[**1. SECTION – 1 : MircroService Key/Vault.** 1](#_Toc173611862)

[**2. SECTION – 2 : Service Discovery using cosul docker.** 9](#_Toc173611863)

[**3. SECTION – 2.1 : Service Discovery using APIM.** 12](#_Toc173611864)

# **1. SECTION – 1 : MircroService Key/Vault.**

1. Open Portal.Azure.Com and create the resource name it as Microservice and Key Vault Name as shtvault.

A screenshot of a computer

Description automatically generated

Next >

A screenshot of a computer

Description automatically generated

A close-up of a person's face

Description automatically generated

Create >

A screenshot of a computer

Description automatically generated

Close the Window >

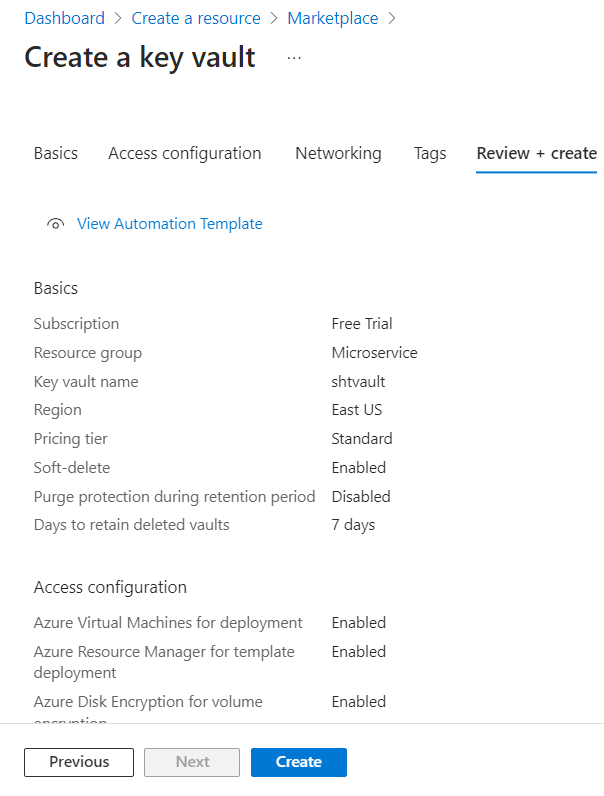
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Next >

