Get started Open in app

Joshua Jarvis

4 Followers About Follow

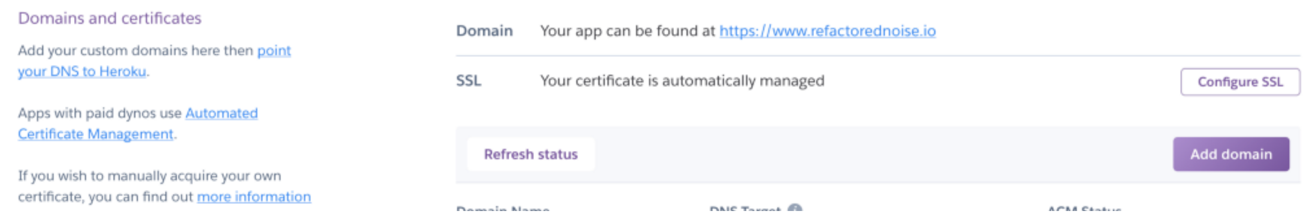
Heroku and AWS Route 53 SSL Configuration

Joshua Jarvis Apr 30, 2019 · 3 min read

Recently I migrated most of the production applications for Carnegie Mellon Robotics Academy over to Heroku. I decided that it would also be a good opportunity to migrate my personal blog over to Heroku as well. Overall the transition has been smooth on both ends. With that said, I really struggled with configuring SSL redirects for my naked domains. Heroku published an article that suggested creating an S3 Bucket to redirect to a default subdomain like `www.example.com` and assigning that bucket to `example.com`. However, S3 will not allow you to add an SSL certificate which is a nonstarter for most sites. Fortunately, there is an easy enough work around.

Getting an SSL Certificate for a Subdomain

This process is simple enough. On Heroku, go to your settings page and click configure SSL under 'Domains and Certificates'. Select 'Automatically configure using Automated Certificate Management'. Add a domain with a default subdomain that you would like to have the naked domain redirect to. In return, you will receive a CNAME that will then need to be added to Route53 for the corresponding domain. It will take a few minutes before you receive an ACM status of 'Ok'.



Domains and certificates

Add your custom domains here then point your DNS to Heroku.

Apps with paid dynos use Automated Certificate Management.

If you wish to manually acquire your own certificate, you can find out more information

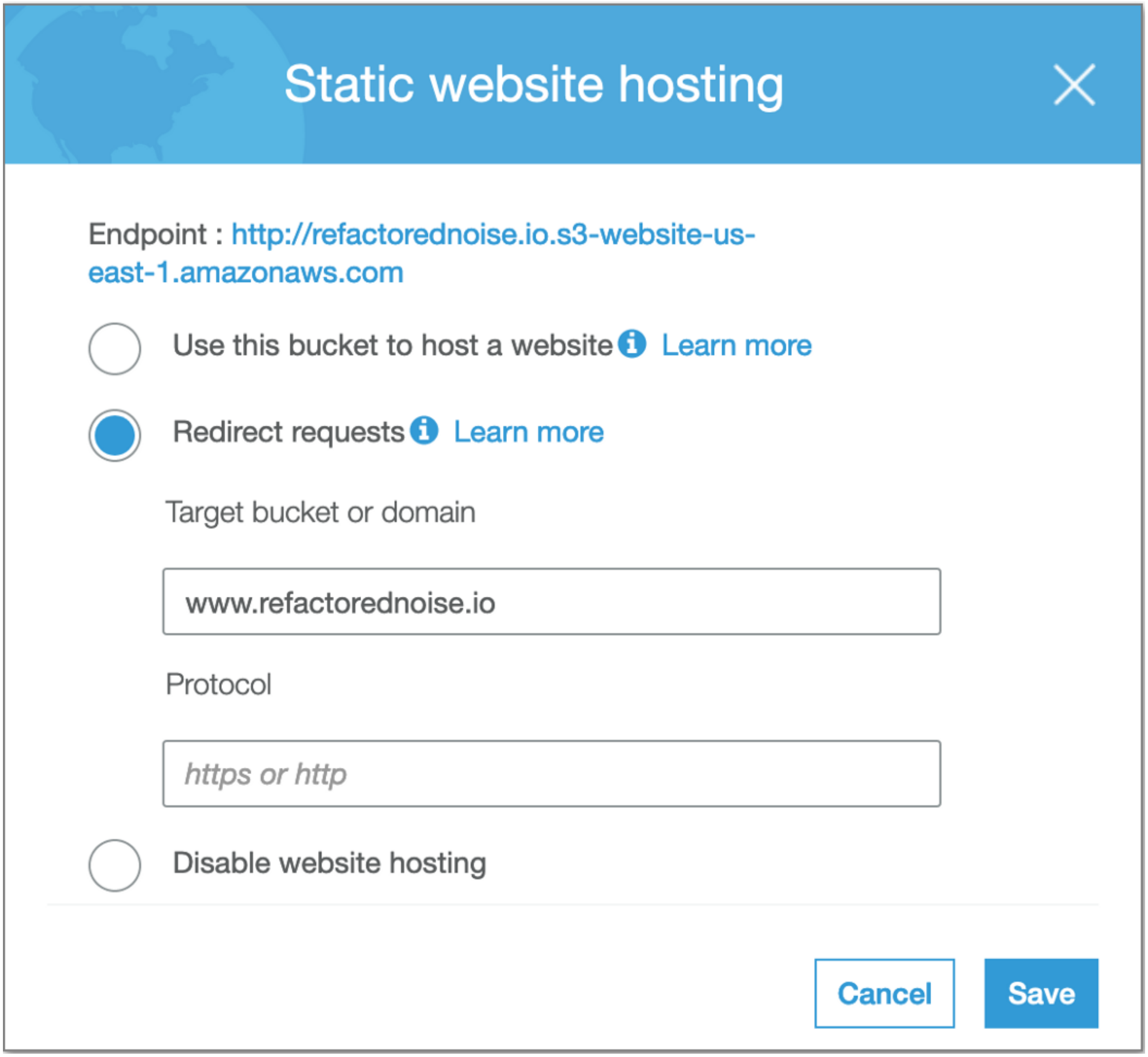
Domain Your app can be found at https://www.refactorednoise.io

SSL Your certificate is automatically managed

At this time you should be able to securely access your application at `www.example.com`.

