



API Management Custom Policies –Exposing Azure Blob Storage

Objective

In this lab, we will be exposing Azure Blob Storage as a REST endpoint through API Management. To do this, we will create a [custom policy](#) for API Management. Azure Blob Storage exposes several [REST operations](#), which can be used to manage the container, as well as execute CRUD actions. In this lab we will use an API call to upload a file to a container, while abstracting the Blob Storage's specific authentication method from the API consumer. This means that our consumer does not need to know how to authenticate and communicate with API Management, but instead only needs to know how to do this with API Management, just like any other API which would be published through here. For more information around this and similar scenario's, check this [blog post](#) by [Eldert Grootenboer](#).



Prerequisites

- Azure account - You can [Open an Azure account for free](#) or [Activate Visual Studio subscriber benefits](#).
- [Postman](#)

Steps

To build the solution in this lab you have to follow the steps described in this section. From a high level view the steps are:

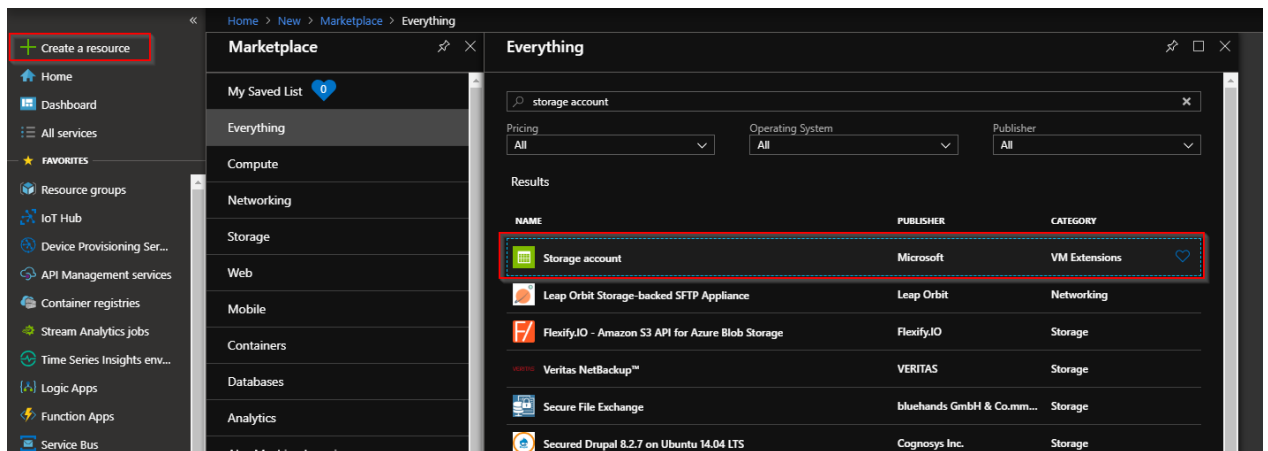
1. Create storage container

Create storage account

The first step in building the solution in this lab is to provision a storage account in Azure. We will be needing a storage account for storing the order request message in Blob Storage.

1. Go to the Azure Portal: <https://portal.azure.com/>
2. Login into the Azure portal with your account.
3. In the Market Place enter storage account and select it from the list as shown below.

API Management Custom Policies - Exposing Azure Blob Storage



- Click Create.
- Specify a name, a Resource Group (you can create a new one here if you haven't created a resource group yet) and a location. Subsequently, click on Review + create.