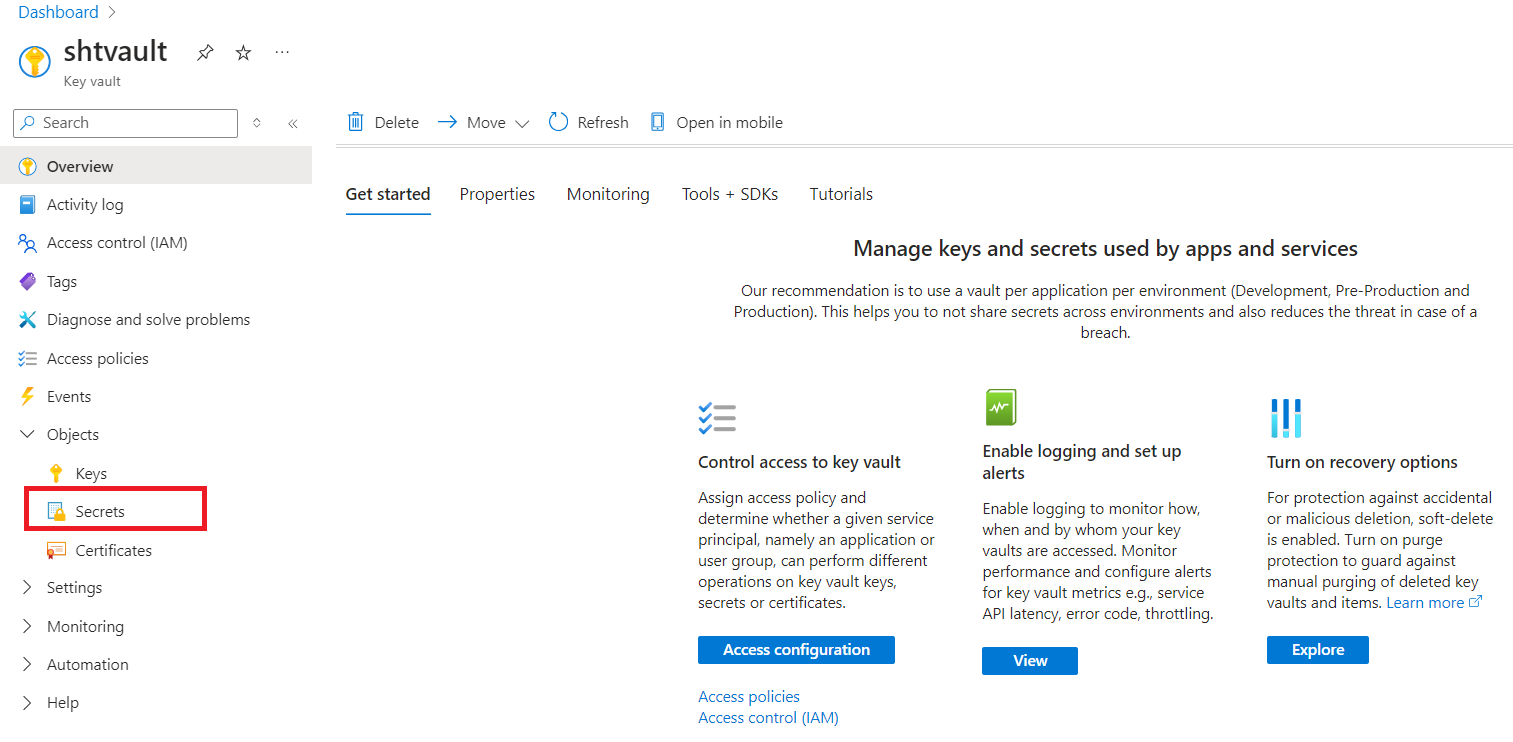


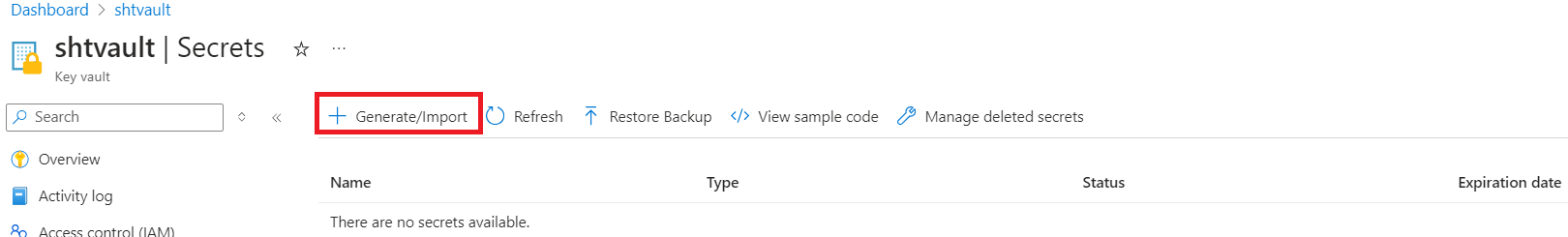
Create

A screenshot of a computer

Description automatically generated

Go to resource >





Enter the below connection string in Secret value field.

Data Source=IBM-PF3CYCH3\SQLEXPRESS; initial catalog=DBName;persist security info=True; user id=Vishwajit;password=November@1417

