

Pit traps in the docs

An anecdote, two fables, and a coda

Sean Moran-Richards | @flying_grizzly

Hi. I'm Sean

Been working on a URL shortener that publishes 'static' redirects

- fast
- resilient
- learn some AWS shit

Uses S3 and Cloudfront

Generally, AWS' docs are good

They cover a lot, and even specify return values and structs. It makes developing with the SDK really easy and predictable.

An anecdote



Magic: The Gathering has a comprehensive and complete set of rules.

They list *everything* that can and cannot be done in the game. If it's not there, it's not legal play.

Funnily, this means they must stipulate that you can meet other players at your friendly neighborhood gaming store to play. If they didn't it would be illegal.

Fable the first

I want a URL shortener

I want it to stick around even when things change

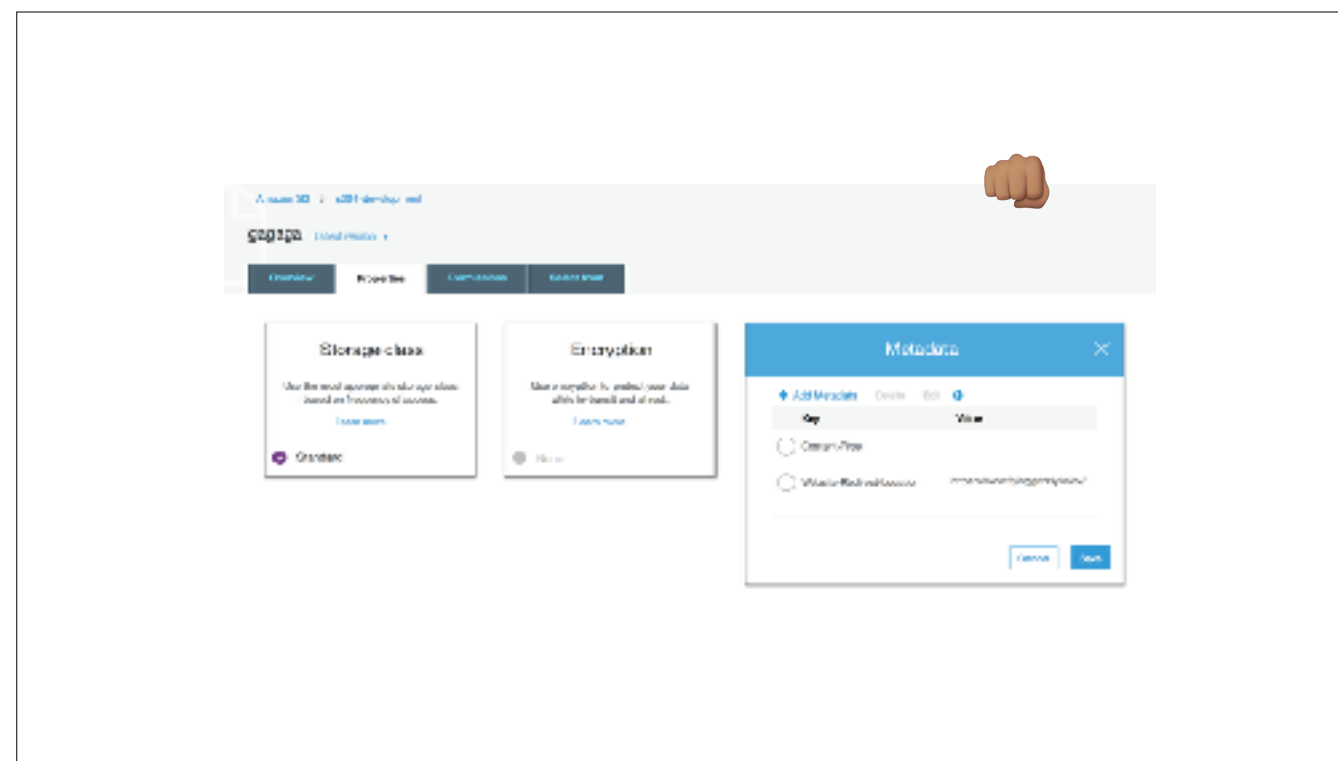
Solution: stick shit on Amazon



grz.li/s301

Look. This is it!

