

Run Serverless code with Azure Functions in Azure portal

Use Case:

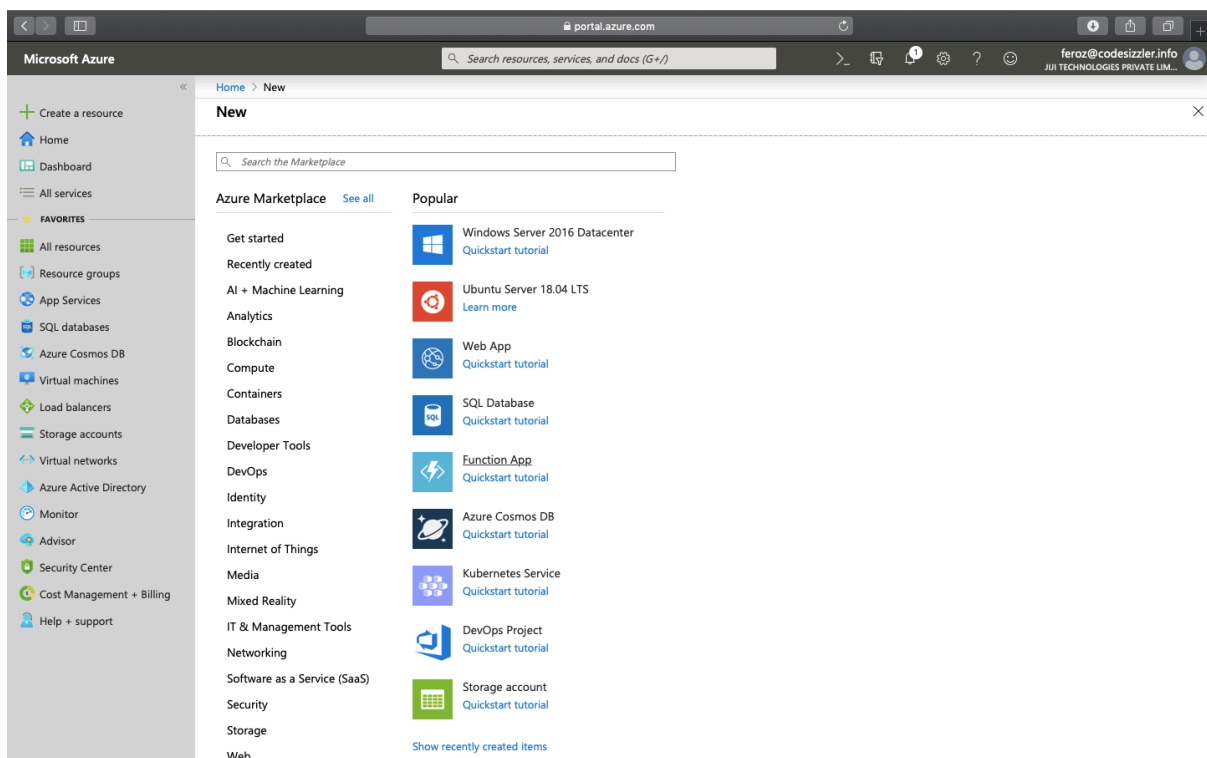
In this walkthrough you will write and run serverless code inside an *Azure Function App* in Azure portal.

Prerequisites

An active Azure subscription is required. If you do not have an Azure subscription, create a [free Azure account](#) before you begin.

Steps

1. To create a new Azure Function App, select the **Deploy to Azure** button. Sign into Azure Portal, when prompted.



2. Fill in the Azure Function App settings fields with the following details.

- App Name: Provide a unique name
- Subscription: Select on your Subscription (Visual Studio Enterprises-MPN)

Subscription * ⓘ

Visual Studio Enterprise – MPN

Resource Group: Choose Create new. Provide a unique name for your new Resource group (az900-rg) finally click on ok button.

Resource Group * ⓘ

Select existing... ▼

Create new

A resource group is a container that holds related resources for an Azure solution.

Name *

az900-rg ✓

OK Cancel

- **Instance details:** Give a unique name for Azure Function App, you should find a green colour tick mark as shown below which reflects the availability of the function app name. (codesizzlerfn)

Instance Details

Function App name *

codesizzlerfn ✓

.azurewebsites.net

- **Publish:** Select the platform on which the function app has to be published, Microsoft Azure Function App now supports both Code & Container environment over Docker. Here on this demo we will be selecting Code platform to publish the function app. (Code)

Publish *

Code Docker Container

- **Runtime stack:** Select .NET Core (this is suitable for running C# and F# functions).

