# Lab: Creating a multi-tier solution by using services in Azure

# Student lab manual

## Lab scenario

The developers in your company have successfully adopted and used the <https://httpbin.org/> website to test various clients that issue HTTP requests. Your company would like to use one of the publicly available containers on Docker Hub to host the httpbin web application in an enterprise-managed environment with a few caveats. First, developers who are issuing Representational State Transfer (REST) queries should receive standard headers that are used throughout the company's applications. Second, developers should be able to get responses by using JavaScript Object Notation (JSON) even if the API that's used behind the scenes doesn't support the data format. You're tasked with using Microsoft Azure API Management to create a proxy tier in front of the httpbin web application to implement your company's policies.

## Objectives

After you complete this lab, you will be able to:

* Create a web application from a Docker Hub container image.
* Create an API Management account.
* Configure an API as a proxy for another Azure service with header and payload manipulation.

## Lab setup

* Estimated time: **45 minutes**

## Instructions

### Before you start

#### Sign in to the lab virtual machine

Ensure that you're signed in to your Windows 10 virtual machine by using the following credentials:

* Username: **Admin**
* Password: **Pa55w.rd**

#### Review the installed applications

Find the taskbar on your Windows 10 desktop. The taskbar contains the icons for the applications that you'll use in this lab:

* Microsoft Edge

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

#### Task 1: Open the Azure portal

1. Sign in to the Azure portal (<https://portal.azure.com>).
2. If this is your first time signing in to the Azure portal, you'll notice a dialog box offering a tour of the portal. Select the **Get Started** button to skip the tour.

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

1. Create a new web app with the following details:
   * New resource group: **ApiService**
   * Name: **httpapi*[yourname]***
   * Publish: **Docker Container**
   * Operating system: **Linux**
   * Region: **East US**
   * New App Service plan: **ApiPlan**
   * SKU and size: **Premium V2 P1v2**
   * Docker options: **Single Container**
   * Image source: **Docker Hub**
   * Access type: **Public**
   * Image and tag: **kennethreitz/httpbin:latest**

* **Note**: Wait for Azure to finish creating the web app before you move forward with the lab. You'll receive a notification when the app is created.

#### Task 3: Test the httpbin web application

1. Access the **httpapi*[yourname]*** web app that you created earlier in this lab.
2. Open the **httpapi*[yourname]*** web app in your browser.
3. Test the API by selecting the **Response formats** option, selecting **GET /xml**, selecting **Try it out**, and then finally by selecting **Execute**.
4. Observe the response from the HTTP request. Specifically, observe the content of the **Request URL**, **Response body**, and **Response headers** text boxes.
5. Return to the Azure portal and the **Web App** blade for the **httpapi*[yourname]*** web app.
6. Access the **Properties** section of the **Web App** blade.
7. In the **Properties** section, record the value of the **URL** text box. You'll use this value later in the lab to make requests against the 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, create a new API Management service instance with the following details:
   * Existing resource group: **ApiService**
   * Name: **prodapi*[yourname]***
   * Location: **East US**
   * Organization name: **Contoso**
   * Pricing tier: **Consumption (99.9 SLA, %)**

#### Task 2: Define a new API

1. Access the **prodapi*[yourname]*** API Management service instance that you created earlier in this lab.
2. Create a new **Blank API** with the following details:
   * Display name: **HTTPBin API**
   * Name: **httpbin-api**
   * Web service URL: *Enter the URL for the web app that you copied earlier in this lab*
   * **Note**: Depending on how you copy the URL, you might need to add an "http://" prefix to create a valid URL value.
3. Add a new **operation** to the recently created API with the following details:
   * Display name: **Echo Headers**
   * Name: **echo-headers**
   * URL: **GET /**
4. Add a new **Set Headers** inbound policy to **All Operations** with the following details:
   * Name: **source**
   * Value: **azure-api-mgmt**
   * Action: **append**
5. Update the **Backend** for the **Echo Headers** operation by overriding the **Service URL** and appending **/headers** to its current value.

* **Note**: For example, if the current value is **http://httpapi*[yourname]*.azurewebsites.net**, the new value will be **http://httpapi*[yourname]*.azurewebsites.net/headers**

1. Test the **Echo Headers** operation in the **HTTPBin API**, observing the results of the API request.

* **Note**: Observe how there's many headers sent as part of your request that are echoed in the response. Specifically, you'll notice the new **Source** header that you created as part of this task.

#### Task 3: Manipulate an API response

1. Add a new **operation** to the API with the following details:
   * Display name: **Get Legacy Data**
   * Name: **get-legacy-data**
   * URL: **GET /xml**
2. Test the **Get Legacy Data** operation in the **HTTPBin API**, observing the results of the API request.

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

1. Add a new custom outbound policy scoped to the **Get Legacy Data** operation by first locating the following block of XML content:

* <outbound>  
   <base />  
  </outbound>

1. Replace that block of XML with the following XML, and then save the policy:

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

1. Test the **Get Legacy Data** operation in the **HTTPBin API**, observing the results of the API request.

* **Note**: The new results are in JSON format.

1. Use the **Trace** feature of the test tool to observe the request sent to the back-end service.

#### Review

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

### Exercise 3: Clean up your subscription

#### Task 1: Open Azure Cloud Shell

1. In the portal, select the **Cloud Shell** icon to open a new shell instance.
2. In the portal, at the **Cloud Shell** command prompt, enter the following command, and then select Enter to list all resource groups in the subscription:

* az group list

1. Enter the following command, and then select Enter to get a list of possible commands to delete a resource group:

* az group delete --help

#### Task 2: Delete resource groups

1. Enter the following command, and then select Enter to delete the **ApiService** resource group:

* az group delete --name ApiService --no-wait --yes

1. Close the Cloud Shell pane in the portal.

#### Task 3: Close the active applications

* Close the currently running Microsoft Edge application.

#### Review

In this exercise, you cleaned up your subscription by removing the resource groups used in this lab.

©2019 Microsoft Corporation. All rights reserved. The text in this document is available under the [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/legalcode), additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are ***not*** included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.