Create a multi-tier solution by using Azure services and API Management

Microsoft Azure user interface

Given the dynamic nature of Microsoft cloud tools, you might experience Azure UI changes that occur after the development of this training content. As a result, the lab instructions and lab steps might not align correctly.

Instructions

Before you start

Sign in to the lab environment

> **Note**: Your instructor will provide instructions to connect to the virtual lab environment.

Review the installed applications

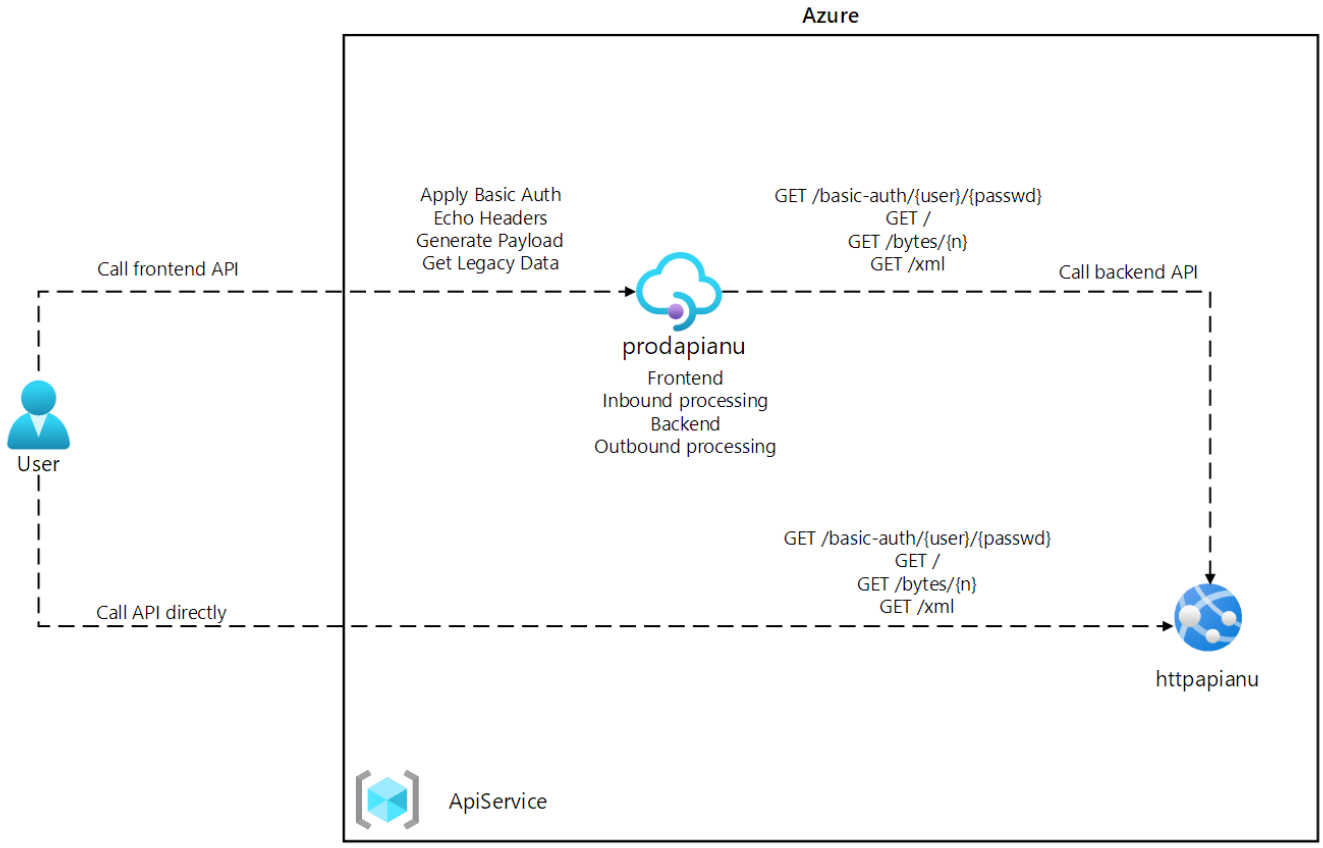
Find the taskbar on your Windows 11 desktop. The taskbar contains the icons for the applications that you'll use in this lab:

- Microsoft Edge

Lab Scenario

In this proof of concept, you will create a containerized application to host a web app on Azure, as the source of information for the API. You will then build an API proxy using the Azure API Management capabilities to expose and test your APIs. Developers can query the APIs to test the service and validate its applicability.

Architecture diagram



Exercise 1: Create an Azure App Service resource by using a Docker container image

Task 1: Open the Azure portal

1. On the taskbar, select the **Microsoft Edge** icon.
2. In the browser window that opens, browse to the Azure portal at `https://portal.azure.com`, and then sign in with the account you'll be using for this lab.

> **Note**: If this is your first time signing in to the Azure portal, you'll be offered a tour of the portal. Select **Get Started** to skip the tour and begin using the portal.

Task 2: Create a web app by using Azure App Service resource by using an httpbin container image

1. In the Azure portal, use the **Search resources, services, and docs** text box to search for **App Services** and, in the list of results, select **App Services**.
2. On the **App Services** blade, select **+ Create**.
3. On the **Create Web App** blade, on the **Basics** tab, perform the following actions:

Setting	Action

Setting	Action
Subscription drop-down list	Retain the default value
Resource group section	Select **Create new**, enter **ApiService**, and then select **OK**
Name text box	Enter **httpapi** *[yourname]*
Publish section	Select **Container**
Operating System section	Select **Linux**
Region drop-down list	Select any Azure region in which you can deploy an Azure web app
Linux Plan section	Select **Create new**, enter the value **ApiPlan** in the **Name** text box, and then select **OK**
Pricing plan section	Select **Explore pricing plans**, on the **Select App Service Pricing Plan** page, select **Basic B1**, and then select **Select**

4. Select **Next: Database >**.
5. Select **Next: Container >**.
6. On the **Docker** tab, perform the following actions, and then select **Review + create**:

Setting	Action
Image Source drop-down list	Select **Docker Hub or other registries**
Options drop-down list	Select **Single Container**
Access Type drop-down list	Select **Public**
Image and tag text box	Enter **kennethreitz/httpbin:latest**

7. On the **Review + create** tab, review the options that you selected during the previous steps.
8. Select **Create** to create the web app by using your specified configuration.