You can use it if you want.



It's pretty easy.

S3 objects accept a `:website_redirect_location` metadatum

That's what it looks like. And it works.

If you hit the S3 endpoint for that, it redirects you.

Fist bump.



Also, I can hook it up to Cloudfront! It makes things faster.

And I get redirected when I hit that endpoint too.

Bullseye.



But wait.

When you change the short URL, it doesn't change.

But it's definitely changed when I check the S3 console.



Maybe I need to invalidate the Cloudfront stuff, so it expires and checks for updates sooner. That will fix it!



No. It does not fix it.

Grrr.



HTTP docs...

Maybe... there's some behavior where the browser and the cache react differently to the redirect?

Turns out there's no cache limit on 301s, and FF and Chrome cache them indefinitely unless you specify cache control.

I can fix this!

```
S3.put_object(  
    bucket: @bucket,  
    key: 'foo',  
    website_redirect_location: 'http://www.newsite.com',  
    cache_control: 'max-age=0, s-maxage=365000'  
)
```

Add cache control!



No. It does not fix it.

Grrr.

```
S3.put_object(  
    bucket: @bucket,  
    key: 'foo',  
    website_redirect_location: 'http://www.newsite.com',  
    cache_control: 'max-age=0, s-maxage=365000',  
    expires: 'a long time ago'  
)
```

Maybe expiration??? The AWS docs say that you can use this to get good behavior

Metadata

+ Add Metadata

Delete

Edit

Key	Value
<input type="radio"/> Cache-Control	max-age=0;max-age=35999
<input type="radio"/> Expires	0
<input type="radio"/> Website-Frontend-Location	https://www.flyingdizzy.com/

Cancel

Save



It's all there. I'm good. Should work now.



GAAAAHHHHH



GAAAAHHHHH



No really... GAAHHHHHHH



No really... GAAHHHHHHH

```
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
```

Now I start CURLing and doing a bunch of things to try.

And I give AWS a few more pennies with each request.

Though it's nowhere in the docs, it looks like setting `:website_redirect_location` means that other metadata headers won't be forwarded to clients or CF.

