```
# Atelier Professionnel Markdown - Rapport de Projet

## Introduction

Ce document détaille la réalisation du projet "Atelier Professionnel Markdown". Il couvre le
- Création d'un document Markdown complet démontrant diverses fonctionnalités.
- Comparaison d'outils de conversion Markdown vers HTML/PDF et sélection d'un outil.
- Versionnement du projet avec Git.
- Dockerisation de la solution pour une consultation facilitée au format HTML.
- Mise en place d'une pipeline d'Intégration Continue et de Déploiement Continu (CI/CD) pour

Chaque choix technique est justifié au fur et à mesure de sa présentation. L'objectif final

---

## 1. Objectifs du Projet (Rappel)

Les objectifs principaux de cet atelier étaient :

*   **Écrire un document en Markdown** complet (titres, mises en forme, images, schémas, cod
*   **Comparer 3 outils de conversion** Markdown vers HTML/PDF et choisir le plus adapté.
*   **Créer un dépôt Git** pour versionner tous les documents.
*   **Créer une image Docker** pour accéder à la documentation au format HTML.
*   **Automatiser le déploiement** (CI/CD) pour visionner le HTML sur un site web de product

---

## 2. Comparaison et Choix de l'Outil de Conversion

L'objectif était de comparer 3 outils pour convertir le Markdown en HTML et PDF.

### 2.1. Critères de Comparaison

Les critères suivants, inspirés du sujet, ont été utilisés :
1.  **Facilité d'Utilisation :** Installation, configuration, courbe d'apprentissage.
2.  **Compatibilité et Intégration :** Fonctionnement avec d'autres technologies (Node.js, P
3.  **Extensions et Personnalisation :** Support des extensions Markdown (GFM), plugins, thè
3.  **Performance :** Vitesse de conversion.
5.  **Support des Extensions Markdown :** Tableaux, listes de tâches, blocs de code, Mermaid
6.  **Communauté et Support :** Disponibilité de l'aide, documentation.
7.  **Licence et Coût :** Gratuit, open-source, payant.
8.  **Sortie et Formats Supportés :** HTML, PDF, DOCX, etc.
9.  **Qualité du Rendu :** Esthétique et fidélité des sorties HTML et PDF.
10. **Maintenance et Mises à Jour :** Fréquence des mises à jour.

### 2.2. Outils Évalués
```

1. **Pandoc :**
    * *Description :* Le "couteau suisse" de la conversion de documents. Écrit en Haskell.
    * *Avantages :* Supporte une multitude de formats d'entrée et de sortie (dont PDF via
    * *Inconvénients :* Installation de LaTeX peut être lourde pour le PDF. Syntaxe des co

2. **MkDocs :**
    * *Description :* Générateur de sites statiques pour la documentation, écrit en Python
    * *Avantages :* Très simple à utiliser, thèmes personnalisables (ex: Material for MkDo
    * *Inconvénients :* Principalement axé HTML. La génération PDF nécessite des plugins s

3. **Markdown-it (via `markdown-it-cli`) :**
    * *Description :* Parseur Markdown pour Node.js, rapide et extensible. `markdown-it-cl
    * *Avantages :* Conforme à CommonMark, très extensible via des plugins, rapide pour la
    * *Inconvénients :* Principalement pour HTML. Pas de support PDF natif (nécessite de c

2.3. Choix de l'Outil : Pandoc

Pandoc a été choisi comme outil principal pour ce projet.

Justification :
* **Polyvalence des formats :** Capacité à générer à la fois du HTML de qualité et des PDF
* **Contrôle fin :** Nombreuses options en ligne de commande pour personnaliser la sortie
* **Support des extensions :** Bonne gestion des extensions Markdown comme GFM, les tablea
* **Intégration :** Facile à intégrer dans des scripts et des workflows CI/CD.
* **Mermaid :** Peut être géré avec des filtres (ex: `pandoc-mermaid-filter` ou une soluti

Gestion du souligné : En HTML, la balise `<u>` sera utilisée. Pour le PDF généré par Pando

---

3. Versionnement avec Git

Ce projet est versionné avec Git depuis son initialisation.