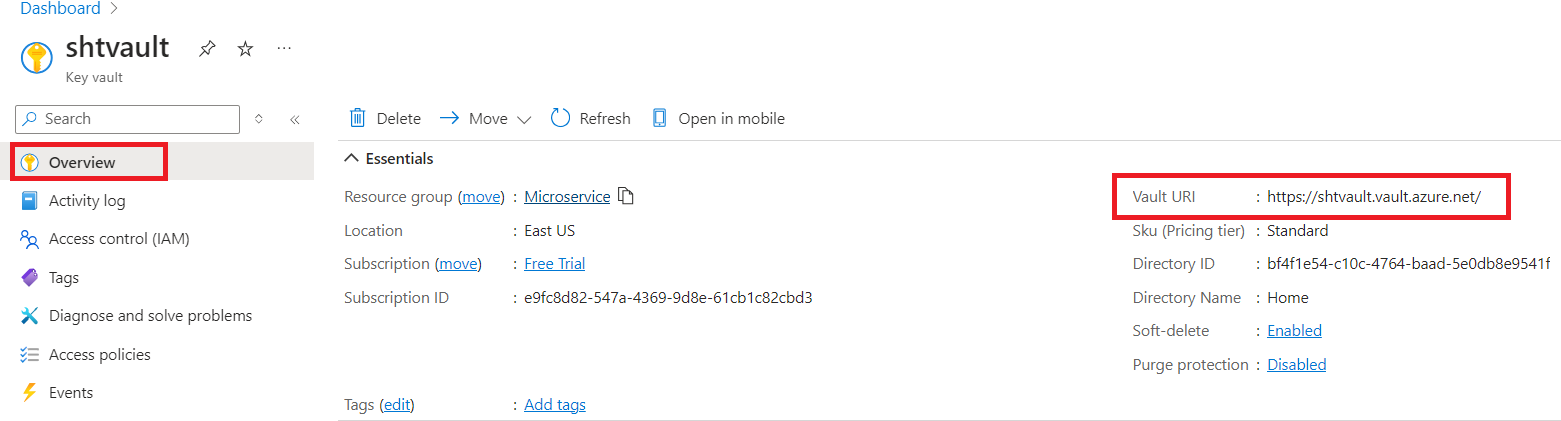
A screenshot of a computer

Description automatically generated

Create >

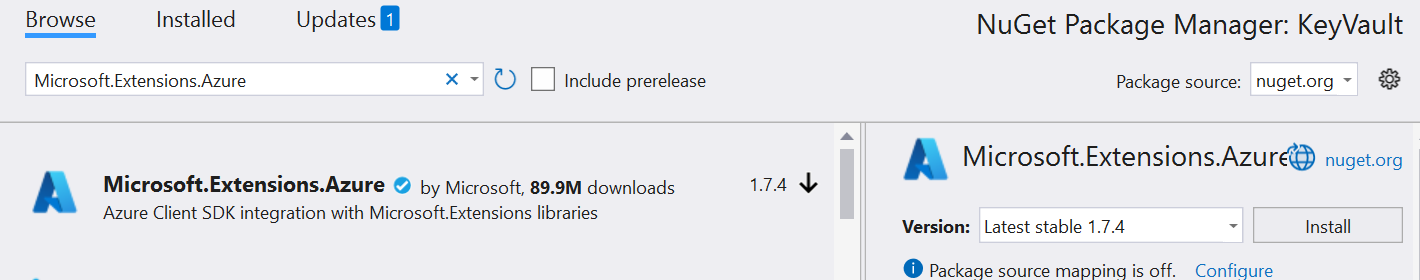
A screenshot of a computer

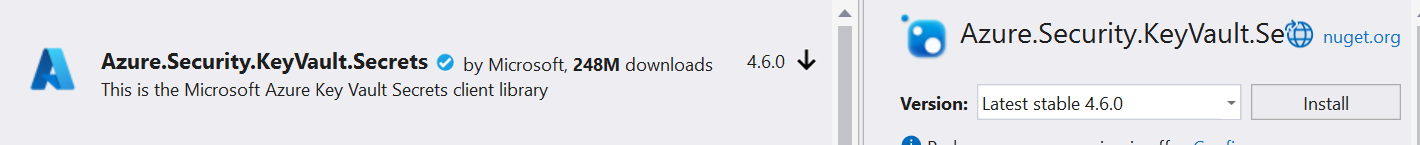
Description automatically generated



Access the above created connection string using the Vault URI.

1. Add the below NuGet packages in the project





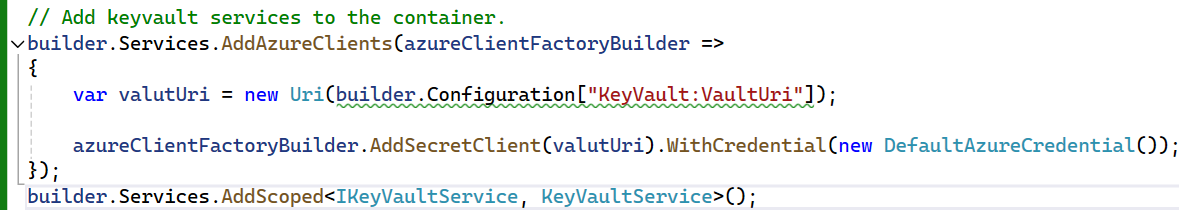
1. Create ASP.Net Core Web API project name it as KeyVault.
2. Create Services (Repositories) folder.
3. Add the KeyVault configuration in appsettings.json file of the project. Mention the Above highlighted Vault URI in appsettings.json file.

