



Serverless Python

And a tangent about the secret origins of the Power Rangers

By Shy Ruparel



Shy Ruparel

Developer Evangelist | Contentful

@ShyRuparel

He/Him

serverless vs Serverless

Serverless Architectures



A traditional internet delivered app has a client communicating with a long-lived server process that handles most aspects of the application's logic

serverless architectures

break up this server

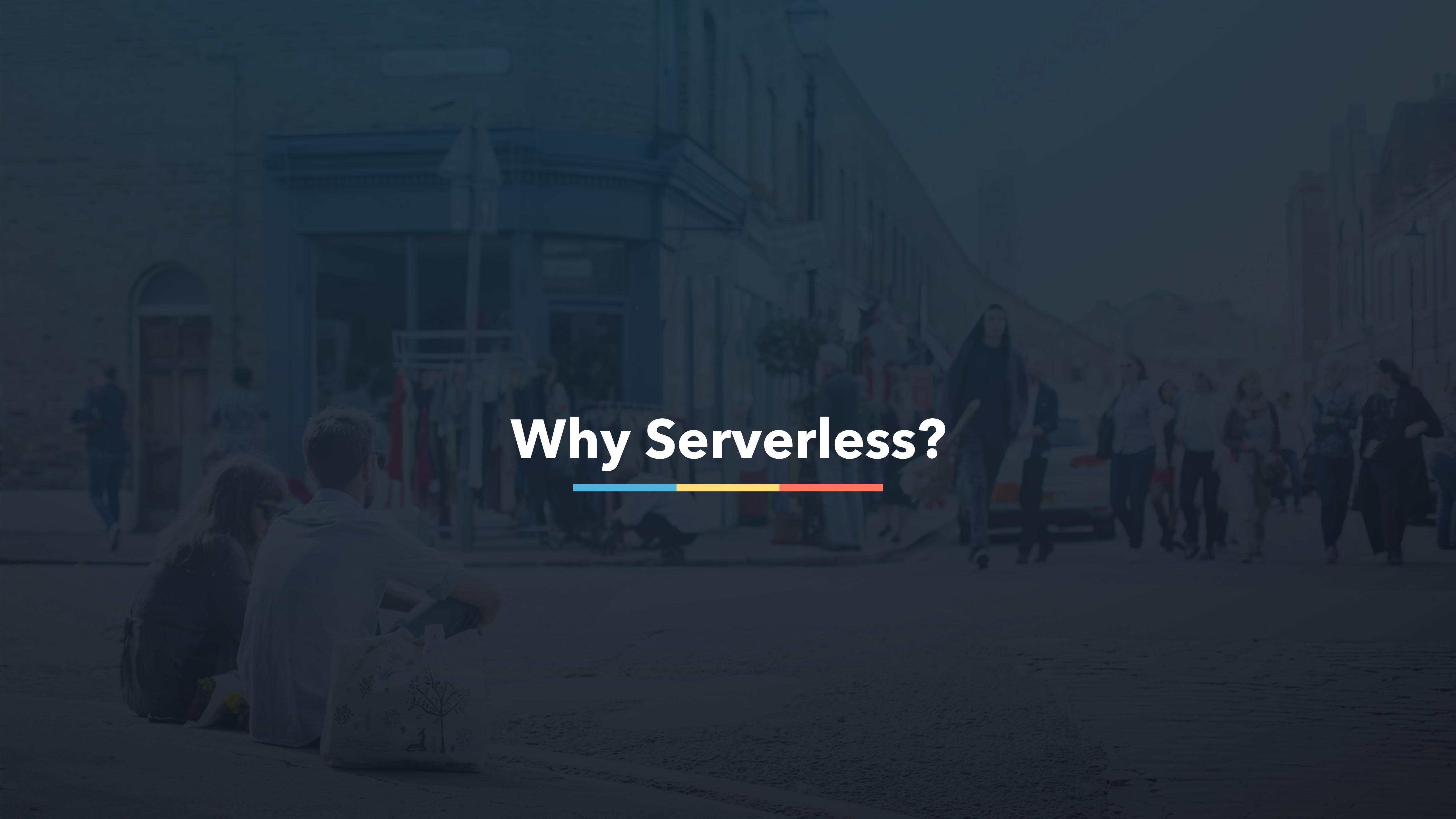
application flow control moves to the client

infrastructural logic, such as authentication, is handled by third-parties

third-parties will also handle business logic as possible

read-only data can be read directly from an internet hosted database

custom logic hosted in a FaaS can process updates and encapsulate databases



Why Serverless?

Pay for
What you Use

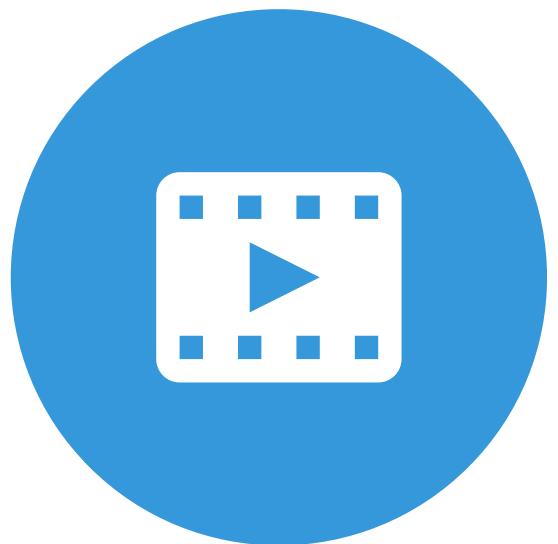


1 Request = 1 Server



SERVERLESS

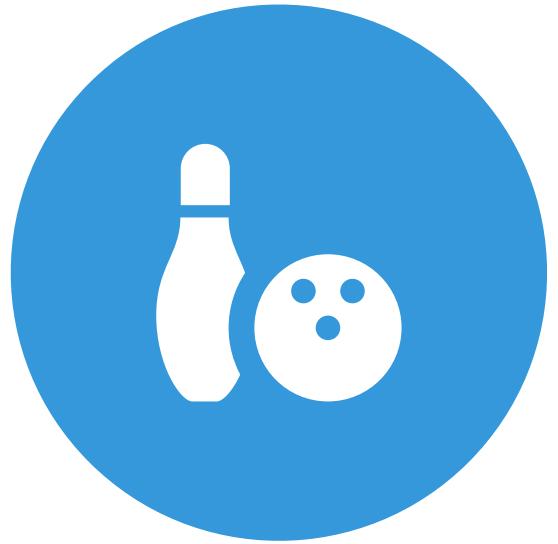
Going serverless allows you to create and run applications and services without having to worry about servers - at least not servers that you have to maintain.



