

# Au-delà de l'Artisanat : L'Ingénierie des Prompts pour les Équipes DevOps

Mettre en place une source de vérité unique, versionnée et validée pour vos assistants de code IA.



Prompt Unifier

# Le Chaos Actuel : Gérer les Prompts en Équipe

Sans un système centralisé, les prompts et les règles de codage deviennent une dette technique. Ils sont dispersés, difficiles à maintenir et incohérents.



**Fichiers Locaux** : Des prompts précieux stockés sur les machines individuelles, inaccessibles aux autres.



**Wikis et Documentation** : Souvent obsolètes, sans versionnement clair ni processus de validation.



**Fils de Discussion (Chat)** : Des 'recettes magiques' partagées de manière éphémère et impossibles à retrouver.



**Absence de Versionnement** : Aucune traçabilité des changements. Comment savoir quelle version du prompt a généré un résultat spécifique ?

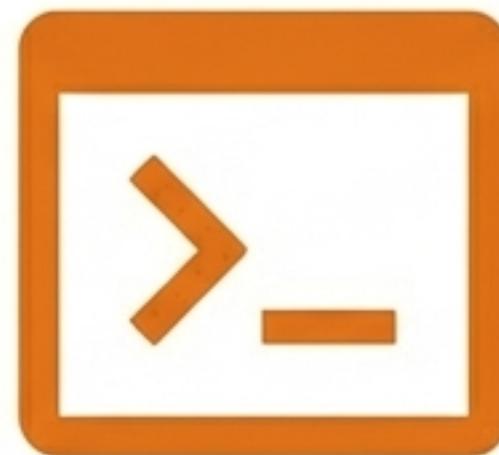


**Manque de Standardisation** : Chaque développeur utilise ses propres variantes, menant à des résultats et une qualité de code hétérogènes.

# La Solution : L'Écosystème Prompt Unifier

Une approche Git-Ops pour la gestion de prompts, transformant le chaos en un workflow structuré, collaboratif et fiable. L'écosystème repose sur deux piliers : un outil CLI et un standard de dépôt.

**‘prompt-unifier’**



**Le Moteur d'Orchestration**

Un outil CLI en Python pour synchroniser, valider et déployer vos prompts et règles.

**‘prompt-unifier-data’**



**La Source de Vérité**

Un dépôt Git structuré servant de collection centralisée pour tous les prompts et règles.

# Un Workflow en Trois Étapes : Synchroniser, Stocker, Déployer



## Dépôts Git Distants

### 1. Synchroniser

Le CLI récupère les dernières versions depuis un ou plusieurs dépôts Git.

## Stockage Local Centralisé

### 2. Stocker

Les fichiers sont consolidés localement en une source unique, prêts à être validés.