Create storage account

Basics Advanced Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

PROJECT DETAILS

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

* Subscription: Visual Studio Enterprise

* Resource group: (New) GIB2019 [Create new](#)

INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

* Storage account name: custompolicyblob ✓

* Location: West Europe

Performance: ☒ Standard ☐ Premium

Account kind: StorageV2 (general purpose v2)

Replication: Read-access geo-redundant storage (RA-GRS)

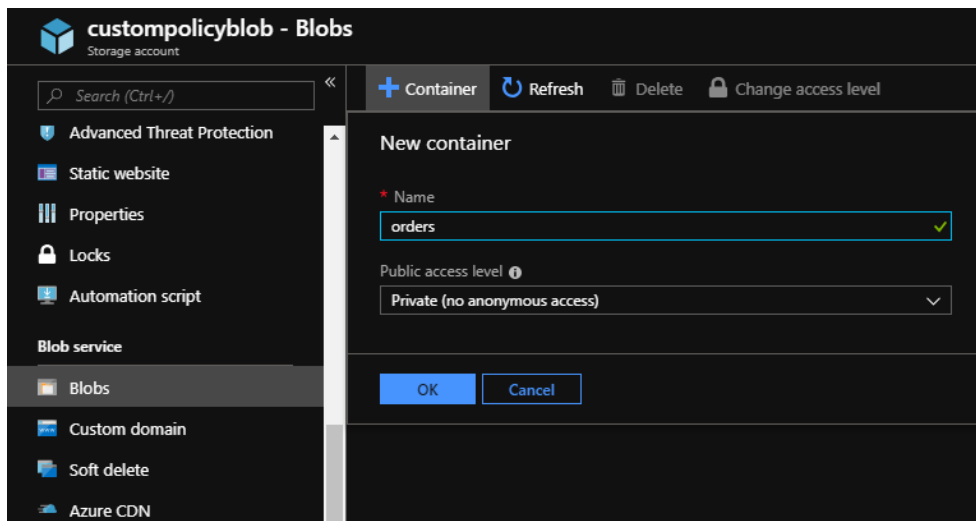
Access tier (default): ☐ Cool ☒ Hot

[Review + create](#) [Previous](#) [Next: Advanced >](#)

Create storage container

Once the storage account has been provisioned you can navigate to it and click on it.

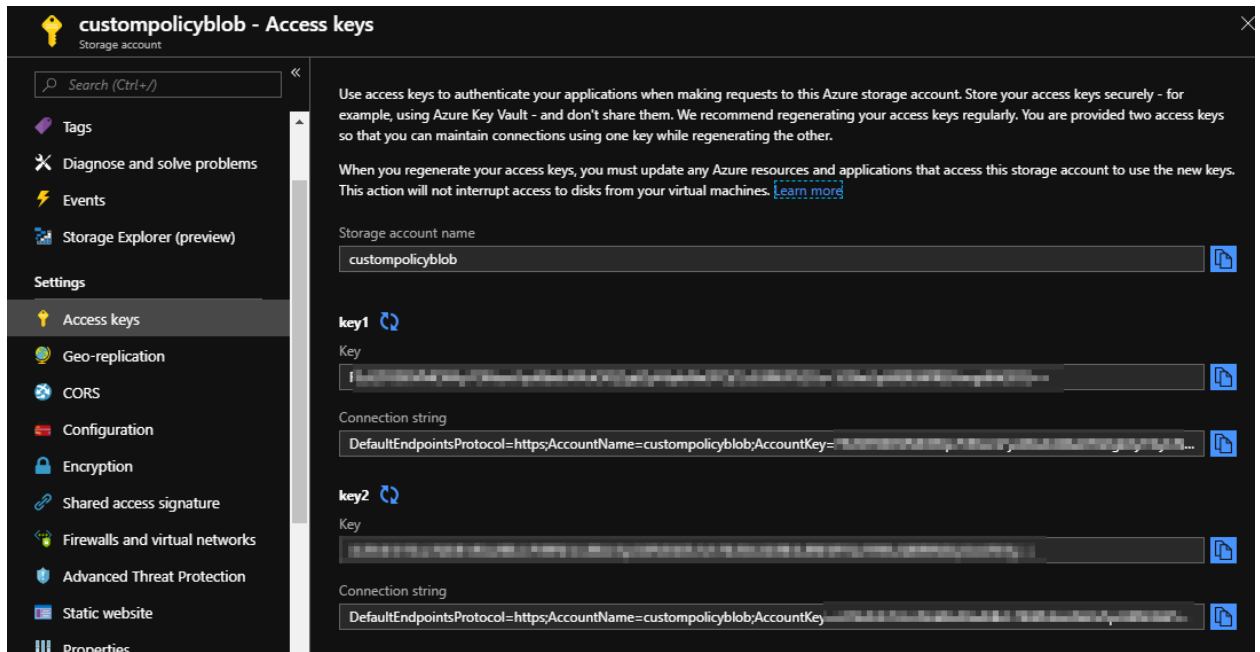
1. In the storage account click on Blobs.
2. Click on + Container and specify the name and Access Type: Private.