"KeyVault": {

"VaultUri": "https://shtvault.vault.azure.net/"

}

1. Add keyvault services to the container.



1. Add the VaulController.
2. Set the following environment variables values in WebApp and VM (System Environment Variables) with the Allowed/Registered App for policy at Key vault service

- AZURE\_CLIENT\_ID

- AZURE\_TENANT\_ID

- AZURE\_CLIENT\_SECRET

9. With Service endpoint we can Add/Get/Delete endpoint.

A screenshot of a application

Description automatically generated

Next: Monitor + secure >

# **2. SECTION – 2 : Service Discovery using cosul docker.**

A screenshot of a computer

Description automatically generated

1. Run the above command in command prompt.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

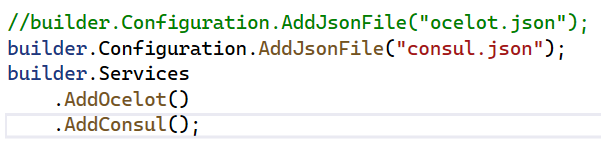
Description automatically generated

1. Add the below NuGet Package in ApiGateway.Web Service project.

A screenshot of a computer

Description automatically generated

1. Add the consul.json file inside the ApiGateway.Web Service project.
2. In Program.cs file of ApiGateway.Web service project add the below setting



1. Add the below NuGet Package in CatalogService and AuthService project.

A screenshot of a computer

Description automatically generated

1. Add the Consul configuration in AuthService of the appsetinngs.json file.

"Consul": {

"Host": "http://localhost:8500",

"ServiceName": "AuthService",

"ServiceId": "AuthService",

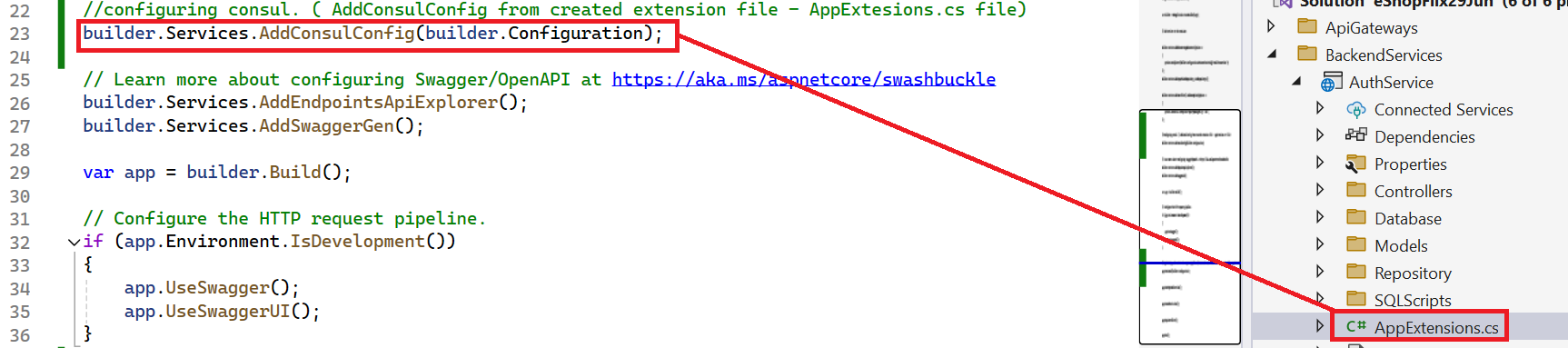
"Port": "7203",

"Address": "localhost"

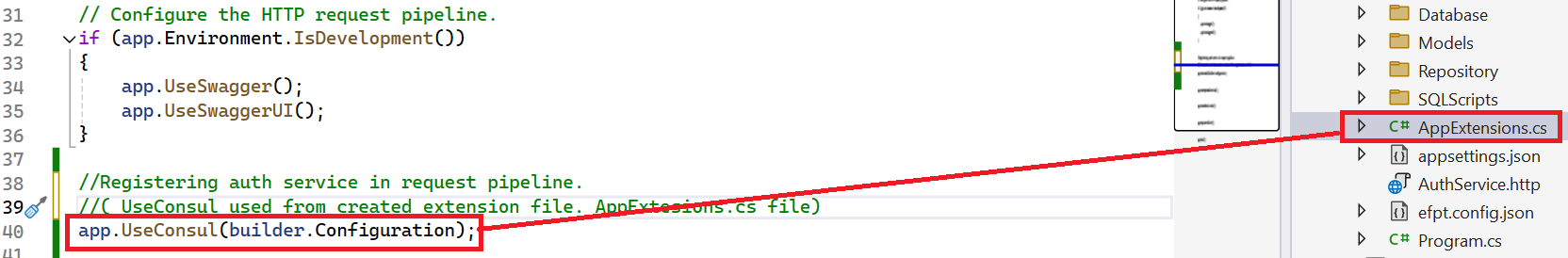
}

Where Port 7203 => AuthService local host port where AuthService is running.