> **Note**: Wait for the creation task to complete before you proceed with this lab.

Task 3: Test the httpbin web application

1. In the Azure portal, use the **Search resources, services, and docs** text box to search for **App Services** and, in the list of results, select **App Services**.
2. On the **App Services** blade, select the newly created web app.
3. On the blade displaying the newly created app properties, select **Browse**.

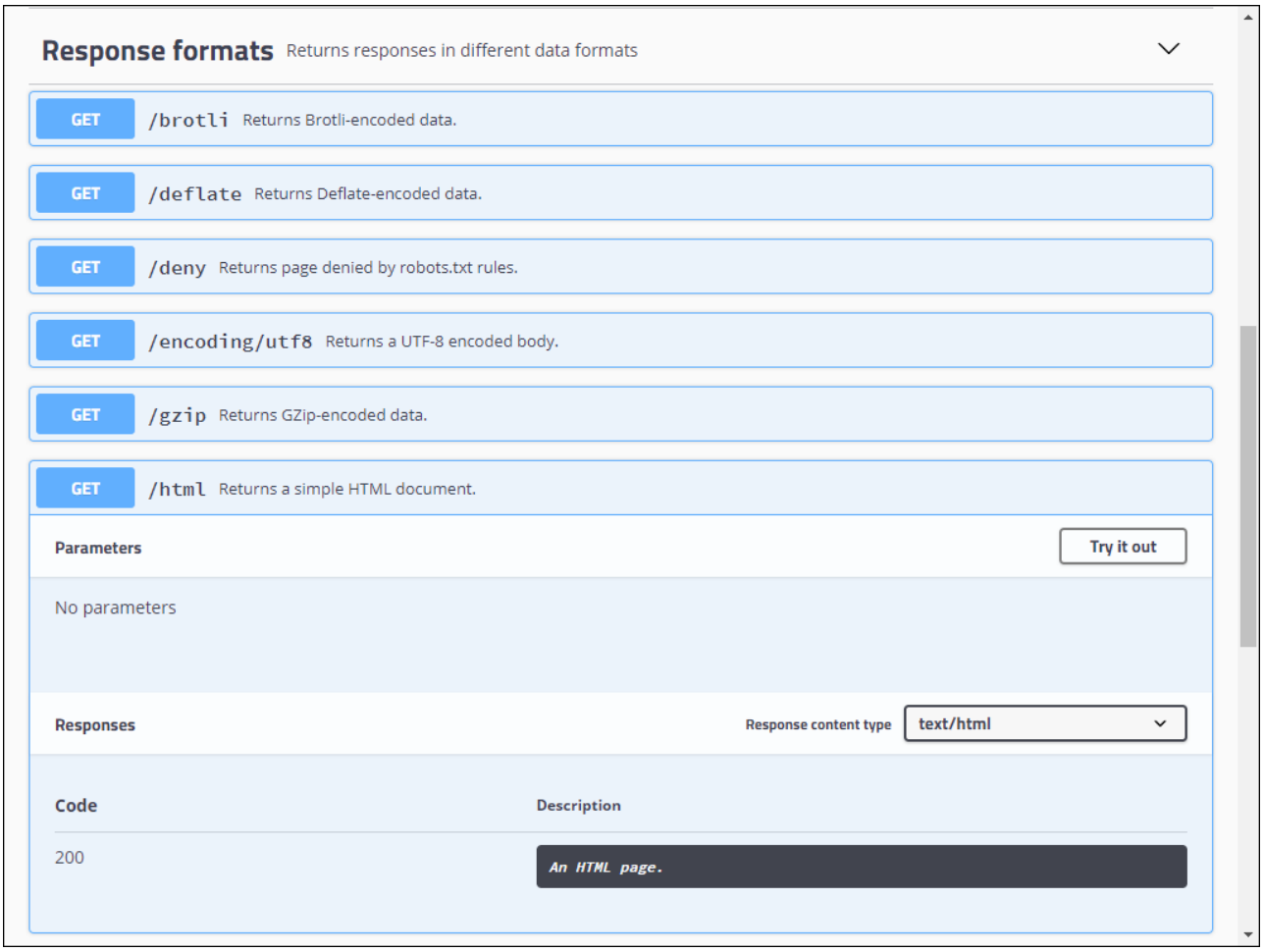
4. Within the web application, perform the following actions:

    a. Select **Response formats**.

    b. Select **GET /html**.

    c. Select **Try it out**.

The following screenshot displays the **Try it out** section of the web application.



    d. Select **Execute**.

    e. Review the value of the **Response body** and **Response headers** text boxes.

    f. Review the value of the **Request URL** text box.

5. Within the web application, perform the following actions:

    a. Select **Dynamic data**.

    b. Select **GET /bytes/{n}**.

    c. Select **Try it out**.

    d. In the **n** text box, enter **25**.

