



**HERALD**  
**COLLEGE**  
K A T H M A N D U



**Module Title: Distributed and Cloud Systems  
Programming  
(5CS022)**

**Subject Title: Workshop 06**

**Student Name: Nayan Raj Khanal**

**Student Code: 2227486**

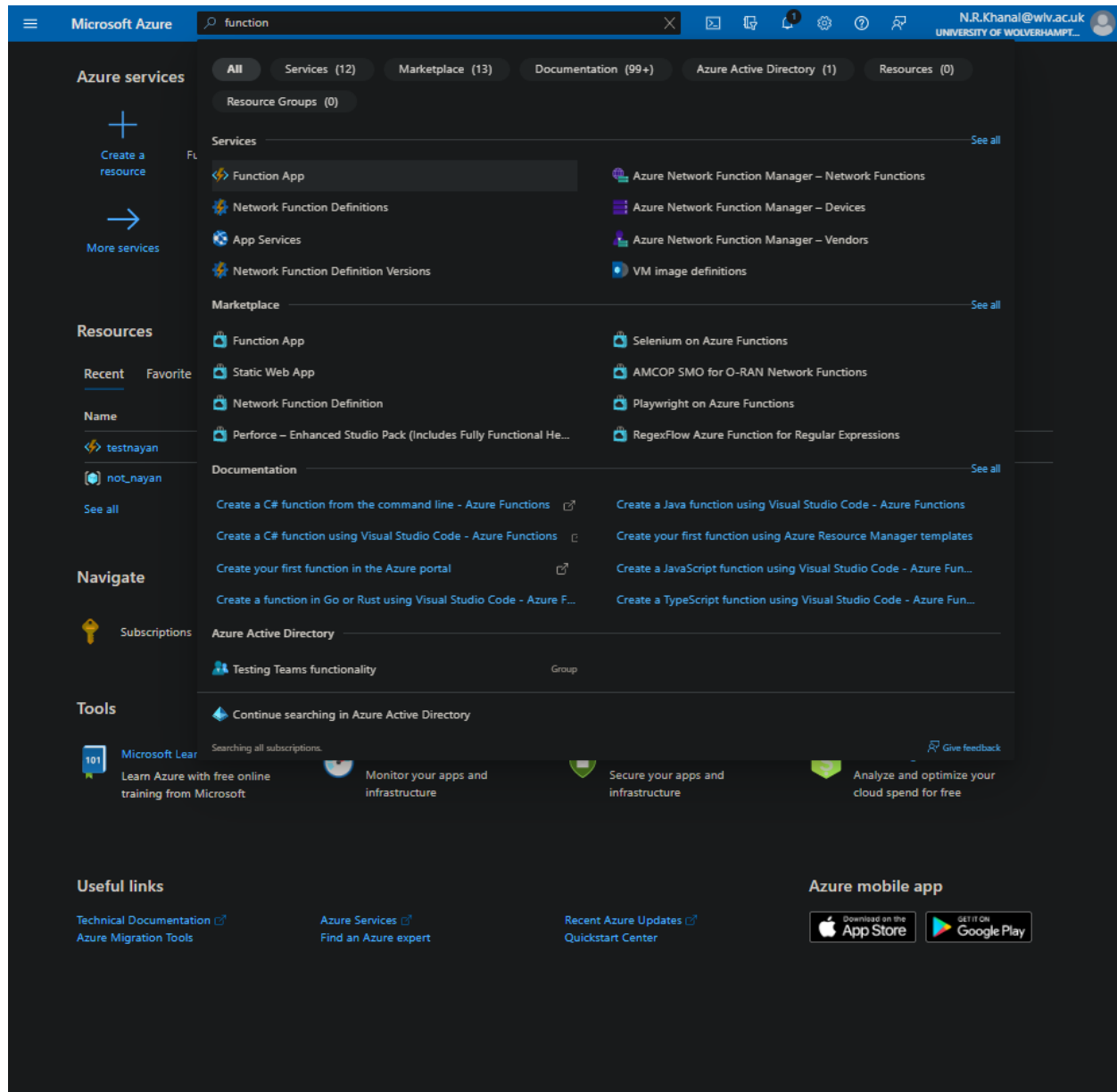
**Instructor: Mr. Prabin Sapkota**

**Submitted on: 28.04.2023**

# FUNCTION APP:

## Step 1: Creating a Function App

➔ After logging in search for “Function App” in the search bar.



➔ Now click on Create

Microsoft Azure

Search resources, services, and docs (G+)

Home >

## Function App

University of Wolverhampton (live!wlv.onmicrosoft.com)

+ Create Manage view Refresh Export to CSV Open query Assign tags

Filter for any field... Subscription equals all Resource group equals all Location equals all Add filter

Showing 1 to 1 of 1 records. No grouping List view

<input type="checkbox"/> Name ↑↓	Status ↑↓	Location ↑↓	Pricing Tier ↑↓	App Service Plan ↑↓	Subscription ↑↓	App Ty
<input type="checkbox"/> testnayan	Running	East US	Dynamic	ASP-notnayan-ac82	Azure for Students	Funcio

< Previous Page 1 of 1 Next >

Give feedback

➔ Fill up the details accordingly

NOTE:

1. The function name must be unique.
2. Choose the Stack according to your requirement.
3. Select the version, region, OS and hosting which best suits you.
4. After filling Basics Part you can create the function or change other parts as your liking

Microsoft Azure

Search resources, services, and docs (G+)

N.R.Khanal@wlv.ac.uk  
UNIVERSITY OF WOLVERHAMPTON

Home > Function App >

## Create Function App

Basics Storage Networking Monitoring Deployment Tags Review + create

Create a function app, which lets you group functions as a logical unit for easier management, deployment and sharing of resources. Functions lets you execute your code in a serverless environment without having to first create a VM or publish a web application.