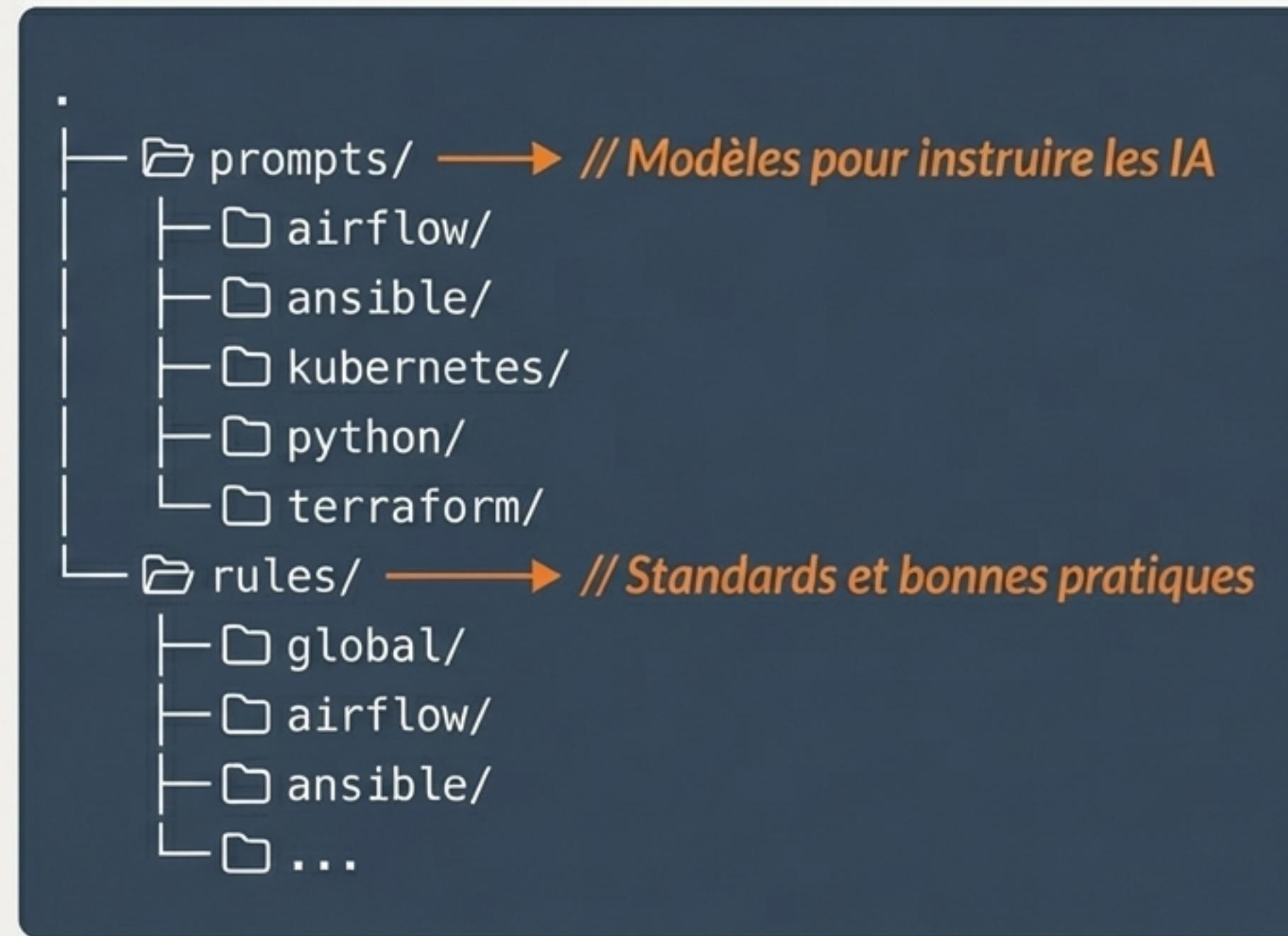
## Configuration Outil IA

### 3. Déployer

Les prompts et règles sont copiés dans les répertoires de configuration de vos outils IA.

# Zoom n°1 : La Structure du Dépôt de Données

L'organisation des fichiers est prévisible et basée sur la technologie, assurant que le contenu est facile à trouver et à gérer.



## `prompts/`

Contient les templates utilisés pour instruire les modèles IA pour des tâches spécifiques (génération, refactoring, analyse, etc.).

## `rules/`

Contient les guides, standards et bonnes pratiques. Ces fichiers servent de contexte enrichi pour les IA, assurant que les résultats respectent les normes de l'équipe.

# Anatomie d'une Règle : Contexte et Standards

Une règle est un document qui définit des standards ou des bonnes pratiques. Le frontmatter YAML structure les métadonnées, tandis que le corps en Markdown fournit le contenu lisible par l'humain et l'IA.

rules/airflow/01-airflow-dag-standards.md

Métadonnées Structurées

```
---
```

```
title: "Airflow DAG Standards"
description: "Core standards for designing and structuring Airflow DAGs..."
tags: [airflow, dag, standards, etl] <--- Métadonnées pour la recherche et le filtrage
version: 1.0.0
author: "prompt-unifier"
category: "standards"
language: "python"
applies_to: ["dags/**/*.py"] <--- Contexte d'application pour l'outilage
---
```

```
# Airflow DAG Standards
## 1. Core DAG Principles
### Idempotency and Determinism
- **Idempotency**: Every task must be idempotent. Rerunning a task...
- **Determinism**: DAG runs must be deterministic...
```

## Métadonnées Structurées

Le frontmatter permet de catégoriser, versionner et attribuer des règles.

