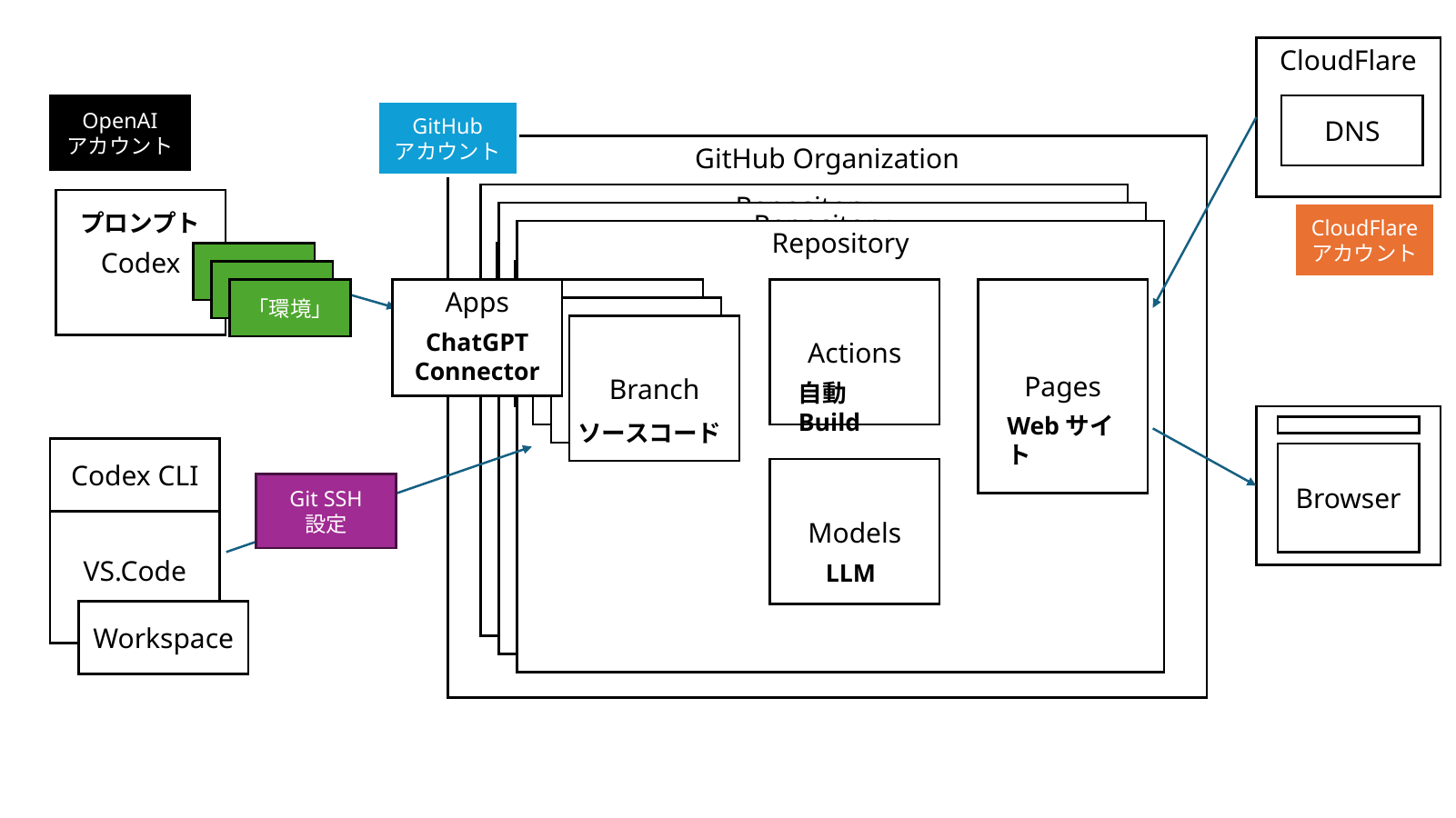
# GitHubとCodexを使ったWebサイト公開手順書

この手順書は、**IT 初心者でも実行できるように、丁寧に背景と具体的な操作を示します**。対象者は Git やターミナルを使ったことがないものの、Excel マクロ程度のITスキルを持つ方を想定しています。ChatGPT Plus アカウントと Web 版の Codex を使い、GitHub でソースコード管理しながら GitHub Pages で静的サイトを公開し、Cloudflare 経由で独自ドメインを割り当てるまでの流れを説明します。以下の図は全体像を表しています。黒・青・緑・紫・オレンジの四角がそれぞれ OpenAI アカウント、GitHub アカウント、Codex 環境、Git SSH 設定、Cloudflare アカウントを示しており、矢印はデータの流れを示しています。