1. Add the extension file in AuthService.
2. Configuring consul in Program.cs file of AuthService Project.



1. Register the UseConsul in Program.cs file of AuthService project.



1. Same steps (From steps 6-9) repeat for CatalogService.cs file.
2. Set up all the project as start up projects.
3. Check with <https://localhost:7160/auth/getall> and <https://localhost:7160/catalog/getall>

A screenshot of a computer

Description automatically generated

# **3. SECTION – 2.1 : Service Discovery using APIM.**

1. Search API Management > Create

A screenshot of a computer

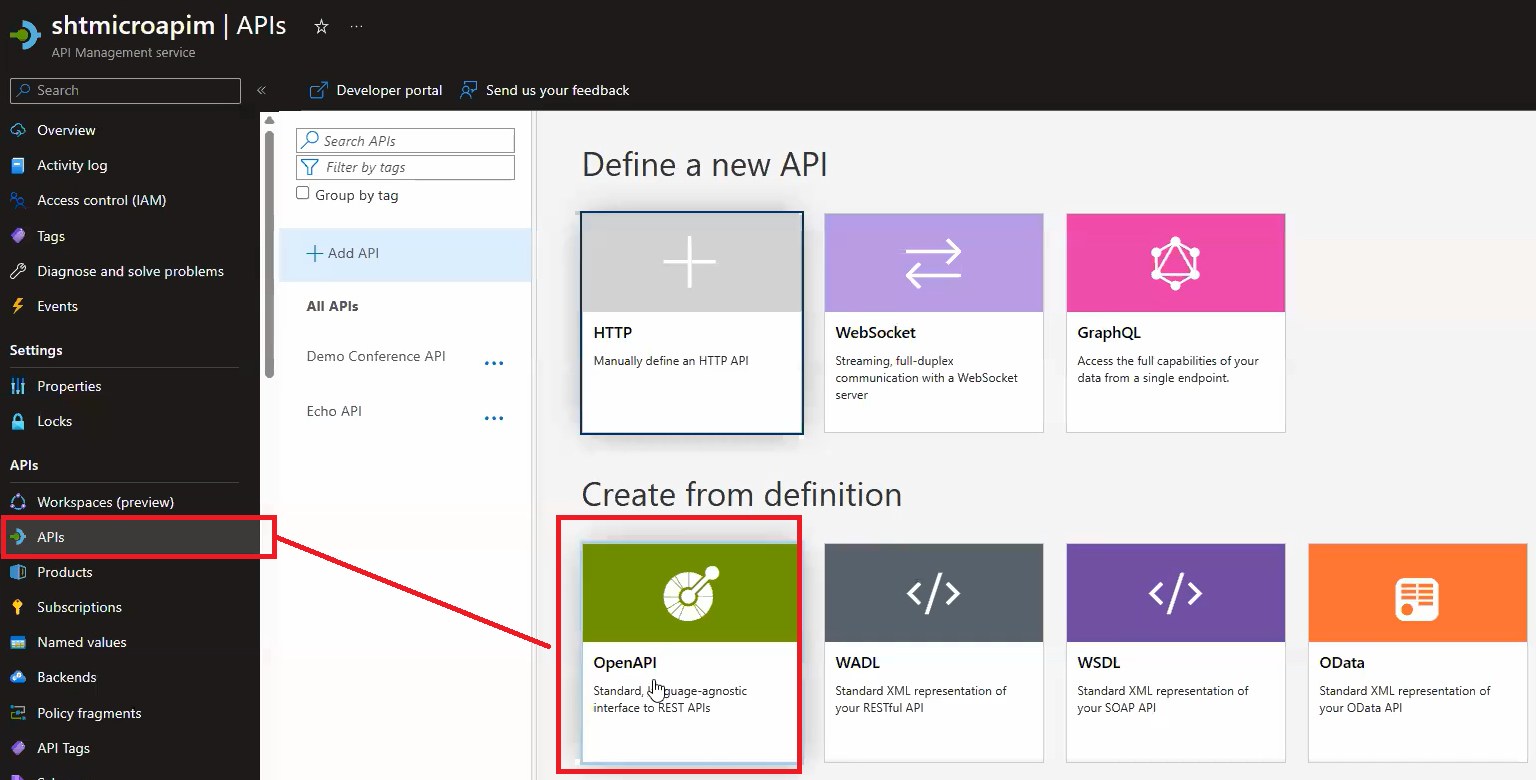
Description automatically generated

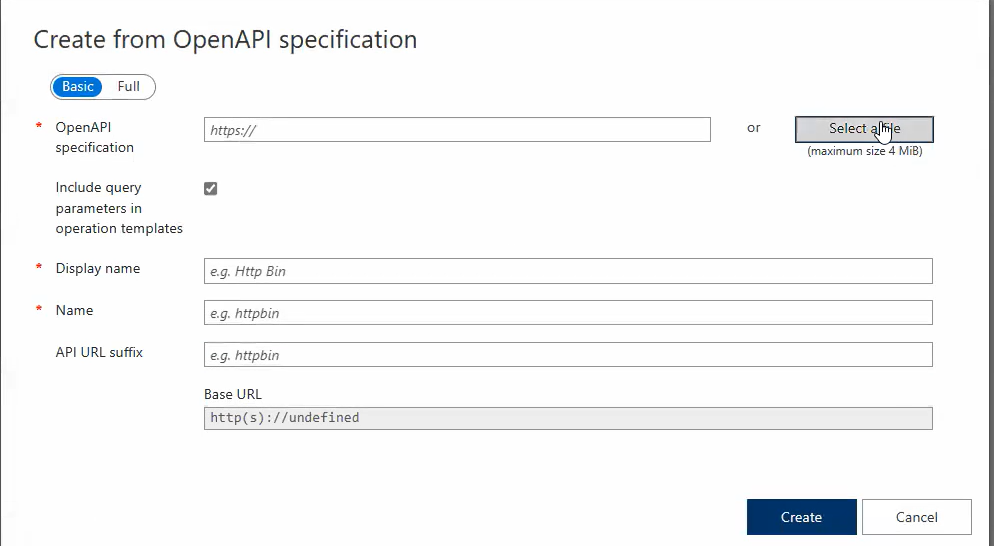
