Serverless Development 101

**Module 06 – Protecting Your API with AWS IAM**

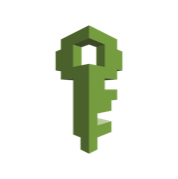
1/10/2019 Developed by Kevin Wang

1/10/2019 Checked by Clark Jason Ngo

1/10/2019 Tested by Tuan Khai and Minh Truong

1/17/2019 Revised by Sam Chung

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

**Learning Outcomes**

* Learn API types
* Ways to secure APIs
* Learn how to secure an internal API with IAM

Different types of API and how to secure them

1. Public APIs

Your public content (Your homepage index.html)

1. Internal APIs

Some APIs should just be called by your functions (Fetch books API)

1. Authenticated APIs

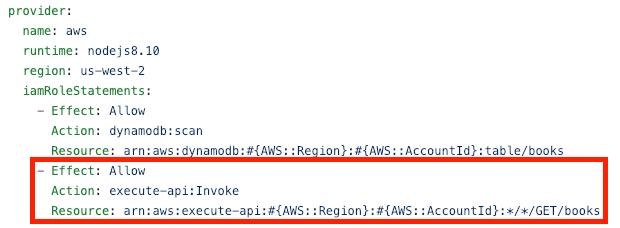
Some APIs should just be used only by your users (Search books API)

