デプロイガイド

バージョン: 1.0.0
最終更新: 2026-1-15
リポジトリ: https://github.com/J1921604/OCR-PDF-Converter

このドキュメントでは、OCR検索可能PDF変換アプリケーション（OnnxOCR 2025.5 + PaddleOCR 2.7.0.3 + Python Backend）のローカルセットアップとGitHub Pagesデプロイ方法を説明します。

重要: GitHub Pages は静的ホスティングのため **フロントエンドのみ**を公開します。OCR処理はバックエンドが必要で、Pages（HTTPS）から `http://localhost:5000` を呼ぶことは mixed content でブロックされます。

目次

前提条件

- **Python 3.10.11** インストール済み
- **Node.js 18以上** インストール済み
- **Git** インストール済み
- GitHubアカウント

ローカル環境セットアップ

1. リポジトリのクローン

```
git clone https://github.com/J1921604/OCR-PDF-Converter.git
cd OCR-PDF-Converter
```

2. Python仮想環境のセットアップ

```
# Python 3.10.11で仮想環境を作成
py -3.10 -m venv .venv

# 仮想環境をアクティベート（PowerShell）
.\.venv\Scripts\Activate.ps1

# 仮想環境をアクティベート（Bash）
source .venv/Scripts/activate

# Pythonパッケージをインストール
python -m pip install --upgrade pip
python -m pip install -r requirements.txt
```

3. Node.jsパッケージのインストール

```
npm install
```

4. アプリケーションの起動

方法1: ワンコマンド起動（推奨）

```
.\start-full.ps1
```

このスクリプトは以下を自動実行します：

- Python仮想環境のセットアップ
- 依存パッケージのインストール
- Pythonバックエンドの起動（localhost:5000）
- Reactフロントエンドの起動（localhost:8080）

方法2: 個別起動

ターミナル1 - Pythonバックエンド:

```
.\.venv\Scripts\Activate.ps1
python backend/app.py
```

ターミナル2 - Reactフロントエンド:

```
npm start
```

5. アプリケーションへのアクセス

ブラウザで以下のURLを開きます：

```
http://localhost:8080
```

バックエンドAPIは `http://localhost:5000` で動作します。

GitHub Pages設定

1. GitHub Pagesの有効化

1. GitHubリポジトリページを開く
2. 「Settings」タブをクリック
3. 左サイドバーの「Pages」をクリック
4. 「Build and deployment」セクションで：
   - **Source**: 「GitHub Actions」を選択
5. 変更を保存

2. ワークフローの確認

`.github/workflows/pages.yml` ファイルが存在することを確認します。このファイルは自動的にビルドとデプロイを実行します。

GitHub Actionsによる自動デプロイ

デプロイの流れ

1. **コードのプッシュ**: `main` ブランチにコードをプッシュ

   ```
   git add .
   git commit -m "Update application"
   git push origin main
   ```

2. **自動ビルド**: GitHub Actions が自動的に起動し、以下を実行：
   - 依存関係のインストール (`npm ci`)
   - テストの実行 (`npm test`)
   - アプリケーションのビルド (`npm run build`)
   - ビルド成果物のアップロード
3. **自動デプロイ**: ビルドが成功すると、GitHub Pages に自動デプロイ

ワークフローの確認

1. GitHubリポジトリの「Actions」タブを開く
2. 最新のワークフロー実行を確認
3. 緑のチェックマークが表示されればデプロイ成功

デプロイステータスの確認

デプロイが完了すると、以下のURLでアプリケーションにアクセスできます：

```
https://YOUR_USERNAME.github.io/OCR-PDF-Converter/
```

手動デプロイ

GitHub Actions を使用せず、手動でデプロイする場合：

1. gh-pages パッケージを使用

```
# 依存関係のインストール
npm install

# ビルド
npm run build

# デプロイ
npm run deploy
```

2. package.json の deploy スクリプト

すでに `package.json` に以下のスクリプトが設定されています：

```
{
  "scripts": {
    "deploy": "gh-pages -d dist"
  }
}
```

カスタムドメイン設定

独自ドメインを使用する場合：

1. DNS設定

ドメインプロバイダーで以下のDNSレコードを追加：

```
A    @    185.199.108.153
A    @    185.199.109.153
A    @    185.199.110.153
A    @    185.199.111.153
```

または、サブドメインの場合：

```
CNAME    www    YOUR_USERNAME.github.io
```

2. GitHub Pages設定

1. リポジトリの「Settings」→「Pages」を開く
2. 「Custom domain」にドメイン名を入力
3. 「Save」をクリック
4. 「Enforce HTTPS」にチェックを入れる（推奨）