Zero Downtime



Zero Maintenance



Pay for what you use



Infinite Scaling



Please don't fire your
ops team

The text is overlaid on a dark, slightly blurred background image of a person working at a desk. The person is facing away from the camera, looking at a computer screen. On the desk, there are two monitors: one showing a globe and another showing a grid-like interface. A keyboard and a mouse are also visible. The overall atmosphere is professional and focused.

A dark, semi-transparent background image of a person with glasses and a patterned shirt, sitting at a desk and working on a laptop. The person is looking towards the camera.

Let's look at the state of serverless python



Serverless



Chalice



Zappa



WSGI



Microservice



WSGI

Serverless
Zappa



Microservice

Serverless
AWS Chalice
Zappa

Vendor Lock-in



zappa



Let's make a Microsite



**But first it's time for a
small tangent**

The Origin of Power Rangers





KYŌRYŪ SENTAI ZYURANGER

1992-1993



MIGHTY MORPHIN POWER RANGERS

1993-1996



AK RANGER



AK RANGER



AK RANGER



AK RANGER

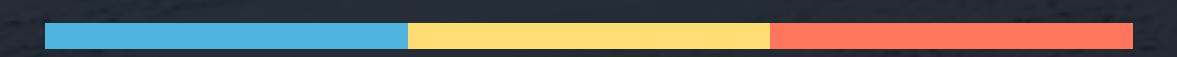


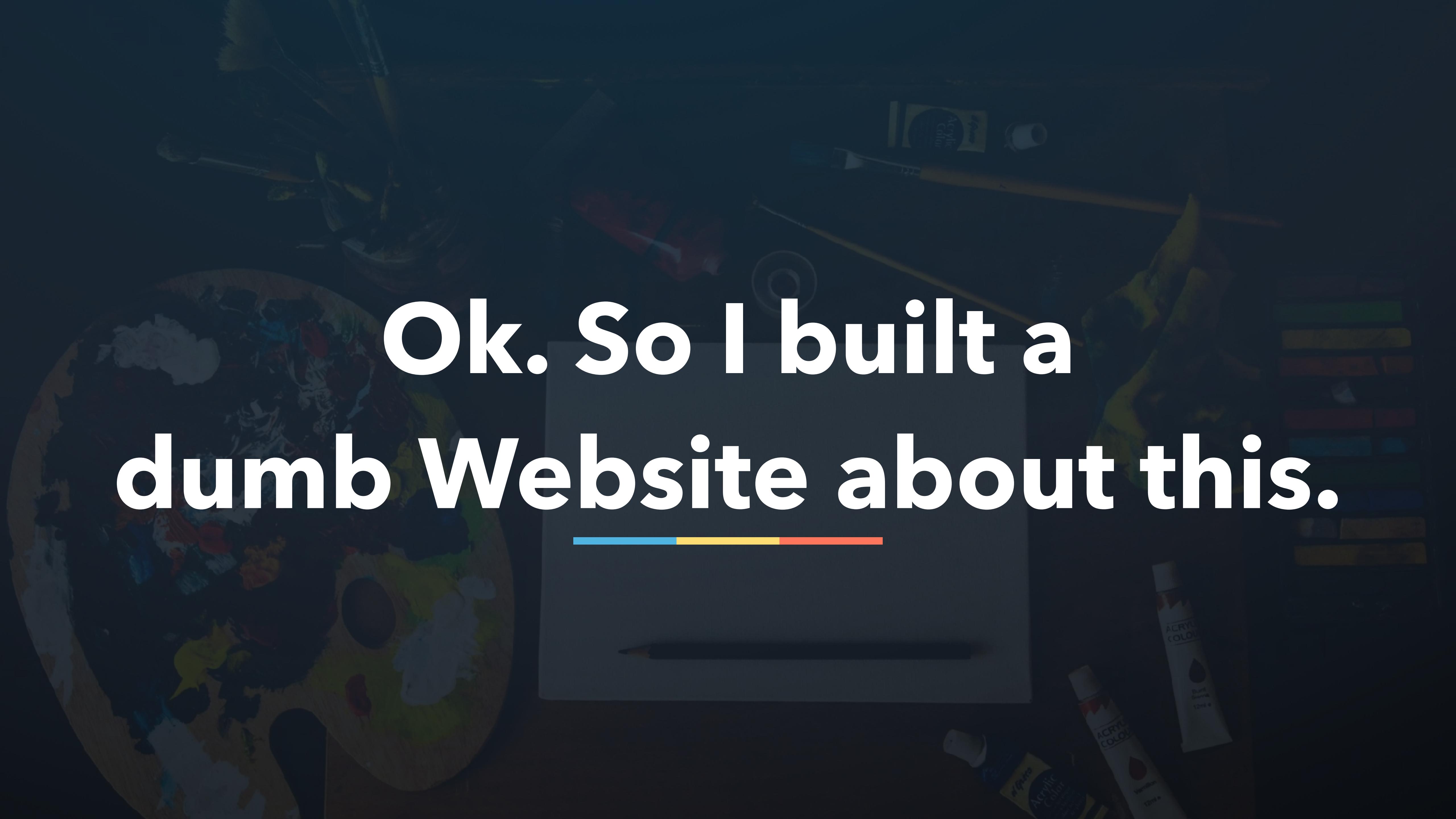
AK RANGER



HOW
YOUR
HIT
IS

WITH GUY RAZ





Ok. So I built a
dumb Website about this.

HENSHIN.RUPAREL.CO

www.contentful.com

Henshin!

"Henshin!" (変身) is the Japanese word for "transformation/transform". It refers to a subset of Japanese Super Hero that transforms between a super-powered form and a normal civilian mode. A [Henshin Hero](#) has distinct normal and powered forms, and needs to actively switch between the two. In essence, the character's powers are all turned off while they are in their Secret Identity.

