# Guide Git & CI/CD

EBER Platform

*Eglise Baptiste de l'Etoile Rouge*

| Stack | Workflow | Pipeline |
|---|---|---|
| Laravel 11 · Filament v4 | Feature Branch | Lint (Pint) |
| Breeze · Alpine.js | Pull Request · CI/CD | Tests · Build (Vite) |

## 1 — Philosophie du Workflow

### Principe fondamental

La branche main est sacree. Elle ne contient que du code propre, teste et valide. Personne ne pousse jamais directement sur main. Tout passe par une branche dediee et une Pull Request.

> Regle d'or : Un ticket = Une branche = Une Pull Request = Un merge sur main

### Pourquoi ce workflow ?

| Sans workflow | Avec notre workflow |
|---|---|
| Code casse merge sur main | Merge impossible si pipeline rouge |
| Bugs decouverts en production | Bugs detectes avant le merge |
| Style de code incohérent | Style uniforme via Laravel Pint |
| Impossible de retracer les changements | Chaque ticket = un commit clair |
| Conflits difficiles a resoudre | Branches courtes = moins de conflits |

## 2 — Convention de Nommage des Branches

### Format obligatoire

```
TYPE/TICKET-ID-description-courte

# Exemples :
feature/AUTH-001-creation-compte-admin
feature/ACTIV-003-generation-qr-codes
fix/AUTH-002-correction-login-telephone
```

```
chore/mise-a-jour-dependances
hotfix/ACTIV-004-scan-qr-expire
```

### Types de branches

| Prefixe | Usage | Exemple |
|---|---|---|
| `feature/` | Nouvelle fonctionnalite (ticket backlog) | `feature/MEMBER-001-crud-jeunes` |
| `fix/` | Correction d'un bug | `fix/AUTH-002-login-telephone` |
| `chore/` | Maintenance, config, dependances | `chore/maj-filament-v4` |
| `hotfix/` | Correction urgente en production | `hotfix/qr-code-expiration` |
| `refactor/` | Refactoring sans changement fonctionnel | `refactor/audit-service` |

Attention : Ne jamais travailler directement sur main. Toujours creer une nouvelle branche depuis main a jour.

## 3 — Convention de Commits (Conventional Commits)

### Format obligatoire

```
type(scope): description courte en francais

# Exemples reels du projet :
feat(AUTH-001): creation compte utilisateur par admin + envoi credentials
fix(AUTH-001): format numero telephone +229 avec prefix et normalisation
chore: mise en place CI/CD GitHub Actions (lint, tests, build)
chore: correction style de code via Laravel Pint
feat(TECH-004): systeme audit trail (trait, service, middleware, tests)
```

| Type | Usage |
|---|---|
| `feat` | Nouvelle fonctionnalite |
| `fix` | Correction de bug |
| `chore` | Maintenance, config, dependances, Pint |
| `refactor` | Refactoring sans changement de comportement |
| `test` | Ajout ou modification de tests |
| `docs` | Documentation uniquement |
| `style` | Formatage, espaces (pas de logique) |

Regle : Un commit = Une seule chose. Toujours lancer ./vendor/bin/pint avant de commiter.

# 4 — Workflow Complet : Du Ticket au Merge

## Etape 1 — Demarrer un nouveau ticket

```
# 1. Se placer sur main et recuperer les derniers changements
git checkout main
git pull origin main

# 2. Creer la nouvelle branche
git checkout -b feature/AUTH-002-login-email-telephone

# 3. Verifier qu'on est sur la bonne branche
git branch
```

## Etape 2 — Developper et commiter

```
# Verifier les fichiers modifies
git status

# Corriger le style AVANT de commiter (obligatoire)
./vendor/bin/pint

# Verifier que les tests passent
php artisan test

# Ajouter et commiter
git add .
git commit -m "feat(AUTH-002): connexion par email ou telephone"
```

## Etape 3 — Pousser sur GitHub

```
git push origin feature/AUTH-002-login-email-telephone

# Le pipeline CI se declenche automatiquement :
# Lint (Pint) + Tests (PHPUnit) + Build (Vite)
```

## Etape 4 — Creer la Pull Request (VSCode)

1. Cliquer sur l'icone GitHub dans la barre laterale VSCode
2. Section Pull Requests -> cliquer + (Create Pull Request)
3. Base : main | Compare : feature/AUTH-002-...
4. Title : feat(AUTH-002): connexion securisee email ou telephone
5. Cliquer Create Pull Request
6. Attendre que le pipeline passe au vert (1 a 3 minutes)

7. Pipeline vert -> cliquer Merge Pull Request

Protection main : Si le pipeline est rouge, le bouton Merge est bloque. Tu dois corriger le probleme avant de pouvoir merger.

