

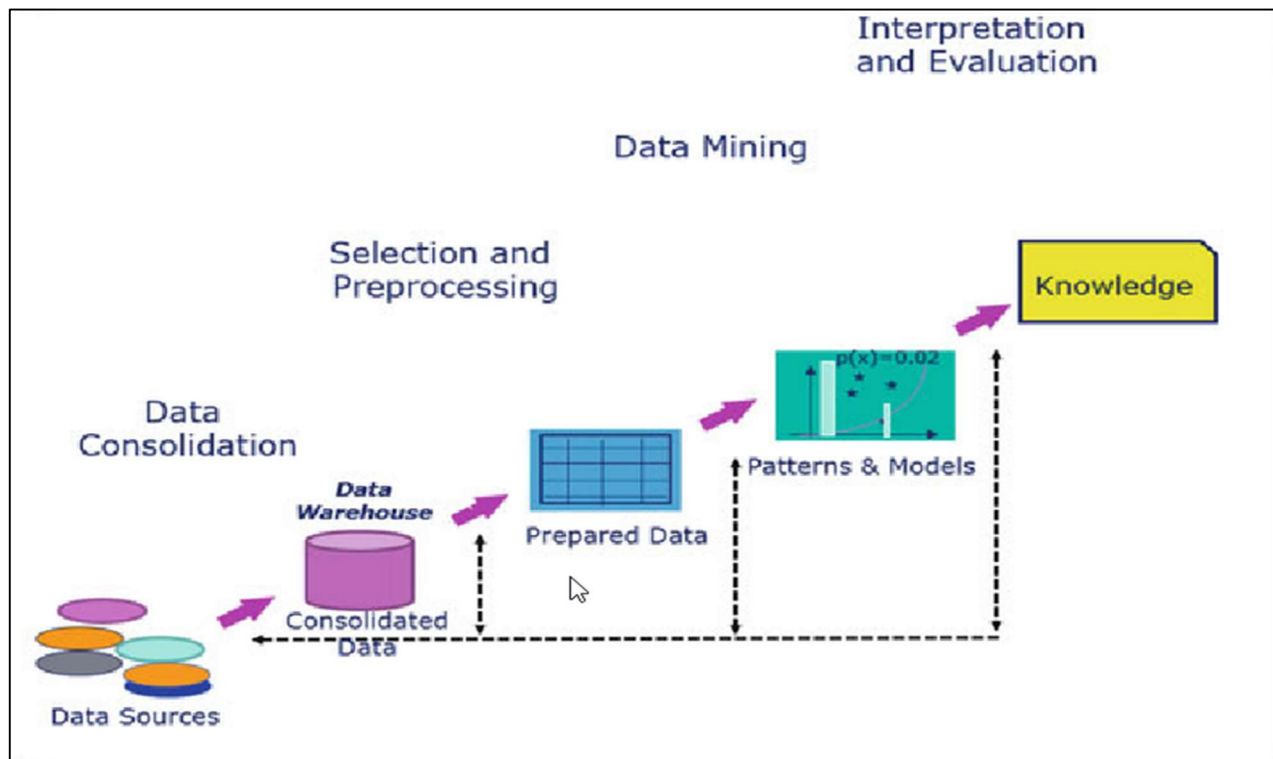
Develop a Fully automated data analytics Platform with Azure

In this blog, I have created a Function App on Azure using Java, Data Bricks workspace and devised a way to “Automate the data analytics platform” with the following Azure Services: IoT Hub, EventHub, Function App, Azure Storage, Azure Data Bricks Azure Cosmos DB, and IAM. We tried multiple use cases across domains Industry 4.0, Manufacturing, Factory of Future and Renewable Energy. By following this blog, you can seamlessly automate your data analytics platform use cases within azure.

Problem statement

All businesses (manufacturing etc.), whether they are growing or established need a data analytics strategy. We tried to address the below problem statements through azure PAAS services.

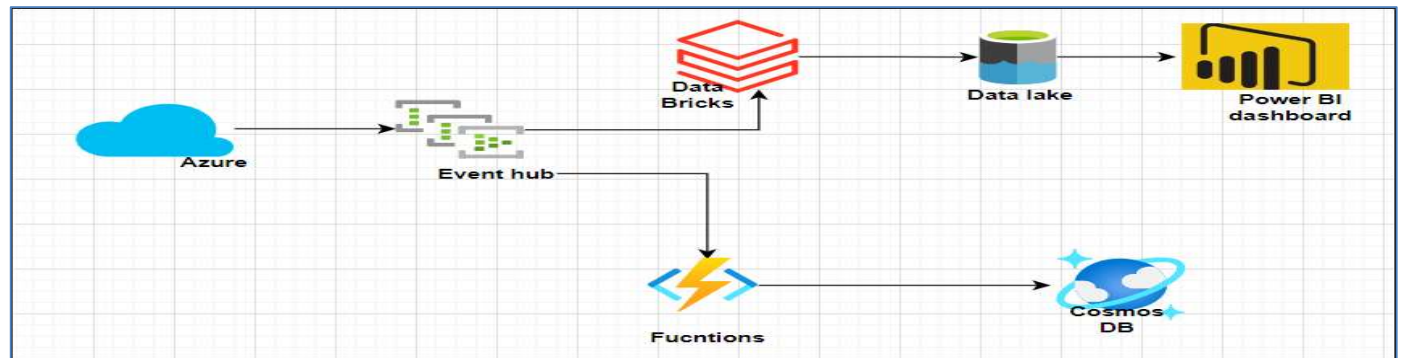
1. The IoT devices were getting disconnected to due to component failures.
2. The Data Aggregation was to complex using Azure functions.
3. There is no near real time visualization dashboard for monitoring the device metrics.
4. Unable to derive meaningful insights from the telemetry data.



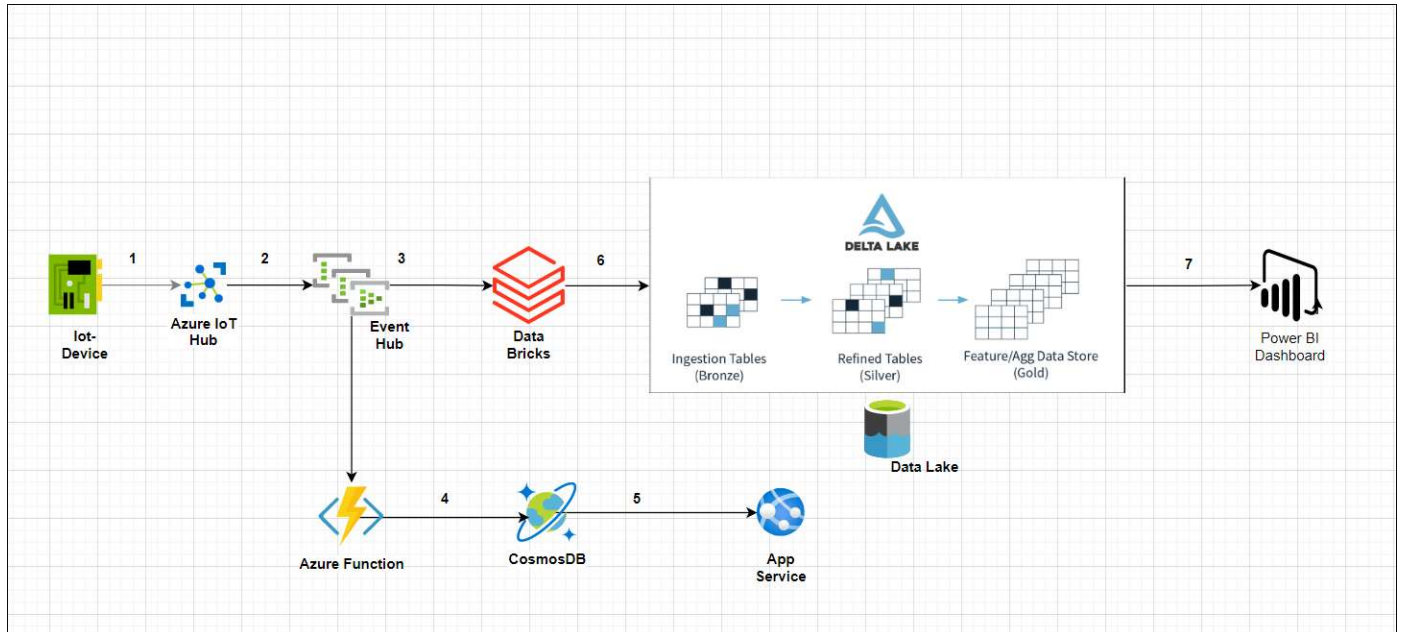
However, data analytics is complex due to multiple reasons:

- The size (millions of messages) of the data received form data sources.
- The frequency at which the data is being collected and how it's sent to cloud.
- Streaming and processing the raw data before storing it in data lake.
- To get meaning full insights and visualizing the same.

Solution Block:



Architecture Diagram:



1. **IoT device** Agent has been developed using Azure IoT SDK, the agent collects the telemetry data and sends it to IoT Hub (blogathon-iothub).
2. **Messaging Route** is in place for routing the message from IoT Hub(blogathon-iothub) to EventHub(azureblogathon-telemetry).
3. **Data Bricks** service consumes the messages from the EventHub through respective consumer group.
4. **Azure function** is configured with EventHub Trigger, the function will write the telemetry data to the cosmos DB.
5. End customers can view the devices status details through webApp.
6. Using **Delta Lake** and python notebook the telemetry data is processed and stored in respective tables (Bronze, Silver, Gold).
7. Near real-time **PowerBI** dashboard has been integrated with Databricks for visualization.

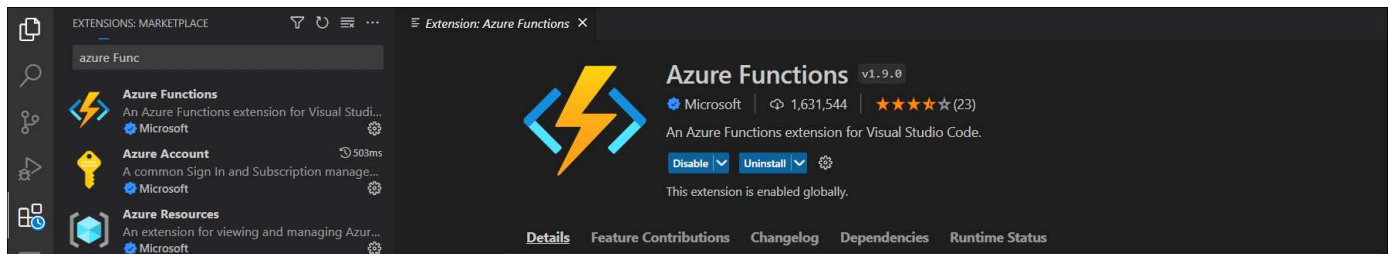
Note: As described in the introduction, the below steps can be used to replicate the whole data analytics process based on your requirement.

The implementation of data analytics has been divided in 5 steps:

- Deploying the required resources.
- Deploying the code to Azure Functions.
- Establishing Function App connection with the Cosmos DB.
- Configuring Data Bricks cluster.
- Testing the workflow by simulating the telemetry data and visualizing the data in Power BI.

Pre-requisites:

- An Azure account with an active subscription. Create an account for free.
- The Azure Functions Core Tools version 3.x.
- Java versions that are supported by Azure Functions.
- Visual Studio Code on one of the supported platforms.
- The Java extension for Visual Studio Code.
- The Azure Functions extension for Visual Studio Code.



Azure Functions VS Code extension

Step 1: Deploying the required resources

1.1 IoThub: - Creating IoT Hub from Azure portal, below are the configurations.

On the Azure homepage, select the + Create a resource button.

On the Basics tab, complete the fields as follows:

Subscription: Select the subscription to use for your hub (**Ex: - Visual studio enterprise subscription**).

