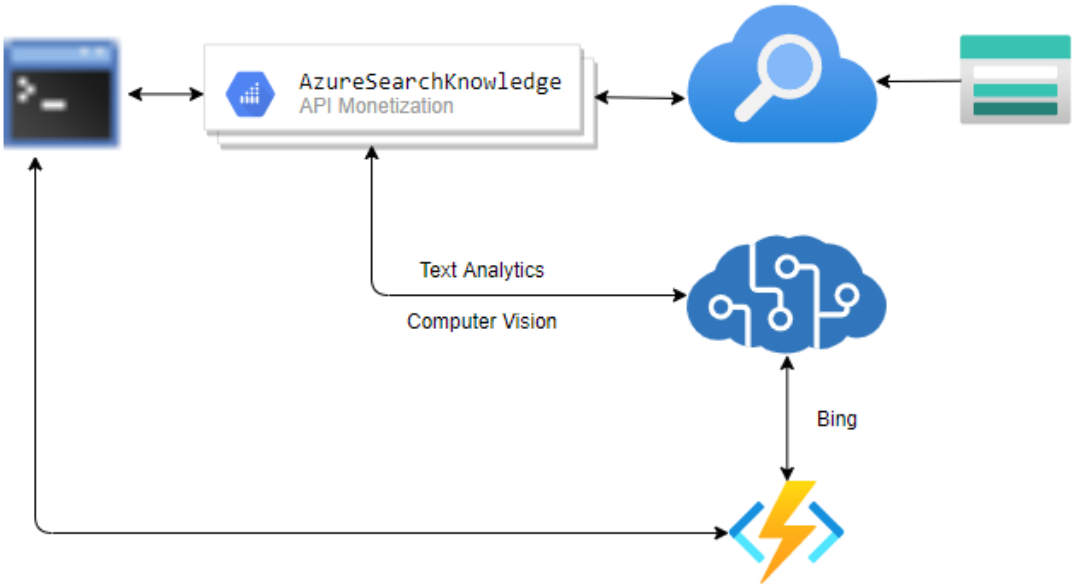
Lab 04. Use Custom skills with Azure Search using Azure Functions in VS Code

Objective

In this lab you are going to create an Azure Function to call Azure Bing and publish the function to Azure.

From your Console application (Lab 03. Use Text and Image skills with Azure Search using VS Code) you will call the Function with custom search.

Architecture



Prerequisites

- Complete **Lab 03. Use Text and Image skills with Azure Search using VS Code.**
- Postman.
- Visual Studio Code 2017 or later.
- Azure functions Core Tools.

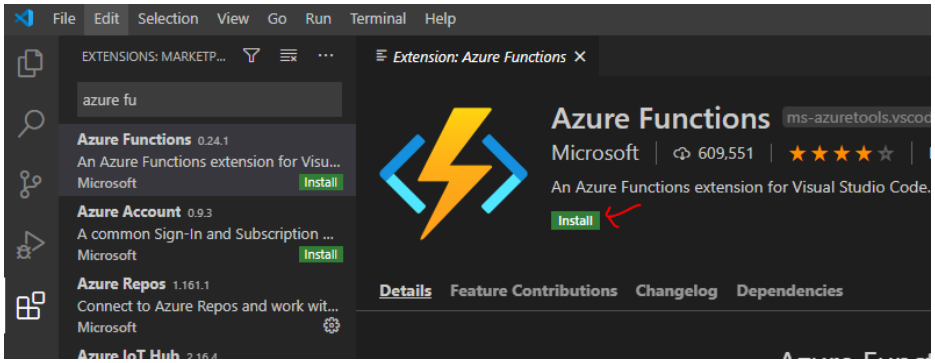
Steps

Create Azure Function

Add Azure FunctionExtension

- Open VS Code.
- Go to Extensions.

- Search Azure Functions and click on Install.

