How to deploy GitHub pages with AWS Route 53 registered custom domain and force HTTPS

Ben Wiz
Dec 21, 2018 · 3 min read ★

In this guide I will explain how to deploy a website to GitHub pages forcing HTTPS over a custom domain that is registered with AWS Route 53. We will set up our domain so that the www subdomain will redirect to the apex domain.

Summary

- Set up the GitHub repo
- Commit and push an index.html or use Jekyll
- Configure AWS Route 53

Step 1: Create GitHub repo and turn on GitHub Pages

1. If it does not exist yet, create a repository using the naming pattern `your-github-username.github.io`. Since my username is *benwiz* my repository is called `benwiz.github.io` .
2. Click the *Settings* tab and scroll down the *GitHub Pages* section
3. From the *Source* dropdown select *master branch*
4. Click *Save*

Step 2: Push source code to GitHub

1. Clone the repo to your local machine

```
git clone git@github.com:your-github-username/your-github-
username.github.io.git && cd your-github-username.github.io
```

```
echo "Hello GitHub Pages!" > index.html
```

1. Looking forward, we will need to have a file named *CNAME* that contains a single row: your custom domain. My *CNAME* file has the following contents.

```
git add . && git commit -m 'Create content and CNAME record' && git
push
```

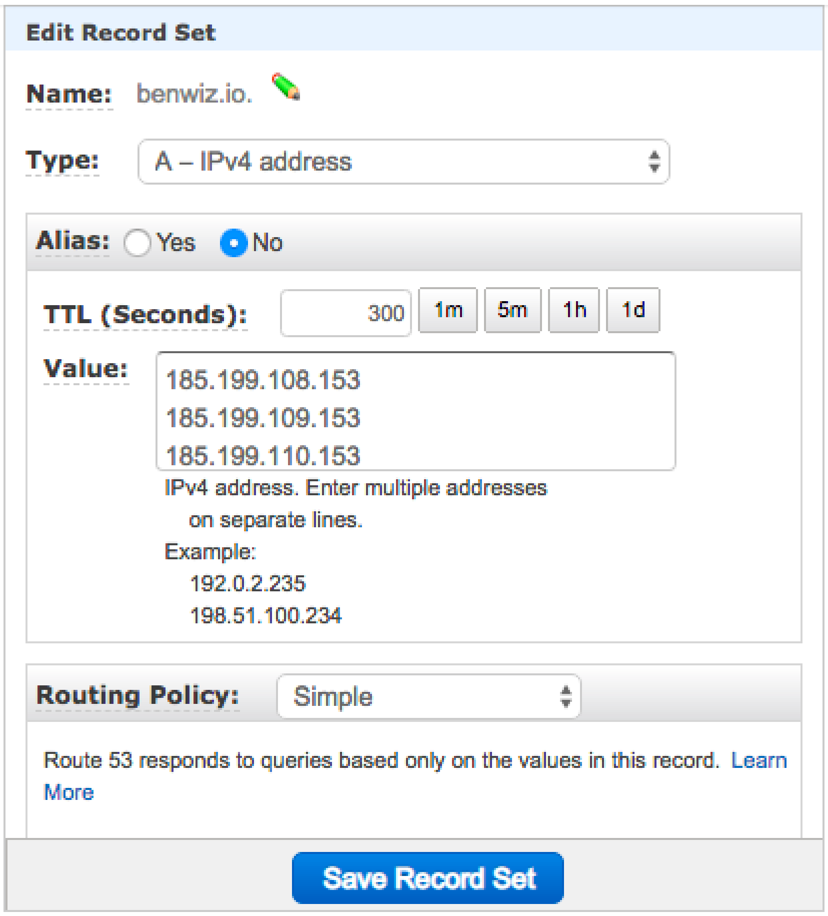
Step 3: Confirm that GitHub pages has been deployed

Visit http://your-github-username.github.io and https://your-github-username.github.io. You should see the contents of your *index.html* file at both the unsecured and secured addresses.

Step 4: Configure AWS Route 53 to use your custom vanity domain

1. Log into the AWS console and go to the Route 53 dashboard.
2. Click *Hosted zones*
3. Click the domain you would like to use
4. Click *Create Record Set*
5. Do not enter anything into the *Name* field
6. Under the *Type* dropdown, select *A — IPv4 addresses*
7. The *Alias* toggle should be set to *No*
8. Enter the following four IP addresses into the *value* text area. Then click *Save Record Set*.

```
185.199.108.153
185.199.109.153
185.199.110.153
185.199.111.153
```


Edit Record Set

Name: benwiz.io.