Kyōryū Sentai Zyuranger
Original release: February 21, 1992 - February 12, 1993



AK RANGER

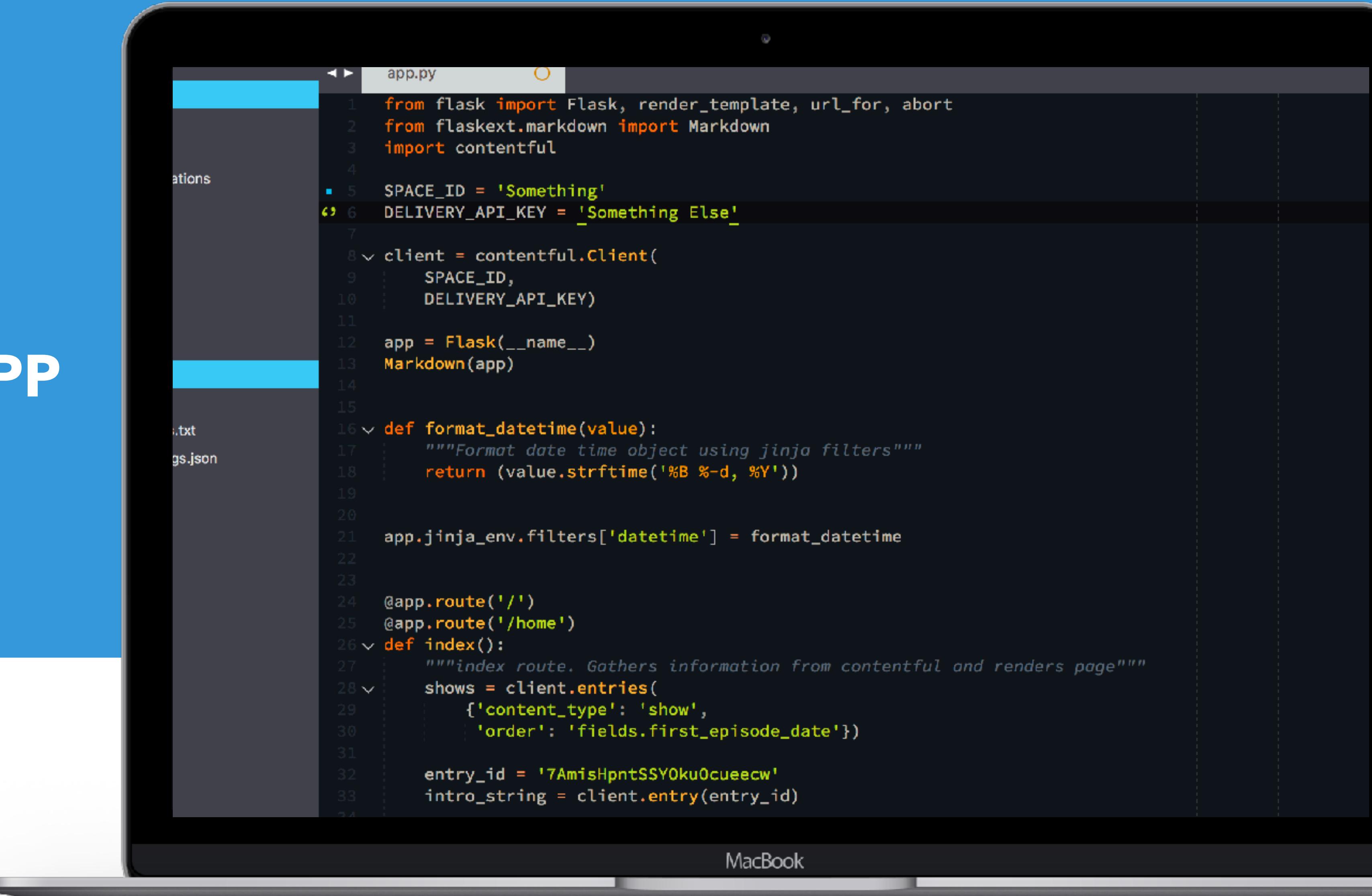
Gosei Sentai Dairanger
Original release: February 19, 1993 - February 11, 1994



