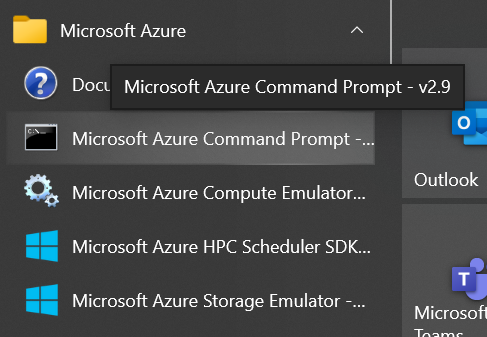
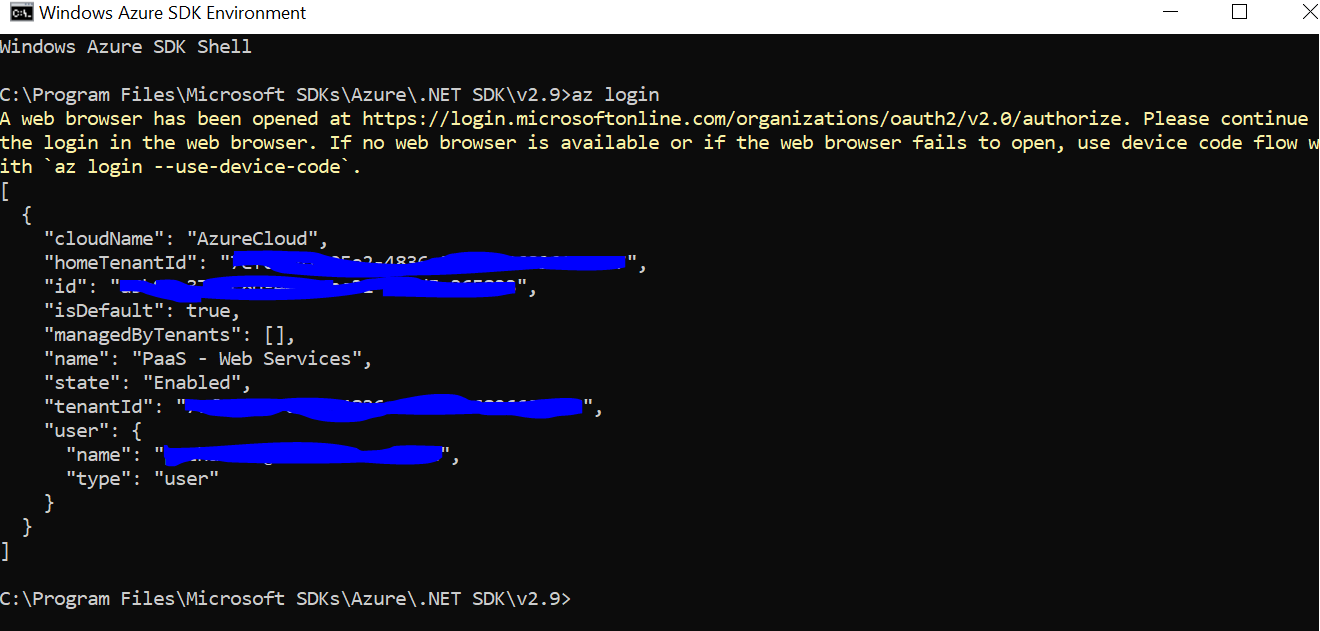
# Create Azure Function in Visual Studio