**Project Details**

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

Subscription \* ☐ Azure for Students

Resource Group \* ☐ (New) not\_nayan [Create new](#)

**Instance Details**

Function App name \* testnayan ☒ .azurewebsites.net

Do you want to deploy code or container image? \* ☒ Code ☐ Container Image

Runtime stack \* .NET

Version \* 6 (LTS)

Region \* East US

**Operating system**

The Operating System has been recommended for you based on your selection of runtime stack.

Operating System \* ☐ Linux ☒ Windows

**Hosting**

The plan you choose dictates how your app scales, what features are enabled, and how it is priced. [Learn more](#)

Hosting options and plans \* ☒ Consumption (Serverless)  
Optimized for serverless and event-driven workloads.

☐ Functions Premium  
Event based scaling and network isolation, ideal for workloads running continuously.

☐ App service plan  
Fully isolated and dedicated environment suitable for workloads that need large SKUs or need to co-locate Web Apps and Functions.

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

➔ Click on next and edit storage

The screenshot shows the 'Create Function App' wizard in the Microsoft Azure portal. The 'Storage' tab is selected, showing instructions on linking a storage account and a dropdown menu for selecting an account. The account name '(New) notnayana5bc' is visible in the dropdown. Navigation buttons at the bottom include 'Review + create', '< Previous', and 'Next : Networking >'.

Microsoft Azure

Search resources, services, and docs (G+/J)

Home > Function App >

## Create Function App

Basics **Storage** Networking Monitoring Deployment Tags Review + create

### Storage

When creating a function app, you must create or link to a general-purpose Azure Storage account that supports Blobs, Queue, and Table storage.

Storage account \*

(New) notnayana5bc







Create new

Review + create < Previous Next : Networking >

➔ Click on next and edit Networking

Microsoft Azure

Search resources, services, and docs (G+/)



N.R.Khanal@wlw.ac.uk  
UNIVERSITY OF WOLVERHAMPT...

Home > Function App >

Create Function App

BasicsStorageNetworkingMonitoringDeploymentTagsReview + create

Function Apps can be provisioned with the inbound address being public to the internet or isolated to an Azure virtual network. Function Apps can also be provisioned with outbound traffic able to reach endpoints in a virtual network, be governed by network security groups or affected by virtual network routes. By default, your app is open to the internet and cannot reach into a virtual network. These aspects can also be changed after the app is provisioned. [Learn more](#)

Enable public access \* ☒ On ☐ Off

⚠ Network injection is only available in Functions Premium and Basic, Standard, Premium, Premium V2, Premium V3 Dedicated App Service plans.

Enable network injection ☐ On ☒ Off

Review + create







< Previous

Next : Monitoring >

➔ Click on next and edit Monitoring

Microsoft Azure

Search resources, services, and docs (G+/J)



N.R.Khanal@wlv.ac.uk  
UNIVERSITY OF WOLVERHAMPTON

Home > Function App >

Create Function App

BasicsStorageNetworkingMonitoringDeploymentTagsReview + create


Azure Monitor application insights is an Application Performance Management (APM) service for developers and DevOps professionals. Enable it below to automatically monitor your application. It will detect performance anomalies, and includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. Your bill is based on amount of data used by Application Insights and your data retention settings. [Learn more](#)

[App Insights pricing](#)

**Application Insights**

Enable Application Insights \* ☐ No ☒ Yes

Application Insights \* 

(New) testnayan (East US) 

  
[Create new](#)

Region 

East US

Review + create







< Previous

Next : Deployment >

➔ Click on next and edit Deployment

Microsoft Azure

Search resources, services, and docs (G+/J)



N.R.Khanal@wlv.ac.uk  
UNIVERSITY OF WOLVERHAMPT...

Home > Function App >

Create Function App

BasicsStorageNetworkingMonitoringDeploymentTagsReview + create

**Enable GitHub Actions to continuously deploy your app.** GitHub Actions is an automation framework that can build, test, and deploy your app whenever a new commit is made in your repository. If your code is in GitHub, choose your repository here and we will add a workflow file to automatically deploy your app to App Service. If your code is not in GitHub, go to the Deployment Center once the web app is created to set up your deployment. [Learn more](#)

GitHub Actions settings

Continuous deployment ☒ Disable ☐ Enable

GitHub Actions details

Select your GitHub details, so Azure Web Apps can access your repository. You must have write access to your chosen repository to deploy with GitHub Actions.