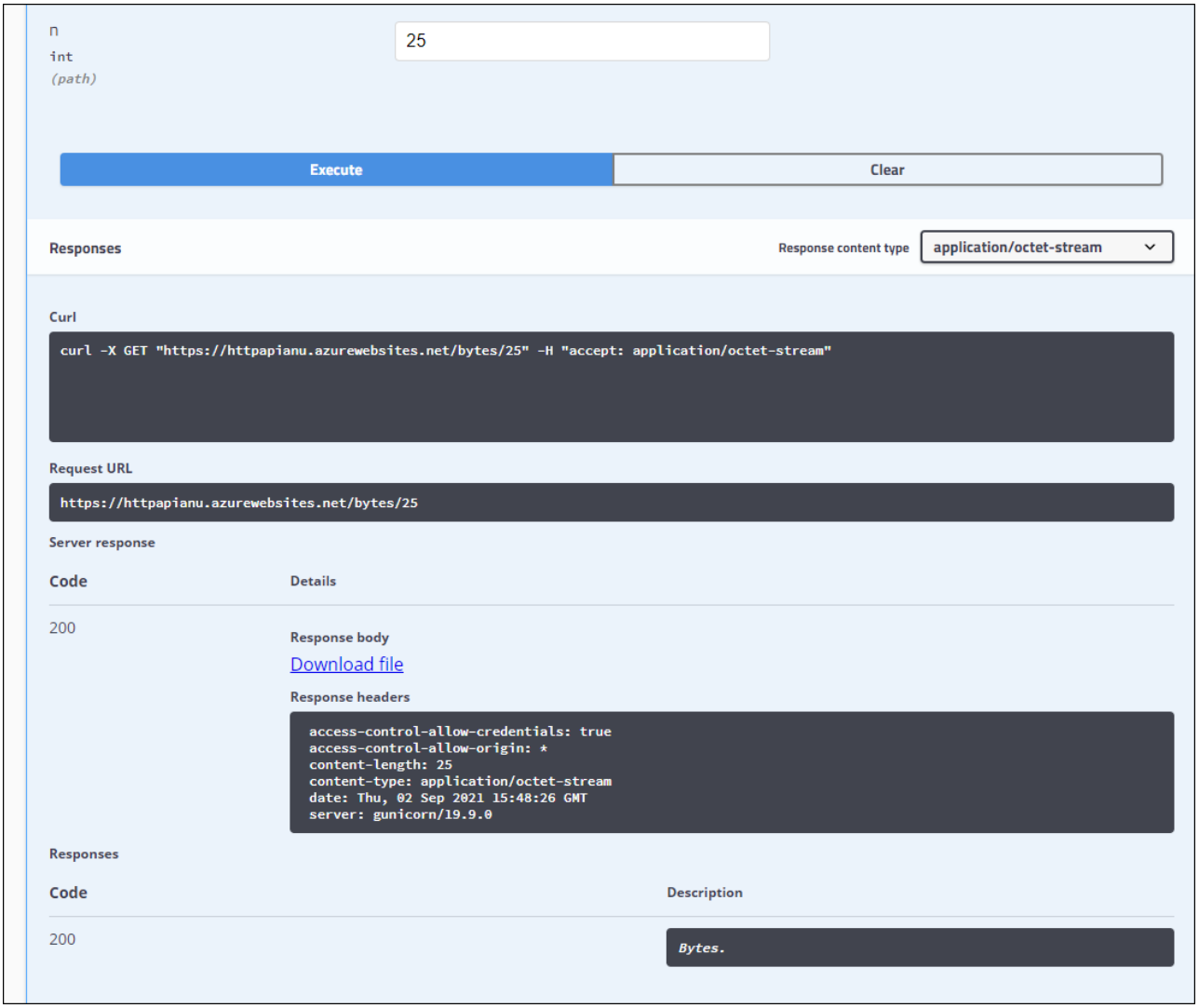
    e. Select **Execute**.

f. Review the value of the **Response body** and **Response headers** text boxes.

g. Select **Download file**, and after the file downloads, open it in Notepad, review its content, and then close it.

> **Note**: The file contains a sequence of randomly generated bytes.

The following screenshot displays the dynamic data section of the web application.



6. Within the web application, perform the following actions:

   a. Select **Status codes**.

   b. Select **GET /status/{codes}**.

   c. Select **Try it out**.

   d. In the **codes** text box, enter **404**.

   e. Select **Execute**.

   f. Review the **Server response** and note that it includes **Error: NOT FOUND** entry.

7. Close the browser window that displays the web application.

8. Switch back to the browser window that displays the **httpapi***[yourname]* web app.

9. On the **App Service Overview** blade, in the **Essentials**, record the value of the **Default domain** link. You'll use this value later in the lab to send requests to the corresponding API.

Review

In this exercise, you created a new Azure web app by using a container image sourced from Docker Hub.

Exercise 2: Build an API proxy tier by using Azure API Management

Task 1: Create an API Management resource

1. In the Azure portal, use the **Search resources, services, and docs** text box to search for **API Management services** and, in the list of results, select **API Management services**.

2. On the **API Management services** blade, select **+ Create**.

3. On the **Create API Management service** blade, perform the following actions, and then select **Review + create**:

Setting	Action
Subscription drop-down list	Retain the default value.
Resource group section	Select the **ApiService** group that you created earlier in the lab
Region list	Select the same region you chose in the previous exercise
Resource name text box	Enter **proapi** *[yourname]*
Organization name text box	Enter **Contoso**
Administrator email text box	Enter `admin@contoso.com`
Pricing tier drop-down list	**Consumption (99.95% SLA)**

The following screenshot displays the configured settings of **Create API Management** blade of the web application.

Home > API Management services >

Create API Management service

API Management service

Basics | Monitoring | Scale | Managed identity | Virtual network | Protocol settings | Tags | Review + install

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Azure Pass - Sponsorship

Resource group * : ApiService

Create new

Instance details

Region * : (US) East US

Resource name * : proapisam111

Organization name * : Contoso

Administrator email * : admin@contoso.com

Pricing tier

API Management pricing tiers vary in computing capacity per unit and the offered feature set - for example, support for virtual networks, multi-regional deployments, or self-hosted gateways. To accommodate more API requests, consider adding API Management service units instead. Learn more

You can create only 20 Consumption tier API Management services in an Azure subscription. Each Consumption tier service can manage up to 50 APIs. Learn more

Pricing tier : Consumption (99.95% SLA)

Review + create | < Previous | Next: Monitoring >

4. On the **Review + create** tab, review the option that you specified in the previous step, and then select **Create**.

> **Note**: Wait for the creation task to complete before you continue with this lab.

