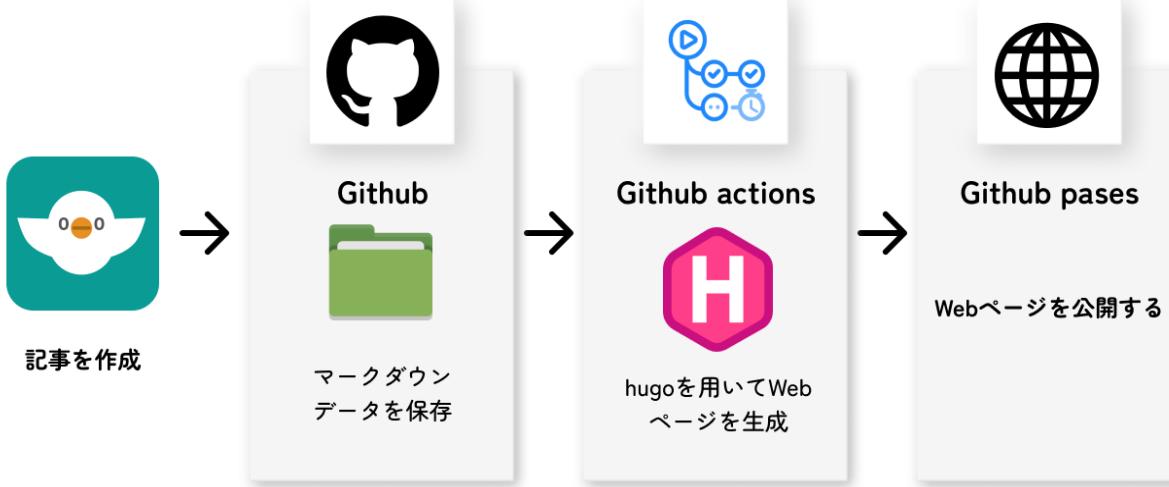


esaをCMSにして GithubActionsと hugoを用いて自動で ホームページを 更新する方法の考案



writer: はるちろ

目次

第 1 章 初めに	3
第 2 章 実際の動作しているサイト	4
第 3 章 用語まとめ	5
3.1 esa	5
3.2 hugo	6
3.3 github Actions	7
3.4 github pages	7
3.5 CMS	8
第 4 章 技術構成	9
第 5 章 hugo について	10
5.1 勉強資料	10
5.2 環境構築	10
5.3 hugo プロジェクト作成	10
5.4 テーマの設定	12
第 6 章 githubpages の設定（デプロイをする）	18
6.1 gitmodules の設定	18
6.2 baseurl の設定	19
6.3 静的なファイルを生成する	20
6.4 github pages の設定をする	23
第 7 章 esa について	25
7.1 設定	25
第 8 章 github Actions の設定	29
第 9 章 動作実験	33
9.1 esa で記事を書く	33
9.2 github の状態	34
9.3 githubActions の状態	34

1

初めに

さて、みなさん。Web サイトは作られていますか？Web サイトを作る時にそのまま HTML を触って Github などで管理をしてもいいのですが、やはり Github を使ったことない人にとっては Web サイトの更新だけでかなり大変な作業になってしまいます。

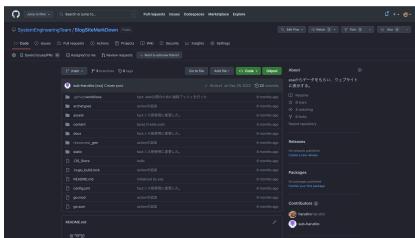
そこで、今サークルの情報共有で用いている esa を用いて Web サイトを更新したら簡単に、誰でも更新できるのではないかと考え、実装してみることにしました。そこまで難しくないので、ぜひ参考にしてみてください。

2

実際の動作しているサイト

現在、下記レポジトリ、サイトで実際に動作しています。実際に esa から記事を取り出して、Github に push しているので、記事を更新するだけで、Web サイトが更新されます。

レポジトリ: <https://github.com/SystemEngineeringTeam/BlogSiteMarkDown>
Web サイト: <https://esa.harutiro.net/>



▲ 図 2.1 Git レポジトリ



▲ 図 2.2 Web サイト

3

用語まとめ

3.1 esa

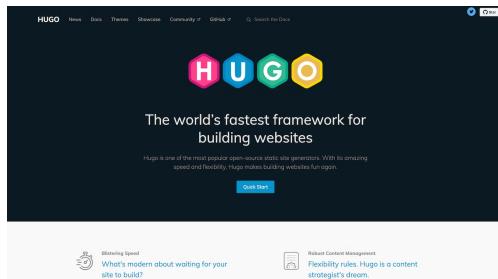


▲ 図 3.1 esa のホームページの写真

esa とは

esa とは、2014 年に設立された合同会社 esa の「情報共有ツール」です。esa は「不完全でも早い段階でチームに共有し、更新を重ねることでより良い情報に育つ」という発想のもと生まれました。そのため「Share（公開）」「Develop（更新して情報を育てる）」「Organize（育った情報を整理）」の 3 つの流れで設計されています。現在は 3,000 社を超える企業に導入されており、主に情報の蓄積や WIP 機能（書いている途中でも共有する機能）を用いて、業務の効率化を実現している企業が多いです。引用：
<https://www.stock-app.info/media/?p=37138>

3.2 hugo



▲ 図 3.2 hugo のホームページの写真

hugo とは

Hugo は Go 言語で実装された「Web サイト構築フレームワーク」で、最初の公開は 2013 年という比較的新しいツールだ。コンテンツ管理システムではなく「Web サイト構築フレームワーク」と名乗っているとおり、コンテンツの管理ではなく、Web サイトで使われる HTML ファイルや RSS ファイルなどの生成に特化した機能を備えている。引用：<https://knowledge.sakura.ad.jp/22908/>

3.3 github Actions



▲ 図 3.3 GitHubActions のホームページの写真

github pages とは

GitHub Actions で、ソフトウェア開発ワークフローをリポジトリの中で自動化し、カスタマイズし、実行しましょう。CI/CD を含む好きなジョブを実行してくれるアクションを、見つけたり、作成したり、共有したり、完全にカスタマイズされたワークフロー内でアクションを組み合わせたりできます。引用：<https://docs.github.com/ja/actions>

3.4 github pages

hugo とは

GitHub Pages は、GitHub のリポジトリから HTML、CSS、および JavaScript ファイルを直接取得し、任意でビルドプロセスを通じてファイルを実行し、ウェブサイトを公開できる静的なサイトホスティングサービスです。引用：<https://docs.github.com/ja/pages/getting-started-with-github-pages/about-github-pages>