Get storage account key

Once the container has been created, go back to the storage account.

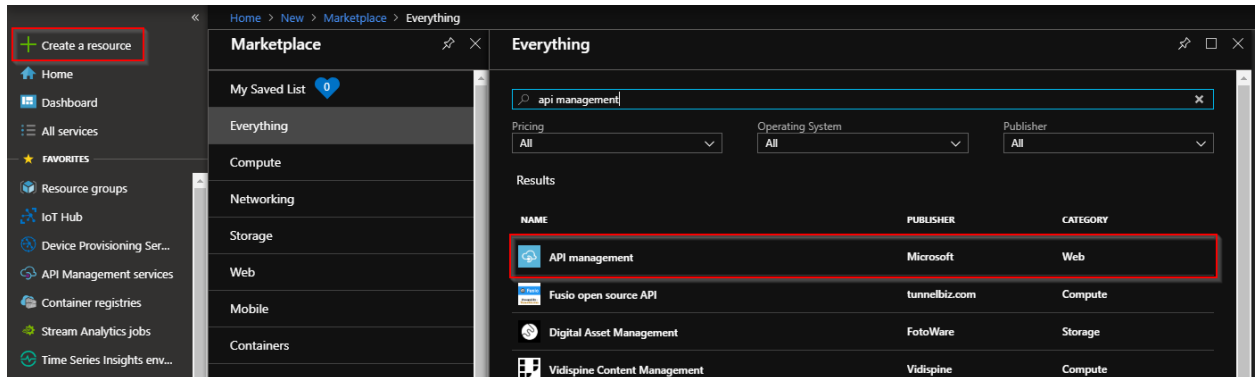
1. Open Access keys.
2. Grab one of the two keys, this will be needed later.



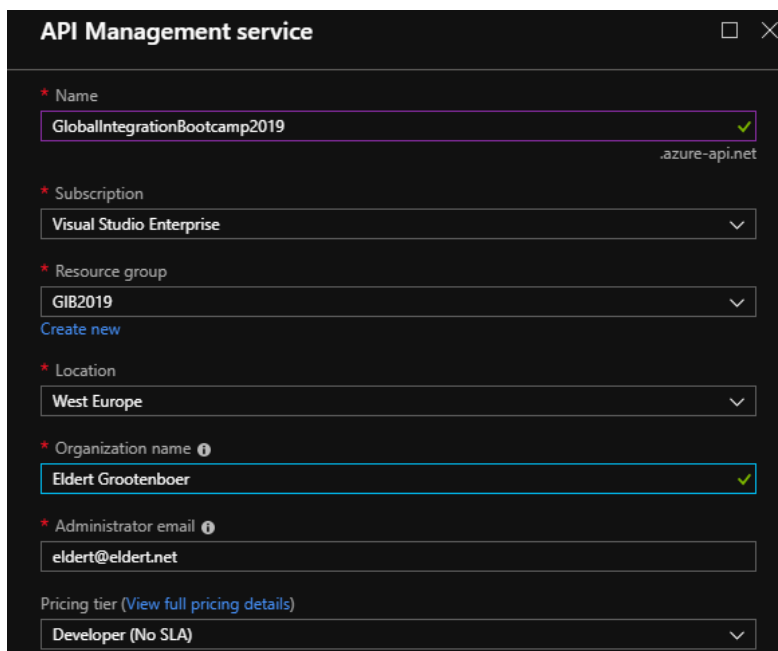
Create API Management instance

Next, we will create an API management instance which will be used to expose our blob storage to our consumer.