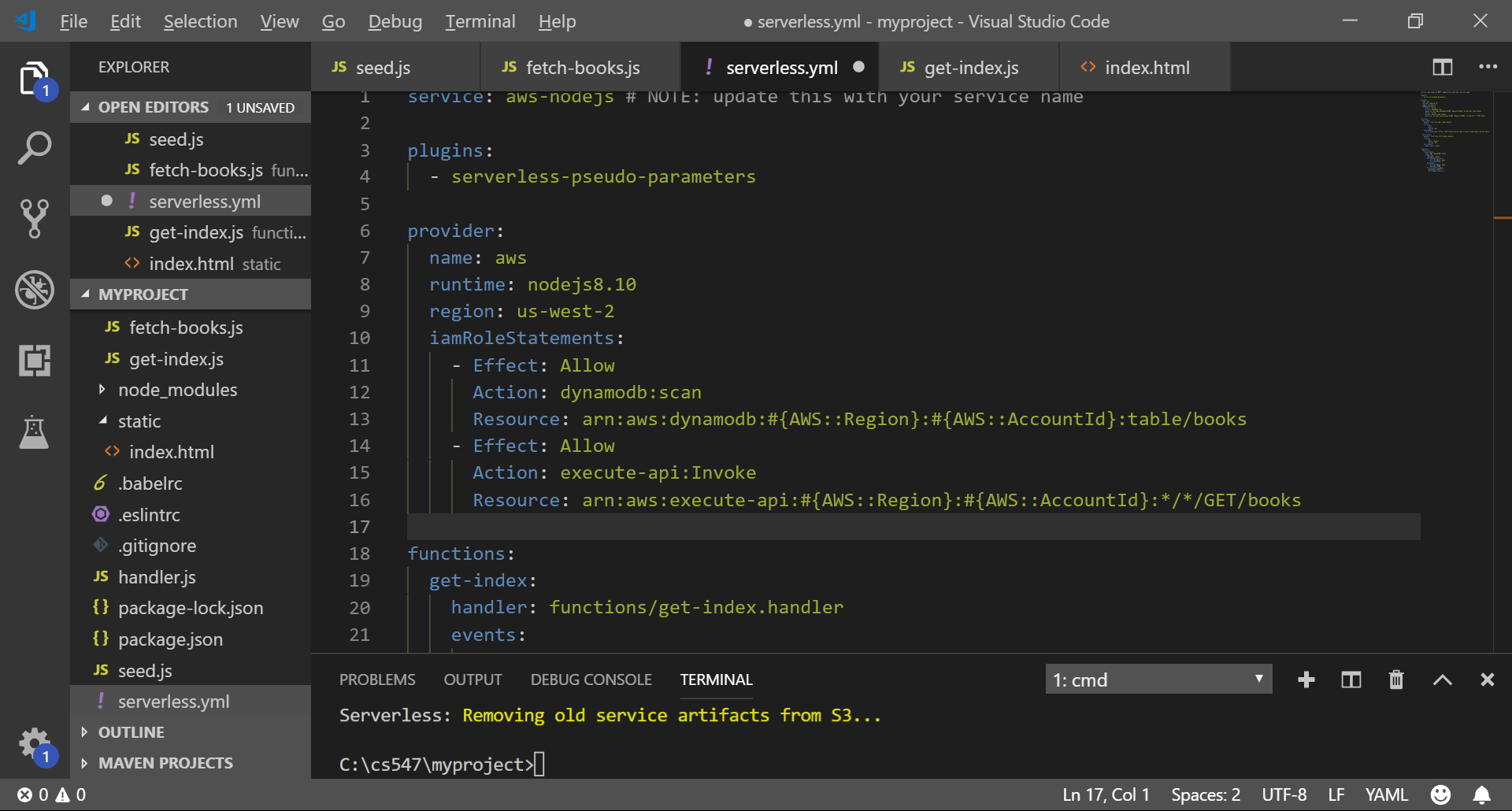
1. APIs for third parties

Some APIs you would like to share with others

**Secure Your Internal API**

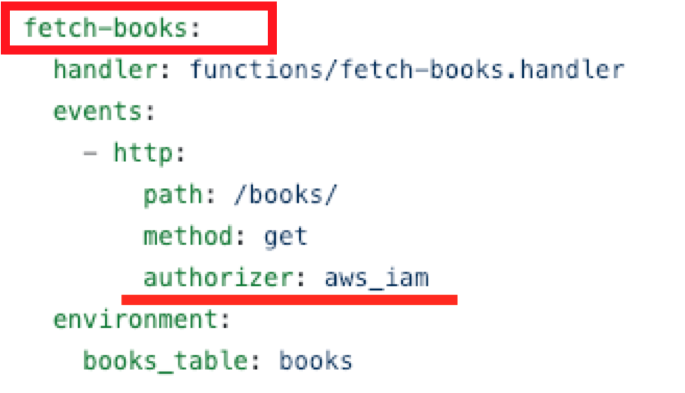
1. Open the VSCode and open the “**myproject**” project folder that we developed a web application using serverless computing in the previous module.
2. Open the “**serverless.yml”** file under the root folder and add changes as below.
3. Give the permission to call the fetch-books API

