

Webサイト高速化のための
**静的サイトジェネレーター
活用入門**

GatsbyJSで実現する高速&実用的なサイト構築



副読本

セットアップPDF

Build blazing-fast websites with GatsbyJS



エピスコム 編著



本 PDF は下記書籍の副読本です。

PDF は GitHub (<https://github.com/ebisucom/gatsbyjs-book/>) で配布しています。



Webサイト高速化のための 静的サイトジェネレーター活用入門

GatsbyJSで実現する高速＆実用的なサイト構築

- ↗ <https://book.mynavi.jp/ec/products/detail/id=115483>
- ↗ <https://ebisu.com/gatsbyjs-book/>
- ↗ <https://amzn.to/2x5nuyq>

- ・本書に記載された内容は、情報の提供のみを目的としております。したがって、本書を用いての運用はすべてお客様自身の責任と判断において行ってください。
- ・本書の制作にあたっては正確な記述につとめましたが、著者や出版社のいずれも、本書の内容に関してなんらかの保証をするものではなく、内容に関するいかなる運用結果についてもいっさいの責任を負いません。あらかじめご了承ください。
- ・本書中に掲載している画面イメージなどは、特定の設定に基づいた環境にて再現される一例です。ハードウェアやソフトウェアの環境によっては、必ずしも本書通りの画面にならないことがあります。あらかじめご了承ください。
- ・本書は 2020 年 4 月段階での情報に基づいて執筆されています。本書に登場するソフトウェアのバージョン、URL、製品のスペックなどの情報は、すべてその原稿執筆時点でのものです。執筆以降に変更されている可能性がありますので、ご了承ください。
- ・本書中に登場する会社名および商品名は、該当する各社の商標または登録商標です。本書では ® および TM マークは省略させていただいております。

CONTENTS

もくじ

SETUP

1

開発環境の準備

6

STEP 1-1	Linux & WSL	7
└	nvm のインストール.....	7
└	node.js のインストール.....	9
└	yarn のインストール.....	10
STEP 1-2	Windows10.....	12
└	Windows 環境にインストールする前に.....	12
└	nvm のインストール.....	12
└	node.js のインストール.....	14
└	yarn のインストール.....	15
└	git のインストール.....	17
STEP 1-3	macOS	19
└	nvm のインストール.....	19
└	node.js のインストール.....	21
└	yarn のインストール.....	22

SETUP

2

アカウントの準備

24

STEP 2-1	3 つのサービスの確認	25
STEP 2-2	GitHub	27
STEP 2-3	Netlify	29
STEP 2-4	Contentful	30

SETUP
3 サイトの公開 **32**

STEP 3-1	サイトの公開・更新環境の設定	33
STEP 3-2	GitHub の設定	34
	リポジトリを用意する	34
	リポジトリへプッシュする	35
	リポジトリを確認する	36
STEP 3-3	Netlify の設定とサイトの公開	37
	サイトを公開する	37
	環境変数の追加設定	39
	サブドメインの変更	40
	COLUMN アクセスを許可するリポジトリの設定	41
STEP 3-4	サイトの更新	42
STEP 3-5	Contentful によるサイトの更新	44
	Netlify の設定	44
	Contentful の設定	45
	記事の投稿	47
	COLUMN Netlify 側でデプロイを実行する	48

SETUP
4 Contentfulによるコンテンツの管理 **49**

STEP 4-1	Contentful によるコンテンツの管理	50
STEP 4-2	スペースの作成	51
	スペースを作成する	51
	ロケールを設定する	52

STEP 4-3	コンテンツを管理できるようにする	54
①	コンテンツタイプを作成する	55
②	タイトルのフィールドを作成する	56
③	スラッグのフィールドを作成する	58
④	投稿日のフィールドを作成する	59
⑤	コンテンツのフィールドを作成する	60
⑥	アイキャッチ画像のフィールドを作成する	61
STEP 4-4	カテゴリーを管理できるようにする	63
①	カテゴリー用のコンテンツタイプを作成する	64
②	カテゴリー名のフィールドを作成する	65
③	カテゴリー・スラッグのフィールドを作成する	66
④	BlogPost コンテンツタイプにカテゴリーのフィールドを追加する	67
STEP 4-5	記事を投稿する	69
■	カテゴリーを用意する	69
■	記事を投稿する	71
COLUMN	画像の管理	72
COLUMN	画像でエラーを回避する	72
STEP 4-6	トークンの確認	73
■	スペース ID とアクセストークンを確認する	73
■	パーソナルアクセストークンを作成する	74
STEP 4-7	Contentful のデータをインポートする	75
COLUMN	contentful-cli をインストールして処理する場合	77
COLUMN	画像のアップロードがタイムアウトした場合	77



SETUP

1

開発環境の準備

ここで紹介しているのはインストール方法の一例です。

1-1 **Linux & WSL**

1-2 **Windows10**

1-3 **macOS**

Build blazing-fast websites with GatsbyJS

GatsbyJS

1-1 Linux & WSL

ここでは、Ubuntu 18.04 LTS を想定して進めていきます。まず、

```
$ git --version
```

で、git がインストールされていることを確認します。

ユーザー名と Email アドレスが設定できていない場合には、ここで設定しておきます。

これは、git が個人を識別するために利用されます。以下のように、名前とメールアドレスを設定します。

メールアドレスには、SETUP 2 で GitHub のアカウントを作成する際に登録するものを使用します。

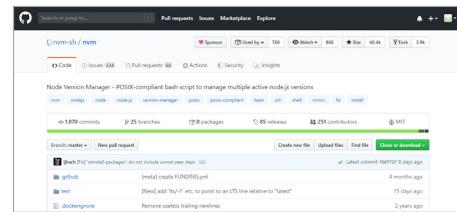
```
$ git config --global user.name "name"  
$ git config --global user.email "moniker@example.com"
```

nvmのインストール

Node.js を直接インストールすることも可能ですが、複数のバージョンを使い分けるケースも少なくありません。そこで、バージョン管理ツールを通してインストールすることをおすすめします。ここでは、nvm (Node Version Manager) を使ってインストールを進めます。

nvm の GitHub (<https://github.com/nvm-sh/nvm>) へアクセスします。

最新の「Install & Update Script」を確認して、コピー＆ペーストで実行します。



nvmのGitHub
<https://github.com/nvm-sh/nvm>

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.2/install.sh | bash
```

この部分が、その時点の最新のものに変わるので注意。

インストールが完了したら、ターミナルを再起動するか

```
$ source ~/.bashrc
```

などで、設定を読み直したうえで、次のように入力します。

```
$ command -v nvm
```

「nvm」と表示されればインストールは完了です。

node.jsのインストール

続いて、node.js をインストールします。まずは、

```
$ nvm ls
```

で、ローカルにインストールされているものを確認し、

```
$ nvm ls-remote
```

で、インストール可能なものを確認できます。

バージョンを指定して Node.js をインストールすることができますが、ここでは以下のコマンドで LTS (Long Term Support) の最新版をインストールします。

```
$ nvm install --lts --latest-npm
```

インストールが完了したら、

```
$ nvm ls
```

ローカルにインストールされているバージョンを確認します。

```
$ node -v
```

で、バージョンが確認できれば完了です。

yarnのインストール

最後に、パッケージマネージャーの yarn もインストールします。

yarn の公式のインストール方法は、以下のとおりです。

```
$ curl -o- -L https://yarnpkg.com/install.sh | bash
```

インストールが完了したら、再び

```
$ source ~/.bashrc
```

で、設定を読み込みなおし

```
$ yarn -v
```

で、バージョンを確認します。

npm を利用して、

```
$ npm install -g yarn
```

として、インストールすることも可能です。どちらでインストールしても問題はありませんが、インストールされる場所が異なりますので、注意が必要です。

```
$ which yarn
~/.nvm/versions/node/v12.16.1/bin/yarn
```

npmでインストールした場合。

```
$ which yarn
~/.yarn/bin/yarn
```

公式のスクリプトでインストールした場合。

WSLの場合

WSL の場合には、最後に以下のコマンドでライブラリをインストールしておきます。

これは、Gatsby を使う際に共有ライブラリとして「libGL.so.1」と「libXi.so.6」が見つからないというエラーが出るのを回避するためです。

```
$ sudo apt update
$ sudo apt install -y libgl1-mesa-glx libxi6
```

1-2 Windows10

Windows環境にインストールする前に

Windows 上にも gatsby を使うための環境を構築することは可能です。ただし、公式でも右のようなページが用意されているとおり、なかなか難しい環境であることは間違いないありません。

Windows10 では、WSL (Windows Subsystem for Linux) という便利な環境が用意されていますので、こちらを利用し、Linux と同様の設定をすることをおすすめします。

どうしても Windows 上に直接インストールしたい場合には、以下の方法でインストールすることができます（ただし、クリーンな環境にインストールしています）。

Gatsby on Windows

Setting up your environment for building native Node.js modules.

If you install Node.js...
gatsby-plugin-sharp requires Node v10
gatsby-plugin-sharp requires libvips
Windows Subsystem for Linux

Setting up your environment for building native Node.js modules.

Many Gatsby plugins and themes require building native Node.js modules, e.g. Sharp (a common Gatsby dependency used for image processing). To do so, you need a functional build environment (Python and Visual C++ + Build Tools).

The recommended way to setup your build environment on Windows is to install the `windows-build-tools` package by running `npm install -g windows-build-tools` on an active Node.js environment. It depends on the `node-gyp` package, in turn depending on Visual C++ + Build Tools 2015, provided free of charge by Microsoft. These tools are required to compile popular native modules. It will also install Python 2.7, configuring you environment for building native modules.

If your `windows-build-tools` installation stalls after Visual Studio Build Tools finishes, this remedy might help.

Gatsby on Windows

<https://www.gatsbyjs.org/docs/gatsby-on-windows/>

nvmのインストール

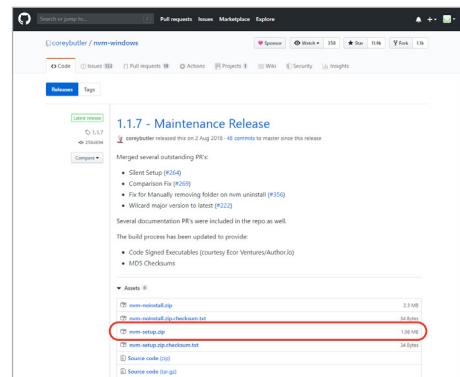
Node.js を直接インストールすることも可能ですが、複数のバージョンを使い分けるケースも少なくありません。

そこで、バージョン管理ツールを通してインストールすることをおすすめします。ここでは、nvm (Node Version Manager) を使ってインストールを進めます。

nvm-windows の GitHub へアクセスします。

Release の最新のものを確認し、Assets から nvm-setup.zip を選択してダウンロードします。

nvm-setup.zip を解凍し、nvm-setup.exe を実行してインストールを行います。



nvm-windows の GitHub

<https://github.com/coreybutler/nvm-windows/releases>

インストールが完了したら、PowerShell を起動し、

> nvm

と入力し、Enter を押します。

図のように表示されれば、nvm のインストールは完了です。

```
PS C:\$Users\$moniker> nvm
Running version 1.1.7.

Usage:
  nvm arch           : Show if node is running in 32 or 64 bit mode.
  nvm install <version> [arch] : The version can be a node.js version or "latest" for the latest stable version.
                               Optionally specify whether to install the 32 or 64 bit version (defaults to system arch).
                               .
                               Set [arch] to "all" to install 32 AND 64 bit versions.
                               Add --insecure to the end of this command to bypass SSL validation of the remote download.
  nvm server         : List the node.js installations. Type "available" at the end to see what can be installed.
  nvm list [available]
  nvm alias [name]    : Aliased as ls.
  nvm on             : Enable node.js version management.
  nvm off            : Disable node.js version management.
  nvm proxy [url]    : Set a proxy to use for downloads. Leave [url] blank to see the current proxy.
  nvm node_mirror [url]
  nvm default url   : Set the node mirror. Defaults to https://nodejs.org/dist/. Leave [url] blank to use default.
  nvm npm_mirror [url]
  nvm default url   : Set the npm mirror. Defaults to https://github.com/npm/cli/archive/. Leave [url] blank.
  nvm uninstall <version>
  nvm use [version] [arch]
  nvm root [path]
  nvm version

PS C:\$Users\$moniker> .
```

node.jsのインストール

続いて、node.js をインストールしていきます。

```
> nvm list available
```

インストール可能な node.js のバージョンが表示されますので、ここでは LTS (Long Term Support) の最新バージョンをインストールします。

```
> nvm install 12.16.1
```

そのときの最新版に読み替えてください。

インストールが完了したら、

```
> nvm use 12.16.1
```

で、インストールした node.js が使えるようになります。

```
> node -v  
> npm -v
```

をタイプして、動くことを確認してください。

yarnのインストール

最後に、パッケージマネージャーの yarn もインストールしておきます。

```
> npm install -g yarn
```

インストールが完了したら、

```
> yarn -v
```

で確認しておきます。ただし、環境によっては、

```
yarn : このシステムではスクリプトの実行が無効になっているため、ファイル C:\Program Files\nodejs\yarn.ps1 を読み込むことができません。詳細については、「about_Execution_Policies」 (https://go.microsoft.com/fwlink/?LinkID=135170) を参照してください。
発生場所 行 :1 文字 :1
+ yarn -v
+ ~~~~
+ CategoryInfo          : セキュリティ エラー: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

といったメッセージが表示され、yarn が実行されません。これは PowerShell の実行ポリシーによるもので、yarn の実行をブロックしています。

まずは、現在の実行ポリシーを確認します。

```
> PowerShell Get-ExecutionPolicy
```

「Restricted」という答えが返ってきます。ここでは、現在のユーザーの設定を変更することにして、次のコマンドを実行します。

```
> Set-ExecutionPolicy -Scope CurrentUser
```

すると、実行ポリシーを変更するためのパラメーターを求められますので

```
> RemoteSigned
```

と入力します。変更してもよいかと確認されますので、Yを入力します。

完了したら、再び、

```
> yarn -v
```

で、実行されることを確認してください。

さらに、管理者権限で PowerShell を起動しなおし、

```
> npm install --global windows-build-tools
```

または

```
> yarn global add windows-build-tools
```

で、windows-build-tools をインストールします。

Visual C++ Build Tools 2017 や Python 3.8 などがインストールされ、ビルド環境が整います。

ただし、環境によっては、うまくインストールできない場合があります。その場合は、Gatsby 公式のページ「Gatsby on Windows」(<https://www.gatsbyjs.org/docs/gatsby-on-windows/>) を参考にしてください。

gitのインストール

最後に、gitをインストールします。

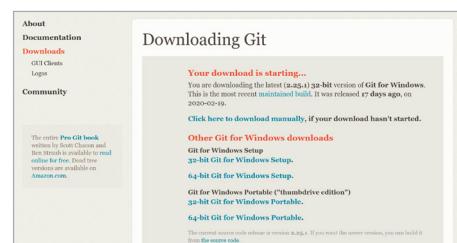
gitの公式ページへアクセスし、「Download ~ for Windows」をクリックします。



Git
<https://git-scm.com/>

ダウンロードページが開くと同時に、OSにあったバージョンがダウンロードされます。

ダウンロードされなかった場合には、OSにあったものを選択してください。



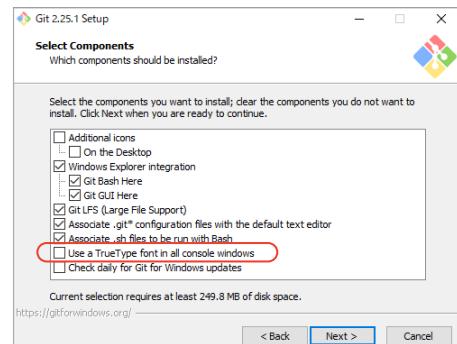
ダウンロードした「Git for Windows」を実行します。好みや環境に応じて設定していくますが、よくわからない場合には、デフォルトで問題ありません。ただし、次の2箇所は注意ポイントです。



ダウンロードした「Git for Windows」を実行し、設定していきます。

Select Components

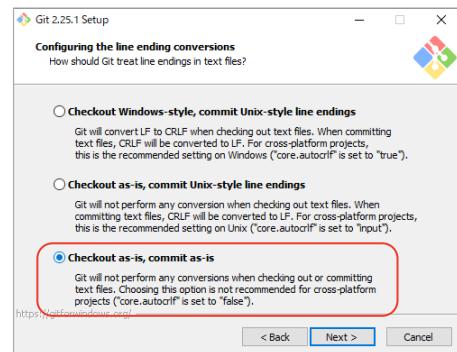
Select Componentsでは、「Use a TrueType font in all console windows」は選択しないことをおすすめします。コンソールウィンドウ用のフォントがインストールされますが、漢字などが文字化けします。



Configuring the line ending conversions

Configuring the line ending conversionsでは、一番下の「Checkout as-is, commit as-is」を選択することをおすすめします。

ここではリポジトリへのチェックイン・チェックアウト時の改行に関する設定をしていますが、一番下の何もしないという設定が一番問題がありません。



インストールが完了したら、git にユーザー名と Email アドレスを指定します。これは、git が個人を識別するために利用されます。

PowerShell を起動して、名前とメールアドレスを設定します。メールアドレスには、SETUP 2 で GitHub のアカウントを作成する際に登録するものを使用します。