1. Go to the Azure Portal: <https://portal.azure.com/>
2. Login into the Azure portal with your account.
3. In the Market Place enter api management and select it from the list as shown below.



6. Click Create.
7. Specify a name, a Resource Group (you can create a new one here or use the resource group we created in the previous steps) and a location. Subsequently, click on Create.



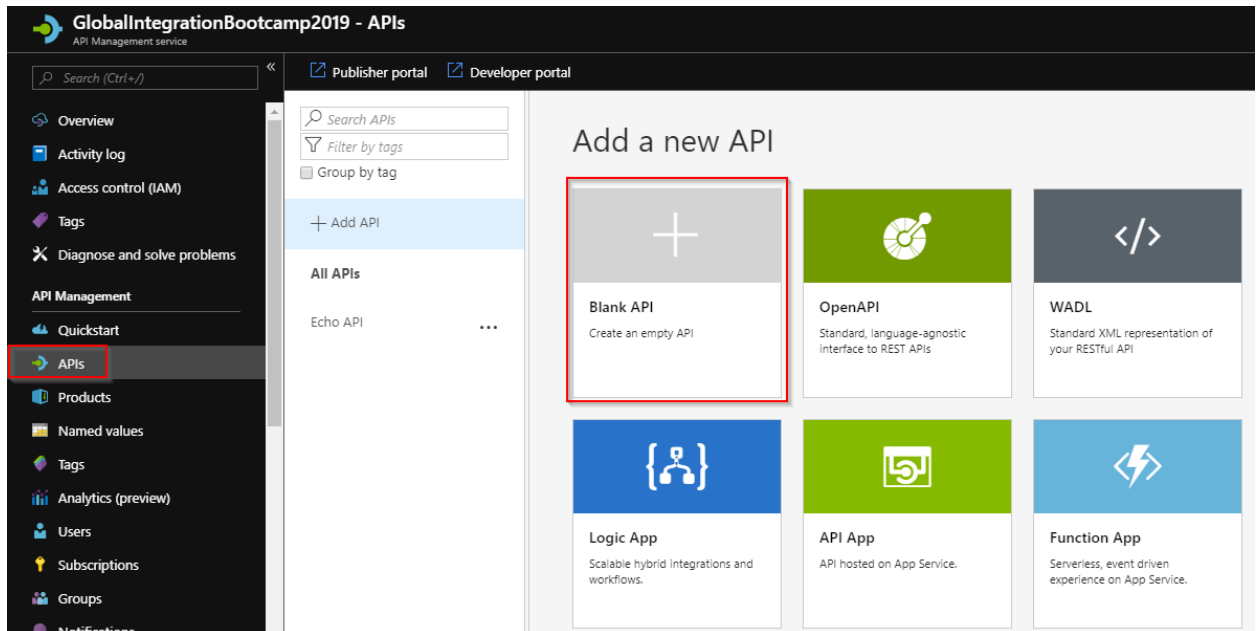
The screenshot shows the 'API Management service' creation form. The fields are filled as follows:

- Name:** GlobalIntegrationBootcamp2019 (with a green checkmark)
- Subscription:** Visual Studio Enterprise
- Resource group:** GIB2019
- Location:** West Europe
- Organization name:** Eldert Grootenboer (with a green checkmark)
- Administrator email:** eldert@eldert.net
- Pricing tier:** Developer (No SLA)

Create Order API

Once the instance has been created, you can click on it to navigate to your API Management.

1. Open APIs tab.
2. Click on Add API – Blank API



3. Give the API a name, suffix and add it to the Unlimited product, and click Create. We won't need to set the web service URL, as we will be defining this from our custom policy.

Create a blank API

Basic | **Full**

*

Display name

*

Name

Web service URL

API URL suffix

Products

Create

Cancel

Add operation

Navigate to the API we just created.

