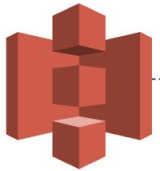
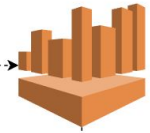




It's serverless all the way down!

Using serverless technologies to build, deploy, and serve scalable data pipelines


Who has used serverless technologies?

what is serverless

what is serverless **technology**

what is serverless **computing**

what is serverless **architecture**

what is serverless **framework**

what is serverless **application**

what is serverless **in aws**

what is serverless **computing in azure**

what is serverless **deployment**

what is serverless **computing in aws**

what is serverless **lambda**

Report inappropriate predictions

Why go serverless?

Why go serverless?

- Eliminates server maintenance

Why go serverless?

- Eliminates server maintenance
- Only pay for what you use

Why go serverless?

- Eliminates server maintenance
- Only pay for what you use
- Inherently and automatically scalable

Why go serverless?

- Eliminates server maintenance
- Only pay for what you use
- Inherently and automatically scalable
- Easy to deploy

When to go serverless?

When to go serverless?

- Non-constant workload

When to go serverless?

- Non-constant workload
- Volume-varying requests

When to go serverless?

- Non-constant workload
- Volume-varying requests
- Non-low latency requirements

When to go serverless?

- Non-constant workload
- Volume-varying requests
- Non-low latency requirements
- If you've asked "how can i meet these requirements with a service that's always available at a low cost?"

When to **NOT** go serverless?

When to **NOT** go serverless?

- Low latency responses

When to **NOT** go serverless?

- Low latency responses
- High memory computations

When to **NOT** go serverless?

- Low latency responses
- High memory computations
- Long running tasks

When to **NOT** go serverless?

- Low latency responses
- High memory computations
- Long running tasks
- Many tasks interacting with each other and require statefulness

how to build serverless

how to build serverless **applications**

how to build serverless **architecture**

how to **create** serverless.**yml file**

how to **create** serverless **website in aws**

how to **create** serverless **project**

how to **develop** serverless **application**

how to **create** serverless **database in azure**

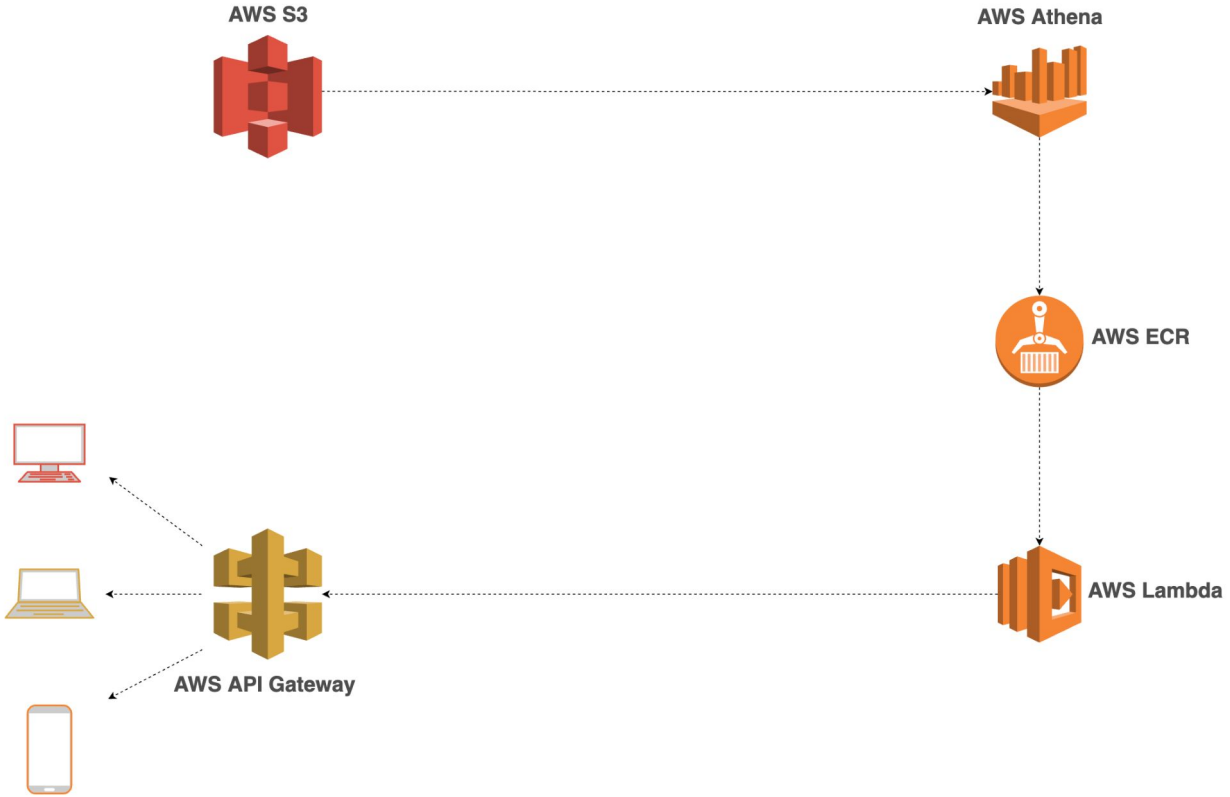
how to **create** serverless **sql pool in azure**

how to **create** serverless **architecture**

how to **create** serverless **api**

Report inappropriate predictions

Before giving an example, let's define some things



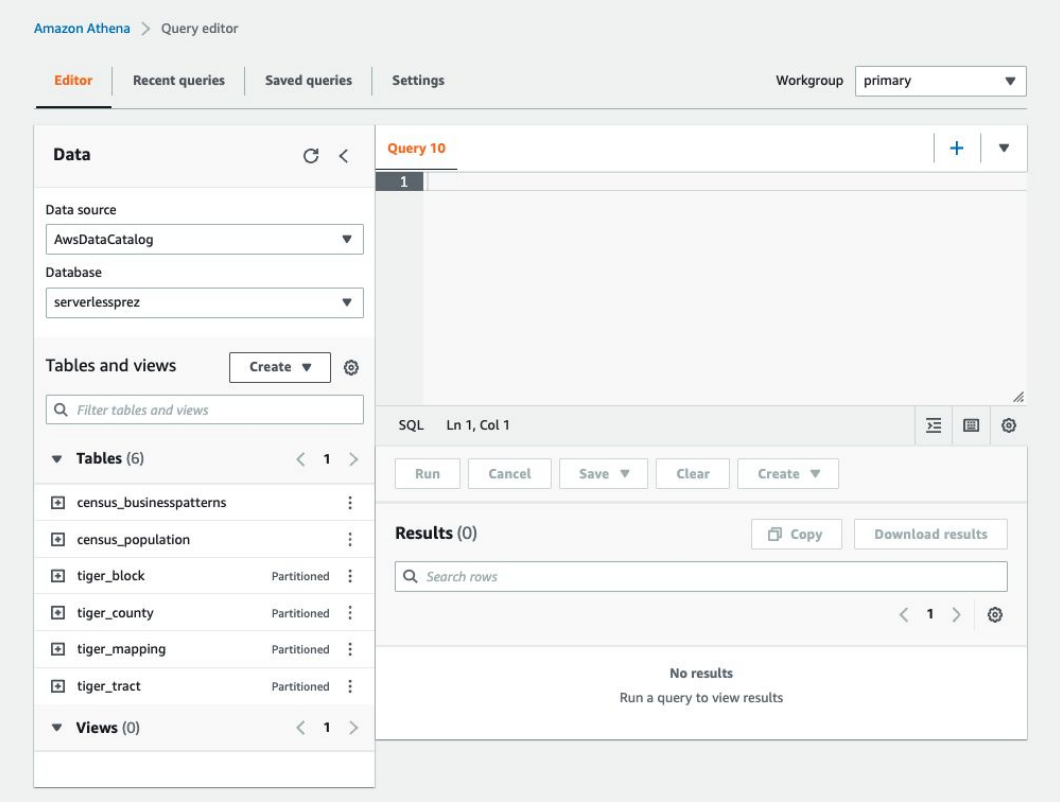
For the sake of time...

- I’ve already populated an **S3** bucket with data that we’ll retrieve here
- I’ve already created our **Athena** query engine that we’ll use to query our data in S3

Here's what our data in S3 looks like

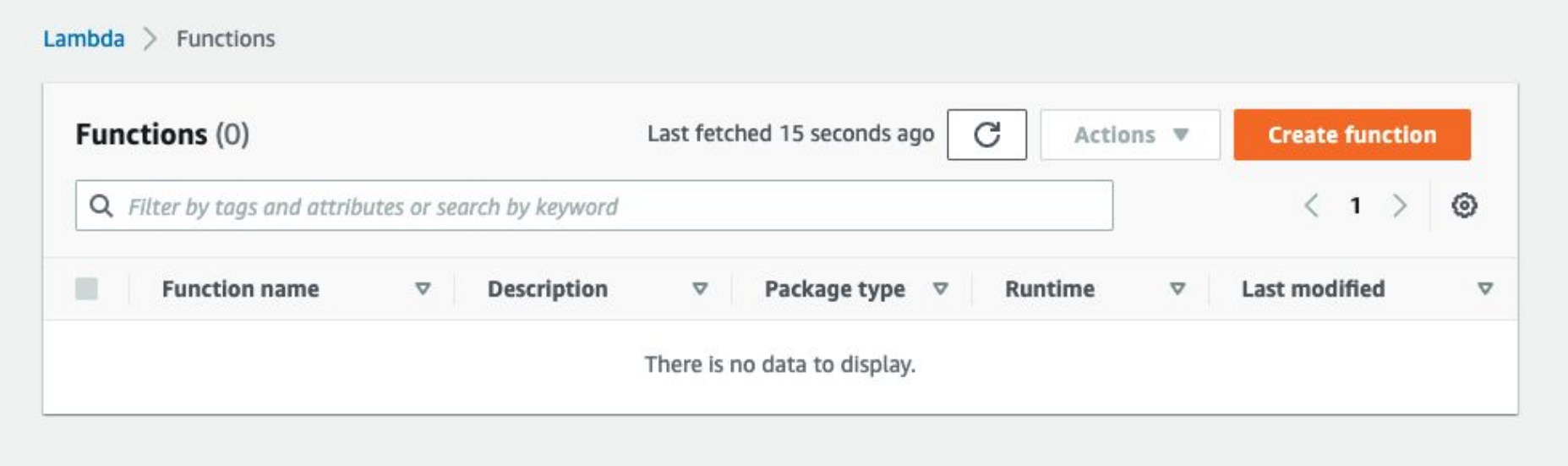
```
~/projects/serverless-prez  main
❯ tree -d -L 2 data/processed
data/processed
├── census
│   ├── businesspatterns
│   └── population
└── tiger
    ├── block
    ├── county
    ├── mapping
    ├── tract
    └── zcta

9 directories
```