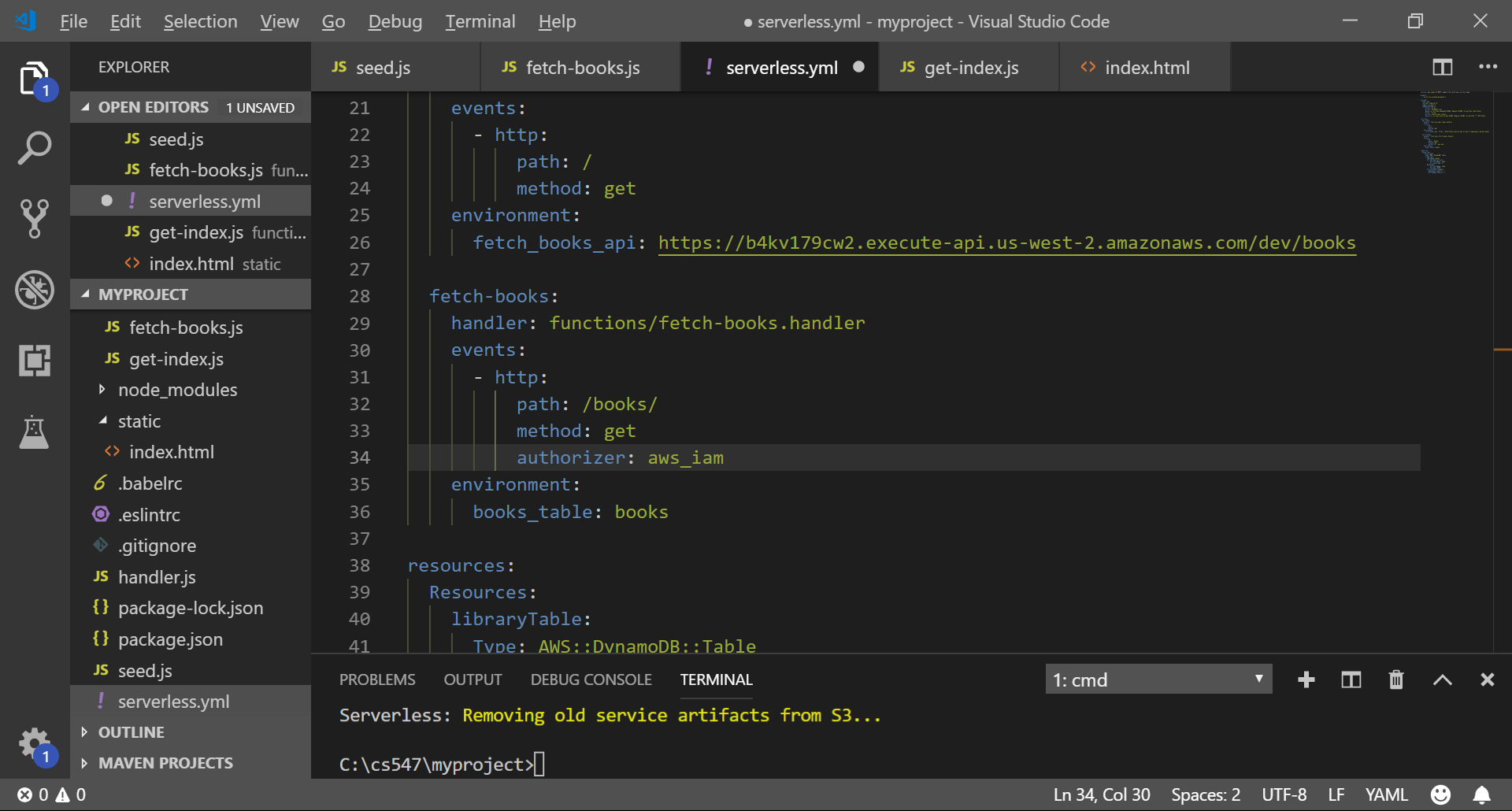
**** 



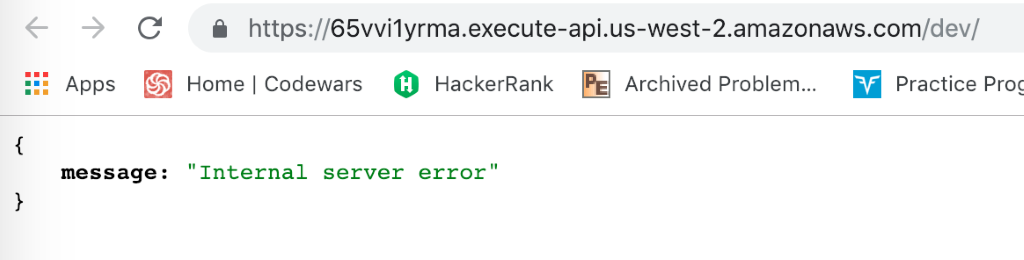
1. Require the fetch-books function to use IAM as the authorizer