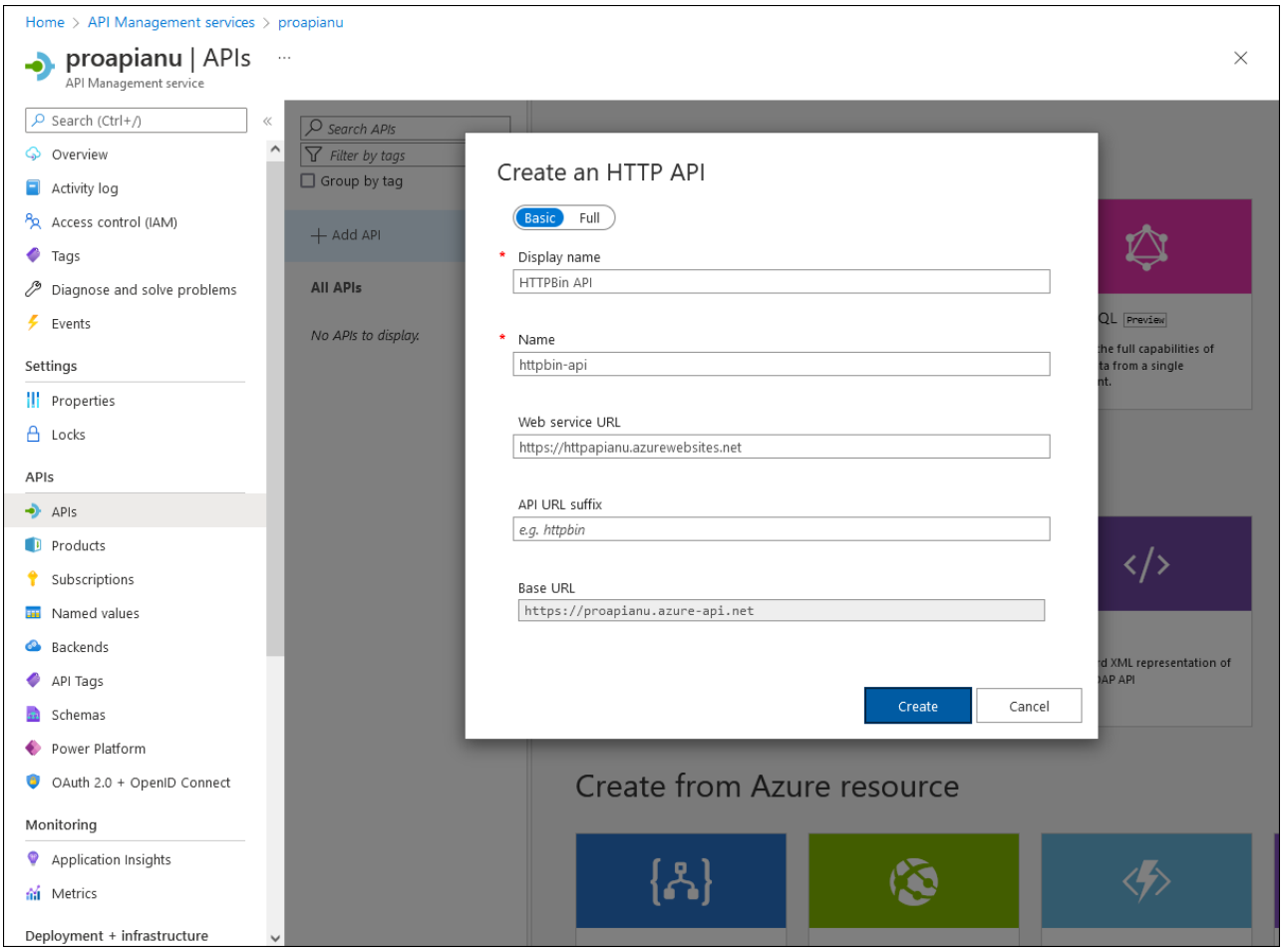
5. On the **Deployment Overview** blade, select **Go to resource**.

Task 2: Define a new API

1. On the **API Management Service** blade, in the **APIs** section, select **APIs**.
2. In the **Define a new API** section, select **HTTP**.
3. In the **Create an HTTP API** window, perform the following actions and select **Create**:

Setting	Action
Display name text box	Enter **HTTPBin API**
Name text box	Enter **httpbin-api**
Web service URL text box	Enter the URL for the web app that you copied earlier in this lab. **Note**: Make sure that the URL starts with the https:// prefix
API URL suffix text box	Leave it empty

The following screenshot displays the configured settings of **Create a blank API** window of the web application.



> **Note**: Wait for the new API to finish being created.