80 LINE FLASK APP

+

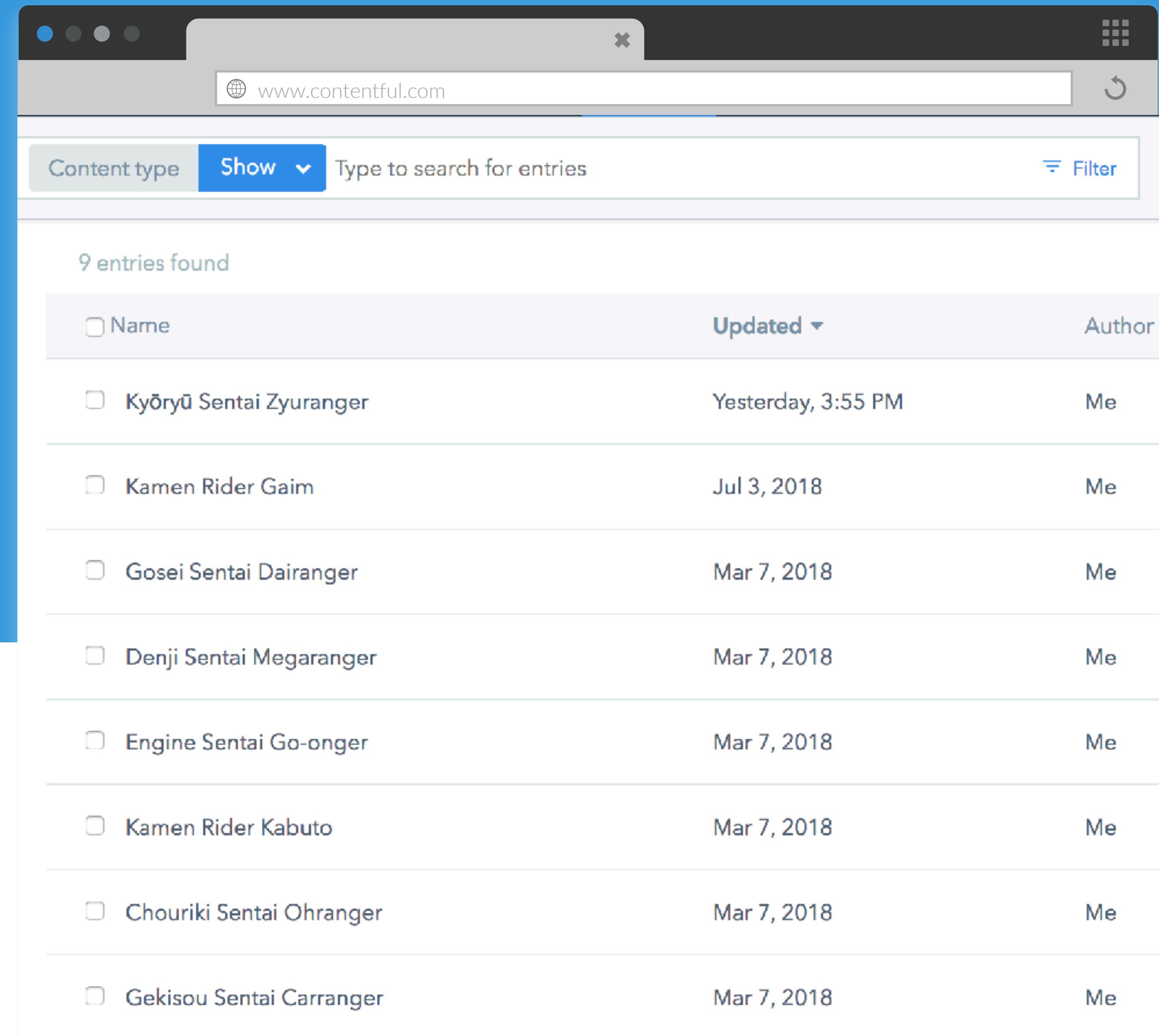
CONTENTFUL



```
app.py
1 from flask import Flask, render_template, url_for, abort
2 from flaskext.markdown import Markdown
3 import contentful
4
5 SPACE_ID = 'Something'
6 DELIVERY_API_KEY = 'Something Else'
7
8 client = contentful.Client(
9     SPACE_ID,
10    DELIVERY_API_KEY)
11
12 app = Flask(__name__)
13 Markdown(app)
14
15
16 def format_datetime(value):
17     """Format date time object using jinja filters"""
18     return (value.strftime('%B %d, %Y'))
19
20
21 app.jinja_env.filters['datetime'] = format_datetime
22
23
24 @app.route('/')
25 @app.route('/home')
26 def index():
27     """index route. Gathers information from contentful and renders page"""
28     shows = client.entries(
29         {'content_type': 'show',
30          'order': 'fields.first_episode_date'})
31
32 entry_id = '7AmisHpntSSY0ku0cueecw'
33 intro_string = client.entry(entry_id)
```