GitHub account

Authorize

Organization

Select organization

Repository


Select repository

Branch

Select branch

Workflow configuration

File with the GitHub Actions workflow configuration.

 Complete the Basics tab and the form above to preview the GitHub Actions workflow file.

Preview file

Review + create

< Previous







Next : Tags >



➔ Click on next and edit Tags

Microsoft Azure

Search resources, services, and docs (G+/J)



N.R.Khanal@wlv.ac.uk  
UNIVERSITY OF WOLVERHAMPT...

Home > Function App >

Create Function App

BasicsStorageNetworkingMonitoringDeploymentTagsReview + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups.

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name	Value	Resource
	:	4 selected

Review + create

< Previous

Next : Review + create >

➔ Lastly, click on next and review your app before creating


Microsoft Azure Search resources, services, and docs (G+/)

Home > Function App >

## Create Function App

Basics Storage Networking Monitoring Deployment Tags Review + create

Summary

 **Function App**  
by Microsoft

**Details**

Subscription	32812db4-95b9-47b5-90b6-a4de0d650ff2
Resource Group	not_nayan
Name	testnayan
Runtime stack	.NET 6 (LTS)

**Hosting**

**Storage (New)**

Storage account	notnayana5bc
-----------------	--------------

**Plan (New)**

Hosting options and plans	Consumption (Serverless)
Name	ASP-notnayan-ac82
Operating System	Windows
Region	East US
SKU	Dynamic

**Monitoring (New)**

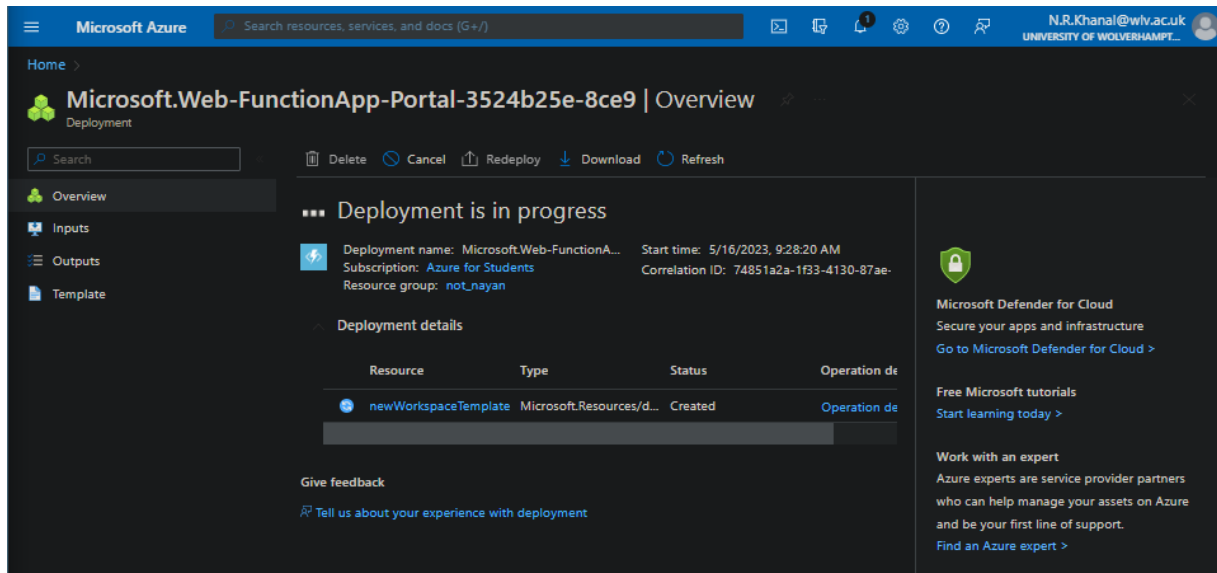
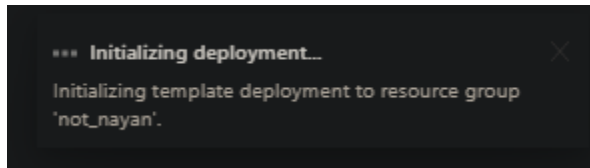
Application Insights	Enabled
Name	testnayan
Region	East US

