

# AWS放浪記

格安ホスティングを目指して

21卒エンジニア職 池田 力

## 背景

格安で、独自ドメインなサイトをちゃちゃっと作りたい！！

自分で自由に改造できるサイトを作りたかった

note, Qiita, はてなブログだとできてスタイル改造程度

AWSを活用すると安い(し、なんかかっこいい)

サーバーレスでサイトを構築すると格安でサイトを作ることができる

独自ドメインのサイトにしたい

( GitHub Actionsの有効活用 )

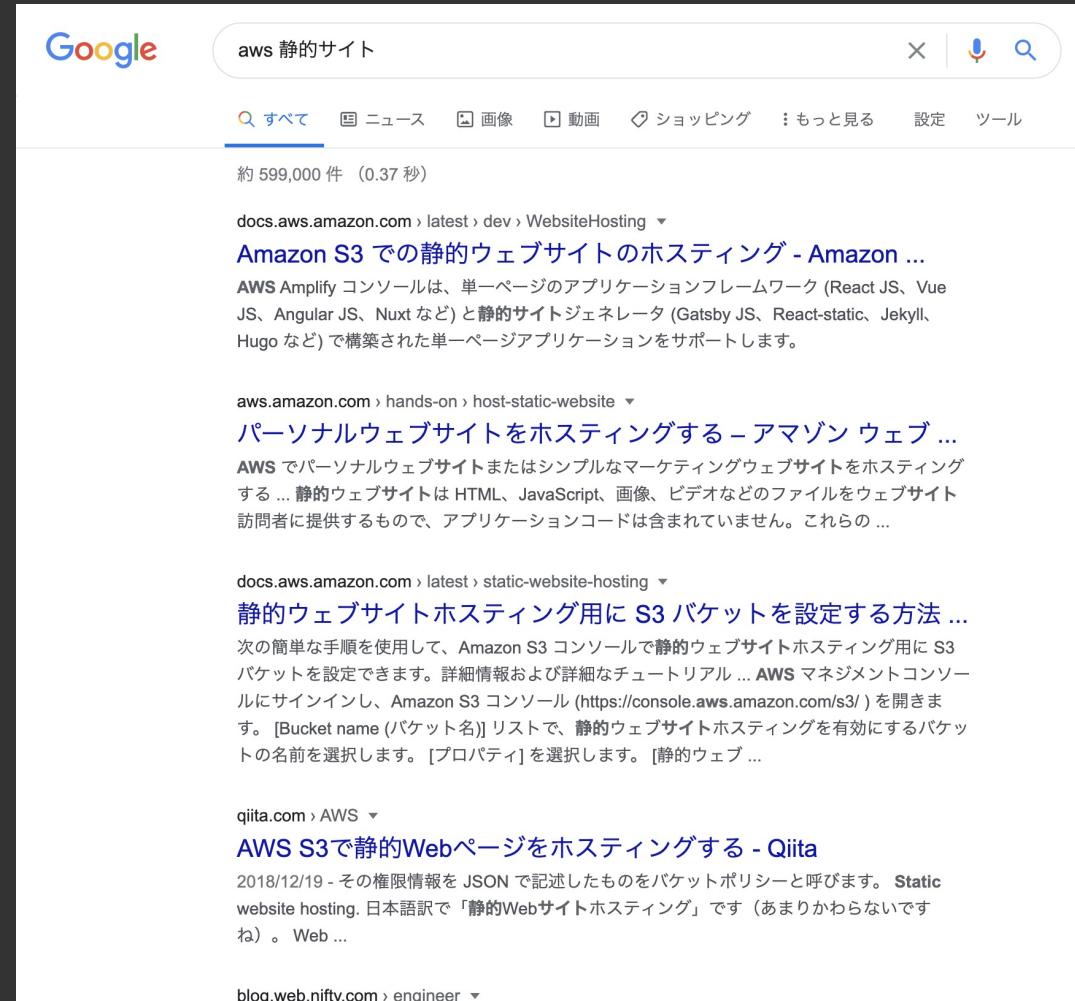
# どうやったか / 実際にやってみた結果

AWS編

フロントエンド編

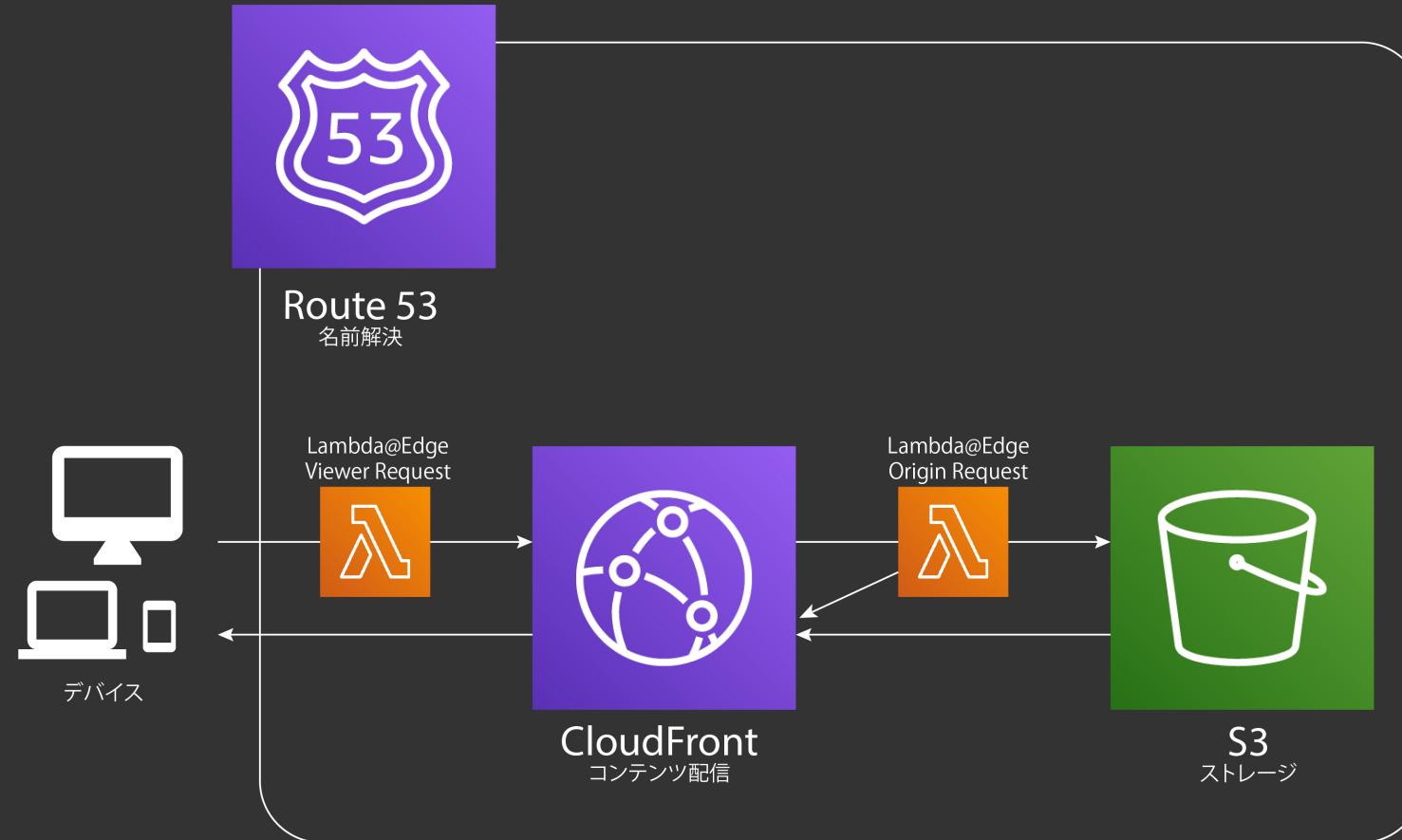
## どうやったか - AWS編

# Google検索するとわんさか情報が出てくる



# どうやったか - AWS編

## 最終的な構成



# どうやったか - AWS編

## この構成で実現できること

### 独自ドメイン

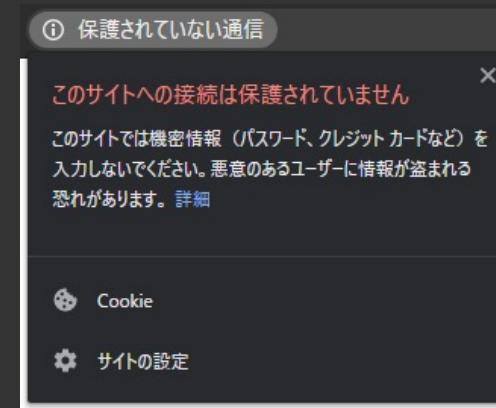
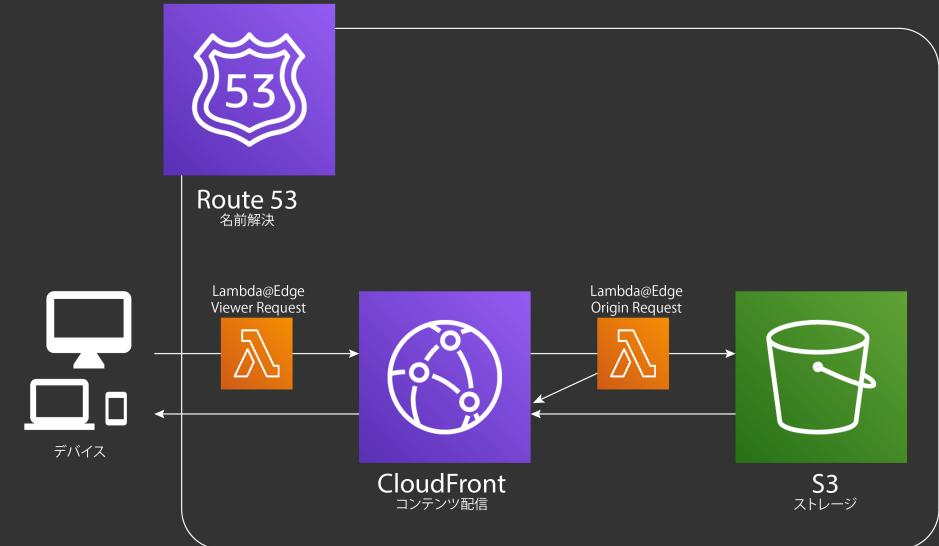