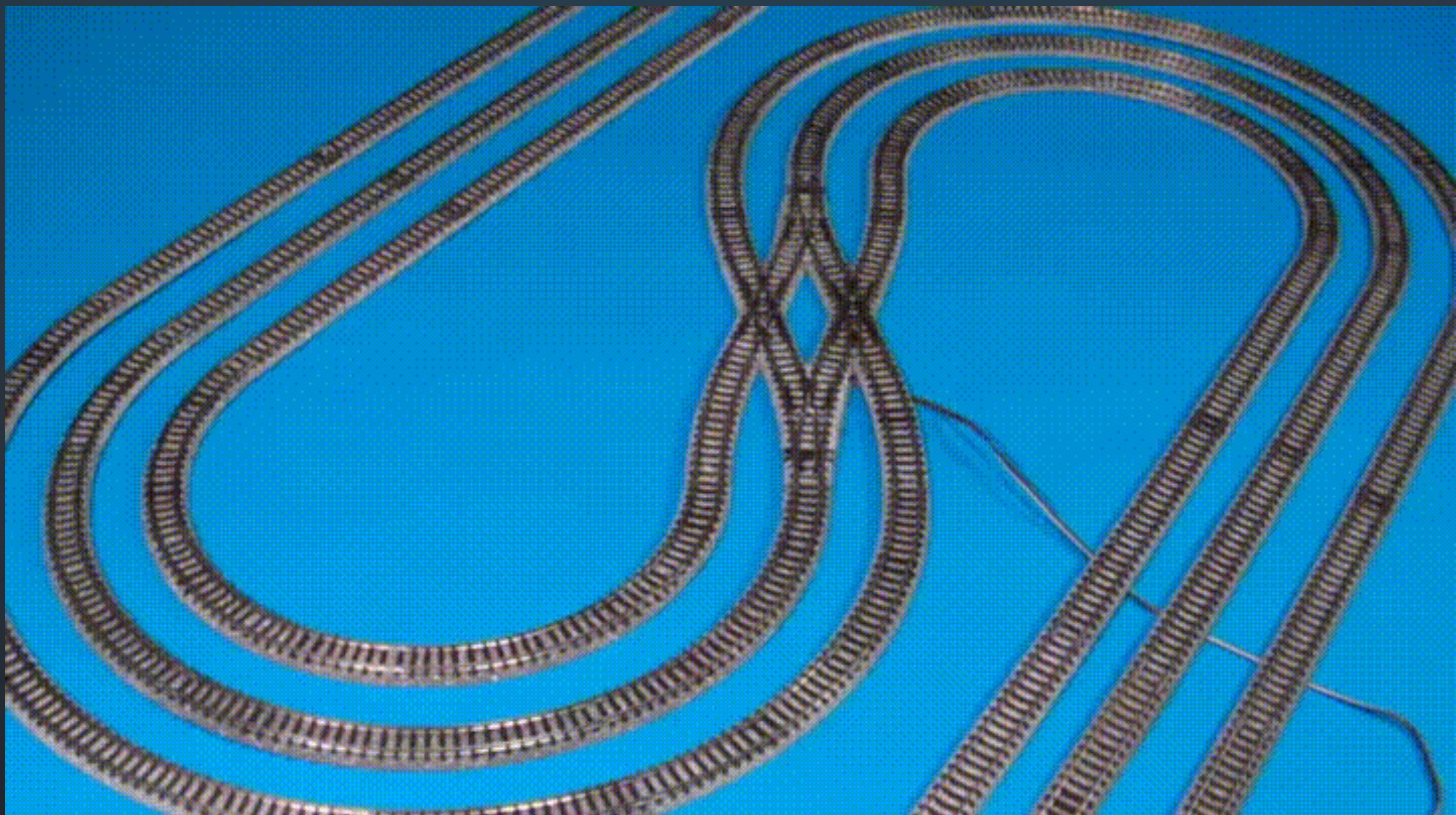
MacBook

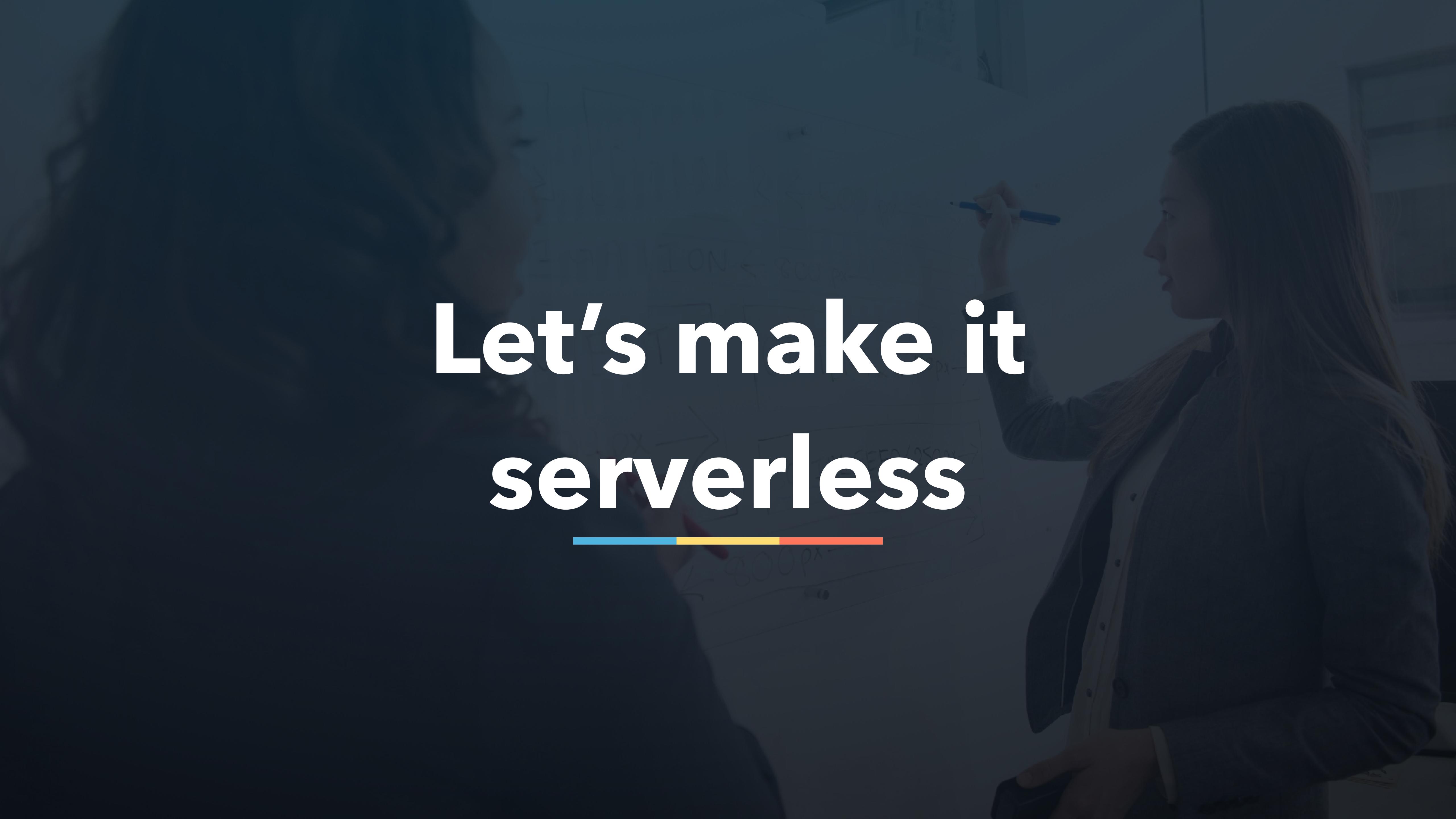
CONTENTFUL



A screenshot of the Contentful web interface. The browser address bar shows "www.contentful.com". The main area displays a table of content entries. The columns are "Name" (with a checkbox header), "Updated" (sorted by date), and "Author" (all entries are marked as "Me"). There are 9 entries found, listed from most recent to oldest.

Name	Updated	Author
Kyōryū Sentai Zyuranger	Yesterday, 3:55 PM	Me
Kamen Rider Gaim	Jul 3, 2018	Me
Gosei Sentai Dairanger	Mar 7, 2018	Me
Denji Sentai Megaranger	Mar 7, 2018	Me
Engine Sentai Go-onger	Mar 7, 2018	Me
Kamen Rider Kabuto	Mar 7, 2018	Me
Chouriki Sentai Ohranger	Mar 7, 2018	Me
Gekisou Sentai Carranger	Mar 7, 2018	Me



A dark, semi-transparent background image of a person with long hair, wearing a light-colored shirt, standing and writing on a whiteboard with a blue marker. The whiteboard has some faint, handwritten text and arrows.