## Contenu Lisible

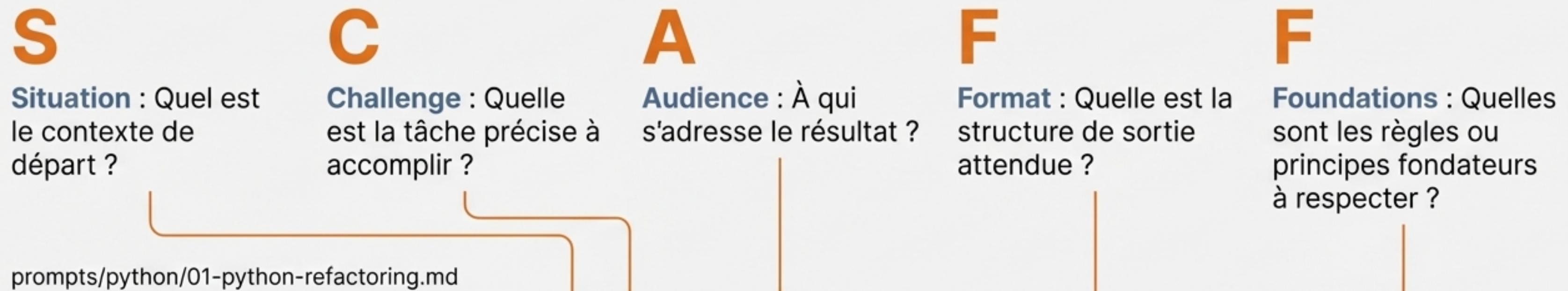
Le Markdown est facile à écrire et à maintenir pour les humains.

## Contexte pour l'IA

Peut être injecté dans un prompt pour guider la génération de code.

# Anatomie d'un Prompt : Le Framework S.C.A.F.F.

Les prompts sont des templates pour instruire une IA. Ils suivent le framework **S.C.A.F.F.** pour garantir clarté, contexte et précision.



```
---  
title: "Python Refactoring Assistant"  
description: "Refactor Python code to improve readability, performance..."  
tags: [python, refactoring, clean-code]  
version: 1.0.0  
category: "development"  
---
```

```
You are an expert Python developer...
```

```
### Situation
```

```
The user provides a piece of Python code...
```

```
### Challenge
```

```
Rewrite the provided code to improve its quality...
```

```
### Audience
```

```
The user is a Python developer...
```

