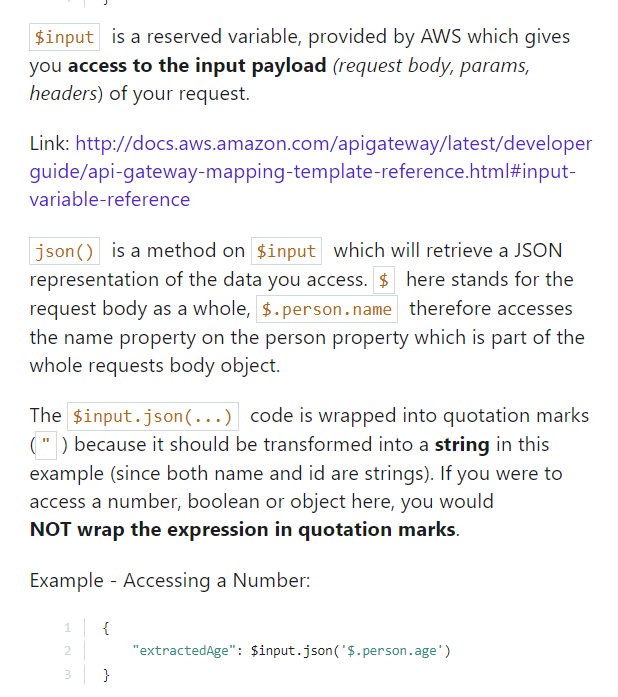
1-4 BodyMappingTemplates

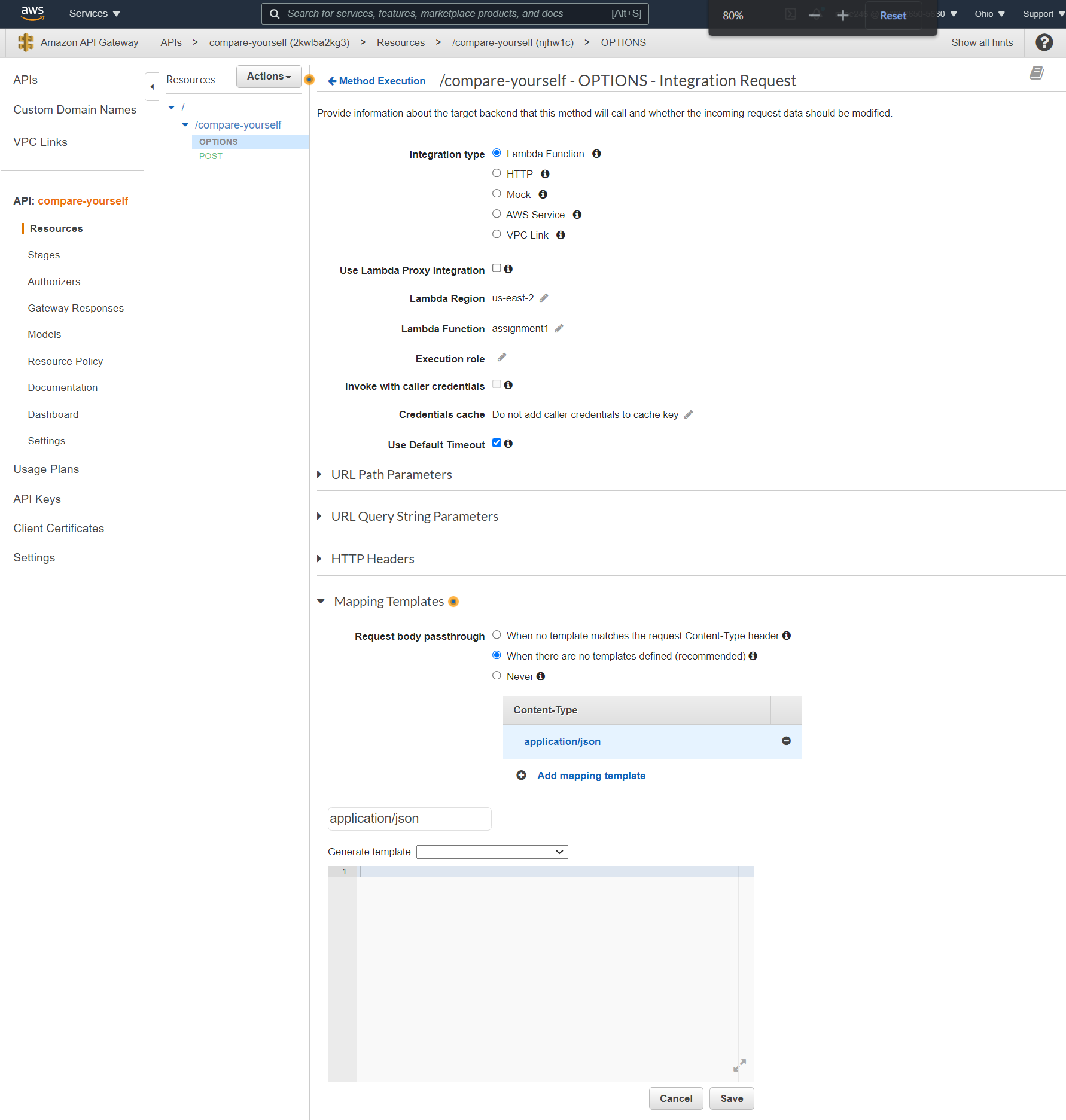
Monday, August 23, 2021

8:13 PM

Body Mapping Templates can be confusing, but in the end, 
they're not that difficult. 
You use the Apache Velocity Language when working with 
these templates, that Will only matter once you create more 
complex transformations though. Basic explanations (see 
below) should be relatively self-explanatory 
First, you need to understand that they simply allow you to 
control which data your action receives (e.g. Lambda). 
You can for example transform incoming data of the following 
shape: 
With the following template: 
"*Tsonuæ": 

**Why do we even use body mapping templates?**

In the integration request and response, we can use body mapping templates to map in the case of request, the data we pass to the action and in the case of response, data we get out of that action. So here integration request ,we can scroll down and first of all switch this to when there are no templates to defined



This is forwarding a mapped template or a mapped event data object with data form that request which is also the reason why we wouldn't have to parse JSON data here because it already mapped Javascript object instead.

We just add a template call "application/json"

\*\*\*Important\*\*\*

This name is not random! "application/json" means that incoming requests with "Content-Type" of "application/json" will be handled by this template!

』

**Let's make a template**

application/json 
Generate template: 
2 "body-json": S input. j son('$') 

{

"age" :28,

"height" : 72,

"income" : 2500

}

{

"your-age": $input.json('$')

}

$input => Variable provided by AWS, gives you access to your request data (body, params, …)

json('$') => Extracts complete request body

application/json 
Generate template: 
2 "age": *input .json( 'S. person. age') 

our template to extract 'age'

Request Body 
"person" • 
"nane" : " 
"age" : 21 
Test Request: /compare-yourself 