```
> git config --global user.name "name"
> git config --global user.email "moniker@example.com"
```

以上で準備は完了です。

1-3 macOS



まずはターミナルを起動し、git がインストールされているかを確認します。

```
$ git -v
```

git がインストールされていない場合には、「"git" コマンドを実行するには ...」というダイアログが表示されるので、クリックしてインストールしてください。

インストールが完了したら、git にユーザー名と Email アドレスを設定します。これは、git が個人を識別するために利用されます。

ターミナルを起動して、名前とメールアドレスを設定します。メールアドレスには、SETUP 2 で GitHub のアカウントを作成する際に登録するものを使用します。

```
$ git config --global user.name "name"  
$ git config --global user.email "moniker@example.com"
```

nvmのインストール

Node.js を直接インストールすることも可能ですが、複数のバージョンを使い分けるケースも少なくありません。そこで、バージョン管理ツールを通してインストールすることをおすすめします。ここでは、nvm (Node Version Manager) を使ってインストールを進めます。

nvm のインストールスクリプトが、パスを書き込むファイルが必要だと言つてきますので、あらかじめ作成しておきます。

```
~/.bashrc
~/.bash_profile
~/.zshrc
~/.profile
```

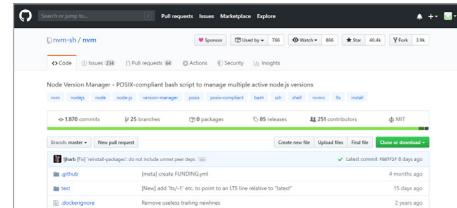
のいずれかが必要です。ここでは、Catalina の zsh 環境を想定し、

```
$ touch ~/.zshrc
```

で、.zshrc を作成しておきます。環境に合わせて作成してください。

続いて、nvm の GitHub へアクセスします。

最新の「Install & Update Script」を確認して、
コピー&ペーストで実行します。



nvmのGitHub

<https://github.com/nvm-sh/nvm>

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.2/install.sh | bash
```

この部分が、その時点の最新のものに変わるので注意。

インストールが完了したら、ターミナルを再起動するか

```
$ source ~/.zshrc
```

などで設定を読み込み直したうえで、次のように入力します。

```
$ command -v nvm
```

「nvm」と表示されればインストールは完了です。

node.jsのインストール

続いて、node.jsをインストールします。まずは、

```
$ nvm ls
```

で、ローカルにインストールされているものを確認し、

```
$ nvm ls-remote
```

で、インストール可能なものを確認できます。

バージョンを指定して、Node.jsをインストールすることができますが、ここでは、以下のコマンドで LTS (Long Term Support) の最新版をインストールします。

```
$ nvm install --lts --latest-npm
```

インストールが完了したら、

```
$ nvm ls
```

□一カルにインストールされているバージョンを確認します。

```
$ node -v
```

で、バージョンが確認できれば完了です。

yarnのインストール

最後に、パッケージマネージャーの yarn もインストールします。

yarn の公式のインストール方法は、以下のとおりです。

```
$ curl -o- -L https://yarnpkg.com/install.sh | bash
```

インストールが完了したら、再び

```
$ source ~/.zshrc
```

で、設定を読み込みなおし

```
$ yarn -v
```

で、バージョンを確認します。

npm を利用して、

```
$ npm install -g yarn
```

として、インストールすることも可能です。どちらでインストールしても問題はありませんが、インストールされる場所が異なりますので、注意が必要です。

```
$ which yarn  
/Users/moniker/.nvm/versions/node/v12.16.1/bin/yarn
```

npmでインストールした場合。

```
$ which yarn  
/Users/moniker/.yarn/bin/yarn
```

公式のスクリプトでインストールした場合。

SETUP

2

アカウントの準備

2-1 3つのサービスの確認

2-2 GitHub

2-3 Netlify

2-4 Contentful

Build blazing-fast websites with GatsbyJS

GatsbyJS

2-1 3つのサービスの確認



本書では、GitHub、Netlify、Contentful の 3 つのサービスを利用します。

いずれのサービスでも、**支払いの設定をすることなしにアカウントを作成できます**ので、安心して使い始めることができます。

ここでは、これらがどのようなサービスなのかを確認しておきます。

アカウントをお持ちでない方は、本書を読み始める前に各サービスのアカウントを作つておいてください。アカウントの作成方法については 2-2 以降を参照してください。

GitHub



GitHub は、ソフトウェアの開発のプラットフォームで、ソースコードをホスティングすることができます。コードのバージョン管理システムには、Git を採用しています。

個人向けには Free アカウントが用意されており、以下のような設定となっています。

- 無制限のパブリックリポジトリ
- 無制限のプライベートリポジトリ



GitHub
<https://github.co.jp/>

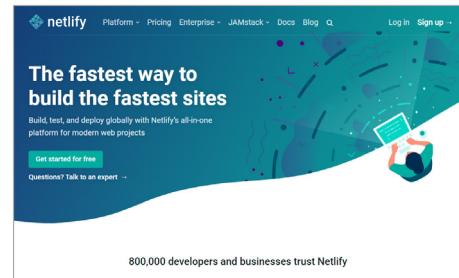
Netlify

Netlify は静的ウェブサイトのホスティングサービスと、サーバーレスのバックエンドサービスを提供しています。

Starter として無料のプランが用意されており、以下のような設定となっています。

- 転送量：100GB/月
- ビルド時間：300 分/月
- ストレージ：100GB
- API リクエスト：500 リクエスト/分
- 3 デプロイ/分

もちろん、SSL にも対応しています。ちょっとしたサイトであれば、無料枠を使い切ることはないでしょう。



Netlify

<https://www.netlify.com/>

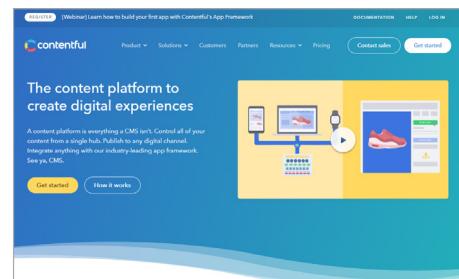
Contentful



ヘッドレス CMS サービスを提供しています。無料のプランが用意されており、

- 5000 レコード
- 24 コンテンツタイプ
- 2 スペース
- 2 つのロケール

といった設定になっています。記事を 5 件、それぞれの記事に 1 つずつのアセット(画像)がある場合には、全部で 10 レコードを消費するといった具合です。ブログなどに使っても、かなりの記事をポストできる環境が提供されています。



Contentful

<https://www.contentful.com/>

2-2 GitHub

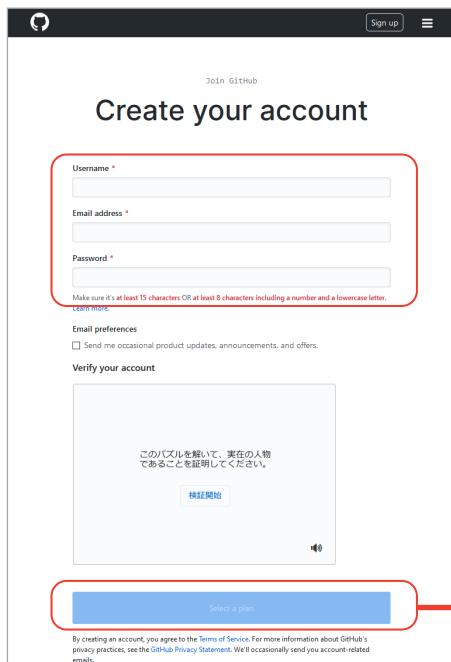


①



GitHub のページにアクセスし、「GitHub に登録する」をクリックします。

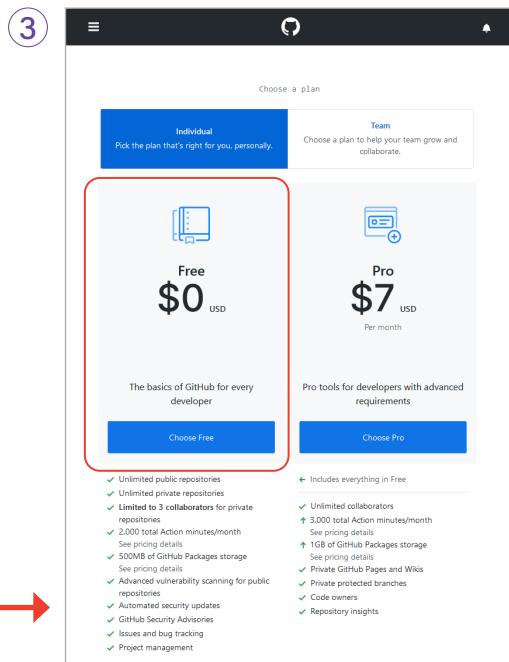
②



- Username (ユーザー名)
- Email address (メールアドレス)
- Password (パスワード)

を入力して、「Select a plan」をクリックします。

③



「Choose a plan」の画面が開きますので、「Choose Free」を選択します。

4

Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.

What kind of work do you do, mainly?

主にどんな仕事をしていますか?

Software Engineer
Student
Product Manager
UX & Design
Data & Analytics
Marketing & Sales
Teacher
Other

How much programming experience do you have?

プログラミングの経験はどれくらいありますか?

None
I don't program at all
A little
I'm new to programming
A moderate amount
I'm somewhat experienced
A lot
I'm very experienced

What do you plan to use GitHub for?
(Select up to 3)

GitHubの使用目的は何ですか?

Learn to code
Learn Git and GitHub
Host a project (repository)
Create a website with GitHub Pages
Collaborating with my team
Find and contribute to open source
School work and student projects
Use the GitHub API
My interests
My interests

I am interested in:

languages, frameworks, industries

We'll connect you with communities and projects that fit your interests.
For example: windows data-structures neo

Complete setup
Skip this step

5

Almost done, @moniker964! To complete your GitHub sign up, we just need to verify your email address:

Verify email address

Once verified, you can start using all of GitHub's features to explore, build, and share projects.

最後に、確認のメールが届きますので、「Verify email Address」をクリックして、ログインします。

6

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository? Import a repository.

Owner: moniker964 / Repository name *

Great repository names are short and memorable. Need inspiration? How about bug-free-parkette?

Description (optional):

Public
Anyone can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.
Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None

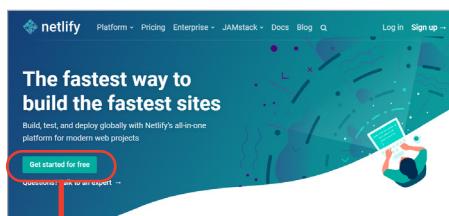
Create repository

「Create a new repository」の画面が開き、準備完了です。

アンケートが開きますので、質問の答えを選択して「Complete setup」をクリックします。

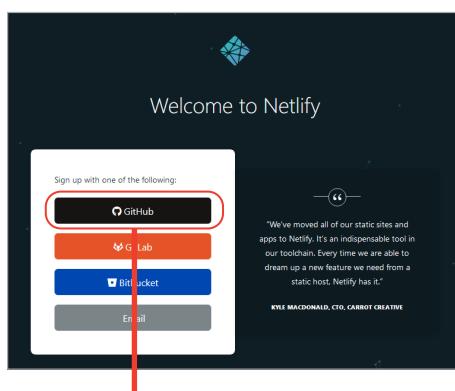
2-3 Netlify

①



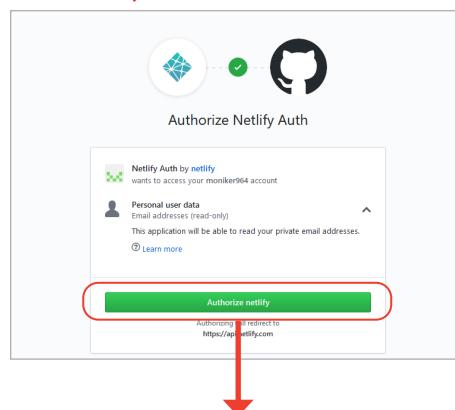
Netlify のページにアクセスし、
「Get started for free」をクリックします。

②



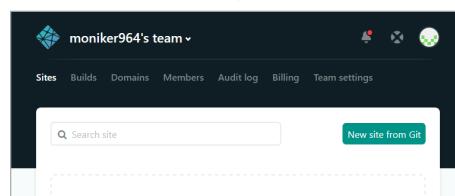
図のような画面が開きます。
ここでは、先ほど作成した GitHub のアカウント情報を使って Netlify のアカウントを作成しますので、「GitHub」をクリックします。

③



「Authorize Netlify Auth」の画面が開きます。GitHub で設定したメールアドレスの情報を使ってもよいかとたずねてきますので、「Authorize netlify」をクリックします。

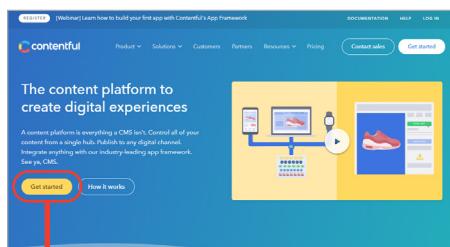
④



図のような画面が開き、こちらも準備完了です。

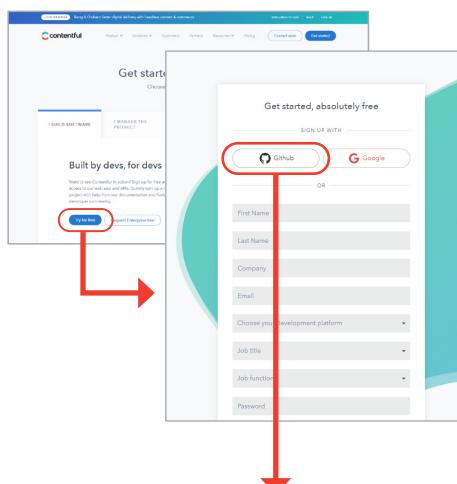
2-4 Contentful

①



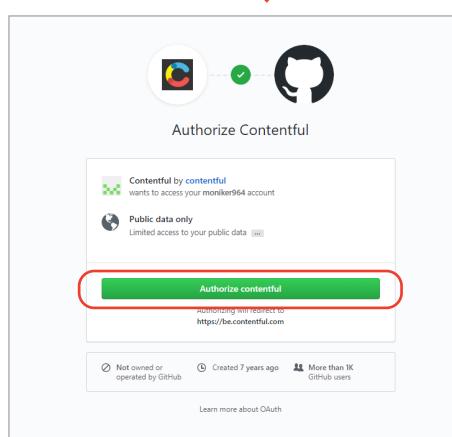
Contentful のページにアクセスし、
「Get started」をクリックします。

②



「Try for free」をクリックし、登録画面を開きます。
ここでは、先ほど作成した GitHub のアカウント情報を
使って Contentful のアカウントを作成するため、「Github」をクリックします。

③



「Authorize Contentful」の画面が開きます。
GitHub で公開しているデータ (Public data) への
アクセスが求められますので、「Authorize contentful」をクリックします。

4

Your account is on its way.
Filling a form takes a minute. Enjoying Contentful lasts for years.