Note: add the following code under **fetch-books:**

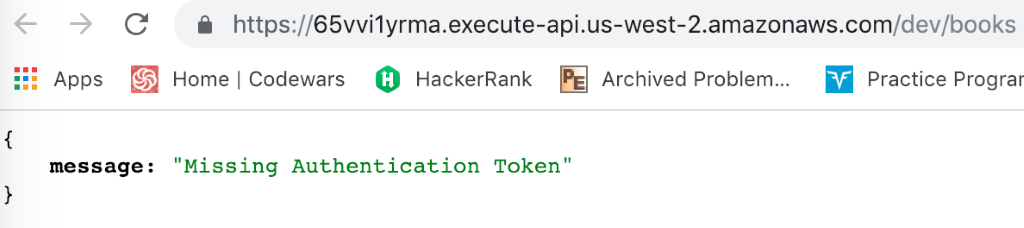
****



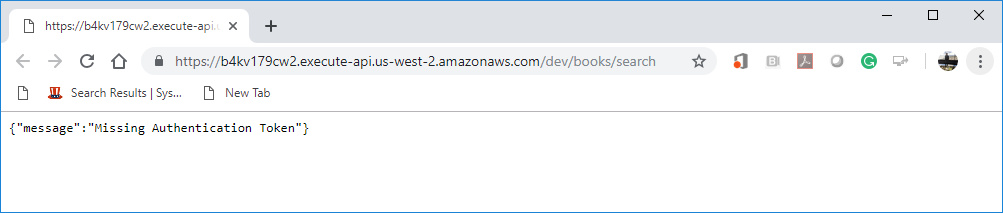
1. Open the terminal in the VSCode and run the ‘sls deploy’ command.
2. Copy the get-index endpoint and open it in the browser to test.  
   Expected output:   
   **dev/**

****This is because we used the IAM role.

**dev/books**

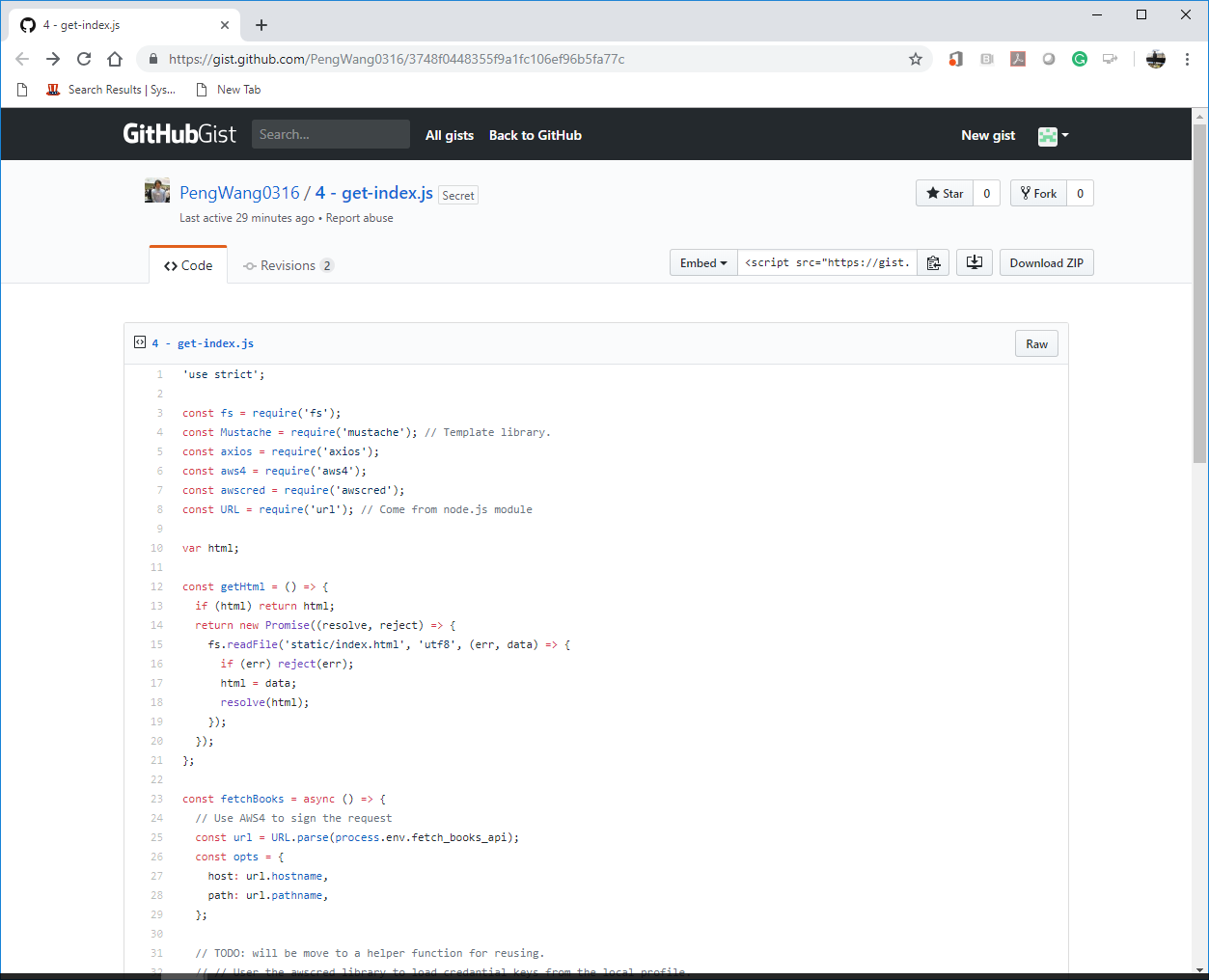
****  
This is because we protected the API.

