Serverless Development 101

**HOP06 – Protecting Your API with AWS IAM**

1/10/2019 Developed by Kevin Wang

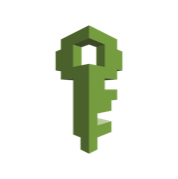
1/10/2019 Checked by Clark Jason Ngo

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

1/17/2019 Revised by Sam Chung

4/27/2020 Updated by Apiwat Chuaphan

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

**IAM**

Every AWS Lambda function needs permission to interact with other AWS infrastructure resources within your account. These permissions are set via an AWS IAM Role which the Serverless Framework automatically creates for each Serverless Service and is shared by all of your Functions.

More info can be found [here](https://aws.amazon.com/premiumsupport/knowledge-center/iam-authentication-api-gateway/)

**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

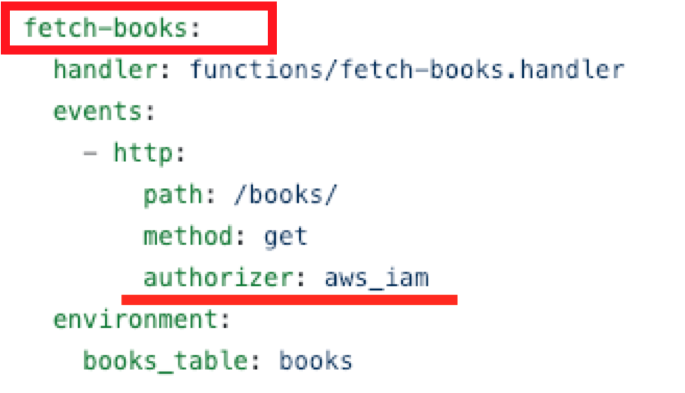
A screenshot of a cell phone

Description automatically generated

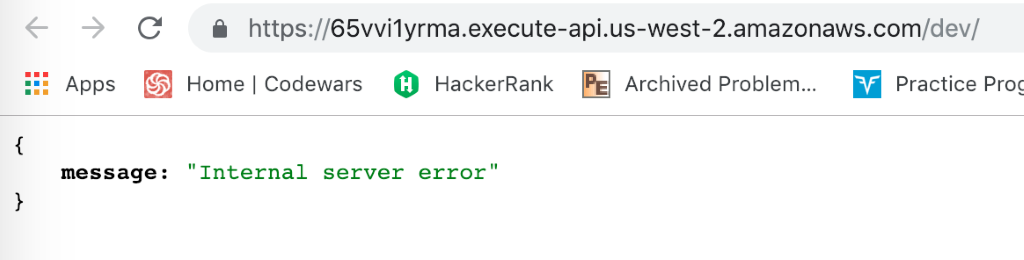
**Note:** we add statements in **provider.iamRoleStatements** so that it will add specific rights to this service-wide Role and will be merged into the generated policy.

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

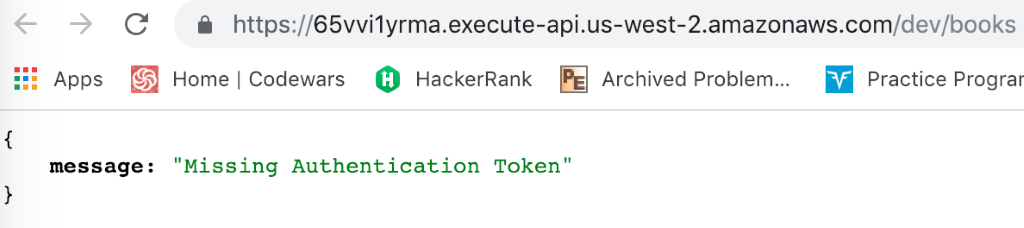
Note: add the following code under **fetch-books:** in “**serverless.yml”**

****

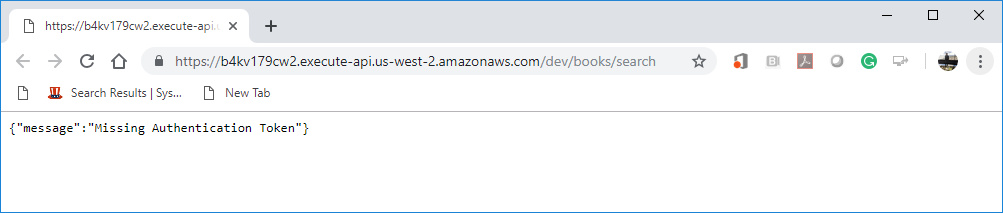
1. Open the terminal in the VSCode and run the ‘**sls deploy**’ command.
2. Copy the get-index endpoint and open it in the browser to test.  
   Expected output:   
   **…/dev/**