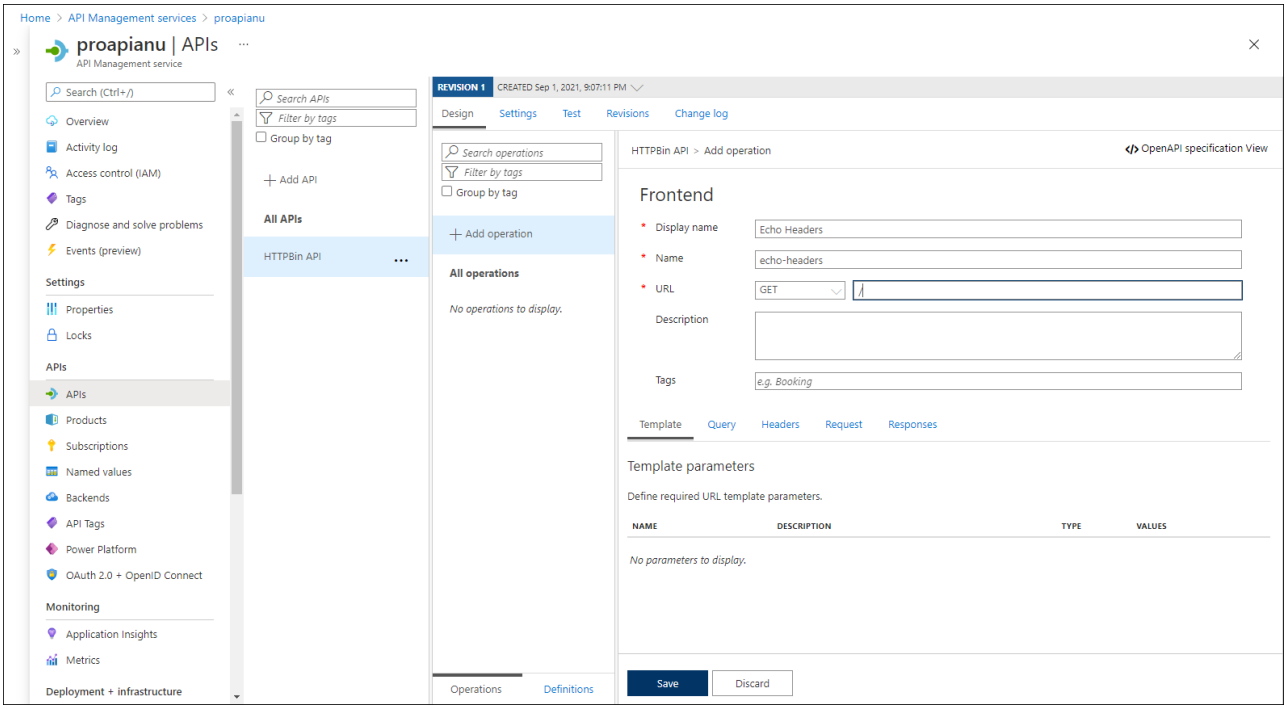
4. On the **Design** tab, select **+ Add operation**.

5. In the **Add operation** section, perform the following actions, and then select **Save**:

Setting	Action
Display name text box	Enter **Echo Headers**

Setting	Action
Name text box	Verify that its value is set to **echo-headers**
URL list	Select **GET**
URL text box	Enter **/**

The following screenshot displays the configured settings of the **Add operation** section.



6. Back on the **Design** tab, in the list of operations, select **Echo Headers**.

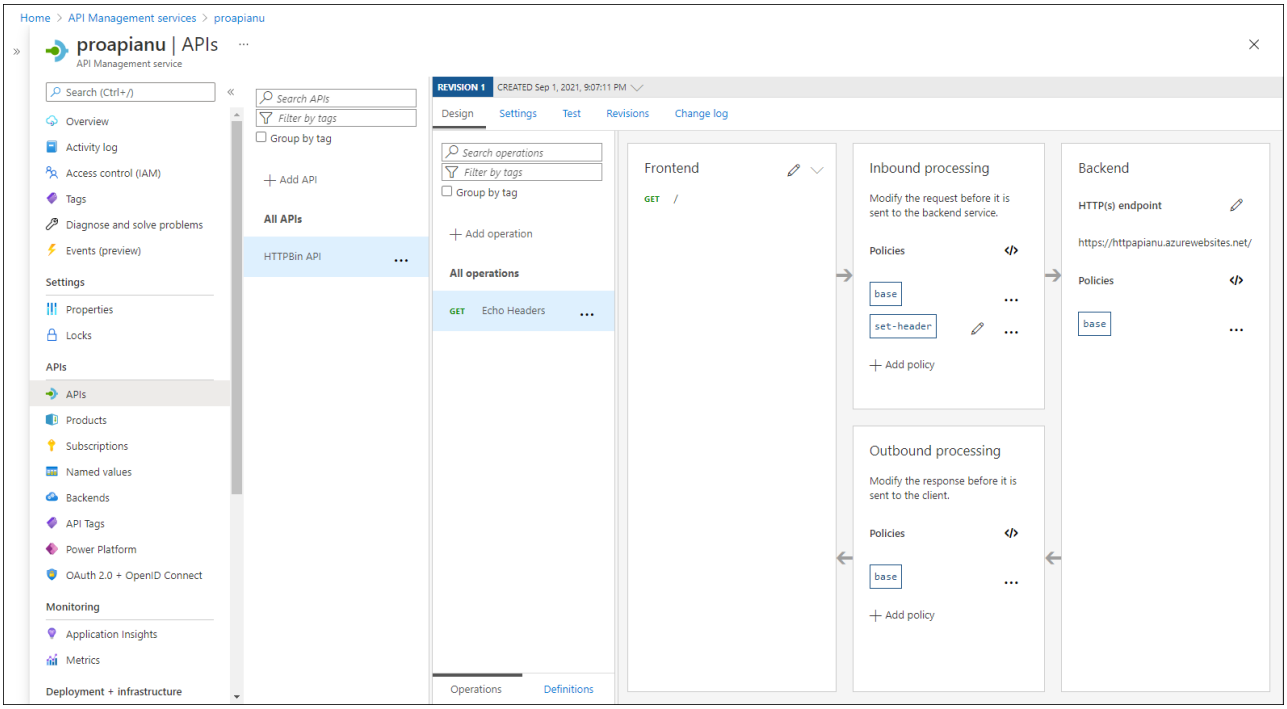
7. In the **Design** section, on the **Inbound processing** tile select **+ Add policy**.

8. In the **Add inbound policy** section, select the **Set headers** tile.

9. In the **Set Headers** section, perform the following actions, and then select **Save**:

Setting	Action
Name text box	Enter **source**
Value text box	Select the list, select **Add Value**, and then enter **azure-api-mgmt**
Action list	Select **append**

The following screenshot displays the configured settings of the **Design** section.