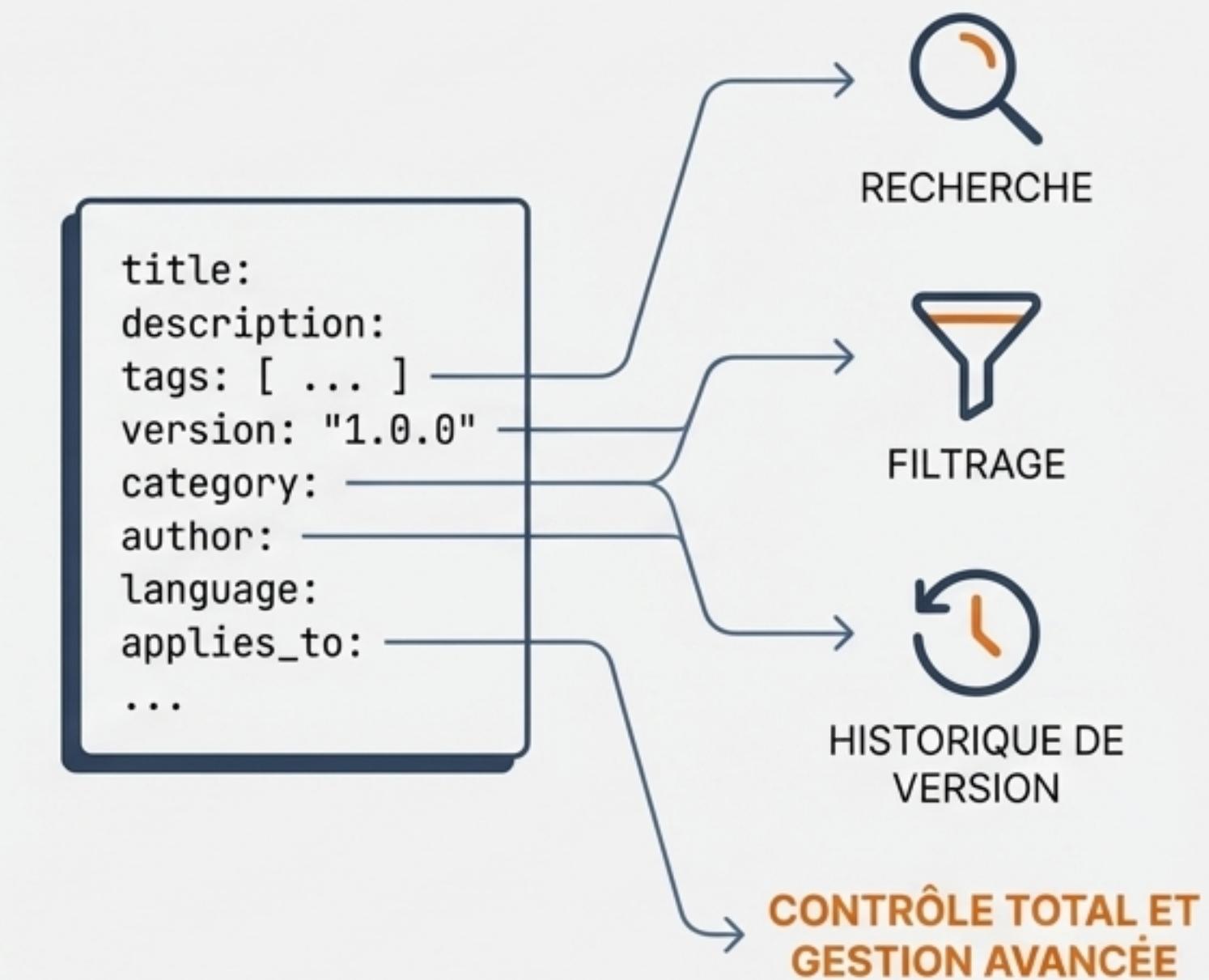
```
### Format
```

# Des Métadonnées Riches pour un Contrôle Total

Chaque fichier est enrichi par un ensemble complet de métadonnées via le frontmatter YAML, permettant une recherche, un filtrage et une gestion avancés.

## Tableau des Champs de Métadonnées

Champ	Type	Description	Requis
title	string	Titre lisible du fichier.	Oui
description	string	Brève description de l'objectif.	Oui
category	string	`development`, `standards`, `testing`, `security`, etc.	Oui
tags	list	Liste de tags pour la catégorisation.	Non
version	string	Version sémantique (ex: "1.0.0").	Non
author	string	L'auteur du contenu.	Non
language	string	`python`, `hcl`, `yaml`, `en`, etc.	Non
applies_to	list	(Pour les règles) Liste de glob patterns où la règle s'applique.	Non



# Le Moteur : `prompt-unifier`, le CLI au Cœur du Système

L'outil CLI est le chef d'orchestre qui automatise le cycle de vie de vos prompts. Chaque fonctionnalité est conçue pour renforcer la collaboration et la qualité.

## Versionnement Git

Bénéfice : Traçabilité et Collaboration

## Gestion Centralisée

Bénéfice : Source de Vérité Unique

## Validation Intégrée

Bénéfice : Qualité et Fiabilité à l'Échelle

## Déploiement Facile

Bénéfice : Intégration dans Votre Workflow Existant

## Support Multi-Dépôts

Bénéfice : Combinez les standards de l'entreprise avec les besoins de l'équipe

## Organisation Structurée

Bénéfice : Découverte et maintenance simplifiées

# Le CLI en Action (1/3) : Initialisation et Synchronisation

Mettez en place votre environnement et synchronisez votre source de vérité en quelques commandes.

