Automatisation du développement

Objectifs du module

- Comprendre les avantages
- Découvrir les prérequis
- Appliquer les concepts étape par étape

Avantages de l'automatisation

- Gain de temps
- Qualité du code
- Meilleure collaboration

Cours du jour

- Gestion des dépendances
- Git & méthodes de travail
- Docker et Docker Compose
- README

Gestion des dépendances

Pourquoi bien gérer ses dépendances

- Maîtriser l’environnement de travail, du local à la production
- Reproduire facilement les comportements
- Faciliter la collaboration et les tests

Composer

- Gestionnaire de dépendances PHP
- Permet d’installer des librairies
- Peut spécifier les versions de PHP et d’extensions

Composer

Conseils d’utilisation

- Bien versionner le fichier `composer.lock`
- Ne pas versionner le dossier `vendor`
- Configurer les scripts PHP dans le `composer.json`

Composer

Quelques commandes

- `composer init` : Créer un fichier composer.json
- `composer install` : Installer les dépendances
- `composer update` : Mettre à jour les dépendances
- `composer require <ma_dépendance>` : Ajouter une dépendance

npm

- Gestionnaire de dépendances pour JavaScript (équivalent de Composer pour PHP)
- Peut être remplacé par Yarn, pnpm, etc.
- Mêmes conseils que pour Composer

npm

Quelques commandes

- `npm init` : Créer un fichier package.json
- `npm install` : Installer les dépendances
- `npm ci` : Installer strictement les dépendances (selon package-lock.json)
- `npm update` : Mettre à jour les dépendances
- `npm install --save <ma_dépendance>` : Ajouter une dépendance

Git & méthodes de travail

Git

Inutile de présenter Git ?

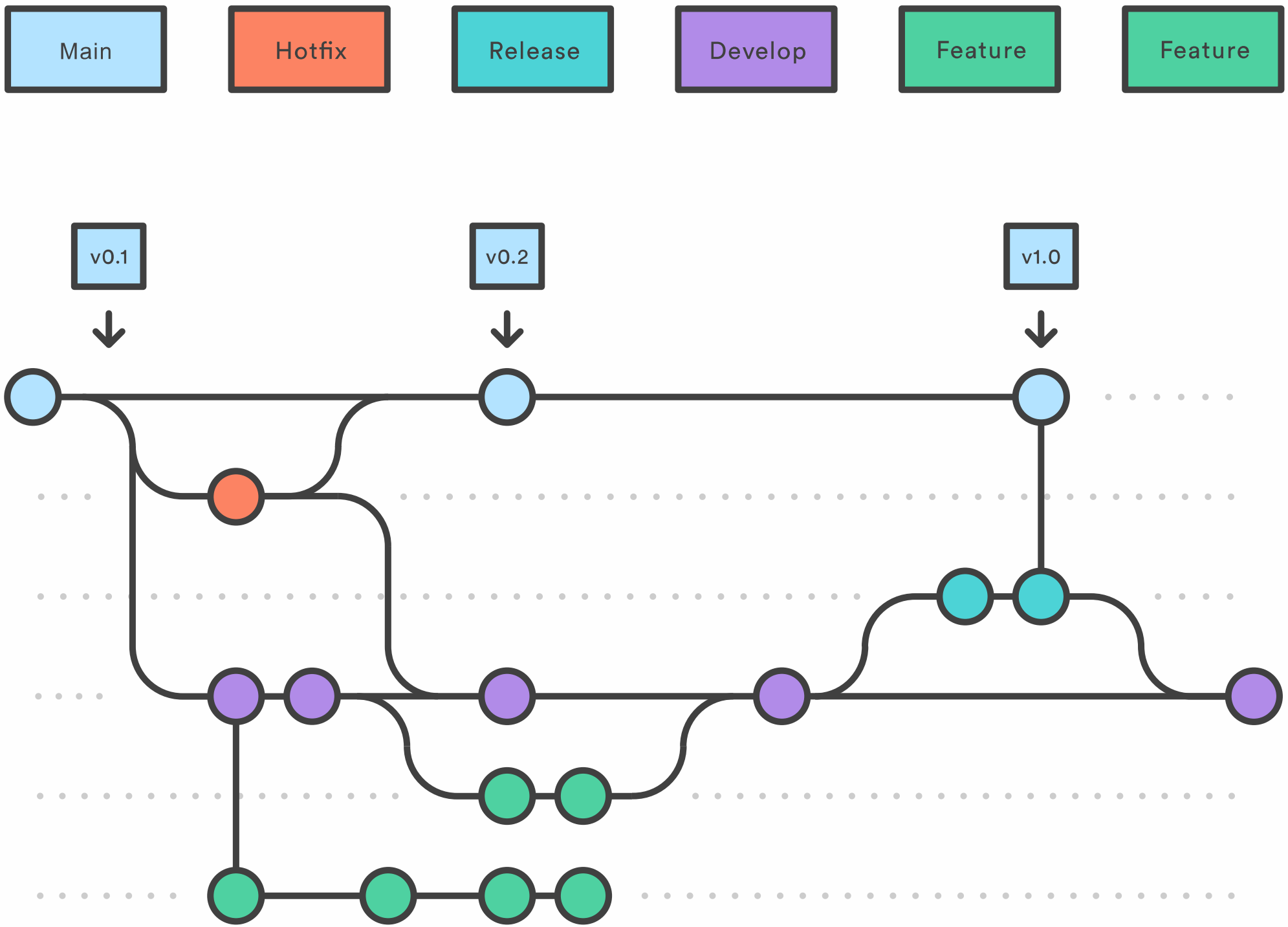
Gitflow

- Méthodologie de travail basée sur Git
- Facilite le travail en équipe
- Permet de gérer plusieurs versions
- Simplifie la mise en production

Gitflow

Principes

- `main` pour la production
- `develop` pour le développement
- `feature/*` pour les fonctionnalités
- `release/*` pour les releases
- `hotfix/*` pour les corrections


Main
Hotfix
Release
Develop
Feature
Feature
v0.1
v0.2
v1.0

Docker & Docker Compose

Docker

- Conteneurisation d'applications
- Reproductibilité des environnements
- Partage facilité des configurations

Dockerfile

- fichier qui permet de décrire comment construire une image docker personnalisée : dépendances, commandes à lancer, extensions nécessaires, etc.
- plutot privilégier les images officielles

Docker Compose

- Gère plusieurs conteneurs simultanément
- Simplifie la configuration des conteneurs
- Facilite l’exécution des commandes Docker

Docker Compose

Quelques commandes

- `docker compose up` : Lance les conteneurs
- `docker compose down` : Arrête les conteneurs
- `docker compose exec <service> <commande>` : Exécute une commande dans un conteneur

Docker Compose

Conseils d’utilisation

- Créer des alias pour les commandes Docker Compose
- Ranger les Dockerfile et fichiers dans un dossier `docker`
- Utiliser des variables d’environnement
- Prioriser les images officielles, sans modification

README & fichier de configuration `.ENV`

.ENV

- Stocke les variables d’environnement
- **NE PAS VERSIONNER**
- Fournir un fichier `.env.example` versionné

README

- Documente le projet
- Doit contenir : informations de base, instructions d’installation

Arborescence

```
mon-projet/
├── docker-compose.yml
├── .env                    # variables d'environnement
├── .env.example            # pour le versionning
├── package.json
├── package-lock.json
├── composer.json
├── composer.lock
├── docker/
│   ├── php/
│   │   └── Dockerfile
│   └── ...
└── src/
    └── ...
├── public/
    └── ...
```

À votre tour