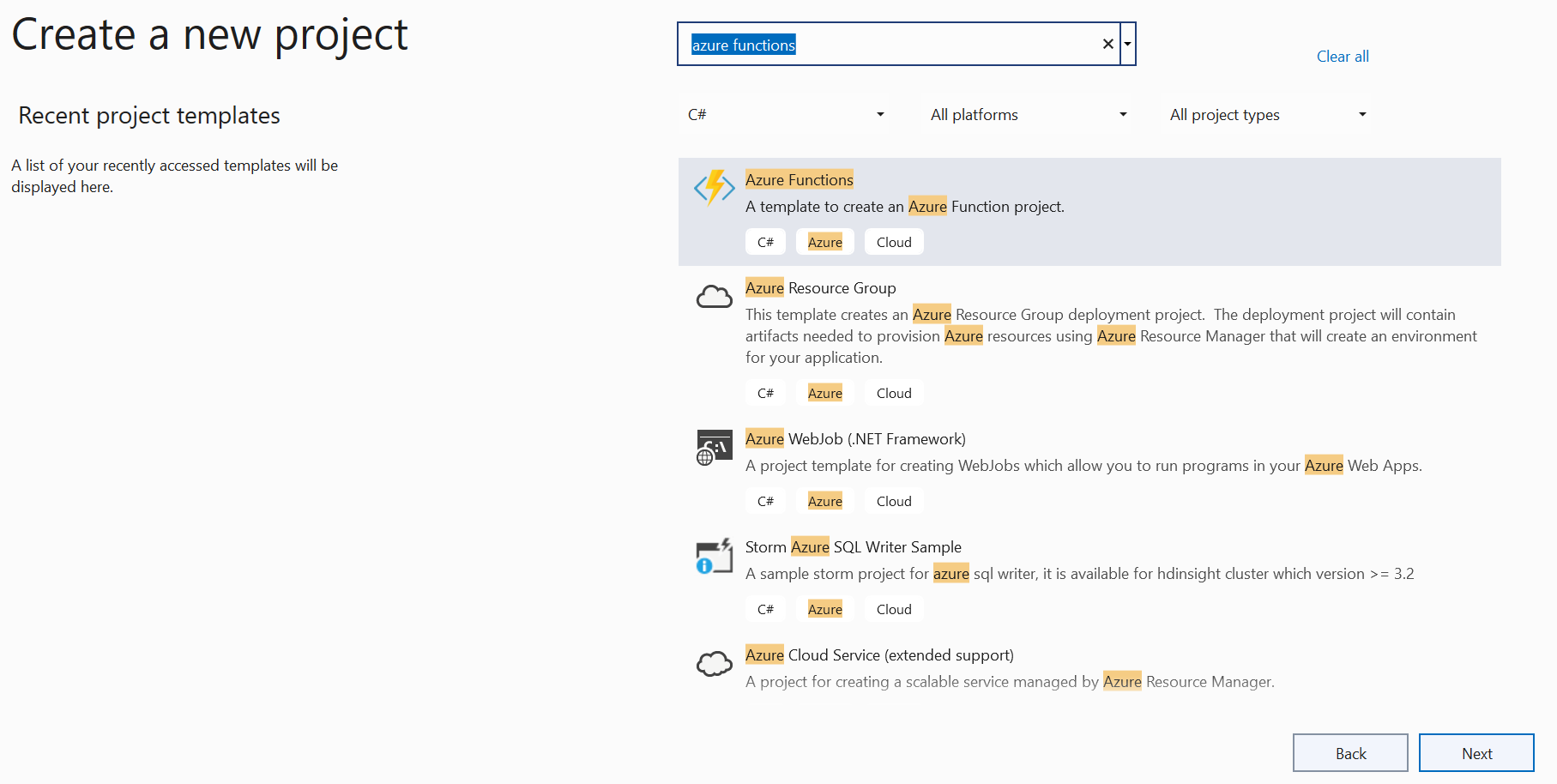
Step 1: Open Azure CLI. If you have not installed Azure CLI, you can install from link: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-windows?tabs=azure-cli>



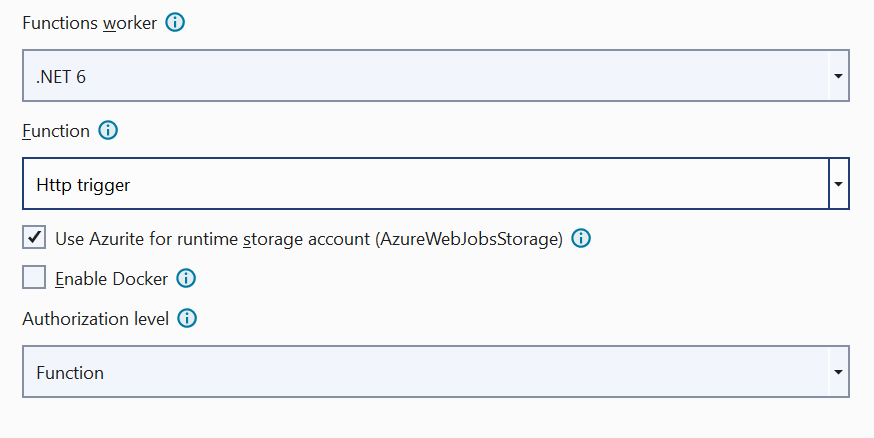
**Step 2:** Use **az login** command to create connection for your Azure account. Browser will open and you have to enter your credentials to create connection. Once connection is created successfully you will see below details in your command prompt.



