

# Scrutart - Système d'automatisation

Système complet d'automatisation pour la génération de contenu artistique basé sur Wikidata, utilisant Prefect pour l'orchestration des workflows et une interface web moderne pour le pilotage.

## Table des matières

- [Vue d'ensemble](#)
- [Architecture](#)
- [Installation](#)
- [Configuration](#)
- [Utilisation](#)
- [API](#)
- [Dépannage](#)
- [Contribution](#)

## Vue d'ensemble

### Fonctionnalités principales

-  **Automatisation complète** : Génération automatique de pages d'analyse d'artistes
-  **Interface web intuitive** : Dashboard moderne pour piloter les traitements
-  **Suivi en temps réel** : Monitoring des workflows avec WebSockets
-  **Traitement par lots** : Gestion efficace de multiples artistes
-  **Support multilingue** : Génération en français, anglais, espagnol
-  **Métriques et historique** : Suivi détaillé des performances

### Technologies utilisées

- **Backend** : Python, Prefect, Flask, SQLite
- **Frontend** : HTML5, CSS3, JavaScript (Vanilla)
- **Intégrations** : WordPress, Piwigo, Wikidata
- **Base de données** : SQLite (dev), PostgreSQL (prod)

## Architecture

mermaid

```
graph TB
```

```
A[Interface Web] --> B[Serveur Flask]
```

```
B --> C[Prefect Flows]
```

```
C --> D[Wikidata API]
```

```
C --> E[WordPress API]
```

```
C --> F[Piwigo API]
```

```
C --> G[Scrutart State]
```

```
H[WebSockets] --> A
```

```
B --> H
```

```
I[SQLite DB] --> B
```

```
J[Logs & Cache] --> B
```

## Composants

1. **Interface Web** (`scrutart_web_interface.html`) : Dashboard de pilotage
2. **Serveur API** (`scrutart_web_server.py`) : API REST et WebSockets
3. **Flows Prefect** (`scrutart_prefect_automation.py`) : Orchestration des workflows
4. **Configuration** (`scrutart_deployment_config.py`) : Déploiement automatisé

## 🔧 Installation

### Prérequis

```
bash
```

```
# Python 3.9+
```

```
python --version
```

```
# Prefect 2.0+
```

```
pip install prefect>=2.0.0
```

```
# Autres dépendances
```

```
pip install flask flask-cors flask-socketio requests pydantic sqlite3
```

## Installation rapide

1. **Cloner le repository**

```
bash
```

```
git clone https://github.com/your-org/scrutart-automation.git
```

```
cd scrutart-automation
```

2. **Installer les dépendances**

```
bash
```

```
pip install -r requirements.txt
```

### 3. Configuration de l'environnement

```
bash
```

```
# Copier le fichier d'exemple
```

```
cp .env.example .env
```

```
# Éditer les variables d'environnement
```

```
nano .env
```

### 4. Démarrer Prefect Server

```
bash
```

```
# Terminal 1 - Serveur Prefect
```

```
prefect server start
```

### 5. Configuration automatique

```
bash
```

```
# Terminal 2 - Setup complet
```

```
python scrutart_deployment_config.py setup
```

### 6. Démarrer l'agent Prefect

```
bash
```

```
# Terminal 3 - Agent
```

```
python scrutart_deployment_config.py agent
```

### 7. Démarrer le serveur web

```
bash
```

```
# Terminal 4 - Serveur web
```

```
python scrutart_web_server.py
```

### 8. Accéder à l'interface

- Interface Scrutart : <http://localhost:5001>
- Interface Prefect : <http://localhost:4200>

## Configuration

### Variables d'environnement

Créer un fichier `.env` avec :

```

bash

# WordPress
WORDPRESS_API_URL=https://scrutart.grains-de-culture.fr/wp-json/wp/v2/
WORDPRESS_USERNAME=your_username
WORDPRESS_PASSWORD=your_app_password

# Piwigo
PIWIGO_URL=https://galeries.grains-de-culture.fr/
PIWIGO_USERNAME=your_username
PIWIGO_PASSWORD=your_password

# Scrutart State
SCRUTART_STATE_URL=http://localhost:5000
SCRUTART_STATE_TOKEN=your_secure_token

# Prefect
PREFECT_API_URL=http://localhost:4200/api

# Application
SECRET_KEY=your_super_secret_key
SCRUTART_BASE_PATH=/opt/scrutart

