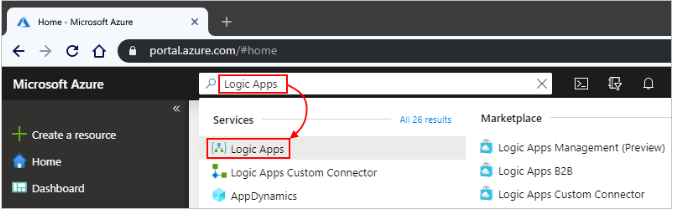
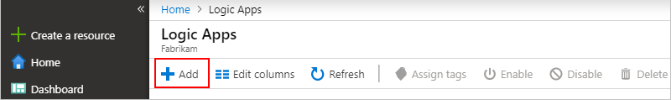
**API Management with Logic Apps**

**Creating Logic App that receive and respond to HTTPS calls in Azure Portal**

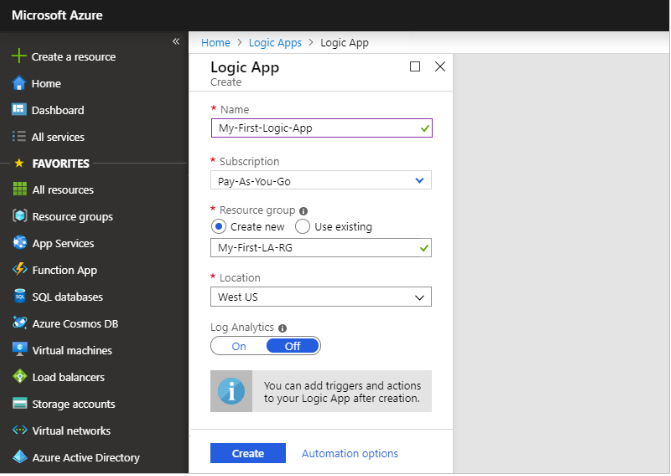
1. Sign in to the Azure portal with your Azure account credentials.
2. From the Azure home page, in the search box, find and select LogicApps.



1. On the LogicApps page, select Add.



1. On the Logic App pane, provide details about your logic app as shown below. After you're done, select Create



Name: Your logic app name, which can contain only letters, numbers, hyphens, underscores, parentheses, and periods. This example uses "My-First-Logic-App".

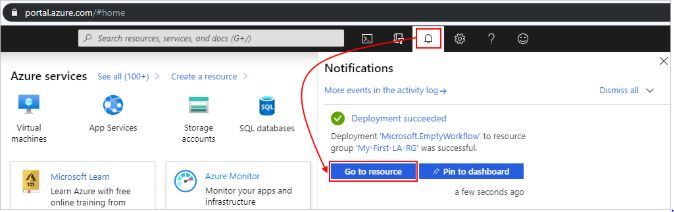
**Subscription:** your azure subscription name.

**Resource group:** The name for the Azure resource group used to organize related resources if created else create new by selecting create new radio button and enter the name.

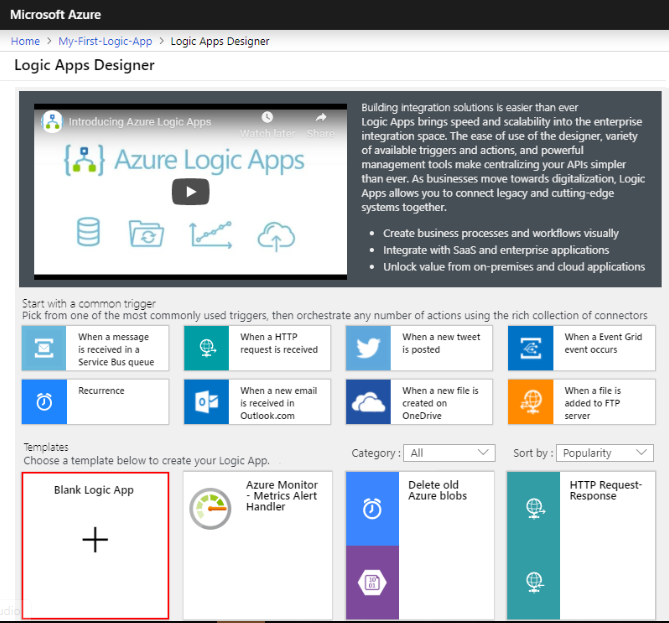
**Location:** The region where to store your logic app information.