Here's what our Athena environment looks like

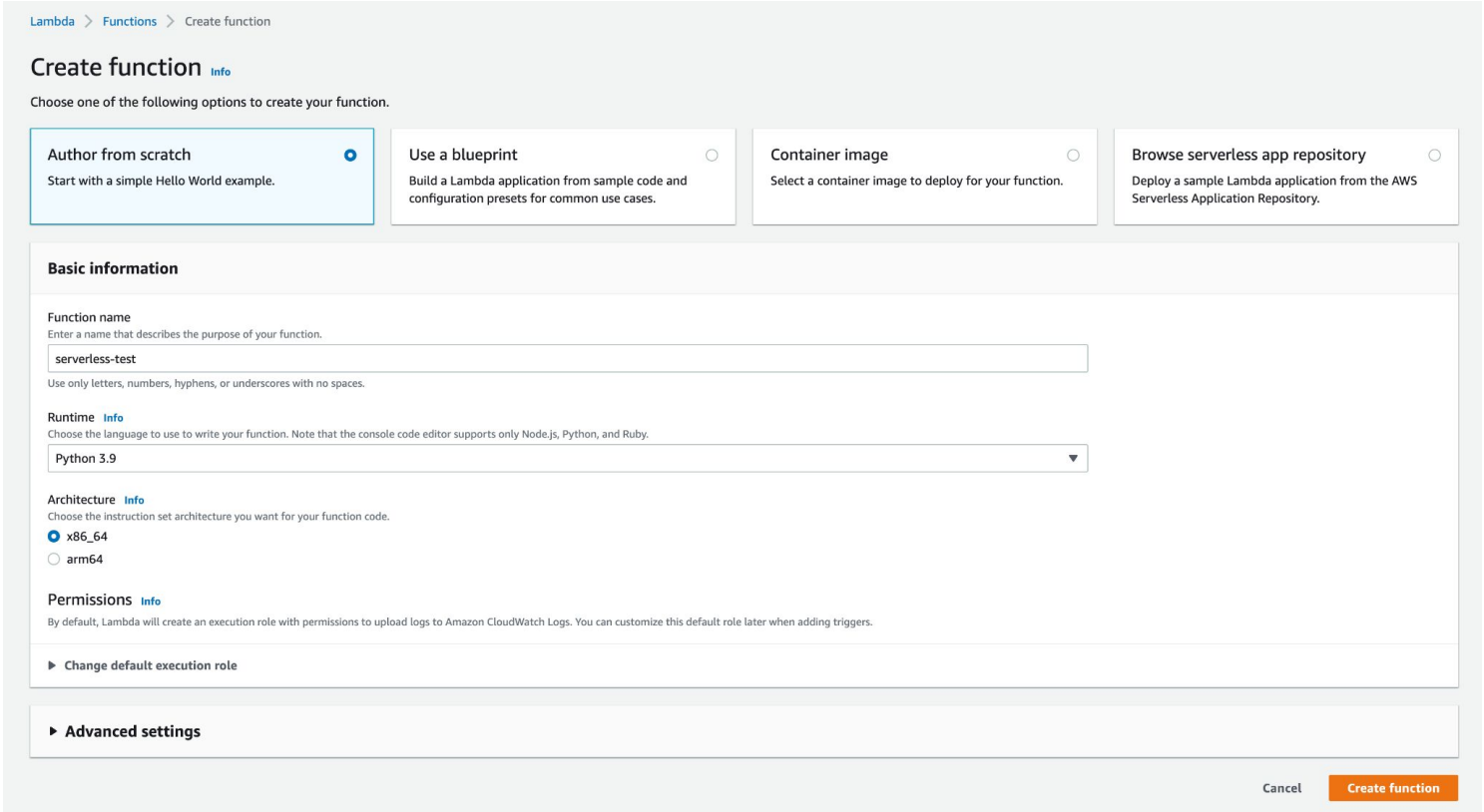


Now let’s build the rest of it!

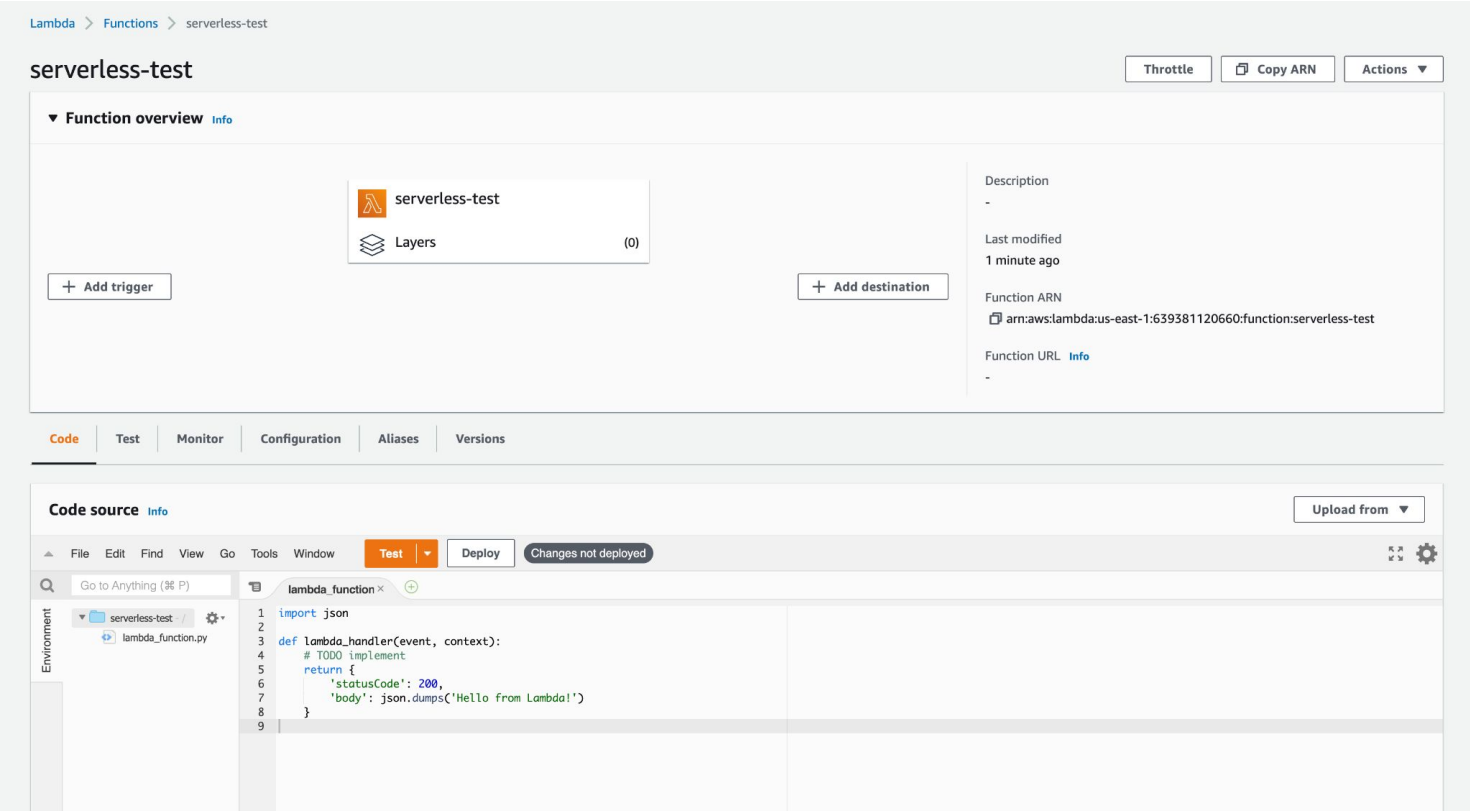
We'll start with creating our Lambda function