Route53での [tomtsutom.com](https://tomtsutom.com) の取得

### 常時SSL化

- ACMによるSSL証明書の取得
- CloudFrontでhttpからhttpsへリダイレクト

### OGP対応

Lambda@Edgeを用いて実現する



# 実際にやってみた結果 - AWS編

## Route53で独自ドメイン取得

昔やったことがあったので難なくクリア

The screenshot shows the AWS Route 53 console with the "Registered Domains" section selected. The main area displays a table with one row of data:

ドメイン名	プライバシーの保護	有効期限	自動更新	移管のロック
tomitsutom.com	すべての連絡先	2021/06/15	✓	✗

The left sidebar contains a navigation menu with the following items under the "Domains" section:

- 登録済みドメイン (selected)
- 保留中のリクエスト

Other sections in the sidebar include:

- ダッシュボード
- ホストゾーン
- ヘルスチェック
- トラフィックフロー
- トラフィックポリシー
- ポリシーレコード
- ルール
- リゾルバー
- VPC
- インバウンドエンドポイント
- アウトバウンドエンドポイント

# 実際にやってみた結果 - AWS編

## SSL証明書の取得・設定

ACMのSSL証明書はus-east-1で作る必要があるのに間違えた  
(この理由を突き止めるだけで3時間くらいは潰した... 😞)

The screenshot shows the AWS Certificate Manager console. On the left, there's a table listing two certificates:

	名前	ドメイン名	追加の名前	状況	種類
<input type="checkbox"/>	-	tomstutom.com	-	発行済み	Amazon が発行
<input type="checkbox"/>	-	*.tomstutom.com	-	発行済み	Amazon が発行

On the right, a sidebar lists AWS Regions and their corresponding endpoints:

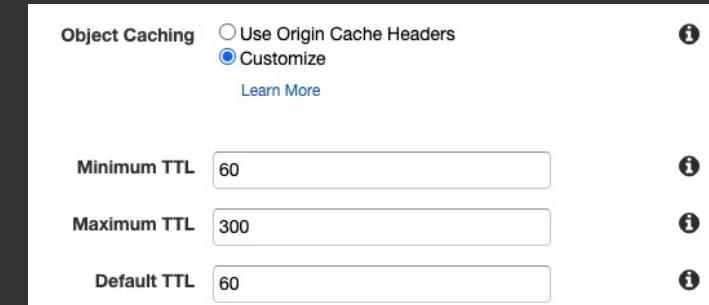
- 米国東部 (バージニア北部) us-east-1
- 米国東部 (オハイオ) us-east-2
- 米国西部 (北カリフォルニア) us-west-1
- 米国西部 (オレゴン) us-west-2
- アフリカ (ケープタウン) af-south-1
- アジアパシフィック (香港) ap-east-1
- アジアパシフィック (ムンバイ) ap-south-1
- アジアパシフィック (ソウル) ap-northeast-2
- アジアパシフィック (シンガポール) ap-southeast-1
- アジアパシフィック (シドニー) ap-southeast-2
- アジアパシフィック (東京) ap-northeast-1
- カナダ (中部) ca-central-1
- 欧州 (フランクフルト) eu-central-1
- 欧州 (アイルランド) eu-west-1
- 欧州 (ロンドン) eu-west-2
- 欧州 (ミラノ) eu-south-1
- 欧州 (パリ) eu-west-3
- 欧州 (ストックホルム) eu-north-1
- 中東 (バーレーン) me-south-1
- 南米 (サンパウロ) sa-east-1