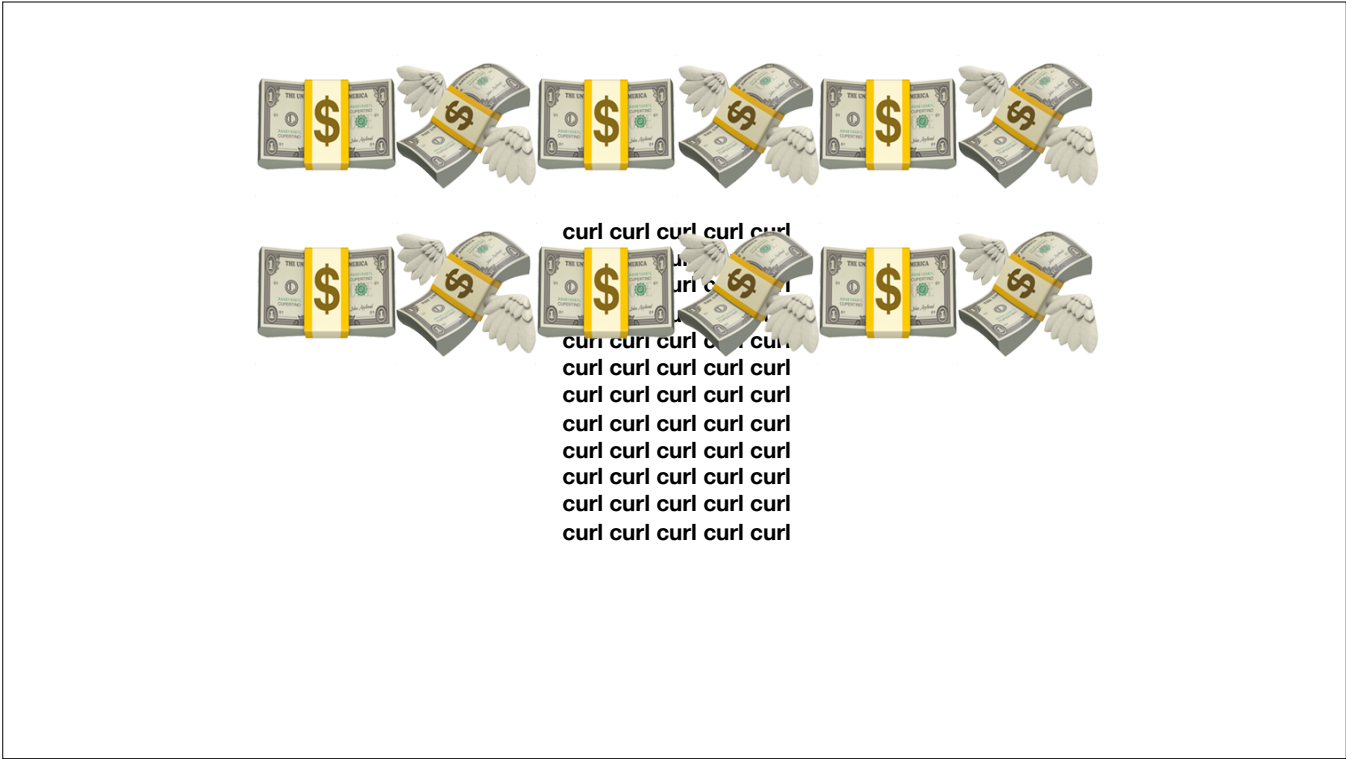


curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl
curl curl curl curl curl

Now I start CURLing and doing a bunch of things to try.

And I give AWS a few more pennies with each request.