10. Back on the **Design** tab, in the list of operations, select **Echo Headers**.

11. In the **Design** section for **Echo Headers**, on the **Backend** tile, select the pencil icon.

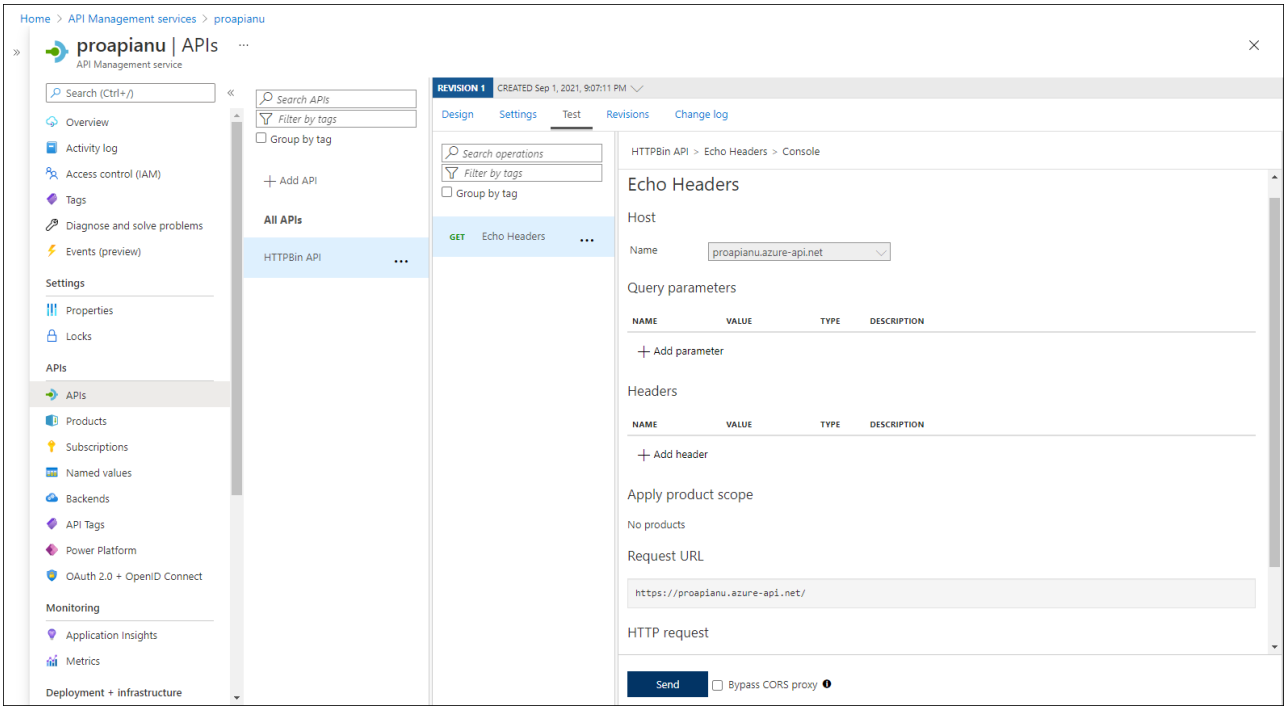
12. In the **Backend** section, perform the following actions, and then select **Save**:

Setting	Action
Service URL section	Select the **Override** check box
Service URL text box	Append the value **/headers** to its current value. **Note**: For example, if the current value is `https://httpapi[yourname].azurewebsites.net`, the new value will be `https://httpapi[yourname].azurewebsites.net/headers`

13. Back on the **Design** tab, in the list of operations, select **Echo Headers**, and then select the **Test** tab.

14. In the **Echo Headers** section, select **Send**.

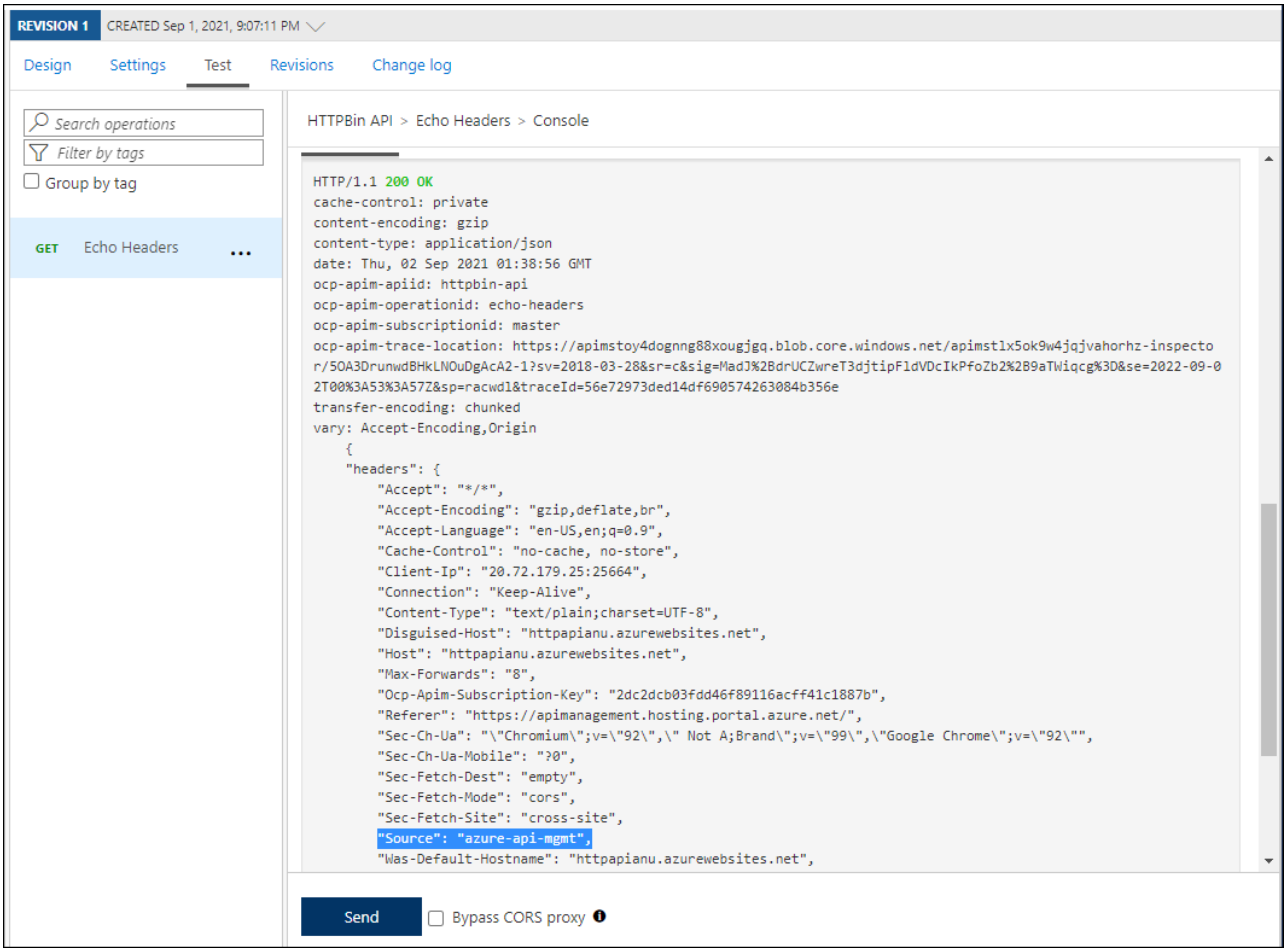
    The following screenshot displays the configured settings of the **Echo Headers** section.