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

**…/dev/books**

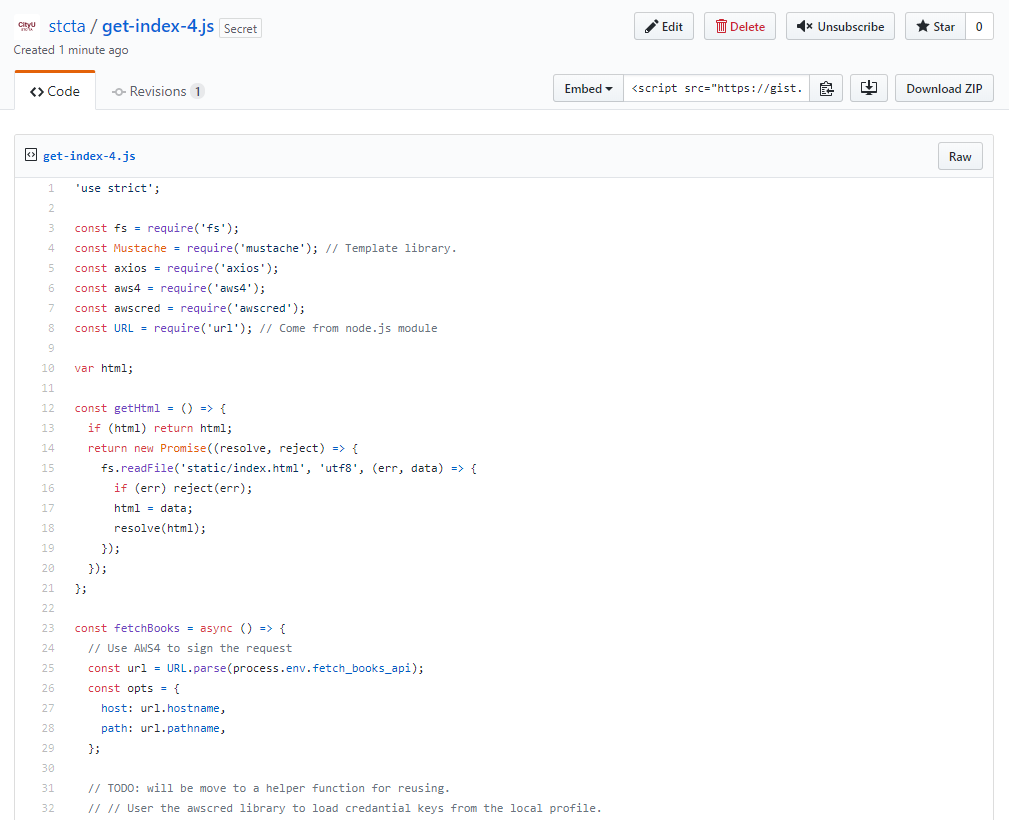
****  
This is because we protected the API.

**…/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 <https://bit.ly/2yP184N>. This code will give the permission to securely access the API.