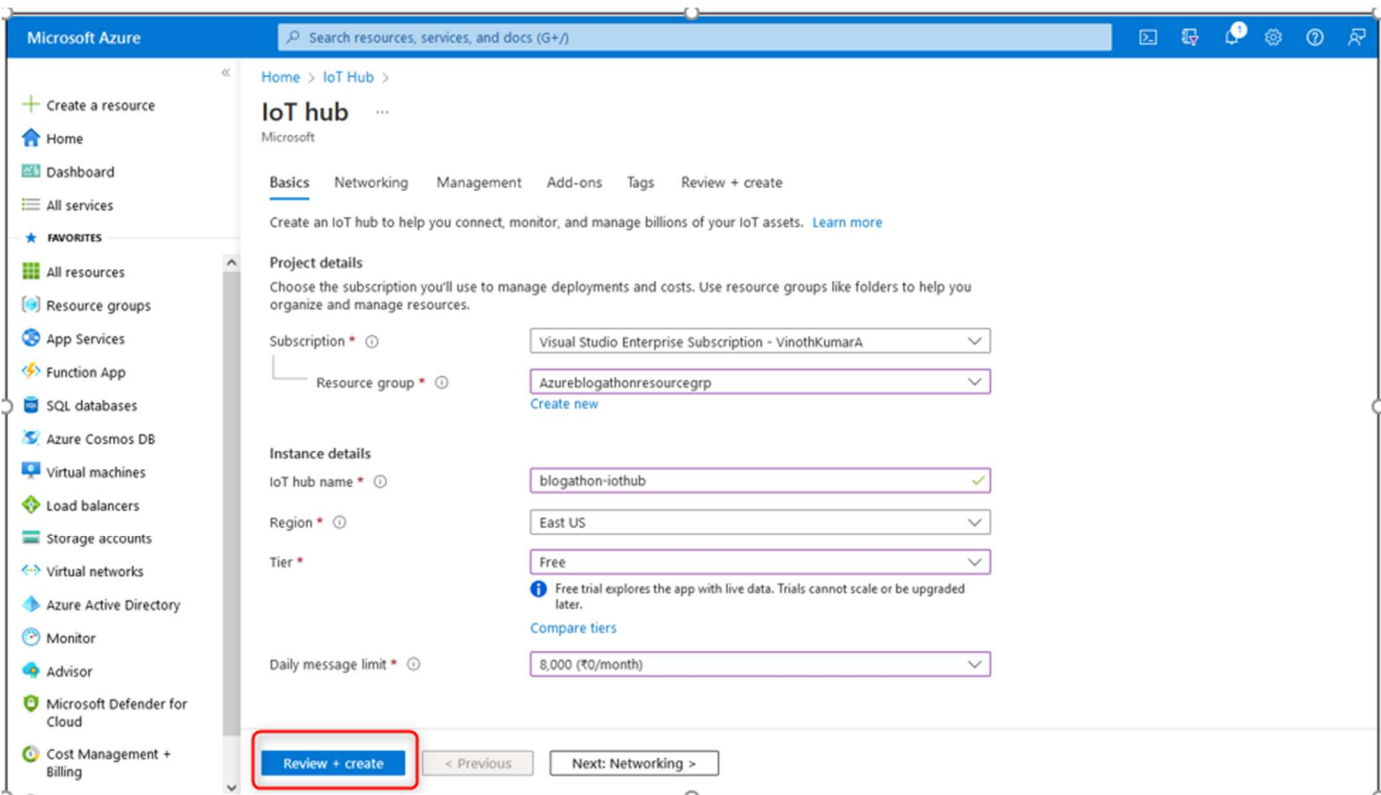
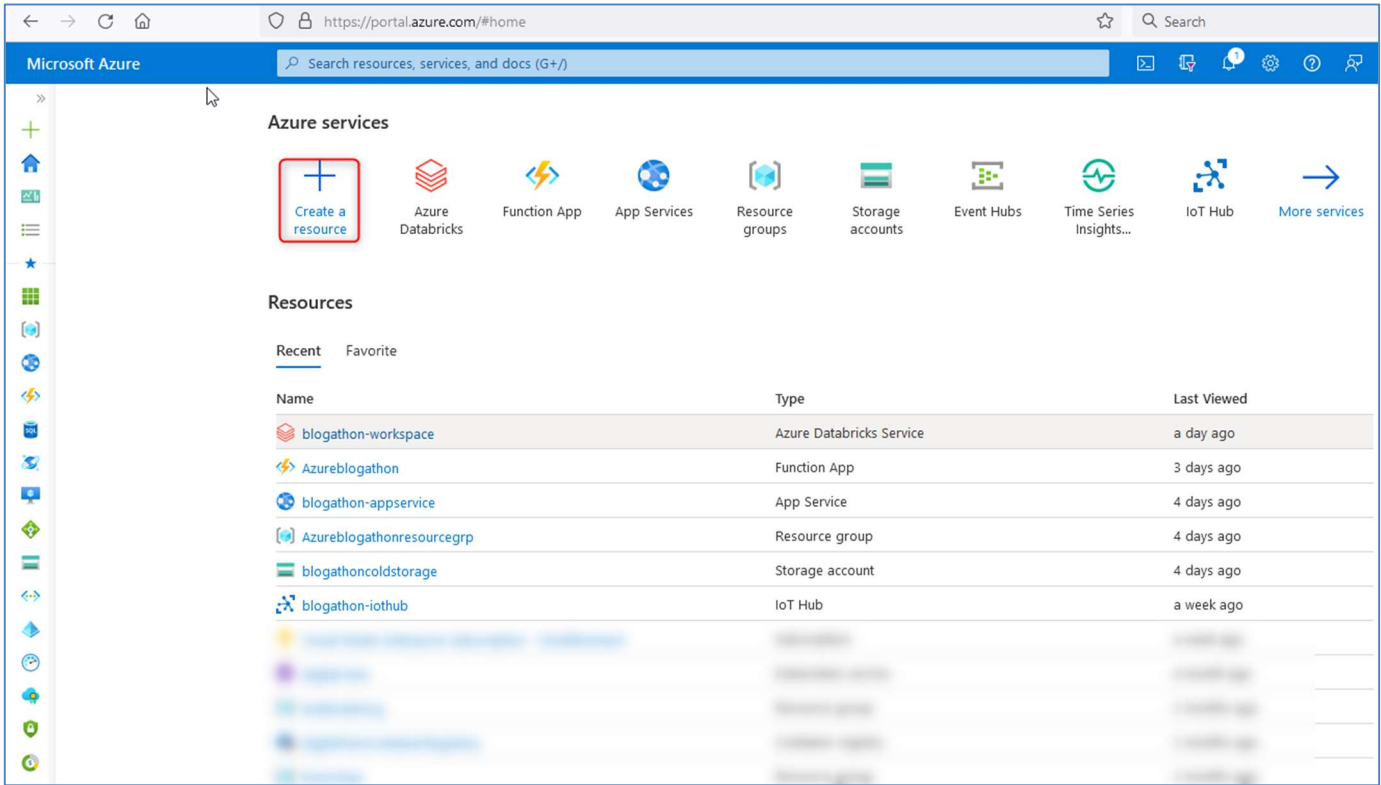
Resource group: Select a resource group or create a new one (**Ex: - Azureblogathonresourcegrp**).

IoT hub name: Enter a name for your hub (**Ex: - blogathon-iothub**).

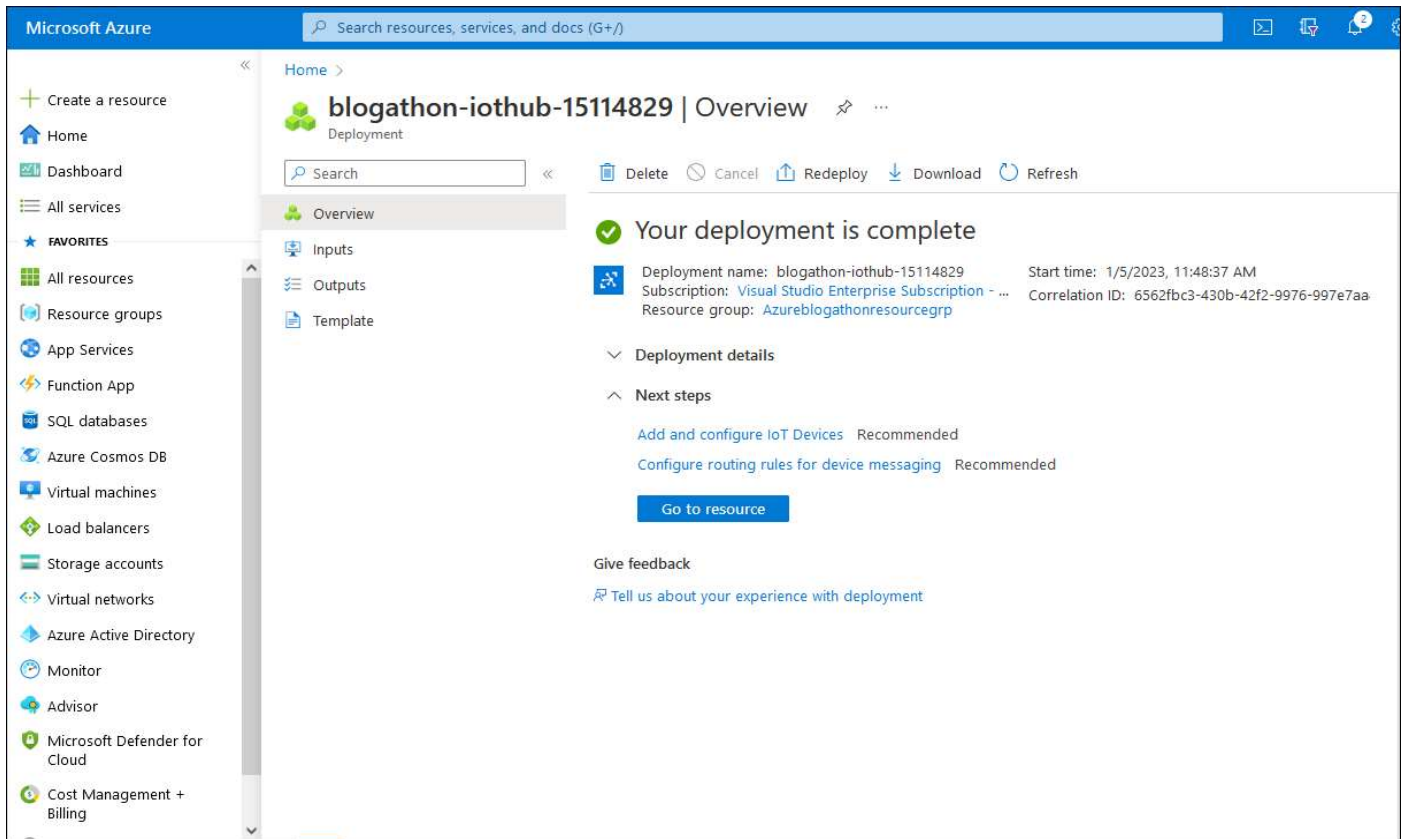
Region: Select the region, closest to you (**Ex: - East US**).

Tier: Select the tier that you want to use for your hub (**EX: -Free**).

Daily message limit: Select the maximum daily quota of messages for your hub (**Ex: - 8000**).



Once the deployment is successful, you will see this screen.



1.2 EventHub: - Creating event hub from Azure portal, below are the configurations.

On the Azure homepage, select the + Create a resource button.

On the Basics tab, complete the fields as follows:

Subscription: Select the subscription to use for your hub (Ex: - **Visual studio enterprise subscription**).