Redirecting a Naked Domain to a Default Subdomain

This is where things get interesting. Create a bucket and name it `example.com`. Once the bucket is created, navigate to the bucket's properties tab and click 'Static website hosting' and select 'Redirect Requests' to `www.example.com`.



Static website hosting

Endpoint : http://refactorednoise.io.s3-website-us-east-1.amazonaws.com

○ Use this bucket to host a website Learn more

● Redirect requests Learn more

Target bucket or domain

www.refactorednoise.io

Protocol

https or http

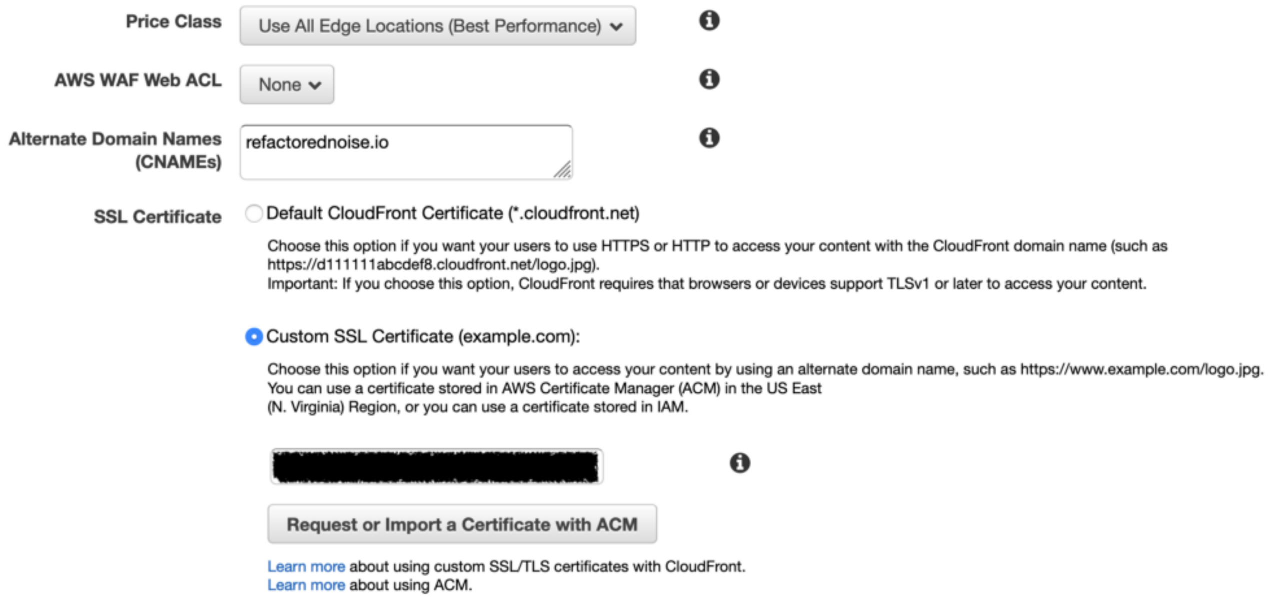
○ Disable website hosting

Distribution that points to the S3 Bucket endpoint.

AWS Cloudfront

Cloudfront is a fast and efficient content delivery network. Typically I use these networks to host my assets. In this case, Cloudfront satisfies our need to add an SSL certificate and the resulting domain can be added to a naked domain in Route 53.

Copy the S3 endpoint from the previous section and create a cloudfront distribution. The ‘Origin Domain’ will be the S3 endpoint. Update the ‘Alternate Domain Names (CNAMEs)’ field to the naked domain name. Select ‘Custom SSL Certificate (example.com)’ and choose the SSL cert for your naked domain. If you don’t have one, click the option to create one. Be sure to request a certificate for both `example.com` and `*.example.com` .



Price Class: Use All Edge Locations (Best Performance)

AWS WAF Web ACL: None

Alternate Domain Names (CNAMEs): refactorednoise.io

SSL Certificate:

○ Default CloudFront Certificate (*.cloudfront.net)

Choose this option if you want your users to use HTTPS or HTTP to access your content with the CloudFront domain name (such as https://d111111abcdef8.cloudfront.net/logo.jpg).
Important: If you choose this option, CloudFront requires that browsers or devices support TLSv1 or later to access your content.

● Custom SSL Certificate (example.com):

Choose this option if you want your users to access your content by using an alternate domain name, such as https://www.example.com/logo.jpg.
You can use a certificate stored in AWS Certificate Manager (ACM) in the US East
(N. Virginia) Region, or you can use a certificate stored in IAM.

[redacted]

Learn more about using custom SSL/TLS certificates with CloudFront.
Learn more about using ACM.

All of the other options can be left at their default settings. It usually takes about 15 minutes for AWS to build a CDN once the request has been submitted.

Conclusion

At this point you should have a working CDN domain that securely redirects to the default subdomain of your application. Simply add this domain to `example.com` as an

Get started Open in app

Originally published at https://refactorednoise.io on May 1, 2019.

AWS Heroku DNS Ssl Route 53

About Write Help Legal

Get the Medium app

Download on the App Store GET IT ON Google Play