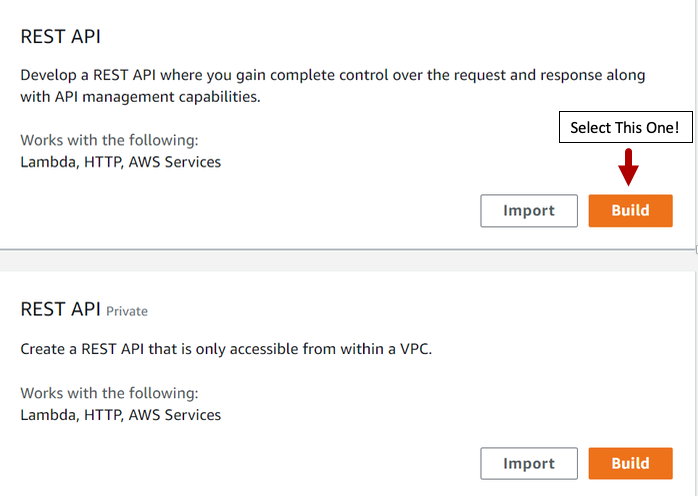
# CS 470 Project One Guide

## APIs to Lambda

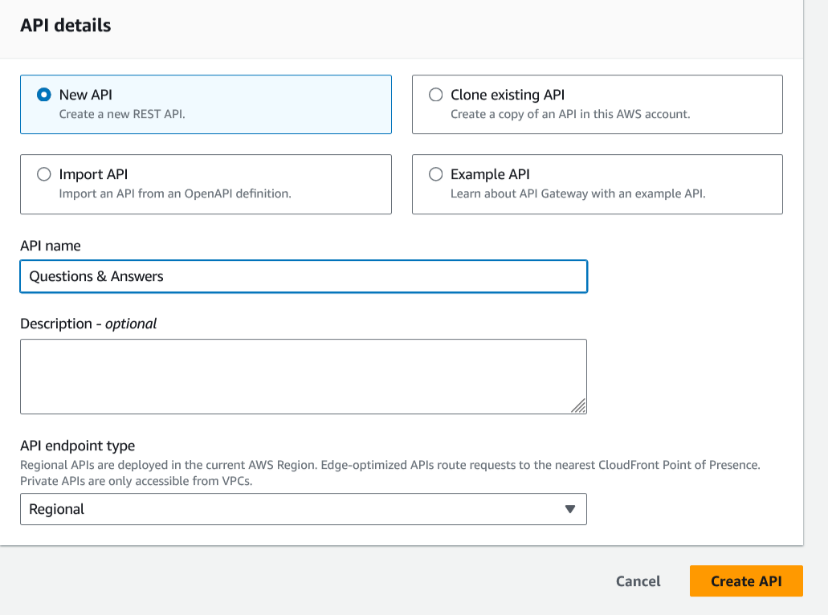
* Be careful with the case of the API resources. REST is a case-sensitive protocol, and the wrong case will prevent your application from working correctly.
* After every API mapping to a Lambda, you will be asked to approve the addition to the permission of the Lambda. Click **OK**. This permission allows each API you map to a Lambda access to that specific Lambda.

## Part One: The Questions API

1. Navigate to the API Gateway in the AWS console. See Module Four Assignment Two for a refresher.
2. Click the orange **Create API** button, then click **Build a REST API**.
   1. Do not click the REST API marked **private**.