3.1. Initialisation et Dépôt
Un dépôt Git local a été créé avec `git init`.
Un dépôt distant a été créé sur GitHub (`VOTRE_NOM_UTILISATEUR/atelier-markdown-pro`) et lié

3.2. Commandes Clés Utilisées
* `git add <fichier>` ou `git add .`: Pour ajouter les fichiers au staging.
* `git commit -m "message"`: Pour enregistrer les modifications.
* `git push origin main`: Pour envoyer les commits vers le dépôt distant.
* `git pull origin main`: Pour récupérer les modifications distantes.
* `git branch <nom_branche>`: Pour créer une branche.
* `git checkout <nom_branche>`: Pour changer de branche.

```
### 3.3. Fichier `.gitignore`
Un fichier `.gitignore` a été créé pour exclure les fichiers et dossiers non nécessaires au
```

.gitignore

output/ node_modules/ site/ **pycache**/ *.pyc .DS_Store

Justification : Git est l'outil standard pour le contrôle de version. Il permet de suivre l'historique des modifications, de collaborer et de revenir à des versions antérieures si besoin.

---

5. Dockerisation pour Consultation HTML

Pour faciliter la consultation du document au format HTML, une image Docker a été créée. Elle utilise Pandoc pour convertir `RAPPORT_PROJET.md` en HTML et Nginx pour servir ce fichier.

5.1. Choix Techniques pour Docker

- **Image de base pour la conversion :** `pandoc/latex:latest`. Cette image contient Pandoc et une distribution TeX Live complète, permettant la conversion vers PDF si nécessaire, mais surtout elle garantit que Pandoc est disponible. Pour une image plus légère uniquement pour HTML, `pandoc/core` aurait pu être utilisé.
- **Serveur web :** `nginx:alpine`. Nginx est un serveur web performant et léger, et la version Alpine est particulièrement optimisée en taille.
- **Build multi-étapes :** Pour minimiser la taille de l'image finale. La première étape (`builder`) génère le HTML, la seconde (`nginx`) ne contient que le serveur et le fichier HTML.

5.2. Dockerfile

Voici le contenu du `Dockerfile` :

```
# Dockerfile

# Étape 1: Construire le HTML avec Pandoc
FROM pandoc/latex:latest AS builder
# Alternative plus légère si PDF n'est pas une priorité pour l'image Docker :
# FROM pandoc/core:latest AS builder

WORKDIR /source
COPY RAPPORT_PROJET.md .
# Si vous avez des images locales dans un dossier 'assets/images/'
```

```
# COPY assets/ ./assets/

# Convertir Markdown en HTML.
# L'option --self-contained inclut le CSS dans le HTML.
# Un CSS externe est utilisé ici pour un style type GitHub.
RUN pandoc RAPPORT_PROJET.md \
    -s \
    -o /app/index.html \
    --metadata title="Rapport Projet Markdown" \
    --css=https://cdnjs.cloudflare.com/ajax/libs/github-markdown-css/5.1.0/github-markdown.m
    --self-contained
    # Pour les images locales, assurez-vous qu'elles sont copiées et que Pandoc peut les tro
    # Si --self-contained ne les embarque pas, il faudra les copier dans l'étape Nginx auss

# Étape 2: Servir le HTML avec Nginx
FROM nginx:alpine
WORKDIR /usr/share/nginx/html
# Supprimer le contenu par défaut de Nginx
RUN rm -rf ./*
# Copier le HTML généré depuis l'étape de build
COPY --from=builder /app/index.html .
# Si des assets (images) ne sont pas embarqués et doivent être servis :
# COPY --from=builder /source/assets/ ./assets/

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

5.3. Commandes Docker

- **Construire l'image :**

  `docker build -t atelier-markdown-doc .`

- **Lancer le conteneur :**

  `docker run -d -p 8080:80 --name markdown_viewer atelier-markdown-doc`

  Le rapport est alors accessible sur `http://localhost:8080`.

Justification : Docker permet de packager l'application (ici, le visualiseur de documentation) avec toutes ses dépendances, garantissant qu'elle fonctionnera de la même manière partout.

---

6. Intégration et Déploiement Continus (CI/CD)

Une pipeline CI/CD a été mise en place avec GitHub Actions pour automatiser la conversion du fichier Markdown en HTML et son déploiement sur GitHub

Pages après chaque `push` sur la branche `main`.

6.1. Choix Techniques pour la CI/CD

- **Plateforme :** GitHub Actions, pour son intégration native avec GitHub et sa gratuité pour les dépôts publics.
- **Déploiement :** GitHub Pages, pour héberger gratuitement des sites statiques.

6.2. Fichier de Workflow (`.github/workflows/main.yml`)