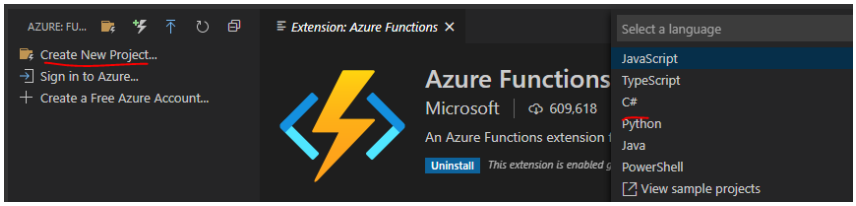


- Go to the new section added (Azure).

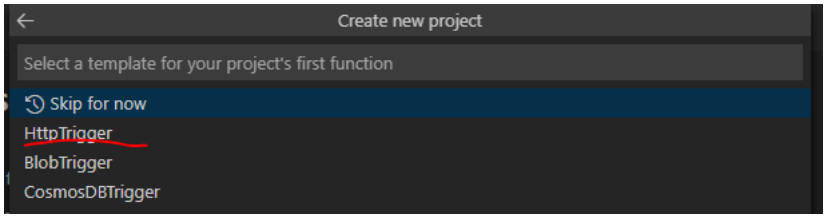
- Open target folder.
    - Go to File -> Folder.
    - Navigate to the folder created in previous steps.

Create Function using Http trigger

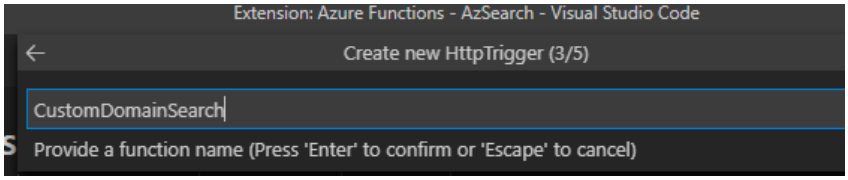
- Click on **Create New Project**.
- Select **C#** for the language type.



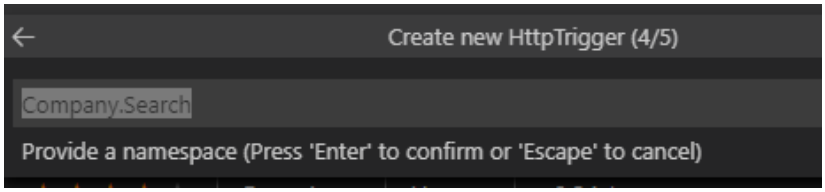
- Select **HttpTrigger** template.



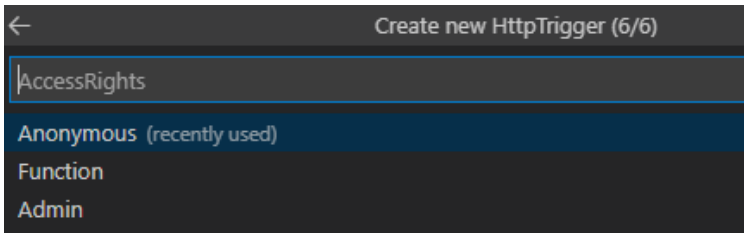
- Provide the function name (**CustomDomainSearch**) and press enter.



- Provide the namespace (**Company.Search**) and press enter.



- Select **Anonymous** for the access rights and press enter.



Create Bing search in the Function

- Remove the code within the **Run** method. After removing the piece of code indicated, it should look like:

```
using Newtonsoft.Json;

namespace Company.Search
{
    public static class CustomDomainSearch
    {
        [FunctionName("CustomDomainSearch")]
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route =
null)] HttpRequest req,
            ILogger log)
        {

        }
    }
}
```

- Add the following namespaces.

```
using System.Collections.Generic;
using System.Net;
using System.Text;
```

- After the class is opened add the credentials and search structure. Change the values between [] for the Cognitive Service created in lab 2 endpoint and key.

```
static readonly string uriBase ="[CognitiveServiceEndpoint]/bing/v7.0/search";
static readonly string accessKey = "[Cognitive Service Key]";
struct SearchResult
{
    public string jsonResult;
    public Dictionary<string, string> relevantHeaders;
}
```