1. Your settings should be **REST**, **New API**, and an **API name** “Questions & Answers”.



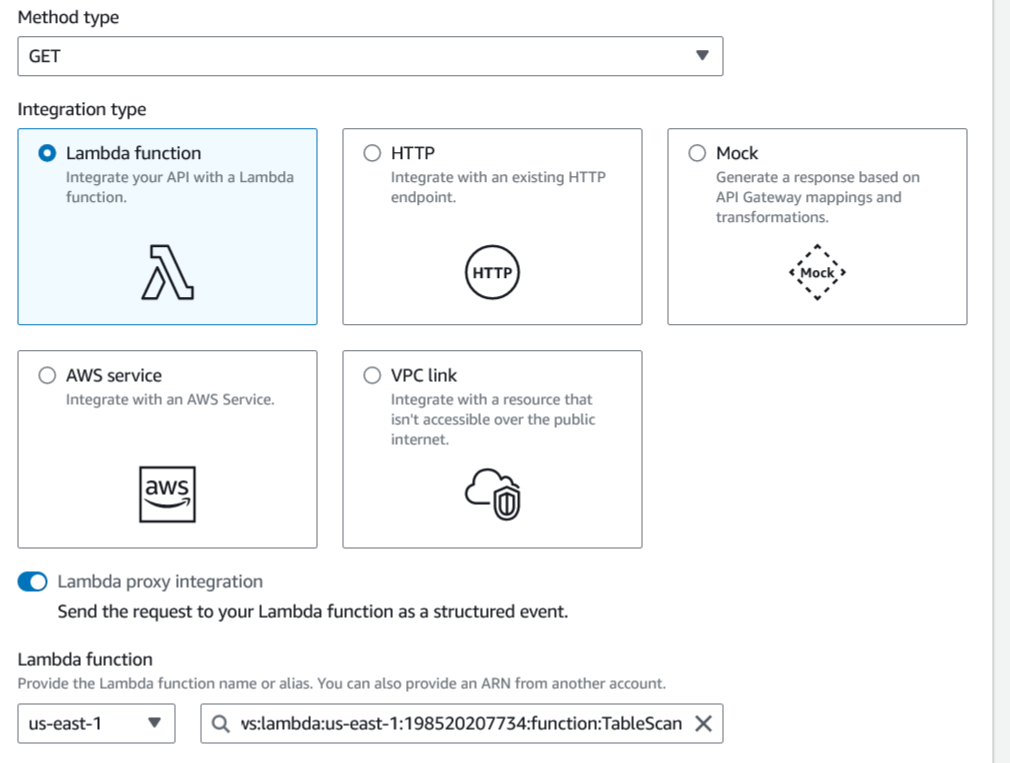
1. Click the **Create API** button.

### Questions Resource

1. Click **Create Resource** on the left side under the **Resources** page title.
2. Create a resource named “Questions”, and make certain the resource path has a capitalized Q.
3. Leave the check boxes unchecked and click **Create Resource**.
4. With **/Questions** selected, click **Create Method** on thebottom-right side of the screen.

Resource details. In the Questions section, resource details uses the path /Questions with the response id e7pjxf.
There are no methods.

1. Select the GET method in the **Method** type drop-down box.
2. Make certain **Lambda Function** and **Lambda Proxy Integration** are selected. Use region **us-east-1** with the Lambda function **TableScan**.



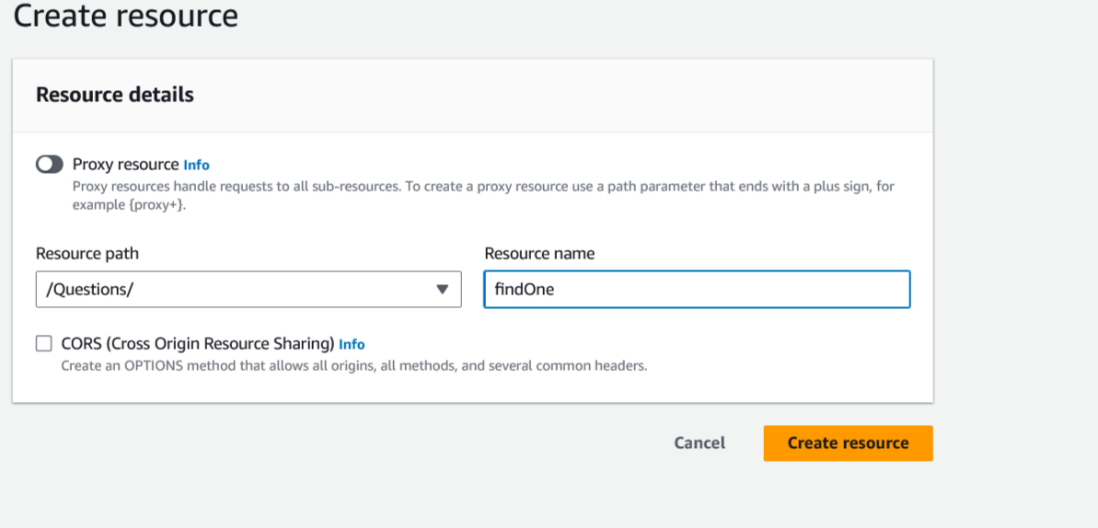
1. Click the **Create method** button.
2. Add the POST method to **Questions** with the following setup:
   1. Integration type: Lambda Function
   2. Use Lambda Proxy Integration: Checked
   3. Lambda region: us-east-1
   4. Lambda Function: UpsertQuestion
3. Add the PUT method to **Questions** with the following setup:
   1. Integration type: Lambda Function
   2. Use Lambda Proxy Integration: Checked
   3. Lambda region: us-east-1
   4. Lambda Function: UpsertQuestion