First name
Last name
Organization name
Email

Sign up

OR SIGN UP WITH

Github Google

By signing up you agree to our [Terms of service](#) and [Privacy policy](#).

アカウント情報の入力が求められます。必要な項目を入力して「Sign up」をクリックします。

5

How do you usually work with content?

I create content
The Contentful web app enables you to create, manage and publish content.
Explore content modeling

I develop content-rich products
Contentful enables you to manage, integrate and deliver content via APIs.
Deploy a module in 10 days

コンテンツを作成していくため、I create content の「Explorer content modeling」をクリックします。

6

We're preparing your example project
Here's what you can do with it

Flexible content and fields
Easily customize the structure of your content fields using content types.

Versioned content
Manage content in multiple languages which is automatically versioned.

Images, videos and more, in the cloud
Manage assets such as images, videos while benefiting from our caching CDN.

RESTful APIs
Quickly deliver your content to any of your platforms or update it programmatically using our APIs.

Explore the example project

example project（サンプルプロジェクト）が準備されますので、「Explorer the example project」をクリックします。

7

Welcome to your The example project space
Use the space to explore the various elements of any content you're using.
After viewing the existing content, by modifying an entry

Modify sample content in 'The example project'
The Example Project is an educational course catalog app
To see the entries of the course catalog, explore the content catalog tab. You can view the courses and the lessons in the catalog tab.

Get started by modifying content
By modifying the title of a lesson entry.
Modify my entry

Explore the Joomla app

図のように管理画面が表示されます。Contentful の準備も完了です。

SETUP

3

サイトの公開

3-1 サイトの公開・更新環境の設定

3-2 GitHubの設定

3-3 Netlifyの設定とサイトの公開

3-4 サイトの更新

3-5 Contentfulによるサイトの更新

Build blazing-fast websites with GatsbyJS

GatsbyJS

3-1 サイトの公開・更新環境の設定

GitHub と Netlify を組み合わせ、サイトの公開・更新が手軽にできる環境を設定していきます。さらに、Contentful で記事を追加・修正・削除した場合にも、サイトが更新されるように設定します。

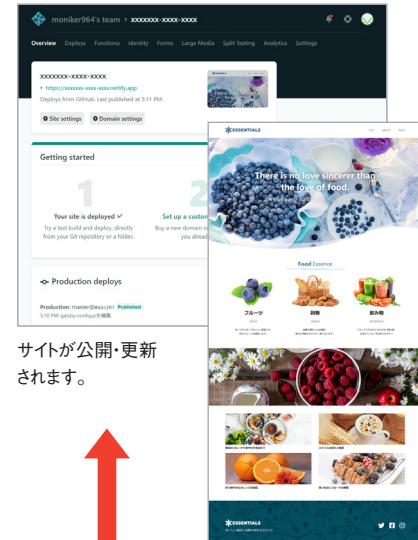
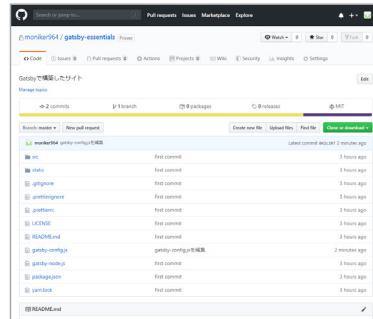


GitHub



Netlify

作成したGatsbyの
プロジェクトをGitHub
にpush(プッシュ)。

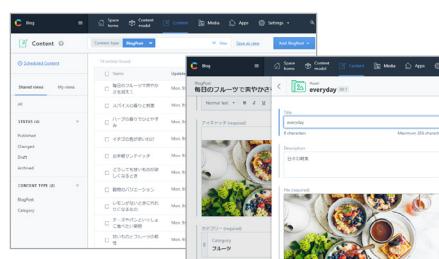


サイトが公開・更新
されます。



Contentful

記事を編集。



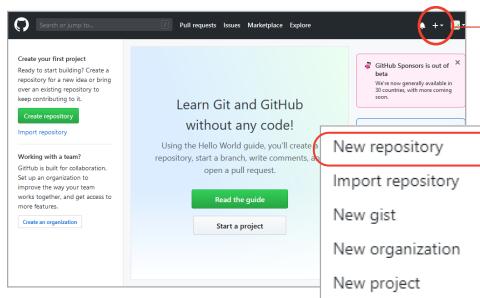
なお、設定を始める前に、ローカル環境でビルド処理 (gatsby build) が問題なく完了することを確認してください。問題がなければ設定を始めましょう。

3-2 GitHubの設定

まずは GitHub の設定を行い、Gatsby のプロジェクトを push (プッシュ) します。

リポジトリを用意する

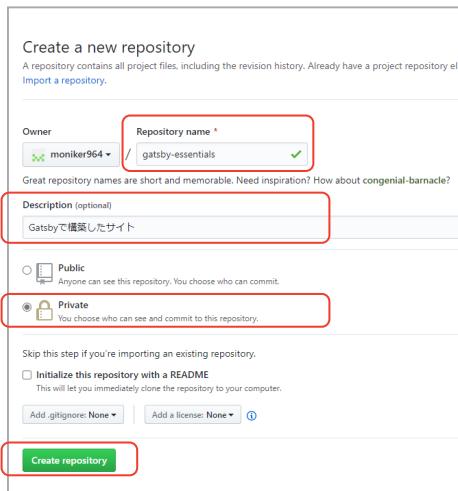
①



Gatsby のプロジェクトを管理するリポジトリを用意します。

GitHub に Sign in (サインイン) し、右上の「+」をクリックして「New repository」を選択します。

②



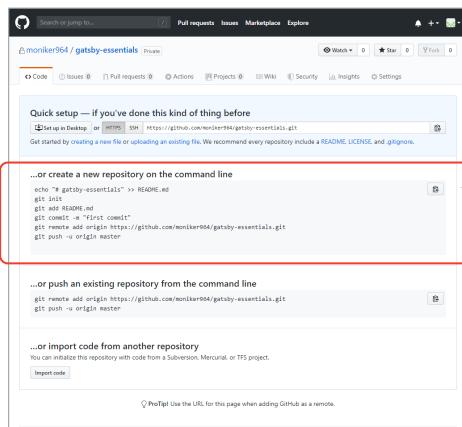
「Create a new repository」という画面が開きますので、Repository name (リポジトリ名) を指定します。ここでは、「gatsby-essentials」と指定しています。

Description (説明) は必要に応じて入力します。

「Public」と「Private」では、このリポジトリを公開にするか、非公開にするかを指定します。ここでは「Private」を選択し、「非公開」にしています。

設定が完了したら、「Create repository」をクリックします。

③



左のような画面が開きます。

...or create a new repository on the command line

```
echo "# gatsby-essentials" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/moniker964/gatsby-essentials.git
git push -u origin master
```

この部分を参考に、gatsby のプロジェクトをこのリポジトリへ push (プッシュ) します。
なお、このページはブラウザで開いたままにしておきます。

リポジトリへプッシュする

Gatsby のプロジェクトフォルダ（本書のサンプルの場合は mysite フォルダ）のルートに移動します。「gatsby-starter-hello-world」スターターを利用した場合、すでに .git フォルダが存在しています。これを流用することができますが、ここではリポジトリの新規作成から行うため、.git フォルダを削除します。

```
$ rm -rf .git
```

続けて、次のコマンドを実行していきます。

①

```
$ git init
```

ローカルのリポジトリを新規作成します。.git フォルダが作られます。

②

```
$ git add -A
```

すべてのファイル (mysite フォルダの中身) をインデックスに登録します。
ただし、.gitignore に設定されているファイルやフォルダは登録しません。

③ `$ git commit -m "first commit"`

メッセージを指定して、コミットします。

④ `$ git remote add origin https://github.com/moniker964/gatsby-essentials.git`

リモートリポジトリを指定します。

ここでは、GitHub に作成したリポジトリを指定しています。

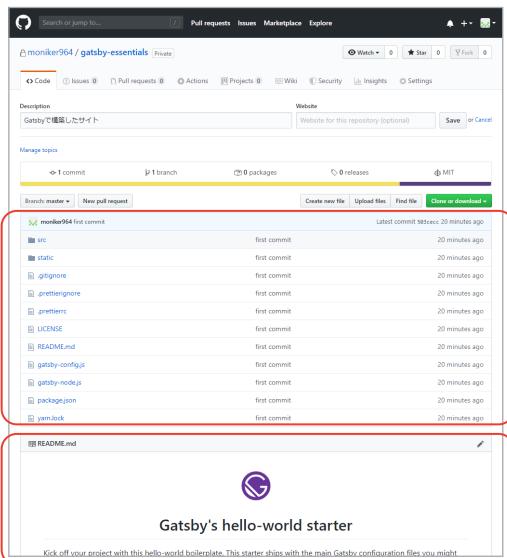
この部分がリポジトリごとに
変わります。

⑤ `$ git push -u origin master`

リモートリポジトリへプッシュします。

ここでは、gatsby-starter-hello-worldを利用した場合を想定しています。スターターや環境によつては、.gitignoreの設定が十分ではない場合があります。その場合は、プッシュするファイルを指定するか、.gitignoreをカスタマイズしてください。

リポジトリを確認する



プッシュが完了したら、開いていた GitHub のページをリロードしてください。プッシュされたファイルが確認できます。

また、README.md の内容が表示されます（作り直していない場合には、スターターに含まれていたものがそのままプッシュされています）。

以上で、GitHub の準備は完了です。

プッシュされたファイル。

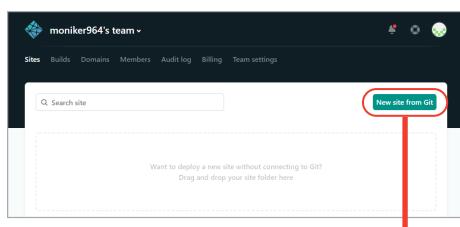
README.mdの内容。

3-3 Netlifyの設定とサイトの公開

Netlify の設定を行い、サイトを公開します。

サイトを公開する

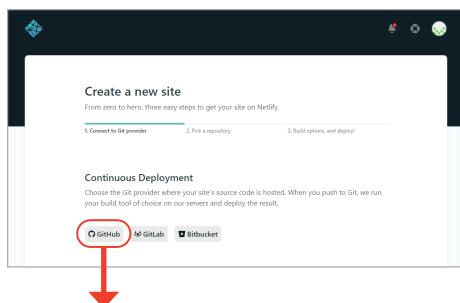
①



Netlify に Log in (ログイン) します。

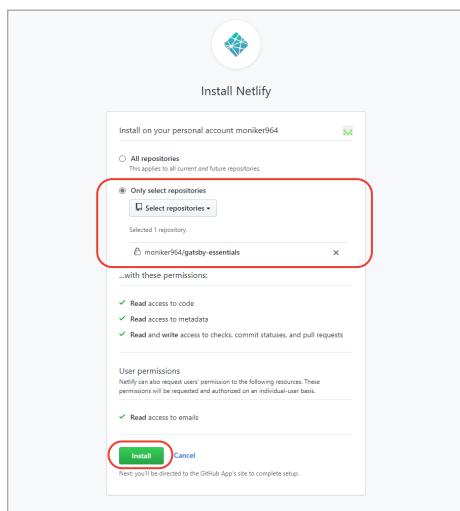
左のようなページが開きますので、「New site from Git」をクリックします。

②



サイト構築に使用するリポジトリを用意した場所を選択します。ここでは「GitHub」をクリックします。

③

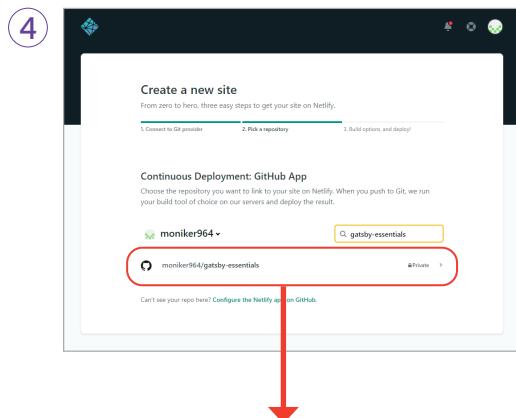


GitHub のページが開きますので、Netlify からのアクセスを許可するリポジトリを選択します。

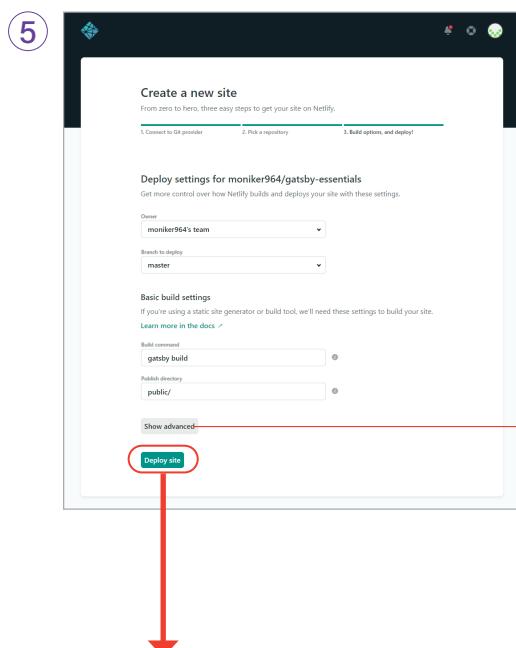
ここでは選択したリポジトリへのアクセスのみを許可するため、「Only select repositories」を選択し、先ほど GitHub で用意したリポジトリを選択します。



設定ができたら「Install」をクリックします。



Netlify の画面に戻りますので、選択したリポジトリをクリックします。



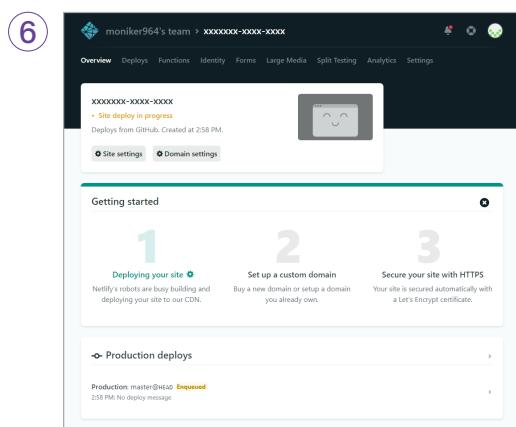
サイトをビルドするのに必要な設定を行います。

環境変数の設定が必要な場合は、「Show advanced」をクリックして設定します（後から設定することもできます）。

他の項目はそのまま問題ありませんので、「Deploy site」をクリックします。

環境変数の設定。

本書のサンプルではP.118の設定後に環境変数の設定が必要になります。後から環境変数の設定を行う方法については次ページを参照してください。

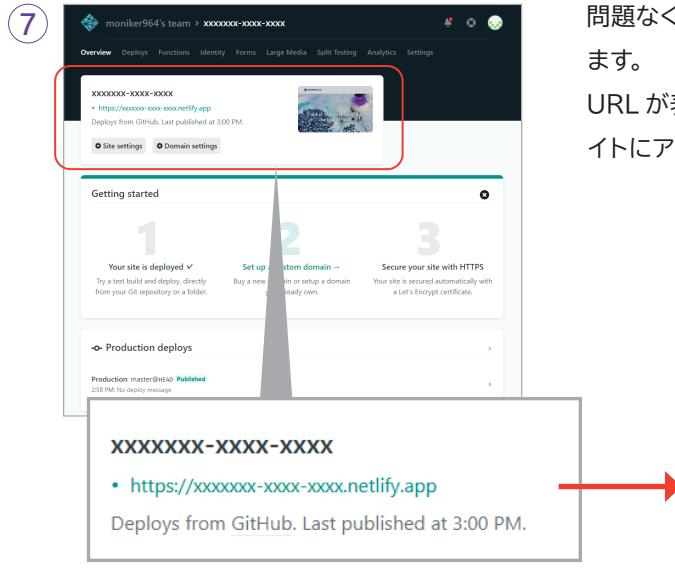


サイトのデプロイが始まります。

XXXXXX-XXXX-XXXX

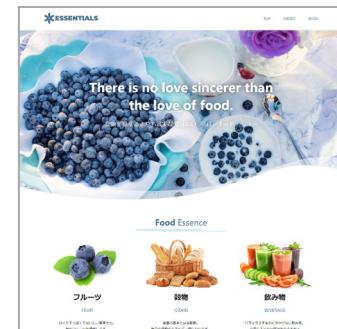
• Site deploy in progress

Deploys from GitHub. Created at 2:58 PM.