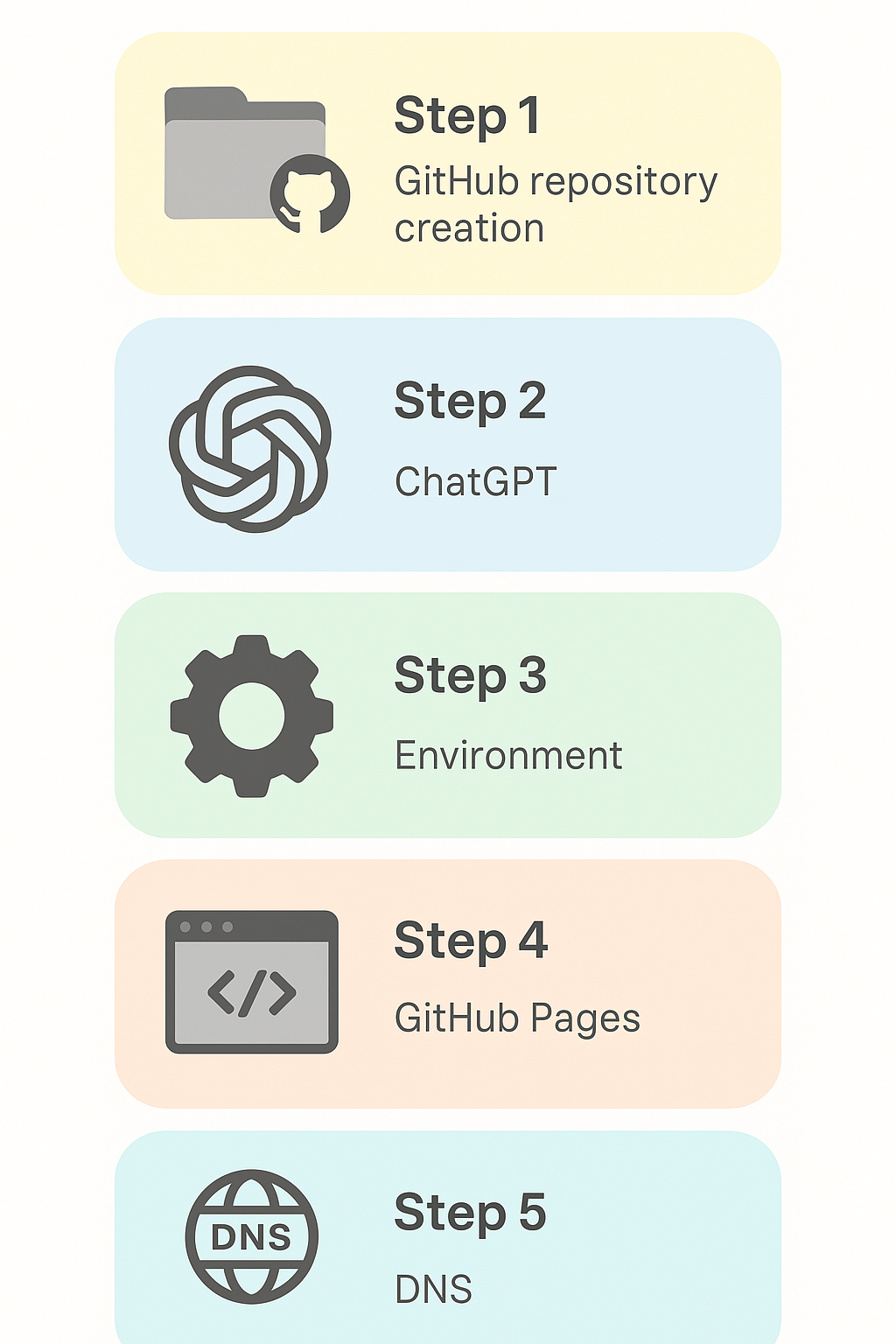
## 用語の解説

以下の用語が頻出します。用語の意味を理解すると手順の背景がわかりやすくなります。

| 用語 | 意味 |
| --- | --- |
| **Codex** | OpenAI 製のコード生成エージェントです。GitHub 上のソースコードを編集するためのツールで、ChatGPT Plus の「Web 版 Codex」を介して利用します。 |
| **環境** | Codex と GitHub を接続する設定のことを指します。Codex は GitHub リポジトリに対する読み書き権限を持つため、対象リポジトリごとに「環境」を作成します。 |
| **Codex CLI** | ローカルにあるソースコードをクラウド上の LLM で編集するためのコマンドラインツールです。本手順書では Web 版のみを使用します。 |
| **Repository** | ソースコード管理の単位です。GitHub ではプロジェクトごとにリポジトリを作り、ファイルや履歴を管理します。 |
| **Branch** | ソースコードのバージョン管理の枝です。新しい機能を追加したり修正を試すときに作ります。 |
| **Actions** | GitHub 内部で Docker を用いて自動実行される処理エンジンです。テストやビルド、デプロイを自動化できます。 |
| **Models** | GitHub 内部に配置された LLM 提供エンジンです。Codex を拡張したり、特定用途に合わせて調整できます。 |
| **Pages** | GitHub Pages の略です。GitHub リポジトリの内容から無料で静的な Web サイトを公開できる機能です。 |

