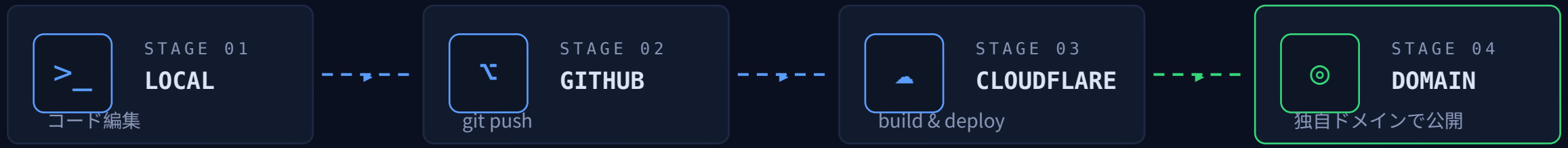
GITOPS DEPLOYMENT PIPELINE

git push が、デプロイになる。

GitHub × Cloudflare Workers — 自動デプロイ環境 構築ガイド



WHAT YOU GET

push するだけで、本番が反映される

01

push が唯一のトリガー

本番ブランチへの push を検知し、自動でビルド・デプロイ。

02

独自ドメインに自動反映

ドメインは常に最新の本番版を指す。手動作業は不要。

03

GitHub で状態確認

デプロイの成否はコミット横のチェックで監視できる。

PREREQUISITES

前提条件

GitHub アカウント

リポジトリを作成できること

Cloudflare アカウント

無料プランで可

独自ドメイン

Cloudflare の DNS（ゾーン）配下にあること

Node.js

LTS 版を推奨 / node -v で確認

Git

git --version で確認

—

workers.dev だけで試すなら独自ドメインは不要

SETUP · 6 STEPS

構築の流れ

01 **プロジェクト作成**

ローカルで Worker を生成

02 **GitHub へ push**

空リポジトリへ初回 push

03 **リポジトリ連携**

Workers Builds で接続

04 **ビルド / デプロイ設定**

ブランチとコマンドを指定

05 **初回デプロイ確認**

workers.dev で動作確認

06 **独自ドメイン割当**

Custom Domain を追加

STEP 01

ローカルで Worker プロジェクトを作成

公式スキャフォールド C3 で雛形を生成。
設定ファイル wrangler.jsonc が作られる。

静的ファイルを配信する場合は
assets ディレクトリを指定する。

ローカル確認は npx wrangler dev

bash

```
npm create cloudflare@latest your-app
cd your-app
```

wrangler.jsonc

```
{
  "name": "your-app",
  "main": "src/index.ts",
  "compatibility_date": "YYYY-MM-DD",  // 作成日
  "assets": { "directory": "./public" }
}
```

STEP 02

Git を初期化し、 GitHub へ push

GitHub 側で空のリポジトリを 1 つ作成しておく。

bash

```
git init
git add -A
git commit -m "Initial commit"
git branch -M main
git remote add origin \
  https://github.com/your-account/your-repo.git
git push -u origin main
```

重要: API キー・パスワード等や .env / .dev.vars をコミットしない。 .gitignore に .dev.vars* と .env* を追加しておく。

STEP 03

Cloudflare と GitHub を連携（Workers Builds）

1. Workers & Pages
2. Create application
3. Import a repository
4. Git アカウントを認可
5. 対象リポジトリを選択

セキュリティ：認可時は「Only select repositories」を選び、対象リポジトリだけにアクセスを限定する。

設定ファイルが無い場合、Cloudflare がフレームワークを自動検出し設定 PR を自動作成する。

STEP 04

ビルド / デプロイ設定

Production branch

`main`

push を検知するブランチ

Build command

`npm run build`

不要なら空欄でも可

Deploy command

`npx wrangler deploy`

既定値で使えることが多い

「Save and Deploy」を選択すると、初回ビルドとデプロイが走る。

STEP 05

初回デプロイを確認

完了すると workers.dev の URL が払い出される。ブラウザでアクセスして動作を確認する。

```
https://your-app.<your-subdomain>.workers.dev
```

> ビルドログは Deployments タブ最下部の「View build history」から確認できる。

STEP 06

独自ドメインを割り当てる

対象ドメインが Cloudflare のゾーン配下にあることが前提。

Worker → Settings → Domains & Routes →
Add → Custom Domain からホスト名を入力。

DNS レコードと TLS 証明書は自動発行される。

DAILY FLOW

あとは push するだけ

— terminal —

```
# ローカルでコードを編集
$ git add -A
$ git commit -m "Update layout"
$ git push
# → 数十秒〜数分後、独自ドメインに自動反映
```

OPTIONAL

値を渡す 3 つの方法

vars

平文の設定値

Git に載るか

Git に載る

用途

API ホスト名など

Secrets

暗号化された秘密文字列

Git に載るか

載らない

用途

パスワード・ API キー

Bindings

リソースへの参照

Git に載るか

参照情報は載る

用途

D1 ・ KV ・ R2 接続

> DB は Secrets ではなく Bindings で接続する。 Cloudflare 純正 D1 ならパスワード管理は不要。

OPTIONAL · DATABASE

データベース（D1）を使う

bash

```
npx wrangler d1 create your-db
```

wrangler.jsonc

```
"d1_databases": [
  { "binding": "DB",
    "database_name": "your-db",
    "database_id": "..." }
]
```

再デプロイで DB の中身は消えない。入れ替わるのは Worker のコードだけ。

コスト注意：読み取り行数＝スキャン行数。無料枠内なら小規模は無料。頻出クエリには必ずインデックスを張る。

TROUBLESHOOTING

うまくいかないとき

push してもデプロイされない	push 先ブランチが Production branch と一致しているか
ビルドが失敗する	Deployments → ビルドログでエラー内容を確認
独自ドメインで表示されない	ドメインが Cloudflare ゾーン配下か / DNS 伝播待ち
Secrets が読めない	wrangler secret put で登録済みか、名前が一致しているか

BEST PRACTICES

セキュリティの要点

- ✓ GitHub アプリは「 Only select repositories 」で対象リポジトリのみに限定する
- ✓ 機密値はソースコードや wrangler.jsonc に書かず、 wrangler secret put で登録する
- ✓ .dev.vars / .env は .gitignore に追加し、コミットしない
- ✓ 本番・ステージング・開発で値を分ける場合は環境ごとに Secrets を分離する

APPENDIX

代替方式： GitHub Actions

CI でテストや Lint を挟みたい場合の代替。API トークンは GitHub Secrets に登録する。

.github/workflows/deploy.yml

```yaml
on:
  push:
    branches: [main]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v6
      - uses: cloudflare/wrangler-action@v3
        with:
          apiToken: ${{ secrets.CF_API_TOKEN }}
```

トークンは直書きせず GitHub Secrets へ。スコープは必要な権限に限定して発行する。

まとめ

作成 → push → 連携 → 設定 → ドメイン。
あとは push するだけで本番反映。