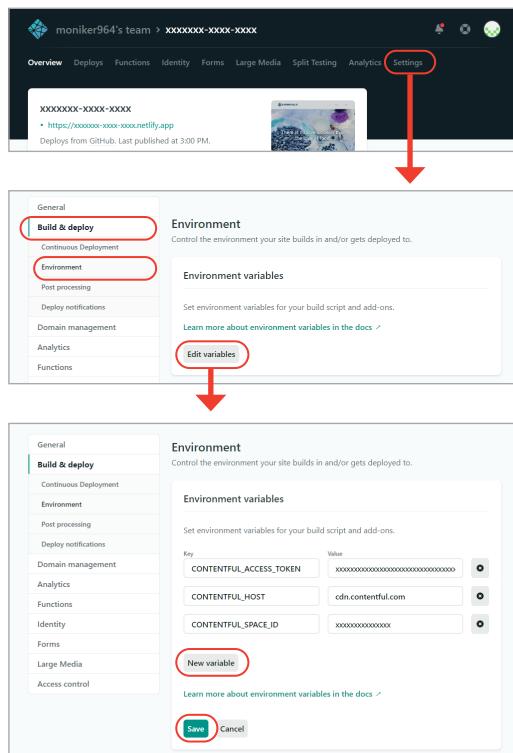
問題なく処理が完了すると、サイトが公開されます。

URL が表示されていますので、この URL でサイトにアクセスすることができます。



公開されたサイトにアクセスできます。

環境変数の追加設定



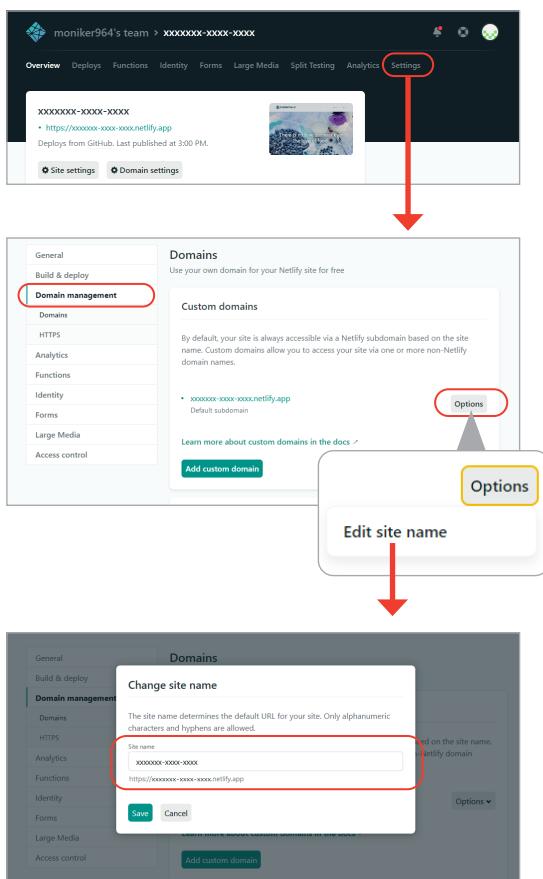
すでに公開しているサイトの設定に環境変数を追加する場合は、上部のメニューから「Settings」を選択します。

左側のメニューから「Build & deploy > Environment」を選択し、「Environment variables」で「Edit variables」をクリックして設定します。

新しい環境変数は「New variable」をクリックして追加していきます。

設定ができたら「Save」をクリックして保存します。

サブドメインの変更

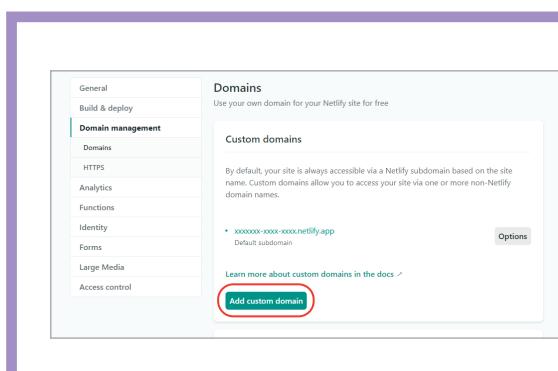


サイトの URL は、サブドメインの部分を変更できます。上部のメニューから「Settings」を選択します。

左側のメニューから「Domain management」を選択します。

「Custom domains」で現在設定されているサブドメインの右にある Options から「Edit site name」をクリックします。

Change site name という設定が開きますので、サブドメインを指定します。「Save」をクリックして保存したら完了です。



「Add custom domain」から、自分が所有しているドメインを設定することもできます。

アクセスを許可するリポジトリの設定

Netlify からのアクセスを許可するリポジトリの追加や削除は、GitHub の「Settings > Applications > Netlify」で行います。

右上のメニューから「Settings」を選択。

「Applications」を選択。

Netlifyの「Configure」をクリック。

「Select repositories」でアクセスを許可するリポジトリを追加します。

アクセスを許可したリポジトリ。
右側の「×」をクリックして削除できます。

3-4 サイトの更新



ページを追加・修正したものをサイトに反映してみましょう。まずは、ローカル環境でビルド処理 (gatsby build) が通ることを確認します。

問題がなければ、以下のコマンドを実行していきます。

① `$ git add -A`

すべての変更をインデックスに登録します。

② `$ git commit -m "コミットの際のメッセージ"`

メッセージを指定して、コミットします。

③ `$ git push`

リモートリポジトリへプッシュします。

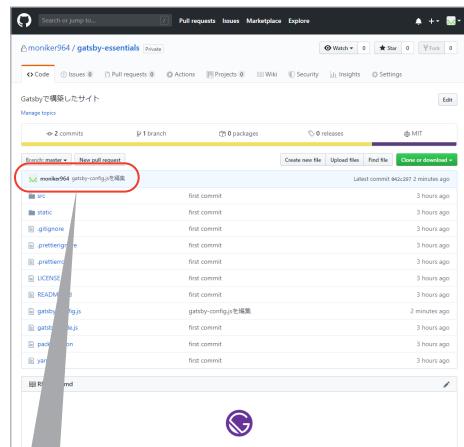
初めてプッシュを実行した際には、次のように「-u」を付けてコマンドを実行しています。そのため、2回目以降は「git push」と指定するだけで同じリモートリポジトリへプッシュすることができます。

`$ git push -u origin master`

修正箇所が、GitHub のリポジトリに反映されます。

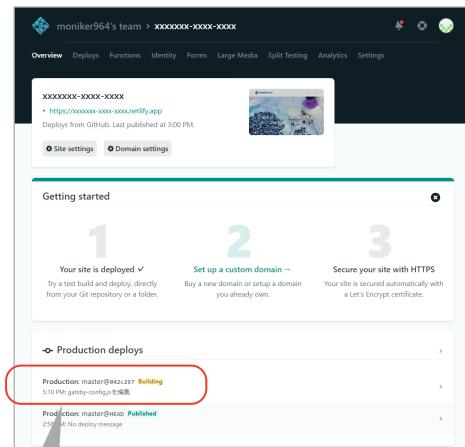
すると、Netlify は GitHub のリポジトリの変化に反応し、自動的にデプロイ処理が行われます。処理が完了すると、修正がサイトに反映されます。

GitHub

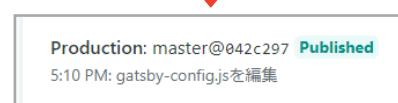


コミット時に指定したメッセージが表示され、修正箇所がリポジトリに反映されたことがわかります。

Netlify

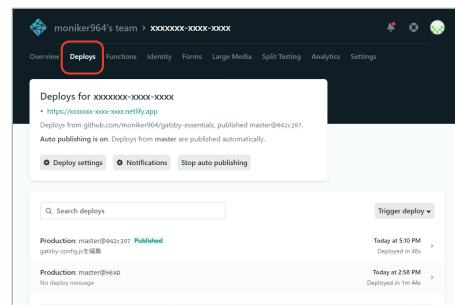


自動的にデプロイ処理が開始されます。



完了すると「Published」と表示され、サイトが更新されます。

デプロイの履歴や詳細は、上部のメニューから「Deploys」を選択して確認することができます。

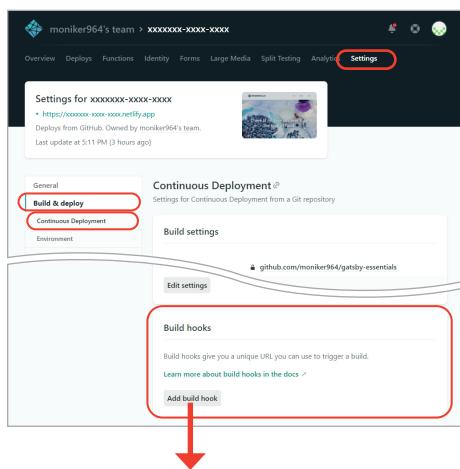


3-5 Contentfulによるサイトの更新

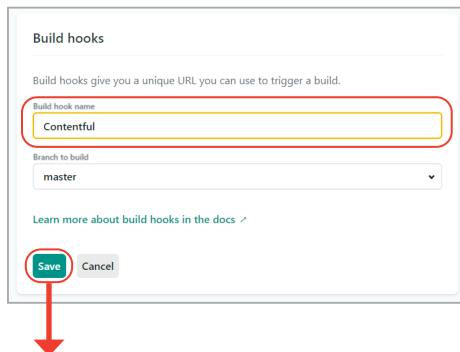
Contentfulで記事を追加・修正・削除した場合にも、サイトが更新されるように設定します。

Netlifyの設定

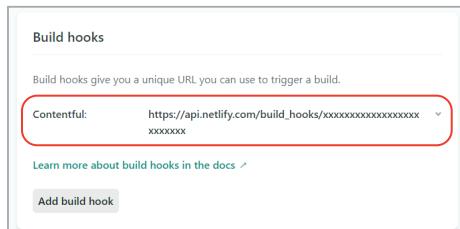
①



②



③



まず、NetlifyでBuild hook（ビルトフック）を作成します。

上部のメニューから「Settings」を選択してサイトの設定画面を開き、「Build & deploy > Continuous Deployment」を選択します。

「Build hooks」で「Add build hook」をクリックします。

「Build hook name」でフック名を指定します。ここでは「Contentful」と指定し、「Save」をクリックして保存します。

URLが表示されますので、メモしておきます。

Contentfulの設定

- ① 続いて、Contentful の設定を進めます。Contentful でスペースを選択した状態で、上部のメニューから Settings > Webhooks を選択します。Webhooks の画面が開きますので、Netlify の「Add」をクリックします。

本書のサンプルでは「Blog」スペースを選択。

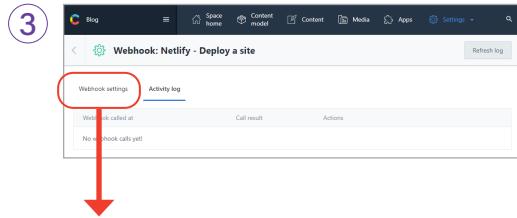
本書のサンプルでは「Blog」スペースを選択。

Settings > Webhooks を選択。

Netlifyの「Add」をクリック。

- ② 「Netlify build hook URL」に、先ほどメモした URL を指定します。指定できたら、「Create webhook」をクリックして設定を保存します。

メモしたURLを指定。



左のような画面になりますので、「Webhook settings」へ移動します。

Netlifyでデプロイが行われるトリガーを Triggers で確認します。すると、「Entry (記事)」と「Asset (アセット)」が Publish / Unpublish されたときに、デプロイが行われる設定になっています。

	Create	Save	Autosave	Archive	Unarchive	Publish	Unpublish	Delete
Content type	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					
Entry	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
Asset	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				

そこで、「Asset」のチェックは外し、「Entry」の Publish / Unpublish / Delete にチェックをつけます。

右上の「Save」をクリックして保存すれば、設定は完了です。

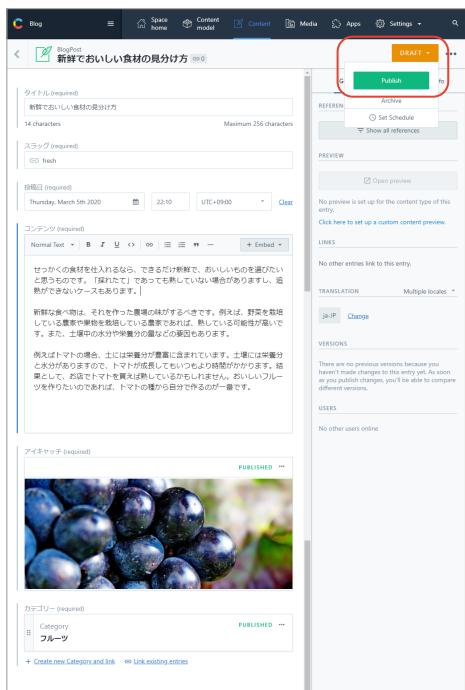
	Create	Save	Autosave	Archive	Unarchive	Publish	Unpublish	Delete
Content type	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					
Entry	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				
Asset	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>					

記事の投稿

Contentful で記事を投稿・修正・削除すると、サイトが更新されます。

たとえば、新しい記事を Publish (投稿) してみると、Netlify でデプロイが実行され、サイトが更新されます。

Contentful



Blog Post: 新鮮でおいしい食材の見分け方

タイトル (required): 新鮮でおいしい食材の見分け方

14 characters Maximum 256 characters

スラッグ (required): fresh

日付 (required): Thursday, March 5th 2020 22:10 UTC+09:00

コンテンツ (required): Normal Text

コメント: せっかくの食材を買入るなら、できるだけ新鮮で、おいしいものを選みたいと思うものです。「新鮮さ」によって熟していない場合がありますし、過熟ができないケースもあります。新鮮な食材は、それを食べた感覚が必ずしもです。例えば、野菜を栽培している農家や卸業者を経由している場合は、それは熟していない可能性があります。また、土壠中の水分や酸素量などの要因があります。

西野トマトの場合、土には半分埋め置かれています。土壠には半分分の水分が残っているので、その分の酸素量が少ないので、熟しない可能性があります。結果として、自分でトマトを育てているかもしれません。おいしいフルーツを作りたいのでは、トマトの種類の日付で選ぶのが一番です。