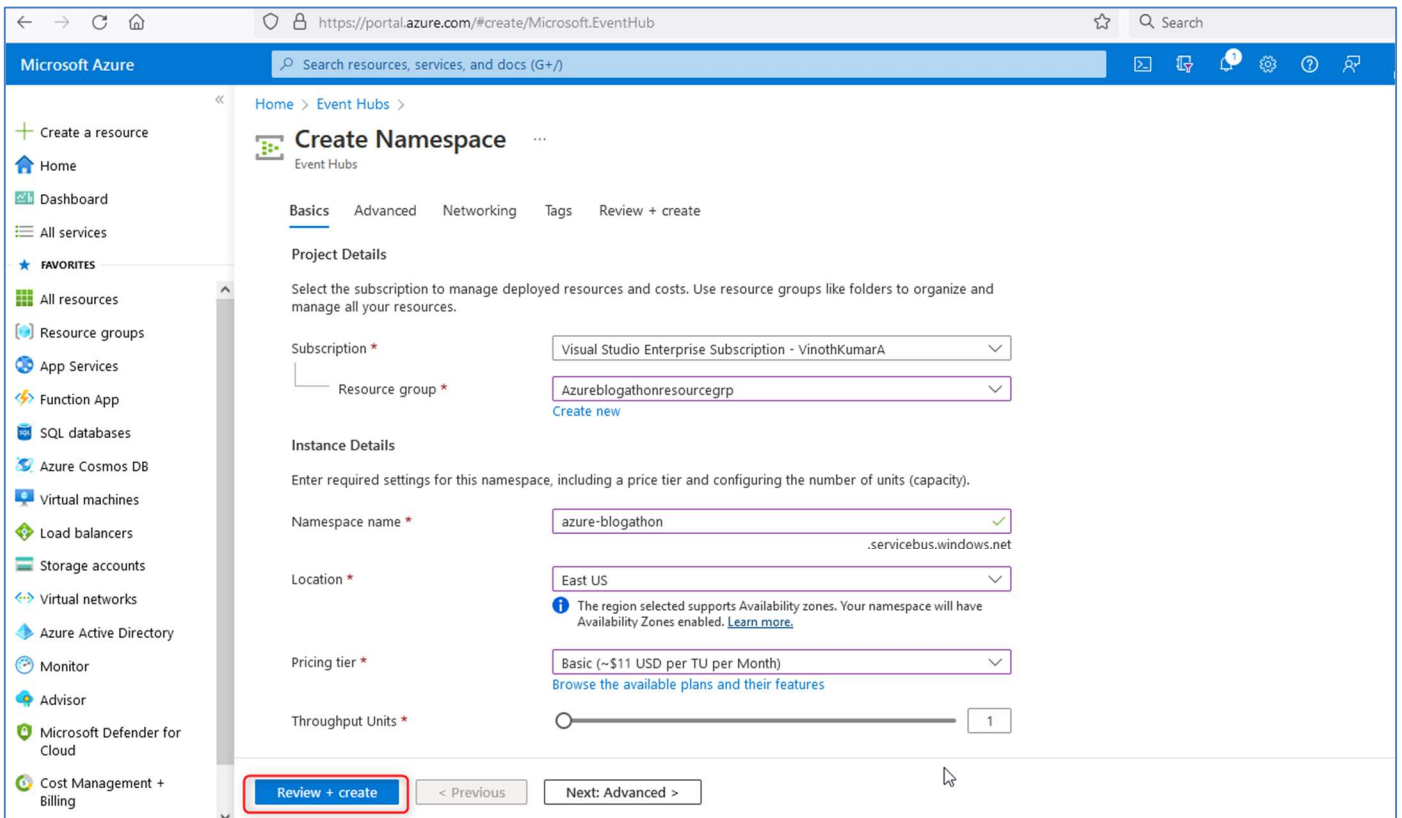
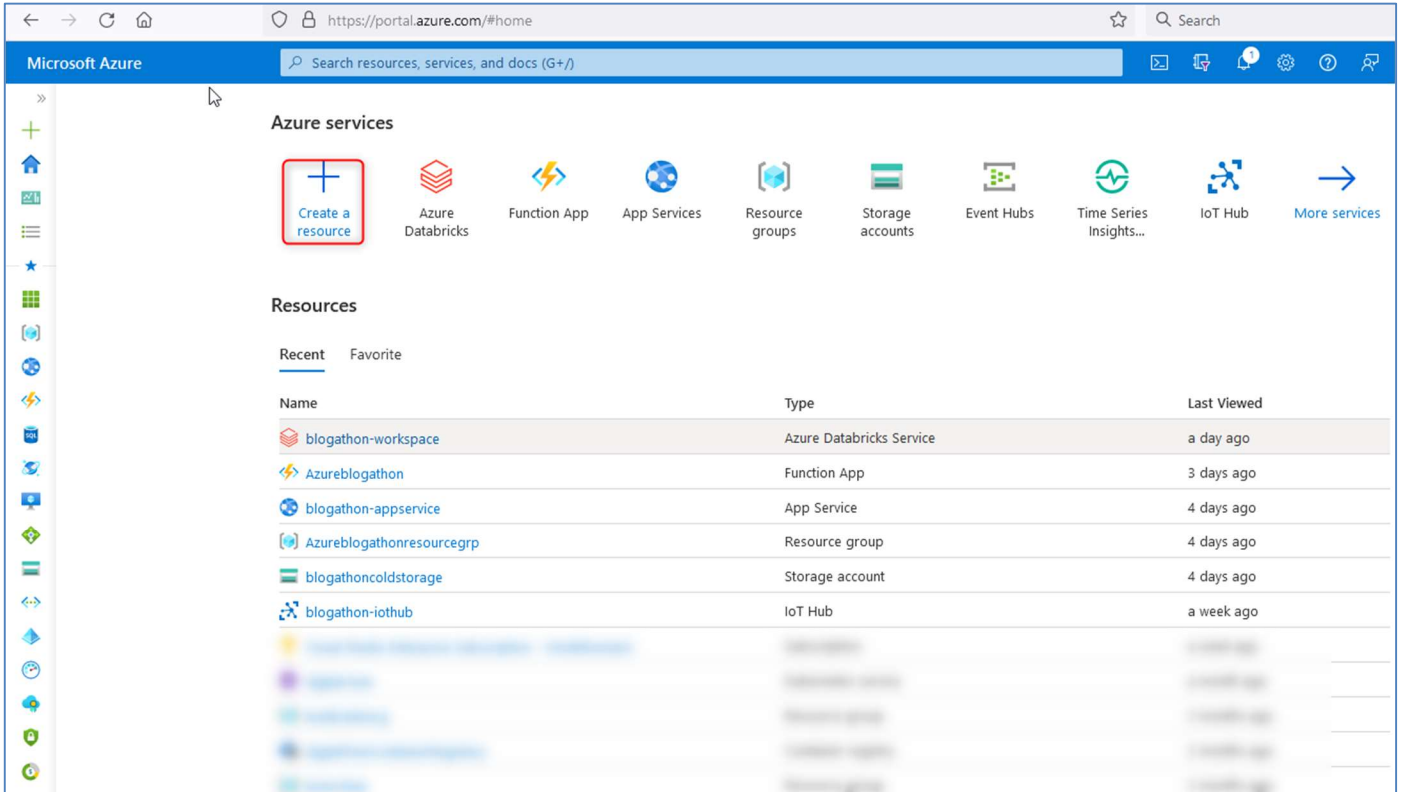
Resource group: Select a resource group or create a new one (Ex: - **Azureblogathonresourcegrp**).

Namespace name: Enter a name for your Namespace (Ex: - **azure-obligation**)

Location: Select the region, closest to you (Ex: - **East US**).

Pricing tier: Select a pricing tier (Ex: - **Basic**)

Throughput units: Select number of throughput units (Ex: - **1**)



Create Event Hub

Event Hubs

Basics **Capture** Review + create

Event Hub Details

Enter required settings for this event hub, including partition count and message retention.

Name *

Partition count

Retention

Configure retention settings for this Event Hub. [Learn more](#)

Cleanup policy

Retention time (hrs) min. 1 hour, max. 24 hours (1day)

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

Once the deployment is successful, you will see this screen.

The screenshot shows the Azure portal interface for a deployment. The main content area displays a success message: "Your deployment is complete". Below this, the deployment details are shown in a table format.

Resource	Type	Status	Operation details
azure-blogathon	Microsoft.EventHub/namespaces	OK	Operation details

Additional deployment information includes: Deployment name: azure-blogathon, Subscription: Visual Studio Enterprise Subscription - VinothKumarA, Resource group: Azureblogathonresourcegrp, Start time: 1/9/2023, 11:58:46 AM, and Correlation ID: 8f79a7ab-3ca7-4644-85de-670d6c443e56.

1.3 Function App: - Creating Function App from Azure portal, below are the configurations.

On the Azure homepage, select the + Create a resource button.

On the Basics tab, complete the fields as follows:

Subscription: Select the subscription to use for your hub (**Ex: - Visual studio enterprise subscription**).

Resource group: Select a resource group or create a new one (**Ex: - Azureblogathonresourcegrp**).

FunctionApp name: Enter a name for your FcuntionApp (**Ex: - Azureblogathon**)

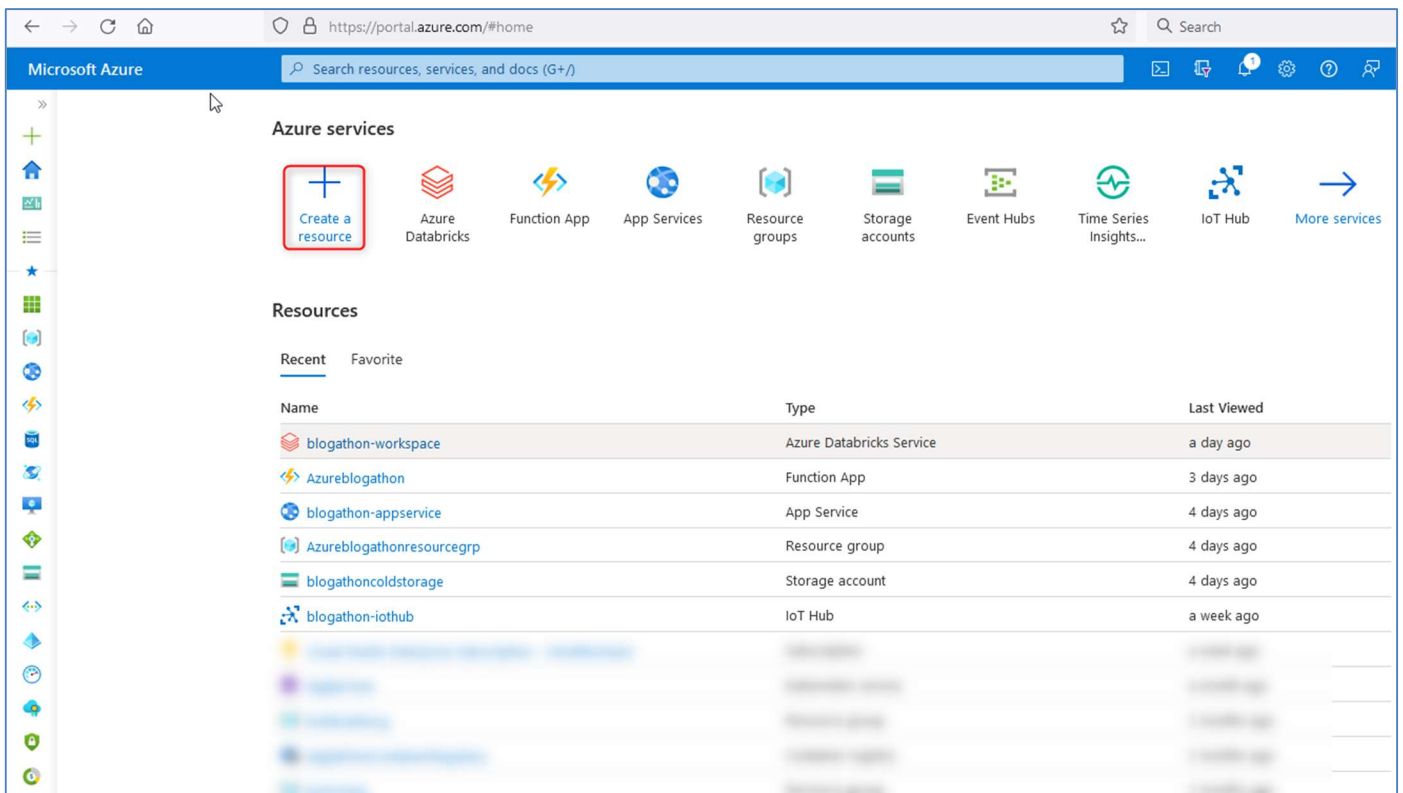
Publish: Option to publish code files or a Docker container (**Ex: - code**).

Runtime stack: Choose a runtime that supports your favorite function programming language (**Ex: - Java**).

Version: Choose the version of your installed runtime **Ex: - 11**).

Location: Select the region, closest to you (**Ex: - East US**).

Plan: Choose a plan type (**Ex: - Serverless**).



Home > Function App >

Create Function App

Subscription *

Resource Group * [Create new](#)

Instance Details

Function App name * .azurewebsites.net

Publish * Code Docker Container

Runtime stack *

Version *

Region *

Operating system

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

Operating System * Linux Windows

Plan

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

Plan type *

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

Once the deployment is successful, you will see this screen.

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

Home > Microsoft.Web-FunctionApp-Portal-4f2a3b35-bbee | Overview

Deployment

Search Delete Cancel Redeploy Download Refresh

Overview

Your deployment is complete

Deployment name: Microsoft.Web-FunctionApp-Portal-4f2a3b35... Start time: 1/9/2023, 12:18:50 PM
 Subscription: Visual Studio Enterprise Subscription - VinothKumarA Correlation ID: a3286401-55ee-405b-8e74-ca48ef6b6c19
 Resource group: Azureblogathonresourcegrp

Deployment details

Resource	Type	Status	Operation details
Azureblogathon	Microsoft.Web/sites	OK	Operation details
Azureblogathon	microsoft.insights/components	OK	Operation details
Azureblogathon	microsoft.insights/components	OK	Operation details
azureblogathonresou8a18	Microsoft.Storage/storageAccounts	OK	Operation details
ASP-Azureblogathonresourcegrp-aa26	Microsoft.Web/serverfarms	OK	Operation details
azureblogathonresou8a18	Microsoft.Storage/storageAccounts	OK	Operation details
newWorkspaceTemplate	Microsoft.Resources/deployments	OK	Operation details

Next steps

- [Create a function.](#) Recommended
- [Manage deployments for your app.](#) Recommended

[Go to resource](#)

1.4 Cosmos DB: - Creating Cosmos DB from Azure portal, below are the configurations.

On the Azure homepage, select the + Create a resource button.

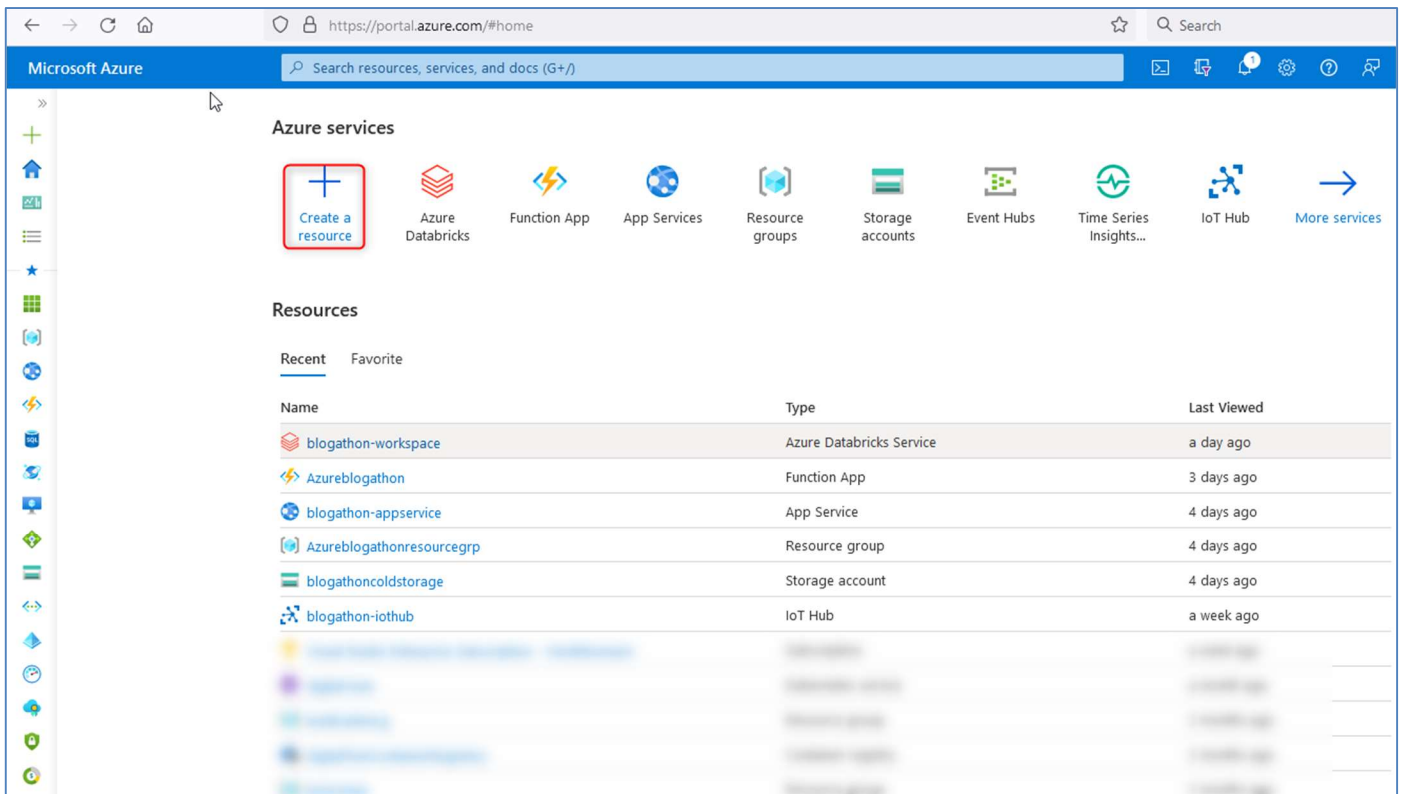
On the Basics tab, complete the fields as follows:

Subscription: Select the subscription to use for your hub (**Ex: - Visual studio enterprise subscription**).