3.5 CMS

hugo とは

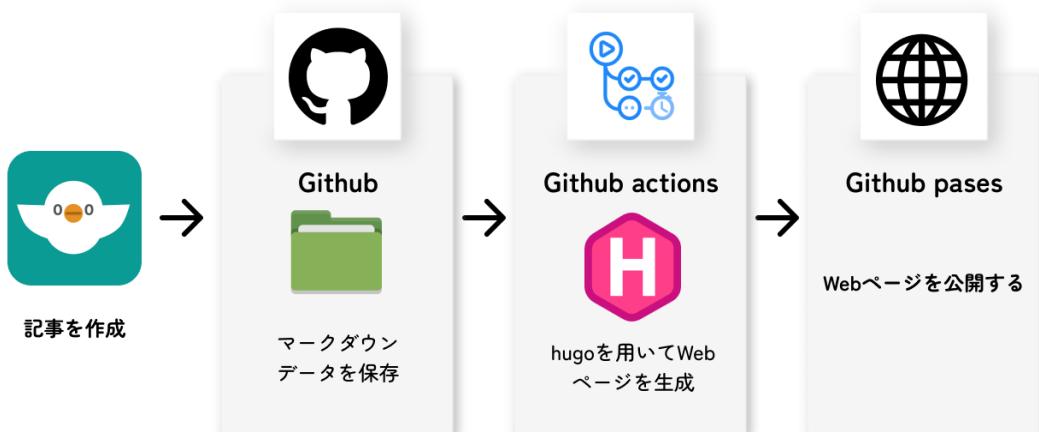
「CMS」とは、「Contents Management System：コンテンツ・マネジメント・システム」の略で、簡単にいうと Web サイトのコンテンツを構成するテキストや画像、デザイン・レイアウト情報（テンプレート）などを一元的に保存・管理するシステムのことです。引用：https://www.hitachi-solutions.co.jp/digitalmarketing/sp/column/cms_v0101/

4

技術構成

基本的には、esa によりマークダウンを作成して、それを Github に保存して、GithubActions を用いて hugo に出力して、Web サイトに公開する方法です。

ほぼほぼノーコードでできる構成になっているので、そこまで手間がかからずに入力することができます。



▲ 図 4.1 実際に用いた技術構成の図

5

hugoについて

hugoは静的なWebジェネレーターということで、軽く環境構築をしていきましょう。

5.1 勉強資料

https://youtu.be/hjD9jTi_DQ4

5.2 環境構築

とりあえず、hugoをインストールしてみましょう。

```
$ brew install hugo
```

たったこれだけで完成です。

5.3 hugoプロジェクト作成

あらかじめ、レポジトリを作成しておきましょう。githubからクローンをして、レポジトリを持ってきてください。

```
$ git clone 自分のリポジトリの URL
```

```
$ cd クローンしたディレクトリ
```

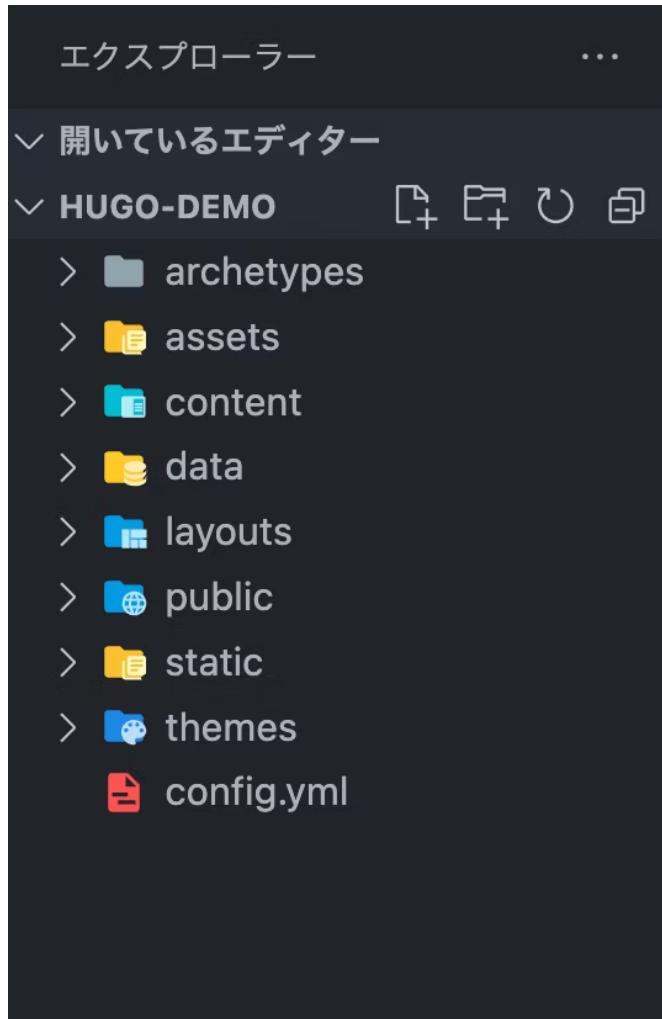
では、次に hugo のプロジェクトを作成します。

```
$ hugo new site hugo-demo -f yml
Congratulations! Your new Hugo site is created in
/Users/k22120kk/Downloads/hugo-demo.
```

Just a few more steps and you're ready to go:

1. Download a theme into the same-named folder.
Choose a theme from <https://themes.gohugo.io/> or
create your own with the "hugo new theme <THEMENAME>" command.
2. Perhaps you want to add some content. You can add single files
with "hugo new <SECTIONNAME>/<FILENAME>. <FORMAT>".
3. Start the built-in live server via "hugo server".

Visit <https://gohugo.io/> for quickstart guide and full documentation.