**Deployment**

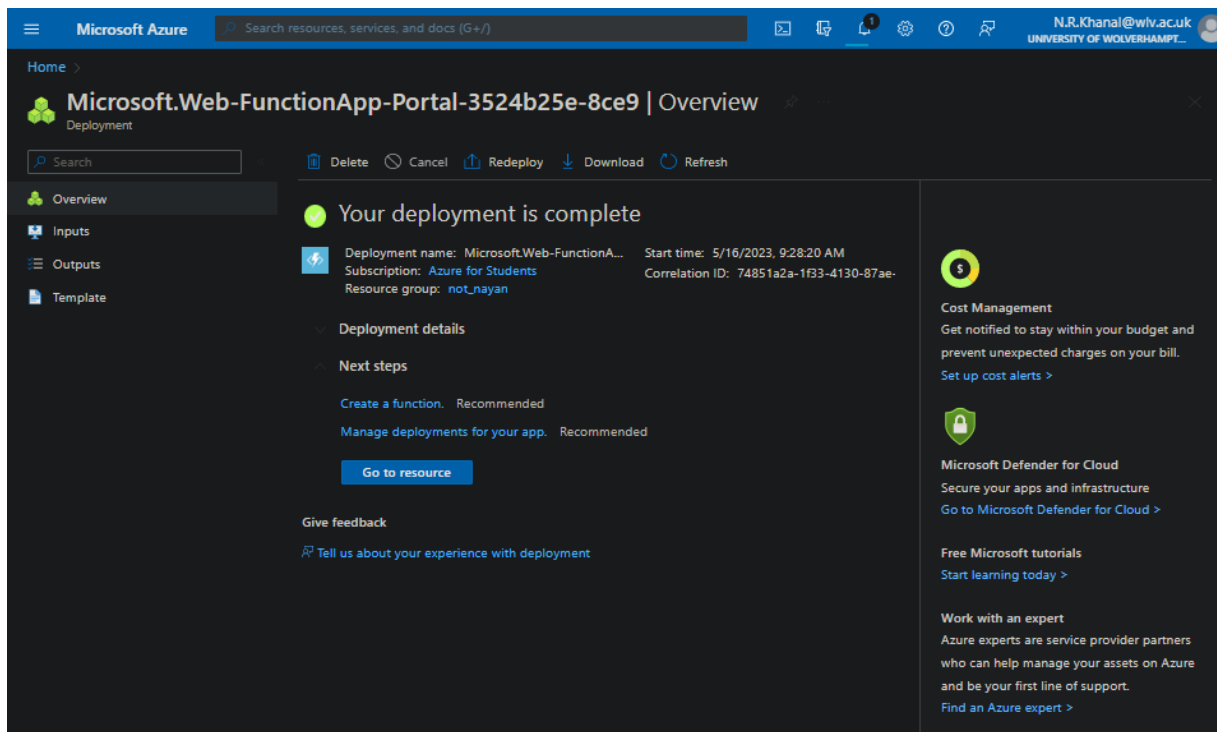
Continuous deployment	Not enabled / Set up after app creation
-----------------------	---

[Create](#) [< Previous](#) [Next >](#) [Download a template for automation](#)

➔ After clicking create it will take some time to deploy, please be patient.



Once deployed you'll get this screen:



And if you click on “Go to resource”:

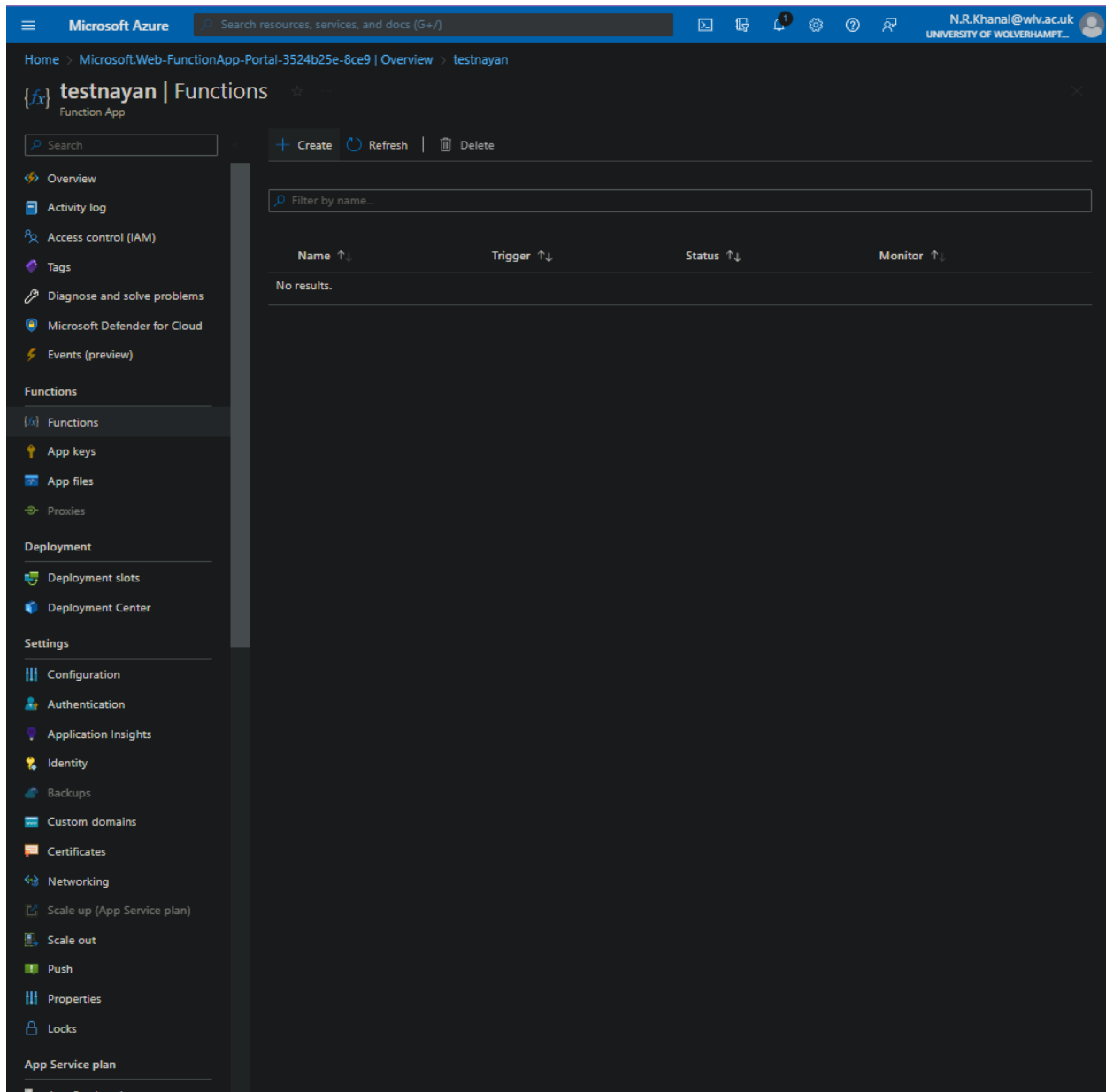
The screenshot displays the Microsoft Azure portal interface for a Function App named 'testnayan'. The top navigation bar includes the Microsoft Azure logo, a search bar, and the user profile 'N.R.Khanal@wlv.ac.uk' from the University of Wolverhampton. The breadcrumb trail shows 'Home > Microsoft.Web-FunctionApp-Portal-3524b25e-8ce9 | Overview >'. The left sidebar contains a search bar and a list of navigation items: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Functions, App keys, App files, Proxies, Deployment, Deployment slots, Deployment Center, Settings, Configuration, Authentication, Application Insights, Identity, Backups, Custom domains, Certificates, Networking, Scale up (App Service plan), Scale out, Push, Properties, Locks, and App Service plan. The main content area features a 'testnayan' Function App header with a search bar and action buttons: Browse, Refresh, Stop, Restart, Swap, Get publish profile, Reset publish profile, and Download app content. A notification banner states: 'Click here to access Application Insights for monitoring and profiling for your app.' Below this is the 'Essentials' section, which includes a 'JSON View' toggle and a table of key properties:

Property	Value
Resource group	<a href="#">not_nayan</a>
Status	Running
Location	East US
Subscription	<a href="#">Azure for Students</a>
Subscription ID	32812db4-95b9-47b5-90b6-a4de0d650ff2
Tags	<a href="#">Click here to add tags</a>
URL	<a href="https://testnayan.azurewebsites.net">https://testnayan.azurewebsites.net</a>
Operating System	Windows
App Service Plan	<a href="#">ASP-notnayan-ac82 (Y1-0)</a>
Properties	<a href="#">See More</a>
Runtime version	4.19.0.20414

Below the Essentials section are tabs for Metrics, Features (9), Notifications (1), and Quickstart. The Metrics section is active, showing three charts: 'Memory working set' (0 MB), 'Function Execution Count' (0), and 'MB Milliseconds' (0). Each chart has a y-axis scale and a time range from 8:45 AM to 9 AM UTC+05:45.

## STEP 2: Creating an HTTP triggered based Azure Function

➔ To add a function, select Function tab from the left side and click on create



➔ As the function is going to be HTTP trigger function, we select HTTP trigger and click on create

The screenshot shows the Microsoft Azure portal interface for creating a new function. The left sidebar contains navigation links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Functions, App keys, App files, Proxies, Deployment, Deployment slots, Deployment Center, Settings, Configuration, Authentication, Application Insights, Identity, Backups, Custom domains, Certificates, Networking, Scale up (App Service plan), Scale out, Push, Properties, Locks, and App Service plan. The main content area is titled 'Create function' and includes a search bar, a 'Filter' button, and a table of templates. The 'HTTP trigger' template is selected, and the 'New Function' field is set to 'HttpTrigger1'. The 'Authorization level' is set to 'Function'.

**Microsoft Azure** Search resources, services, and docs (G+/I) N.R.Khanal@wlv.ac.uk UNIVERSITY OF WOLVERHAMPT...

Home > Microsoft.Web-FunctionApp-Portal-3524b2 **Create function**

**testnayan | Functions** Function App

Search + Create

Filter

**Select development environment**  
Instructions will vary based on your development environment. [Learn more](#)

Development environ... Develop in portal

**Select a template**  
Use a template to create a function. Triggers describe the type of events that invoke your functions. [Learn more](#)

Filter

Template	Description
HTTP trigger	A function that will be run whenever it receives an HTTP request, responding based on data in the body or query string
Timer trigger	A function that will be run on a specified schedule
Azure Queue Storage trigger	A function that will be run whenever a message is added to a specified Azure Storage queue
Azure Service Bus Queue trigger	A function that will be run whenever a message is added to a specified Service Bus queue
Azure Service Bus Topic trigger	A function that will be run whenever a message is added to the specified Service Bus topic
Azure Blob Storage trigger	A function that will be run whenever a blob is added to a specified container
Azure Event Hub trigger	A function that will be run whenever an event hub receives a new event