**Log Analytics:** Keep the **Off** setting for diagnostic logging.

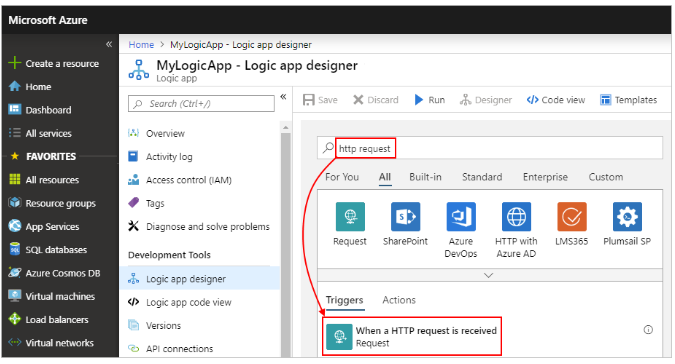
1. After Azure deploys your app, on the Azure toolbar, select **Notifications** > **Go to resource** for your deployed logic app or you can find and select your logic app by typing the name in the search box.



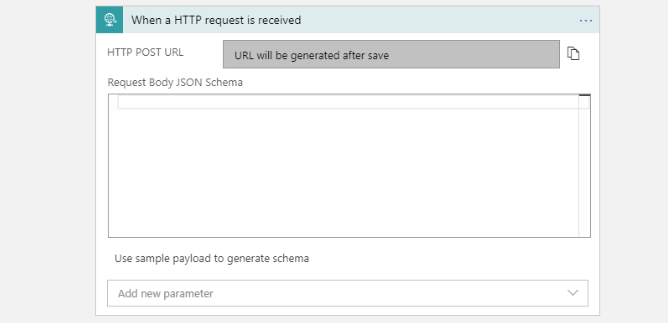
1. The Logic Apps Designer opens and shows a page with an introduction video and commonly used triggers. Under Templates, select Blank Logic App.



1. After Logic App Designer opens, in the search box, enter "http request" as your filter. From the triggers list, select the **When an HTTP request is received** trigger, which is the first step in your logic app workflow.



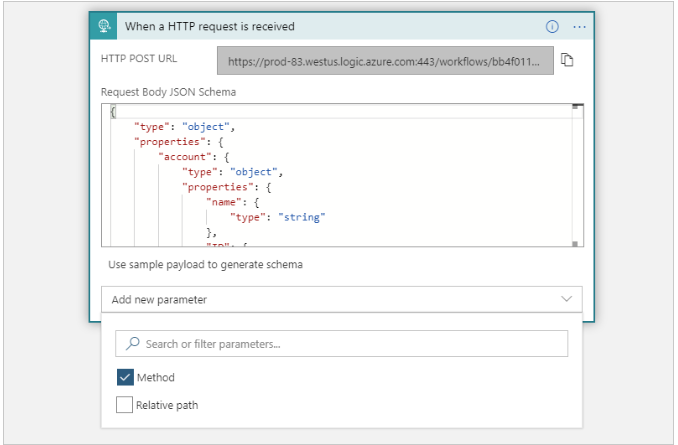
1. The Request trigger shows these properties:



**HTTP POST URL:** It is a mandatory field**.** The endpoint URL will be generated after you save the logic app and is used for calling your logic app (from API Management, or http calls)

**Request Body JSON Schema:** It is optional fieldand it is a JSON schema that describes the properties and values in the incoming request body.

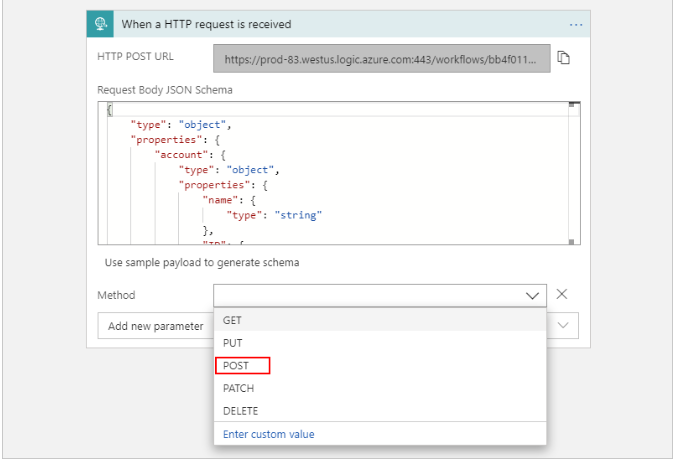
1. To specify additional properties, open the Add newparameter list, and select the parameters that you want to add