▲ 図 5.1 実際に用いた技術構成の図

5.4 テーマの設定

5.4.1 テーマの選択

hugo ではテーマを用いることで簡単にそれっぽい Web ページを作成することができます。<https://themes.gohugo.io/> この中から気に入ったテーマを探してみてください。注意：テーマによって若干設定項目が変わる場合があるので気をつけてください。では今回は下記のテーマを用いて作成していくかと思います。<https://themes.gohugo.io/themes/hugo-theme-stack/>

ちなみに、各テーマに大体ドキュメントがありますので、その started などをみることをお勧めします。今回のドキュメントはこちらになります。<https://stack.jimmycai.com/>

5.4.2 テーマの反映

では、ドキュメントの指示に従いながら、テーマの反映を行っていきます。<https://stack.jimmycai.com/guide/getting-started>

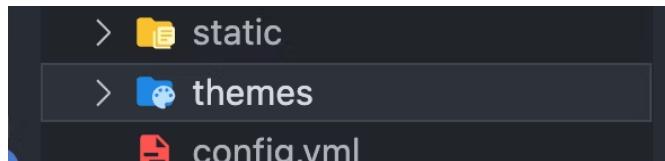
cd を用いて、hugo のプロジェクトページまできてください。下記を実行することでテーマを反映させることができます。

```
$ git submodule add https://github.com/CaiJimmy/hugo-theme-stack/
themes/hugo-theme-stack
```

git init をしていない場合はこちらで実行してみてください。

```
$ git clone https://github.com/CaiJimmy/hugo-theme-stack/
themes/hugo-theme-stack
```

そうすることにより、themes ディレクトリが作成されて、テーマの素材が入ります。



▲ 図 5.2 themes フォルダーができている様子の図

では、config ファイルを設定して、theme を設定しましょう

```
baseURL: ""
languageCode: ja
title: My New Hugo Site
theme: hugo-theme-stack
```

では、実際に動かしてみて、動作するか確認してみましょう。

```
$ hugo server
Start building sites ...
hugo v0.108.0+extended darwin/arm64 BuildDate=unknown
WARN 2022/12/23 10:44:43 found no layout file for "HTML" for kind
"taxonomy": You should create a template file which matches Hugo
Layouts Lookup Rules for this combination.
WARN 2022/12/23 10:44:43 found no layout file for "HTML" for kind
"home": You should create a template file which matches Hugo
Layouts Lookup Rules for this combination.
WARN 2022/12/23 10:44:43 found no layout file for "HTML" for kind
"taxonomy": You should create a template file which matches Hugo
Layouts Lookup Rules for this combination.
```

| EN

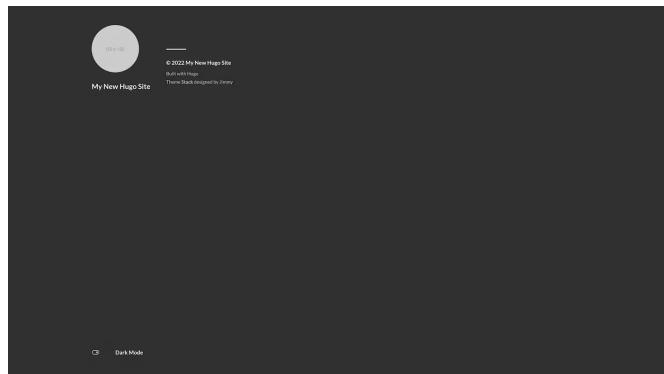
	-----+
Pages	3
Paginator pages	0
Non-page files	0
Static files	0
Processed images	0
Aliases	0
Sitemaps	1
Cleaned	0

```
Built in 6 ms
Watching for changes in /Users/k22120kk/Downloads/hugo-demo/
{archetypes,assets,content,data,layouts,static}
Watching for config changes in /Users/k22120kk/Downloads/
hugo-demo/config.yml
Environment: "development"
Serving pages from memory
Running in Fast Render Mode. For full rebuilds on change:
hugo server --disableFastRender
Web Server is available at http://localhost:1313/
```

```
(bind address 127.0.0.1)
```

```
Press Ctrl+C to stop
```

ウェブブラウザで `http://localhost:1313/` を URL に入れて読み込んでみましょう。
下のような画像が出力されれば成功です。



▲ 図 5.3 テーマが反映された様子の図

5.4.3 Web ページを作成してみる

では、軽くマークダウンを作成をして Web ページを作成してみましょう下のコマンドを入力して、マークダウンを作成してみましょう。

```
hugo new post/first.md
```

そうすると下記のようなファイルができているはずです。

```
---
```

```
title: "First"
```

```
date: 2022-12-23T10:49:46+09:00
```

```
draft: true
```

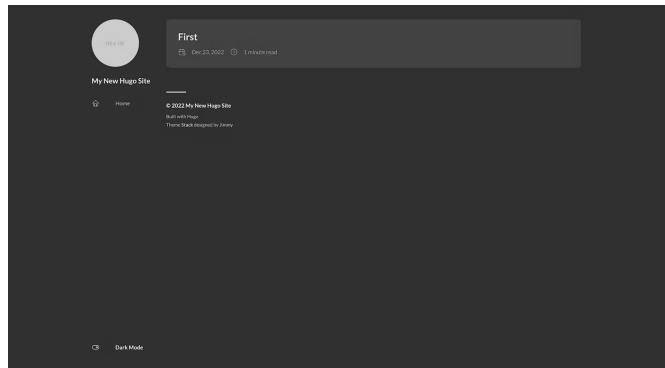
```
--
```

それでは追加で書き加えてみます。ここで状態が編集モードになっているため、draft:trueをdraft:falseに変更しましょう。こうしないと、デプロイした時に表示されなくなります。

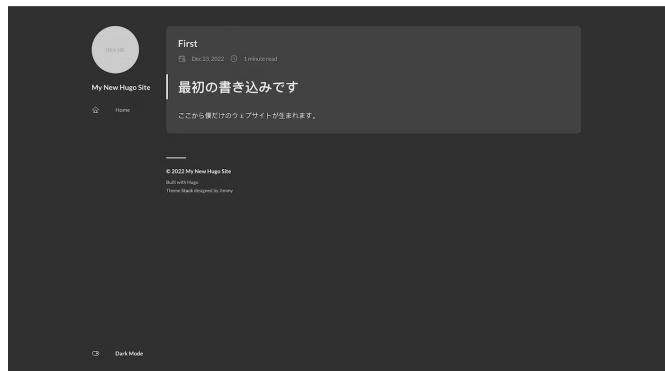
```
---  
title: "First"  
date: 2022-12-23T10:49:46+09:00  
draft: false  
---
```

```
# 最初の書き込みです  
ここから僕だけのウェブサイトが生まれます。
```

下のように、作成したWebページが作られているはずです。



▲ 図 5.4 ファイルが追加されている様子の図



▲ 図 5.5 ファイルの中身が追加されている様子の図

5.4.4 細かい設定など

ここでは、細かく説明はしませんが、config ファイルや、md ファイルの上の部分で様々な設定をすることができます。詳しい説明をしてくれている Web ページを紹介しますので、ぜひご自身で色々触って自分だけのオリジナルサイトを作ってみてください。

config の設定

<https://github.com/CaiJimmy/hugo-theme-stack/blob/master/exampleSite/config.yaml>