- Add the following code between the **Run** method to get the request (search query) and call the method for the Azure Bing.

```
log.LogInformation("Microsoft Bing Search function: C# HTTP trigger function p
rocessed a request.");
string dataQuery = req.Query["search"];
if (dataQuery == null)
{
    return new BadRequestObjectResult("Pass a search query in the query string
.");
}
return (ActionResult)new OkObjectResult(JsonPrettyPrint(BingSearch(dataQuery).
jsonResult));
```

- After the **Run** method add the **BingSearch** method to send the request based on the search query.

```
private static SearchResult BingSearch(string searchQuery)
{
    var uriQuery = uriBase + "?q=" + Uri.EscapeDataString(searchQuery);
    WebRequest request = WebRequest.Create(uriQuery);
    request.Headers["Ocp-Apim-Subscription-Key"] = accessKey;
    HttpWebResponse response = (HttpWebResponse)request.GetResponseAsync().Res
ult;
    string json = new StreamReader(response.GetResponseStream()).ReadToEnd();
    var searchResult = new SearchResult()
    {
        jsonResult = json,
        relevantHeaders = new Dictionary<string, string>()
    };
    foreach (string header in response.Headers)
    {
        if (header.StartsWith("BingAPIs-") || header.StartsWith("XMSEdge-"))
        {
            searchResult.relevantHeaders[header] = response.Headers[header];
        }
    }
    return searchResult;
}
```

- After the method Run add the **JsonPrettyPrint** method, to print the output in an easier to read Json view.

```
static string JsonPrettyPrint(string json)
```

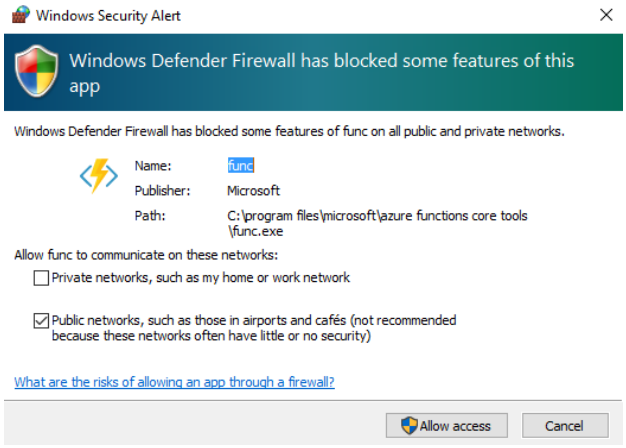
```
{
    if (string.IsNullOrEmpty(json))
    {
        return string.Empty;
    }
    json = json.Replace(Environment.NewLine, "").Replace("\t", "");
    StringBuilder sb = new StringBuilder();
    bool quote = false;
    bool ignore = false;
    char last = ' ';
    int offset = 0;
    int indentLength = 2;
    foreach (char ch in json)
    {
        switch (ch)
        {
            case '"':
                if (!ignore) quote = !quote;
                break;
            case '\\':
                if (quote && last != '\\') ignore = true;
                break;
        }
        if (quote)
        {
            sb.Append(ch);
            if (last == '\\' && ignore) ignore = false;
        }
        else
        {
            switch (ch)
            {
                case '{':
                case '[':
                    sb.Append(ch);
                    sb.Append(Environment.NewLine);
                    sb.Append(new string(' ', ++offset * indentLength));
                    break;
                case '}':
                case ']':
                    sb.Append(Environment.NewLine);
                    sb.Append(new string(' ', --offset * indentLength));
                    sb.Append(ch);
                    break;
                case ',':
                    sb.Append(ch);
                    sb.Append(Environment.NewLine);
                    sb.Append(new string(' ', offset * indentLength));
```

```
                    break;
                case ':':
                    sb.Append(ch);
                    sb.Append(' ');
                    break;
                default:
                    if (quote || ch != ' ') sb.Append(ch);
                    break;
            }
        }
        last = ch;
    }
    return sb.ToString().Trim();
}
```

- Save the document (**CTRL + S**).