## Bloc de Commande 1 : init

Crée un fichier de configuration ` `.prompt-unifier/config.yaml` et prépare le répertoire de stockage local (` `~/.prompt-unifier/storage/` `).

```
# 1. Initialiser prompt-unifier dans votre projet  
prompt-unifier init
```

## Bloc de Commande 2 : sync

Clone ou met à jour un ou plusieurs dépôts Git dans votre stockage local. Gère les conflits avec une stratégie 'le dernier gagne' (last-wins), idéal pour surcharger des prompts globaux avec des versions spécifiques à une équipe.

```
# 2. Synchroniser les prompts depuis un dépôt Git  
prompt-unifier sync --repo https://gitlab.com/waewoo/  
prompt-unifier-data.git
```

```
# Synchroniser plusieurs dépôts (les prompts de  
l'équipe surchargent les prompts globaux)  
prompt-unifier sync \  
--repo https://github.com/company/global-prompts.git  
\  
--repo https://github.com/team/team-prompts.git
```

# Le CLI en Action (2/3) : Vérification et Listing

Validez la syntaxe et la conformité de vos fichiers avant de les déployer, et explorez facilement le contenu disponible.

## Bloc de Commande 1 : validate

Vérifie la syntaxe du frontmatter YAML, la présence des champs requis et la structure des fichiers. Peut être exécuté sur le stockage central ou un répertoire local.

```
# Valider l'intégralité du stockage
# synchronisé
prompt-unifier validate

# Valider uniquement les prompts avec une
# sortie détaillée
prompt-unifier -v validate --type prompts
./mes-prompts/
```

## Bloc de Commande 2 : list et status

Affichez une table de tous les prompts/règles disponibles, avec des options de tri et de filtrage. Vérifiez l'état de synchronisation de vos dépôts.

```
# Lister tous les contenus triés par date de
# modification
prompt-unifier list --sort date

# Lister les prompts avec le tag "python"
prompt-unifier list --tag python

# Vérifier si de nouveaux commits sont
# disponibles sur les dépôts distants
prompt-unifier status
```

# Le CLI en Action (3/3) : Déploiement vers vos Outils

Déployez vos prompts et règles validés vers vos assistants de code IA. Le système gère les spécificités de chaque outil via des "handlers".

## Handlers Supportés

- **Continue** : Support complet, préserve le frontmatter YAML.
  - Destination : `./.continue/prompts/` et `./.continue/rules/`
- **Kilo Code** : Convertit en Markdown pur (sans frontmatter).
  - Destination : `./.kilocode/workflows/` et `./.kilocode/rules/`