```

## Structure des dossiers

```

scrutart-automation/
├── src/
│   ├── scrutart_prefect_automation.py  # Flows Prefect
│   ├── scrutart_web_server.py          # Serveur API
│   ├── scrutart_deployment_config.py  # Configuration déploiement
│   └── templates/
│       └── dashboard.html           # Interface web
├── data/
│   ├── ttl_files/                  # Graphe d'état TTL
│   ├── users/                      # Configuration utilisateurs
│   └── cache/                      # Cache temporaire
├── logs/                          # Fichiers de logs
├── config/                        # Fichiers de configuration
├── tests/                         # Tests unitaires
├── .env                            # Variables d'environnement
├── requirements.txt                # Dépendances Python
└── README.md                      # Cette documentation

```

# Utilisation

## Interface Web

1. Accéder au dashboard : <http://localhost:5001>

2. Traitement individuel :

- Entrer un QID Wikidata (ex: Q296 pour Claude Monet)
- Cliquer sur "Lancer le traitement"
- Suivre le progrès en temps réel

3. Traitement par lots :

- Entrer plusieurs QID (un par ligne)
- Optionnel : donner un nom au lot
- Cliquer sur "Traitement par lot"

4. Suivi des traitements :

- Panneau de statut en temps réel
- Historique des traitements
- Logs détaillés

## Utilisation programmatique

```
python

import asyncio
from scrutart_prefect_automation import process_artist_complete, process_artists_batch

# Traitement d'un artiste
async def example_single():
    result = await process_artist_complete("Q296") # Claude Monet
    print(f"Résultat: {result.success}")

# Traitement par lots
async def example_batch():
    qids = ["Q296", "Q5582", "Q5597"] # Monet, Van Gogh, Cézanne
    results = await process_artists_batch(qids)
    print(f"{len(results)} artistes traités")

# Lancer les exemples
asyncio.run(example_single())
```

## API REST

### Traitement individuel

bash

```
curl -X POST http://localhost:5001/api/flows/process-artist \
-H "Content-Type: application/json" \
-d '{"qid": "Q296", "languages": ["fr"], "auto_publish": "draft"}'
```

## Traitement par lots

bash

```
curl -X POST http://localhost:5001/api/flows/process-batch \
-H "Content-Type: application/json" \
-d '{
  "qids": ["Q296", "Q5582", "Q5597"],
  "batch_name": "Impressionnistes",
  "languages": ["fr", "en"]
}'
```

## Aperçu d'un artiste

bash

```
curl http://localhost:5001/api/artist/Q296/preview
```



## Endpoints principaux

Méthode	Endpoint	Description
POST	/api/flows/process-artist	Lance le traitement d'un artiste
POST	/api/flows/process-batch	Lance un traitement par lots
GET	/api/flows/<id>/status	Statut d'un flow
GET	/api/flows/active	Flows actifs
GET	/api/flows/history	Historique des flows
POST	/api/flows/<id>/cancel	Annule un flow
GET	/api/artist/<qid>/preview	Aperçu d'un artiste
GET	/api/status	Statut de l'API

## WebSocket Events

Event	Description
flow_started	Nouveau flow lancé
flow_status_update	Mise à jour de statut

Event	Description
flow_completed	Flow terminé
flow_cancelled	Flow annulé

## Codes de réponse

Code	Description
200	Succès
400	Erreur de requête
404	Ressource non trouvée
500	Erreur serveur

## Dépannage

### Problèmes courants

#### Erreur de connexion Prefect

```
bash

# Vérifier que le serveur Prefect est démarré
curl http://localhost:4200/api/health

# Redémarrer le serveur si nécessaire
prefect server start
```

#### Erreur de base de données

```
bash

# Réinitialiser la base de données
python scrutart_web_server.py init-db
```

#### Problème d'authentification WordPress

```
bash

# Tester les credentials
curl -u username:password https://scrutart.grains-de-culture.fr/wp-json/wp/v2/users/me
```

#### Agent Prefect non connecté

```
bash
```

```
# Vérifier les work pools  
prefect work-pool ls  
  
# Redémarrer l'agent  
python scrutart_deployment_config.py agent
```

## Logs et monitoring

### Localisation des logs

- **Application** : `logs/scrutart.log`
- **Prefect** : `~/.prefect/logs/`
- **Flask** : Console ou `logs/flask.log`

### Monitoring en temps réel

```
bash  
  
# Suivre les logs  
tail -f logs/scrutart.log  
  
# Monitoring Prefect  
prefect server start --host 0.0.0.0  
# Puis accéder à http://localhost:4200
```

## Base de données

```
bash  
  
# Inspecter la base SQLite  
sqlite3 scrutart_tracking.db  
.tables  
SELECT * FROM flow_runs LIMIT 10;
```