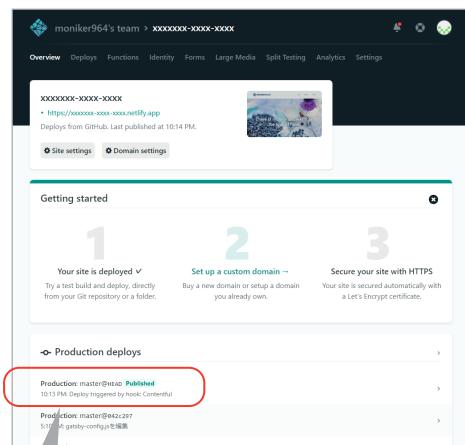
アイキャッチ (required): 

カテゴリ (required): Category フルーツ

Published

新しい記事を Publish (投稿) します。

Netlify



moniker964's team > x000000-x000-x000

Overview Deploy Functions Identity Forms Large Media Split Testing Analytics Settings

Getting started

1 Your site is deployed ✓ Try a test build and deploy directly from your Git repository or a folder.

2 Set up a custom domain → Buy a new domain or setup a domain you already own.

3 Secure your site with HTTPS Your site is secured automatically with a Let's Encrypt certificate.

Production deploys

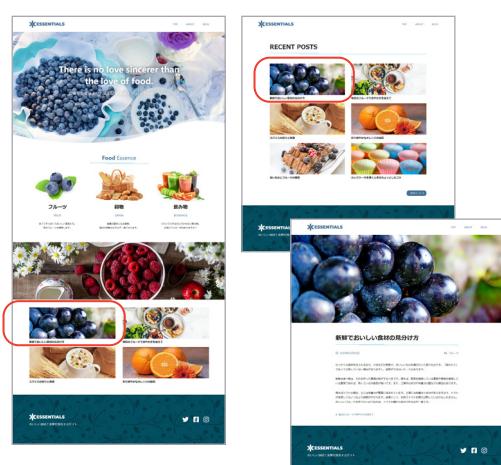
Production: master@HEAD Published 10:13 PM: Deploy triggered by hook: Contentful

Production: master@HEAD 2027 5/10 11:41:46 UTC+09:00

Production: master@HEAD

Production: master@HEAD Published
10:13 PM: Deploy triggered by hook: Contentful

「Contentful」フックによってデプロイが実行され、サイトが更新されます。



サイトを確認すると、右のように記事一覧に新しい記事へのリンクが追加され、記事ページが作成されています。

以上で、Contentful によるサイトの更新の設定は完了です。

Netlify側でデプロイを実行する

Netlify と連携させた GitHub や Contentful を連続して更新した場合などには、うまくデプロイされず、最新の状態にならないケースがあります。

そのようなときには、Netlify 側でデプロイを実行することができます。上部のメニューで「Deploys」を選択し、「Trigger deploy」でデプロイを実行します。すると、GitHub や Contentful からデータが読み込まれ、デプロイが実行されます。

The screenshot shows the Netlify Deploy page for a team. The 'Deploys' tab is selected. A list of previous deployments is shown, each with a timestamp, duration, and a link to view the deployment details. To the right of the list, a 'Trigger deploy' button is highlighted with a red box. Below the list, there is a search bar and buttons for 'Deploy settings', 'Notifications', and 'Stop auto publishing'.

Trigger deployで「Deploy site(デプロイする)」または「Clear cache and deploy site(キャッシュを消してデプロイする)」を選択します。

The screenshot shows the 'Deploy in progress' page. It displays the status 'Production: master@HEAD. Today at 9:15 AM' and a message that Netlify's robots are busy building and deploying the site to the CDN. Below this, there are buttons for 'Deploy settings' and 'Cancel deploy'. The 'Deploy log' section shows the build process, including the creation of a build image tag (v1.3.7), the buildbot version (1.36.0), and the starting of the build process. The log ends with the message 'Starting to extract cache'.

デプロイが実行されます。

The screenshot shows the 'Published deploy' page. It displays the status 'Production: master@HEAD. Today at 9:15 AM' and a 'Preview deploy' button. Below this, there are buttons for 'Deploy settings', 'Lock publishing to this deploy', and 'Retry deploy'. The 'Deploy summary' section shows that all files are already uploaded and no redirect rules were processed. The message '[Published deploy]' is highlighted with a red box.

「Published deploy」と表示されたら完了です。

SETUP

4

Contentfulによる コンテンツ管理

- 4-1 Contentfulによるコンテンツの管理
- 4-2 スペースの作成
- 4-3 コンテンツを管理できるようにする
- 4-4 カテゴリーを管理できるようにする
- 4-5 記事を投稿する
- 4-6 トーカンの確認
- 4-7 Contentfulのデータをインポートする

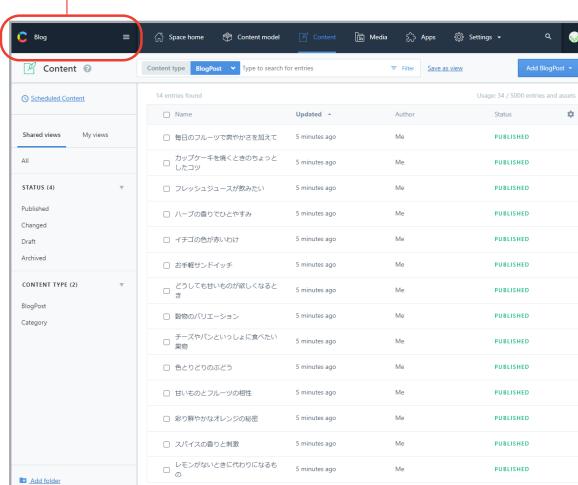
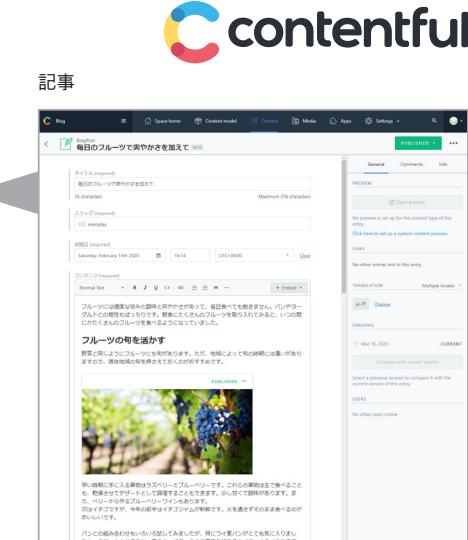
Build blazing-fast websites with GatsbyJS

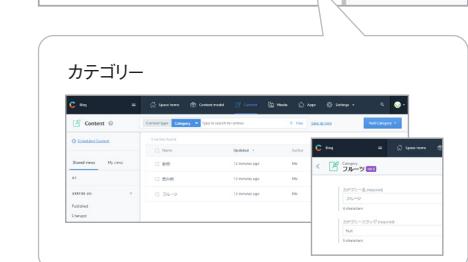
GatsbyJS

4-1 Contentfulによるコンテンツの管理

Contentfulでは「スペース」を作成し、コンテンツを管理します。ここでは本書で作成するブログの管理を行う設定をしていきます。

なお、各種設定や記事の投稿などを行ったインポートデータも用意しています。インポートデータを利用する場合、4-2の「スペースの作成」を行ってから、4-7に進んでください。

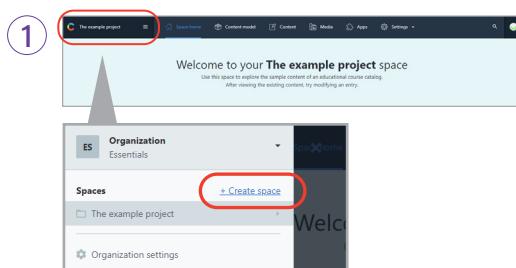





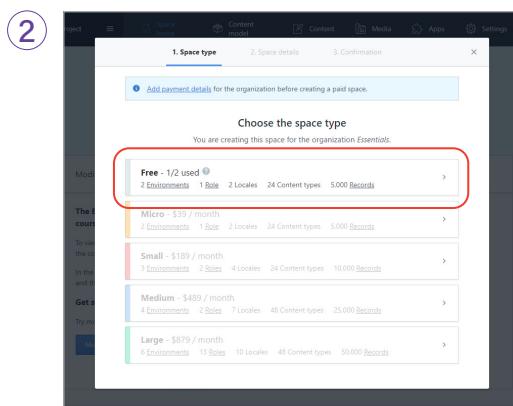
4-2 スペースの作成

まずは、コンテンツを管理する「スペース」を作成し、ロケール（言語）を設定します。

スペースを作成する

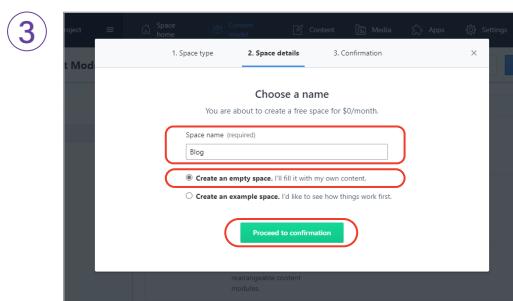


Contentful に LOG IN (ログイン) します。
新しいスペースを作成するため、左上のメニューを開き、「Create space」をクリックします。

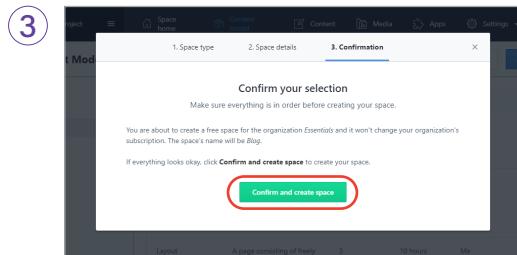


スペースの種類を選択します。ここでは無料で利用できる「Free」を選択します。

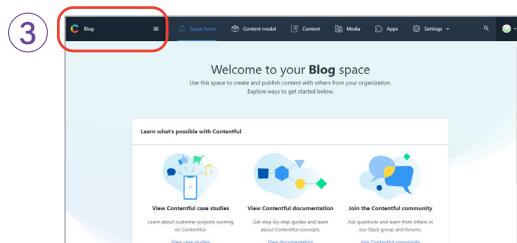
無料で作成できるスペースは2つまでとなっています。
アカウント作成時に用意された「Explorer the example project」が1つのスペースを使用しているため、「1/2」と表示されています。
本書では「Explorer the example project」は使用しませんので、削除しても問題ありません。



スペースは空の状態で作成するため、「Create an empty space」を選択し、「Proceed to confirmation」をクリックします。



「Confirm and create space」をクリックします。



作成したスペースの管理画面が開きます。左上のメニューに「Blog」と表示されていることを確認します。

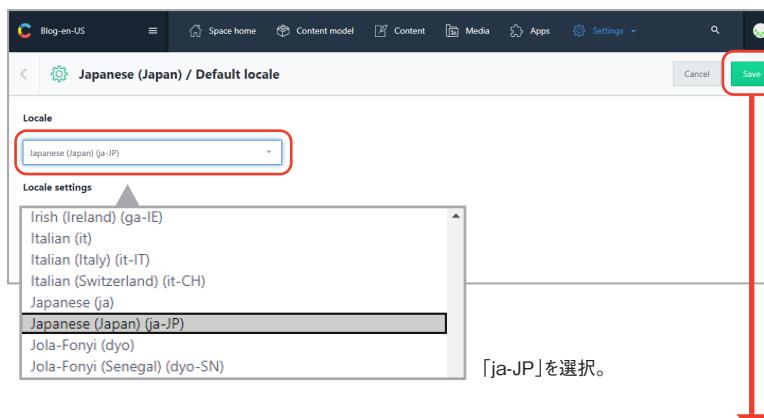
ロケールを設定する

- ① スペースの Locale (ロケール) を日本語に変更するため、「Settings > Locales」を選択し、DEFAULTとして指定されているロケールをクリックします。

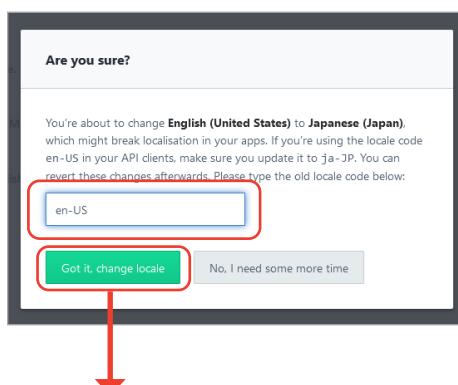
ロケールをクリック。

Settings>Localesを選択。

- ② Locale のプルダウンから「ja-JP」を選択し、「Save」をクリックします。

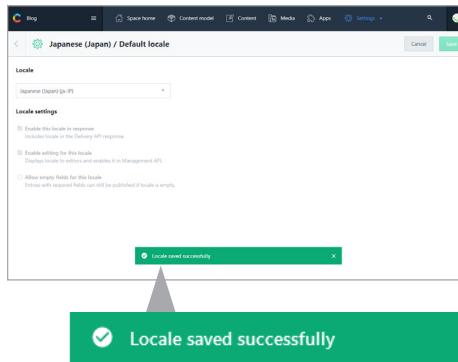


- ③



変更の確認として、直前のロケールの入力を求められます。初期状態では「en-US」となっていますので、これを入力して「Got it, change locale」をクリックします。

- ④



「Locale saved successfully」と表示されたら完了です。

次のステップからコンテンツの管理に必要な設定と、記事の投稿を行います。
各種設定と記事データをまとめてインポートする場合は、4-7に進んでください。

4-3 コンテンツを管理できるようにする



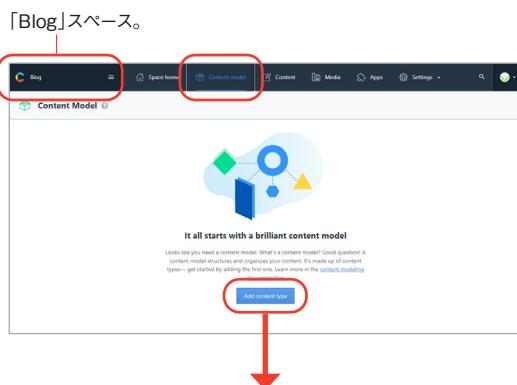
コンテンツの管理に必要な設定を行います。ここでは、本書で作成するブログを管理するため、記事の構成要素を入力フィールドとして用意します。