### The {id} Sub-Resource

1. Select the **/Questions** resource and click **Create Resource** above it.
2. Create a resource named “{id}”. Pay attention to the resource path. It should be **/Questions/{id}**. Leave CORS unchecked.
3. Click the **Create method** button with **/{id}** selected.
4. Add a GET method to **/Questions/{id}** with the following setup:
   1. Integration type: Lambda Function
   2. Use Lambda Proxy Integration: Checked
   3. Lambda region: us-east-1
   4. Lambda Function: GetSingleRecord
5. Add a DELETEmethod to **/Questions/{id}** with the following setup:
   1. Integration type: Lambda Function
   2. Use Lambda Proxy Integration: Checked
   3. Lambda region: us-east-1
   4. Lambda Function: DeleteRecord
6. Add a PUTmethod to **/Questions/{id}** with the following setup:
7. Integration type: Lambda Function
8. Use Lambda Proxy Integration: Checked
9. Lambda region: us-east-1
10. Lambda Function: UpsertQuestion

### The findOne Sub-Resource

1. With **Questions** selected, create a new resource named “findOne” under **Questions**. Pay attention to the casing of the resource name.



1. Add a GET method to **/Questions/findOne** with the following setup:
2. Integration type: Lambda Function
3. Use Lambda Proxy Integration: Checked
4. Lambda region: us-east-1
5. Lambda Function: FindOneQuestion

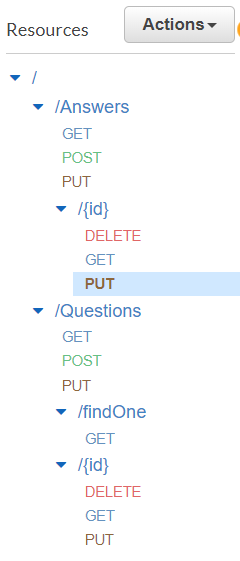
## Part Two: The Answers API

### The Answers Resource

1. Create a new top-level resource named “Answers”. Select the forward slash (/). Make certain the resource path is **/Answers**. Then click **Create resource.**
2. Add a GET method to **/Answers** with the following setup:
3. Integration type: Lambda Function
4. Use Lambda Proxy Integration: Checked
5. Lambda region: us-east-1
6. Lambda Function: TableScan
7. Add a POSTmethod to **/Answers** with the following setup:
8. Integration type: Lambda Function
9. Use Lambda Proxy Integration: Checked
10. Lambda region: us-east-1
11. Lambda Function: UpsertAnswer
12. Add a PUTmethod to **/Answers** with the following setup:
13. Integration type: Lambda Function
14. Use Lambda Proxy Integration: Checked
15. Lambda region: us-east-1
16. Lambda Function: UpsertAnswer

### The {id} Sub-Resource

1. Create an {id} resource under **Answers**. Make certain the Resource Path is **/Answers/{id}**.
2. Add a GETmethod to **/Answers/{id}** with the following setup:
3. Integration type: Lambda Function
4. Use Lambda Proxy Integration: Checked
5. Lambda region: us-east-1
6. Lambda Function: GetSingleRecord
7. Add a DELETEmethod to **/Answers/{id}** with the following setup:
8. Integration type: Lambda Function
9. Use Lambda Proxy Integration: Checked
10. Lambda region: us-east-1
11. Lambda Function: DeleteRecord
12. Add a PUTmethod to **/Answers/{id}** with the following setup:
    1. Integration type: Lambda Function
    2. Use Lambda Proxy Integration: Checked
    3. Lambda region: us-east-1
    4. Lambda Function: UpsertAnswer
13. You should now have an API Resources tree that looks like this:

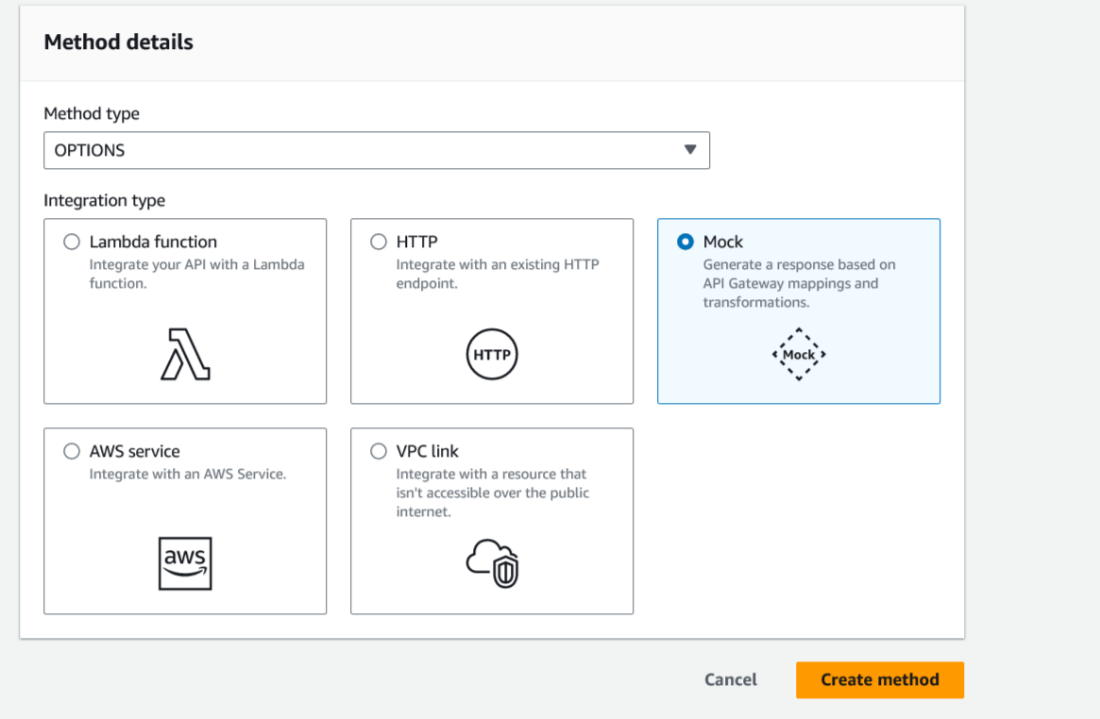


## Supporting CORS

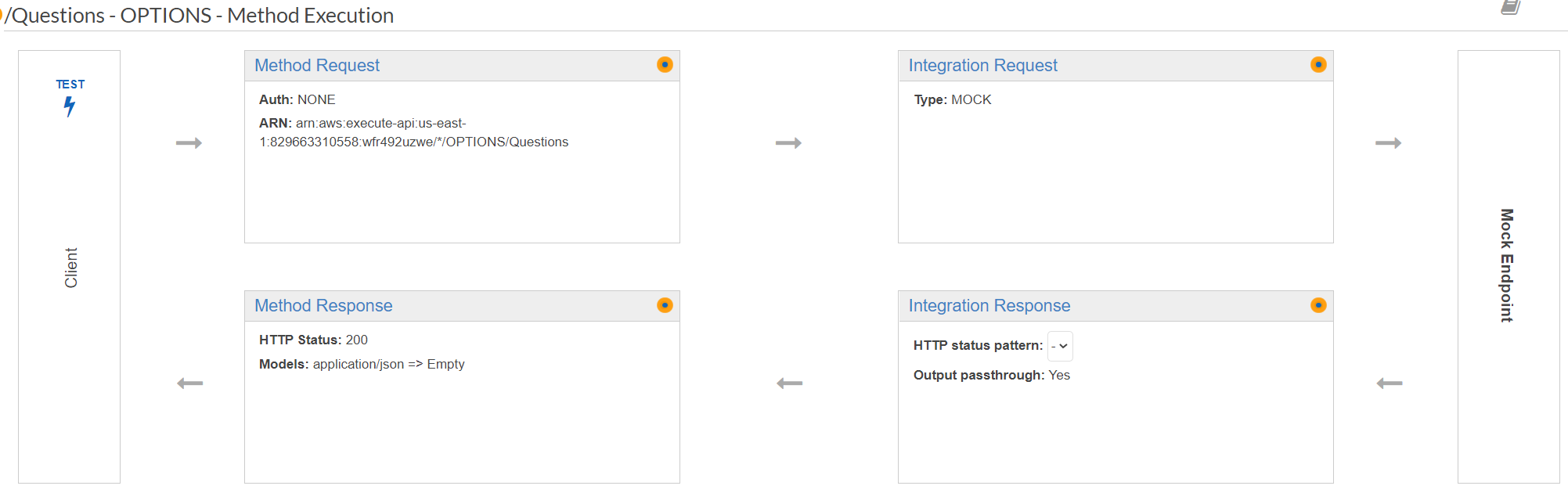
Cross-origin resource sharing (CORS) is a mechanism that uses HTTP headers to allow resource sharing across different domains. AWS Gateway has an action to enable CORS, but you will not be using it. This action is only partially successful, so you will do this manually as defined on the [CORS on AWS API Gateway](https://enable-cors.org/server_awsapigateway.html) website. The requirements are to have an OPTIONS method for each resource with specific headers returned and for some specific headers to be returned on each REST method called.

### Adding CORS to Questions Resource

1. Select the **/Questions** resource and add an **OPTIONS** method.
2. Select **Mock** as the integration type and click the **Create method** button.



1. Click the **Method Response** link.



1. You must now add some headers. Click **Edit** next to **Response 200** and then click **Add Header**.
2. Enter “X-Requested-With” in the box. Be careful to use the correct case and do not include spaces.
3. Repeat the process to add the following Response headers:
   1. Access-Control-Allow-Headers
   2. Access-Control-Allow-Origin
   3. Access-Control-Allow-Methods
   4. Access-Control-Allow-Credentials

When complete, the Response Headers section will look like this:

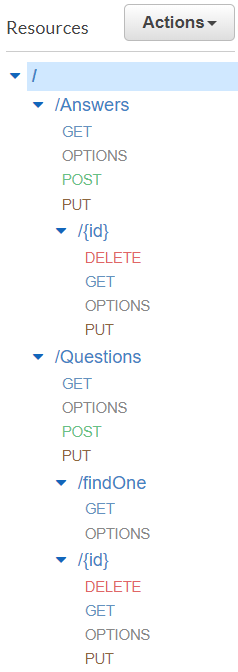
Response 200 window. Response headers is the header. The Name options are: 