Account Name: Enter a name for your Cosmos DB(**Ex:- azureblogathon-cosmosdb**).

Region: Select the region, closest to you (**Ex: - East US**).

Capacity Mode: Choose a capacity mode (**Ex: - Serverless**).



Home > Azure Cosmos DB > Create an Azure Cosmos DB account >

Create Azure Cosmos DB Account - Azure Cosmos DB for NoSQL

Validation Success

Basics Global Distribution Networking Backup Policy Encryption Tags Review + create

Azure Cosmos DB is a fully managed NoSQL and relational database service for building scalable, high performance applications. [Try it for free](#), for 30 days with unlimited renewals. Go to production starting at \$24/month per database, multiple containers included. [Learn more](#)

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

Subscription * Visual Studio Enterprise Subscription - VinothKumarA

Resource Group * Azureblogathonresourcegrp
[Create new](#)

Instance Details

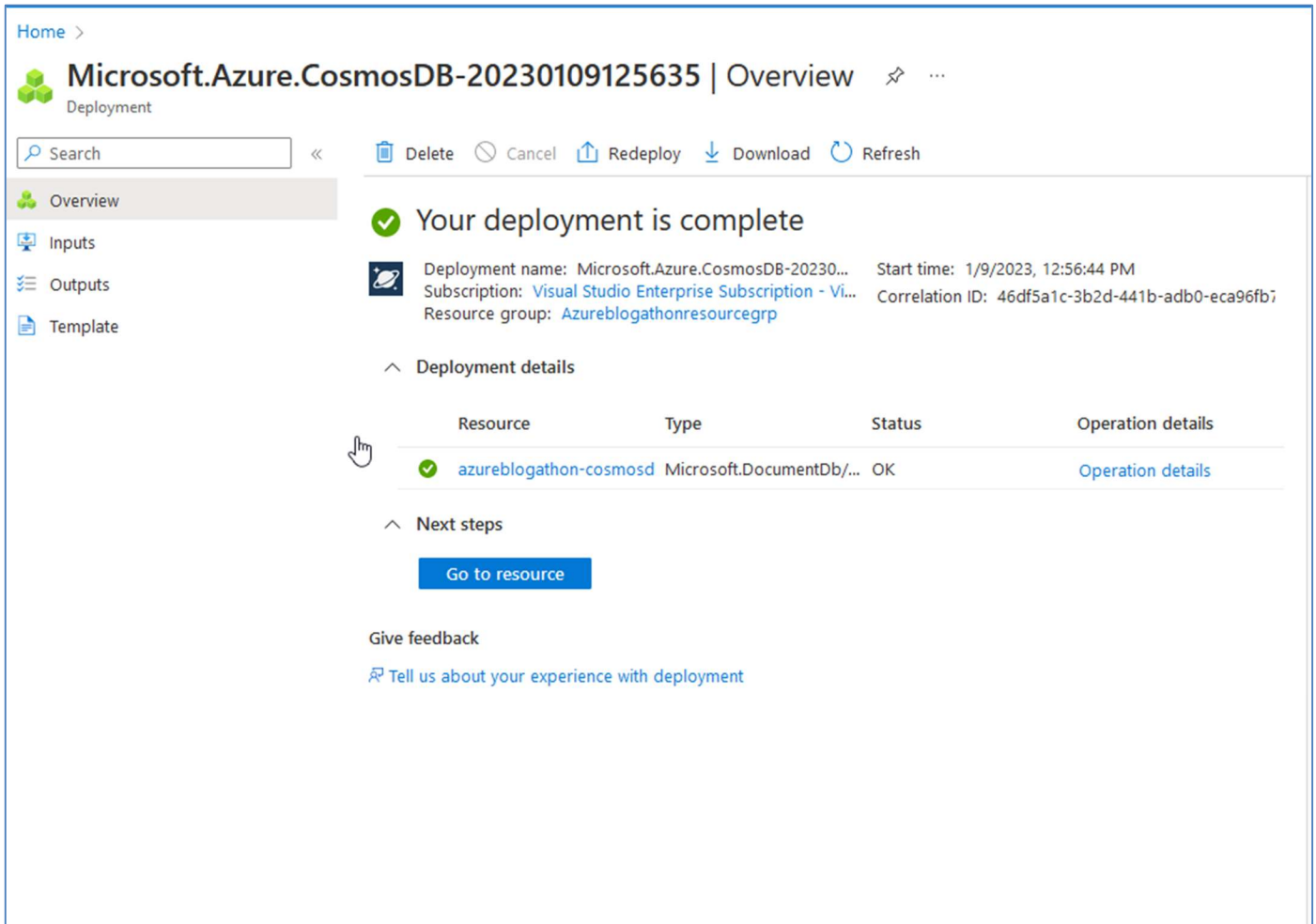
Account Name * azureblogathon-cosmosdb

Location * (US) East US

Capacity mode ⓘ Provisioned throughput Serverless
[Learn more about capacity mode](#)

[Review + create](#) Previous Next: Global Distribution

Once the deployment is successful, you will see this screen.



1.5 Storage account: - Creating Storage account from Azure portal, below are the configurations.

On the Azure homepage, select the + Create a resource button.

On the Basics tab, complete the fields as follows:

Subscription: Select the subscription to use for your hub (**Ex: - Visual studio enterprise subscription**).

Resource group: Select a resource group or create a new one (**Ex: - Azureblogathonresourcegrp**).

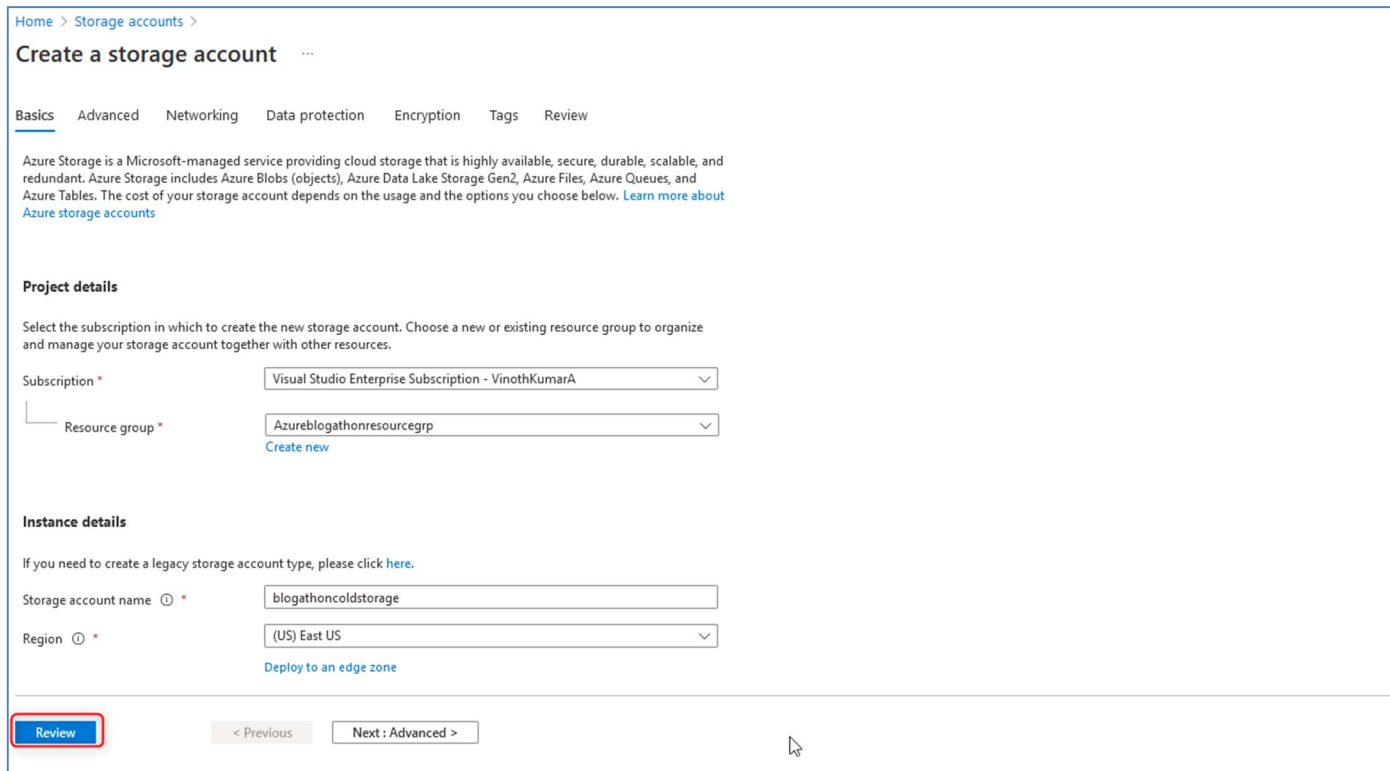
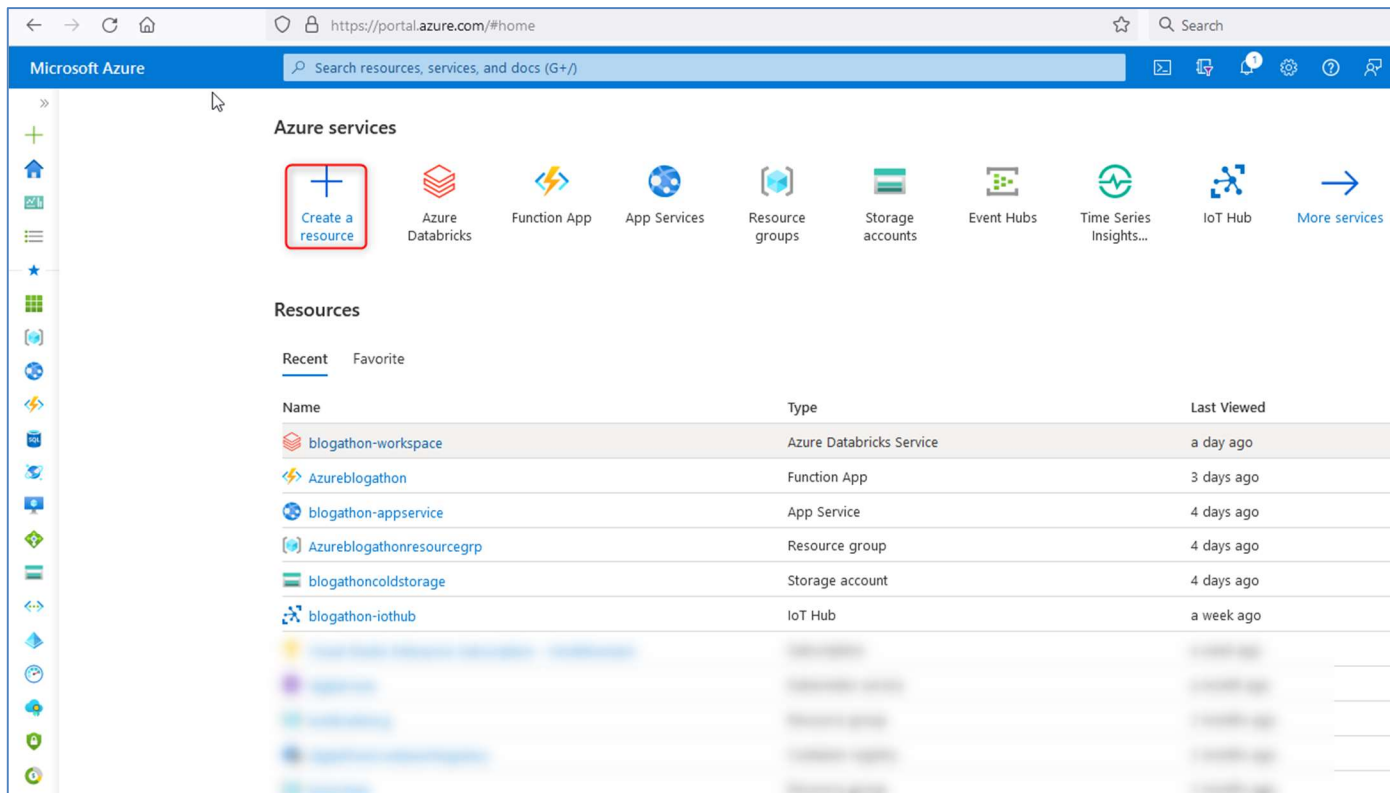
Storage account name: Enter a name for your Storage account (**Ex:- blogathoncoldstorage**)

Region: Select the region, closest to you (**Ex: - East US**).