status: 200 
Latency: 115 ms 
Response Body 

CloudWatch logs

d263bcc2.gøc9-a91f.a91b-enbd5S2ggas INFO { age: 
24 } 
: 21. OIBZ 
d263bcc2-8ec9-a97f- a91b-e33bd552gga5 INFO 
{ age: 24 } 

고

We have no other metadata about the requests because in our mapping template under integration request here, we only extract that and the difficult part is to grasp the language we use with this **$input.json('$')**

**Idea behind Body Mappings**

$input refers to request data (e.g. params, body)

'$' refers to just the request body and with '.' notation infers different level of layouts, meaning accessing data

In short,

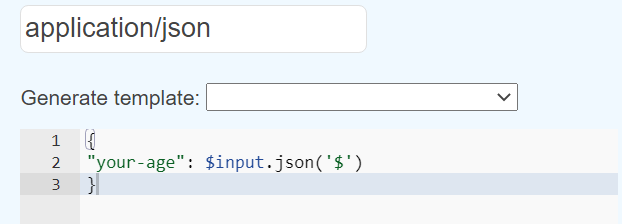
We are basically restructuring the input data and mapping a complex data to an easier one.

There is a clear separation between API Gateway where we receive our request and work. And lambda where we expect to get data in a certain structure and then do something with it.

Therefore lambda that doesn't have to care about which structure our request has, an API Gateway simply needs to provide the right structure to lambda.

屯

**Mapping Response Data**



=> this is integration response mapping templates

$input is the callback info sent back by Lambda,

json('$') then gives us the data passed with the callback

Request: 'compare-yourself 
status: 200 
Latency: 120 ms 
Response Body 
"your-age" • 
. 52 Request Body 
" person" 
"name" • 

This is how we can also map the response, lambda gives us a number but let's say our client, our application which calls the API in the end expects to get an object where this is stored in a your age property, nothing easier than that.

We don't have to adjust our lambda function to our client, we only have to adjust our API Gateway to our client and transform the data we get by lambda into the format we want to use and that's the power of body mapping templates.

We can clearly separate lambda from API Gateway and we can control what goes into lambda and what you do with what comes out of lambda.