```yaml
# .github/workflows/main.yml
name: Build and Deploy Markdown to GitHub Pages

on:
  push:
    branches:
      - main # Se déclenche sur push vers la branche main

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    permissions:
      contents: write # Nécessaire pour que peaceiris/actions-gh-pages puisse écrire sur la

    steps:
      - name: Checkout code
        uses: actions/checkout@v3 # Récupère le code du dépôt

      - name: Setup Pandoc
        uses: r-lib/actions/setup-pandoc@v2
        # with:
        #   pandoc-version: 'latest' # ou une version spécifique

      # Optionnel : Si des filtres sont nécessaires pour Mermaid ou autre
      # - name: Install Node.js (pour pandoc-mermaid-filter par exemple)
      #   uses: actions/setup-node@v3
      #   with:
      #     node-version: '18'
      # - name: Install pandoc-mermaid-filter
      #   run: npm install -g pandoc-mermaid-filter

      - name: Convert Markdown to HTML
        run: |
          mkdir public
          pandoc RAPPORT_PROJET.md \
```

```
        -s \
        -o public/index.html \
        --metadata title="Rapport Projet Markdown" \
        --css=https://cdnjs.cloudflare.com/ajax/libs/github-markdown-css/5.1.0/github-ma
        --self-contained
        # Si vous avez des images locales dans assets/images:
        # mkdir -p public/assets/images
        # cp -r assets/images/* public/assets/images/
        # Et ajustez la commande pandoc :
        # pandoc RAPPORT_PROJET.md -s -o public/index.html --metadata title="Rapport" --
        # (sans --self-contained pour les images, qui seront des fichiers séparés)

    - name: Deploy to GitHub Pages
      uses: peaceiris/actions-gh-pages@v3
      with:
        github_token: ${{ secrets.GITHUB_TOKEN }}
        publish_dir: ./public
        # cname: docs.votre-domaine.com # Si vous avez un domaine personnalisé
```

6.3. Configuration de GitHub Pages

Dans les paramètres du dépôt GitHub (`Settings > Pages`): * **Source :** "Deploy from a branch" (avant la première exécution) ou "GitHub Actions" (après que l'action ait créé la branche `gh-pages` ou directement si l'action est configurée pour). * **Branch :** Si "Deploy from a branch", la branche `gh-pages` et le dossier `/ (root)` sont sélectionnés une fois que l'action `peaceiris/actions-gh-pages` a poussé la première fois.

Après la première exécution réussie du workflow, le site est accessible à l'adresse : `https://noe-abbati.github.io/atelier-markdown-pro/`.

Justification : La CI/CD automatise les tâches répétitives, réduit les erreurs humaines et assure que la documentation en ligne est toujours à jour avec la dernière version du code source Markdown.

---

7. Conclusion

Cet atelier a permis de mettre en pratique un ensemble de technologies et de méthodologies modernes pour la création, la gestion et la diffusion de documentation. En partant d'un simple fichier Markdown, nous avons exploré : * La richesse de la syntaxe Markdown. * Le processus de sélection d'outils de conversion. * L'importance du versionnement avec Git. * La portabilité offerte par Docker. * L'efficacité de l'automatisation avec la CI/CD via GitHub Actions et GitHub Pages.

Ce projet démontre comment ces outils, combinés, peuvent significativement

améliorer la productivité et la qualité dans la gestion de la documentation technique et d'autres types de contenu. "'