**dev/books /search**



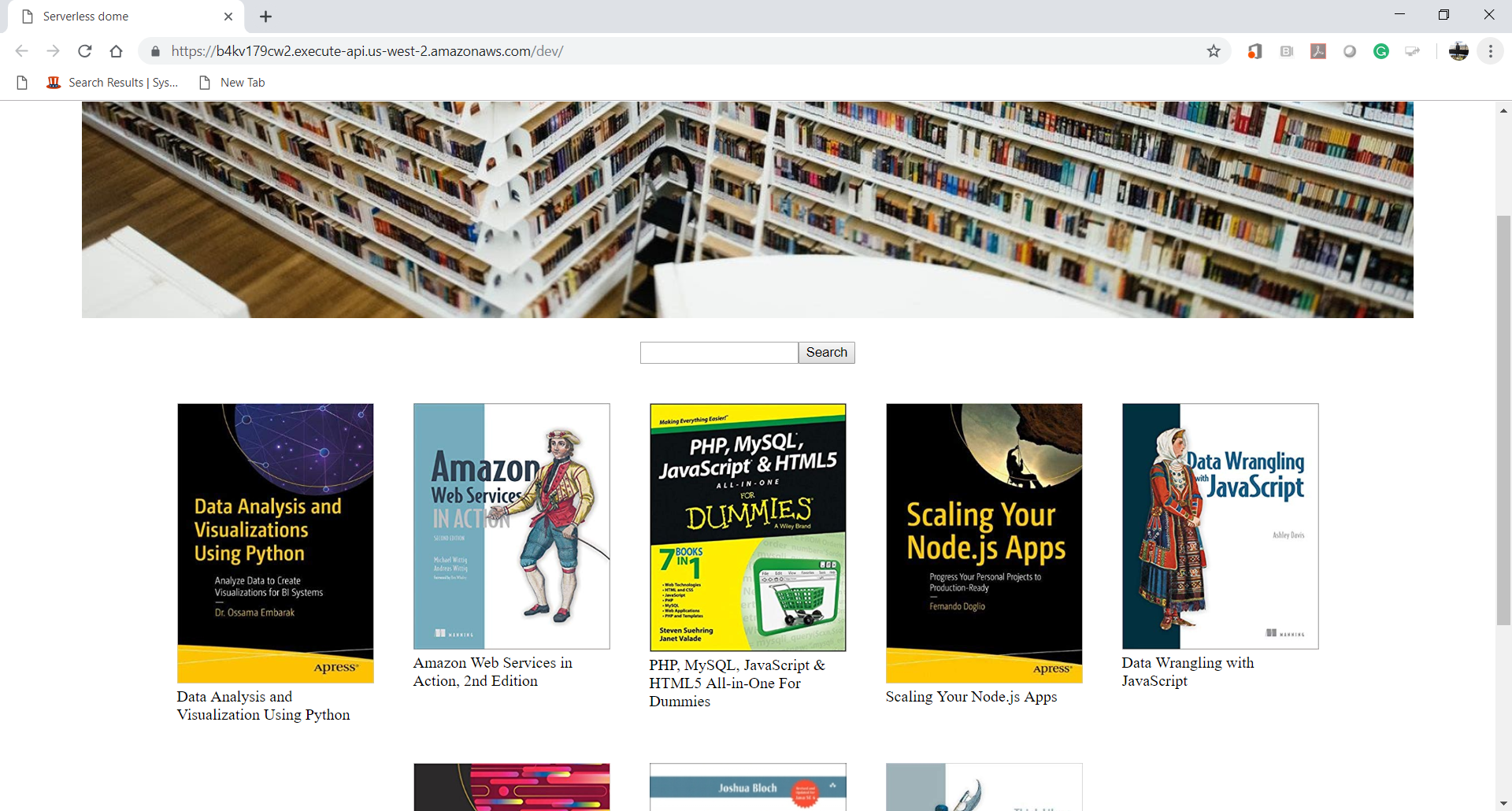
This is because we protected the API.