## Etape 5 — Apres le merge

```
# Retourner sur main et recuperer le merge
git checkout main
git pull origin main

# Supprimer la branche locale (optionnel)
git branch -d feature/AUTH-002-login-email-telephone

# Creer la branche du prochain ticket
git checkout -b feature/AUTH-003-changement-mot-de-passe
```

# 5 — Pipeline CI/CD (GitHub Actions)

## Quand se declenche le pipeline ?

| Evenement | Branches | Jobs executes |
|---|---|---|
| `git push` | Toutes les branches (**) | Lint + Tests + Build |
| `Pull Request` | Vers main | Lint + Tests + Build |
| `Merge sur main` | main | Lint + Tests + Build |

## Les 3 jobs du pipeline

| Lint (Laravel Pint) | Tests (PHPUnit) | Build (Vite) |
|---|---|---|
| Verifie PSR-12.<br>Si rouge : Jobs 2+3 skips.<br>`Fix : ./vendor/bin/pint` | Verifie les tests.<br>SQLite en memoire.<br>`Fix : php artisan test` | Verifie les assets.<br>Compile CSS/JS.<br>`Fix : npm run build` |

## Configuration .github/workflows/ci.yml

```
on:
  push:
    branches: ['**']        # Toutes les branches
  pull_request:
    branches: [main]        # PRs vers main seulement

jobs:
  lint:   ./vendor/bin/pint --test
  tests:  php artisan test --env=testing
  build:  npm run build
```

# 6 — Pull Request : Bonnes Pratiques

## Pull Request vs Push direct

| git push | Pull Request |
|---|---|
| Envoie le code sur GitHub | Demande de fusion dans main |
| Declenche le pipeline sur ta branche | Declenche le pipeline sur la PR |
| main n'est pas touche | Si pipeline vert -> merge autorise |
| Toujours faire avant une PR | Obligatoire pour modifier main |

## Creer une PR depuis VSCode

8. Pousser sa branche : git push origin feature/TICKET-XXX
9. Cliquer sur l'icone GitHub dans la barre laterale VSCode
10. Section Pull Requests -> cliquer + (Create Pull Request)
11. Base : main | Compare : feature/TICKET-XXX
12. Title : feat(TICKET-XXX): description de la fonctionnalite
13. Cliquer Create Pull Request
14. Attendre les checks (cercles jaunes -> verts)
15. Pipeline vert + pas de conflits -> cliquer Merge Pull Request
16. Retourner sur main : git checkout main && git pull origin main

Conseil : Garder les PR petites. Une PR = Un ticket. Plus la PR est petite, plus la review est rapide et moins il y a de conflits.

# 7 — Configuration de la Protection de main

## Ruleset GitHub configure

Un Ruleset GitHub bloque tout merge sur main si le pipeline echoue. Cette configuration est deja en place sur le projet EBER Platform.

| Parametre | Valeur configuree |
|---|---|
| Ruleset name | Protect main |
| Enforcement status | Active |
| Target branches | main |
| Require pull request | **Activé** |
| Require status checks | **Activé** |
| Check : Lint (Laravel Pint) | **Required** |
| Check : Tests (PHPUnit) | **Required** |
| Check : Build Assets (Vite) | **Required** |

| | |
|---|---|
| Require branches up to date | **Activé** |

# 8 — Checklist Avant Chaque Merge

## A verifier avant de creer la Pull Request

| |
|---|
| **OK** Le code est sur une branche dediee (pas sur main) |
| **OK** La branche suit la convention : feature/TICKET-ID-description |
| **OK** ./vendor/bin/pint a ete lance (style de code corrige) |
| **OK** php artisan test passe sans erreur |
| **OK** Les nouveaux fichiers sont commites (git status propre) |
| **OK** Les commits suivent la convention : type(scope): description |
| **OK** Le pipeline GitHub Actions est vert sur la branche |
| **OK** La Pull Request pointe vers main |
| **OK** Le titre de la PR suit la convention Conventional Commits |
| **OK** Aucun conflit avec main |
| **OK** Le pipeline de la PR est vert (6 successful checks) |
| **OK** Le bouton Merge Pull Request est disponible |

# 9 — Reference Rapide des Commandes Git

| Commande | Description |
|---|---|
| `git checkout main` | Se placer sur main |
| `git pull origin main` | Recuperer les derniers changements |
| `git checkout -b feature/TICKET-XXX` | Creer une nouvelle branche |
| `git branch` | Voir toutes les branches locales |
| `git status` | Voir les fichiers modifies |
| `./vendor/bin/pint` | Corriger automatiquement le style |
| `./vendor/bin/pint --test` | Verifier le style (sans modifier) |
| `php artisan test` | Lancer tous les tests |
| `php artisan test --filter=NomTest` | Lancer un test specifique |
| `git add .` | Ajouter tous les fichiers modifies |
| `git commit -m "feat(X): description"` | Commiter avec message conventionnel |
| `git push origin feature/TICKET-XXX` | Pousser la branche sur GitHub |
| `git log --oneline -10` | Voir les 10 derniers commits |
| `git diff` | Voir les changements non commites |

| | |
|---|---|
| `git stash` | Mettre de cote des changements |
| `git stash pop` | Recuperer les changements mis de cote |
| `git branch -d feature/TICKET-XXX` | Supprimer une branche locale mergee |