## 作業全体の流れ

下の図は本手順書の全体的な作業フローをイメージ化したものです。各ステップの詳細は後述しますが、全体像を頭に入れておくと混乱しにくくなります。

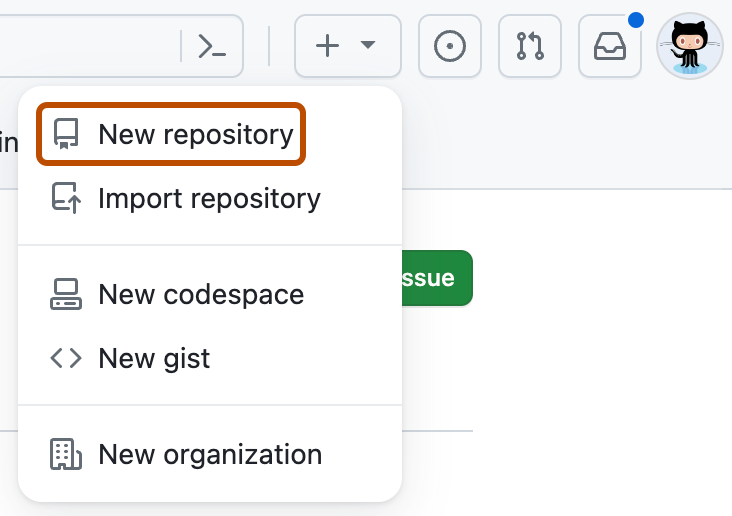


## ステップ1 – 公開用リポジトリの作成と初期化

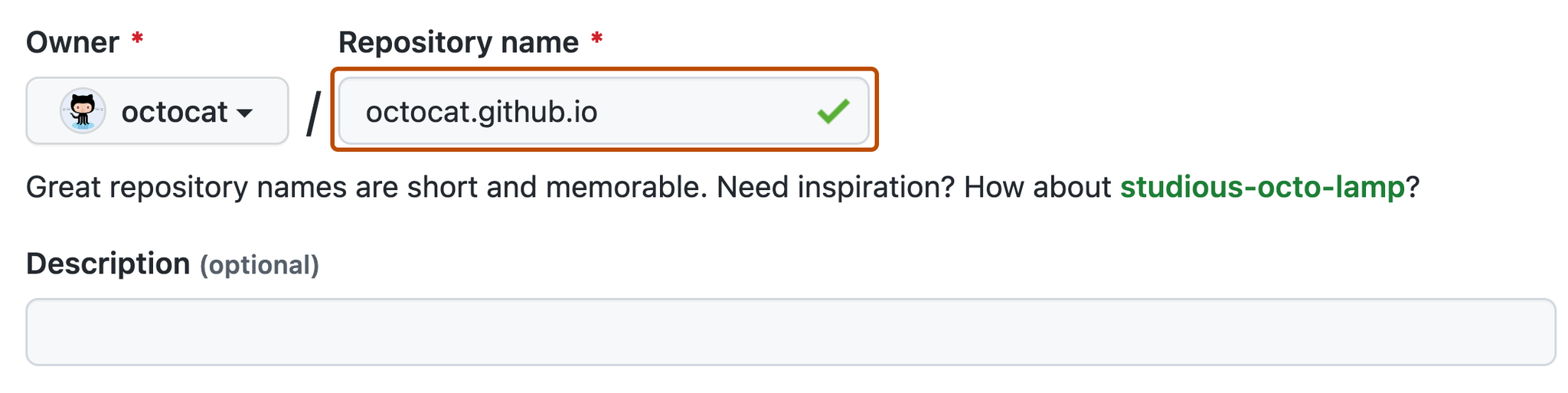
**目的：** 公開したいサイトのソースコードを保存する空のリポジトリを GitHub で作成します。GitHub Pages では **空のリポジトリを認識できないため、README.md など1ファイルを追加し初期化**しておく必要があります。また、このリポジトリが Codex の編集対象になります。

1. ブラウザで [GitHub](https://github.com/) にログインします。まだアカウントがない場合は無料で作成できます。
2. 画面右上の「＋」メニューから **“New repository”** を選択します。

*視覚サポート：* 以下のスクリーンショットは GitHub の「＋」メニューを開いた様子です。オレンジ色で **New repository** がハイライトされています。実際の画面と見比べながら操作してください。

  
3. **Repository name** にサイト名を入力し、アクセス権（Public/Private）を選びます。GitHub Pages はパブリックでもプライベートでも利用可能です。

*視覚サポート：* 下図はリポジトリ作成フォームの例です。Repository name フィールドに任意の名前を入力し、リポジトリの公開/非公開を選択します。この画面で README を追加するチェックボックスも確認できます。

  
4. *Initialize this repository with:* のチェックボックスで **“Add a README file”** をオンにします。これにより空のリポジトリにならず、後で Codex が認識できます。 5. 必要に応じて .gitignore やライセンスを追加し、[Create repository]ボタンをクリックしてリポジトリを作成します。

**技術的背景：** GitHub Pages はリポジトリの内容を元に静的サイトを生成しますが、空のリポジトリでは対象ファイルを検出できないためサイトを構築できません。README.md を入れておくことで Git の履歴が1コミット存在し、Pages の設定や Codex の操作が可能になります。また README はサイトの説明ページとしても機能します。