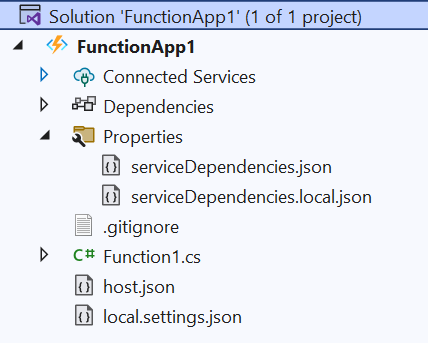
**Step 3:** Create Azure Function project in Visual Studio.



**Step 4:** Select Http Trigger, and Function Authorization.



Now project is created. The code for all the functions in a specific function app is located in a root project folder that contains a host configuration file and one or more subfolders. Each function has its own code file. The folder structure is shown in the following representation.



The **host.json** metadata file contains global configuration options that affect all functions for a function app

{

"version": "2.0",

"logging": {

"applicationInsights": {

"samplingSettings": {

"isEnabled": true,

"excludedTypes": "Request"

}

}

}

}

The static FunctionName attribute marks the method as a function entry point. Project templates often create a method named **Run**, but the method name can be **any valid C# method** name. The parameters to the Run method are an **HttpRequest** object containing the details of the request that triggered the function

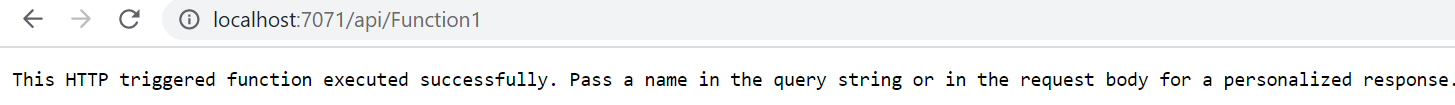
[FunctionName("Function1")]

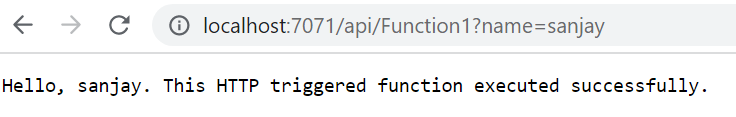
public static async Task<IActionResult> Run( [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,ILogger log)

**Step 5:** Build and Run Azure Function. Run the function app project. If prompted, accept the request from Visual Studio to download and install Azure Functions Core (CLI) tools. Since we have already done it, we will not get the message. You may also need to enable a firewall exception (you’ll get message for the same) so that the tools can handle HTTP requests. Copy the URL of your function from the Azure Functions runtime output.