Type: A – IPv4 address

Alias: Yes [✓ No]

TTL (Seconds): 300

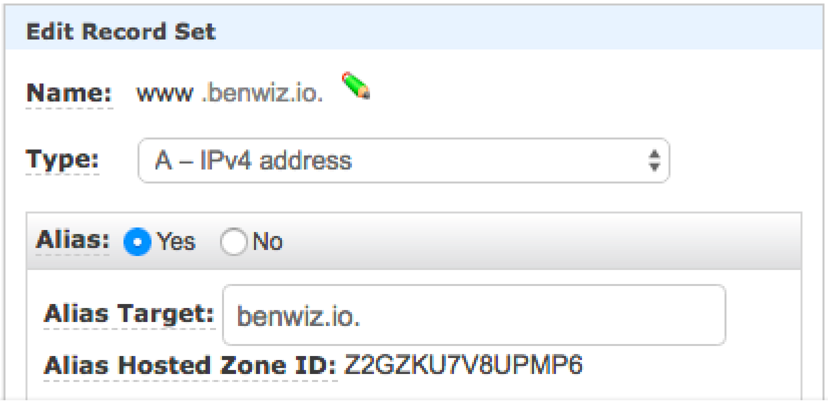
Value:
185.199.108.153
185.199.109.153
185.199.110.153

IPv4 address. Enter multiple addresses on separate lines.
Example:
192.0.2.235
198.51.100.234

Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. Learn More

1. Click *Create Record Set*, again
2. Into the *Name* field, enter `www`
3. Under the *Type* dropdown, select *A — IPv4 addresses*, again
4. The *Alias* toggle should be set to *Yes*, unlike before
5. In the *Alias Target* field, select the apex domain we previously set up. For me this is *benwiz.io*.
6. Click *Save Record Set*, again


Edit Record Set

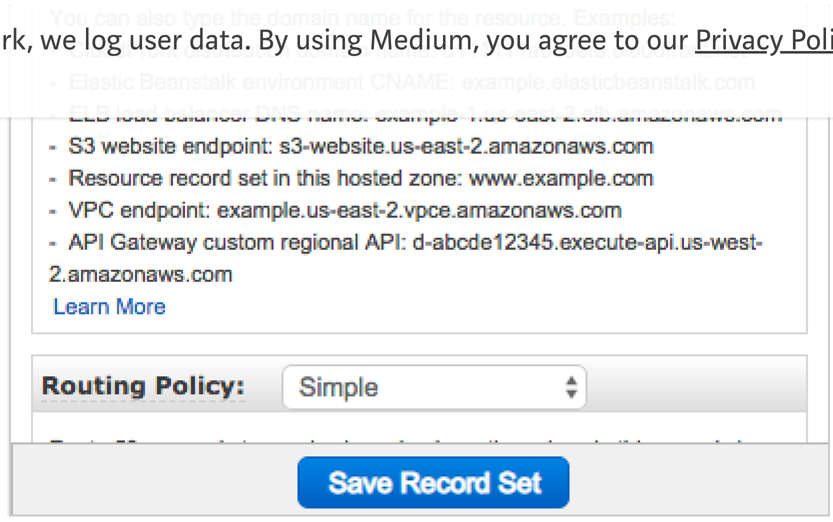
Name: www .benwiz.io.

Type: A – IPv4 address

Alias: [✓ Yes] No

Alias Target: benwiz.io.

Alias Hosted Zone ID: Z2GZKU7V8UPMP6


- S3 website endpoint: s3-website.us-east-2.amazonaws.com
- Resource record set in this hosted zone: www.example.com
- VPC endpoint: example.us-east-2.vpce.amazonaws.com
- API Gateway custom regional API: d-abcde12345.execute-api.us-west-2.amazonaws.com

Step 5: Configure GitHub to serve over your custom domain

1. Return to your GitHub repository's settings tab
2. Scroll down to the GitHub Pages section
3. In the *Custom domain* field enter your custom domain: `your-custom-domain.com`
4. Click *Save*
5. Check *Enforce HTTPS*

Step 6: Confirm that your page is accessible at your custom domain

Visit `https://your-custom-domain.com`. You should see the contents of your *index.html*.

1. Visit https://www.your-custom-domain.com. You should be redirected to https://your-custom-domain.com.
2. Visit http://your-custom-domain.com. You should be redirected to https://your-custom-domain.com.
3. Visit http://www.your-custom-domain.com. You should be redirected to https://your-custom-domain.com.

. . .

Originally published at benwiz.io on December 21, 2018.