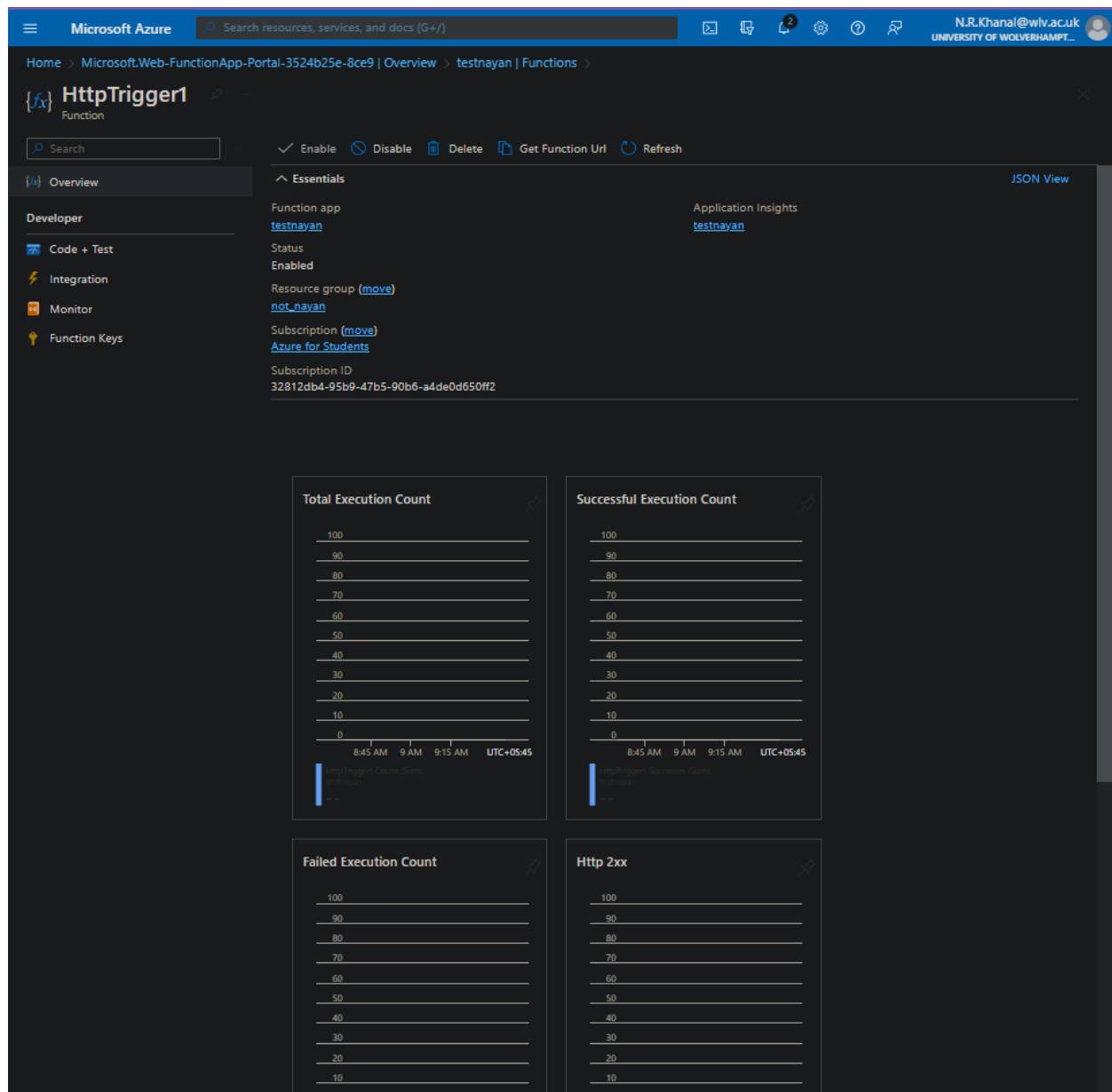
**Template details**  
We need more information to create the HTTP trigger function. [Learn more](#)