Runtime stack *

.NET Core ▼

- **Location:** Choose the Azure region that is closest to your location. (South India)

Region *

South India ▼

- **Storage:** Choose Create new. Provide a unique name for your new storage account, if Azure has not provided a name automatically. (storageaccountaz900900d)

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 * 
[Create new](#)

- **OS:** Select Windows. For Linux hosting, see [Create your first function running on Linux using the Azure CLI](https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function-azure-cli-linux) at <https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function-azure-cli-linux>


Operating system

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

Operating System * ☐ Linux ☒ Windows

- **Hosting plan:** Choose (Consumption plan). With Azure server less hosting, you only pay for the time that your functions run. Using the default Consumption Plan means that resources are added dynamically as required by your functions.

Plan


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

Plan type *  

- **Application Insights:** Leave this set to the default value, provided by Azure automatically.

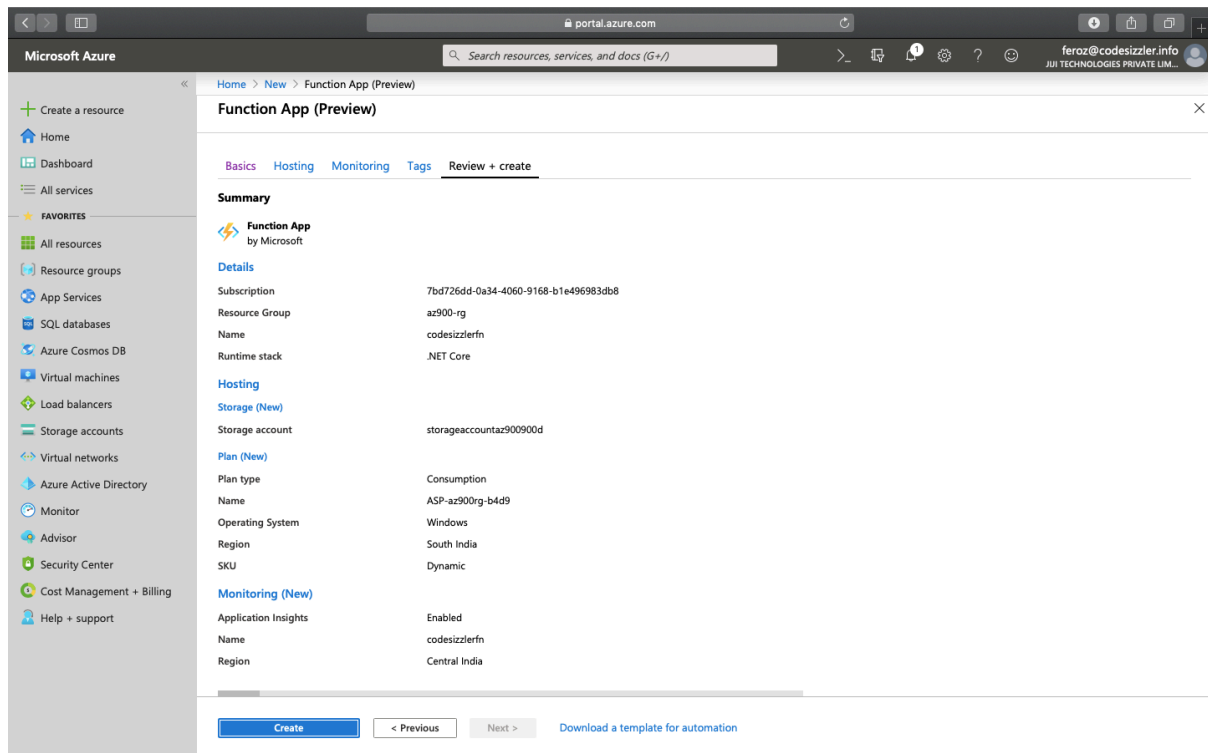
Application Insights

Enable Application Insights * ☐ No ☒ Yes

Application Insights * 
[Create new](#)

Region

3. Select the Create button to begin provisioning and deploying your new Azure Function App.



Note: When the deployment starts, a notification appears in Azure Portal indicating the deployment is in progress. Another notification is displayed when the deployment has completed successfully.

Your deployment is complete

Deployment name: Microsoft.Web-FunctionApp-Portal-ed97ba35-...
 Subscription: [Visual Studio Enterprise – MPN](#)
 Resource group: [az900-rg](#)

Start time: 10/24/2019, 5:04:41 PM
 Correlation ID: b46366c4-caf0-435c-9708-7fae36efbcbd

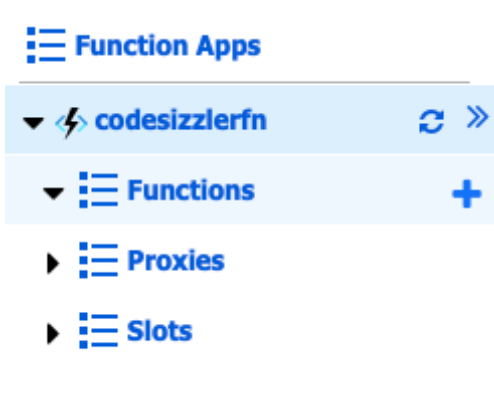
Deployment details [\(Download\)](#)