Let's make it serverless

zappa

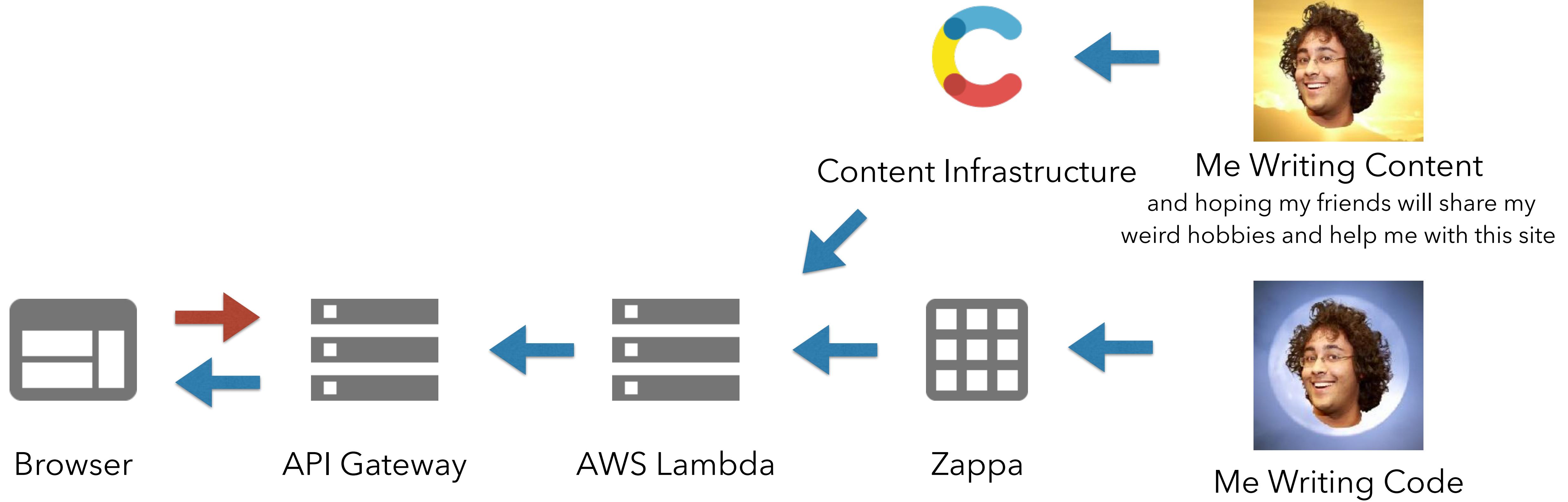


AWS Lambda

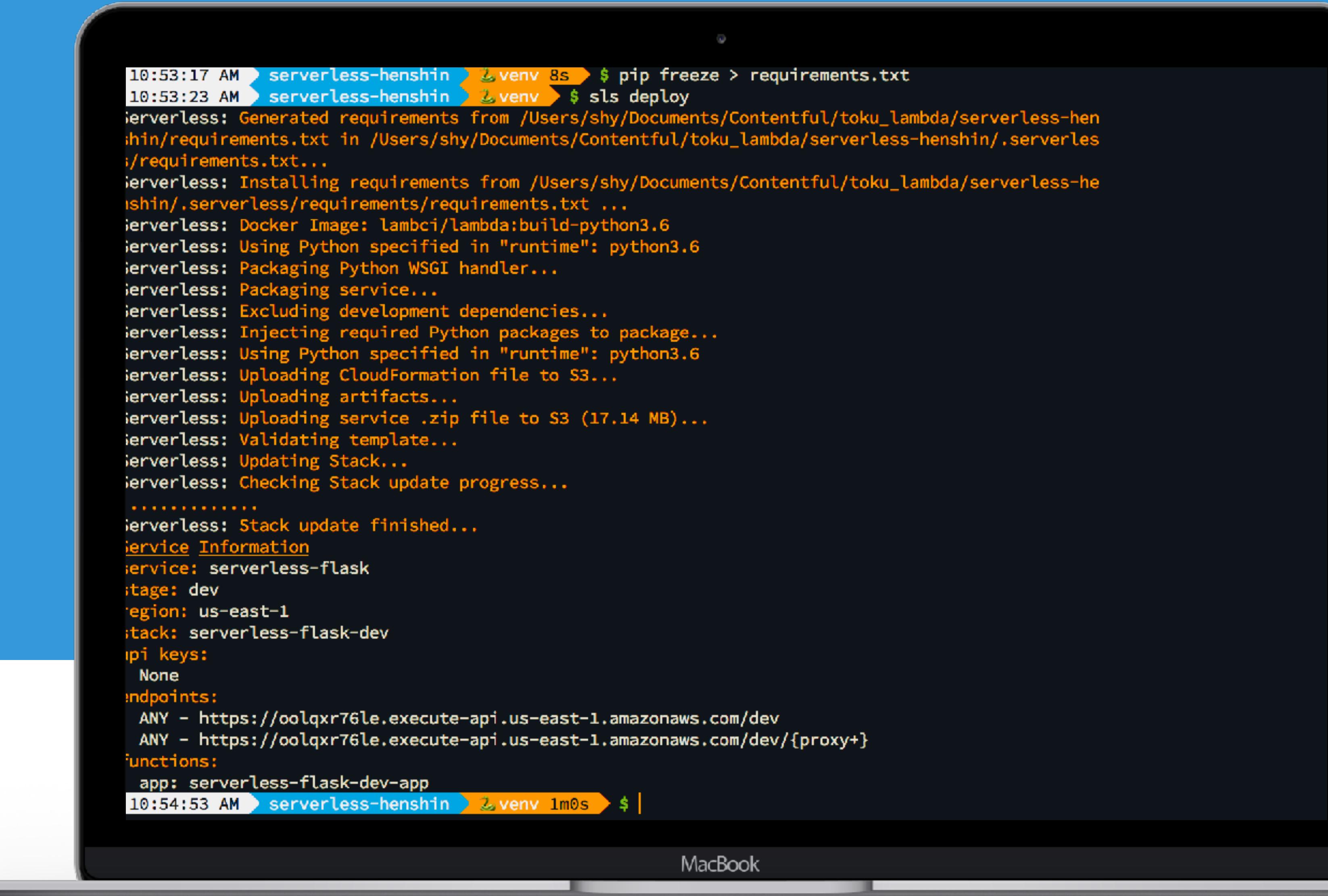


API Gateway





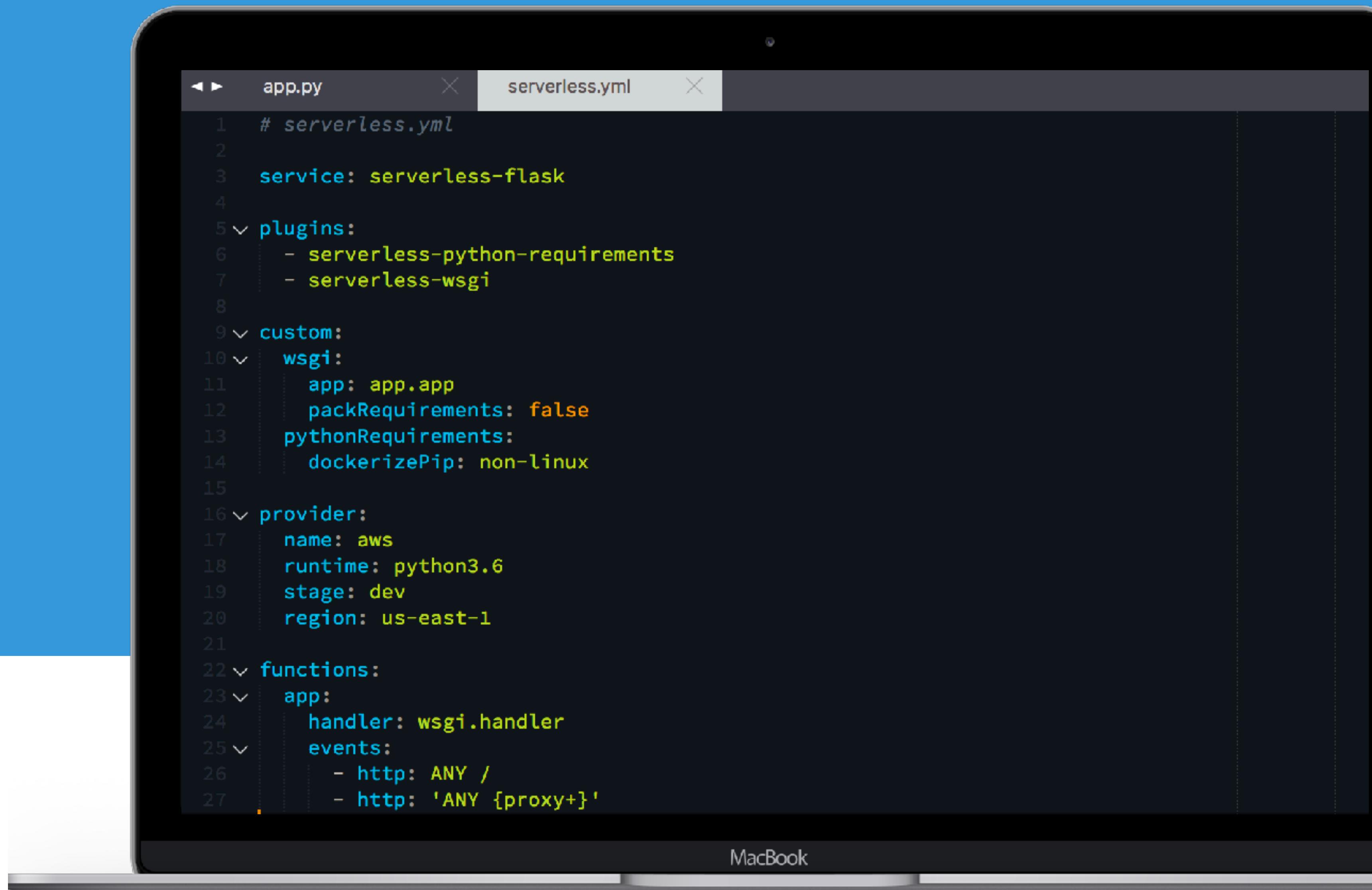
SERVERLESS FRAMEWORK



```
10:53:17 AM ➜ serverless-henshin ➜ venv 8s ➤ $ pip freeze > requirements.txt
10:53:23 AM ➜ serverless-henshin ➜ venv ➤ $ sls deploy
serverless: Generated requirements from /Users/shy/Documents/Contentful/toku_lambda/serverless-henshin/requirements.txt in /Users/shy/Documents/Contentful/toku_lambda/serverless-henshin/.serverless/requirements.txt...
serverless: Installing requirements from /Users/shy/Documents/Contentful/toku_lambda/serverless-henshin/.serverless/requirements/requirements.txt ...
serverless: Docker Image: lambci/lambda:build-python3.6
serverless: Using Python specified in "runtime": python3.6
serverless: Packaging Python WSGI handler...
serverless: Packaging service...
serverless: Excluding development dependencies...
serverless: Injecting required Python packages to package...
serverless: Using Python specified in "runtime": python3.6
serverless: Uploading CloudFormation file to S3...
serverless: Uploading artifacts...
serverless: Uploading service .zip file to S3 (17.14 MB)...
serverless: Validating template...
serverless: Updating Stack...
serverless: Checking Stack update progress...
.....
serverless: Stack update finished...
service Information
service: serverless-flask
stage: dev
region: us-east-1
stack: serverless-flask-dev
api keys:
  None
endpoints:
  ANY - https://oolqxr76le.execute-api.us-east-1.amazonaws.com/dev
  ANY - https://oolqxr76le.execute-api.us-east-1.amazonaws.com/dev/{proxy+}
functions:
  app: serverless-flask-dev-app
10:54:53 AM ➜ serverless-henshin ➜ venv 1m0s ➤ $ |
```

