

DOCUMENTATION

Git & Docker

Innov'Events

1. Objectif du document

Ce document décrit ce qui a été mis en place sur Git et Docker pour le projet Innov'Events, afin de permettre :

- une collaboration et un suivi de versions propres (Git / GitHub) ;
- un environnement local reproductible (Docker Compose).

Références principales : `.github/workflows/ci-cd.yml`, `.gitignore`, `docker-compose.yml`, `backend/Dockerfile`, `mobile/Dockerfile`

2. Git (gestion de versions)

2.1 Outils et configuration

Git est utilisé pour versionner l'ensemble du projet (backend, frontend, mobile, docs). Le dépôt est hébergé sur GitHub et utilise GitHub Actions pour l'intégration continue.

Fichiers clés :

- **.gitignore** : exclusion des dépendances (node_modules, vendor), builds (dist), logs, artifacts Cypress, etc.
- **.github/workflows/ci-cd.yml** : pipeline CI/CD.

2.2 Stratégie de branches

Le projet est organisé avec au minimum 2 branches :

- **main** : branche de référence / production.
- **develop** : branche de développement (équivalent d'une branche dev).

Principe : tout développement se fait sur develop (ou sur une branche de fonctionnalité issue de develop). Les fusions vers main se font uniquement quand la fonctionnalité est testée.

Exemple (workflow recommandé) :

```
git checkout develop
git pull

# (optionnel) créer une branche de fonctionnalité
git checkout -b feature/quotes-pdf

# travailler, puis commit
git add .
git commit -m "feat(quotes): génération PDF"

# pousser la branche
git push -u origin feature/quotes-pdf
```

Ensuite : Pull Request vers develop, puis Pull Request develop → main lorsque validé.

2.3 Convention de commits

Les commits sont réalisés souvent avec des messages clairs. Format conseillé :

Préfixe	Usage	Exemple
feat(scope):	Nouvelle fonctionnalité	feat(auth): ajout vérification email
fix(scope):	Correction de bug	fix(events): correction affichage images
refactor(scope):	Refactor sans changement fonctionnel	refactor(api): simplification endpoints

docs:	Documentation	docs: mise à jour README
chore:	Maintenance (outillage, scripts)	chore: update docker-compose

2.4 Gestion des merges et qualité

Bonnes pratiques appliquées :

- Éviter de commit des dépendances (node_modules, vendor) grâce à .gitignore
- Privilégier la fusion via Pull Request (revue et historique propre)
- Conserver main stable (fonctionnel, testé)

3. CI/CD (GitHub Actions) — état actuel

3.1 Déclenchement

Le pipeline se déclenche :

- sur push sur main et develop
- sur pull_request vers main et develop

3.2 Étapes exécutées (CI)

Le workflow .github/workflows/ci-cd.yml contient :

Job Frontend (Angular)

- Installation Node.js 20
- npm ci
- Lint (si présent)
- Tests unitaires + génération coverage (ChromeHeadless)
- Build de production
- Upload de l'artifact de couverture (dossier frontend/coverage/)

Job Backend (PHP)

- Installation PHP (setup-php)
- Vérification syntaxe PHP
- Installation Composer si composer.json présent
- Tests PHPUnit si disponible

Job Sécurité

- npm audit (niveau high)

3.3 Déploiement (CD)

Un job deploy existe et se déclenche uniquement sur push sur main. Cependant, le déploiement automatique n'est pas finalisé : la partie « déploiement réel » est encore à compléter (le job fournit une structure et des logs, mais ne pousse pas vers un hébergeur).

4. Docker (conteneurisation)

4.1 Objectif

Docker Compose est utilisé pour fournir un environnement local reproductible, incluant :

- le backend API (PHP/Apache)
- les bases de données MySQL et MongoDB

- les outils d'administration (phpMyAdmin, Mongo Express)
- un serveur mail de développement (MailHog)
- le mobile (service conteneurisé pour exécution en local)

Référence : `docker-compose.yml`

4.2 Services (docker-compose.yml)

Service	Image/Build	Conteneur	Ports hôte → conteneur	Rôle
backend	build: ./backend	ecf_backend	8080:80	API PHP/Apache
db	mysql:8.0	ecf_db	—	Base MySQL
phpmyadmin	phpmyadmin/phpmyadmin	ecf_pma	8081:80	Admin MySQL
mongo	mongo:latest	ecf_mongo	27017:27017	Base MongoDB
mongo-express	mongo-express	ecf_mongo_express	8082:8081	Admin MongoDB
mailhog	mailhog/mailhog	ecf_mailhog	1025:1025, 8090:8025	SMTP dev + UI
mobile	build: ./mobile	ecf_mobile	4300:4300	App mobile (dev)

URLs pratiques :

- API : `http://localhost:8080`
- phpMyAdmin : `http://localhost:8081`
- Mongo Express : `http://localhost:8082`
- MailHog UI : `http://localhost:8090`
- Mobile : `http://localhost:4300`

4.3 Variables d'environnement (dev)

Service	Variable	Valeur
db (MySQL)	MYSQL_ROOT_PASSWORD	root
db (MySQL)	MYSQL_DATABASE	innovevents
mongo	MONGO_INITDB_ROOT_USERNAME	root
mongo	MONGO_INITDB_ROOT_PASSWORD	root
mongo-express	ME_CONFIG_MONGODB_URL	<code>mongodb://root:root@mongo:27017/</code>

4.4 Volumes (persistence)

Volume	Montage	Usage
db_data	<code>/var/lib/mysql</code>	Persistence données MySQL
mongo_data	<code>/data/db</code>	Persistence données MongoDB
backend_vendor	<code>vendor/ PHP</code>	Évite de recréer vendor à chaque relance
<code>./backend (bind)</code>	<code>→ /var/www/html</code>	Code backend en live (développement)
<code>./mobile (bind)</code>	<code>→ /app</code>	Code mobile en live (développement)

4.5 Commandes d'usage (Docker)

Démarrer l'environnement :

```
docker compose up -d --build
docker compose ps
```

Voir les logs :

```
docker compose logs -f backend
docker compose logs -f db
docker compose logs -f mongo
docker compose logs -f mailhog
docker compose logs -f mobile
```

Arrêter :

```
docker compose down
```

Reset complet (⚠️ supprime les volumes et donc les données) :

```
docker compose down -v
```

Entrer dans un conteneur :

```
docker exec -it ecf_backend bash
docker exec -it ecf_mobile sh
```

4.6 Initialisation de la base MySQL (import via phpMyAdmin)

L'initialisation se fait via import de fichier SQL (dump) :

1. Ouvrir <http://localhost:8081>
2. Se connecter : utilisateur root / mot de passe root
3. Sélectionner la base innovevents
4. Onglet Importer
5. Importer innovevents.sql

5. Remarques

- CD non finalisé : le déploiement automatisé (push vers un serveur/hébergeur) reste à compléter dans le workflow GitHub Actions.
- Les identifiants root/root sont utilisés uniquement en développement dans Docker Compose.

Auteur : Florian Gourinat — Projet ECF 2026 — Innov'Events