Access-Control-Allow-Credentials
Access-Control-Allow-Headers
Access-Control-Allow-Methods
Access-Control-Allow-Origin
X-Requested-With

1. Click the **Integration Response** link.
2. Open the 200 method response status by clicking the **Edit** button next to **Default - Response**.
3. Scroll down to the **Header Mappings** section.
4. For X-Requested-With, enter **‘\*’**, with the quotes.
5. For Access-Control-Allow-Headers, enter **'Content-Type,X-Amz-Date,Authorization,X-Api-Key,x-requested-with'**, with the quotes.
6. For Access-Control-Allow-Origin, enter **'\*'**, with the quotes.
7. For Access-Control-Allow-Methods, enter **'OPTIONS,\*'**, with the quotes.
8. For Access-Control-Allow-Credentials, enter **‘true’**, with the quotes.
9. When done, the **Header Mappings** page should look like this:

Header mappings page. The method response status code is 200. The content handling is Passthrough. The default mapping is True. There are five header mappings. Their names are 
"method.response.header.Access-Control-Allow-Credentials" with mapping value 'true'; "
method.response.header.Access-Control-Allow-headers" with mapping value 'Content-Type-Amz-Date,Authorization,X-Api-Key,x-requested'; "method.response.header.Access-Control-Allow-Methods" with the mapping value 'OPTIONS,*'; "method.response.header.Access-Control-Allow-Origin"; and "method.response.header.X-Requested-With".