Test Function using Postman

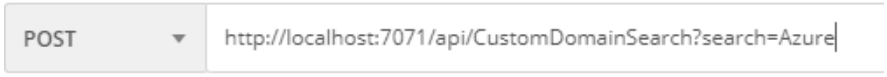
- Press **F5**.
- If asked **Allow access** to func.



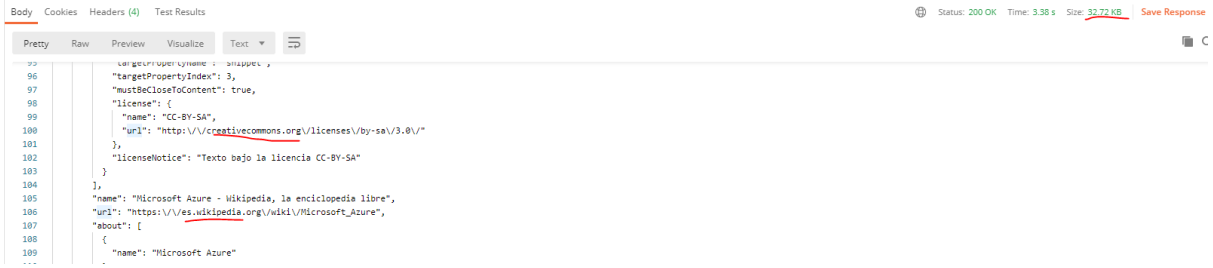
- Copy the URL of your function.

```
http://localhost:7071/api/CustomDomainSearch
```

- Open Postman and paste the URL. Select the method **POST** and add the **search** parameter (In this example is going to search for **Azure**).



- Click on Send. You will receive the Json result with all the coincidences Bing found for the query (this example using **Azure**). Also note the **size** and the **url** from the sites where it found the matches. In next exercise we are going to customize to only search in Microsoft domain.



- Go back to VS Code.
- Stop the function by clicking on the **disconnect** button.

Create Custom Search with Bing

- In the **Run** method locate the line where you call the **BingSearch** method and after the parameter add the constraint to search only in Microsoft site (+ " site:microsoft.com"). the line should look like the following code:

```
return (ActionResult)new OkObjectResult(JsonPrettyPrint(BingSearch(dataQuery +
 " site:microsoft.com").jsonResult));
```

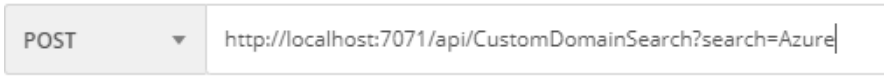
- Save the document (**CTRL + S**).

Test Function using Postman

- Press **F5**.
- Copy the URL of your function.

```
http://localhost:7071/api/CustomDomainSearch
```

- Open Postman and paste the URL. Select the method **POST** and add the **search** parameter (In this example is going to search for **Azure**).

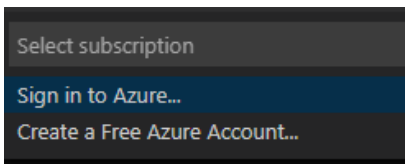


- Click on Send. You will receive the Json result with all the coincidences Bing found for the query (this example using **Azure**). Also note the **size** and the **url** from the sites where it found the matches, all should be from Microsoft.com site.

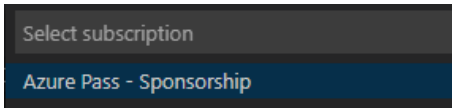
- Go back to VS Code.
- Stop the function by clicking on the **disconnect** button.

Publish Function to Azure

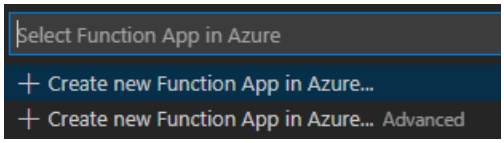
- Go to Azure menu ().
- Click on **Deploy Function App…** icon ().
- Select the option **Sign in to Azure…** and press enter.