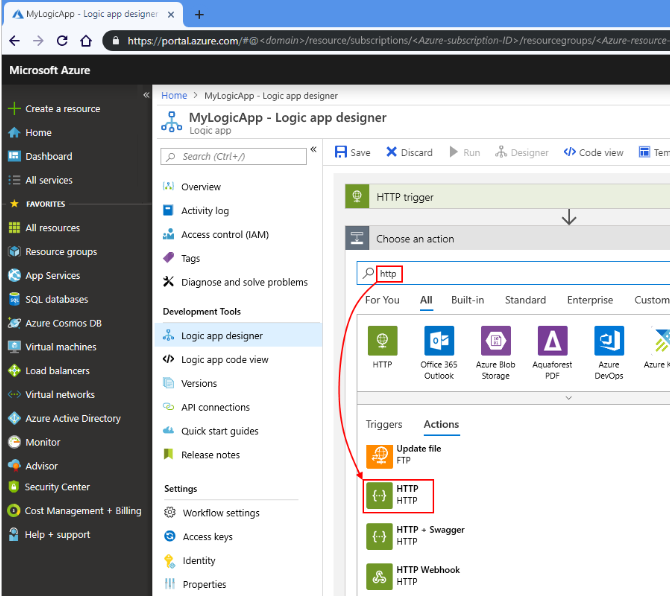
**Method:** The method that the incoming request must use to call the logic app

**Relative path:** The relative path for the parameter that the logic app's endpoint URL can accept.

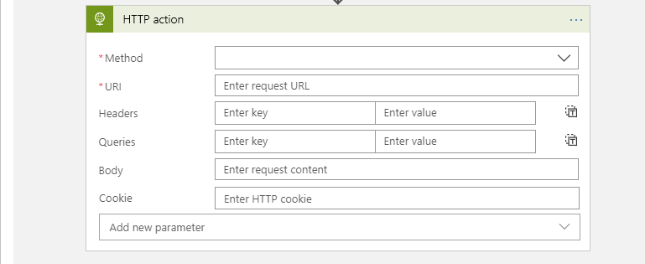
1. The **Method** property appears in the trigger so that you can select a method from the list.



1. Under Choose an action, in the search box, enter "http" as your filter. From the **Actions** list, select the **HTTP** action.



1. Provide the values for the HTTP action parameters that you want to include in the call to the target endpoint.



Method: The method type of request we are making from the logic app.

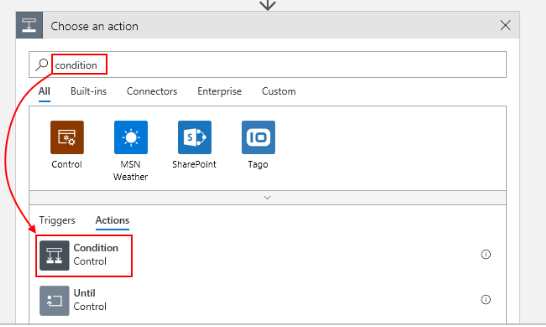
URL: the endpoint we are requesting.

Header: Header of request.

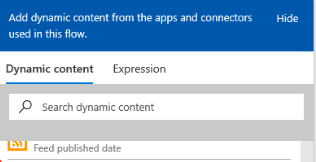
Body: Body of the request.

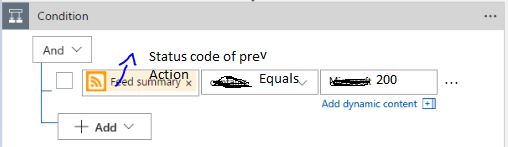
Add Parameters: to add parameters to request.

1. Then In the search box, enter "condition" as your filter. Select this action: Condition – Controlto perform next actions based on previous Action result

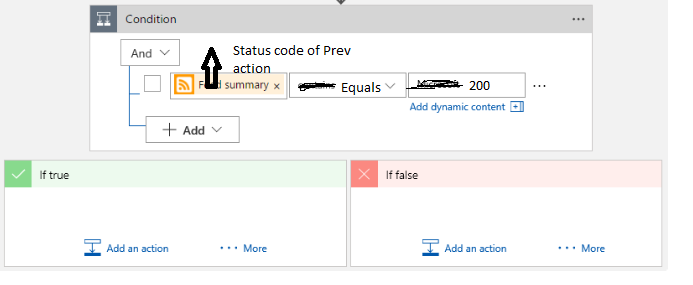


1. In the **Condition** box, build your condition.
2. In the left box, specify the data or field that you want to compare i.e the response of previous action. Lets build condition to check the status code of prev action response equals 200 or nor . Get the status code of previous action from dynamic content tab.
3. In middle box, select the operation to perform. For this example, select "Equals".
4. In the right box, specify a value or field as your criteria. For this example, specify this status code: 200