**New Function \*** HttpTrigger1

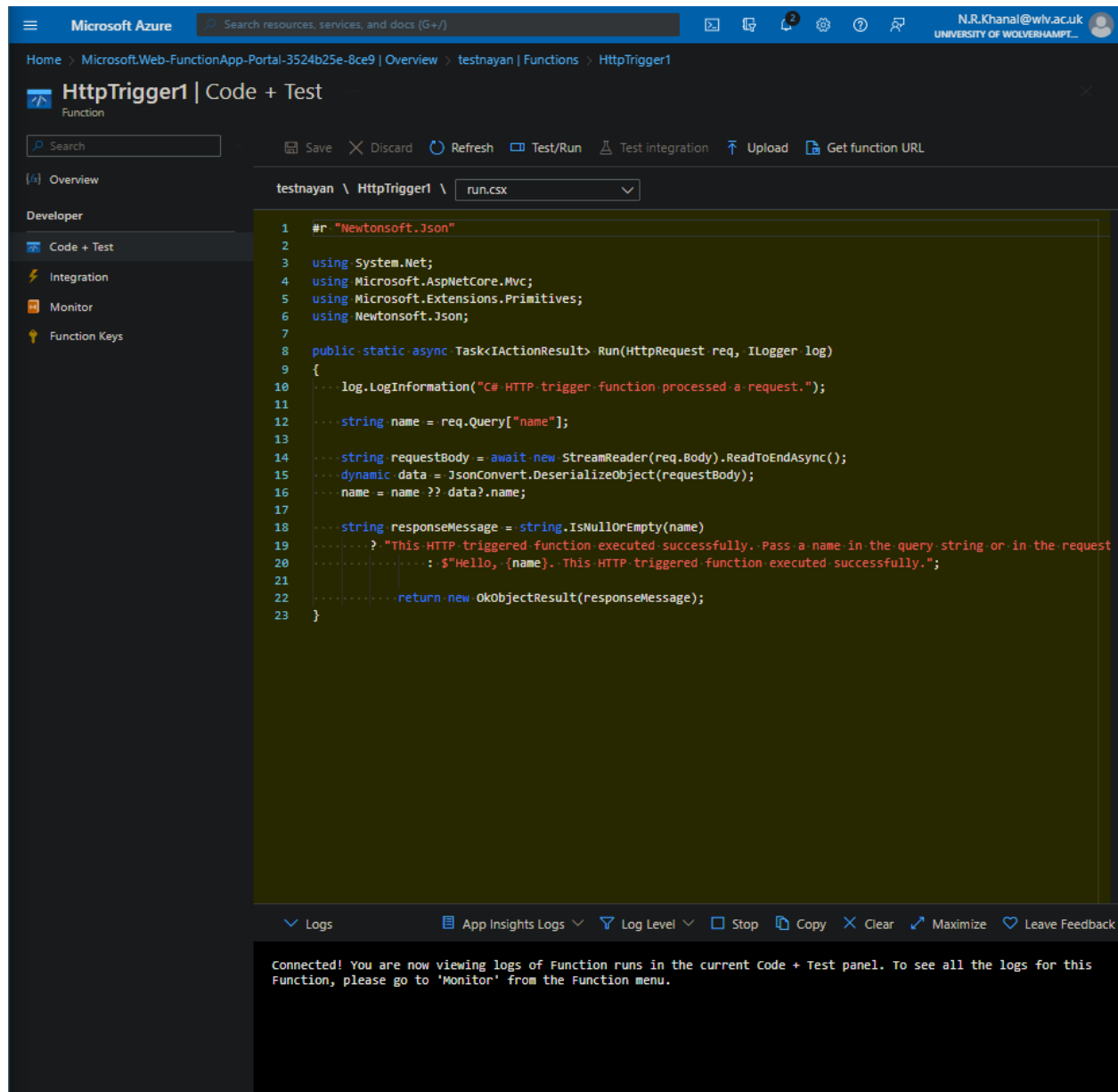
**Authorization level \*** Function

Create Cancel

➔ After sometime, you'll see this interface which means the function was created successfully



➔ Click on “Code + Test” from the left side panel. Here, we can write codes and test the function.



The screenshot displays the Microsoft Azure portal interface for a function named 'HttpTrigger1'. The left-hand navigation pane is open, showing the 'Code + Test' tab selected under the 'Developer' section. The main area shows the C# code for the function, which is a simple HTTP trigger that logs the request and returns a JSON response. The code is as follows:

```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task<IActionResult> Run(HttpRequest req, ILogger log)
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = req.Query["name"];
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17
18     string responseMessage = string.IsNullOrEmpty(name)
19         ? "This HTTP triggered function executed successfully. Pass a name in the query string or in the request body."
20         : $"Hello, {name}. This HTTP triggered function executed successfully.";
21
22     return new OkObjectResult(responseMessage);
23 }
```

Below the code editor, there is a 'Logs' section with a message: 'Connected! You are now viewing logs of Function runs in the current Code + Test panel. To see all the logs for this Function, please go to 'Monitor' from the Function menu.'



### STEP 3: Testing of Http triggered based Azure Function

➔ Click on Test/Run and enter your name in the Body Section and click run

The screenshot displays the Microsoft Azure portal interface for testing an Azure Function. The top navigation bar shows the user is logged in as N.R.Khanal@wlv.ac.uk. The breadcrumb trail indicates the path: Home > Microsoft.Web-FunctionApp-Portal-3524b25e-8ce9 | Overview > testnayan | Functions > HttpTrigger1.

The main content area is titled 'HttpTrigger1 | Code + Test'. On the left, a sidebar shows the 'Developer' menu with options: Overview, Code + Test (selected), Integration, Monitor, and Function Keys. The central pane shows the C# code for 'run.csx' in the 'testnayan \ HttpTrigger1 \ ' directory. The code is as follows:

```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Logging;
6 using Newtonsoft.Json;
7
8 public static async Task<HttpResponseMessage> Run(HttpRequest req, ILogger log)
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = req.Query["name"];
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17
18     string responseMessage = $"This HTTP trigger processed a request for {name}";
19
20     return new HttpResponseMessage(HttpStatusCode.OK)
21     {
22         Content = new StringContent(responseMessage),
23     };
24 }
```

On the right, the 'Test' panel is active, showing the 'Input' tab. It includes a dropdown for 'HTTP method' set to 'POST', a dropdown for 'Key' set to 'master (Host key)', and sections for 'Query', 'Headers', and 'Body'. The 'Body' section contains the JSON input: { "name": "Nayan" }. At the bottom of the 'Test' panel, there are 'Run' and 'Close' buttons.