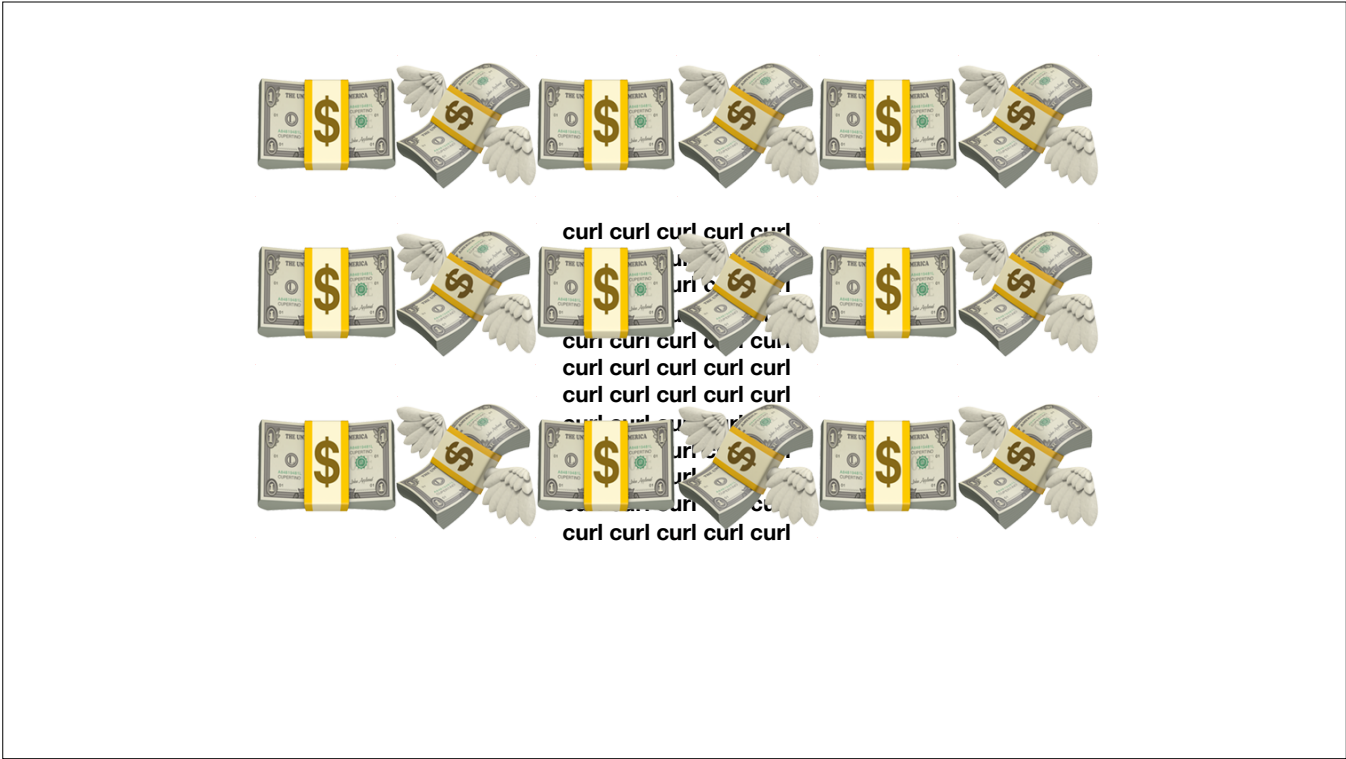
Though it's nowhere in the docs, it looks like setting `:website_redirect_location` means that other metadata headers won't be forwarded to clients or CF.