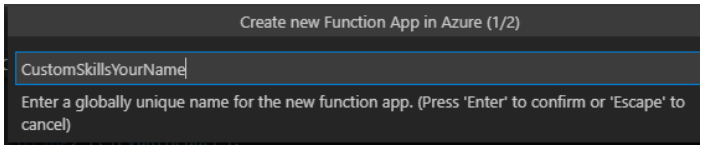
- Select your subscription and press enter.



- Select the option **+ Create new Function in Azure…** and press enter.



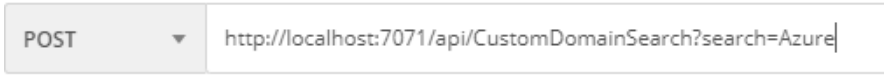
- Provide a unique name for the function (Ex. **CustomSkills[YourName]**) and press enter.



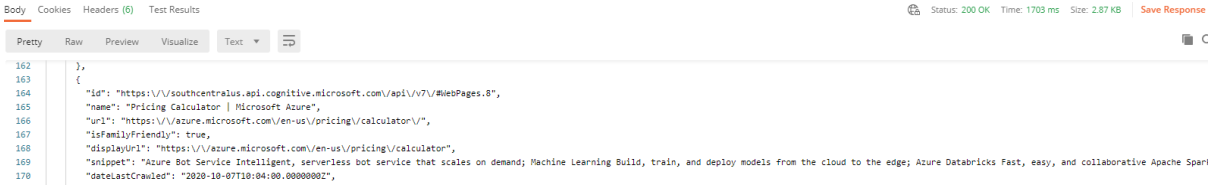
- Select the region and press enter.
- Wait for the Function App to be deployed and your function published.
- Go to Azure portal (www.portal.azure.com) and verify you have the new Function App is created.

Test published Function using Postman

- Open the Function App and go to **Functions -> Functions**, there should be your new **CustomDomainSearch** function.
- Open the **CustomDomainSearch** function.
- In the new blade, go to **Developer -> Code + Test**.
- From the top menu click on **Get function URL**.
- Copy the **URL** (https://[Your Resource Name].azurewebsites.net/api/CustomDomainSearch).
- Open Postman and paste the URL. Select the method **POST** and add the **search** parameter (In this example is going to search for **Azure**).



- Click on Send. You will receive the Json result with all the coincidences Bing found for the query (this example using **Azure**).



Call published Function from Azure Search results

In this exercise you will use Bing to search in Mirosoft.com the first organization gotten from the Azure Search index.

Integrate call from results

- Open in VS Code the console project created in **lab 3**.
- Open to program.cs file.
- Add the following namespaces

```
using System.Threading.Tasks;
using System.Net.Http;
using System.Linq;
```

- Add the HttpClient object before the Main method.

```
private static readonly HttpClient client = new HttpClient();
```

- Add the Function URL variable. You can find the URL by following the 5 first steps of the **Test published Function using Postman** task.

```
private static readonly string functionUrl = "https://[Your Resource
Name].azurewebsites.net/api/CustomDomainSearch";
```

- Add the following method to call the Azure Fuction and sent the first organization founded to search in Bing.

```
static async Task SearchOrganizationInBing(SearchResult<DocumentModel> result)
{
    var content = new FormUrlEncodedContent(new Dictionary<string, string>());
    Console.WriteLine($"metadata_storage_name: {result.Document.metadata_stora
ge_name}");
    Console.WriteLine($"Organization: {result.Document.organizations.First()}"
);

    var response = await client.PostAsync(functionUrl + "?search="+ result.Doc
ument.organizations.First(), content);

    var responseString = await response.Content.ReadAsStringAsync();
    Console.WriteLine(responseString);
}
```

- Modify the Main method to only have the next lines of code:

```
AzureCognitiveSearch ccs = new AzureCognitiveSearch();
var results = ccs.SearchingQueries("CEO");
SearchOrganizationInBing(results.First()).Wait();
```

- Save the document (**CTRL + S**).

Test Console application

- Back to the terminal run the console project.

```
dotnet run
```

- Explore the results.

NOTE: You can change the text to search and the output to print.