On the Advanced tab, complete the fields as follows:

Data Lake Gen2:

Choose the option to **Enable Hierarchical namespace**.



Home > Storage accounts >

Create a storage account

Basics **Advanced** Networking Data protection Encryption Tags Review

Enable storage account key access

Default to Azure Active Directory authorization in the Azure portal

Minimum TLS version

Permitted scope for copy operations (preview)

Data Lake Storage Gen2

The Data Lake Storage Gen2 hierarchical namespace accelerates big data analytics workloads and enables file-level access control lists (ACLs). [Learn more](#)

Enable hierarchical namespace

Blob storage

Enable SFTP

Enable network file system v3

Allow cross-tenant replication

Review < Previous Next : Networking >

Once the deployment is successful, you will see this screen.

Home >

blogathoncoldstorage_1673255597841 | Overview

Deployment

Search << Delete Cancel Redeploy Download Refresh

- Overview
- Inputs
- Outputs
- Template

✓ Your deployment is complete

Deployment name: blogathoncoldstorage_1673255597841 Start time: 1/9/2023, 2:43:23 PM
 Subscription: Visual Studio Enterprise Subscription - VinothKumarA Correlation ID: 0bf80977-de1d-4815-857a-b22d2b95f37a
 Resource group: Azureblogathonresourcegrp

Deployment details

Resource	Type	Status	Operation details
blogathoncoldstorage/default	Microsoft.Storage/storageAcc...	OK	Operation details
blogathoncoldstorage/default	Microsoft.Storage/storageAcc...	OK	Operation details
blogathoncoldstorage	Microsoft.Storage/storageAcc...	OK	Operation details

Next steps

Go to resource

Give feedback
[Tell us about your experience with deployment](#)

1.6 Data Bricks workspace: - Creating workspace from Azure portal, below are the configurations.

On the Azure homepage, select the + Create a resource button.

On the Basics tab, complete the fields as follows:

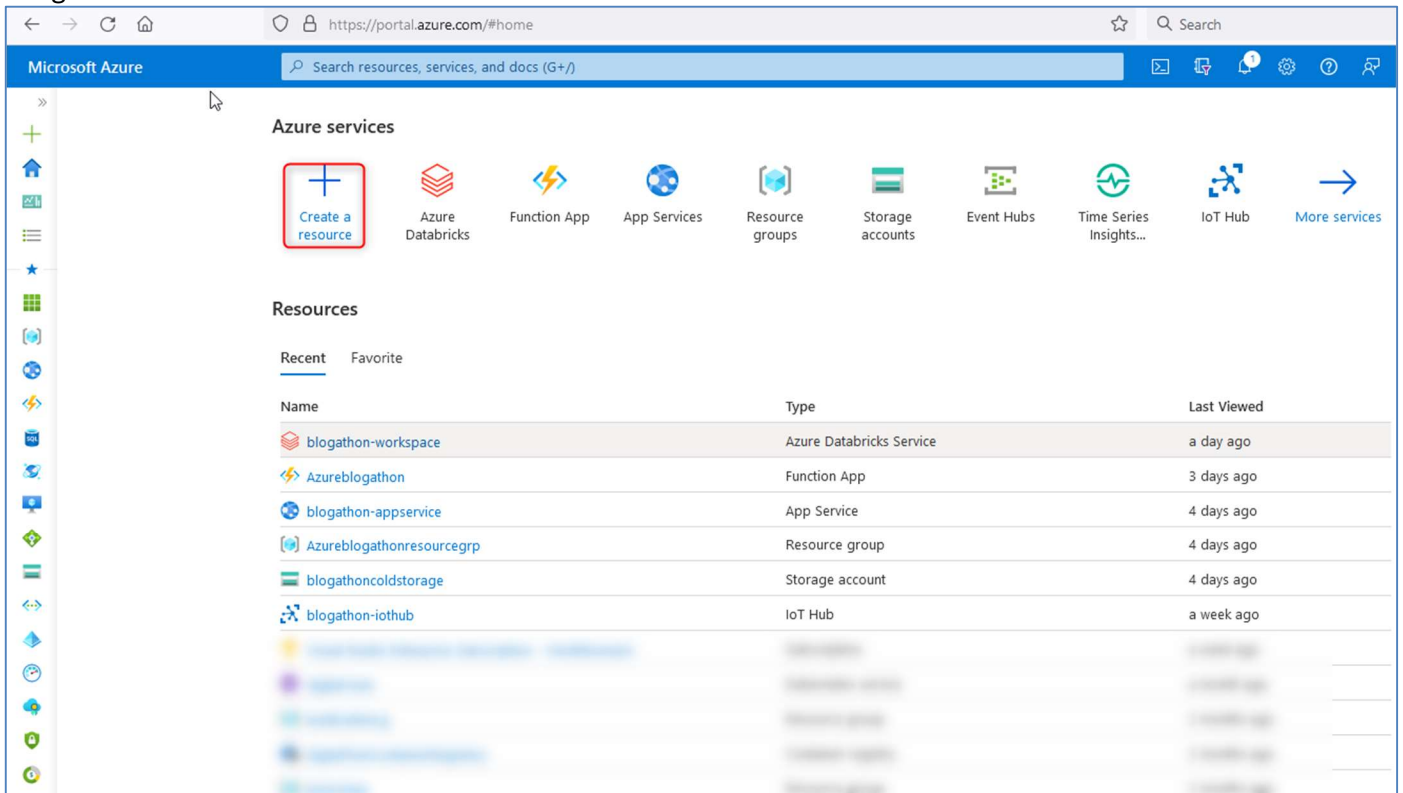
Subscription: Select the subscription to use for your hub (**Ex: - Visual studio enterprise subscription**).

Resource group: Select a resource group or create a new one (**Ex: - Azureblogathonresourcegrp**).

Workspace name: Enter a workspace name(**Ex:- blogathon-workspace**).

Region: Select the region, closest to you (**Ex: - East US**).

You need to have access to an Azure Databricks workspace to perform Structured Streaming with batch jobs by using Delta Lake.



Home > Azure Databricks >

Create an Azure Databricks workspace

Basics Networking Advanced Tags Review + create

Project Details

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

Subscription * ⓘ Visual Studio Enterprise Subscription - VinothKumarA

Resource group * ⓘ Azureblogathonresourcegrp [Create new](#)

Instance Details

Workspace name * blogathonworkspace ✓

Region * East US

Pricing Tier * ⓘ Trial (Premium - 14-Days Free DBUs)

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

Once the deployment is successful, you will see this screen.

Home >

Azureblogathonresourcegrp_blogathon-workspace | Overview

Deployment

Search << Delete Cancel Redeploy Download Refresh

- Overview
- Inputs
- Outputs
- Template

✓ Your deployment is complete

Deployment name: Azureblogathonresourcegrp_blogat... Start time: 1/9/2023, 2:54:13 PM
 Subscription: Visual Studio Enterprise Subscription - Vi... Correlation ID: fe81a49a-c7b6-4a9d-8915-d7f18b9
 Resource group: Azureblogathonresourcegrp

Deployment details

Resource	Type	Status	Operation details
blogathon-workspace	Microsoft.Databricks/workspaces	OK	Operation details

Next steps

[Go to resource](#)

Give feedback

[Tell us about your experience with deployment](#)

1.7 **Web App** - Creating Webapp service from Azure portal, below are the configurations.

On the Azure homepage, select the + Create a resource button.

On the Basics tab, complete the fields as follows:

Subscription: Select the subscription to use for your hub (Ex: - Visual studio enterprise subscription).

Resource group: Select a resource group or create a new one (Ex: - Azureblogathonresourcegrp).

WebApp name: Enter a name for your WebApp name (Ex: - blogathon-Appservice)

Publish: Option to publish code files or a Docker container (Ex: - code).

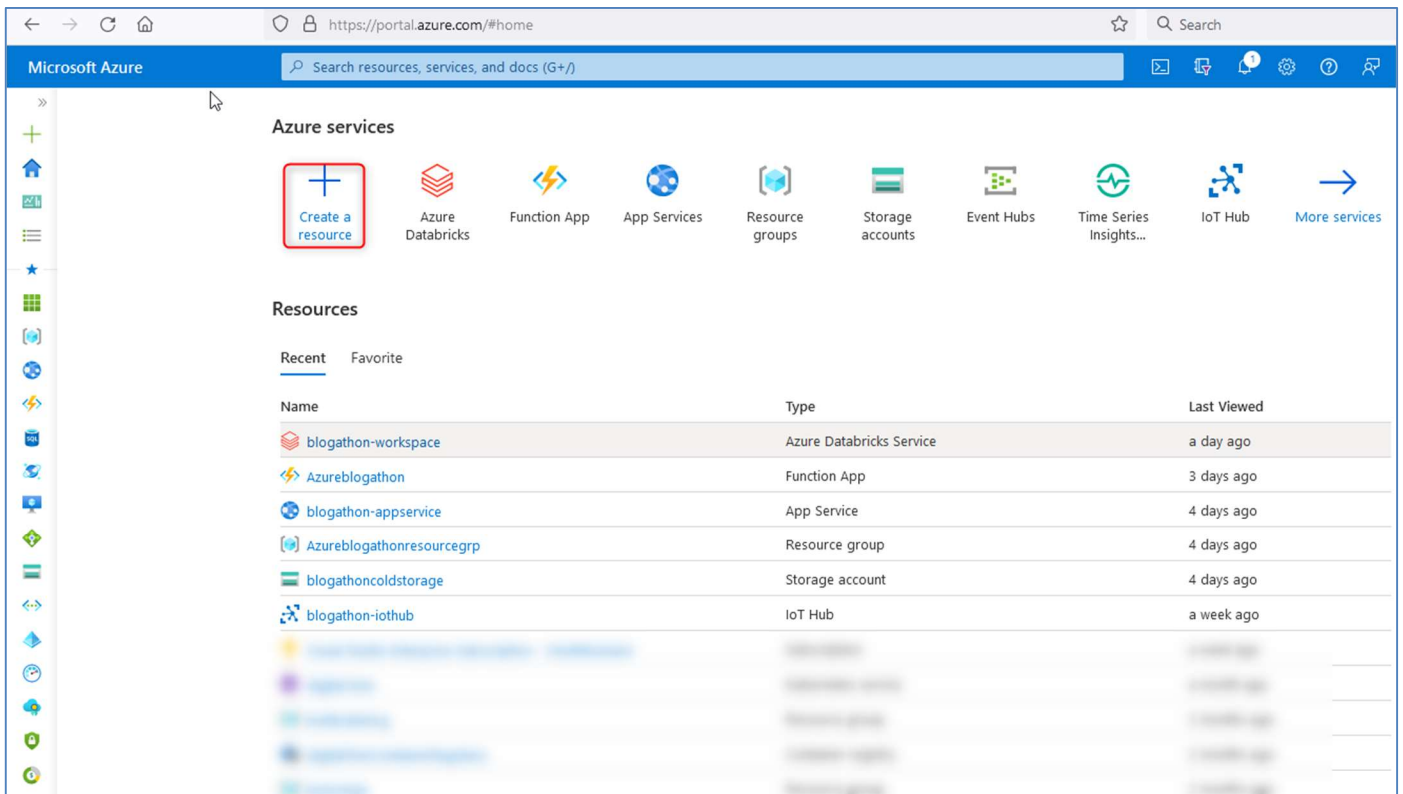
Runtime stack: Choose a runtime that supports your favorite function programming language (Ex: - Java 11).

Java web server stack: Choose a java web server stack (Ex: - Java SE (Embedded web server)).

Operating system: Choose an operating system (Ex: - Linux).

Region: Select the region, closest to you (Ex: - East US).

Pricing Plan: Choose a plan type (Ex: - Free F1).



Home > App Services >

Create Web App

Basics Deployment Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

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

Subscription *

Resource Group *
[Create new](#)

Instance Details

Need a database? [Try the new Web + Database experience.](#)

Name * .azurewebsites.net

Publish * Code Docker Container Static Web App

Runtime stack *

Java web server stack *

Operating System * Linux Windows

Region *

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

Home > App Services >

Create Web App

Subscription *

Resource Group *
[Create new](#)

Instance Details

Java web server stack *

Operating System * Linux Windows

Region *
i Not finding your App Service Plan? Try a different region or select your App Service Environment.

Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Linux Plan (East US) *
[Create new](#)

Pricing plan * **Free F1**
1 GB memory
[Change size](#)

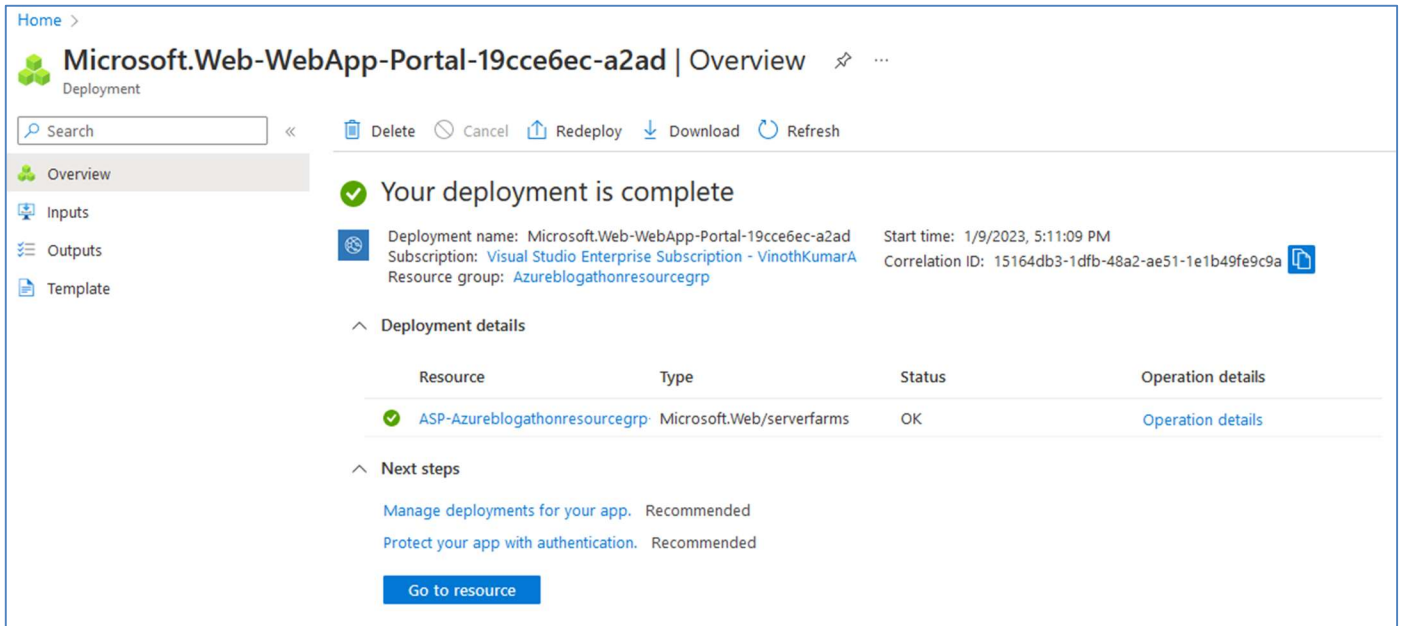
Zone redundancy

An App Service plan can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make an App Service plan zone redundant after it has been deployed [Learn more](#)

Zone redundancy Enabled: Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be three.
 Disabled: Your App Service Plan and the apps in it will not be zone redundant. The minimum App Service plan instance count will be one.

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

Once the deployment is successful, you will see this screen.



Step 2: Deploying the code to Azure Functions

```
package com.function;

import com.microsoft.azure.functions.annotation.*;
import com.microsoft.azure.functions.*;
import java.util.*;

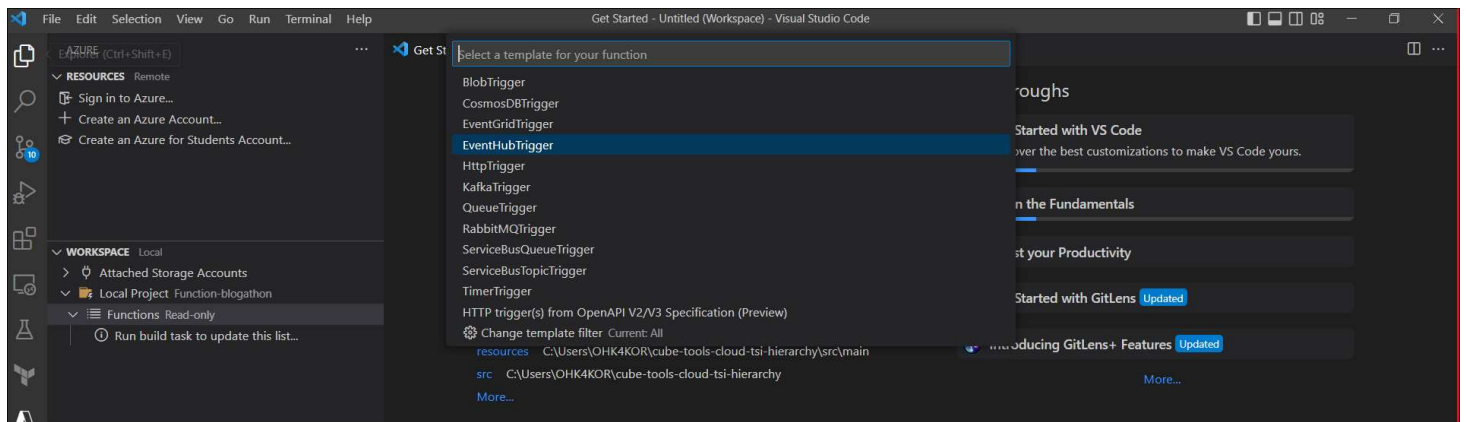
/**
 * Azure Functions with Event Hub trigger.
 */
public class EventHubTriggerJava1 {
    /**
     * This function will be invoked when an event is received from Event Hub.
     */
    @FunctionName("EventHubTriggerJava1")
    public void run(
        @EventHubTrigger(name = "message", eventHubName = "Name of your eventhub",
        connection = "FUNCTIONS_WORKER_RUNTIME", consumerGroup = "$Default", cardinality =
        Cardinality.ONE) String message,
```

```
@CosmosDBOutput(
    name = "databaseOutput",
    databaseName = "Cosmos DB name",
    collectionName = "Cosmos DB collection name",
    connectionStringSetting = "CosmosDBConnectionString")
    OutputBinding<String> document,
    final ExecutionContext context
) {
    context.getLogger().info("Java Event Hub trigger function executed.");
    context.getLogger().info(message);
}
}
```

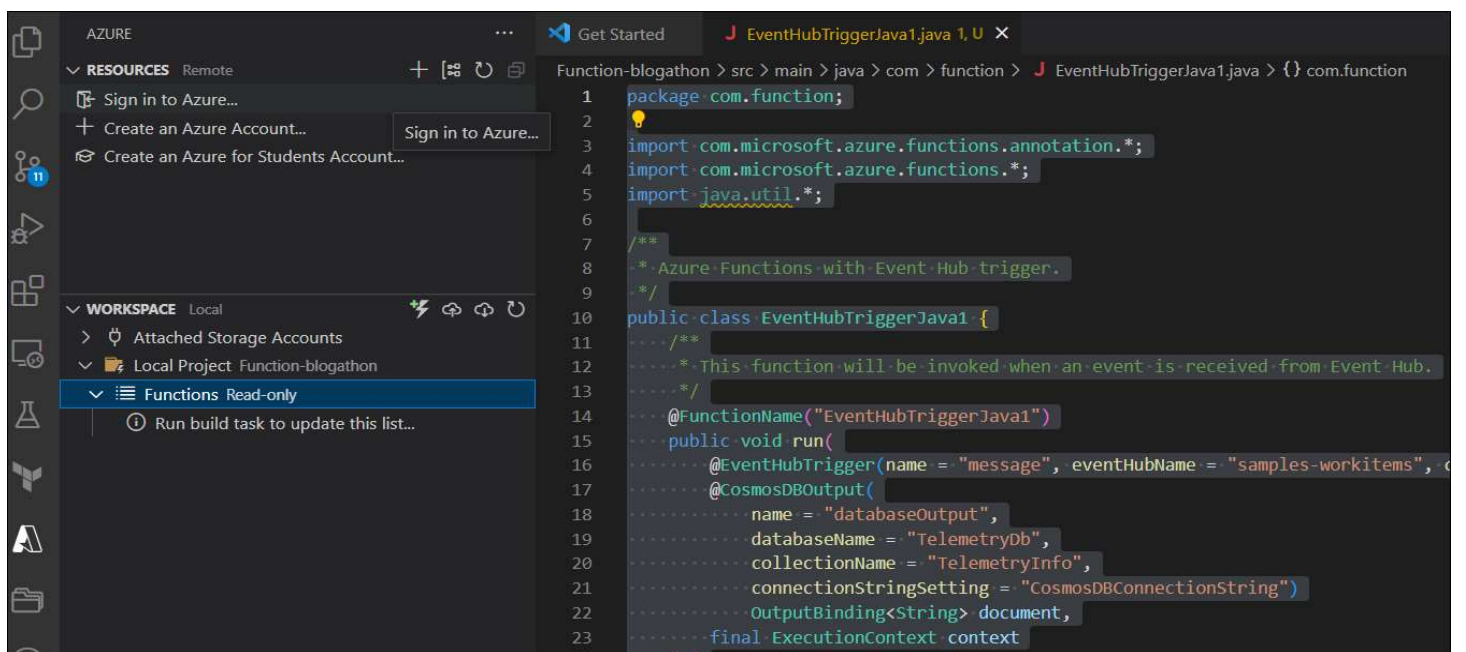
Note:- Please go through reference (1. **Deploying Azure Functions**) for more details.

Creating the function locally

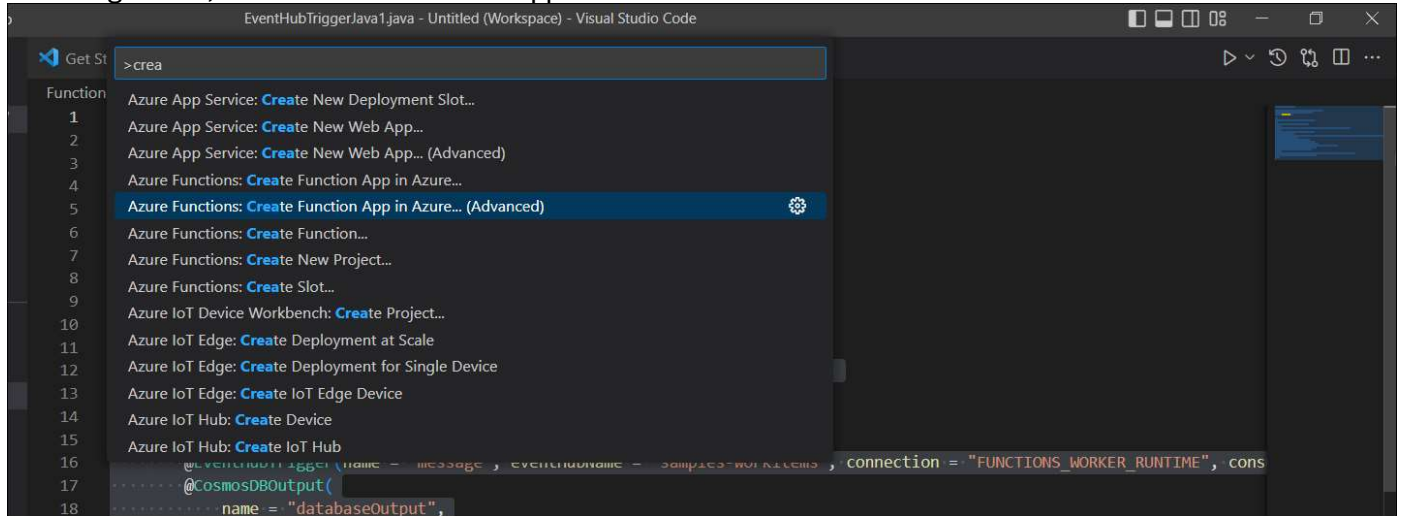
Click on you workspace's "+" icon to create a duction with Azure Event hub trigger



Connect your VS code to Azure by signing in.



Once Signed in , click on create function app.



You will need to fill in the following details:

Subscription: Select the subscription to use for your hub(Ex:- Visual studio enterprise subscription).

Resource group: Select a resource group or create a new one(Ex:- Azureblogathonresourcegrp).

FuctionApp name: Enter a name for your FcuntionApp (Ex:- Azureblogathon)

Publish: : Option to publish code files or a Docker container(Ex:- code).

Runtime stack: Choose a runtime that supports your favorite function programming language(Ex:- Java).

Version: Choose the version of your installed runtime Ex:- 11).

Location: Select the region, closest to you(Ex:- East US).

Plan: Choose a plan type(Ex:- Serverless).