3. CNAME ファイルの追加

`public/CNAME` ファイルを作成し、ドメイン名を記載：

```
your-domain.com
```

トラブルシューティング

ビルドエラー

症状: GitHub Actions のビルドが失敗する

解決方法:

1. 「Actions」タブでエラーログを確認
2. ローカルで `npm run build` を実行してエラーを再現
3. 依存関係を更新: `npm install`
4. エラーを修正してプッシュ

ページが表示されない

症状: デプロイ後、404エラーが表示される

解決方法:

1. GitHub Pages の設定を確認（Settings → Pages）
2. URL が正しいか確認: `https://YOUR_USERNAME.github.io/OCR-PDF-Converter/`
3. デプロイが完了するまで5〜10分待つ
4. キャッシュをクリア: Ctrl+Shift+R（Windows/Linux）、Cmd+Shift+R（Mac）

アセットが読み込まれない

症状: CSS や JavaScript が読み込まれない

解決方法:

1. `webpack.config.js` で `output.publicPath` が正しく設定されているか確認
2. ビルド時に `PUBLIC_URL` 環境変数を設定:

   ```
   PUBLIC_URL=/OCR-PDF-Converter npm run build
   ```

3. GitHub Actions ワークフローで環境変数が設定されているか確認

CORS エラー

症状: ブラウザのコンソールに CORS エラーが表示される

解決方法:

1. すべてのアセットが同じドメインから提供されているか確認
2. PDF.js ワーカーが CDN から読み込まれているか確認
3. `public/index.html` の CSP ヘッダーを確認

GitHub Pages 上でOCRが動かない

症状: Pages ではUIは表示されるが、OCR処理が実行できない / APIに接続できない

原因: Pages は HTTPS で提供されるため、`http://localhost:5000` へのアクセスは mixed content としてブロックされる

解決方法:

1. OCRを使う場合はローカルで `\.\start-full.ps1` を実行して `http://localhost:8080` から利用する
2. Pages上でもOCRを動かしたい場合は、バックエンドを **HTTPS** で公開し、フロント側の `REACT_APP_API_URL` をそのURLに向けてビルドする

PaddleOCR SSL証明書検証エラー

症状: PaddleOCRエンジン使用時に以下のエラーが表示される

```
SSLError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed:
self signed certificate in certificate chain
```

原因: 企業プロキシ環境やファイアウォール配下で、PaddleOCRのモデルダウンロード時に自己署名証明書が使用されている

実装済み対応: backend/main.py（L1-L40）で以下の対応が実装されています：

1. SSL証明書検証を無効化:

   ```
   import ssl
   ssl._create_default_https_context = ssl._create_unverified_context
   ```

2. urllib3警告を抑制:

   ```
   import urllib3
   urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
   ```

3. 環境変数を設定:

   ```
   os.environ['REQUESTS_CA_BUNDLE'] = ''
   os.environ['CURL_CA_BUNDLE'] = ''
   ```

4. requests.getをパッチング:
   - `ensure_paddleocr_available()` 関数でrequests.getに `verify=False` を自動注入

手動対応（上記が効かない場合）：

```
# PowerShellで環境変数を設定
$env:REQUESTS_CA_BUNDLE = ""
$env:CURL_CA_BUNDLE = ""

# 仮想環境をアクティベート
.\.venv\Scripts\Activate.ps1

# バックエンド起動
py -3.10 backend/app.py
```

セキュリティ注意事項:

- この設定はローカル開発環境でのみ使用してください
- プロダクション環境では適切なCA証明書を設定することを推奨します

デプロイ後の確認事項

デプロイ後、以下を確認してください：

1. 基本機能

- [ ] ページが正常に表示される
- [ ] ファイルアップロードが動作する
- [ ] （ローカル起動時）OCR処理が実行できる
- [ ] （ローカル起動時）PDFダウンロードが動作する

2. パフォーマンス

- [ ] ページの読み込み速度が許容範囲内
- [ ] OCR処理が適切な速度で実行される
- [ ] メモリ使用量が適切

3. ブラウザ互換性

- [ ] Chrome で動作する
- [ ] Firefox で動作する
- [ ] Safari で動作する
- [ ] Edge で動作する

4. セキュリティ

- [ ] HTTPS で提供されている
- [ ] CSP ヘッダーが適切に設定されている
- [ ] 機密情報が漏洩していない

参考リンク

- GitHub Pages ドキュメント
- GitHub Actions ドキュメント
- gh-pages パッケージ
- Webpack デプロイガイド

サポート

問題が解決しない場合：

1. GitHub Issues で質問
2. プロジェクトの README.md を確認
3. 完全仕様書 を参照

最終更新: 2026-1-15
バージョン: 1.0.0