## Exemples de Déploiement Avancé

```
# Déployer uniquement les prompts tagués "python" et "refactoring"
prompt-unifier deploy --tags python,refactoring

# Déployer vers Kilo Code et nettoyer les anciens fichiers
prompt-unifier deploy --handlers kilocode --clean

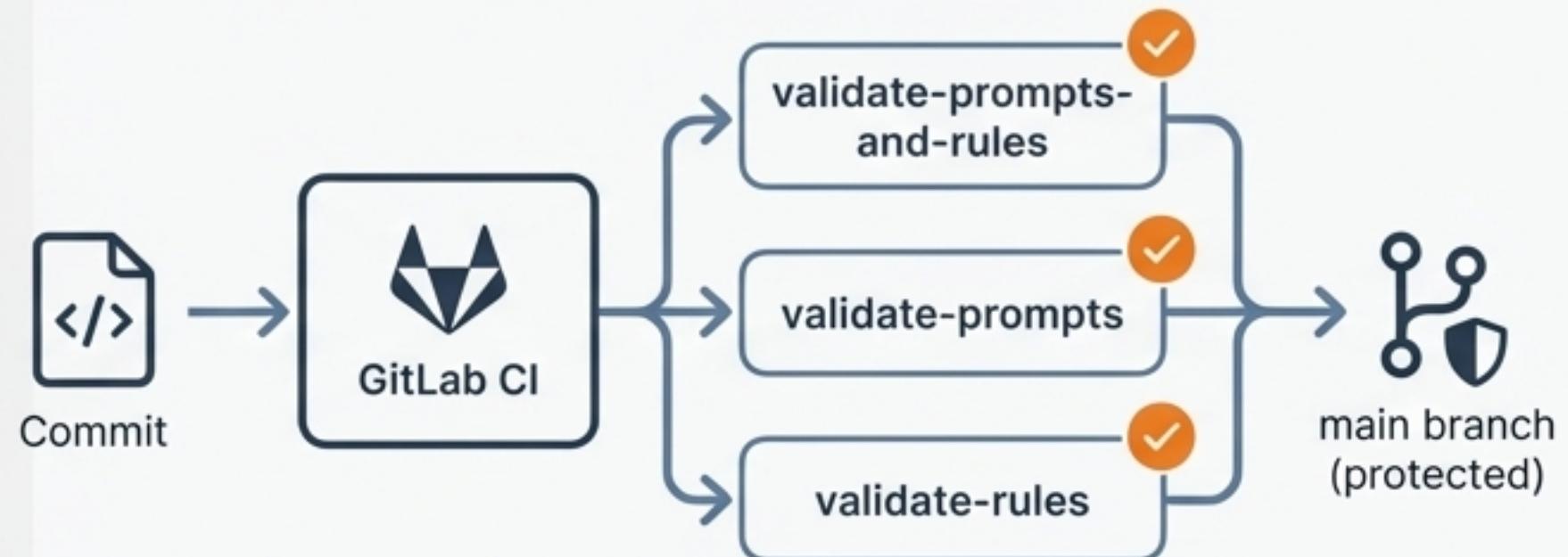
# Simuler un déploiement pour voir les changements sans les appliquer
prompt-unifier deploy --dry-run
```

# Qualité à l'Échelle : Intégration Continue (CI/CD)

Le dépôt prompt-unifier-data inclut un pipeline GitLab CI (`.gitlab-ci.yml`) qui garantit que chaque contribution est validée automatiquement, empêchant les erreurs d'atteindre la branche principale.

## Fonctionnement du Pipeline CI

- **Déclenchement** : S'exécute à chaque `push` et sur chaque `Merge Request`.
- **Jobs** :
  - `validate-prompts-and-rules` : Valide l'ensemble du dépôt.
  - `validate-prompts` : Job optimisé qui ne s'exécute que si des fichiers dans `prompts/` sont modifiés.
  - `validate-rules` : Job optimisé pour les modifications dans `rules/`.



## Avantages

- **Automatisation de la Qualité** : Aucune validation manuelle requise.
- **Feedback Rapide** : Les développeurs savent immédiatement si leurs changements sont conformes.
- **Confiance et Fiabilité** : Seuls des prompts et règles valides sont fusionnés, garantissant la stabilité de la source de vérité.

# Passez à l'Action : Adoptez une Approche Structurée

Commencez à utiliser l'écosystème Prompt Unifier dès aujourd'hui pour professionnaliser la gestion de vos prompts.

## Étapes de Démarrage Rapide

### 1. Installer le CLI

```
pip install prompt-unifier
```

### 2. Initialiser dans votre projet

```
prompt-unifier init
```

### 3. Utiliser le dépôt d'exemple comme base

- **Cloner directement** pour un usage immédiat.
- **Forker le dépôt** pour créer votre propre bibliothèque de prompts d'équipe.
- Lien : <https://gitlab.com/waewoo/prompt-unifier-data>

## Contribution

Ce projet est open source (Licence MIT). Vos contributions sont les bienvenues. Consultez CONTRIBUTING.md pour les directives.

Avant de soumettre, assurez-vous que vos changements passent la validation locale :

```
make validate
```

# De l'Artisanat à l'Ingénierie

## Le Chaos des Prompts



Incohérent  
Opaque  
Fragile  
Isolé

## L'Ingénierie des Prompts



Standardisé  
Fiable  
Collaboratif  
Scalable

L'écosystème Prompt Unifier fournit la structure et les outils nécessaires pour faire de l'assistance par IA une discipline d'ingénierie fiable au sein de votre équipe.