1. Here's the complete condition:



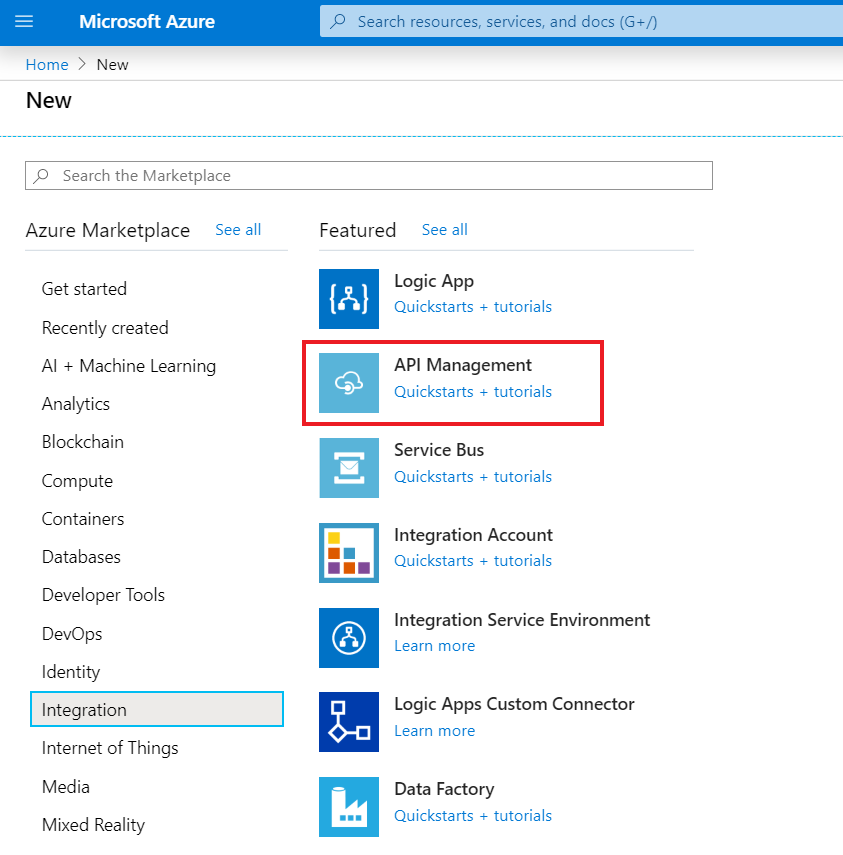
If result of the condition is true, perform another action specific to True result.

If result of the condition is false, perform another action specific to False result.

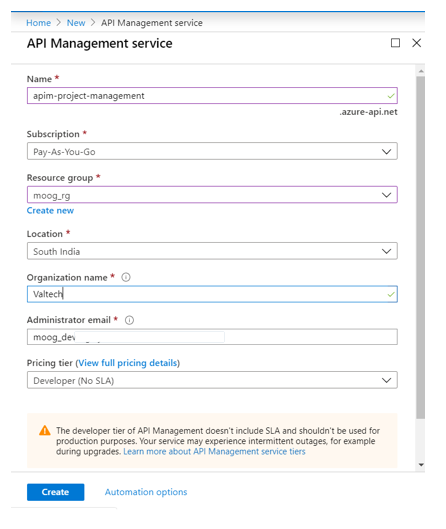
1. When we are done, let us save logic app by select Save on the designer toolbar.

**Azure API Management**

* Create a New API Management Service



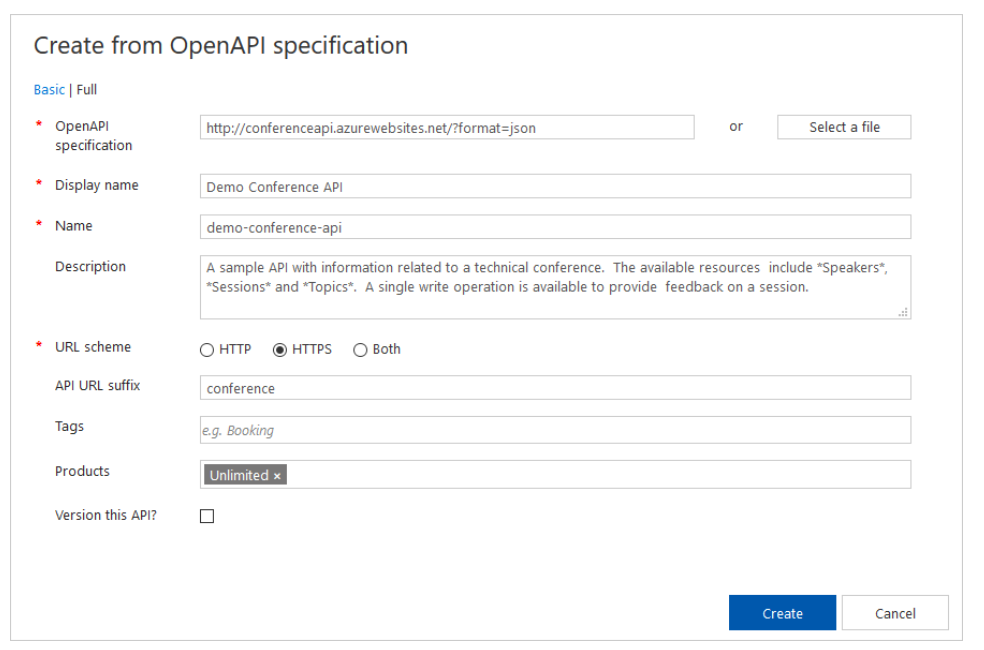
* In the **API Management service** screen, enter settings and click on create to deploy the API Management service. Generally, it will take 20-30 min to successfully deploy the service.