1. Click on Add operation.
2. Provide a display name and name for the operation.
3. Set the method to Put and provide the suffix.
4. Click on Save.

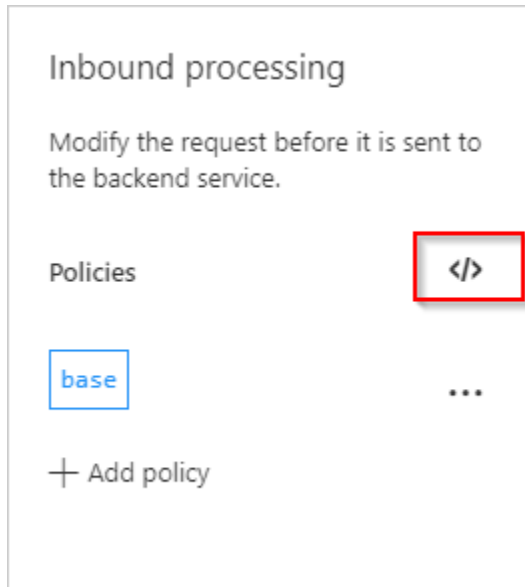
Frontend

* Display name	<input type="text" value="Create"/>
* Name	<input type="text" value="create"/>
* URL	<div><div>PUT</div><div>▼</div></div> <input type="text" value="/create"/>
Description	<div><div></div><div></div></div>
Tags	<input type="text" value="e.g. Booking"/>

Set policy

Navigate to the operation we just added.

1. Open the Inbound processing policy for the operation which was just created.



2. We will start by creating some variables in the inbound policy section. These variables will be used to calculate the SAS key [authorization header](#), which is needed when communicating with Azure Blob Storage through the REST API. Everything we do in the following steps need to be inside the <inbound> section, right after the <base /> statement.
 - a. Add a variable for the current date and time.
`<set-variable name="UTCNow" value="@{DateTime.UtcNow.ToString("R")}" />`
 - b. Get the name of the blob which we will be placing in the container from the incoming request headers.
`<set-variable name="BlobName" value="@{context.Request.Headers.GetValueOrDefault("Blob")}" />`
3. Now we will add the headers needed by the Azure Blob Storage API. These headers will be sent to the backend service to do the authentication.
 - a. Add the x-ms-date header.
`<set-header name="x-ms-date" exists-action="override">
 <value>@{context.Variables.GetValueOrDefault<string>("UTCNow")}</value>
 </set-header>`
 - b. Add the x-ms-version header.
`<set-header name="x-ms-version" exists-action="override">
 <value>2018-03-28</value>
 </set-header>`
 - c. Add the x-ms-blob-type header.
`<set-header name="x-ms-blob-type" exists-action="override">`



API Management Custom Policies - Exposing Azure Blob Storage

```
<value>BlockBlob</value>
</set-header>
```

4. Now we will add our custom policy which will calculate the authorization header and adds it to our request headers. This policy uses the input from our variables and request to calculate the HMACSHA256 which is then hashed. The values for the storageAccount, storageKey and containerName come from the previous steps in this lab.

```
<set-header name="Authorization" exists-action="override">
    <value>@{
        string storageAccount = "yourStorageAccountName";
        string containerName = "yourContainerName";
        string storageKey = "yourStorageAccountKey";
        string blobName = context.Variables.GetValueOrDefault<string>("BlobName");
        string contentLength = context.Request.Headers.GetValueOrDefault("Content-
Length");
        string ContentType = context.Request.Headers.GetValueOrDefault("Content-Type");
        string dateToSign = context.Variables.GetValueOrDefault<string>("UTCNow");
        var stringToSign = $"PUT\n\n{n}{contentLength}\n\n{ContentType}\n\n\n\n\n\n\n\nx-
ms-blob-type:BlockBlob\nx-ms-date:{dateToSign}\nx-ms-version:2018-03-
28\n/{storageAccount}/{containerName}/{blobName}";
        HMACSHA256 hmac = new HMACSHA256(Convert.FromBase64String(storageKey));
        string signature =
Convert.ToBase64String(hmac.ComputeHash(Encoding.UTF8.GetBytes(stringToSign)));
        string authorizationHeader = String.Format("{0} {1}:{2}", "SharedKey",
storageAccount, signature);
        return authorizationHeader;
    }</value>
</set-header>
```