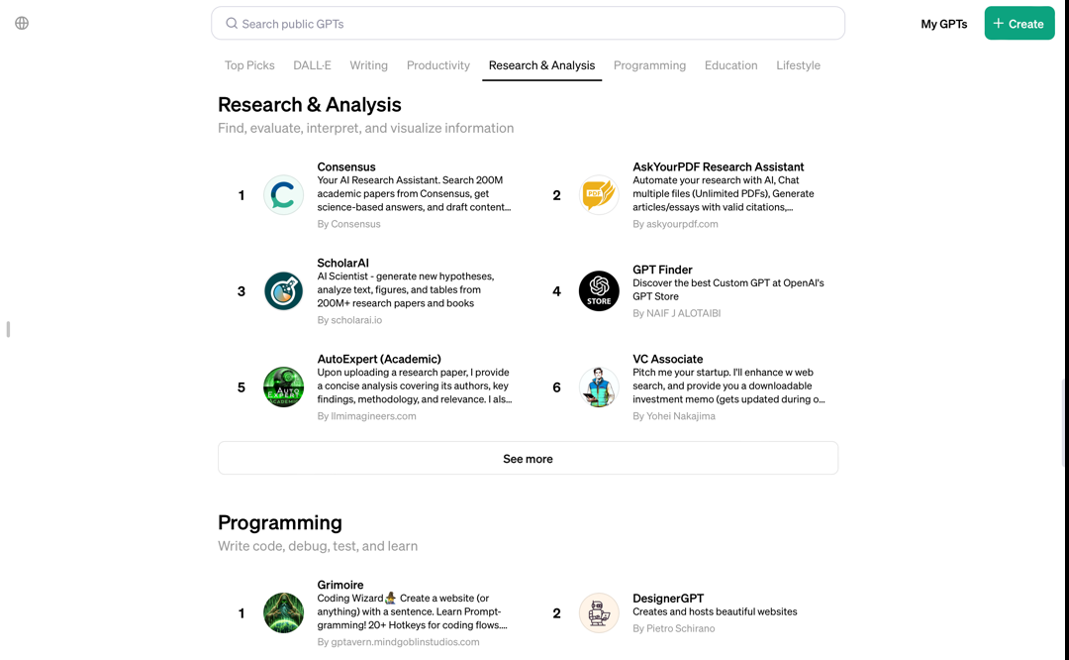
## ステップ2 – ChatGPT から Codex を開く

**目的：** ChatGPT Plus から Web 版 Codex を見つけ、起動する手順を分かりやすく示します。Codex は ChatGPT の拡張機能として提供され、リポジトリを編集するための AI 開発環境です。

このステップでは、一連の操作を4つに分解して説明します。手順ごとに「なぜその操作が必要なのか」を補足し、実際の画面イメージも挿入します。

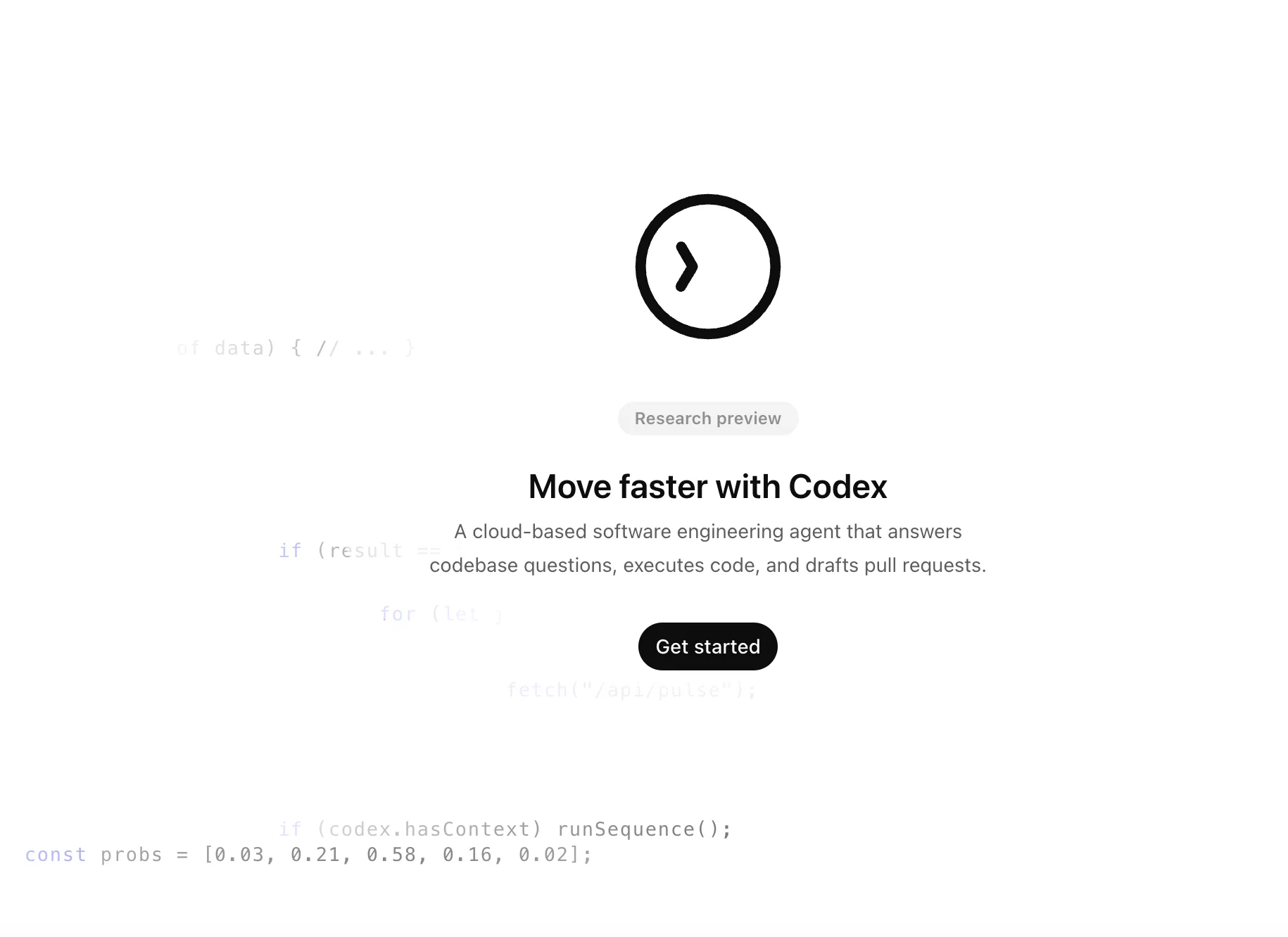
1. **ChatGPT にログインする** – [ChatGPT](https://chat.openai.com/) にアクセスし、Plus アカウントでサインインします。左側のサイドバーが表示され、メニューの最下部付近に **Explore GPTs（GPT を探す）** または **Explore** リンクがあります。これはカスタム GPT 一覧に移動するための入り口です。
2. **「Explore GPTs」メニューを開いてストアを眺める** – サイドバーの **Explore GPTs** をクリックすると、OpenAI が公開しているカスタム GPT を検索・閲覧できるストアが表示されます。このストアには **Search public GPTs** と書かれた検索バーと、トピック別タブ（Top Picks, DALLE, Writing, Productivity, Research & Analysis など）が並んでおり、目的の GPT を探すことができます。

*視覚サポート：* 下図は GPT ストアの実際の画面です。上部中央に検索バーがあり、その下に **Top Picks** や **Research & Analysis** などのカテゴリが表示されています。この画面で検索バーに「Codex」や「Coding Agent」と入力すると、Codex に関連する GPT がリストアップされます。



1. **Codex を検索して選択する** – 検索バーに「Codex」と入力すると、検索結果として **“Codex”**（場合によっては「Vibe Coding Agent」等と表記されていることもあります）が表示されます。カードをクリックすると詳細ページが開きます。[Use GPT] もしくは [Try Codex] ボタンをクリックして起動します。これにより Codex の起動画面が新しいタブで開きます。

*視覚サポート：* 下図は Codex の紹介画面の一例で、「Move faster with Codex」と記載されたカードが表示されています。ここで **Get started** ボタンを押すと Codex のセットアップフローが始まります。



1. **初回ガイドを読み GitHub 連携準備に進む** – 初めて Codex を起動すると、概要説明や注意事項が表示されます。内容を確認しながら [Next] や [Continue] をクリックします。最後に「GitHub と連携しますか？」といった質問が表示されるので、そのまま次のステップに進みます。

**技術的背景：** Codex は LLM を活用したコードアシスタントであり、ChatGPT の拡張機能として動作します。ユーザーは自然言語の指示を通じてソースコードへの変更を依頼し、Codex が GitHub リポジトリに対して提案・コミットを行います。このステップで Codex を起動しておくことにより、以後の作業全体を Web 上で完結できます。