* After successful deployment, open the API Management instance.

**Import and publish a backend API:**

1. Select **APIs** from under **API MANAGEMENT**.
2. Select **OpenAPI specification** from the list and click **Full** in the pop-up.
3. We can set the API values during creation or later by going to the **Settings** tab.



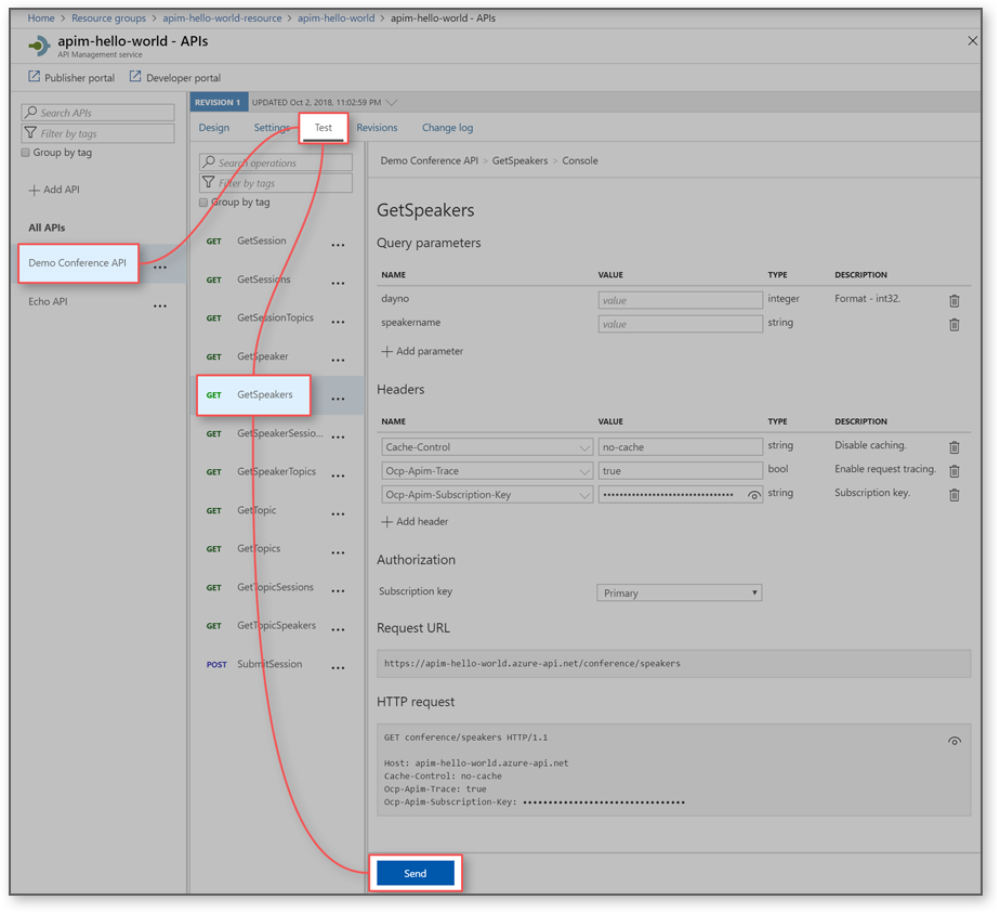
1. Select **Create**.

