshot précédent

│ └─ ...

├─ Tools/ # scripts PowerShell / bash (snapshot, rollback…)

├─ .gitignore

└─ README.md

👉 Ton **code GAS** est versionné (via clasp pull → fichiers .gs/.js.json).  
👉 Tes **données** sont versionnées via **CSV exportés** par snapshot (un dossier par date).

**3) Sauvegarder (backup) avec VS Code**

**But :** figer l’état actuel (code + CSV exportés) et l’envoyer sur GitHub.  
**Finalité :** pouvoir restaurer *exactement* cet état plus tard.

**Une fois pour toutes (si pas encore fait)**

1. Ouvre le dossier racine dans VS Code → **Source Control** (Ctrl+Shift+G) → **Initialize Repository**.
2. Connecte GitHub (bouton **Publish to GitHub** ou Command Palette “Git: Publish to GitHub”).
3. Crée .gitignore (exemple plus bas).

**À chaque snapshot**

1. Exécute ton script de snapshot (il régénère CSV + manifest.json + brief.md + diff.md).
2. VS Code → onglet **Source Control** :
   * **Stage Changes** (le + à côté de chaque fichier ou “Stage All”).
   * **Commit** avec un message clair (ex. feat(snapshot): SNAPSHOT\_20250828\_210722).
   * **Push** (icône “Sync” ⇅).
3. (Optionnel recommandé) **Tag** la version :
   * Command Palette → “**Git: Create Tag**” → v2025.08.28-2107, puis **Push Tags**.

**Équivalent en lignes de commande**

git add .

git commit -m "feat(snapshot): SNAPSHOT\_20250828\_210722"

git push

git tag v2025.08.28-2107

git push --tags

**4) Restaurer (rollback) proprement**

Il y a **deux plans** : (A) restaurer **le code GAS** ; (B) restaurer **les données (CSV)**.

**A) Restaurer le code GAS**

**But :** remettre l’ancienne logique côté scripts Apps Script.  
**Finalité :** revenir à un état stable identifié.

1. **Choisir le point de restauration** :
   * VS Code → **Source Control** → **View & More actions** → “**Git: Checkout to…**” → choisis le **tag** ou le **commit**.
   * (CLI) git checkout v2025.08.28-2107
2. **Pousser vers GAS** (dans chaque sous-dossier 01\_..., 02\_..., etc.) :

clasp push

1. (Selon ton setup) **Re-déployer** le projet (bouton “Deploy” dans Apps Script) ; parfois **reconsentement** des scopes si des autorisations ont changé.

**Incidences possibles**

* **Triggers/Scopes** : il peut falloir **réautoriser**.
* **Compatibilité schéma** : si l’ancien code suppose d’anciens **noms d’onglets/colonnes**, et que les Sheets actuels ont changé, il peut **planter** (voir §6).

**B) Restaurer les données (CSV)**

**But :** rétablir le contenu des onglets à une date donnée.  
**Finalité :** avoir des données cohérentes avec la version de code restaurée.

1. Prends le dossier export-onglets-csv/SNAPSHOT\_... visé.
2. **Dry-run** conseillé (script de restauration en mode simulation) pour détecter : onglets manquants, colonnes en plus/en moins.
3. **Import effectif** : réimporte chaque CSV dans l’onglet correspondant (en écrasant le contenu).
   * ⚠️ Le CSV **ne contient pas** de formats, couleurs, validations, formules → juste **les valeurs** (la première ligne servant d’en-têtes).

**Incidences possibles**

* Toute **mise en forme** disparaît si tu recrées un onglet à partir d’un CSV.
* Les **formules** sont “évaluées” dans le CSV (valeurs), pas les expressions.
* Si tu as changé la **structure des onglets** entre-temps, adapte soit le **code** soit le **CSV** (ou reviens à une **copie Drive** du fichier si tu veux structure + mise en forme).

**5) Réponses directes à tes questions**

* **“Comment on fait la sauvegarde ?”**  
  → Par **commit** + **push** sur GitHub (depuis VS Code). Dans ton cas, lance d’abord ton **snapshot** (qui exporte CSV + manifest + diff), puis commit/push. **Ajoute un tag** pour figer la version.
* **“Comment on fait la restauration ?”**  
  → **Code** : git checkout <tag> puis clasp push (et redeploy).  
  → **Données** : réimporte les **CSV** du snapshot choisi (avec un **dry-run** si possible).
* **“Incidences de la restauration ?”**  
  → **GAS** : possibles **redemandes d’autorisations**, re-déploiement, et attention à la **compatibilité avec le schéma** des Sheets.  
  → **Google Sheets** : tant que l’**ID du fichier** ne change pas, les **partages/permissions** restent ; mais les **scripts** restaurés doivent être **compatibles** avec la structure réelle des onglets.
* **“GitHub sauvegarde-t-il le colonnage des onglets Google Sheets ?”**  
  → GitHub **sauvegarde des fichiers**. Si tu **exportes** tes onglets en **CSV**, il sauvegarde bien **l’ordre et les noms des colonnes** (ligne d’en-têtes) **et les valeurs**.  
  → En revanche, il **ne sauvegarde pas** : formats, couleurs, validations, filtres, **formules** (elles deviennent des **valeurs**), vues filtrées, protections, graphiques, etc.  
  → Pour capturer **structure + mise en forme**, il faut soit **conserver une copie Drive** du classeur (versionning Google), soit exporter en **.xlsx** (avec limites), soit **reconstruire** via un script GAS dédié.

**6) Mini check-lists “But / Finalité”**

**A. Backup standard (après un travail)**

* **But** : versionner code + données exportées.
* **Finalité** : traçabilité, rollback facile.

1. clasp pull (dans chaque projet)
2. Script **snapshot** (CSV + manifest/brief/diff)
3. VS Code → **Stage** → **Commit** → **Push**
4. Créer **Tag** vYYYY.MM.DD-HHmm → **Push Tags**

**B. Rollback “code GAS”**

* **But** : revenir à une version stable.
* **Finalité** : rétablir un comportement connu.

1. git checkout <tag>
2. clasp push (tous projets)
3. Redeploy + vérifier **triggers/scopes**

**C. Rollback “données CSV”**

* **But** : aligner les données avec le code.
* **Finalité** : cohérence fonctionnelle.

1. Choisir **SNAPSHOT\_**\* → **dry-run**
2. Importer les **CSV** (onglet ↔ fichier)
3. Vérifier rapports/exports en sortie

**7) .gitignore conseillé (sécurité & propreté)**

# Secrets / tokens

\*\*/token.json

\*\*/.clasp.json

\*\*/.env

\*\*/credentials.json

# Artefacts lourds ou locaux

node\_modules/

dist/

\*.zip

\*.log

.DS\_Store

Thumbs.db

Ajuste selon ce que tu veux **garder** dans l’historique (ex. tu peux **inclure** les ZIP de snapshot si tu tiens à les versionner — c’est juste plus lourd).

Si tu veux, je te fournis maintenant :

1. un **mémo VS Code** avec captures (nom des boutons) ;
2. un **script tag\_release.ps1** qui crée le tag depuis manifest.json et **pousse** le tout d’un clic ;
3. un **script rollback\_gas.ps1** (sélection d’un tag → git checkout → clasp push en chaîne).

Haut du formulaire

Bas du formulaire

ChatGPT peut commet