<https://miiitomi.github.io/p/hugo/>

フロントマターの設定

<https://takaken.tokyo/dev/hugo/post/write-post/>

6

githubpages の設定（デプロイをする）

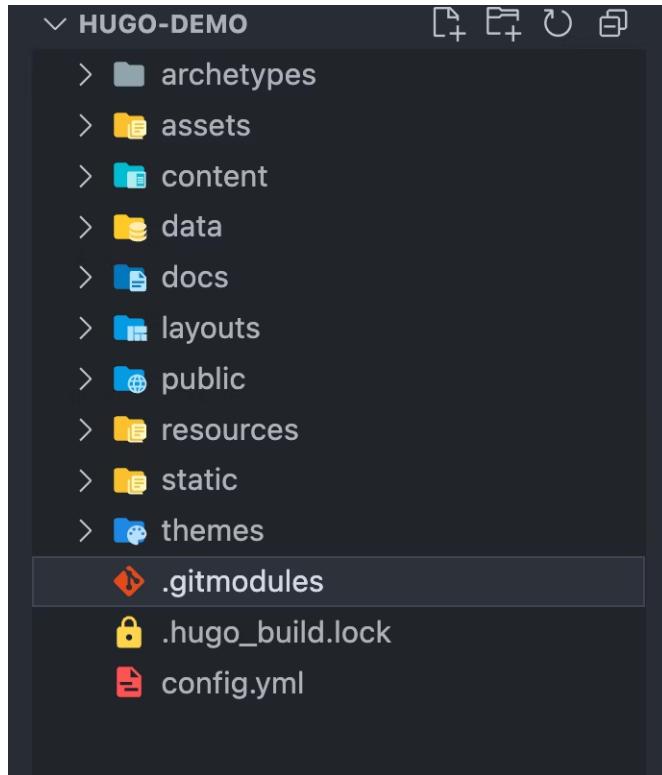
6.1 gitmodules の設定

gitmodules とは

Git サブモジュールは、あるリポジトリの内容を別のリポジトリ内に含めることを、参照されるリポジトリの場所を指定するだけでできるようになる Git SCM の機能です。これは、外部ライブラリのソースをアプリケーションのソースツリーに含めるメカニズムを提供します。引用：
<https://devcenter.heroku.com/ja/articles/git-submodules#:~:text=Git%20%E3%82%B5%E3%83%96%E3%83%A2%E3%82%B8%E3%83%A5%E3%83%BC%E3%83%AB%E2%80%8B%E3%81%AF,%E3%83%A1%E3%82%AB%E3%83%8B%E3%82%BA%E3%83%A0%E3%82%92%E6%8F%90%E4%BE%9B%E3%81%97%E3%81%BE%E3%81%99%E3%80%82>

ということで、デプロイする時に、この設定をしないとエラーが発生するので、設定をしましょう

```
$ touch .gitmodules
```



▲ 図 6.1 GitModule を作った時の Tree 構造

テーマによって、設定内容が変わってくるので、テーマに合わせた設定にしてください。

```
[submodule "themes/hugo-theme-stack"]
path = themes/hugo-theme-stack
url = https://github.com/CaiJimmy/hugo-theme-stack
```

6.2 baseurl の設定

あとは baseURL を指定しましょう。ここが正しくないと、CSS がうまいこと読み取られないなどのバグが発生します。

ということで、config ファイルを編集しましょう。

今回は、自分の設定をそのまま表示させていますが、カスタムドメインを使っていない場合は、<https://github> のユーザー名.github.io/レポジトリ名/と設定しましょう。カスタムドメインを使用している場合は、設定したドメインをそのまま書き込んでください。

```
baseURL: "https://harutiro.github.io/hugo_test_qiita/"  
languageCode: ja  
title: My New Hugo Site  
theme: hugo-theme-stack  
publishDir: "docs"  
  
menu:  
  main:  
    - identifier: home  
      name: Home  
      url: /  
      params:  
        icon: home
```

6.3 静的なファイルを生成する

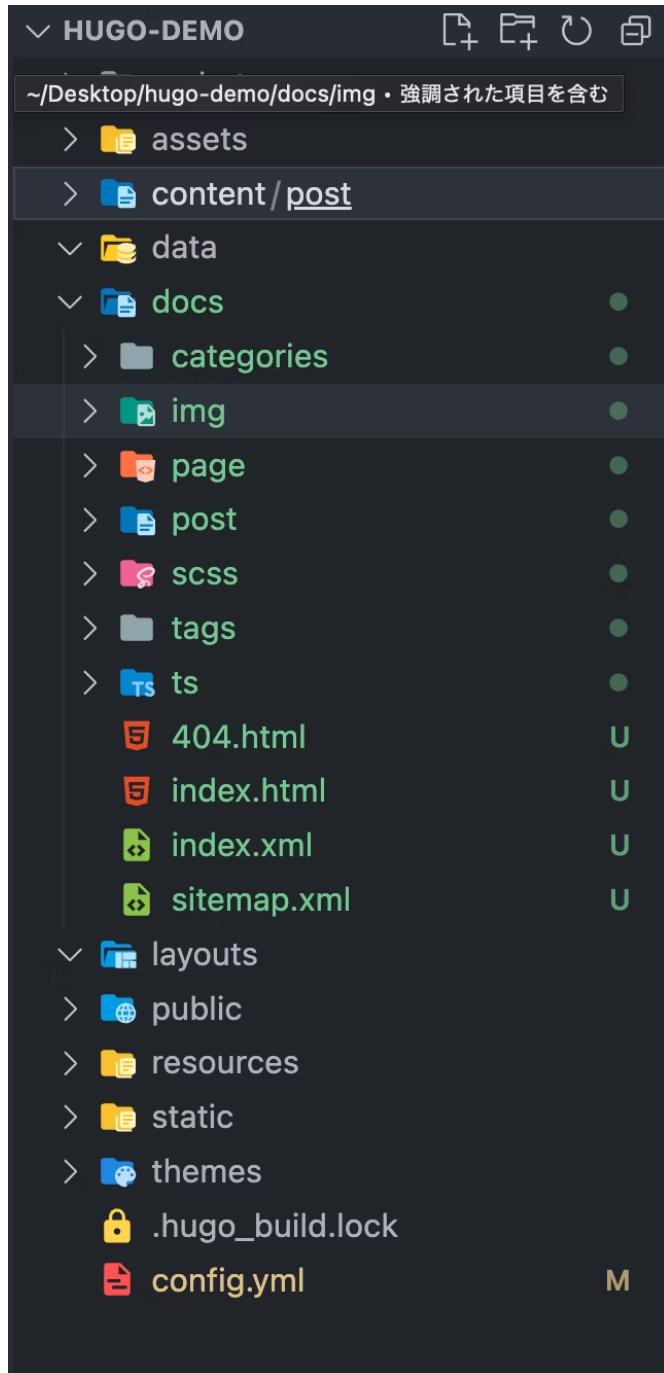
それでは、githubpages の公開するために、静的なファイルを生成します。生成する時に docs に作ってもらえると都合がいいので、config を設定しておきましょう。

