Fiche d’activité — CI/CD & Distribution NPM

Rendu : Publication automatisée & Semantic Versioning

Étudiant : Abdoul **Date :** 27/02/2026

Objectif : transformer l’application React en bibliothèque NPM et intégrer une étape de publication automatisée dans le workflow GitHub Actions (avec chaînage needs, bypass SemVer, secrets).

1) Livrables intégrés dans le dépôt

- **.github/workflows/build_test_react.yml** : pipeline en 3 jobs (build-and-test → publish-npm → deploy).
- **Job publish-npm** : comparaison version locale vs version publiée sur NPM ; publication seulement si version locale > NPM ; sinon skip sans échec.
- **app/package.json** : package configuré pour la distribution (name/version, exports, files, scripts build-npm).
- **app/.npmignore** : exclusion des fichiers inutiles à l’exécution du composant.
- **app/src/index.js** : point d’entrée qui exporte le composant PersonForm.
- **README.md** : documentation mise à jour avec lien et exemple d’installation/utilisation NPM.

2) Barème : points couverts

- **Chaînage des jobs (3 pts)** : needs impose l’ordre build-and-test → publish-npm → deploy.
- **Logique de Bypass (4 pts)** : comparaison version locale vs NPM ; publication seulement si version locale strictement supérieure ; sinon skip sans échec.
- **Rigueur SemVer (3 pts)** : 3 commits patch/minor/major (bump version cohérent), + 1 commit README pour démontrer le skip.

3) Procédure de validation (à exécuter pour obtenir les preuves)

1. **Créer le secret GitHub** : Settings → Secrets and variables → Actions → New repository secret → NPM_TOKEN.
2. **Publier une première fois** : pousser la version 1.0.0 (si le nom du package est disponible sur NPM).
3. **Simuler SemVer** : faire 3 commits sur main avec bump de version (patch, minor, major).
4. **Tester le skip** : modifier uniquement le README sans changer la version ; le job publish-npm doit être ignoré (skip) sans faire échouer le pipeline.

4) Plan de commits SemVer recommandé (exemples)

- **Commit A — Patch** (1.0.0 → 1.0.1) : correction mineure (ex : libellé/validation), pas de changement d'API.
- **Commit B — Minor** (1.0.1 → 1.1.0) : ajout rétrocompatible (ex : nouvelle prop optionnelle submitLabel avec valeur par défaut).
- **Commit C — Major** (1.1.0 → 2.0.0) : changement cassant (ex : renommer la prop addPerson → onSubmit).
- **Commit D — Skip** : changement texte README uniquement, version inchangée (publish-npm doit être bypassed).

Preuves attendues

- Historique Git montrant les 4 commits (Patch/Minor/Major/README).
- Onglet GitHub Actions : exécutions avec publications réussies et un run avec publish-npm skippé.
- Page NPM : versions visibles (1.0.0, 1.0.1, 1.1.0, 2.0.0).

Note : si le nom du package est déjà pris sur NPM, utiliser un scope (ex : @votre-scope/test-cycle-tdd-person-form) et mettre à jour package.json + README.