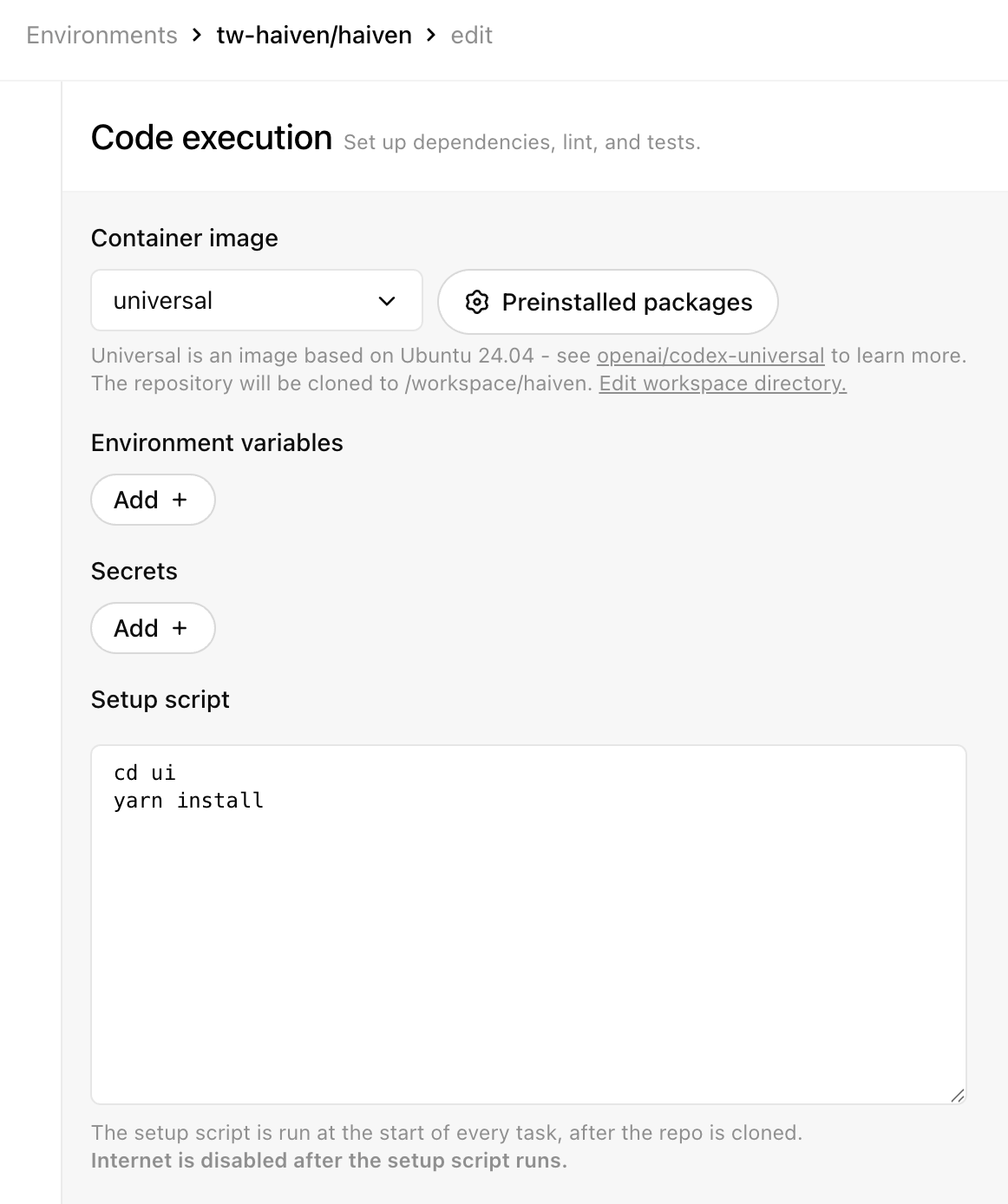
## ステップ3 – Codex で「環境」を作成し GitHub と接続

**目的：** Codex と GitHub を連携させるために「環境」を新規作成し、リポジトリへのアクセス権を付与します。この設定により、Codex が特定のリポジトリに対して読み書きできるようになります。

以下の作業は複数の画面を経由するため、順に操作していくと混乱しません。画面遷移ごとに役割を理解しながら進めてください。

1. **Codex から環境作成ページへ移動する** – Codex の上部メニューに **“Settings”** または **“Environments”** と表示されたリンクがあります。これをクリックすると既存環境の一覧ページに移動します。ここで [Add environment] や [Create new environment] ボタンを選び、新しい環境の作成フォームを開きます。
2. **環境名を入力しリポジトリを選ぶ** – フォームでは環境名を入力する欄があり、名称に手順1で作成したリポジトリ名を付けると分かりやすくなります。続いて **“Connect GitHub”** というボタンをクリックして GitHub との連携を開始します。この操作により別タブで GitHub の認証画面が開きます。
3. **GitHub でアプリをインストールして連携する** – GitHub の画面では OpenAI の **“ChatGPT Connector”**（あるいは **“Codex Connector”** と表記される場合もあります）というアプリのインストールページが表示されます。ここでインストール先のリポジトリとしてステップ1で作成した公開用リポジトリを選択し、**Install & Authorize** をクリックします。OAuth 認証により Codex に対象リポジトリの読み書き権限が付与されます。
4. **環境の詳細設定（コンテナ・環境変数・スクリプト）を行う** – 認証が完了すると Codex に戻り、環境の詳細設定画面が表示されます。ここでは実行環境となる **コンテナイメージ**（デフォルトは universal）や、必要に応じて **環境変数** や **Secrets**（秘密情報）、**Setup script**（依存関係インストール用のスクリプト）を設定できます。

*視覚サポート：* 下図は Codex の環境設定画面の例です。左上でコンテナイメージ universal が選択されており、環境変数・Secrets・Setup script を登録できるフォームが見えます。Setup script には cd ui や yarn install などのコマンドを記述し、リポジトリのビルド手順を自動化します。



1. **環境を保存して有効化する** – すべての設定を確認したら [Save] または [Create environment] ボタンをクリックします。これで Codex の環境一覧に新しい環境が追加されます。リストから該当環境を選択し、[Activate] や [Use] ボタンで有効化すると、以後の作業でこの環境が使用されます。