publishDir: "docs" を追記してあげてください。

```
baseURL: ""  
languageCode: ja  
title: My New Hugo Site  
theme: hugo-theme-stack  
publishDir: "docs"  
  
menu:  
  main:  
    - identifier: home  
      name: Home  
      url: /  
      params:  
        icon: home
```

あとは下記のコマンドを打ち込んで静的なファイル (html,css など) を作成しましょう

```
$ hugo
```

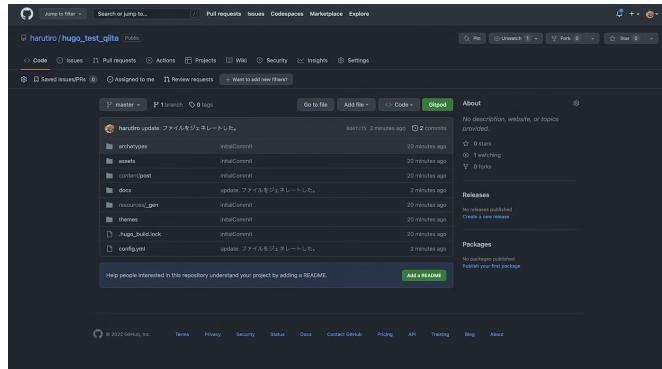


▲ 図 6.2 docs に静的なファイルが生成された図

6.4 github pages の設定をする

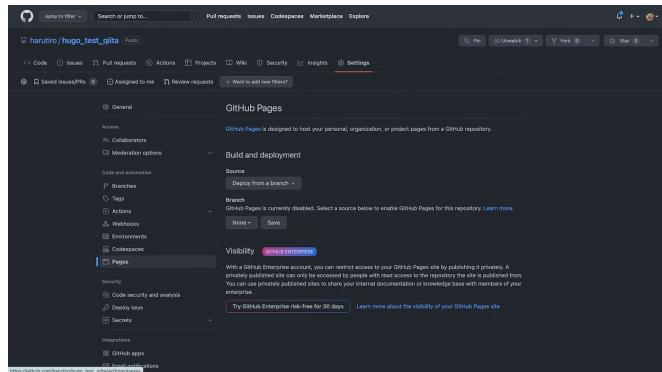
それでは、github を用いてデプロイを行ってみましょう。とりあえず、いつもの手順でgithub に公開しましょう。

```
$ git init  
$ git add -A  
$ git commit -m "initialCommit"  
$ git remote add origin 個人のレポジトリ  
$ git push origin master
```



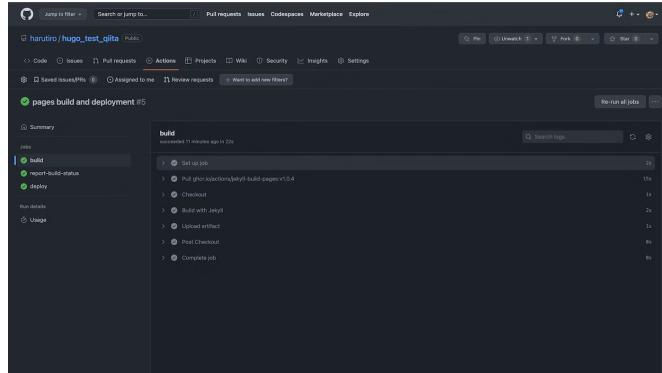
▲ 図 6.3 レポジトリが作成された様子

それでは、pages の設定をしていきましょう。setting/pages を開きましょう。



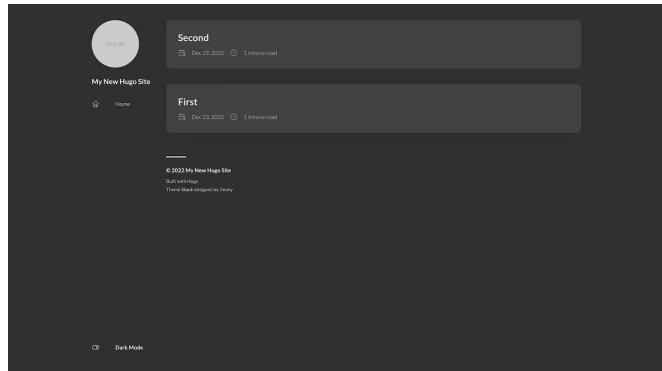
▲ 図 6.4 pages の設定画面

source を Deploy from a branch に設定して Branch を master /docs に設定しましょう
カスタムドメインは今回は説明しません。これで、master にデプロイして、しばらく待つとデプロイが始まります。