At the bottom of the main content area, there is a message: 'Connected! You are now viewing logs of Function runs in the current Code + Test panel. To see all the logs for this Function, please go to 'Monitor' from the Function menu.'

➔ If everything is correct it returns response code 200 as the output code and the content should have your name.

The screenshot displays the Microsoft Azure portal interface for a function named 'HttpTrigger1'. The top navigation bar shows the user is logged in as N.R.Khanal@wlv.ac.uk. The breadcrumb trail indicates the path: Home > Microsoft.Web-FunctionApp-Portal-3524b25e-8ce9 > Overview > testnayan > Functions > HttpTrigger1.

The main content area is titled 'HttpTrigger1 | Code + Test'. On the left, a sidebar shows the 'Developer' section with options like 'Overview', 'Code + Test' (selected), 'Integration', 'Monitor', and 'Function Keys'. The 'Code + Test' view shows the C# code for 'run.csx' in the file explorer.

```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task<IActionResult>
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = req.Query["name"];
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17
18     string responseMessage = string.Format("This HTTP triggered function executed successfully. Hello, {0}!", name);
19
20     return new OkObjectResult(responseMessage);
21 }
22
23 }
```

On the right, the 'Output' tab is active, showing the 'HTTP response code' as '200 OK' and the 'HTTP response content' as 'Hello, Nayan. This HTTP triggered function executed successfully.'.

At the bottom, the 'App Insights Logs' section shows the following log entries:

```
2023-05-16T03:56:46Z [Information] C# HTTP trigger function processed a request.
2023-05-16T03:56:46Z [Information] Executed 'Functions.HttpTrigger1' (Succeeded, Id=57be0f82-7a9e-4d6e-bb6c-29d919d3a8cf, Duration=28ms)
```

Buttons for 'Run' and 'Close' are located at the bottom right of the output pane.

➔ That all to test the function in the portal now let's head to our browser and see if it runs there too or not.

For that click on "Get function URL" and copy the URL.

The screenshot displays the Microsoft Azure portal interface for a function named 'HttpTrigger1'. The top navigation bar shows the user is logged in as 'N.R.Khanal@wlv.ac.uk' from the 'UNIVERSITY OF WOLVERHAMPTON'. The breadcrumb trail indicates the path: Home > Microsoft.Web-FunctionApp-Portal-3524b25e-8ce9 | Overview > testnayan | Functions > HttpTrigger1.

The main section is titled 'HttpTrigger1 | Code + Test'. It features a search bar and several action buttons: Save, Discard, Refresh, Test/Run, Test integration, Upload, and Get function URL. On the left, a sidebar shows the 'Developer' view with options for Overview, Code + Test (selected), Integration, Monitor, and Function Keys.

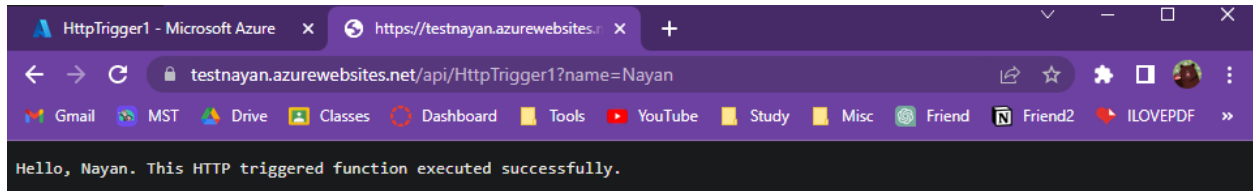
The 'Code + Test' panel shows the source code for 'run.csx' in a dark-themed editor. The code is as follows:

```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async Task<ActionResult> Run(HttpRequest req, ILogger log)
9 {
10     log.LogInformation("C# HTTP trigger function processed a request.");
11
12     string name = req.Query["name"];
13
14     string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
15     dynamic data = JsonConvert.DeserializeObject(requestBody);
16     name = name ?? data?.name;
17
18     string responseMessage = string.IsNullOrEmpty(name)
19         ? "This HTTP triggered function executed successfully. Pass a name in the query string or in the request body."
20         : $"Hello, {name}. This HTTP triggered function executed successfully.";
21
22     return new OkObjectResult(responseMessage);
23 }
```

To the right of the code editor is the 'Get function URL' section, which includes a 'Key' dropdown set to 'default' and a 'URL' text box containing 'https://testnayan.azurewebsites.net/api/...'. A 'Copy to clipboard' button is located next to the URL.

At the bottom of the portal, the 'Logs' section is expanded, showing a list of log entries. The first entry states: 'Connected! You are now viewing logs of Function runs in the current Code + Test panel. To see all the logs for this Function, please go to 'Monitor' from the Function menu.' Subsequent entries show the function being executed successfully at 2023-05-16T03:56:46Z.

➔ Paste the URL in the browser and append '?name=Nayan'



And we're done, we have successfully created an Azure Function.