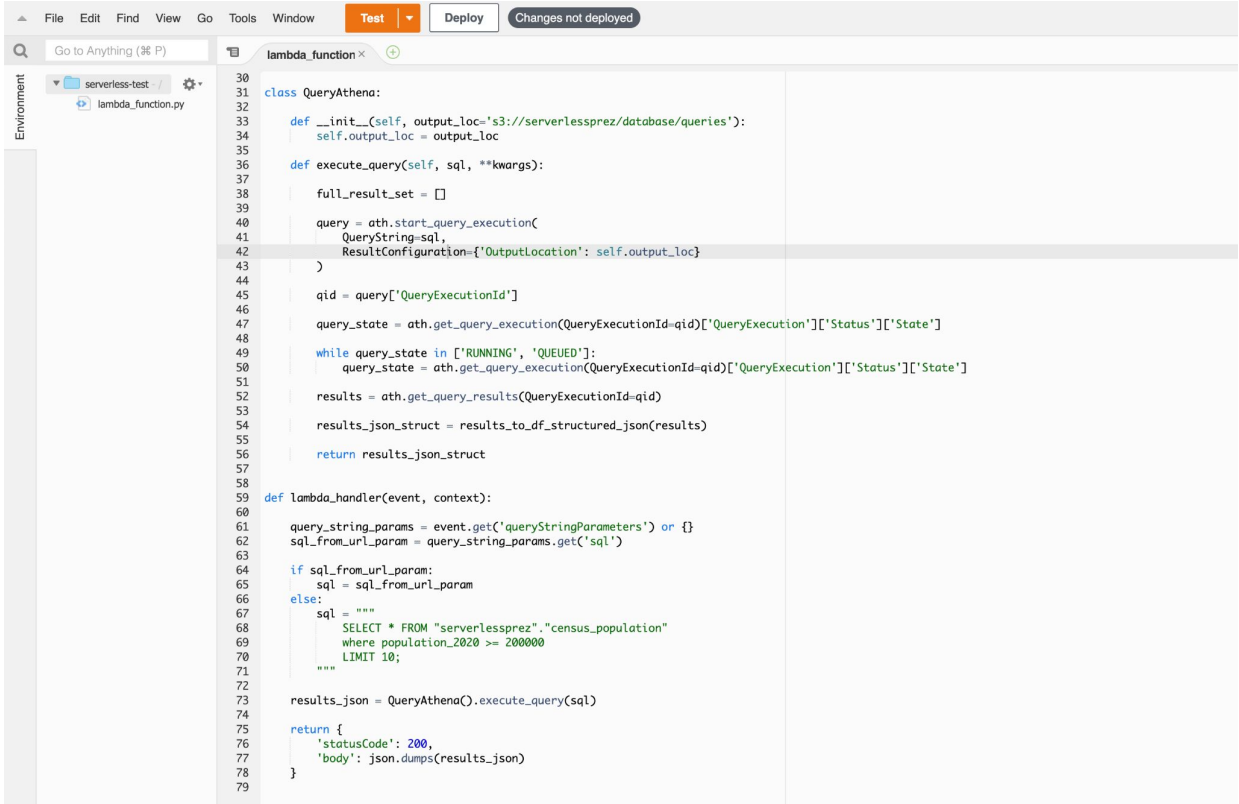
Configure and create



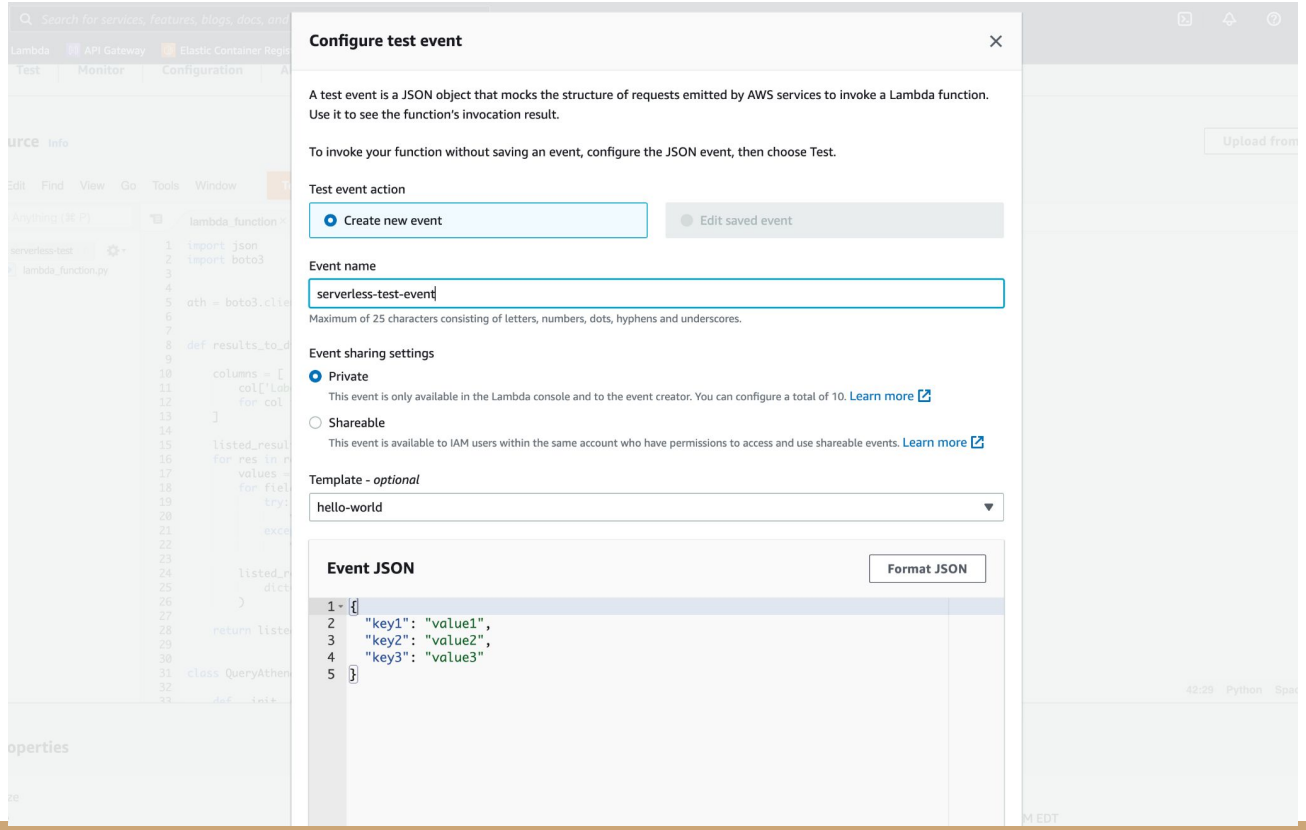
Edit function code



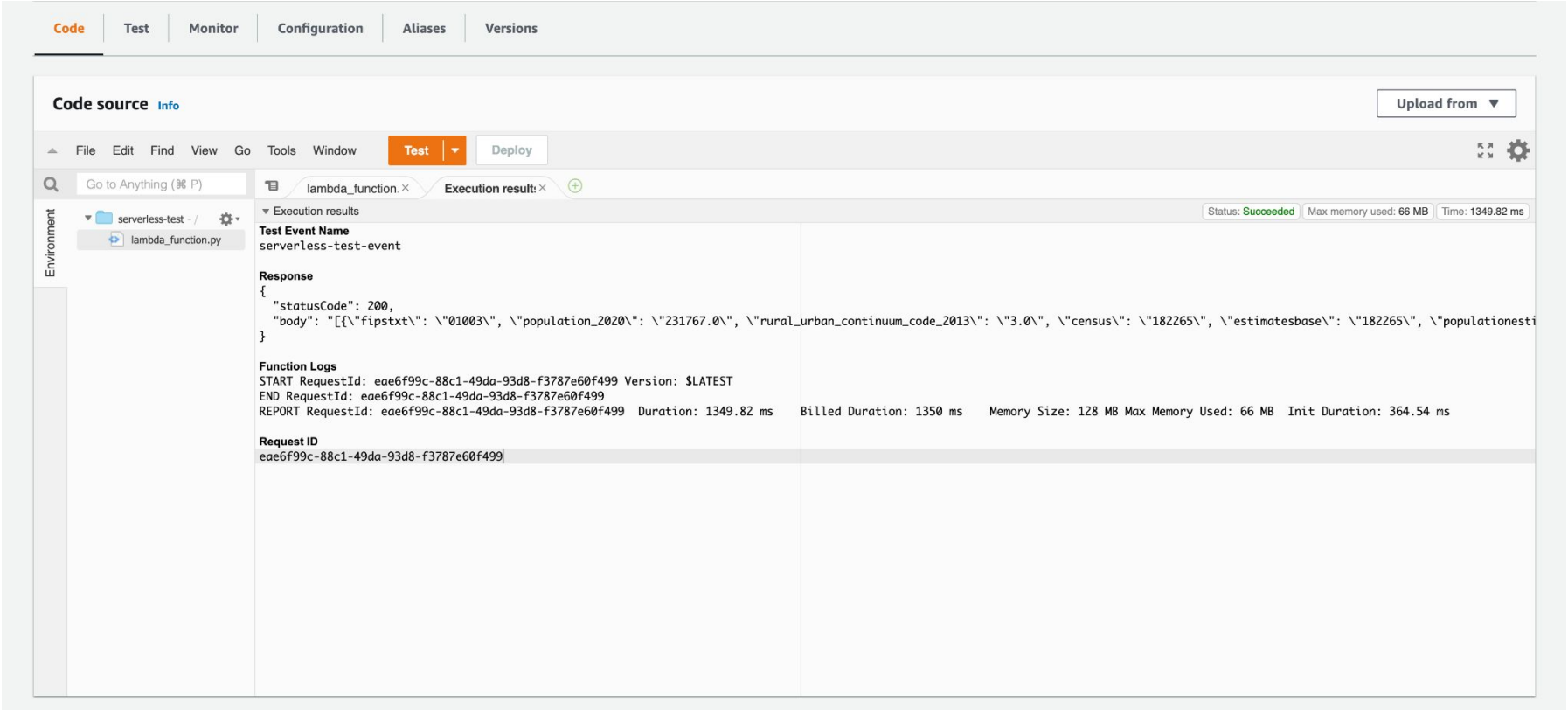
I'll use code I've already written for this