▲ 図 6.5 デプロイが始まった様子

うまくいった場合、設定した baseURL の場所に行くとうまく表示されているはずです。



▲ 図 6.6 公開がうまくいった様子

7

esaについて

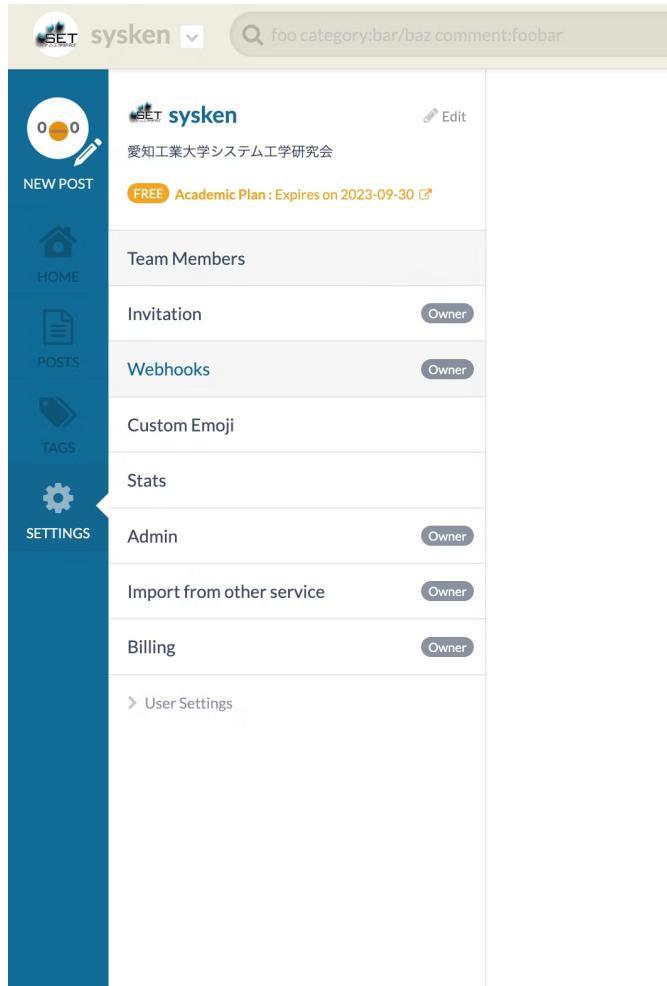
esa ではいつかのアップデートで、直接 Github に記事を保存ができるようになりました。
2016 年のアップデートらしいですね。細かい設定などは下の記事でご確認ください。

<https://docs.esa.io/posts/176>

では、とりあえず設定をしていきましょう。

7.1 設定

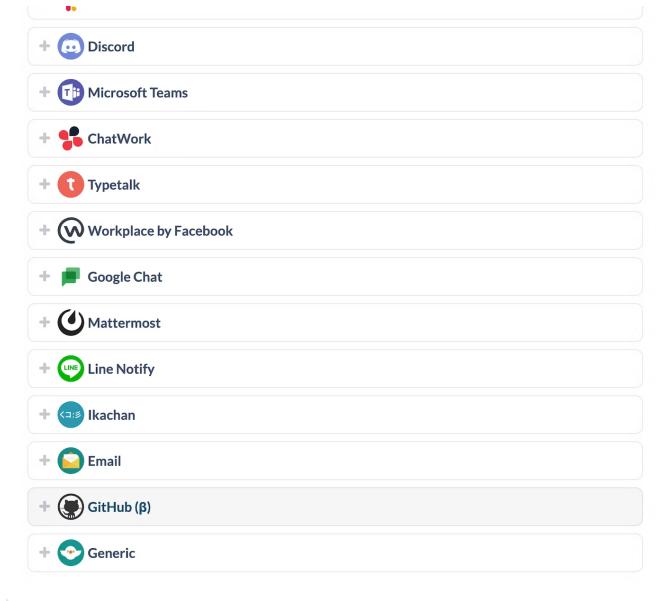
まずは、esa の設定をしましょう。



▲ 図 7.1 esa の設定画面

setting -j Webhooks を選択します。

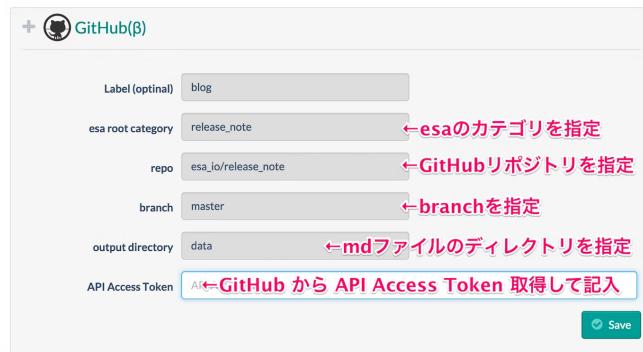
add を選択して、Github(β) を選択します。



▲ 図 7.2 esa の設定画面 API 連携の部分

あとは、様々な設定を行って、保存をします。

esa root category ここで、Web ページに更新されて欲しい記事のディレクトリを選択することもできるので設定してみてください。output directory 出力の場所は hugo の関係で指定しないといけないので、/content/post/と設定しましょう。



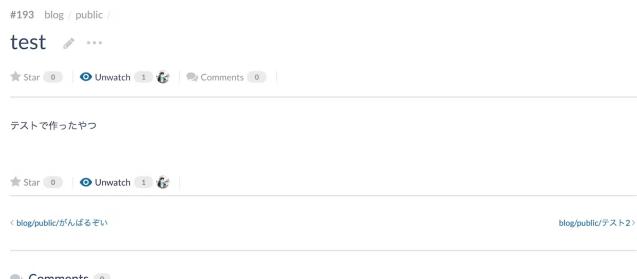
▲ 図 7.3 esa の設定画面 Github の設定



▲ 図 7.4 esa の設定画面 Github の設定 実際の設定項目

これで記事が更新（新規）で作られたら、Github のレポジトリの更新されました。では、実際に esa が更新されたら post に新しいファイルが追加されるか確認してみましょう。

こういったファイルを投稿してみました。



▲ 図 7.5 esa に投稿した記事

ファイルが新しく追加されていますね。



▲ 図 7.6 Github に記事が追加された

しかし、このままだと Web ページは更新されていません。それは hugo を用いて静的なファイルを生成していないためです。

では次の章では GitHubActions を用いて hugo を実行するプログラムを書いていきましょう。

8

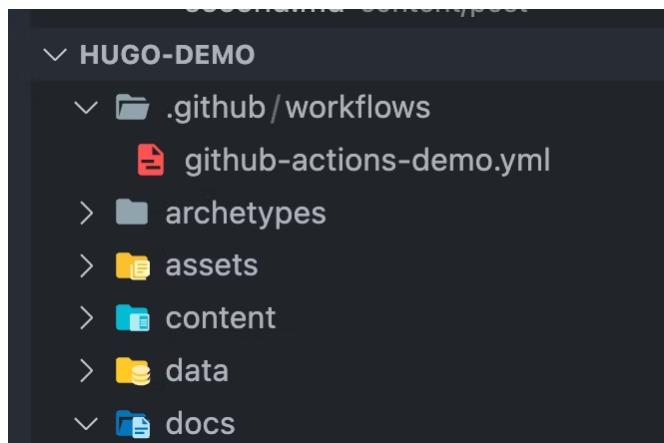
github Actions の設定

githubActions を設定していきましょう。

流れとしては、新しいブランチを作成して、その中に hugo を実行させて静的なファイルを作成して、強制プッシュを用いて更新をかけていくやり方です。無理くり動かしている雰囲気は感じますが、今はこのままでいきましょう。何かいい方法がありましたら、教えてください。