### Adding CORS to the Remaining Resources

1. Repeat this process for each resource and sub-resource:
   1. /Answers
   2. /Answers/{id}
   3. /Questions/findOne
   4. /Questions/{id}
2. When complete, your Resources tree will look like this:



### CORS Headers for the Other Methods

The other methods must return CORS headers as well. The good news is there is nothing for you to do now. Because you configured the APIs as proxies to the Lambdas, you can handle this in the Lambda code, which you have already done. Near the start of each Lambda you created in Module Five is the following code:

const responseHeaders = (headers) => {

const origin = headers.origin || headers.Origin;

return {

// HTTP headers to pass back to the client

"Content-Type": "application/json",

// the next headers support CORS

"X-Requested-With": "\*",

"Access-Control-Allow-Headers":

"Content-Type,X-Amz-Date,Authorization,X-Api-Key,x-requested-with",

"Access-Control-Allow-Origin": origin,

"Access-Control-Allow-Methods": "OPTIONS,\*",

Vary: "Origin", // for proxies

// the "has-cors" library used by the Angular application wants this set

"Access-Control-Allow-Credentials": "true",

};

};

Toward the bottom of the Lambda is the code where the response to the client is created. You had this line of code:

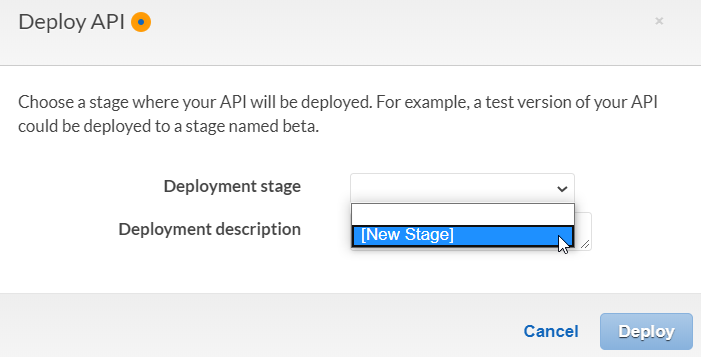
// HTTP headers to pass back to the client

headers: responseHeaders,

Between these two blocks of code and the APIs being configured as proxies, you are handling the CORS requirements for the other methods.

## Deployment

1. Deploy the API using the “api” deployment name by clicking the **Deploy API** button in the top right of the API resource page.
2. Create a new stage from the **Stage** drop-down menu.



1. Enter “api” as the **Stage name** and click **Deploy**.

**Note:** A common error at this stage is **No integration defined for method**. If you see this error, you forgot to click the **Save** button when mapping API methods to your Lambdas. Check each API method and fix it as needed. Then try again.

1. Record the **Invoke URL** displayed after deployment. It will look like this:

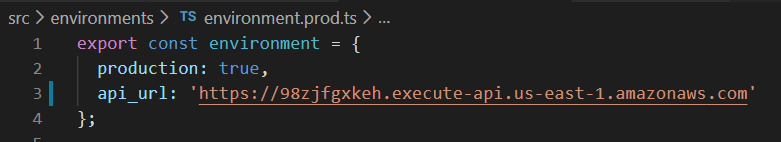
api Stage Editor screen. The Invoke URL is displayed on the right side. 


## Modifying the Angular Application

1. Modify **main.ts**. Add the line **LoopBackConfig.filterOnUrl();** to the end of the file. The resulting change will look like this:

A window displays the following code: 
import 'hammerjs';
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { LoopBackConfig } from '../sdk';
 
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';
 
if (environment.production) {
  enableProdMode();
}
 
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
 
LoopBackConfig.setBaseURL(environment.api_url);
 
LoopBackConfig.Url();

1. Modify **environment.prod.ts**.Change the line **api\_url: 'https://q-a-example-loopback-api.herokuapp.com** to **api\_url: 'https://{deployed-url}.us-east-1.amazonaws.com’**