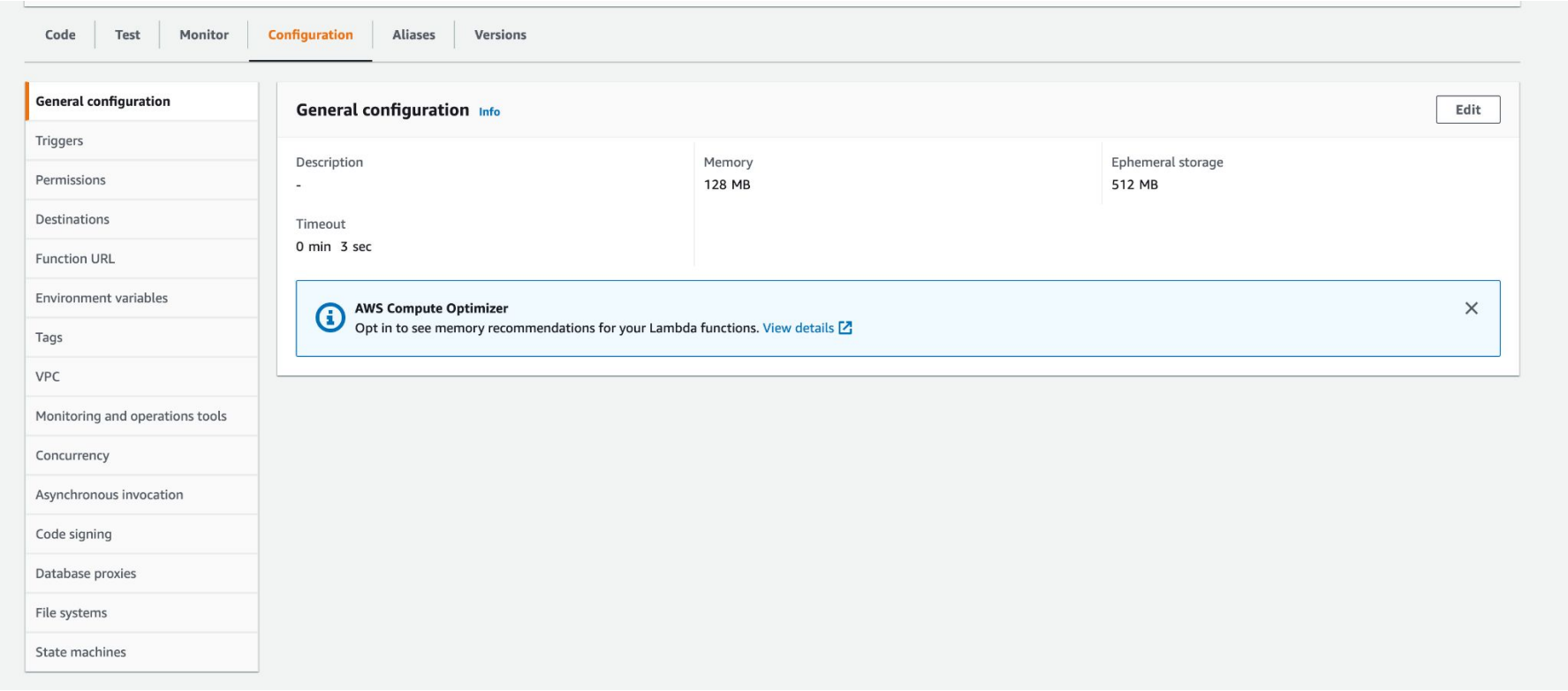
Test our Lambda function before deploying



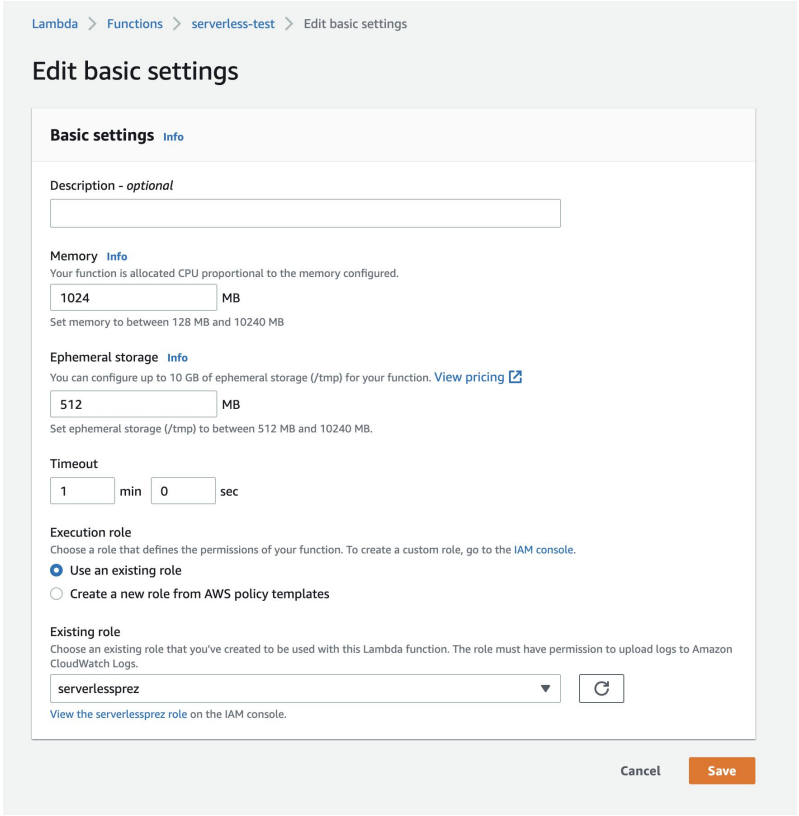
It works!


Code
Test
Monitor
Configuration
Aliases
Versions
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (⌘ P)
lambda_function ×
Execution result: ×
Environment
serverless-test
lambda_function.py
Execution results
Status: Succeeded
Max memory used: 66 MB
Time: 1349.82 ms
Test Event Name
serverless-test-event
Response
{
"statusCode": 200,
"body": "[{\"fipstxt\": \"01003\", \"population_2020\": \"231767.0\", \"rural_urban_continuum_code_2013\": \"3.0\", \"census\": \"182265\", \"estimatesbase\": \"182265\", \"populationesti
}
Function Logs
START RequestId: eae6f99c-88c1-49da-93d8-f3787e60f499 Version: $LATEST
END RequestId: eae6f99c-88c1-49da-93d8-f3787e60f499
REPORT RequestId: eae6f99c-88c1-49da-93d8-f3787e60f499 Duration: 1349.82 ms Billed Duration: 1350 ms Memory Size: 128 MB Max Memory Used: 66 MB Init Duration: 364.54 ms
Request ID
eae6f99c-88c1-49da-93d8-f3787e60f499