Finally, deploy the local function to azure.

```

3 import com.microsoft.azure.functions.annotation.*;
4 import com.microsoft.azure.functions.*;
5 import java.util.*;
6
7 /**
8  * Azure Functions with Event Hub trigger.
9  */
10 public class EventHubTriggerJava1 {
11     /**
12     * This function will be invoked when an event is received from Event Hub.
13     */
14     @FunctionName("EventHubTriggerJava1")
15     public void run(
16         @EventHubTrigger(name = "message", eventHubName = "samples-workitems", connection =
17         @CosmosDBOutput(
18             name = "databaseOutput",
19             dbName = "TelemetryDb",
20             collectionName = "TelemetryInfo",
21             connectionStringSetting = "CosmosDBConnectionString")
22         OutputBinding<String> document,
23         final ExecutionContext context
24     ) {
25         context.getLogger().info(msg: "Java Event Hub trigger function executed.");

```

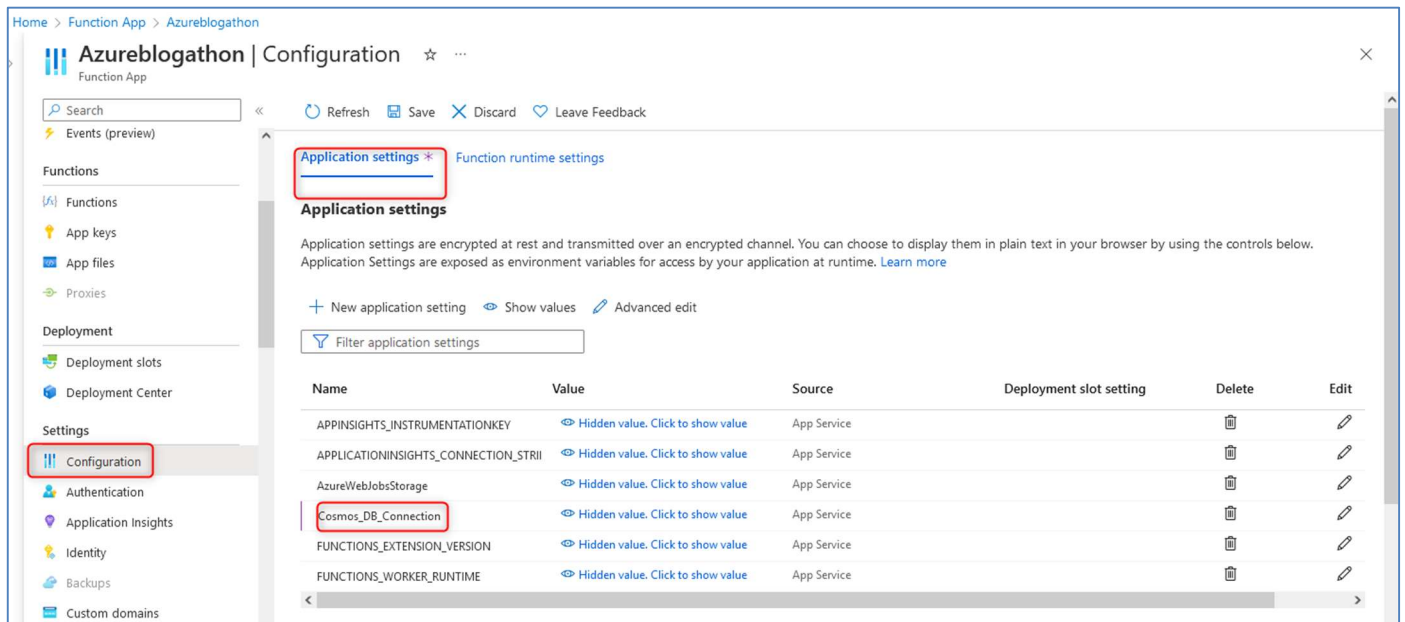
Step 3: Establishing Function App connection with the Cosmos DB

To enable the function app to connect with the Azure Cosmos DB

Once the database is created, create a container for saving the telemetry data.

All we need to do is update the database connection string in the Application setting of the function.

Since our function can connect with the Azure Cosmos DB, we can query it directly.



Step 4: Configuring Data Bricks cluster

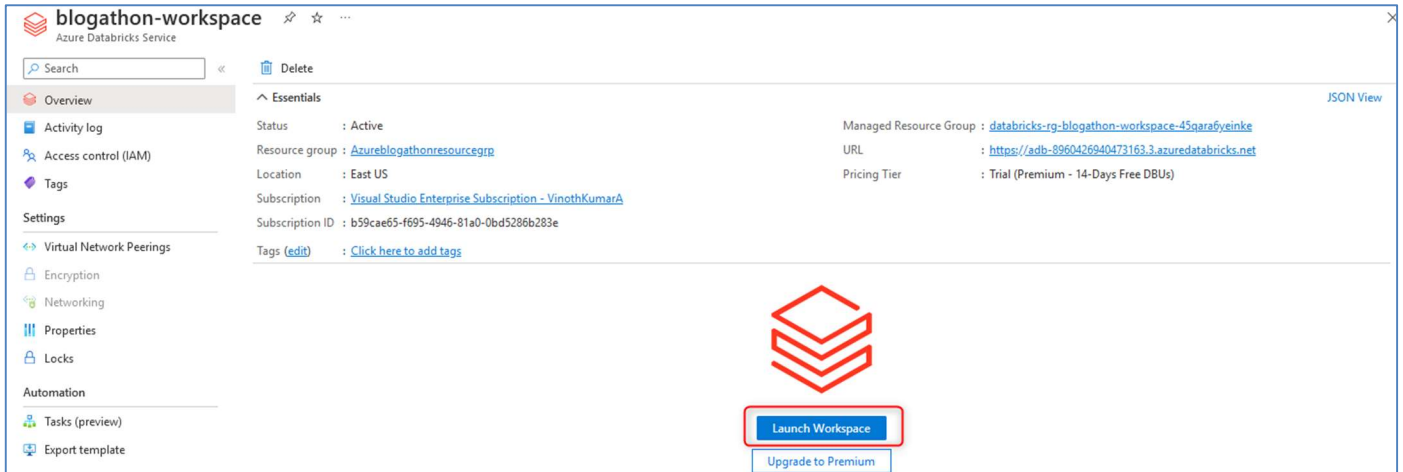
Perform Batch and Stream Processing with Delta Lake

You need to have access to an Azure Databricks workspace to perform Structured Streaming with batch jobs by using Delta Lake

Create A Cluster

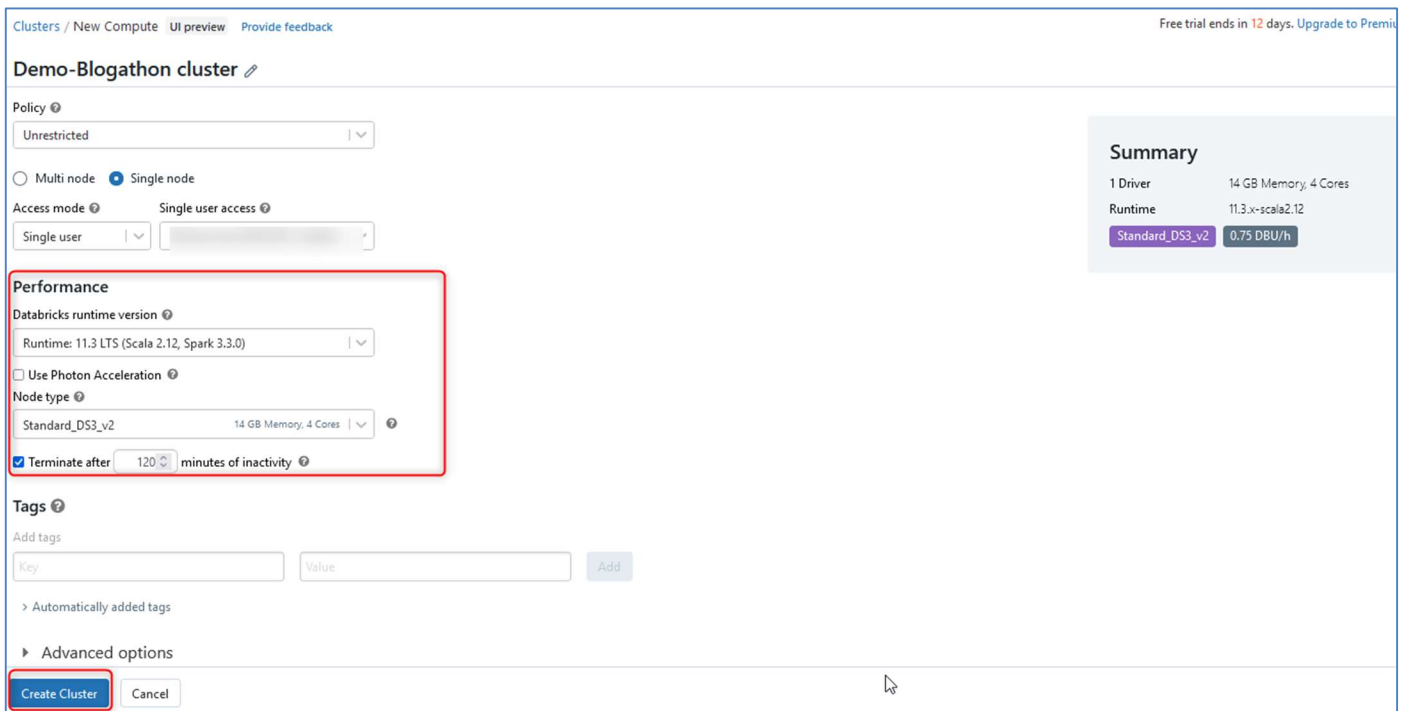
We have already deployed Databricks workspace in step1.

1) Click on the button Launch Workspace to open your Databricks workspace in a new tab.



2) In the left-hand menu of your Databricks workspace, select Clusters.

3) Select Create Cluster to add a new cluster.



4) Select **Create Cluster**.

Perform Batch and Stream Processing

1) After the cluster is created, in the left pane, select **Workspace > Users**, and select your username (the entry with the house icon).

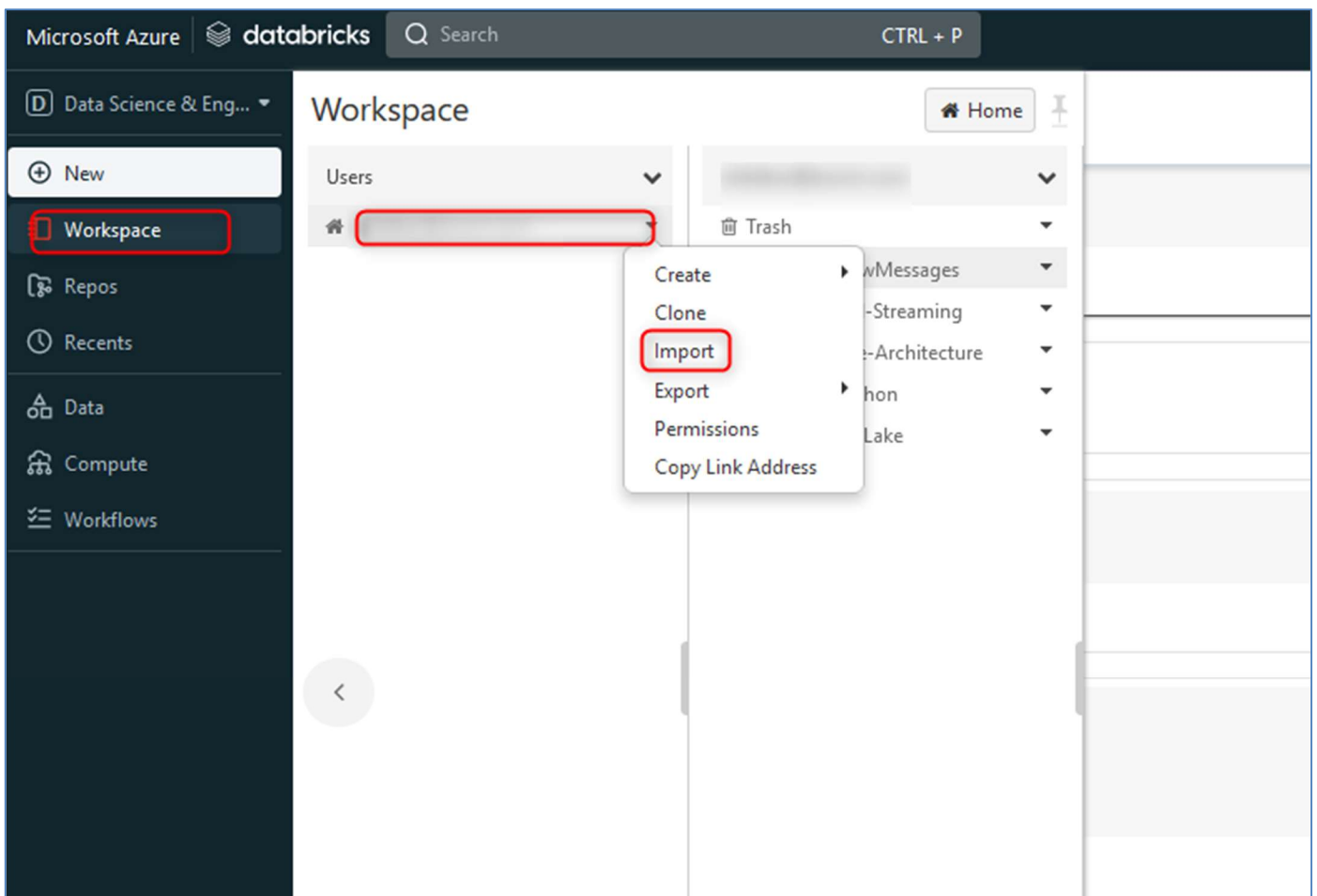
2) In the pane that appears, select the arrow next to your name, and select **Import**.

<https://github.com/MicrosoftLearning/DP-203-Data-Engineer/raw/master/Allfiles/microsoft-learning-paths-databricks-notebooks/data-engineering/DBC/10-Structured-Streaming.dbc>

<https://github.com/solliancenet/microsoft-learning-paths-databricks-notebooks/blob/master/data-engineering/DBC/11-Delta-Lake-Architecture.dbc>

The above generic notebooks can help you get started with data analytics ; you can customize the notebook as per you need.

Note:- Please go through reference (2. **Data Bricks configuration**) for more details.