# 実際にやってみた結果 - AWS編

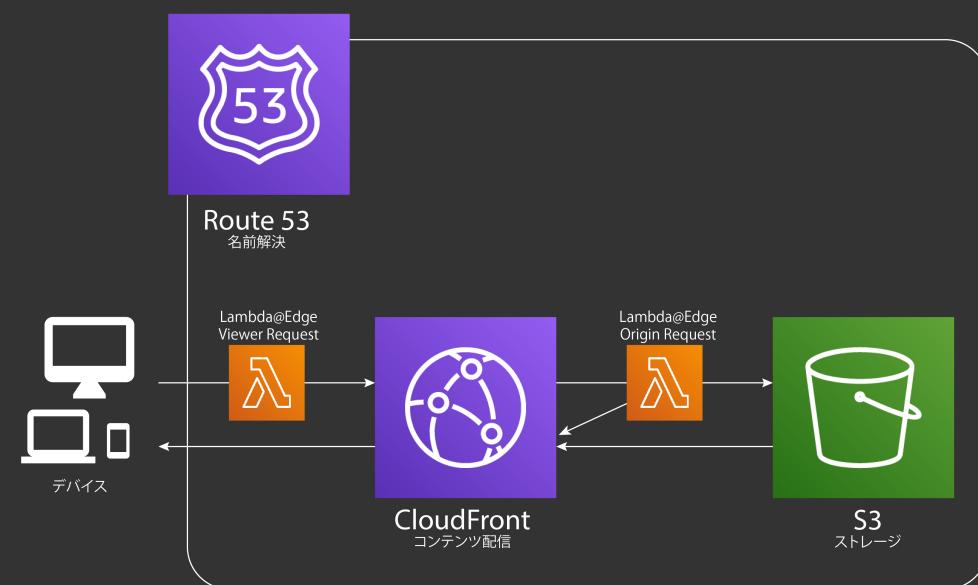
## CloudFrontのデフォルトキャッシュが1日で長過ぎる



デフォルト



キャッシュを1分に設定した場合



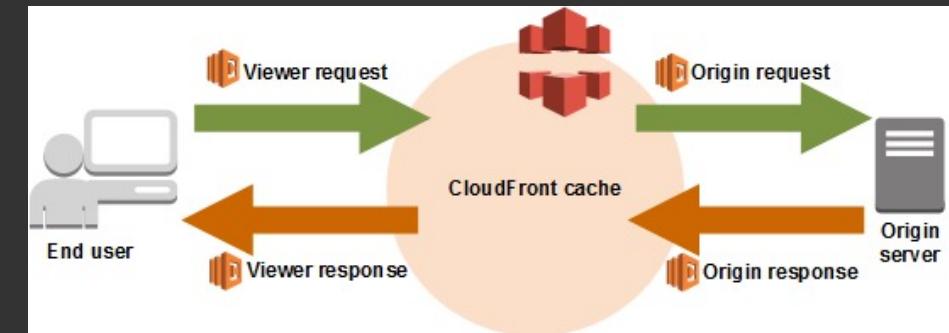
## 実際にやってみた結果 - AWS編

### OGP対応(Twitterでサイトの情報が表示されるやつ)

Lambda@Edgeでサーバーレスの弱点を克服！

### Lambda@Edgeのここがすごい

- リクエストに対して柔軟に対応できる
- 料金はコードが実行された分だけ
- Edgeで動くのでレスポンスが早い



## 実際にやってみた結果 - AWS編

### OGP対応

SSRをやめる。OGP対応はLambda@Edgeでダイナミックレンダリングする。

<https://qiita.com/geerpm/items/78e2b85dca3cb698e98d> を参考にして作った。

### 原理

ブラウザをLambda上で動かして、HTMLのコードを生成している。

しかし、ブラウザバージョンが古いなどの理由で色々動かなかった

### 結果

試行錯誤をした結果できたコードが以下の通り

<https://github.com/Tsutomu-Ikeda/origin-request-crawler>

<https://github.com/Tsutomu-Ikeda/viewer-request-fixer>

## 実際にやってみた結果 - AWS編

かかった費用は一ヶ月あたり \$1.58 ≈ 169円(2020年7月現在)

ドメイン取得費用 \$12.00 / 年

Route 53使用料 \$0.50 / 月

CloudFront使用料 \$0.03 / 月

Lambda@Edgeの料金

S3使用料 \$0.00 / 月

転送量があまり多くないので安い

消費税 \$0.05 / 月

どうやったか - フロントエンド編

# React + TypeScript + Material UI

TypeScript

Visual Studio Code でコード補完が出る！

レスポンシブデザイン

GitHubで管理

# どうやったか - フロントエンド編

## GitHubで管理

GitHub Actionsを用いてコミット後にS3へアップロードするように設定した

The screenshot shows the GitHub Actions page for the repository 'Tsutomu-Ikeda / tomstutom.com'. The 'Actions' tab is selected. The 'All workflows' tab is active. Two workflow runs are listed:

- CI #13: Commit 2053bb6 pushed by Tsutomu-Ikeda** (17 days ago, 1m 46s)
- CI #12: Commit 597c348 pushed by Tsutomu-Ikeda** (17 days ago, 1m 54s)

Both runs show a green checkmark and the message 'キャッシュを5分間効かせるように設定'.

ビルドして、S3のコンソールを開き、アップロードするというタスクを自動化できた！

## 実際にやってみた結果 - フロントエンド編

# Google PageSpeed Insightsを用いて計測した



## 実際にやってみた結果 - フロントエンド編

73点は悪くないけど改善の余地あり！

### 具体的に指摘された内容

- 使わないJavaScriptのコードが読み込まれている
- ページを開いた瞬間の画像読み込み枚数が多い

# 実際にやってみた結果 - フロントエンド編

## JavaScriptの分割

これはめっちゃ簡単

```
16      - import Activities from "./views/Activities";
17      - import Top from "./views/Top";
18      - import Profile from "./views/Profile";
19      - import Links from "./views/Links";
20      - import NoteBook from "./views>NoteBook";
21      17    import NotFound from "./views/NotFound";
22      18
23      + const Activities = React.lazy(() => import("./views/Activities"));
24      + const Top = React.lazy(() => import("./views/Top"));
25      + const Photos = React.lazy(() => import("./views/Photos"));
26      + const Profile = React.lazy(() => import("./views/Profile"));
27      + const Links = React.lazy(() => import("./views/Links"));
28      + const Logs = React.lazy(() => import("./views/Logs"));
29      + const NoteBook = React.lazy(() => import("./views>NoteBook"));
```

import文を関数にする。本当に必要なとき(関数が呼ばれたときだけ)importされる。

実装したコミット: [8292305](#)