5. And finally, we will set the URL of the Blob Storage API as our backend service, which will be called from API Management.

```
<set-backend-service base-url="https://yourStorageAccountName.blob.core.windows.net/" />
<rewrite-uri template="@{
    string containerName = "yourContainerName";
    string blobName = context.Variables.GetValueOrDefault<string>("BlobName");
    return String.Format("/{0}/{1}", containerName, blobName);
}" />
```

6. Save the policy.



API Management Custom Policies - Exposing Azure Blob Storage

7. As a result, the complete policy which we have just set is as following.

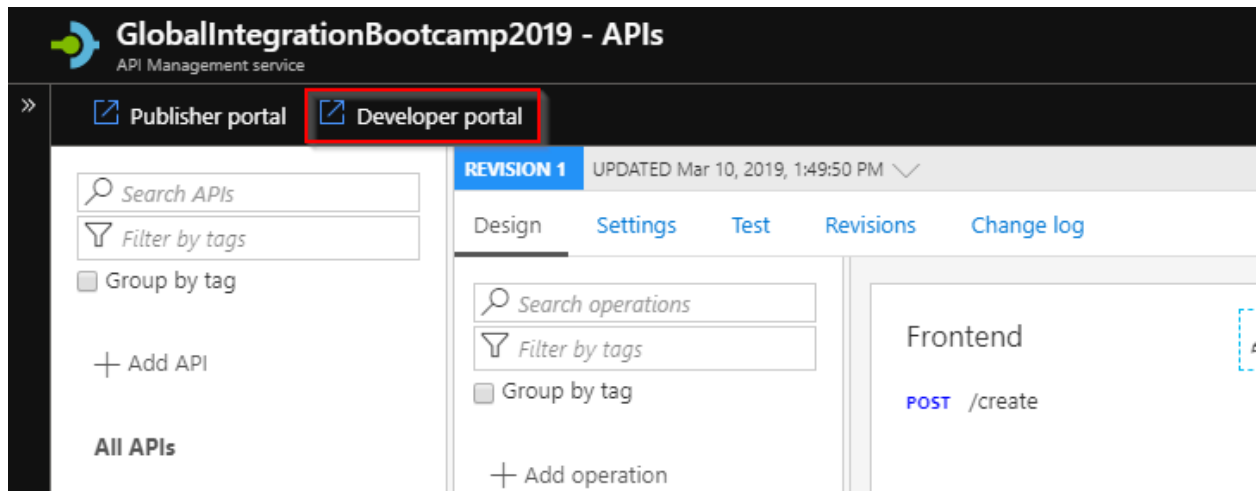
```
<policies>
<inbound>
  <base />
  <set-variable name="UTCNow" value="@{(DateTime.UtcNow.ToString("R"))}" />
  <set-variable name="BlobName" value="@{(context.Request.Headers.GetValueOrDefault("Blob"))}" />
  <set-header name="x-ms-date" exists-action="override">
    <value>@(context.Variables.GetValueOrDefault<string>("UTCNow"))</value>
  </set-header>
  <set-header name="x-ms-version" exists-action="override">
    <value>2018-03-28</value>
  </set-header>
  <set-header name="x-ms-blob-type" exists-action="override">
    <value>BlockBlob</value>
  </set-header>
  <set-header name="Authorization" exists-action="override">
    <value>@{
      string storageAccount = "yourStorageAccountName";
      string containerName = "yourContainerName";
      string storageKey = "yourStorageAccountKey";
      string blobName = context.Variables.GetValueOrDefault<string>("BlobName");
      string contentLength = context.Request.Headers.GetValueOrDefault("Content-Length");
      string ContentType = context.Request.Headers.GetValueOrDefault("Content-Type");
      string dateToSign = context.Variables.GetValueOrDefault<string>("UTCNow");
      var stringToSign = $"PUT\n\n\n{contentLength}\n\n{ContentType}\n\n\n\n\n\n\nx-ms-blob-type:BlockBlob\n\nms-
date:{dateToSign}\n\nx-ms-version:2018-03-28\n/{storageAccount}/{containerName}/{blobName}";
      HMACSHA256 hmac = new HMACSHA256(Convert.FromBase64String(storageKey));
      string signature = Convert.ToBase64String(hmac.ComputeHash(Encoding.UTF8.GetBytes(stringToSign)));
      string authorizationHeader = String.Format("{0} {1}:{2}", "SharedKey", storageAccount, signature);
      return authorizationHeader;
    }</value>
  </set-header>
  <set-backend-service base-url="https://yourStorageAccountName.blob.core.windows.net/" />
  <rewrite-uri template="@{
    string containerName = "yourContainerName ";
    string blobName = context.Variables.GetValueOrDefault<string>("BlobName");
    return String.Format("/{0}/{1}", containerName, blobName);
  }" />
</inbound>
<backend>
  <base />
</backend>
<outbound>
  <base />
</outbound>
<on-error>
  <base />
</on-error>
</policies>
```