Now I start CURLing and doing a bunch of things to try.

And I give AWS a few more pennies with each request.

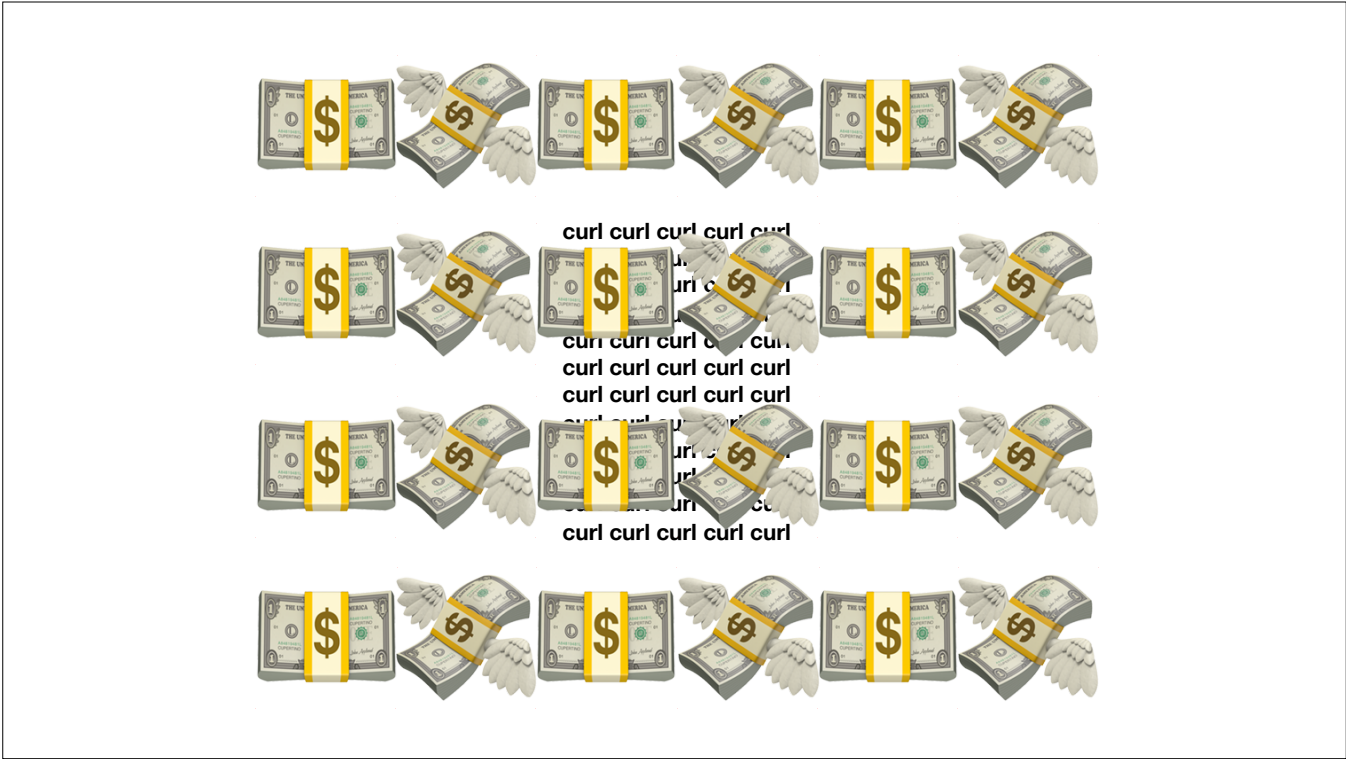
Though it's nowhere in the docs, it looks like setting :website_redirect_location means that other metadata headers won't be forwarded to clients or CF.