**技術的背景：** Codex から GitHub にアクセスするためには、GitHub App を介した OAuth 認証が必要です。アプリのインストールにより、指定したリポジトリの読み取り・書き込み権限が Codex に付与されます。環境設定ではリポジトリをクローンするためのコンテナイメージや依存関係のインストール方法を指定でき、タスク実行時にこの環境が自動的に構築されます。複数のリポジトリや設定を管理する場合は環境を分けておくと便利です。

## ステップ4 – Codex に GitHub Pages 用の Web サイト作成を指示

**目的：** Codex に対して、GitHub Pages を用いた静的サイトの雛形を生成するよう依頼します。自然言語で指示を出すと、Codex がリポジトリに必要なファイルを追加し、Git のコミットを作成します。

1. Codex のプロンプト（チャット入力欄）に次のような日本語の指示を入力します。

新しく作成したリポジトリに GitHub Pages 用の静的サイトの雛形を追加してください。日本語のホームページ（index.html）とスタイルを含み、GitHub Actions ワークフローもセットアップしてください。

2. Codex はリポジトリ内容を読み取り、index.html や styles.css、必要なワークフロー（main.yml など）を生成・コミットする提案を返します。内容を確認し、必要に応じて追加の指示を出して調整します。 3. “Apply changes” や “Accept” ボタンをクリックすると Codex が GitHub にコミットを送り、Pull request を作成します。Codex の画面で差分が確認できるので、内容を確認してマージします。

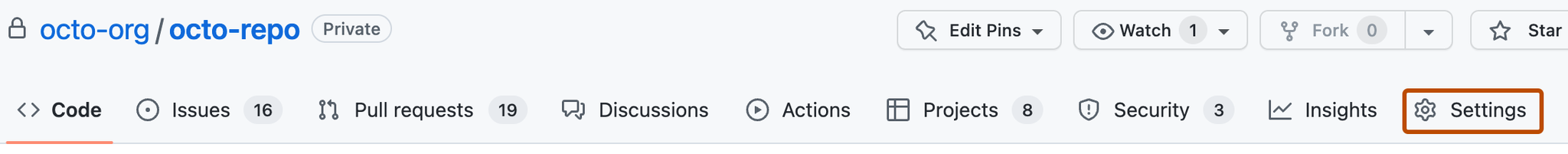
**技術的背景：** Codex はリポジトリに対して直接コミットする権限を持っており、ユーザーの同意の下でファイルを追加・編集します。GitHub Pages サイトが正常に表示されるには、index.html というファイル名でトップページを置く必要がある点に注意してください。また GitHub Actions を使った自動ビルド／デプロイ設定を併用すると、静的サイトジェネレータ（Astro 等）を使用する際に便利です。

## ステップ5 – GitHub で Pages と独自ドメインを設定

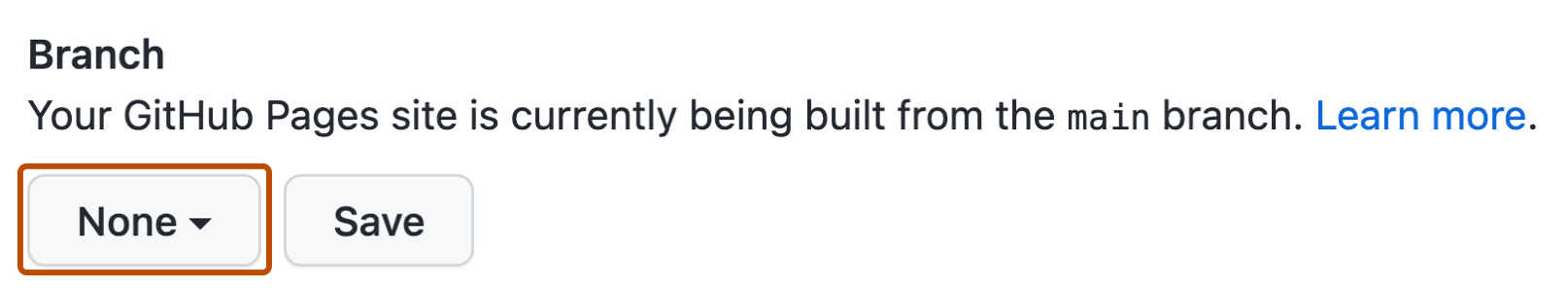
**目的：** Codex が生成したサイトを公開するために、GitHub Pages の設定を有効にし、Cloudflare で取得した独自ドメインを関連付けます。

1. GitHub で対象のリポジトリを開き、リポジトリ上部の **Settings** タブをクリックします。メニューが隠れている場合は ... ドロップダウンから選びます。

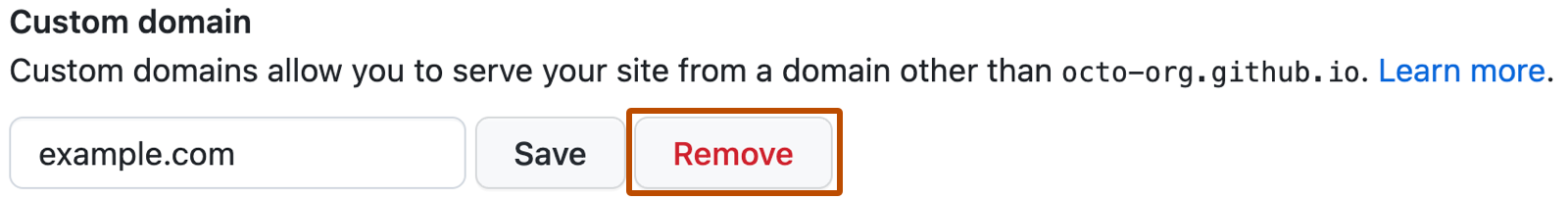
*視覚サポート：* 下図は GitHub リポジトリの上部メニューで **Settings** タブがハイライトされている例です。実際の画面と照合しながらクリックしてください。