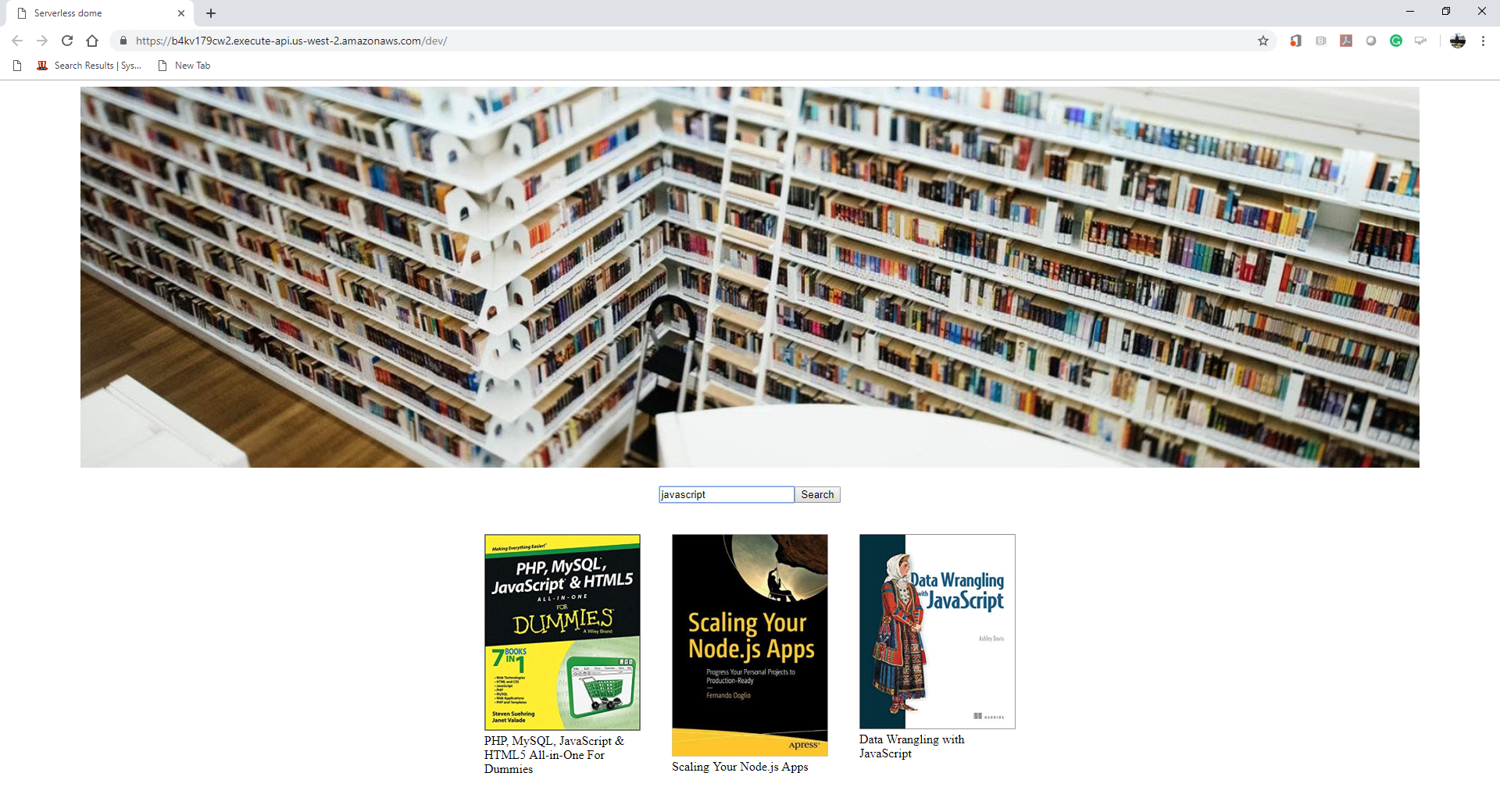
1. Open the “**get-index.js**” file under the functions folder and replace the content with <http://bit.ly/2rx9lDu>. This code will give the permission to securely access the API.