.github/workflows フォルダーを作成しましょうその中に github-actions-demo.yml として、ファイルを作成しておきましょう

```
$ mkdir .github
$ cd .github
$ mkdir workflows
$ cd workflows
$ touch github-actions-demo.yml
```



▲ 図 8.1 github-actions の設定ファイルを作成

中のファイルはこのように書き込んでいきましょう。一部人によって変わる場所があるので気をつけてください。

** 変えるもの ** 起爆させるブランチ名 コミットする時のユーザー名、ファイル名

```
name: Deploy Sakura Server

on:
  push:
    branches:
      - master # master ブランチが更新された時に発火させる

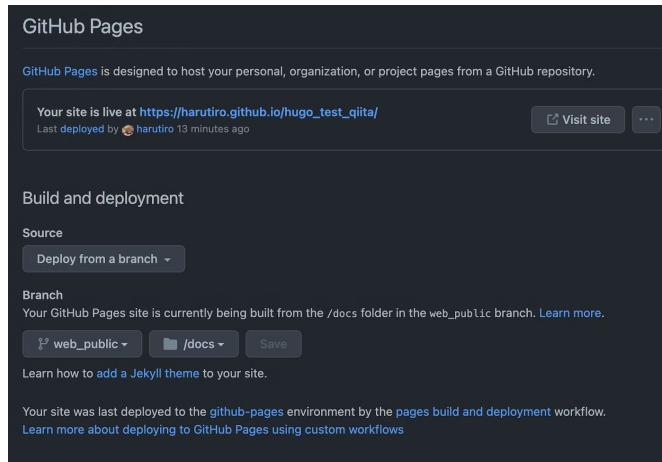
jobs:
  deploy:
    name: deploy
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
        with:
          submodules: true # Fetch Hugo themes (true OR recursive)
          fetch-depth: 0    # Fetch all history for .GitInfo and .Lastmod

      - name: Setup Hugo
        uses: peaceiris/actions-hugo@v2
```

```
with:  
  hugo-version: '0.102.3' # コンパイルに使用する Hugo の  
バージョンを指定  
  
  - name: Build  
    run: hugo --minify # 実際に Hugo でコンパイルする (--minify  
はファイルを圧縮するオプション)  
  
  - name: git commit  
    uses: EndBug/add-and-commit@v9  
    with:  
      author_name: harutiro # 投稿するユーザーに合わせてください。  
      author_email: hogehoge@example.com # 投稿するユーザー  
に合わせてください。  
      new_branch: web_public  
      message: 'hogehoge' # いい感じにメッセージを書いてあげてく  
ださい。  
      add: '* --force'  
      push: origin web_public --force
```

あとは、新しく githubpages の設定を書き換えて完成です。ブランチの位置を web_public に書き換えましょう。



▲ 図 8.2 github-pages の設定を書き換える

これで完成です。新しく push がされたタイミングで更新がかかるっていきます。

9

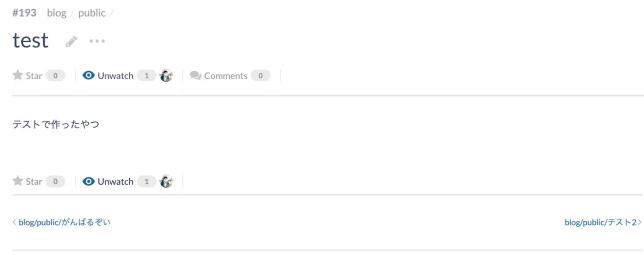
動作実験

9.1 esa で記事を書く

今回はこの二つを投稿したとします。



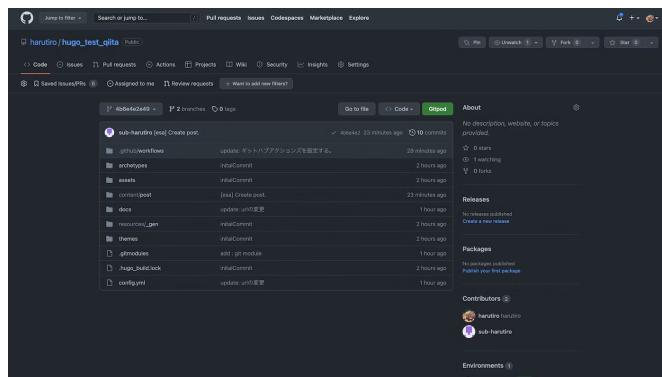
▲ 図 9.1 esa に投稿した記事 1



▲ 図 9.2 esa に投稿した記事 2

9.2 github の状態

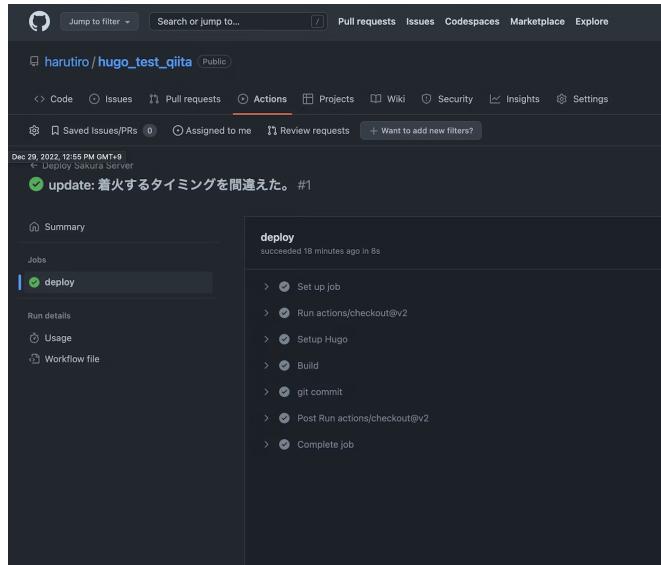
しばらくすると、esa の Webhooks が走って、md を保存してくれます。