  
2. 左側の **“Code and automation”** セクションにある **Pages** を選択します。 3. **“Branch”** と **“Folder”** を設定します。通常は main ブランチのルート (/) を指定します。設定を保存すると GitHub がサイト用の URL（https://ユーザー名.github.io/リポジトリ名/）を発行します。

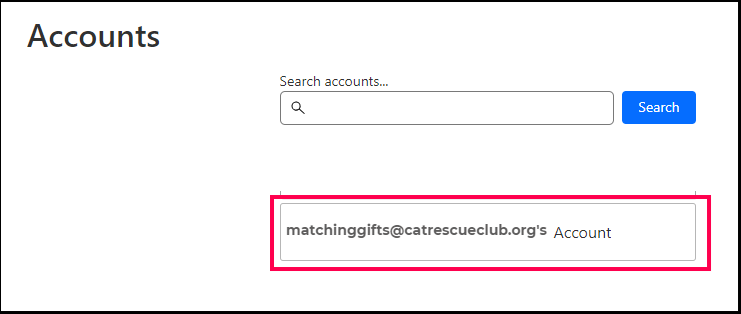
*視覚サポート：* GitHub Pages 設定画面の Branch ドロップダウンと Save ボタンの例です。ブランチ選択欄が None の場合は main を選択し、下の Save ボタンを押してください。

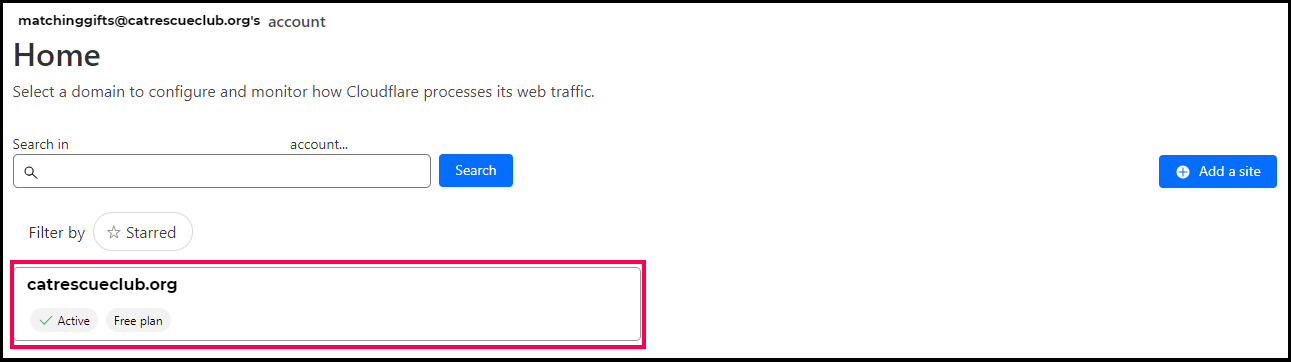
  
4. **Custom domain** 欄に使用したい独自ドメイン（例：example.com）を入力し **Save** をクリックします。この操作は [GitHub Docs にも記載されている通り][[1]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=1,your%20site%27s%20repository)、ソースブランチに CNAME ファイルを追加します。

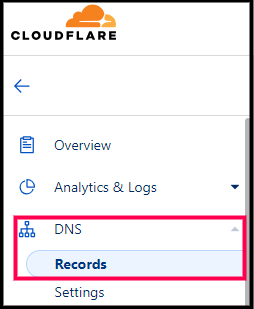
*視覚サポート：* 下図は Custom domain フィールドに独自ドメインを入力し、Save をクリックする画面例です。既にドメインが設定されている場合は [Remove] ボタンも表示されます。

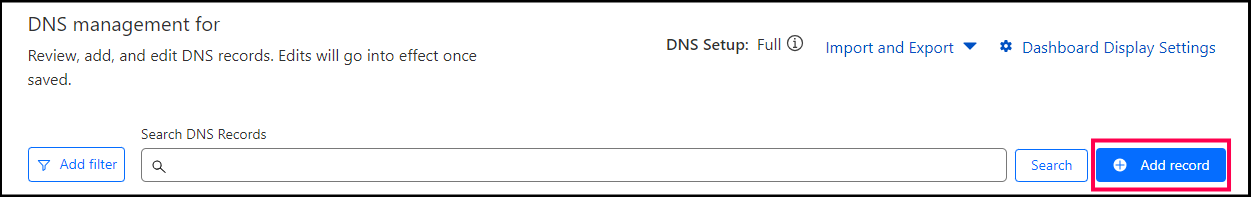
  
5. 別タブで [Cloudflare ダッシュボード](https://dash.cloudflare.com/) にログインし、対象のドメインの DNS 設定画面を開きます。GitHub Pages 用に次のレコードを作成します。

*視覚サポート：* Cloudflare ダッシュボードのホーム画面やドメイン選択画面、DNS 設定画面、レコード追加ボタンは以下のスクリーンショットを参考にしてください。







  
- **Aレコード**（ルートドメインの場合）：ホスト名に @ を指定し、IPv4 アドレス 185.199.108.153 185.199.109.153 185.199.110.153 185.199.111.153 をそれぞれ設定します。これは GitHub Docs の指示[[2]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=5,see%20your%20DNS%20provider%27s%20documentation)に基づきます。 - **CNAME レコード**（サブドメインの場合）：ホスト名に www など希望するサブドメインを指定し、値に ユーザー名.github.io を設定します。これにより www.example.com が GitHub Pages のデフォルトドメインへ転送されます[[3]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=Navigate%20to%20your%20DNS%20provider,without%20including%20the%20repository%20name)。 6. Cloudflare 側でネームサーバーを変更済みであれば、DNSレコードが反映されるまで最大24時間待ちます。[[4]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=Note) 7. GitHub に戻り、Pages 設定画面の「Enforce HTTPS」を有効にすると、SSL 証明書が自動で発行され HTTPS で安全にアクセスできるようになります。反映までしばらく時間がかかることがあります[[5]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=4,GitHub%20Pages%20site%20with%20HTTPS)。