赤字はエディタ画面に表示するフィールド名、青字は内部的に使用される ID として設定していきます。

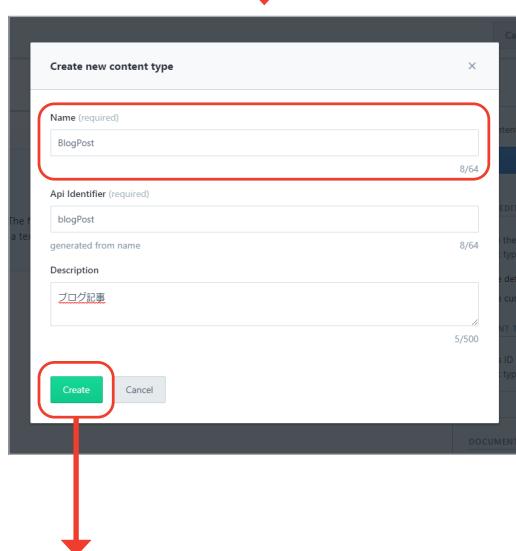


1 コンテンツタイプを作成する

まずは、ブログの記事を管理するコンテンツタイプを作成します。



「Blog」スペースで設定していきます。上のメニューから「Content model」を選択し、「Add content type」をクリックします。

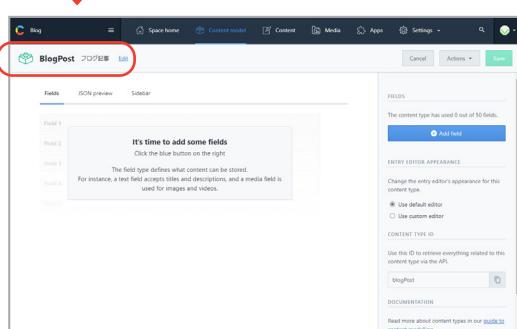


「Name」でコンテンツタイプ名を指定します。ここでは「BlogPost」と指定しています。

「Api Identifier」は名前から自動的に生成されます。

「Description」にはコンテンツタイプについての説明を「ブログ記事」と入力しています。入力しなくとも問題はありません。

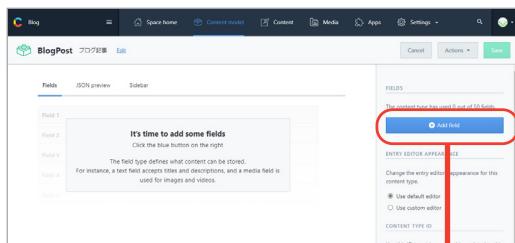
入力できたら「Create」をクリックします。



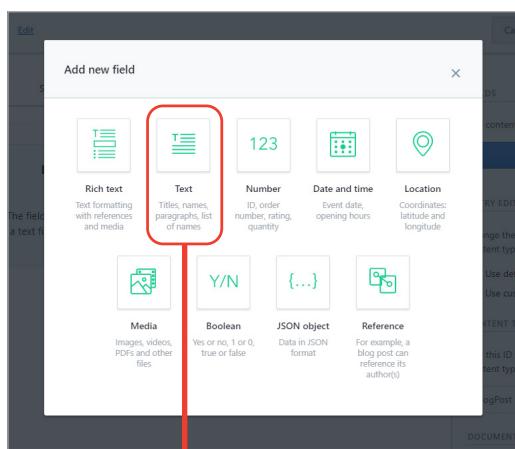
画面の左上に「BlogPost」とコンテンツタイプ名が表示されていることを確認します(表示されるまでしばらく時間がかかる場合もあります)。

記事の入力フィールドはこの画面で作成していきます。

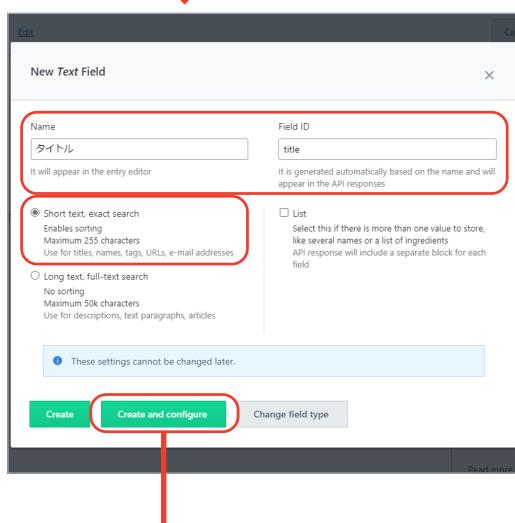
2 タイトルのフィールドを作成する



タイトルのフィールドを作成するため、「Add field」をクリックします。



フィールドの種類の選択肢が表示されます。ここではテキスト形式のデータを入力するフィールドにするため、「Text」を選択します。

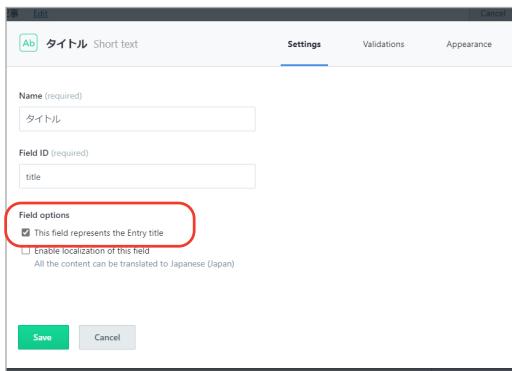


「Name」ではエディタに表示するフィールド名を、「Field ID」ではIDを指定します。IDはGraphQLでデータを取得する際に使用されます。

ここではフィールド名を「タイトル」、IDを「title」と指定しています。

さらに、「Short text」を選択し、短いテキスト用の入力フィールドにします。

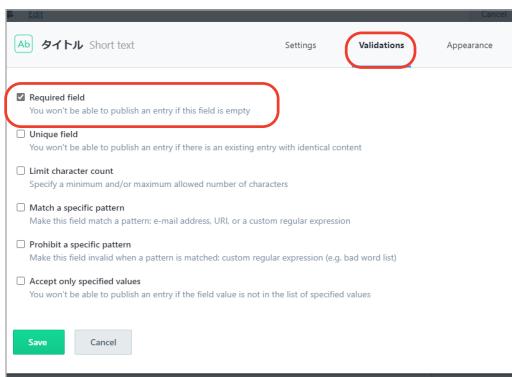
「Create and configure」をクリックしてフィールドを作成し、追加の設定を行います。



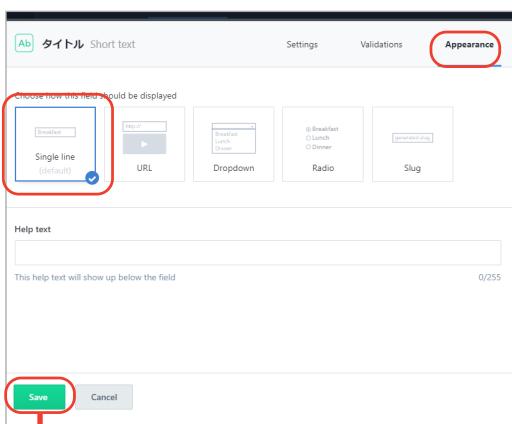
フィールドの設定画面が開きます。

Field options では「This field represents the Entry title」にチェックを付け、タイトルのフィールドであることを明示します。

コンテンツタイプごとに、どれか1つのフィールドをタイトルのフィールドにする必要があります。

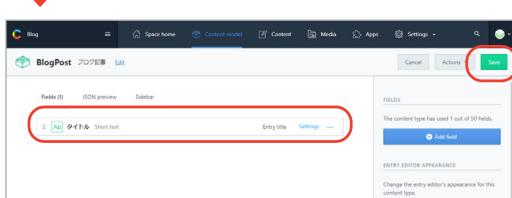


入力を必須にするため、Validations で「Required field」をチェックします。



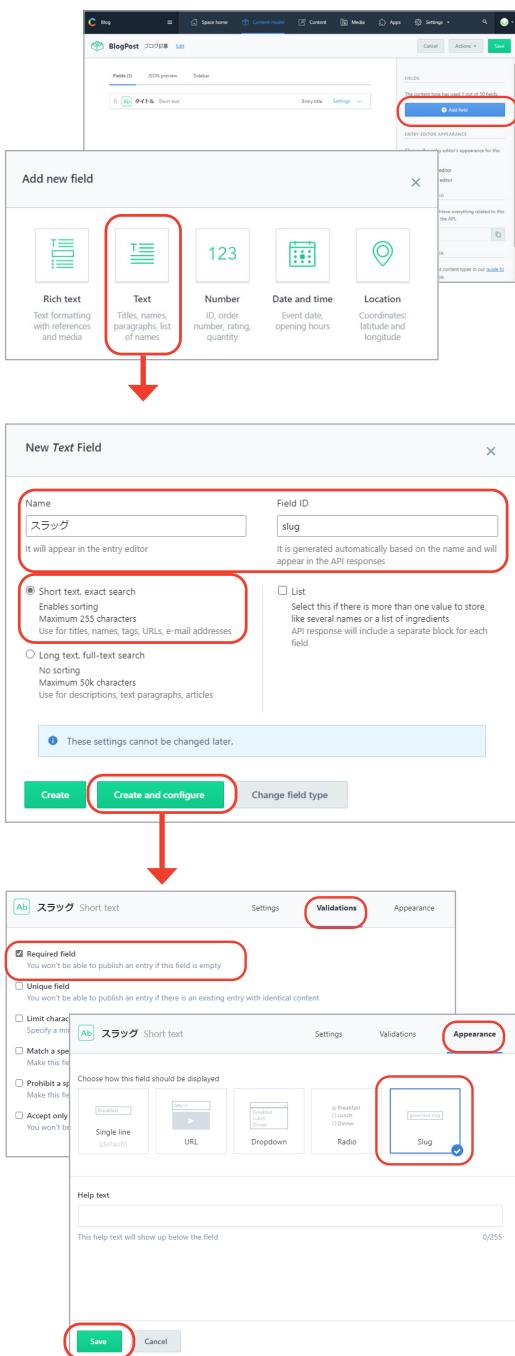
Appearance でフィールドの表示形式が「Single line」になっていることを確認します。

設定ができたら「Save」をクリックして保存します。



元の画面に戻りますので、BlogPost コンテンツタイプに「タイトル」フィールドが追加されたことを確認します。右上の「Save」をクリックし、保存しておきます。

3 スラッグのフィールドを作成する



The screenshot shows the Contentful interface for creating a new field. The top navigation bar includes 'Blog', 'Space home', 'Content model', 'Content', 'Media', 'Apps', and 'Settings'. The 'Content' tab is selected. The main area shows a 'BlogPost' content type with two fields: 'タイトル' (Title) and 'スラッグ' (slug). A modal window titled 'Add new field' is open, showing field types: Rich text, Text, Number, Date and time, and Location. The 'Text' type is selected and highlighted with a red box. A red arrow points down to the 'New Text Field' configuration window. This window has fields for 'Name' (slug) and 'Field ID' (slug). It also includes a 'Short text, exact search' section with sorting and search options, and a 'List' section for multiple values. A note says 'These settings cannot be changed later.' A red box highlights the 'Create and configure' button, which is also highlighted by a red arrow. The bottom part of the screenshot shows the 'slug' field configuration with validation rules (Required field checked), appearance settings (slug selected), and a 'Save' button highlighted with a red box.

タイトルと同じように、スラッグのフィールドを作成します。

「Add field」をクリックし、「Text」を選択します。

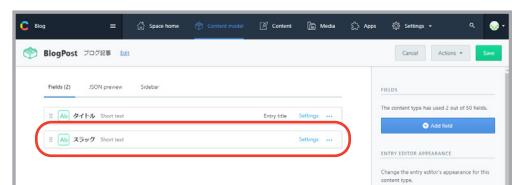
フィールド名を「**スラッグ**」、IDを「**slug**」と指定します。

「Short text」を選択して「Create and configure」をクリックします。

Validationsで「Required field」をチェックし、入力を必須にします。

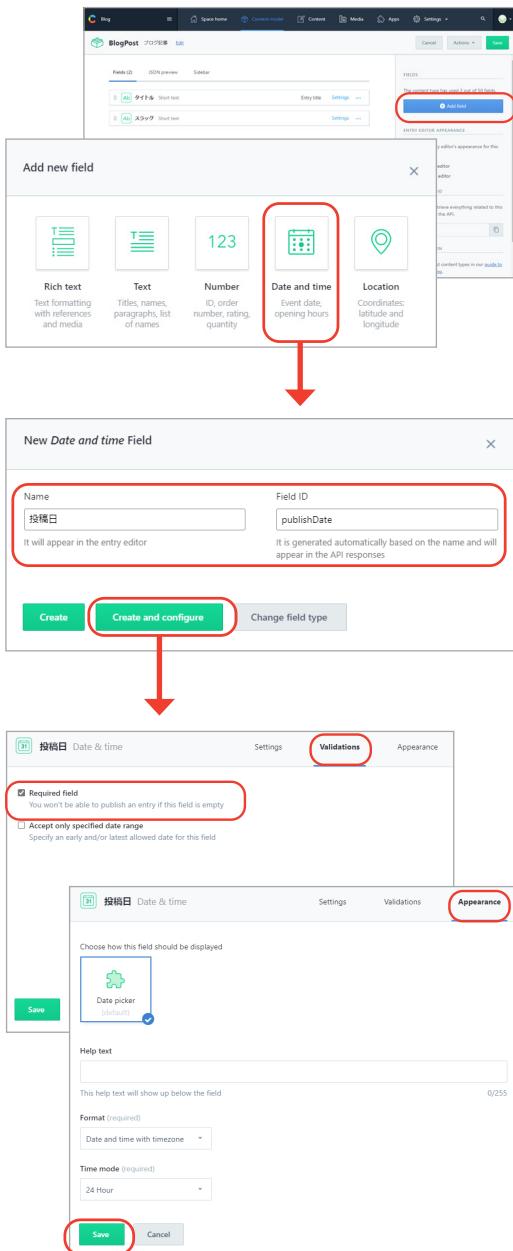
Appearanceではフィールドの表示形式を「Slug」にして、「Save」をクリックします。

これで、「スラッグ」フィールドが追加されます。



The screenshot shows the 'BlogPost' content type in the Contentful interface. The 'slug' field is now listed under the 'FIELDS' section, highlighted with a red box. The 'Save' button at the bottom of the configuration window is also highlighted with a red box.

4 投稿日のフィールドを作成する



投稿日のフィールドを作成します。

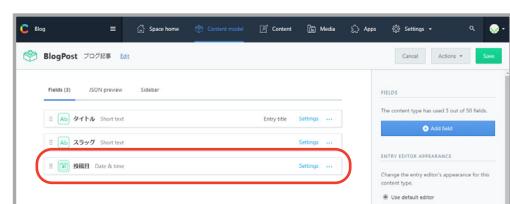
日付形式のデータを入力するフィールドにしたいので、「Add field」をクリックし、「Date and time」を選択します。

フィールド名を「**投稿日**」、IDを「**publishDate**」と指定して、「Create and configure」をクリックします。

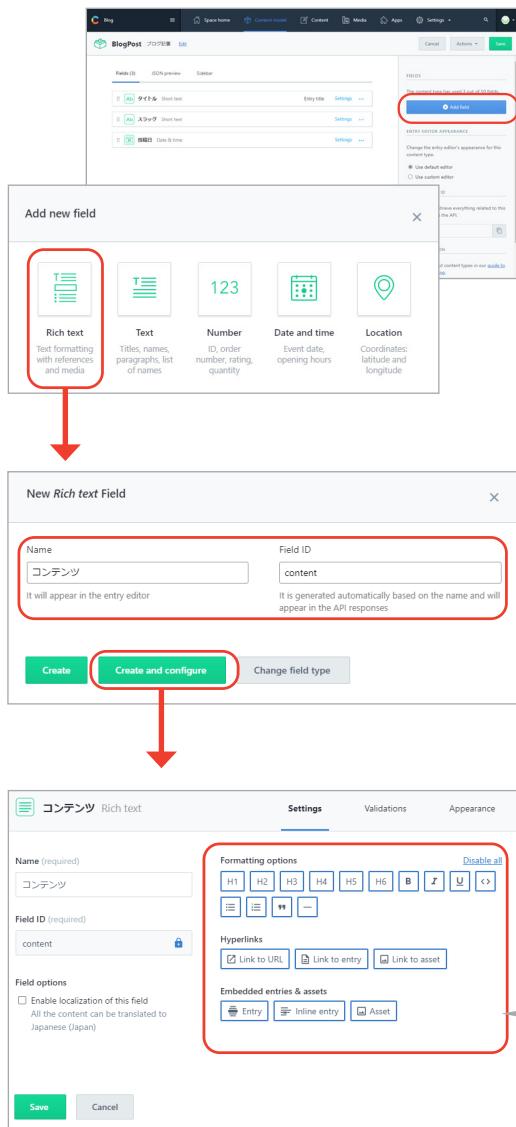
Validationsで「Required field」をチェックして入力を必須にします。