## Performance

### Optimisation des traitements

- Ajuster `batch_size` en fonction des ressources
- Utiliser des work queues séparées pour différents types de traitements
- Configurer des limites de concurrence appropriées

### Monitoring des ressources

```
bash
```

```
# Utilisation CPU/mémoire  
top -p $(pgrep -f "prefect agent")  
top -p $(pgrep -f "scrutart_web_server")
```

## Tests

### Tests unitaires

```
bash  
  
# Lancer tous les tests  
python -m pytest tests/  
  
# Tests spécifiques  
python -m pytest tests/test_flows.py -v  
python -m pytest tests/test_api.py -v
```

### Tests d'intégration

```
bash  
  
# Test complet avec un artiste de test  
python tests/integration_test.py Q296
```

### Tests de performance

```
bash  
  
# Test de charge  
python tests/load_test.py --concurrent=5 --duration=300
```

## Métriques et monitoring

### Dashboard Prefect

- **URL** : <http://localhost:4200>
- **Métriques** : Flows runs, durées, taux de succès
- **Alertes** : Configuration des notifications

### Métriques applicatives

- **Taux de succès** par type d'artiste
- **Temps de traitement** moyen
- **Utilisation des ressources**

- **Erreurs fréquentes**

## Sécurité

### Authentification

- Tokens d'API pour l'accès aux services externes
- Authentification par bearer token pour l'API
- Variables d'environnement pour les secrets

### Bonnes pratiques

- Ne jamais commiter de credentials
- Utiliser HTTPS en production
- Limiter les accès réseau
- Rotation régulière des tokens

## Déploiement en production

### Docker

```
dockerfile

# Dockerfile exemple
FROM python:3.9-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY src/ ./src/
COPY config/ ./config/

CMD ["python", "src/scrutart_web_server.py"]
```

### Docker Compose

```
yaml
```

```
version: '3.8'
services:
  scrutart-web:
    build: .
    ports:
      - "5001:5001"
    environment:
      - PREFECT_API_URL=http://prefect:4200/api
    depends_on:
      - prefect
      - postgres

  prefect:
    image: prefecthq/prefect:2.0.0
    ports:
      - "4200:4200"
    environment:
      - PREFECT_SERVER_DATABASE_CONNECTION_URL=postgresql://user:pass@postgres:5432/prefect

  postgres:
    image: postgres:14
    environment:
      - POSTGRES_DB=prefect
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=pass
```

## Kubernetes

yaml

```
# Exemple de déploiement K8s
apiVersion: apps/v1
kind: Deployment
metadata:
  name: scrutart-web
spec:
  replicas: 2
  selector:
    matchLabels:
      app: scrutart-web
  template:
    metadata:
      labels:
        app: scrutart-web
    spec:
      containers:
        - name: scrutart-web
          image: scrutart:latest
          ports:
            - containerPort: 5001
          env:
            - name: PREFECT_API_URL
              value: "http://prefect-service:4200/api"
```

## 🤝 Contribution

### Structure du projet

- **Flows** : `src/scrutart_prefect_automation.py`
- **API** : `src/scrutart_web_server.py`
- **Frontend** : `templates/dashboard.html`
- **Tests** : `tests/`

### Processus de contribution

1. Fork du repository
2. Branche feature : `git checkout -b feature/nouvelle-fonctionnalite`
3. Commits : `git commit -am 'Ajout nouvelle fonctionnalité'`
4. Push : `git push origin feature/nouvelle-fonctionnalite`
5. Pull Request

### Standards de code

- **Python** : PEP 8, type hints, docstrings

- **JavaScript** : ES6+, JSDoc

- **Tests** : Couverture >80%

## Changelog

### v1.0.0 (2024-08-30)

-  Première version complète
-  Interface web moderne
-  Intégration Prefect
-  WebSockets temps réel
-  Dashboard de monitoring

## Roadmap

- Support PostgreSQL
- Intégration réseaux sociaux
- API GraphQL
- Mobile app
- IA pour optimisation des contenus

## Support

### Documentation

- **Documentation complète** : [docs/](#)
- **API Reference** : [api-docs.md](#)
- **Tutoriels** : [tutorials/](#)

### Contact

- **Issues GitHub** : [Issues](#)
- **Discussions** : [Discussions](#)
- **Email** : [scrutart@grains-de-culture.fr](mailto:scrutart@grains-de-culture.fr)

### Communauté

- **Discord** : [Serveur Scrutart](#)
- **Twitter** : [@scrutart\\_ai](#)

## Licence

Ce projet est sous licence MIT. Voir le fichier [LICENSE](#) pour plus de détails.

---