Now let's extend the timeout and memory

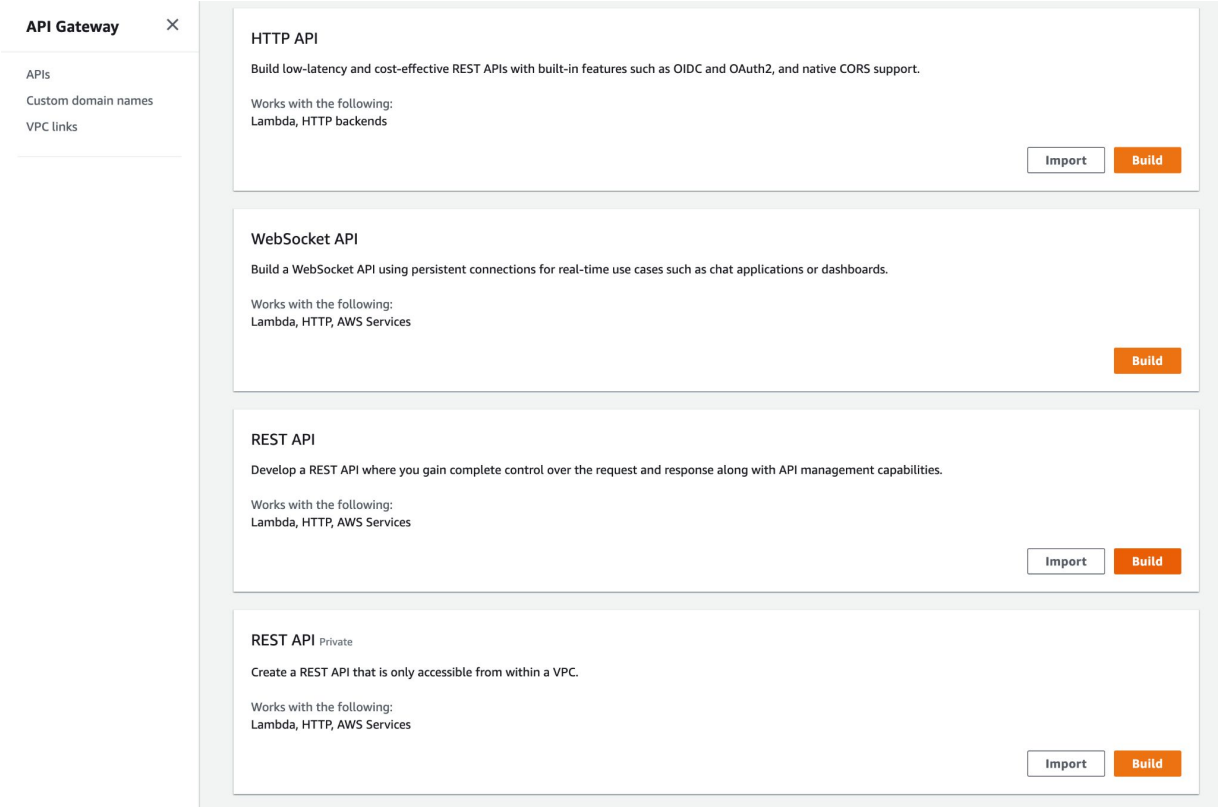


Now let's extend the timeout and memory

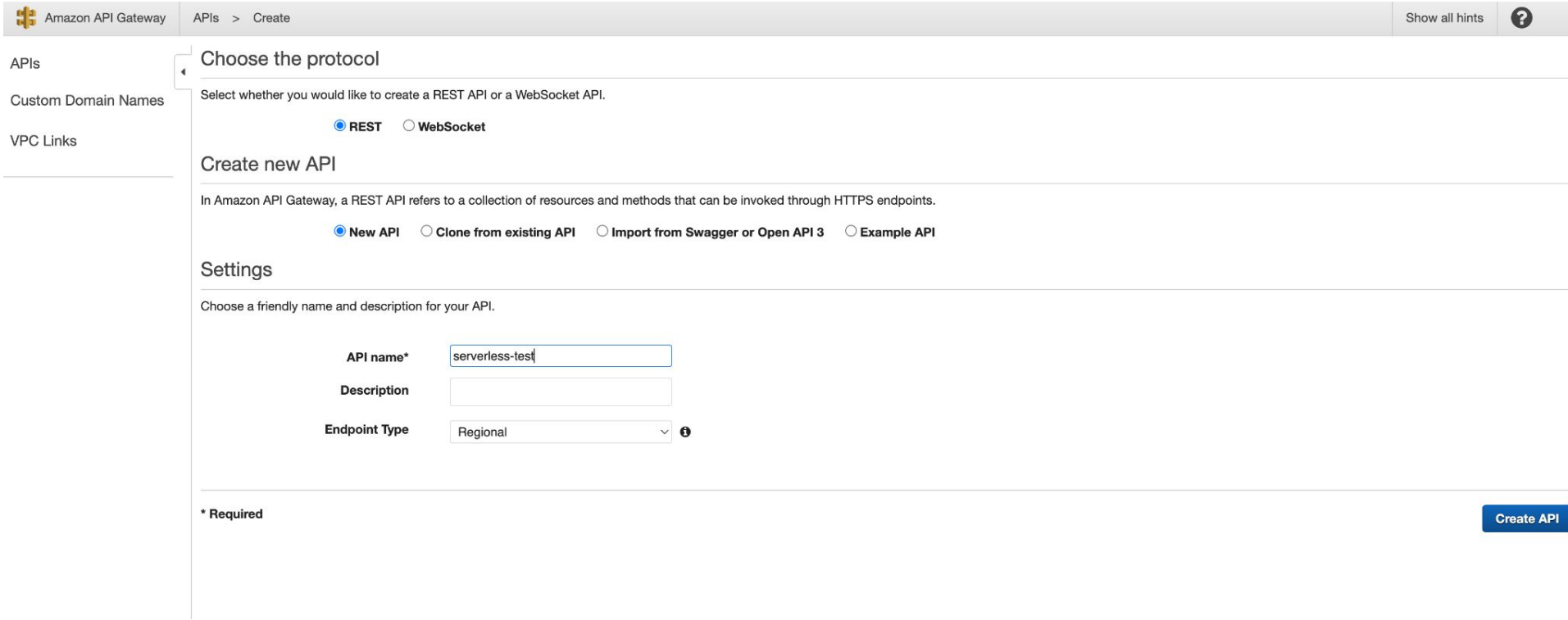


Now let’s make our data accessible over HTTPS

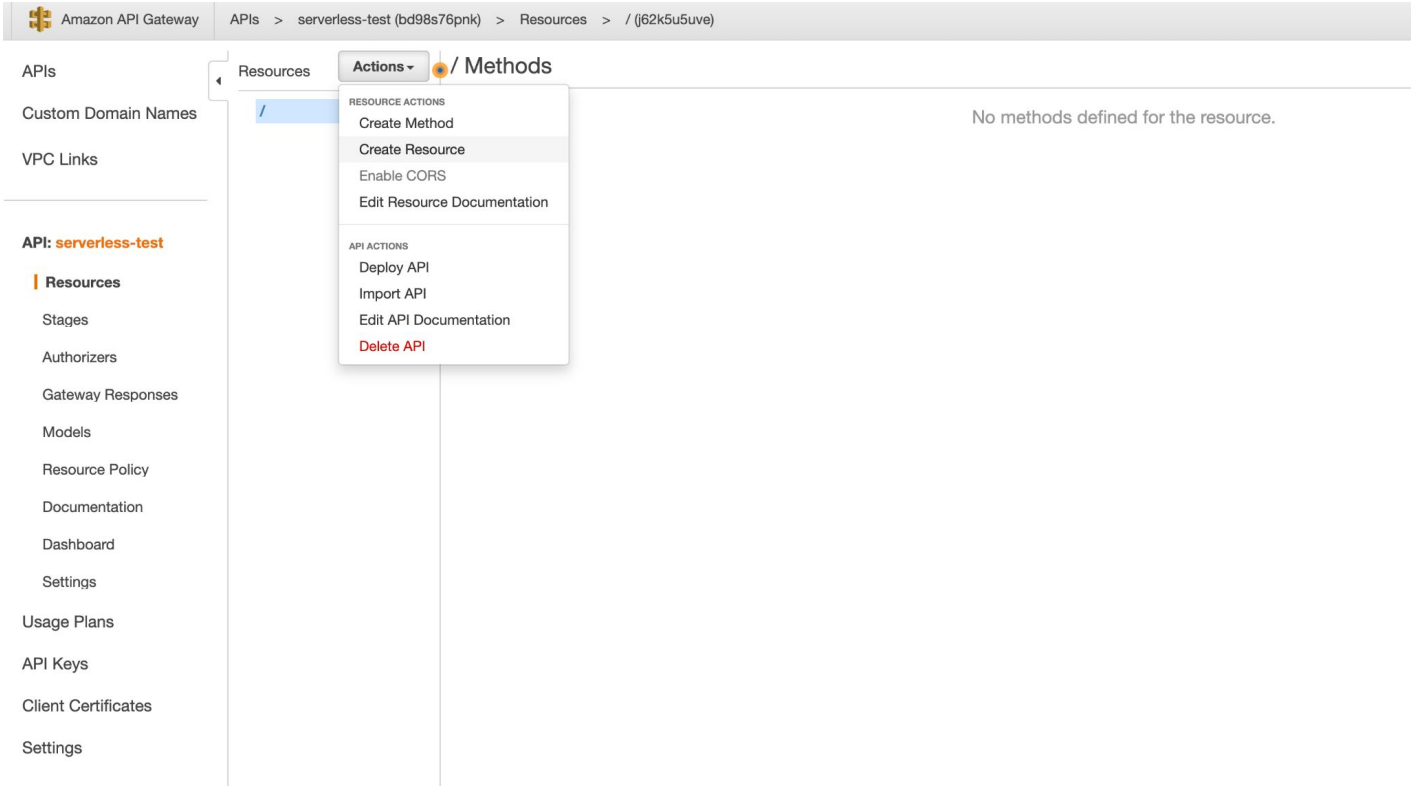
Create an API on AWS API Gateway



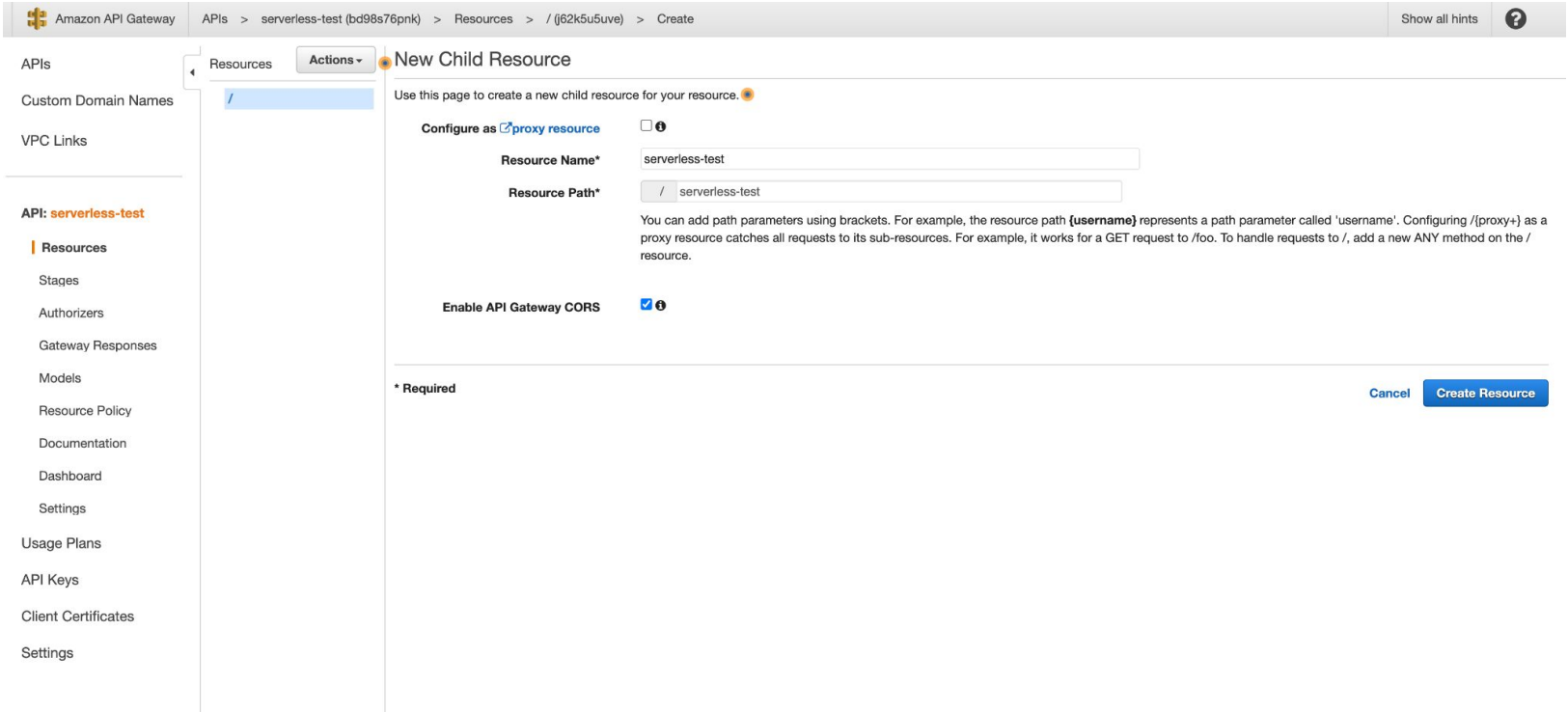
Name the API


Amazon API Gateway
APIs > Create
Show all hints
APIs
Custom Domain Names
VPC Links
Choose the protocol
Select whether you would like to create a REST API or a WebSocket API.
REST
WebSocket
Create new API
In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.
New API
Clone from existing API
Import from Swagger or Open API 3
Example API
Settings
Choose a friendly name and description for your API.
API name*
serverless-test
Description
Endpoint Type
Regional
* Required
Create API