Appearanceは標準の設定のまま、「Save」をクリックします。

これで、「投稿日」フィールドが追加されます。



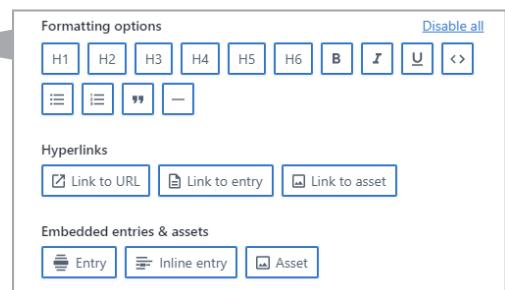
5 コンテンツのフィールドを作成する

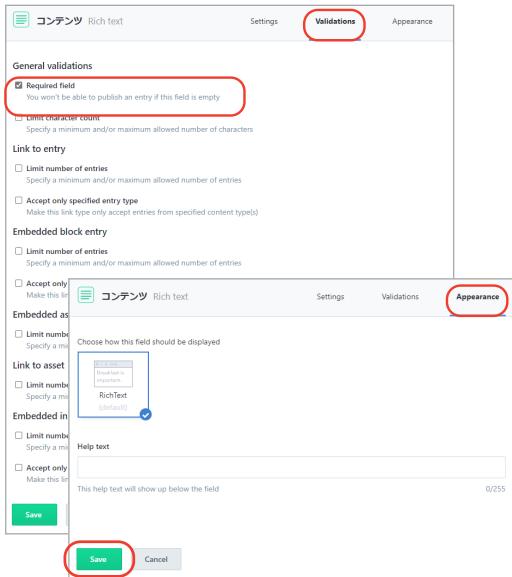


コンテンツ（記事本文）のフィールドを作成します。リッチテキスト形式で入力するフィールドにしたいので、「Add field」をクリックし、「Rich text」を選択します。

フィールド名を「コンテンツ」、ID を「content」と指定し、「Create and configure」をクリックします。

リッチテキストで使用できるようにする機能を選択します。ここでは標準の設定のまま、すべて選択した状態にしておきます。

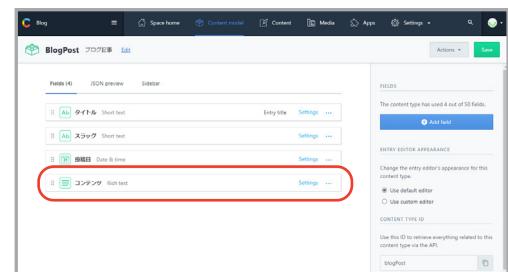




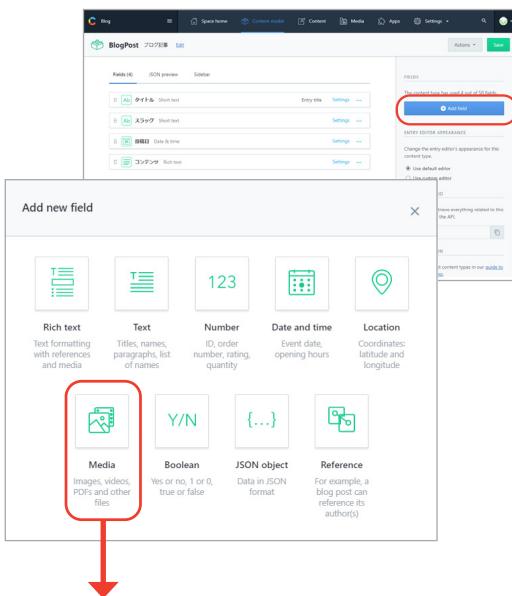
Validations で「Required field」をチェックし、入力を必須にします。

Appearance は標準の設定のまま、「Save」をクリックします。

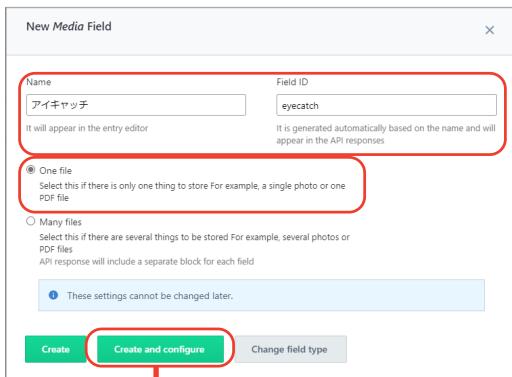
これで、「コンテンツ」フィールドが追加されます。



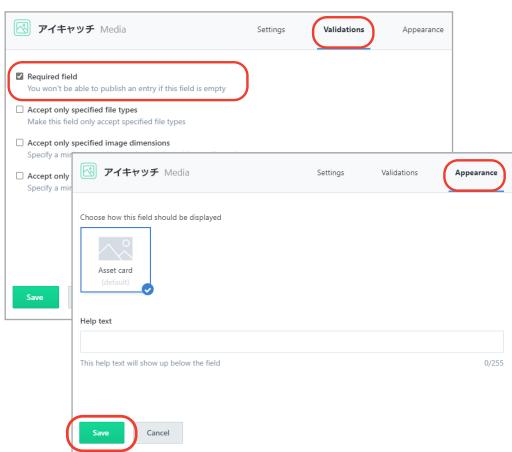
6 アイキャッチ画像のフィールドを作成する



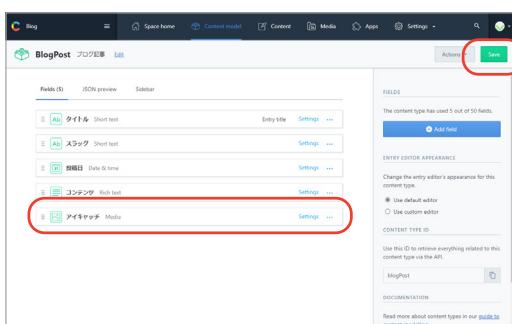
アイキャッチ画像のフィールドを作成します。画像を指定するフィールドにしたいので、「Add field」をクリックし、「Media」を選択します。



フィールド名を「**アイキャッチ**」、IDを「**eyecatch**」と指定し、「Create and configure」をクリックします。



Validationsで「Required field」をチェックして入力を必須にします。
Appearanceは標準の設定のまま、「Save」をクリックします。



「アイキャッチ」フィールドが追加されます。

これで、5つのフィールドの作成は完了です。
カテゴリーフィールドについては個別の設定が必要になるため、次のステップで作成します。

右上の「Save」をクリックして、BlogPost
コンテンツタイプを保存しておきます。

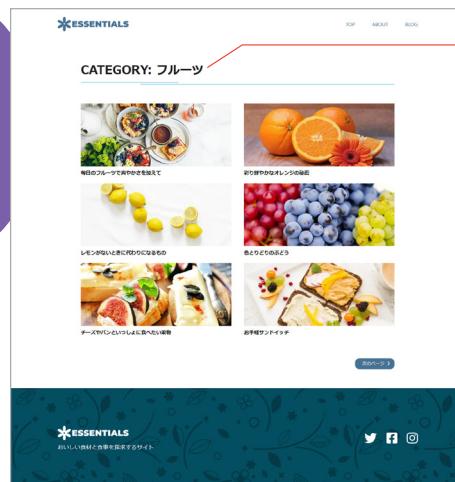
4-4 カテゴリーを管理できるようにする



記事を分類するカテゴリーを管理できるようにします。ここではカテゴリーページを作成することを前提に、カテゴリーの構成要素を入力フィールドとして用意します。



Category
カテゴリーの構成要素



カテゴリー名
category

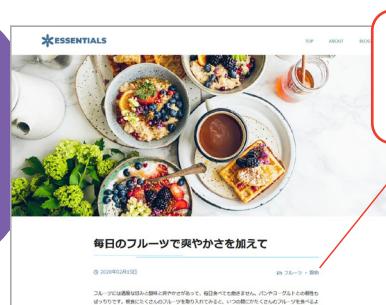


/cat/*fruit*/
カテゴリーURL
categorySlug

その上で、記事ごとに属するカテゴリーを複数選択できるように設定していきます。



Blog Post
記事の構成要素



フルーツ・穀物

カテゴリー
category

1 カテゴリー用のコンテンツタイプを作成する

カテゴリーを管理するコンテンツタイプを作成します。

上のメニューから「Content model」を選択し、「Add content type」をクリックします。

「Name」でコンテンツタイプ名を指定します。ここでは「Category」と指定しています。

「Description」にはコンテンツタイプについての説明を「カテゴリー」と入力しています。入力しなくても問題はありません。

入力できたら「Create」をクリックします。

画面の左上に「Category」とコンテンツタイプ名が表示されていることを確認します(表示されるまでしばらく時間がかかる場合もあります)。

続けて、フィールドを追加していきます。

2 カテゴリー名のフィールドを作成する

The first screenshot shows the 'Add new field' interface. A red box highlights the 'Text' field type icon. A red arrow points down to the 'New Text Field' configuration screen. Another red arrow points down to the 'Validations' and 'Appearance' tabs of the field settings.

The second screenshot shows the 'New Text Field' configuration screen. A red box highlights the 'Name' field with 'カテゴリー名' and the 'Field ID' field with 'category'. A red box highlights the 'Short text, exact search' radio button. A red arrow points down to the 'Create and configure' button, which is also highlighted with a red box.

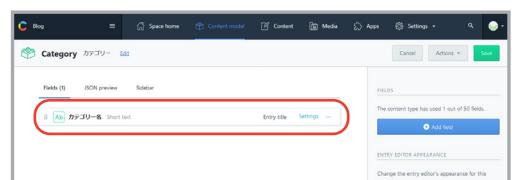
The third screenshot shows the 'Validations' tab of the field settings. A red box highlights the 'Required field' checkbox. A red arrow points down to the 'Appearance' tab, which is also highlighted with a red box. A red box highlights the 'Single line' display option. A red arrow points down to the 'Save' button, which is also highlighted with a red box.

カテゴリー名のフィールドを作成します。
「Add field」をクリックし、「Text」を選択します。

フィールド名を「**カテゴリー名**」、IDを「**category**」と指定します。
「Short text」を選択して「Create and configure」をクリックします。

Validations で「Required field」をチェックし、入力を必須にします。
Appearance ではフィールドの表示形式が「Single line」になっていることを確認し、「Save」をクリックします。

これで、フィールドが追加されます。



③ カテゴリースラッグのフィールドを作成する

The screenshot shows the Contentful interface for creating a new field. In the 'Add new field' dialog, the 'Text' field type is selected, indicated by a red box and arrow. The 'New Text Field' configuration dialog shows the field name 'カテゴリースラッグ' and field ID 'categorySlug'. The 'Short text, exact search' option is selected. The 'Create and configure' button is highlighted with a red box and arrow. The 'Validations' and 'Appearance' tabs are also highlighted with red boxes and arrows. The 'Save' button at the bottom is highlighted with a red box and arrow.

同じように、カテゴリースラッグのフィールドを作成します。

「Add field」をクリックし、「Text」を選択します。

フィールド名を「カテゴリースラッグ」、IDを「categorySlug」と指定します。

「Short text」を選択して「Create and configure」をクリックします。

Validationsで「Required field」をチェックして入力を必須にし、Appearanceは「Slug」に指定して、「Save」をクリックします。

これで、以下のようにフィールドが追加されます。「Save」をクリックして Category コンテンツタイプを保存します。

The screenshot shows the 'Category' content type settings. The 'categorySlug' field is highlighted with a red box and arrow. The 'Save' button in the top right corner is highlighted with a red box and arrow.

4 BlogPostコンテンツタイプにカテゴリーのフィールドを追加する

記事ごとに記事が属するカテゴリーを選択できるようにするために、BlogPost コンテンツタイプにカテゴリーのフィールドを追加します。

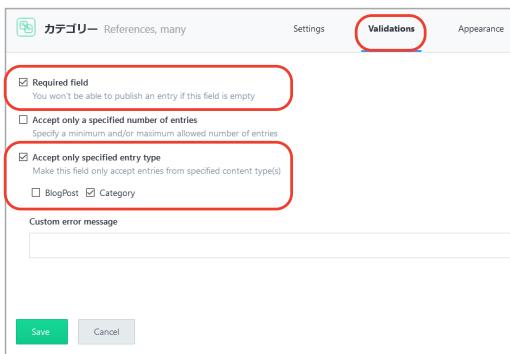
上のメニューから「Content model」を選択します。作成済みのコンテンツタイプがリストアップされますので、「BlogPost」をクリックして編集していきます。

新しいフィールドを追加するため、「Add field」をクリックします。

ここでは Category コンテンツタイプのデータを参照するフィールドを作成するため、「Reference」を選択します。

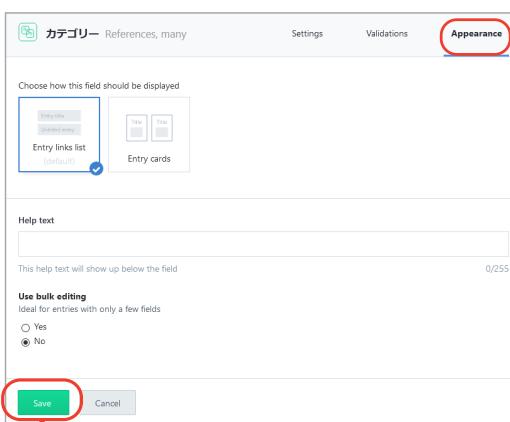
フィールド名を「カテゴリー」、ID を「category」と指定します。

さらに、記事ごとに複数のカテゴリーを選択できるようにするため、「Many references」を選択して「Create and configure」をクリックします。

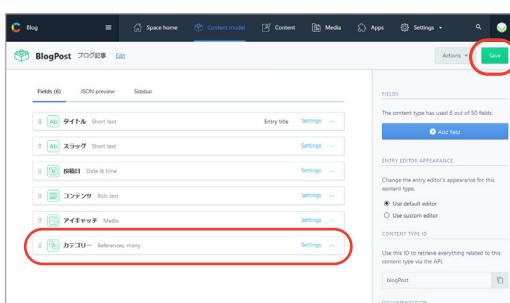


Validations で「Required field」をチェックし、入力を必須にします。

「Accept only specified entry type」にもチェックを付けて、参照先にしたいコンテンツタイプを選択します。ここでは「Category」コンテンツタイプを選択します。



Appearance は標準の設定のまま、「Save」をクリックして保存します。



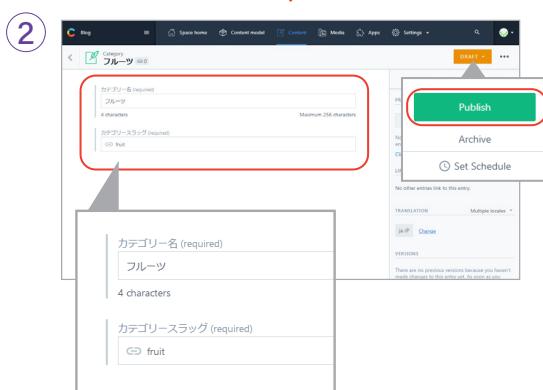
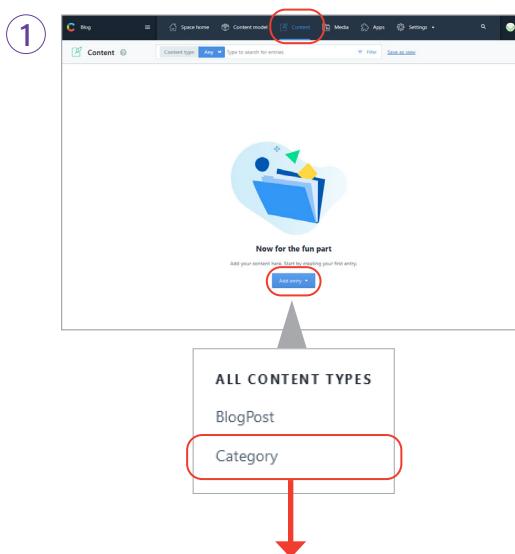
フィールドが追加されますので、「Save」をクリックして BlogPost コンテンツタイプを保存します。