**Test the new API in the Azure portal:**

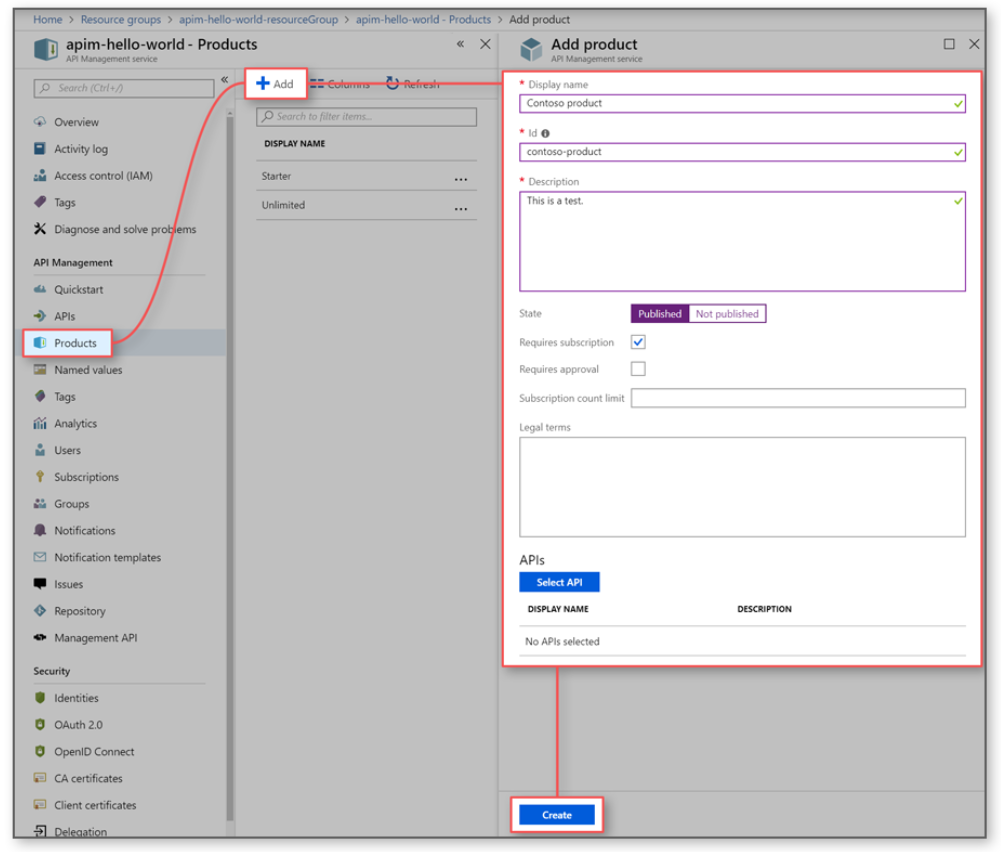
Operations can be called directly from the Azure portal, which provides a convenient way to view and test the operations of an API.

1. Select the API you created in the previous step (from the **APIs** tab).
2. Press the **Test** tab.
3. Click on **GetSpeakers**. The page displays fields for query parameters, in this case none, and headers. One of the headers is "Ocp-Apim-Subscription-Key", for the subscription key of the product that is associated with this API. The key is filled in automatically.
4. Press **Send**.

Backend responds with **200 OK** and some data.