****

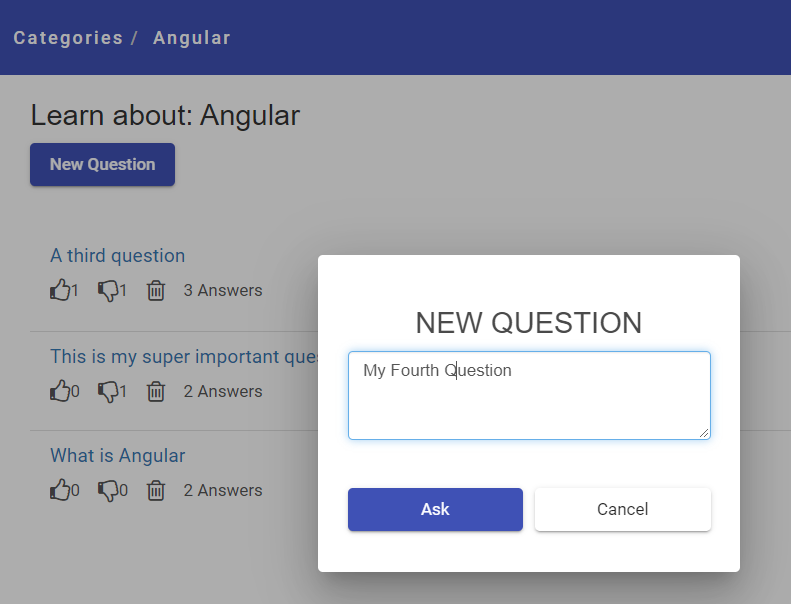
**Remember:** Your deployed URL is the one you created in the Deployment section of this guide. Do not include the ‘api’ or a trailing forward slash. The Angular application adds those.

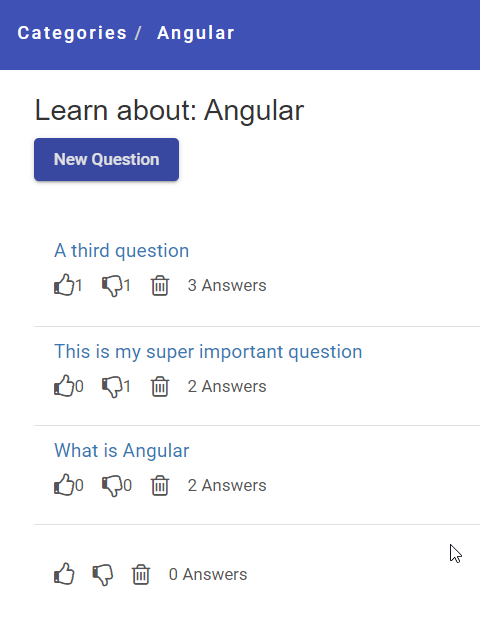
1. Build and deploy the Angular application. See the Module Three Assignment Guide for a refresher.
   * 1. **Optional:** You can safely delete the existing elements in your S3 bucket before copying new files over if you choose.

## You Are Done!

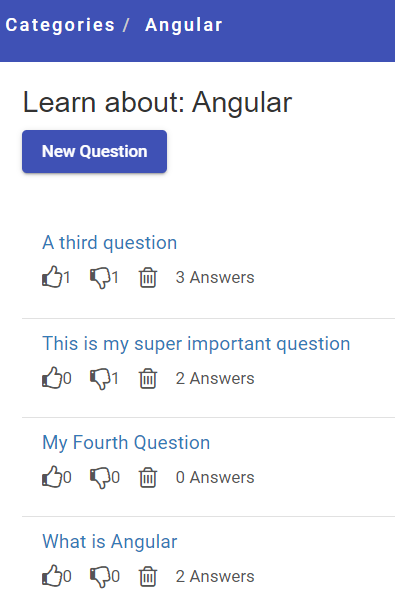
Try testing your application. Consider the following notes:

* You shifted the application from using the LoopBack SDK on the client that was tied to the LoopBack REST API on the server.
* In completing the transfer, you would change the client side to not depend on the LoopBack SDK and its assumption of a LoopBack REST server.
* Why does this matter? When you add a new Question or Answer, the added text is not always seen right away. The call is successful. A new entry is visible, but the title is not.





* If you click on **Categories** and go back into **Angular**, the new content will show up.



Changing this behavior may be beyond the scope of this class.

Time to celebrate a complete project!