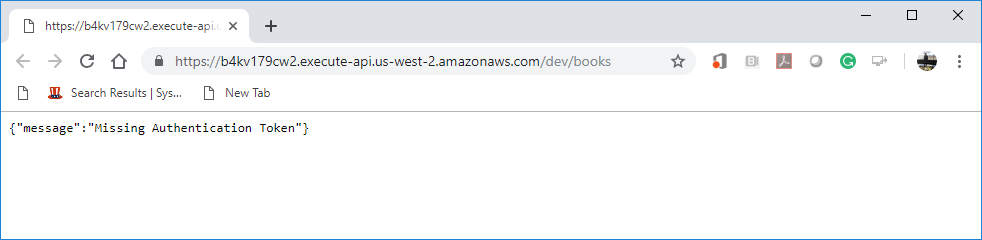


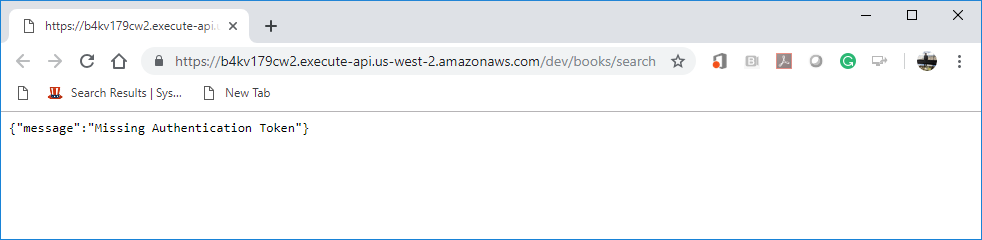
1. Open the terminal and type “sls deploy”
2. Copy the get-index endpoint and open it in the browser to test

Expected output:

**dev**   




**dev/books**  


**dev/books/search**

**Questions:**

1. Is protecting API necessary?   
   (Answer)  
     
   Yes. With current increasing of API usage, there are adversaries focusing on this type of attack.
2. If so, what consequences may occur when we fail to do so?   
   (Answer)

Failing to protect our API can cause data breaches. Since API means automation, hackers can use tools like automated bots to crawl data. It is a challenge to detect “bad bots” since high volume API calls can also be from “good bots”.

Here is the list of potential API security threats:

1. Injection Attacks (We practiced those in our LABs at this course with PHP/SQL injections)
2. DoS Attacks
3. Broken Authentication
4. Sensitive Data Exposure
5. Broken Access Control
6. Parameter Tampering
7. MITM (Man-In-The-Middle-Attack)

*Reference:*

https://blog.restcase.com/top-7-rest-api-security-threats/