15. Review the results of the API request.

> **Note**: Verify that there are many headers sent as part of your request that are echoed in the response. They should include the new **Source** header that you created as part of this task.

The following screenshot displays the response to the **Echo Headers** request.



16. Select the **Design** tab to return to the list of operations.

Task 3: Manipulate an API response

1. On the **Design** tab, select **+ Add operation**.

2. In the **Add operation** section, perform the following actions, and then select **Save**:

Setting	Action
Display name text box	Enter **Get Legacy Data**
Name text box	Verify that its value is set to **get-legacy-data**
URL list	Verify that its value is set to **GET**
URL text box	Enter **/xml**

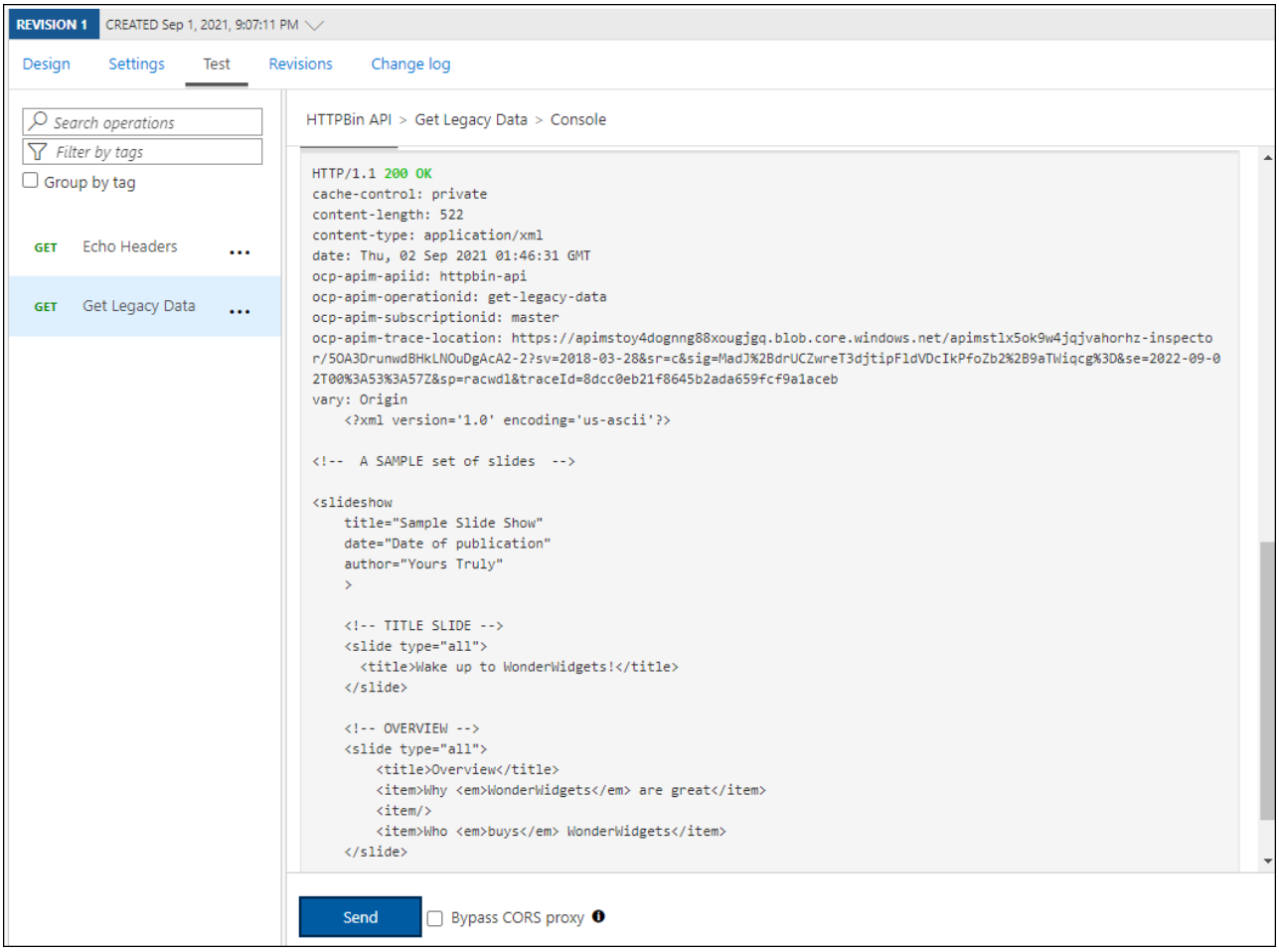
3. Back on the **Design** tab, in the list of operations, select **Get Legacy Data**.

4. Select the **Test** tab, and then select **Send**.

5. Review the results of the API request.

> **Note**: At this point, the results should be in XML format.