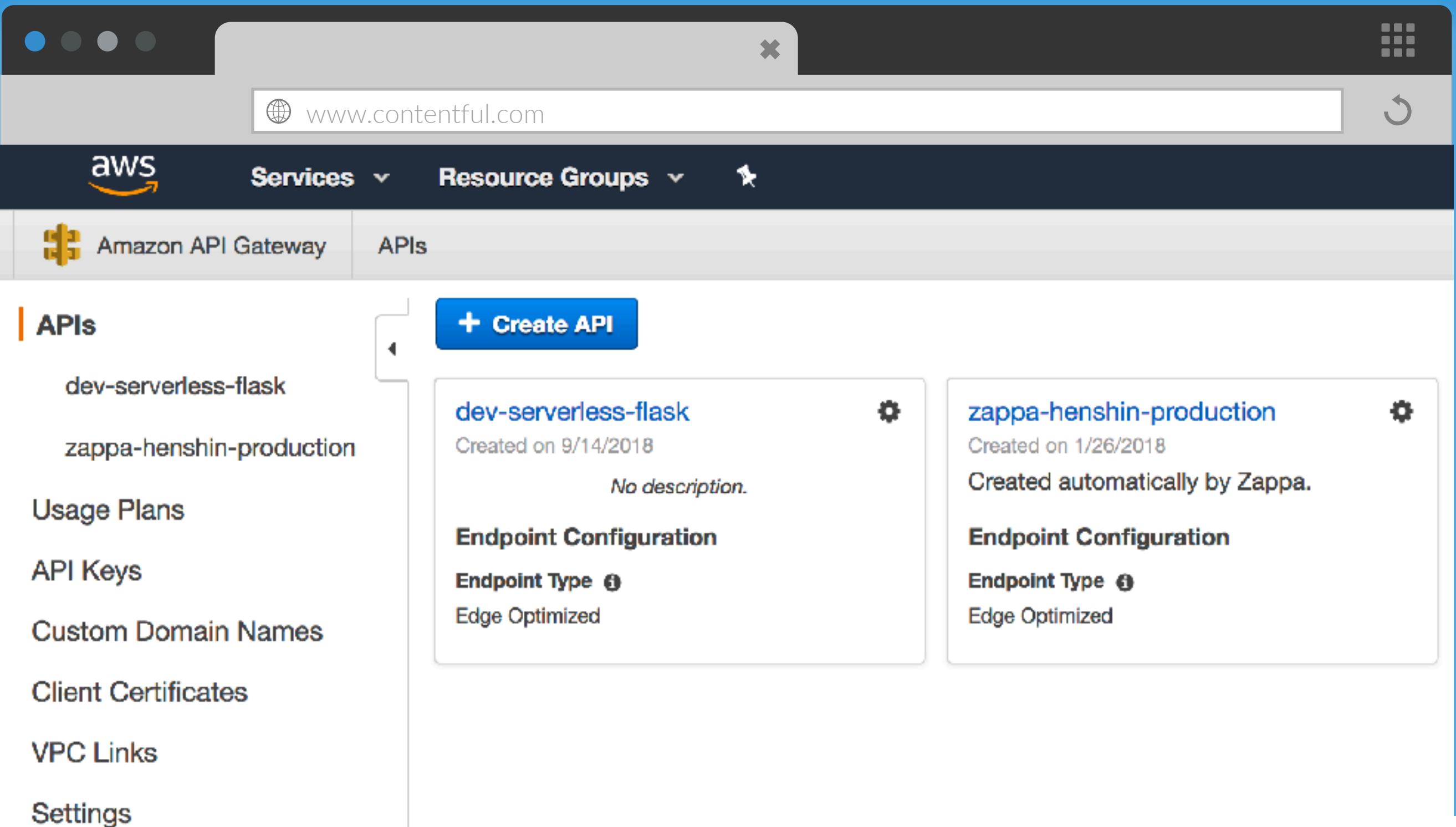
MacBook

SERVERLESS FRAMEWORK CONFIG

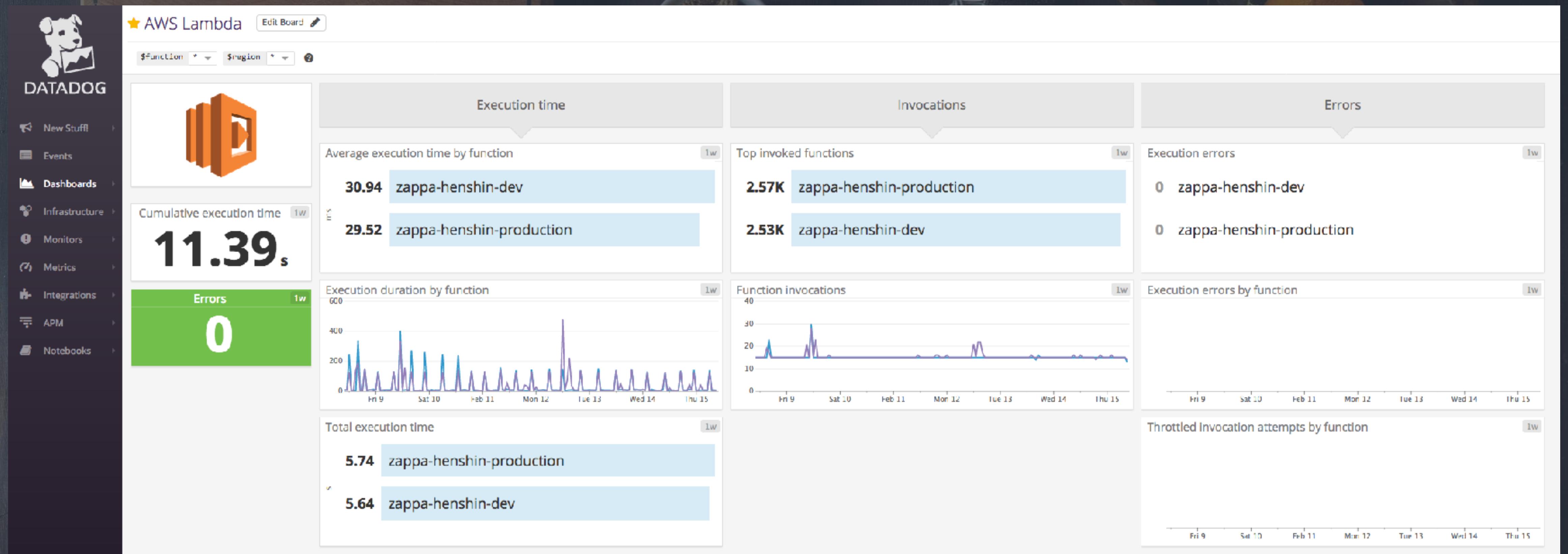


```
app.py serverless.yml
1 # serverless.yml
2
3 service: serverless-flask
4
5 plugins:
6   - serverless-python-requirements
7   - serverless-wsgi
8
9 custom:
10  wsgi:
11    app: app.app
12    packRequirements: false
13    pythonRequirements:
14      dockerizePip: non-linux
15
16 provider:
17  name: aws
18  runtime: python3.6
19  stage: dev
20  region: us-east-1
21
22 functions:
23  app:
24    handler: wsgi.handler
25    events:
26      - http: ANY /
27      - http: 'ANY {proxy+}'
```

BOTH DEPLOYMENT FRAMEWORKS HANDLE AWS LAMBDA AND API GATEWAY



Let's talk Metrics



30 ms

RENDER TIME

\$0.000000002/ms

COST ON LAMBDA

3,200,000

FREE FUNCTION CALLS ON LAMBDA

\$3.50

PER MILLION CALLS ON API GATEWAY

**Basically no time
TO GET UP AND RUNNING**



Shy Ruparel

Developer Evangelist | Contentful

@ShyRuparel

He/Him

Slides -> GitHub.com/shy/talks

Background Images -> wocintechchat.com