Now I start CURLing and doing a bunch of things to try.

And I give AWS a few more pennies with each request.

Though it's nowhere in the docs, it looks like setting :website_redirect_location means that other metadata headers won't be forwarded to clients or CF.



Now I start CURLing and doing a bunch of things to try.

And I give AWS a few more pennies with each request.

Though it's nowhere in the docs, it looks like setting :website_redirect_location means that other metadata headers won't be forwarded to clients or CF.



And now time passes. I will find a new way.

Fable the second

I found a new way.

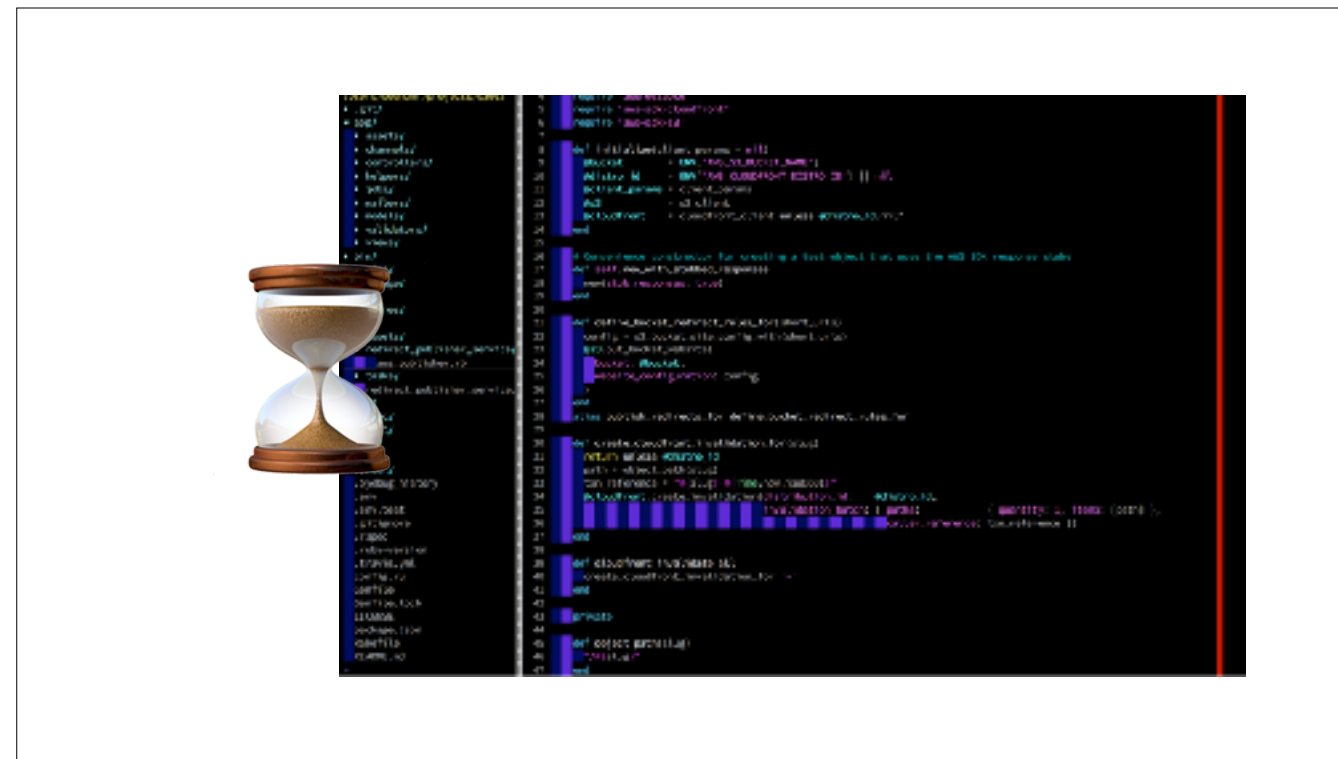
```
<Condition>
  <KeyPrefixEquals>slug-10</KeyPrefixEquals>
</Condition>
<Redirect>
  <Protocol>https</Protocol>
  <HostName>www.flyinggrizzly.io</HostName>
  <ReplaceKeyWith/>
  <HttpRedirectCode>307</HttpRedirectCode>
</Redirect>
</RoutingRule>
<RoutingRule>
  <Condition>
    <KeyPrefixEquals>slug-11</KeyPrefixEquals>
  </Condition>
  <Redirect>
    <Protocol>https</Protocol>
    <HostName>www.flyinggrizzly.io</HostName>
    <ReplaceKeyWith/>
    <HttpRedirectCode>307</HttpRedirectCode>
  </Redirect>
</RoutingRule>
<RoutingRule>
  <Condition>
    <KeyPrefixEquals>slug-12</KeyPrefixEquals>
```