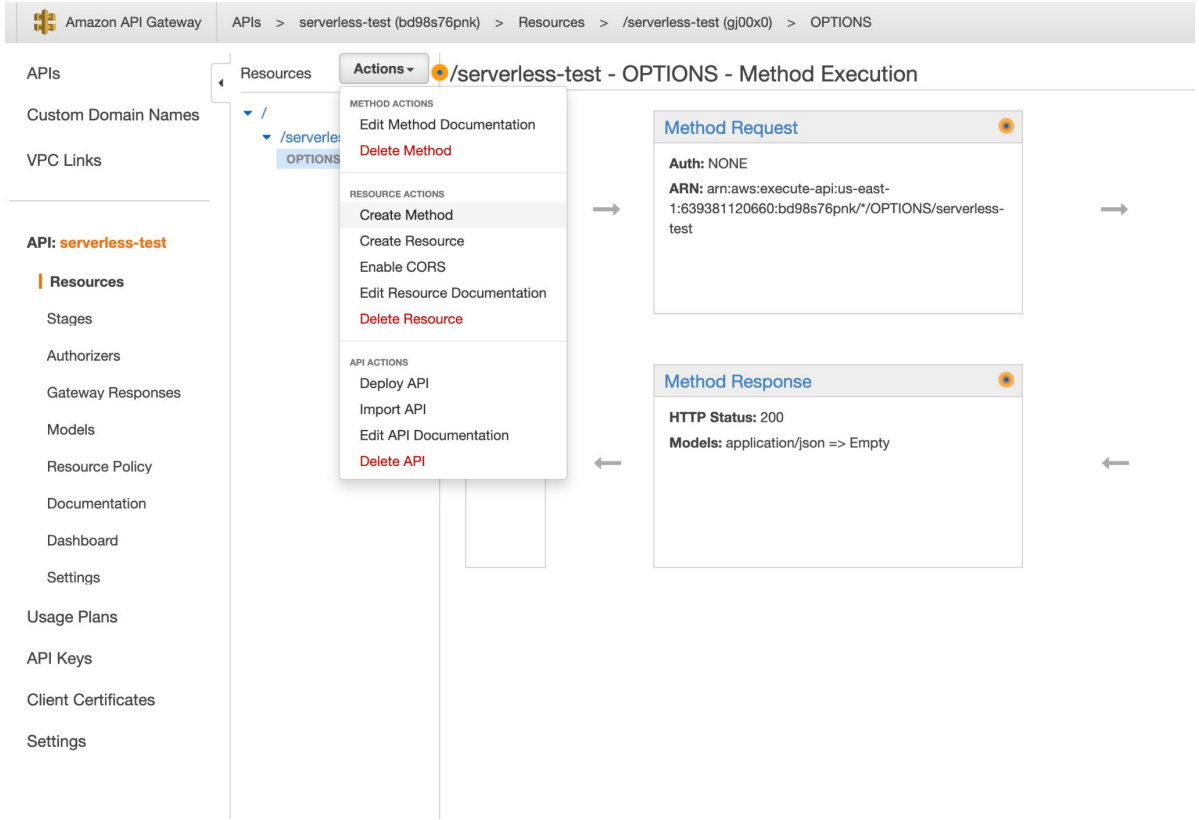
Create a resource on the API



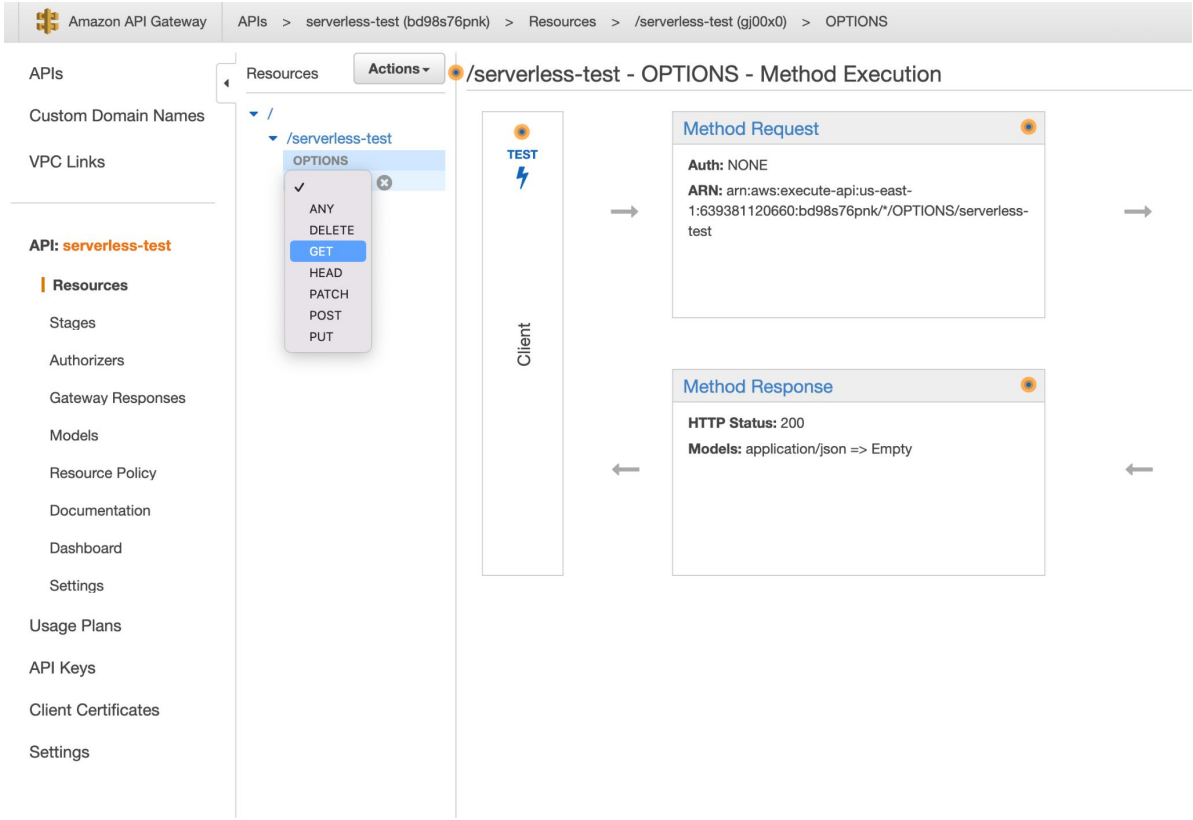
Create a resource on the API



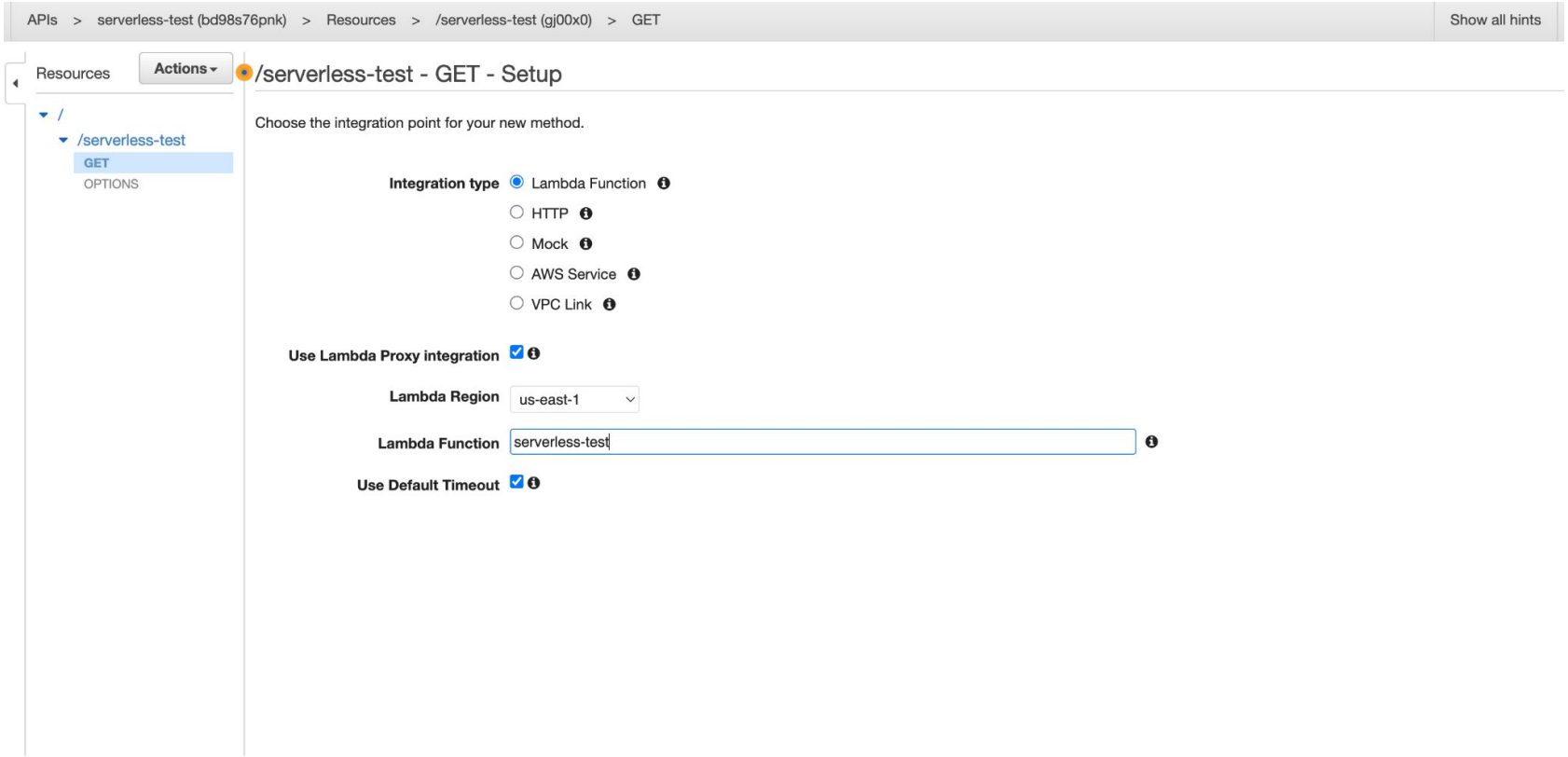