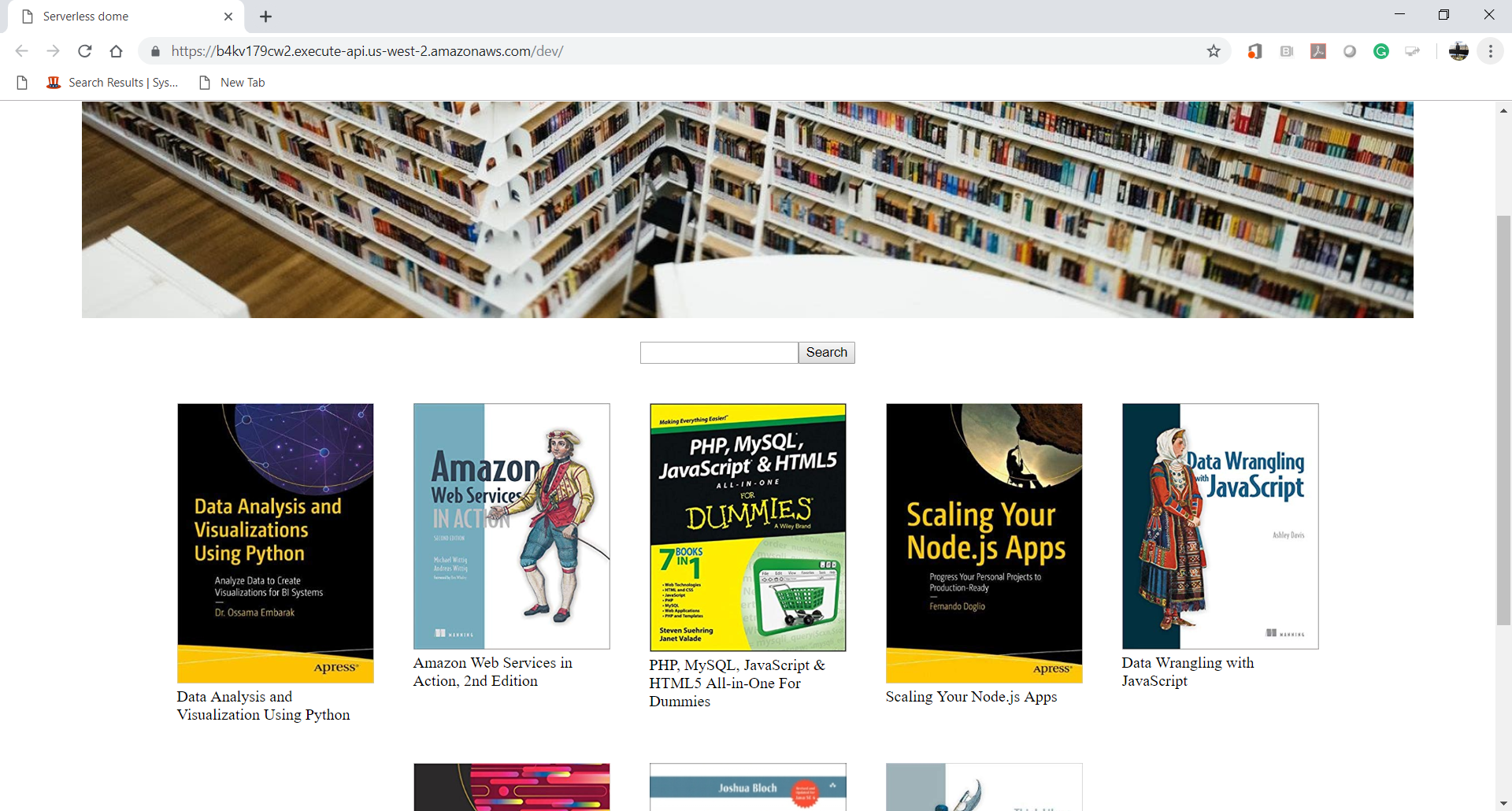


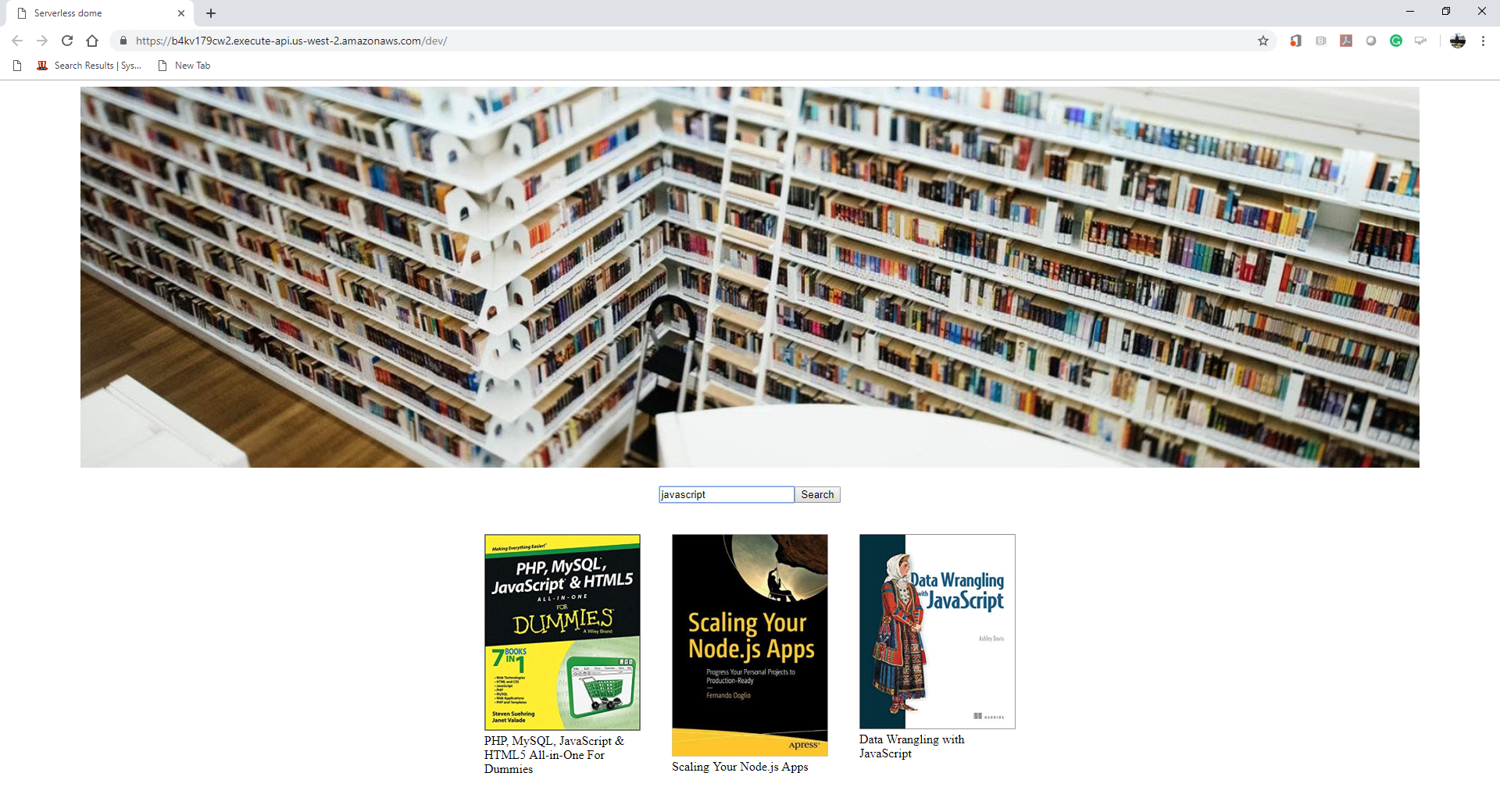
**Note:** we add

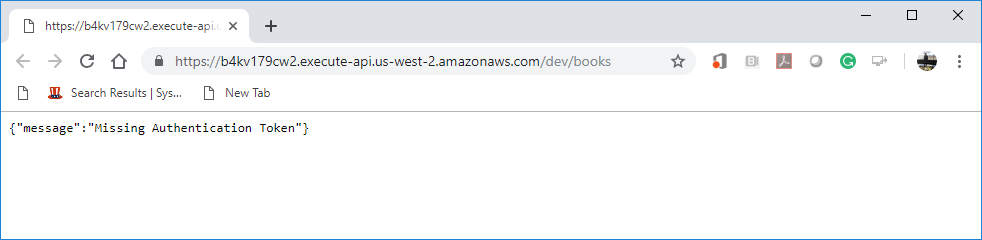
* **aws4** which is a small utility to sign vanilla Node.js http(s) request options using Amazon's AWS Signature Version 4.
* **Awscred** is a small standalone library in NodeJS to resolve AWS credentials and region details which helps you keep up to date your access token for AWS connection.

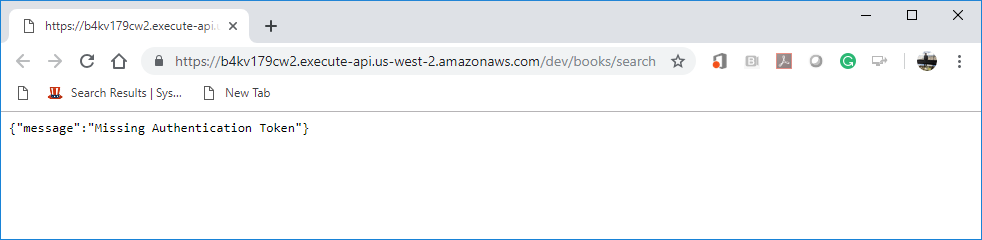
1. Open the terminal and type “**sls deploy**”
2. Copy the get-index endpoint and open it in the browser to test

Expected output:

**…/dev/**   




**…/dev/books**  


 …/**dev/books/search**

**Push your work to GitHub**

Open the terminal from the VSCode by hit the control + ~ key and type the following command:

Run the following commands to push your work to the GitHub repository:

>>> git add .

>>> git commit -m “Submission for Module 6”

>>> git push origin YOUR\_BRANCH\_NAME

**Note**: you should change the YOUR\_BRANCH\_NAME to your own branch name. It should be firstname-lastname (e.g. maria-gracia).

If you cannot remember, run the command “git status” to check