XML!!!

Not quite.

S3 websites allow you to specify routing rules on the bucket, which respond with *any* HTTP response.

Including 307. No caching!!!



So I rewrite the AWS service to push things to the bucket.

And I did check the docs first. This definitely looks like the way to do it.

I don't love that I now need to publish/republish all short URLs at the same time, and it makes me worry about scalability

But I wrote a discovery story to check that later. For now, MVP.



Everything is working!

URLs update within 30 seconds, which is acceptable timing for Cloudfront invalidations.

Everything is lovely!!!



Everything is working!

URLs update within 30 seconds, which is acceptable timing for Cloudfront invalidations.

Everything is lovely!!!



Time to test the scalability a littler.

Let's seed out 100 short URLs to start.

That fails. Uh oh...

But 20 works.

But 50 fails.

Context:

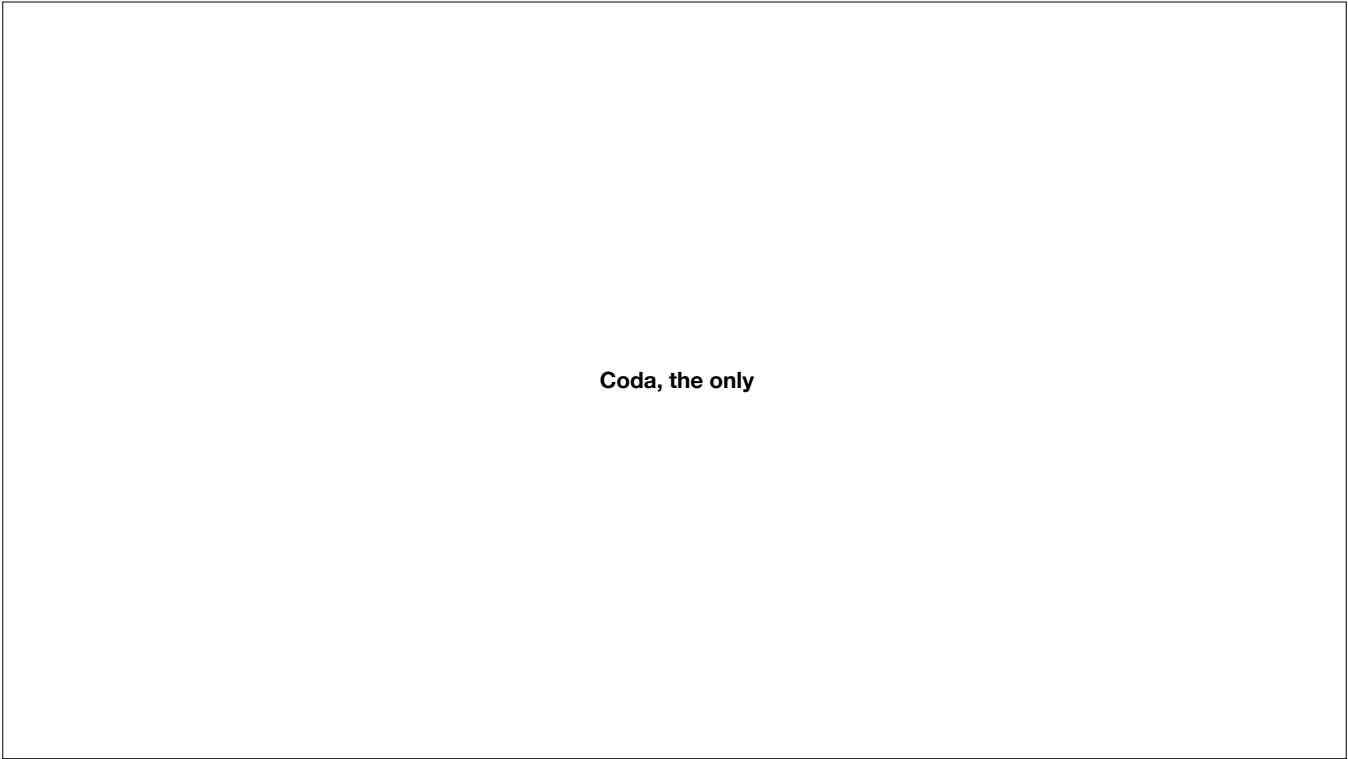
Already had a few short URLs. and the logs point to it failing when the 51st is being published (callback is run before each save on the database)

Not in the docs... again

Turns out there's a 50 rule limit on s3 websites.

Which I only found out on SO. (At least it was on the internet somewhere this time)

But it's not in the AWS docs...



Coda, the only

So it's probably going back to objects if I want this on S3. Which I'm questioning.

I might do it on Nginx on a cheap VPS and just front it with CF or something.

Or use Lambda@Edge

Both of these feel a little dirty, since they both involve more moving parts than I anticipated.

Oh, and a nice extra for Lambda docs:

- they say only Node...
- why the fuck is there Python in there?!?!?! (not a problem, but indicative of the same thing)



So it's probably going back to objects if I want this on S3. Which I'm questioning.

I might do it on Nginx on a cheap VPS and just front it with CF or something.

Or use Lambda@Edge

Both of these feel a little dirty, since they both involve more moving parts than I anticipated.

Oh, and a nice extra for Lambda docs:

- they say only Node...
- why the fuck is there Python in there?!?!?! (not a problem, but indicative of the same thing)



the Moral

Docs are hard. We get it.

We fuck up docs ourselves a lot.

But there's something very weird here.

Because the AWS docs are so good, and so exhaustive, they give the impression of being complete and comprehensive (like the MTG rules...)

So when things like this are missing, it can make people feel a little bit crazy.

This isn't limited to just docs, or just AWS. You make an implicit guarantee to your users with the things you deliver.

It's important to follow through on things like that.

There's a difference between M:TG and docs. In M:TG, you can either houserule an inconsistency, or there's a judge to adjudicate.

You can't really houserule the AWS API.

I'm not leaving AWS, but I'm definitely trusting the docs less. It's a difference. It means that if something else comes along that looks comparable, I'll be more inclined to try it and move on.