Create a method on the resource



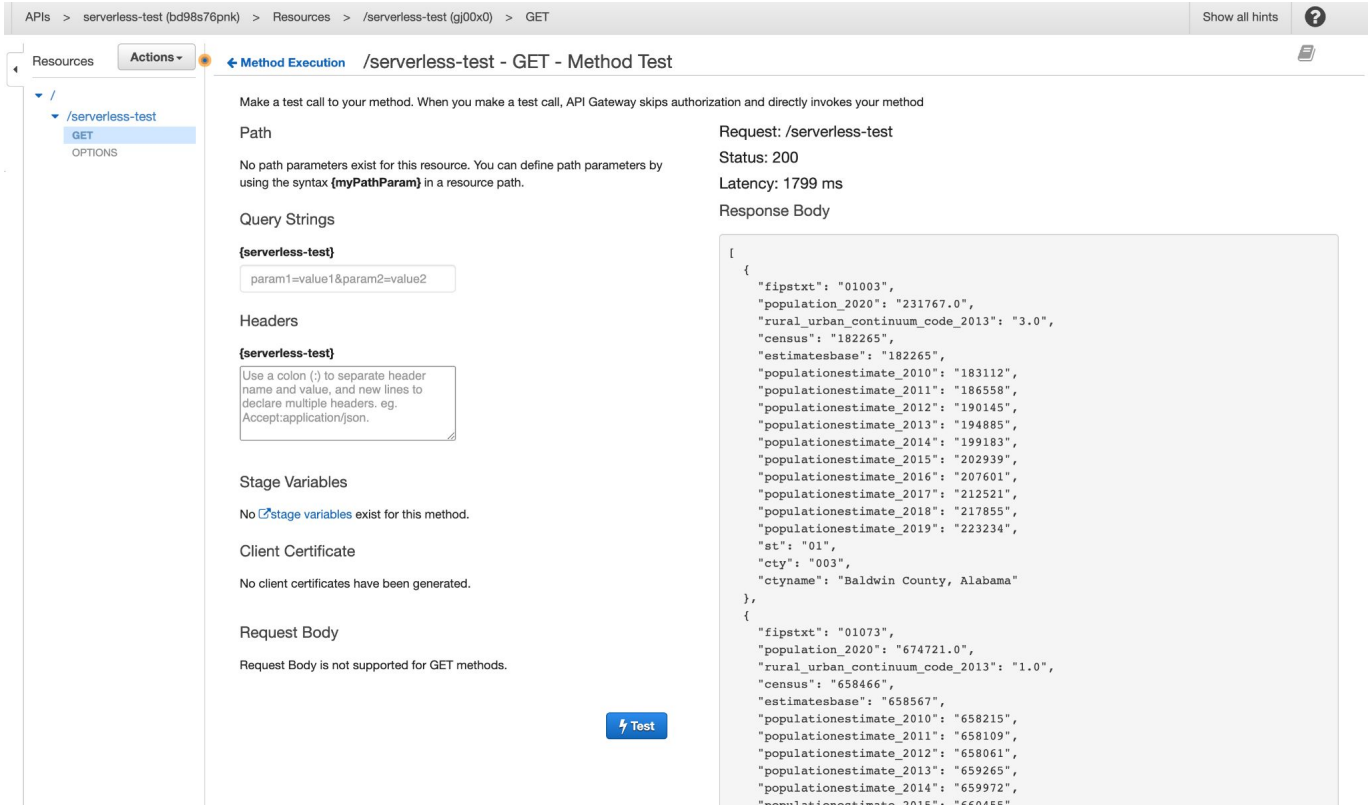
Create a method on the resource



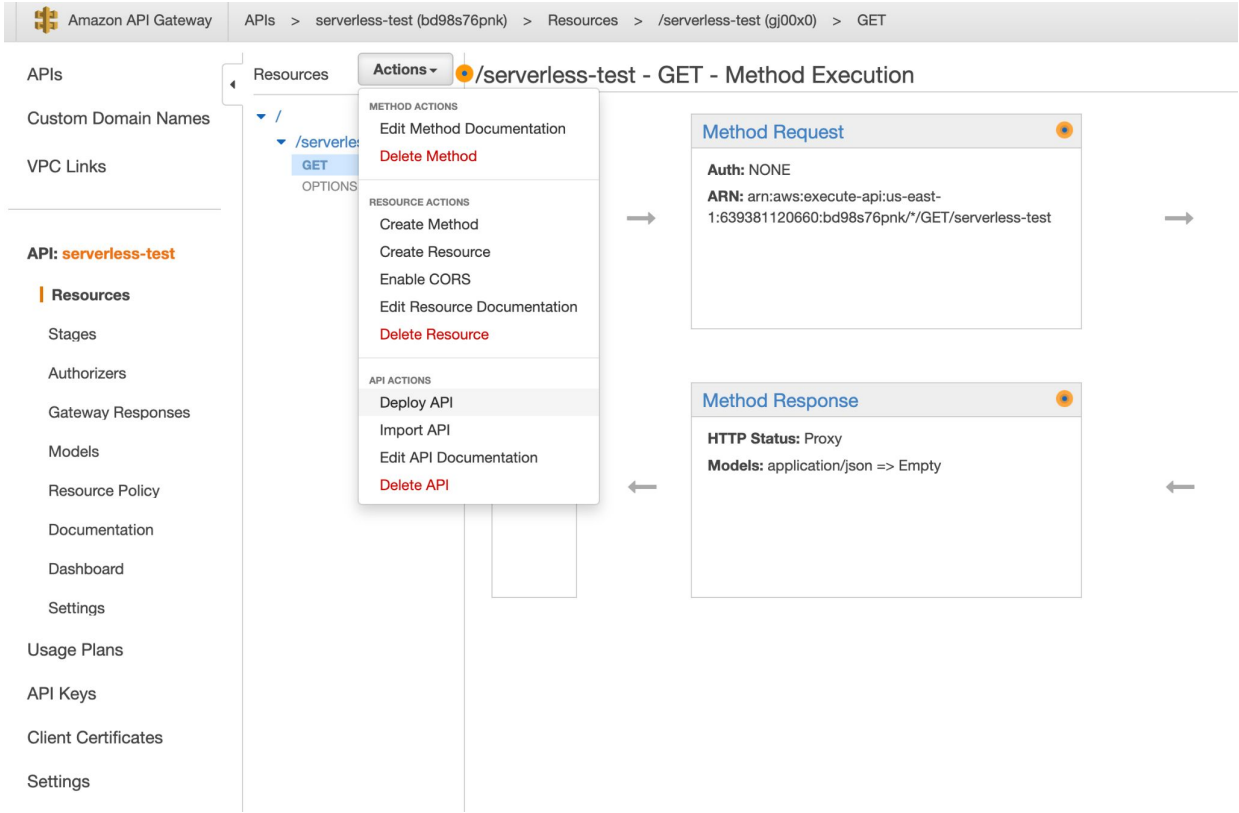
Use the Lambda function as the integration point