**Create and publish a product:**

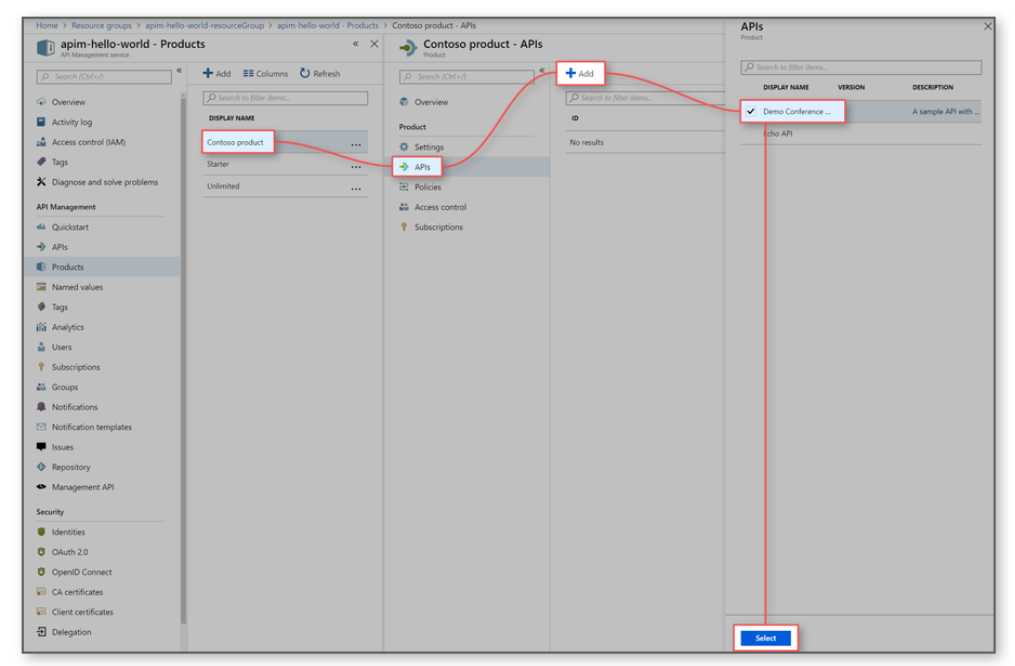


1. Click on **Products** in the menu on the left to display the **Products** page.
2. Click **+ Add**.
3. Fill the necessary fields.
4. Click **Create** to create the new product.

**Add more configurations**

* You can continue configuring the product after saving it by choosing the **Settings** tab.
* View/add subscribers to the product from the **Subscriptions** tab.
* Set visibility of a product for developers or guests from the **Access control** tab.

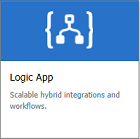
**Add an API to an existing product:**



1. From the **Products** tab, select a product.
2. Navigate to the **APIs** tab.
3. Click **+ Add**.
4. Choose an API and click **Select**.

**Import a Logic App as an API:**

1. Select **APIs** from under **API MANAGEMENT**.
2. Select **Logic App** from the **Add a new API** list.



1. Press **Browse** to see the list of Logic Apps with HTTP trigger in your subscription.
2. Select the app. API Management finds the swagger associated with the selected app, fetches it, and imports it.
3. Add an API URL suffix. The suffix is a name that identifies this specific API in this API Management instance. It has to be unique in this API Management instance.
4. Publish the API by associating the API with a product. In this case, the "*Unlimited*" product is used. If you want for the API to be published and be available to developers, add it to a product. You can do it during API creation or set it later.

By default, each API Management instance comes with two sample products:

* + **Starter**
  + **Unlimited**

1. Select **Create**.

**Test the API in the Azure portal:**

Operations can be called directly from the Azure portal, which provides a convenient way to view and test the operations of an API.

1. Select the API you created in the previous step.
2. Press the **Test** tab.
3. Select some operation.

The page displays fields for query parameters and fields for the headers. One of the headers is "Ocp-Apim-Subscription-Key", for the subscription key of the product that is associated with this API. If you created the API Management instance, you are an administrator already, so the key is filled in automatically.

1. Press **Send**.

Backend responds with **200 OK** and some data.

**Note:**

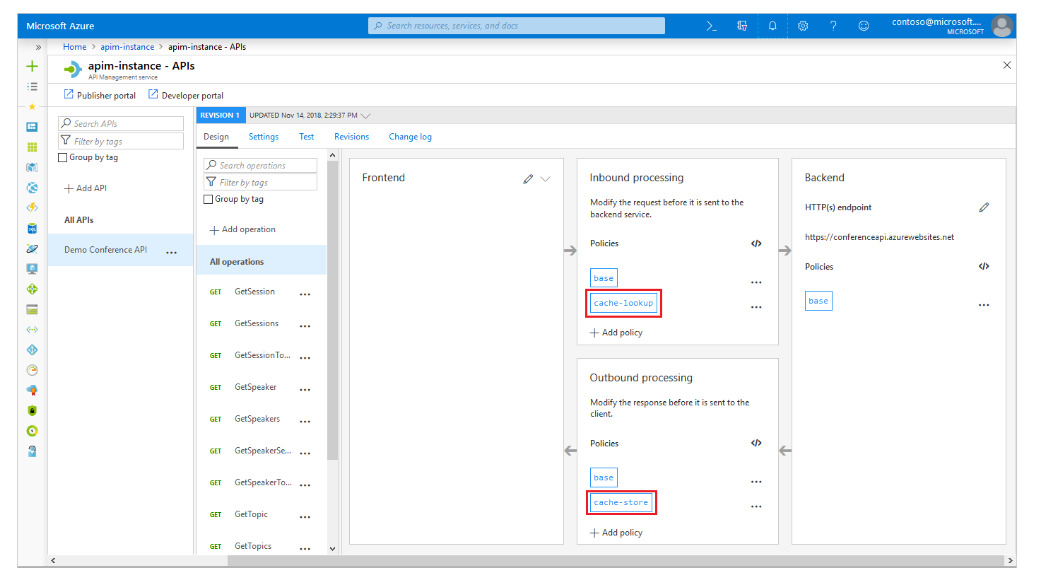
* Launch the developer portal to test and run all the APIs, which are subscribed to the products associated with the APIM instance. In addition, we can get all the information like subscription key and product details.