以上で、コンテンツとカテゴリーの管理に必要な設定は完了です。

4-5 記事を投稿する

記事を投稿していきます。ここでは、先に記事を分類する3つのカテゴリーを用意してから、記事の投稿を行います。

カテゴリーを用意する



上のメニューから「Content」を選択し、コンテンツの管理画面を開きます。

何も投稿していない段階では左のような画面になりますので、「Add entry」をクリックします。

コンテンツタイプの選択肢が表示されます。カテゴリーを用意するため、「Category」コンテンツタイプを選択します。

カテゴリーの投稿画面が開きますので、カテゴリー名とスラッグを入力します。ここでは「フルーツ」、「fruit」と入力しています。

入力ができたら、右上の「DRAFT」をクリックし、「Publish」を選択します。

③

投稿したカテゴリー。

上のメニューから「Content」を選択してコンテンツの管理画面に戻ると、投稿したカテゴリーが表示されます。

「Add entry」をクリックし、①～②と同じ手順でカテゴリーを追加していきます。

④

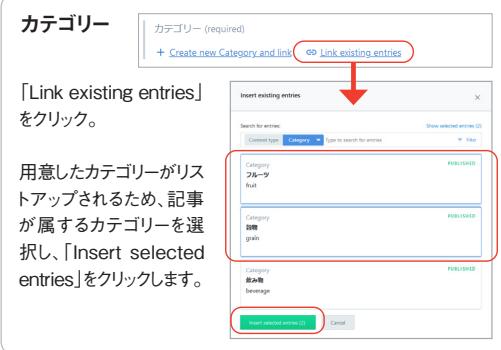
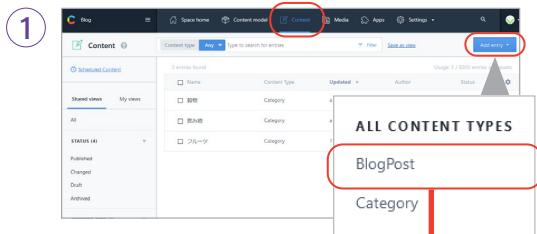
3つのカテゴリーを用意。

ここでは「フルーツ」、「飲み物」、「穀物」の3つのカテゴリーを投稿しました。

以上で、カテゴリーの用意は完了です。続けて、記事を投稿していきます。

連続して投稿する場合、コンテンツの管理画面まで戻らずに、②の画面で右上のメニューから「Create new Category」を選択して投稿を続けることもできます。

記事を投稿する



記事を投稿していきます。

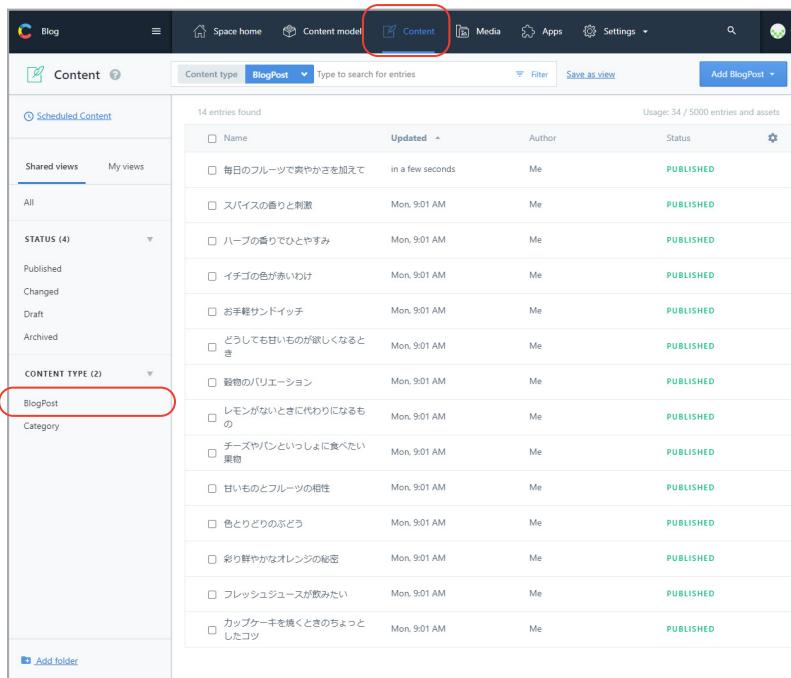
カテゴリーのときと同じように、上のメニューから「Content」を選択し、コンテンツの管理画面を開きます。「Add entry」をクリックし、「BlogPost」コンテンツタイプを選択します。

記事の投稿画面が開きますので、タイトルなどを入力していきます。入力が完了したら「Publish」をクリックします。



- ③ ①～②の作業を繰り返し、記事を増やしていきます。ここでは全部で14件の記事を投稿しています。コンテンツの管理画面には投稿した記事とカテゴリーの両方がリストアップされますが、記事のみをリストアップする場合、左側のメニューから「CONTENT TYPE > BlogPost」を選択します。

以上で、記事の投稿は完了です。

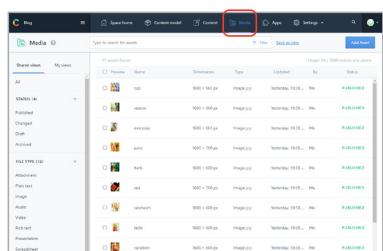


Contentful Content management interface showing a list of 14 published BlogPost entries. The 'Content' tab is selected in the top navigation bar, and the 'BlogPost' content type is selected in the left sidebar. A red box highlights the 'Content' tab and the 'BlogPost' content type selection.

コンテンツの
管理画面。

画像の管理

アップロードした画像は「Media」で
管理されます。



Contentful Media management interface showing a list of uploaded images. The 'Media' tab is selected in the top navigation bar. A red box highlights the 'Media' tab.

画像でエラーを回避する

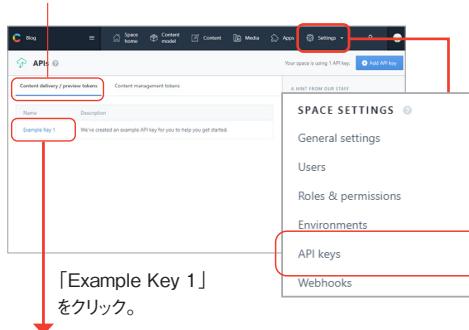
Contentfulのデータを扱うプラグイン
(gatsby-source-contentful) では、画像
が1つもアップロードされていないとエ
ラーが出ます。
そのため、少なくとも1つは画像をアッ
プロードしておきます。

4-6 トークンの確認

Contentfulのデータを利用するためには各種トークンが必要になりますので、確認しておきます。

スペースIDとアクセストークンを確認する

「Content delivery / preview tokens」タブ



「Example Key 1」
をクリック。

上のメニューから「Settings > API keys」を選択し、APIの管理画面を開きます。まずは、「Content delivery / preview tokens」のタブが開きます。

すでにある「Example Key 1」を利用して構いませんし、右上の「Add API key」をクリックして新規に作成してもよいので、以下のトークンを確認します。

The screenshot shows the 'Example Key 1' details page. It includes fields for Name (Example Key 1), Description (We've created an example API key for you to help you get started.), and three access token fields: Space ID, Content Delivery API - access token, and Content Preview API - access token. Each token field has a copy icon to its right.

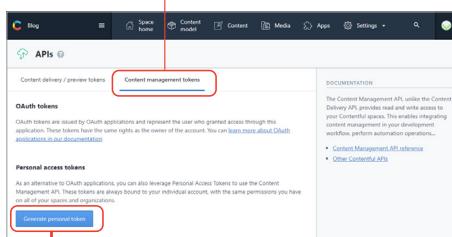
Space ID
スペースID

Content Delivery API - access token
アクセストークン

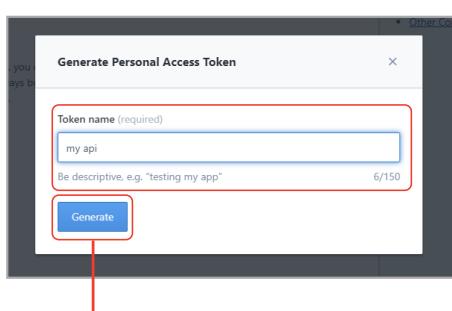
Content Preview API - access token
アクセストークン(プレビュー用)

パーソナルアクセストークンを作成する

「Content management tokens」タブ



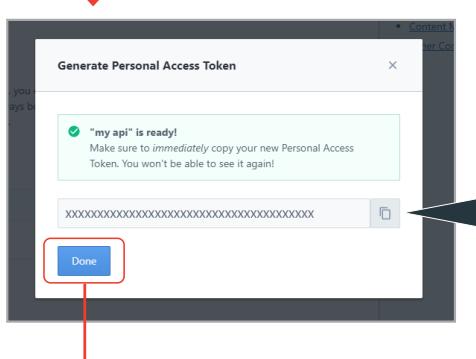
「Generate Personal Token」をクリック。



Contentful のデータをインポートする場合には、Personal access token (パーソナルアクセストークン) が必要になりますので、作成しておきます。

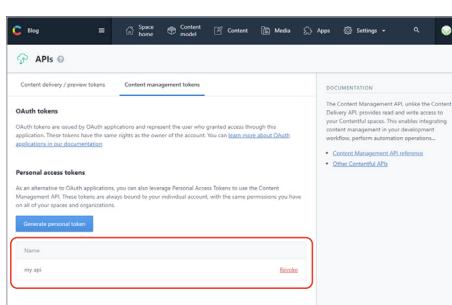
API の管理画面に戻り、「Content management tokens」のタブに切り替え、「Generate Personal Token」をクリックします。

トークン名を指定し、「Generate」をクリックします。ここでは「my api」と指定しています。



トークンが表示されます。この画面を閉じると、もう一度確認することはできなくなります。確認できたら「Done」をクリックして画面を閉じます。

Personal Access Token
パーソナルアクセストークン

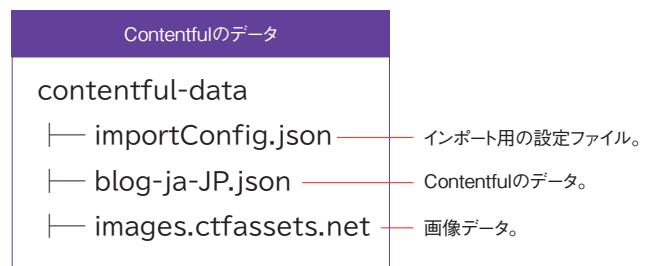


元の画面に戻ると、作成したトークンが追加されていることがわかります。

以上で、トークンの確認は完了です。

4-7 Contentfulのデータをインポートする

本PDFの4-3から4-5で設定・投稿したContentfulのデータは、本書ダウンロードデータの「contentful-data」フォルダに収録しています。ここではこのデータをインポートする手順を見ていきます。



- 4-2の手順でスペースを作成し、ロケールを「ja-JP」に変更しておきます。ロケルが異なる場合はインポートできません。
- 「contentful-data」フォルダ内にある importConfig.json を開きます。4-6の手順で確認したスペースIDを「Space ID」で、パーソナルアクセストークンを「managementToken」で指定します。

```
{
  "spaceId": "xxxxxxxxxxxx",
  "managementToken": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "contentFile": "blog-ja-JP.json"
}
```

importConfig.json

- インポートには contentful-cli を使用します。ここでは contentful-cli をインストールせずに使用するため、「contentful-data」フォルダ内で次のようにコマンドを実行していきます。まずは、Contentfulにログインします。--management-token ではパーソナルアクセストークンを指定します。

```
$ npx contentful-cli login --management-token xxxxxxxxxxxxxxxxxxxxxxx
```

④ インポートを実行します。

```
$ npx contentful-cli space import --config importConfig.json
```

インポートされるコンテンツタイプや記事の数などが示され、処理が完了すると「The import was successful.」と表示されます。

The following entities are going to be imported:	
Content Types	2
Editor Interfaces	2
Entries	17
Assets	17
Locales	1
Webhooks	0

✓ Validating content-file
 ✓ Initialize client (1s)
 ✓ Checking if destination space already has any content and retrieving it (3s)
 ✓ Apply transformations to source data (1s)
 > Push content to destination space
 ✓ Connecting to space (2s)
 " Importing Locales
 Importing Content Types
 Publishing Content Types
 Importing Editor Interfaces
 Importing Assets
 Publishing Assets
 Archiving Assets
 Importing Content Entries

✓ Creating Web Hooks (1s) Finished importing all data	
Imported entities	
Locales	1
Content Types	2
Editor Interfaces	2
Assets	17
Published Assets	17
Archived Assets	0
Entries	17
Published Entries	17
Archived Entries	0
Webhooks	0

The import took a minute (63s)
 No errors or warnings occurred
 The import was successful.
 [moniker@workserver contentful-data]\$

⑤ Contentful からログアウトします。

```
$ npx contentful-cli logout
```

ログアウトするか確認されますので、「Y」と入力します。以上で、インポートの作業は完了です。

This will log you out by deleting the CMA token stored on your system.
 ? Do you want to log out now? (y/n) Y

This will log you out by deleting the CMA token stored on your system.
 ? Do you want to log out now? Yes
 Successfully logged you out.
 [moniker@workserver contentful-data]\$

contentful-cliをインストールして処理する場合

contentful-cli をインストールしても構わない場合は、

```
$ yarn global add contentful-cli
```

で、contentful-cli をインストールしてください。

インポートの処理は「npx」を付けずに、

```
$ contentful-cli login --management-token xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
$ contentful space import --config importConfig.json
```

```
$ contentful logout
```

で完了します。

画像のアップロードがタイムアウトした場合

インポート中に「AssetProcessingTimeout」と表示され、画像の処理がタイムアウトするケースがあります。

Contentful で「Media」を開き、タイムアウトした画像の Status が「DRAFT」となっている場合は、「PUBLISH」に変更します。

インポートしなおす場合は時間帯を変えて試してみてください。

Preview	Name	Dimensions	Type	Updated	By	Status
4 assets selected: Archive Delete Publish						
	everyday	1600 × 661 px	Image (jpg)	7 minutes ago	Me	PUBLISHED
	variation	1600 × 600 px	Image (jpg)	7 minutes ago	Me	PUBLISHED
	herb	1600 × 600 px	Image (jpg)	7 minutes ago	Me	DRAFT
	sandwich	1600 × 600 px	Image (jpg)	7 minutes ago	Me	DRAFT
	table	1600 × 600 px	Image (jpg)	7 minutes ago	Me	DRAFT
	red	1600 × 700 px	Image (jpg)	7 minutes ago	Me	DRAFT

■著者

エビスコム

<https://ebisu.com/>

さまざまなメディアにおける企画制作を世界各地のネットワークを駆使して展開。コンピュータ、インターネット関係では書籍、デジタル映像、CG、ソフトウェアの企画制作、WWWシステムの構築などを行う。

主な編著書：『CSS グリッドレイアウト デザインブック』マイナビ出版刊
『HTML5&CSS3 デザイン 現場の新標準ガイド』同上
『6ステップでマスターする「最新標準」HTML+CSS デザイン』同上
『WordPress レッスンブック 5.x 対応版』ソシム刊
『フレキシブルボックスで作る HTML5&CSS3 レッスンブック』同上
『CSS グリッドで作る HTML5&CSS3 レッスンブック』同上
『HTML&CSS コーディング・プラクティスブック 1』エビスコム刊
『HTML&CSS コーディング・プラクティスブック 2』同上
『グーテンベルク時代の WordPress ノート テーマの作り方（入門編）』同上
『グーテンベルク時代の WordPress ノート テーマの作り方（ランディングページ＆ワンカラムサイト編）』同上
ほか多数

Web サイト高速化のための 静的サイトジェネレーター活用入門【副読本】

セットアップ PDF

2020 年 6 月 1 日 ver.1.0 発行