**技術的背景：** 独自ドメインを GitHub Pages に割り当てる際は、GitHub 側と DNS 側の両方で設定が必要です。GitHub では CNAME ファイルを追加することで「このリポジトリがこのドメインを利用する」ことを示し、DNS ではそのドメインが GitHub Pages のサーバーを指すようにレコードを設定します。Cloudflare を使うことでネームサーバーの管理や SSL 証明書の自動更新を行えます。

### ドメインの所有者確認

GitHub Pages では第三者によるドメイン乗っ取りを防ぐため、ドメインの所有者を検証できます。GitHub Docs の手順[[6]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=1.%20In%20the%20upper,picture%2C%20then%20click%20%20Settings)を参考に、ドメインに TXT レコードを追加します。

1. GitHub 右上のプロフィールアイコンから **Settings** を開き、左側メニューの **Pages** を選びます。
2. **Add a domain** をクリックし、検証したいドメインを入力します。[[7]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=1.%20In%20the%20upper,picture%2C%20then%20click%20%20Settings)
3. GitHub が表示する TXT レコードの値（\_github-pages-challenge-USERNAME.example.com など）をコピーし、Cloudflare の DNS 設定に TXT レコードとして追加します。[[8]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=5,with%20your%20domain%20hosting%20service)
4. DNS 変更が反映されるまで最大24時間待ちます。[[9]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=6,your%20DNS%20configuration%20by%20running) 反映後、GitHub の Pages 設定で **Continue verifying** → **Verify** をクリックするとドメインが確認済みになります。[[10]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=7,and%20then%20click%20Continue%20verifying)
5. 検証が完了したら、TXT レコードは削除せず残しておきます[[11]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=9,in%20your%20domain%27s%20DNS%20configuration)。削除すると再検証が必要になります。

## ステップ6 – Astro などの静的サイトジェネレータを導入

**目的：** シンプルな HTML ファイルだけでなく、最近は Astro や Next.js などの静的サイトジェネレータ(SSG)を使うと、コンポーネントベースでメンテナンスしやすい Web サイトを作れます。ここでは Astro を例に、Codex と GitHub Actions を使ってビルドとデプロイを自動化する方法を示します。

1. Codex のチャットに次のような指示を入力します。

Astro を使ったブログサイトの雛形をこのリポジトリに追加し、GitHub Actions を使って `npm install` と `npm run build` を行い、ビルド成果物を `dist/` に出力して GitHub Pages で公開できるようにしてください。

2. Codex は package.json、Astro の設定ファイル、サンプルページ、astro.config.mjs などを作成し、.github/workflows/deploy.yml に自動ビルド用のワークフローを追加します。内容を確認して必要に応じて修正します。 3. GitHub Actions のワークフローでは、push や PR マージ時に自動で npm ci、npm run build を実行し、dist/ ディレクトリの内容を gh-pages ブランチまたは docs フォルダに配置して GitHub Pages にデプロイします。ワークフローがエラーなく完了するとサイトが更新されます。 4. ビルド時に Node.js のバージョンや依存パッケージが原因でエラーが発生する場合は、Codex に依頼して volta のインストールやキャッシュ設定を行うよう修正させます。また、動作検証用にローカル環境が必要な場合は、VS Code ではなく GitHub Codespaces を利用する方法もあります。

**技術的背景：** Astro は静的サイトジェネレータの一種で、コンポーネント指向・軽量・高速なビルドが特徴です。GitHub Actions により、ソースコードが更新されるたびに自動でビルドとデプロイが行われるため、コマンドラインを触れなくても継続的な運用が可能です。その他のフレームワーク（Next.js, Hugo, Jekyll など）でも同様のワークフローを利用できます。

## まとめとベストプラクティス

1. **少しずつ進める** – 一度に多くの処理を依頼すると Codex が誤解することがあります。小さなタスクに分けて指示を与えましょう。
2. **コミット内容を確認する** – Codex が生成した Pull Request をマージする前に、ファイルの変更内容や削除されたファイルを確認します。必要であればコメントで修正指示を出せます。
3. **ドメイン設定を丁寧に行う** – Cloudflare で DNS レコードを設定する際は、GitHub Docs の指示[[1]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=1,your%20site%27s%20repository)をよく読み、Aレコードと CNAME の違いを理解して設定します。不完全な DNS 設定はサイトの表示不良の原因となります。
4. **セキュリティを意識する** – Enforce HTTPS を有効にし、ドメイン検証を行うことで、第三者による乗っ取りや中間者攻撃を防止できます[[5]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=4,GitHub%20Pages%20site%20with%20HTTPS)。
5. **学習を続ける** – Codex を使いこなすには継続的な試行錯誤が必要です。GitHub Pages や Cloudflare、Astro などの公式ドキュメントを参照しながら、少しずつステップアップしていきましょう。

この手順書があれば、コマンドラインに不慣れな担当者でも ChatGPT と Codex を活用して Web サイトを構築・公開できるはずです。質問やエラーが出た場合は、Codex や ChatGPT に日本語で質問すれば助言をもらえます。安全かつ効率的に作業を進めてください。

[[1]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=1,your%20site%27s%20repository) [[2]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=5,see%20your%20DNS%20provider%27s%20documentation) [[3]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=Navigate%20to%20your%20DNS%20provider,without%20including%20the%20repository%20name) [[4]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=Note) [[5]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site#:~:text=4,GitHub%20Pages%20site%20with%20HTTPS) Managing a custom domain for your GitHub Pages site - GitHub Docs

<https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/managing-a-custom-domain-for-your-github-pages-site>

[[6]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=1.%20In%20the%20upper,picture%2C%20then%20click%20%20Settings) [[7]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=1.%20In%20the%20upper,picture%2C%20then%20click%20%20Settings) [[8]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=5,with%20your%20domain%20hosting%20service) [[9]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=6,your%20DNS%20configuration%20by%20running) [[10]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=7,and%20then%20click%20Continue%20verifying) [[11]](https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages#:~:text=9,in%20your%20domain%27s%20DNS%20configuration) Verifying your custom domain for GitHub Pages - GitHub Docs

<https://docs.github.com/en/pages/configuring-a-custom-domain-for-your-github-pages-site/verifying-your-custom-domain-for-github-pages>