Next steps

[Go to resource](#)

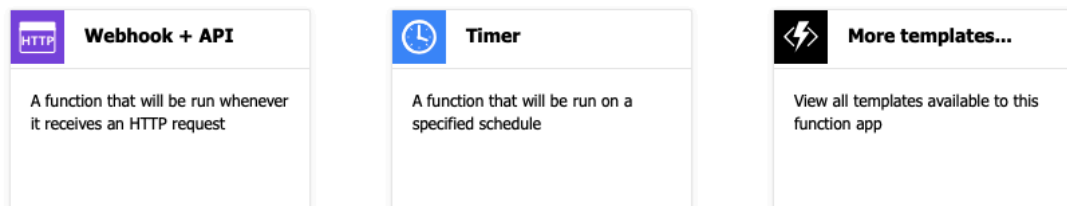
4. When the deployment has completed, choose Go to resource from the notification area to view your new Azure Function App. You can also select all resources from the main Azure menu, then choose your Azure Function App from the list of resources.

5. To create an HTTP Triggered Function, use the down arrow icon to expand your Azure Function App. Select the “+” button next to Functions. Choose In-portal, and select Continue



Continue

6. Choose WebHook + API, and then select Create.



Back

7. Select `</>` Get function URL from the within the function editor. Set the Key value to default (Function key) using the dropdown. Then, select Copy to copy the function URL.

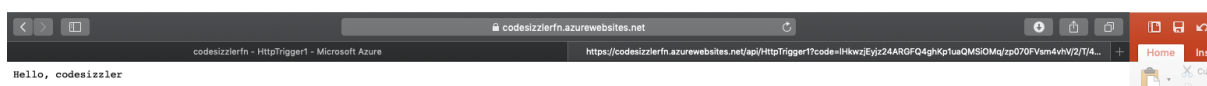
`</>` Get function URL

8. Paste the copied function URL into your web browser's address bar. Append `&name=<yourname>` to the end of the URL.

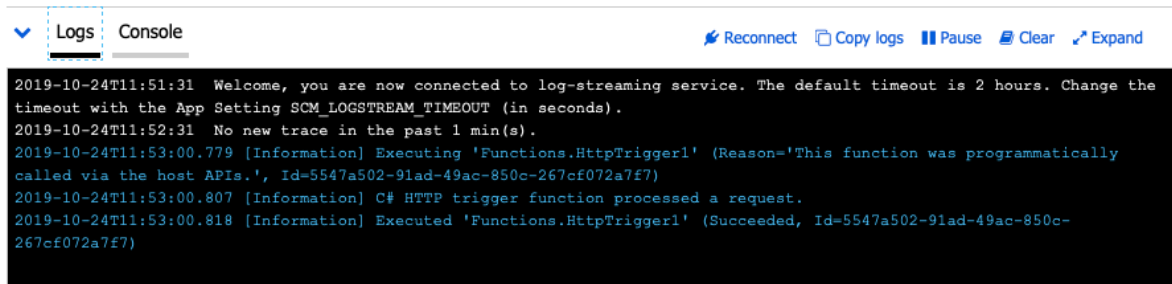
Get function URL

Key	URL	
default (Function key)	https://codesizzlerfn.azurewebsites.net/api/HttpTrigger1?code=IH	Copy

Note: Here, `<yourname>` refers to your given first name. Navigate to the URL to see the “Hello” message, followed by the name you provided, displayed in your browser. The URL should be similar to <https://azfunc01.azurewebsites.net/api/HttpTrigger1?code=X9xx9999xXXXXX9x9xxxXX==&name=glen>



9. When your function runs, trace information is written to log files in Azure. To view the logs in Azure portal, return to the function editor, and select the Logs button.



```
2019-10-24T11:51:31 Welcome, you are now connected to log-streaming service. The default timeout is 2 hours. Change the
timeout with the App Setting SCM_LOGSTREAM_TIMEOUT (in seconds).
2019-10-24T11:52:31 No new trace in the past 1 min(s).
2019-10-24T11:53:00.779 [Information] Executing 'Functions.HttpTrigger1' (Reason='This function was programmatically
called via the host APIs.', Id=5547a502-91ad-49ac-850c-267cf072a7f7)
2019-10-24T11:53:00.807 [Information] C# HTTP trigger function processed a request.
2019-10-24T11:53:00.818 [Information] Executed 'Functions.HttpTrigger1' (Succeeded, Id=5547a502-91ad-49ac-850c-
267cf072a7f7)
```