</policies>

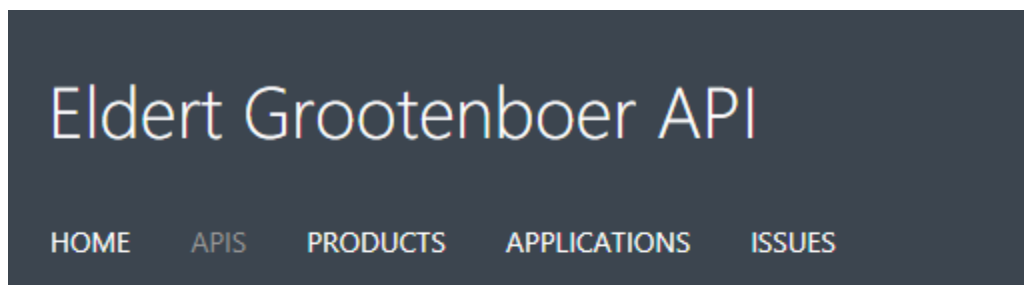
Test

To test the solution, you can use the developer portal of API Management.

1. Open the developer portal.



2. Navigate to the API we just created.



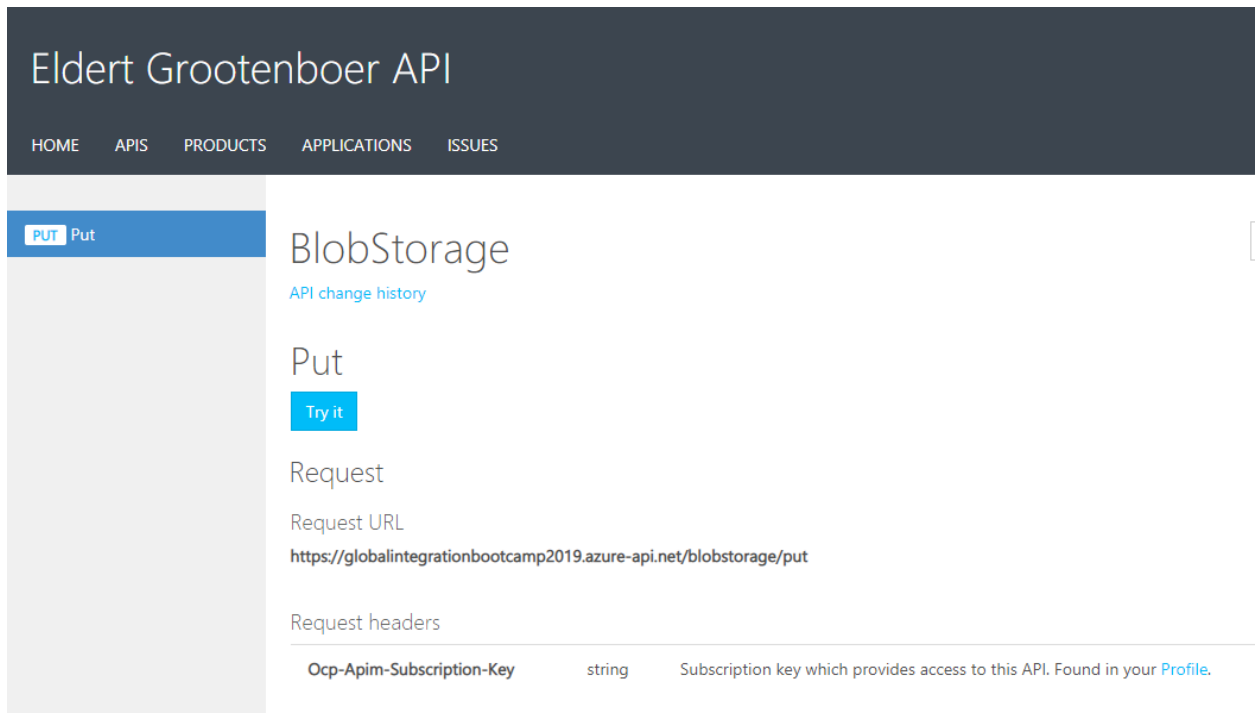
APIs

BlobStorage

Echo API

Service Bus

- Click on Try it.



The screenshot shows the API console for 'Eldert Grootenboer API'. The left sidebar has a 'PUT Put' button. The main area displays 'BlobStorage' with a link to 'API change history'. Below this, the 'Put' method is highlighted with a 'Try it' button. The 'Request' section shows the 'Request URL' as `https://globalintegrationbootcamp2019.azure-api.net/blobstorage/put`. The 'Request headers' section shows a table with one header: 'Ocp-Apim-Subscription-Key' of type 'string', with a description: 'Subscription key which provides access to this API. Found in your [Profile](#).'

- Add a header called Blob with a value of Test.json. Use the following message body and click on Send.

```
{
  "name" : "John Smith",