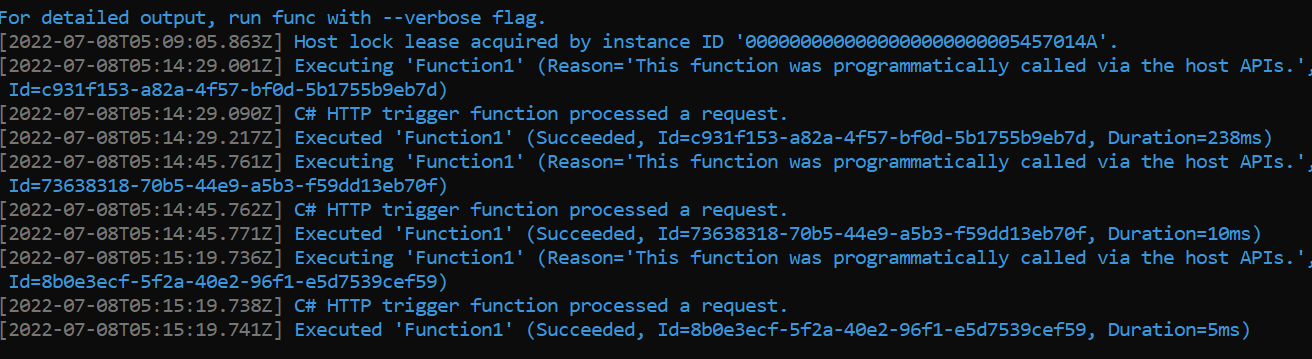


**Step 6:** Paste the URL for the HTTP request into your browser’s address bar



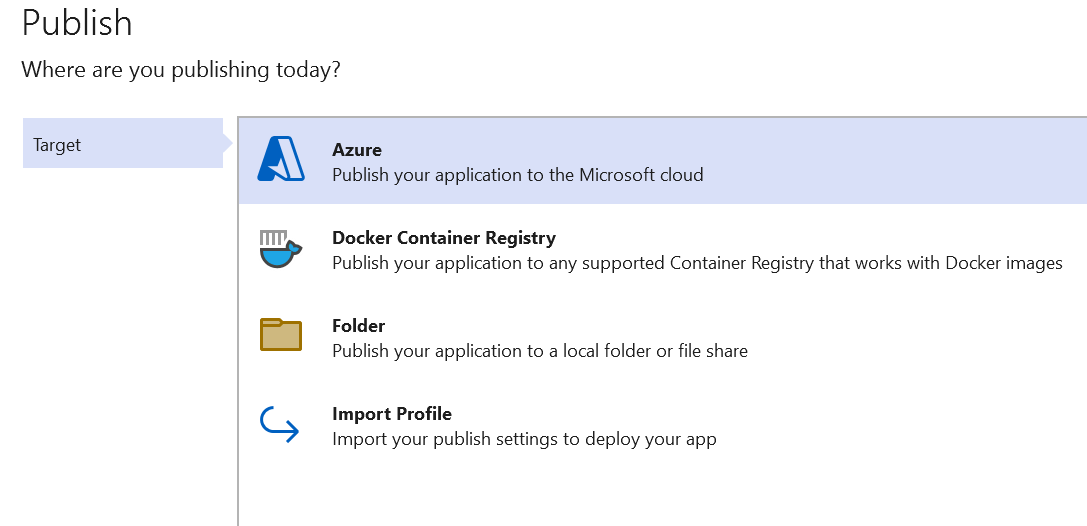


You can see the messages in console window.