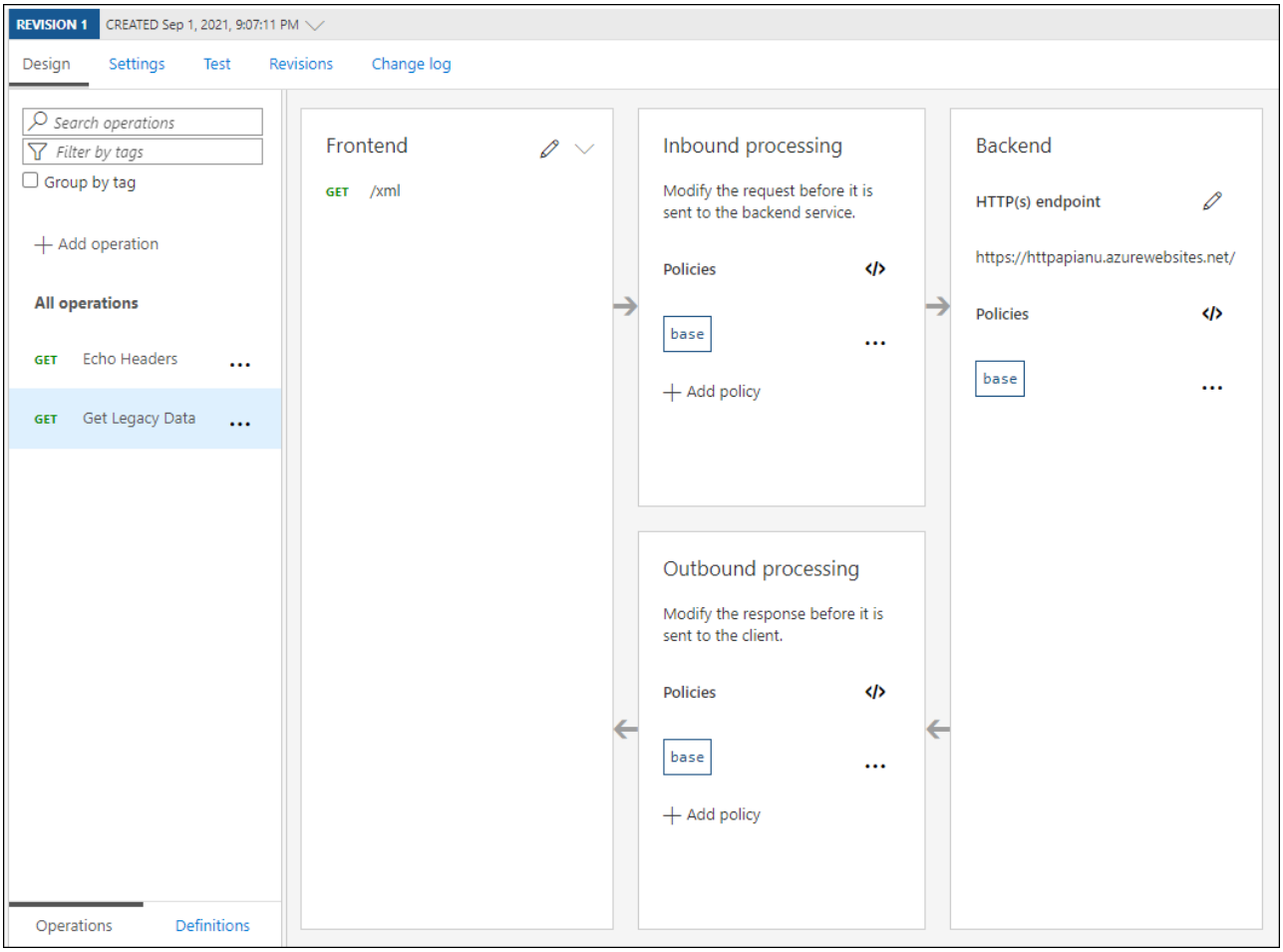
The following screenshot displays the results of the API request.



6. Select the **Design** tab, and then select **Get Legacy Data**.

7. On the **Design** pane, in the **Outbound processing** section, select **Add policy**.

   The following screenshot displays the **Outbound processing** section.



8. In the **Add outbound policy** section, select the **Other policies** tile.

9. In the policy code editor, find the following block of XML content:

```
<outbound>
    <base />
</outbound>
```

10. Replace that block of XML with the following XML:

```
<outbound>
    <base />
    <xml-to-json kind="direct" apply="always" consider-accept-header="false" />
</outbound>
```

11. In the policy code editor, select **Save**.

12. Back on the **Design** tab, in the list of operations, select **Get Legacy Data**, and then select **Test**.

13. In the **Get Legacy Data** section, select **Send**.

14. Review the results of the API request.

> **Note**: The new results are in JavaScript Object Notation (JSON) format.

15. Within the **HTTP response** section, perform the following actions:

    1. Select **Trace**.

    2. If prompted, select **Enable tracing for one hour**.

    3. Select the **Trace** tab, review the content in the **Backend** and **Outbound** text boxes, and note that they include details of the corresponding API operations with their timing information.

Task 4: Manipulate an API request

1. On the **Design** tab, select **+ Add operation**.

2. In the **Add operation** section, perform the following actions, and then select **Save**:

Setting	Action
Display name text box	Enter **Modify Status Code**
Name text box	Verify that its value is set to **modify-status-code**
URL list	Select **GET**
URL text box	Enter **/status/404**

3. Back on the **Design** tab, in the list of operations, select **Modify Status Code**.

4. In the **Design** section, on the **Inbound processing** tile, select **+ Add policy**.

5. In the **Add inbound policy** section, select the **Rewrite URL** tile.

6. In the **Rewrite URL** section, perform the following actions:

    a. In the **Backend** text box, enter **/status/200**.

    b. Select **Save**.

7. Back on the **Design** tab, in the list of operations, select **Modify Status Code**, and then select the **Test** tab.

8. In the **Modify Status Code** section, select **Send**.

9. Review the results of the API request.

> **Note**: Verify that the request returned the **HTTP/1.1 200 OK** response.

Review

In this exercise, you built a proxy tier between your App Service resource and any developers who wish to make queries against its API.