**Mock API responses:**

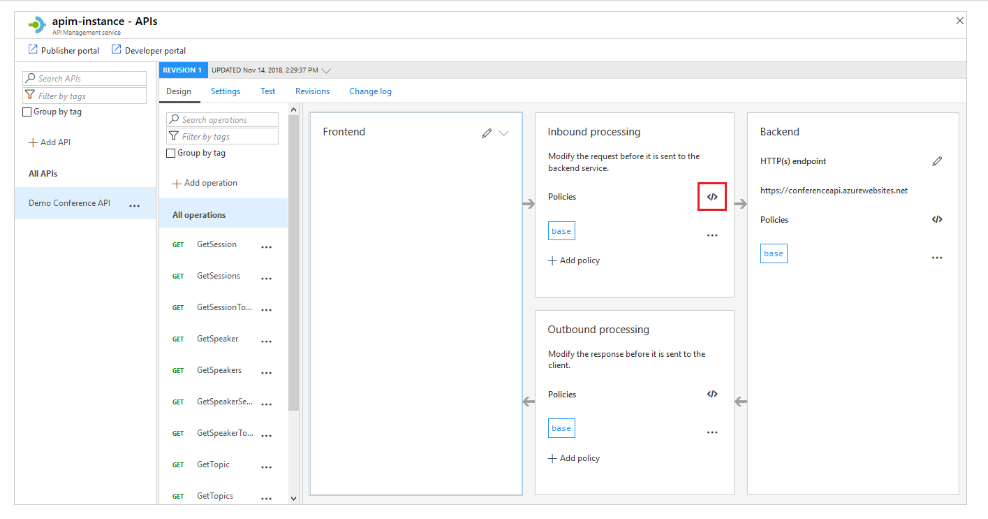
We can mock API responses by creating a blank API with no backend. Go through the below link for more info.

“<https://docs.microsoft.com/en-us/azure/api-management/mock-api-responses>”

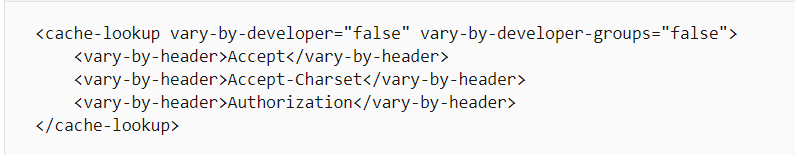
**Add the caching policies:**



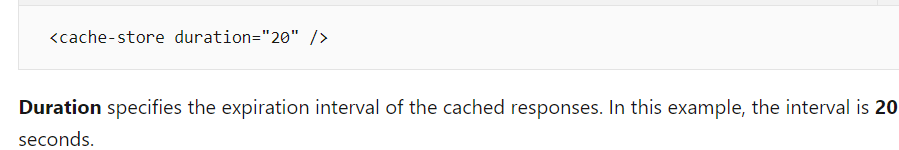
1. Browse to your APIM instance.
2. Select the **API** tab.
3. Click **your API** from your API list.
4. Select **an API**.
5. On the top of the screen, select **Design** tab.
6. In the **Inbound processing** section, click the **</>** icon.



1. In the **inbound** element, add the following policy:



1. In the **outbound** element, add the following policy:



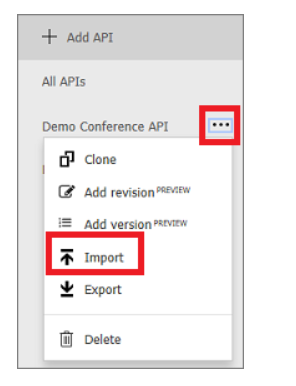
1. To see the caching in action, call the operation from the developer portal.

**Note:**

* We can add policies to protect the APIs, “<https://docs.microsoft.com/en-us/azure/api-management/transform-api>”
* Launch the developer portal to run and test the APIs.

**Append other APIs:**

An API can be composed of APIs exposed by different services, including the OpenAPI Specification, a SOAP API, the API Apps feature of Azure App Service, Azure Function App, Azure Logic Apps, and Azure Service Fabric.



To append a different API to your existing API, complete the following steps. When you import another API, the operations are appended to your current API.

1. Go to your Azure API Management instance in the Azure portal.
2. Select **APIs** from the menu on the left.
3. Click **...** next to the API that you want to append another API to.
4. Select **Import** from the drop-down menu.
5. Select a service from which to import an API.