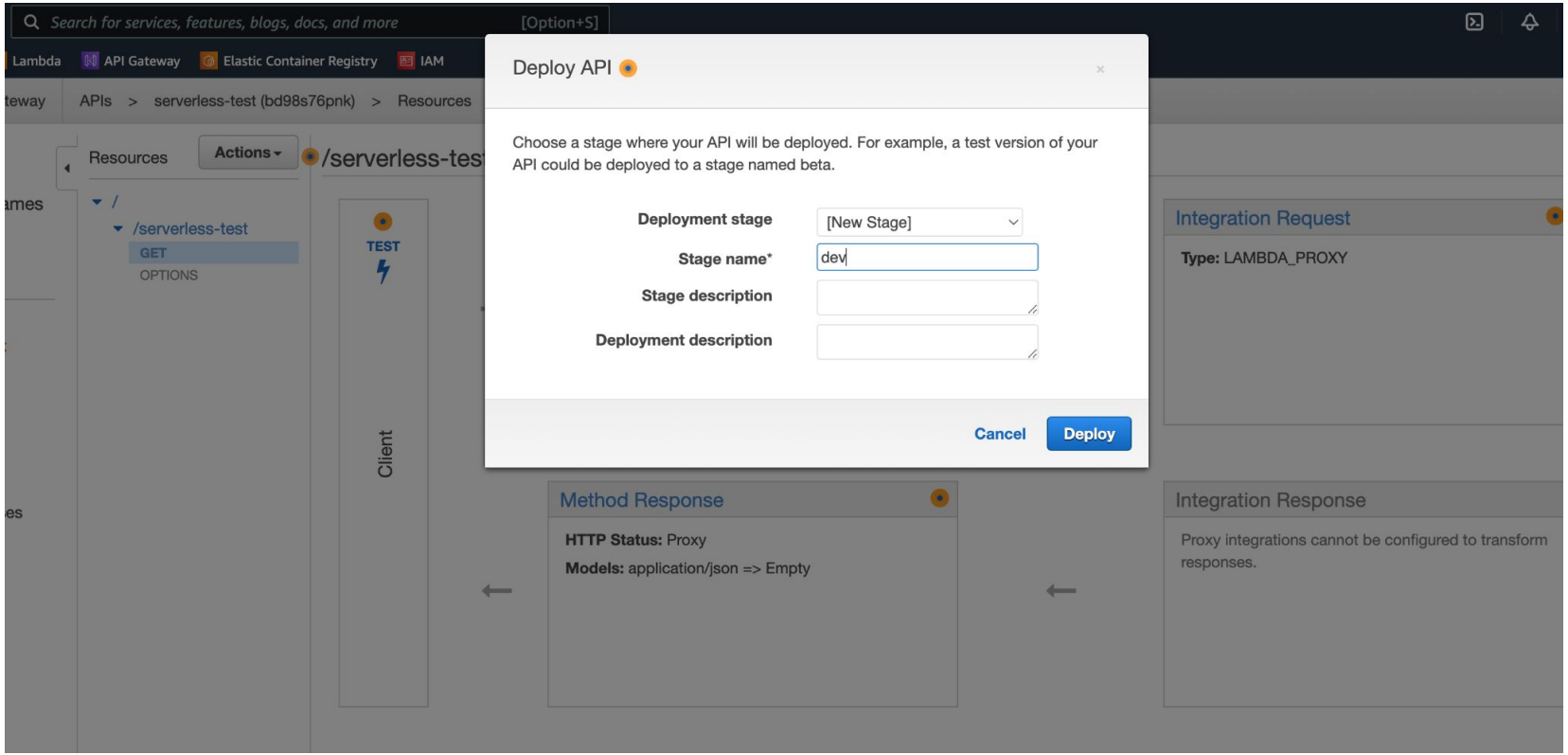
Test it out


APIs > serverless-test (bd98s76pnk) > Resources > /serverless-test (gj00x0) > GET
Show all hints
Resources
Actions
Method Execution
/serverless-test - GET - Method Test
/
/serverless-test
GET
OPTIONS
Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method
Path
No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.
Query Strings
{serverless-test}
param1=value1&param2=value2
Headers
{serverless-test}
Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.
Stage Variables
No stage variables exist for this method.
Client Certificate
No client certificates have been generated.
Request Body
Request Body is not supported for GET methods.
Test
Request: /serverless-test
Status: 200
Latency: 1799 ms
Response Body
[
{
"fipstxt": "01003",
"population_2020": "231767.0",
"rural_urban_continuum_code_2013": "3.0",
"census": "182265",
"estimatesbase": "182265",
"populationestimate_2010": "183112",
"populationestimate_2011": "186558",
"populationestimate_2012": "190145",
"populationestimate_2013": "194885",
"populationestimate_2014": "199183",
"populationestimate_2015": "202939",
"populationestimate_2016": "207601",
"populationestimate_2017": "212521",
"populationestimate_2018": "217855",
"populationestimate_2019": "223234",
"st": "01",
"cty": "003",
"ctyname": "Baldwin County, Alabama"
},
{
"fipstxt": "01073",
"population_2020": "674721.0",
"rural_urban_continuum_code_2013": "1.0",
"census": "658466",
"estimatesbase": "658567",
"populationestimate_2010": "658215",
"populationestimate_2011": "658109",
"populationestimate_2012": "658061",
"populationestimate_2013": "659265",
"populationestimate_2014": "659972",

Deploy the API (so we can access it)



Name the deployment



Now let's call it from Jupyter

In [1]:
```
%%time
import urllib.parse
import pandas as pd
import requests

sql = urllib.parse.quote_plus(
    """SELECT
        county.statefp,
        county.countyfp,
        pop.population_2020,
        pop.rural_urban_continuum_code_2013,
        pop.ctyname,
        ST_GeomFromLegacyBinary(county.geometry) as geometry
    FROM serverlessprez.census_population pop
    JOIN serverlessprez.tiger_county county
        ON county.statefp = pop.st
        AND county.countyfp = pop.cty
    WHERE pop.st = '47';"""
)

url = f"https://bd98s76pnk.execute-api.us-east-1.amazonaws.com/dev/serverless-test?sql={sql}"
resp = requests.get(url)
df = pd.DataFrame(resp.json())
```

```
CPU times: user 948 ms, sys: 269 ms, total: 1.22 s
Wall time: 4.83 s
```

In [2]:
```
df.head()
```

Out[2]:

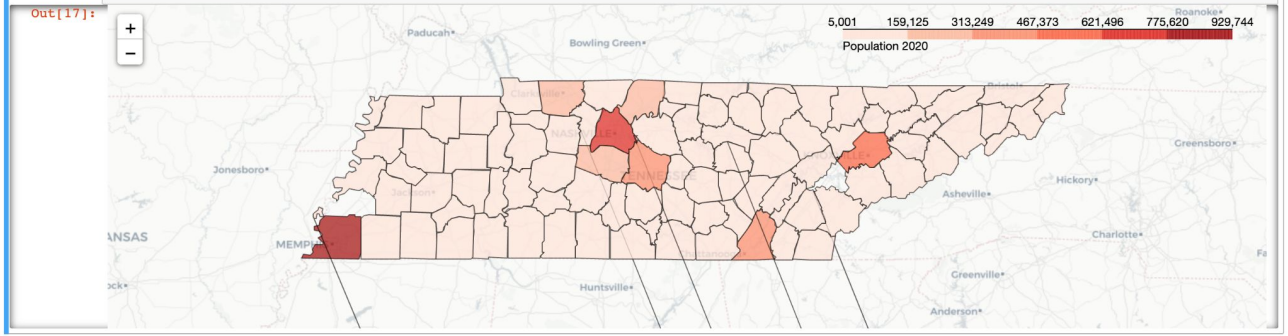
	statefp	countyfp	population_2020	rural_urban_continuum_code_2013	ctyname	geometry
0	47	039	11435.0	9.0	Decatur County, Tennessee	POLYGON ((-87.976434 35.506732, -87.976406 35....
1	47	043	54315.0	1.0	Dickson County, Tennessee	POLYGON ((0 -87.178306, -87.178306 36.228262, ...
2	47	049	18489.0	9.0	Fentress County, Tennessee	POLYGON ((-84.852506 36.292433, -84.852417 36....
3	47	057	23527.0	2.0	Grainger County, Tennessee	POLYGON ((-83.381502 36.265431, -83.381297 36....
4	47	059	70152.0	4.0	Greene County, Tennessee	POLYGON ((-82.785041 35.987096, -82.785074 35....

Doing more stuff with that data in Jupyter

In [17]:

```
import geopandas as gpd
import folium

df['geometry'] = gpd.GeoSeries.from_wkt(df['geometry'])
gdf = gpd.GeoDataFrame(df, geometry="geometry").reset_index()
gdf['population_2020'] = gdf['population_2020'].astype(float)
gdf = gdf.set_crs(epsg = 4326)


m = folium.Map(location=[35.59, -86.94], zoom_start=7, tiles='CartoDB positron')

choropleth = folium.Choropleth(
    geo_data=gdf.to_json(),
    name='choropleth',
    data=gdf,
    columns=["ctyname", "population_2020"],
    fill_color='Reds',
    fill_opacity=0.75,
    line_opacity=.6,
    key_on="feature.properties.ctyname",
    legend_name='Population 2020',
    highlight=True,
    reset=True
).add_to(m)

choropleth.geojson.add_child(folium.GeoJsonTooltip(['ctyname', 'population_2020'], labels=False))

m
```

Out[17]:



How would I have built this using non-serverless?

- Create an AWS EC2 instance
- Build a postgres database on the EC2 instance
- Create the API endpoint using a web framework like Flask or FastAPI
- Put the python code behind the Flask resources
- Paid too much

Need help with a project or have questions?

Timothy Dobbins
Data Scientist|Data Engineer|Consultant

Github: github.com/tmthyjames
Linkedin: linkedin.com/in/tim-dobbins
Email: tmthyjames@gmail.com

Or scan this →

Presentation Github link: github.com/tmthyjames/serverless-prez