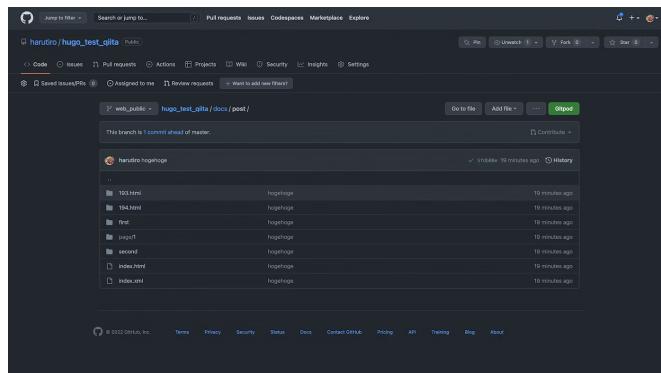
▲ 図 9.3 git のリポジトリの状態

9.3 githubActions の状態

あたらしく Push されると、GithubActions が走って、web_public のブランチに静的ファイルが生成されます。



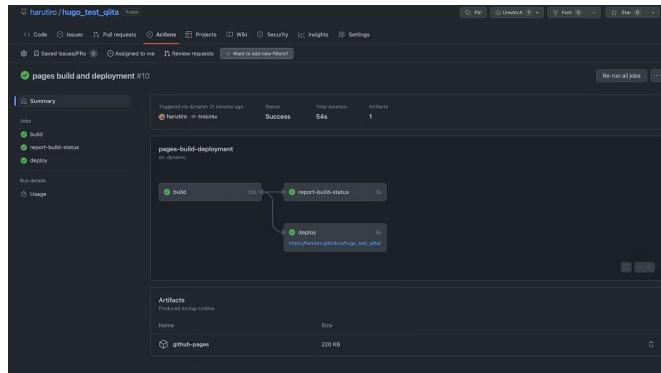
▲ 図 9.4 githubActions の状態



▲ 図 9.5 githubActions の状態 hugo の生成をしている状態

9.4 githubPases の状態

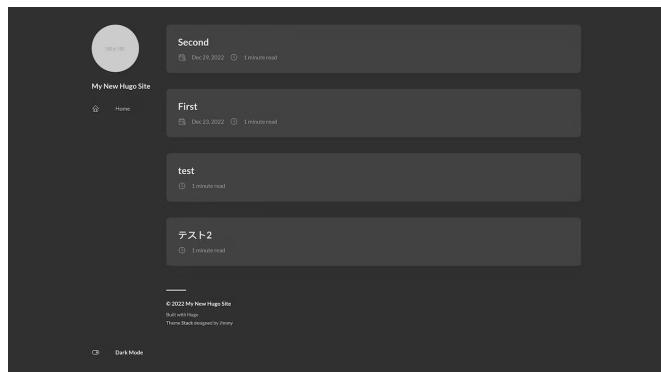
web_public に push されると GithubPases が走って、自動的にデプロイされます。



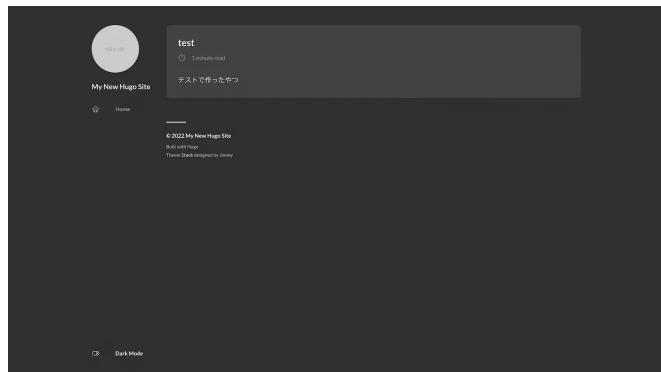
▲ 図 9.6 githubPases の状態

9.5 web ページの状態

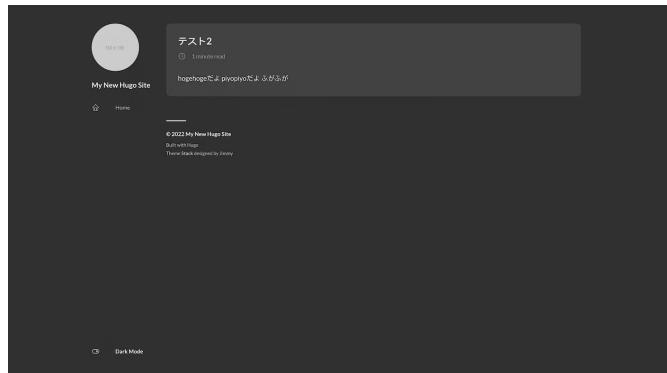
自動で更新がされていることが確認できました。



▲ 図 9.7 web ページの状態



▲ 図 9.8 web ページの状態 記事 1



▲ 図 9.9 web ページの状態 記事 2

10

終わりに

あまり esa を CMS 化しようという試みがないかもしれません、大変便利なものなのでぜひ使ってみてはいかがでしょうか？ 2022 年 /12 月時点では、esa は学生団体でしたら、1 年間無料で使って更新も無料らしいので、ぜひ使ってみてはいかがでしょうか？

11

参考文献

https://zenn.dev/ryo_kawamata/articles/4bf52f97165058

<https://nabinno.github.io/posts/67>

https://youtu.be/hjD9jTi_DQ4

11.0 奥付け

今回はこの本を手に取っていただきありがとうございました。企画・編集を行いました、牧野です。さて、10年ほど前のシス研では、本という形で技術をアウトプットするという文化があったと先輩から教えてもらいました。今では、Qiita や Zenn のような技術ブログが発達して、物理的なもので残す文化がちょっとずつ廃れてしまったりらしいです。そんな中、私が C101 の冬コミで技術本というものに出会いました、一つ一つ個人で調べたり研究した情報を、本という形で残すことにとってもいいと思い、今回このような本を執筆しました。本というものは知識としてずっと蓄えられるものですので、私たちが書いた本が誰かの役にたてればと思い、締めさせてもらいます。次回も余裕があれば出したいな。

企画・編集 牧野遙斗

esa を CMS にして GithubActions と hugo を用いて自動でホームページを更新する方法の考察

発行日	2023 年 5 月 28 日	(初版)
サークル	愛知工業大学 システム工学研究会	
Instagram ID	@ait.sysken	
Twitter ID	@set_official	
QiitaOrganizationURL	https://qiita.com/organizations/sysken	
代表	牧野遙斗	
代表者メールアドレス	harutiro2027@icloud.com	
著者	牧野遙斗 (Twitter: @minesu1224)	
印刷所	しまや出版	

※本書の無断複写、複製、データ配信はかたくお断りいたします。



writer: はるちろ