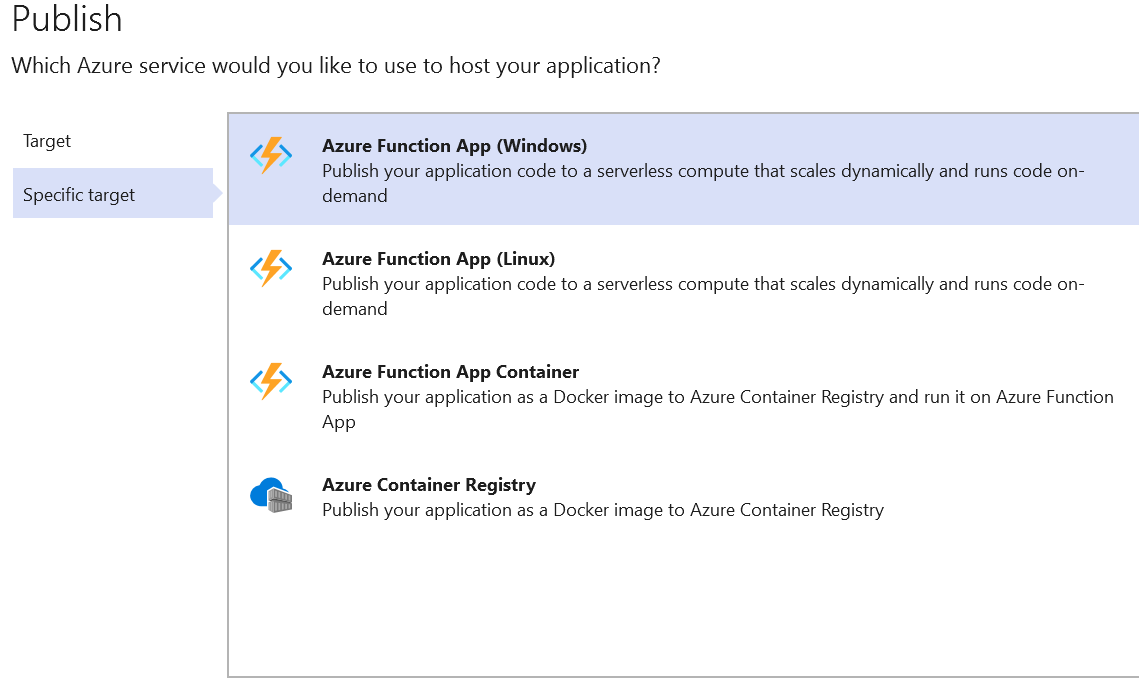


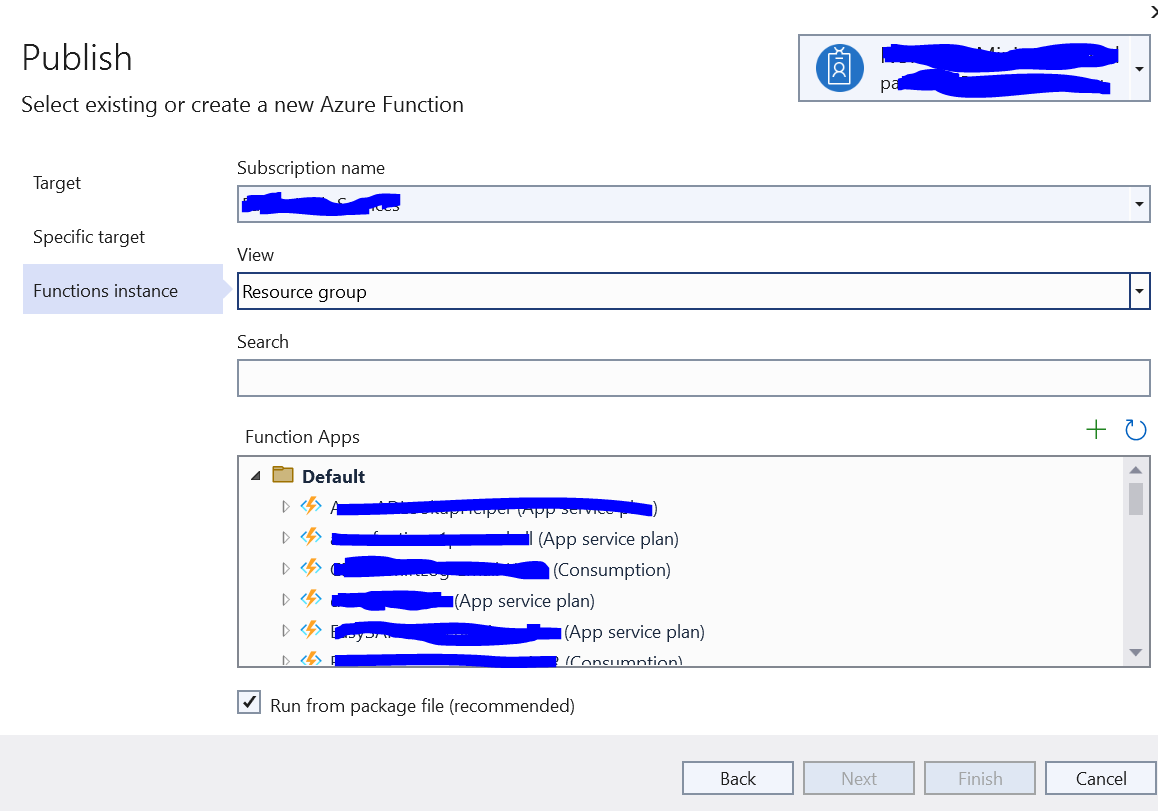
After testing you can close the console or stop debugging from VS.

**Step 7**: **Publish** the project to Azure. Before you can publish your project, you must have a **function app** in your Azure subscription. If you don't already have a function app in Azure, Visual Studio publishing creates one for you the first time you publish your project. In **Solution Explorer**, right-click the **project** and select **Publish**. In Target, select **Azure** then Next.



**Step 8**: Select **Azure Function App (Windows)**



**Step 8**: Select **Subscription** and select the **function app** where you want to publish your app.

After the deployment completes, the root URL of the function app in Azure is shown in the **Publish** tab.

**Step 9:** Verify your function in Azure. Open you **function app** in browser. This opens the root of your function app in your default web browser and displays the page that indicates your function app is running. Append the string **/api/Function1?name=Sanjay** to the base URL and run the request.

The URL that calls your trigger function is in the following format:

**http://<APP\_NAME>.azurewebsites.net/api/ Function1?name=Sanjay**

**Ref:** To check how to create Azure function in portal, please refer link: <https://www.c-sharpcorner.com/article/how-to-easily-create-azure-functions-using-azure-portal/>

<https://www.youtube.com/watch?v=M4m6gq6-nOk>