# 実際にやってみた結果 - フロントエンド編

## 画像の遅延ロードを実装

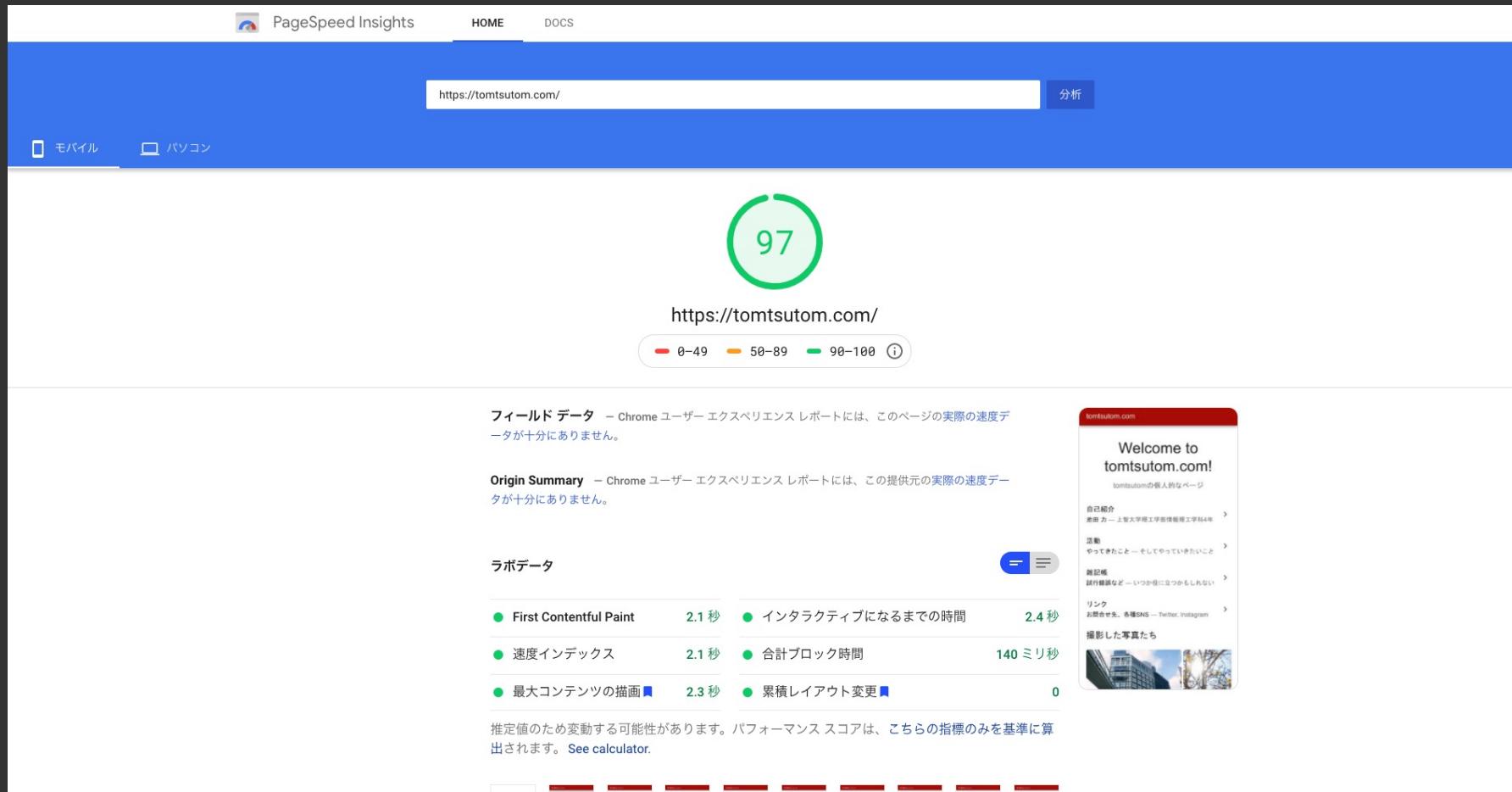
必要なときまで画像を読み込まないようにした



実装したコミット [bd16f9e](#)

# 実際にやってみた結果 - フロントエンド編

その結果ほぼ満点になった！



まとめ

AWSをフル活用すると格安でサイトが作れる

その額 169円/月

コードをチューニングすると爆速サイトが作れる

73点 → 97点

みなさんもぜひ自分のページを作ってみてください！

<https://tomtsutom.com>

<https://github.com/Tsutomu-Ikeda/tomtsutom.com>