1. [conferenceapi.azurewebsites.net](https://conferenceapi.azurewebsites.net/) ( It’s a replacement of ocelot.api gateway with **Azure APIM**, that we have created in our code. (i.e ApiGateway.Web) service project.
2. Below is example of how we can do it for conferenceapi.azurewebsites.net.

A screenshot of a computer

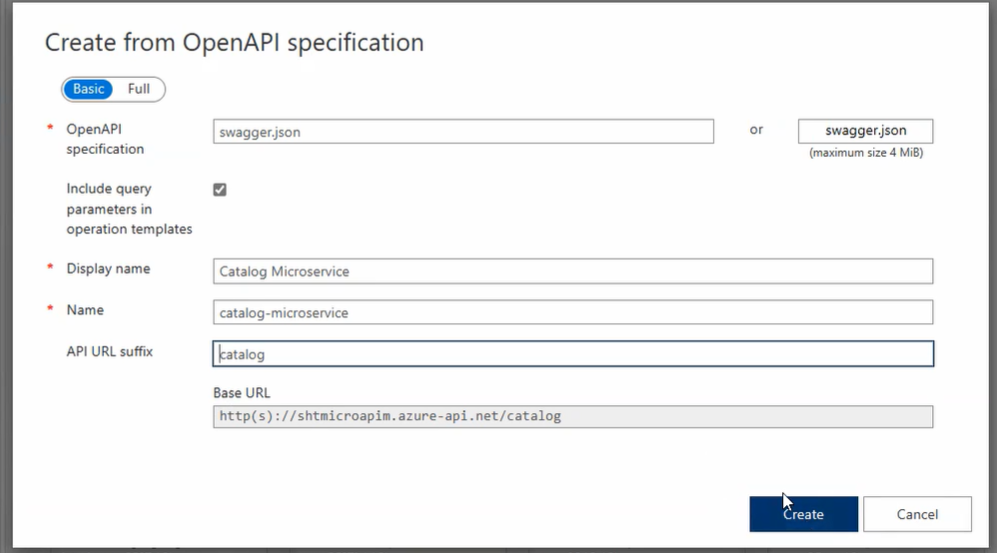
Description automatically generated

1. Similarly we have save the CatalogService swagger as swagger.json file and import in the Azure APIM as below screenshot.





Select a file > Import catalog-swagger.json file that we saved in step 3.





A screenshot of a computer

Description automatically generated

Allow IP Address. So we can test API in Postman.

A screenshot of a computer

Description automatically generated