  "sku" : "20223",
  "price" : 23.95,
  "shipTo" : {
    "name" : "Jane Smith",
    "address" : "123 Maple Street",
    "city" : "Pretendville",
    "state" : "NY",
    "zip" : "12345"
  },
  "billTo" : {
    "name" : "John Smith",
    "address" : "123 Maple Street",
    "city" : "Pretendville",
    "state" : "NY",
    "zip" : "12345"
  }
}
```

- | Response | Trace |
|--|-------|
| Response status | |
| 201 Created | |
| Response latency | |
| 72 ms | |
| Response content | |
| <pre>x-ms-request-id: 920bed9c-c01e-0099-7f52-d7929b000000 x-ms-version: 2018-03-28 x-ms-request-server-encrypted: true Ocp-Apim-Trace-Location: https://apimgmtsttgqry0y7kqh4trh.blob.core.windows.net/apiinspectorcontainer/CeBZmDLtT5JYGta-Xb1bSA2-101?sv=2017-04-17&sr=b&sig=UwdeXokgYLeHYhq4BbxY8fUwhX7P8nfwRVLuRNwSh4o%3D&se=2019-03-11T15%3A03%3A35Z&sp=r&traceId=8e0eae9282124556b73578cc1ccd32c2 Date: Sun, 10 Mar 2019 15:03:35 GMT ETag: "0x8D6A56991FA3A6A" Content-Length: 0 Content-MD5: uBQxcsp2VB1xhthm3U4dtg== Last-Modified: Sun, 10 Mar 2019 15:03:35 GMT</pre> | |