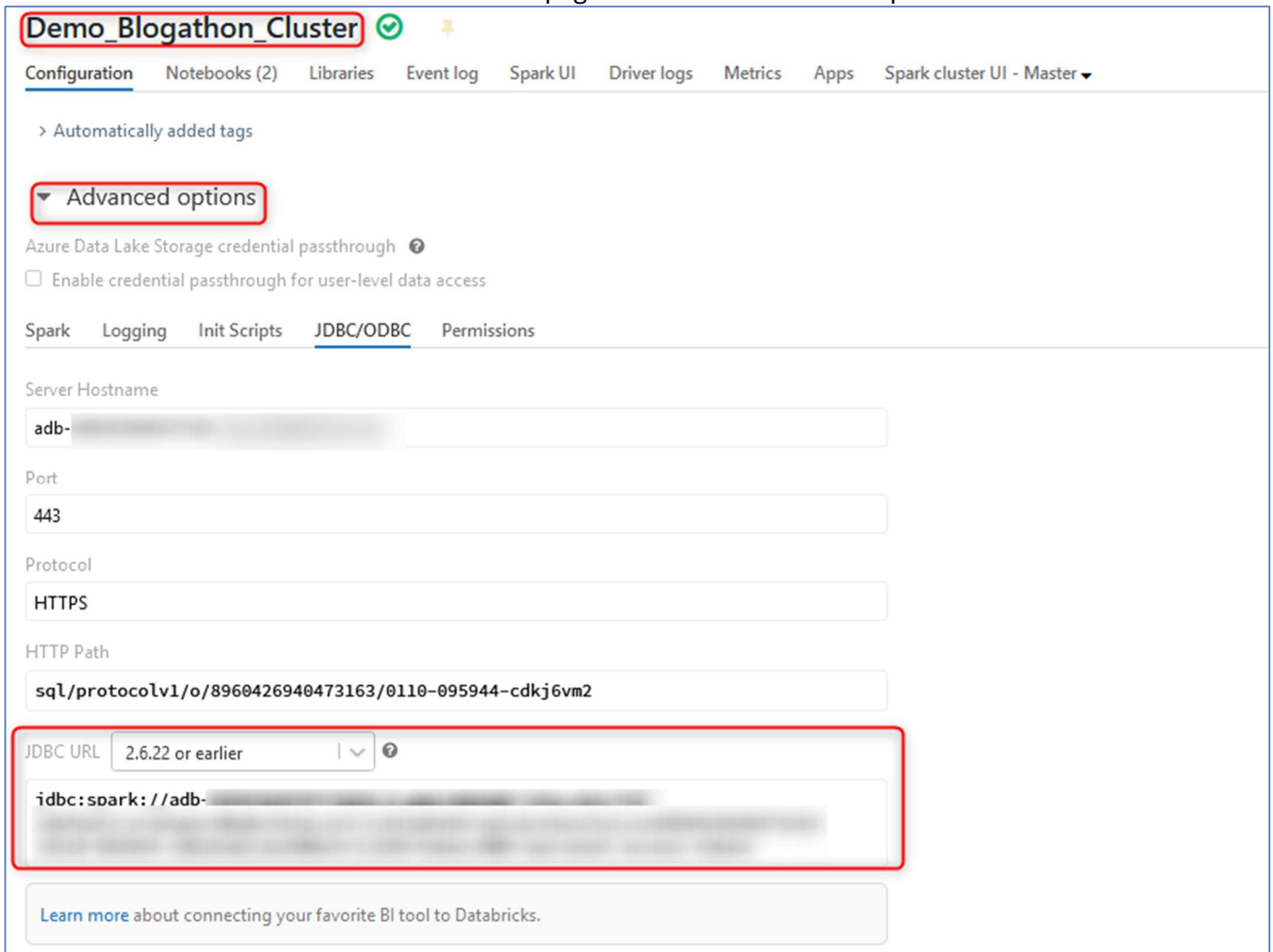
Step 5: Testing the workflow by simulating the telemetry data and visualizing the data in Power BI

Set up Power BI Databricks Integration

Step 1: Get Databricks Connection Information

The first step involved in Power BI Databricks Integration requires you to extract some details about the Databricks connection such as Server Name and HTTP Path. Follow the steps given below to do so:

- Log in to the Microsoft Azure portal using the appropriate credentials.
- Navigate to the left menu bar and click on the Clusters option.
- Scroll down the Clusters page and click on Advanced Options.



The screenshot shows the Azure Databricks Clusters page for a cluster named "Demo_Blogathon_Cluster". The "Advanced options" section is expanded, showing the following configuration:

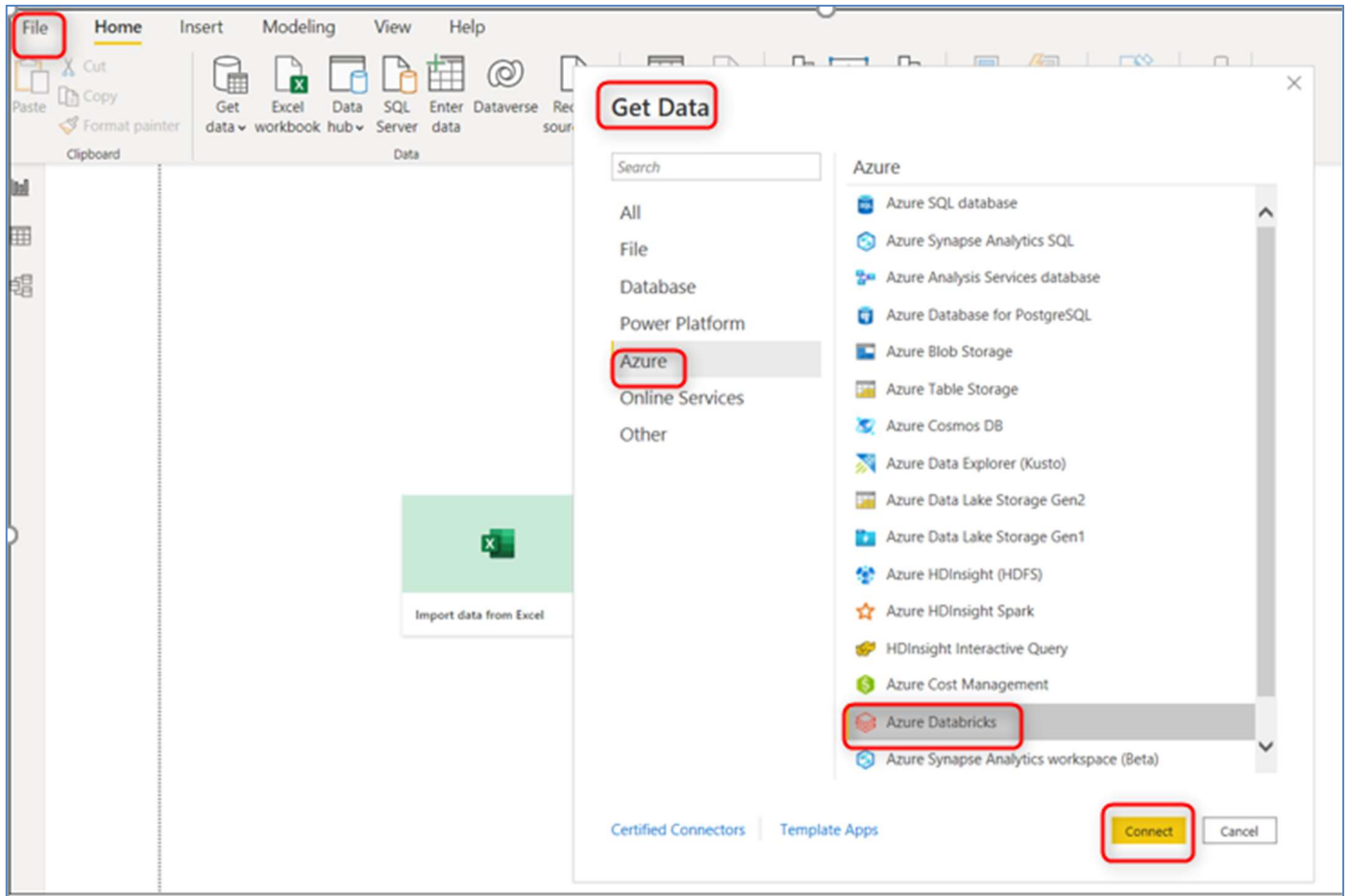
- Server Hostname: adb-...
- Port: 443
- Protocol: HTTPS
- HTTP Path: sql/protocolv1/o/8960426940473163/0110-095944-cdkj6vm2
- JDBC URL: 2.6.22 or earlier |

A red box highlights the JDBC URL field, which contains the URL "jdbc:spark://adb-...". Below the JDBC URL field, there is a link that says "Learn more about connecting your favorite BI tool to Databricks."

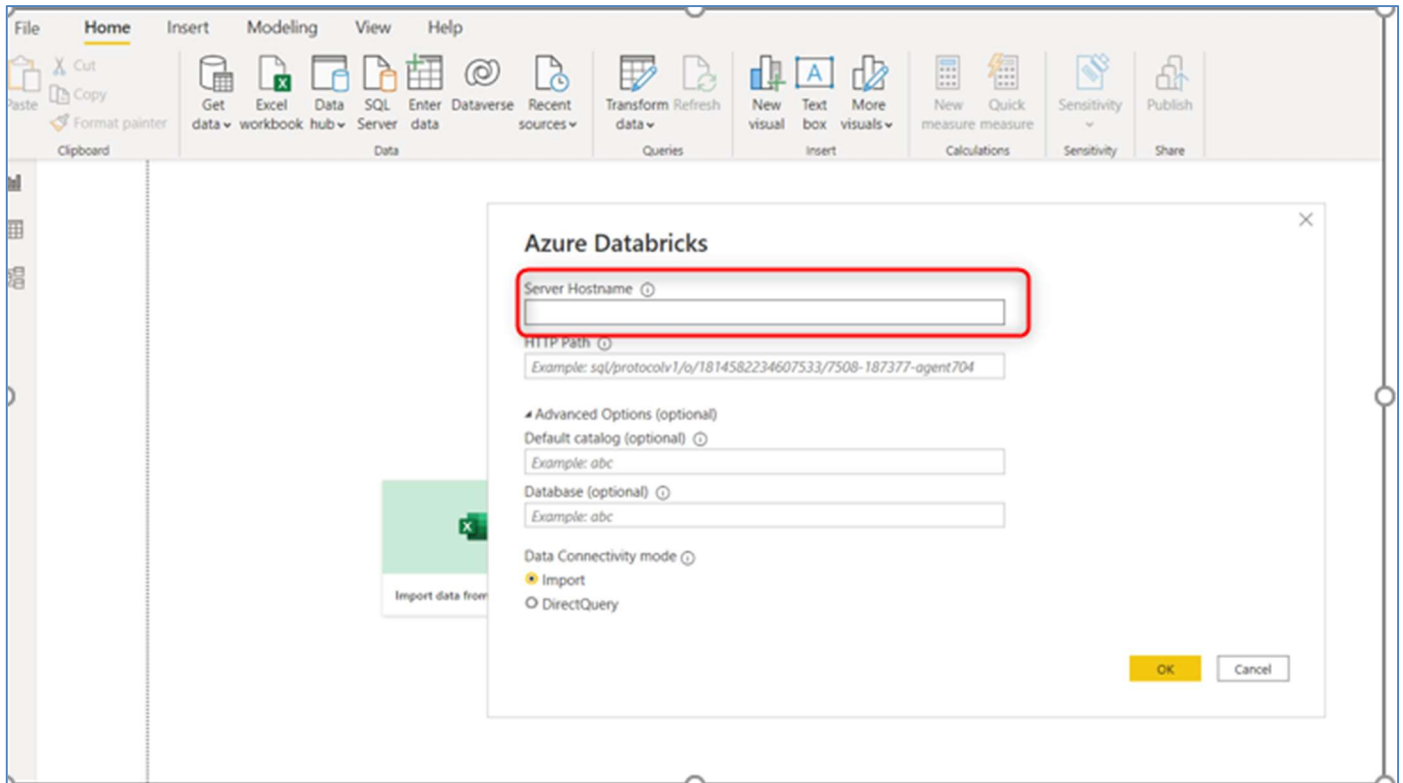
Step 2: Configure Azure Databricks Connection in Power BI

Once you have retrieved the connection information, you need to configure the Databricks connection in Power BI to set up Power BI Databricks Integration:

- Open Microsoft Power BI Desktop.
- Once you open Power BI Desktop, you will see a bunch of options on top of the home page. Click on the Get Data option.
- From the Get Data dialog box, select the Azure option and then click on Azure Databricks Connector to set up Power BI Databricks Integration.



Now, click on the **Connect** button and paste the **Server Name** and **HTTP Path** that you retrieved in the first step.



In terms of Data Connectivity Mode, you will have the following two options:

Import: If you select the Import option, Power BI Desktop will make use of the imported data whenever you create, design, or interact with a visualization. To see underlying data changes that have occurred since the initial import or the most recent refresh, you must refresh the data, which will re-import the entire dataset.

Direct Query: Selecting the Direct Query option will allow you to view the current data at any instant.

- At the Authentication Prompt, enter your Azure account credentials and click on the Connect button. If you want to authenticate using the token, you can click here to learn how to retrieve it and build the Power BI Databricks Integration.
- Finally, select the Databricks data that you wish to query from the Power BI Connector.

Once you follow the above steps in the correct sequence, you will be able to establish Power BI Databricks Integration and create visualizations.

Knowledge Sharing and Best Practices:

- ✓ Using system-assigned managed identity prevents the usage of keys within code which is a bad practice for production environments.
- ✓ The authorization level “anonymous” for a function app is not recommended for the production environment.

Challenges Faced:

- The automation was tricky to implement due to the multiple components involved. Integrating it with Azure Cosmos DB, creating an Event hub-triggered function using Java.
- Secondly, the data bricks notebook for Delta Lake architecture. was the most difficult part to implement.

Business Benefits:

- ✓ We have utilized 100% of Azure PaaS components in implementing solution to reduce the administrative (Scalability and Availability) overhead.
- ✓ Develop Once Deploy Multiple times on different IOT based data analytics use cases across domains
- ✓ Out of the box Azure Data Bricks services helped in implementing Data Analytics solution with ease and simplified the Data Aggregation process.
- ✓ The insights from the solution will help us in identifying real time faults in devices.
- ✓ The visualization dashboard will help in performing predictive maintenance and detailed service level reporting for the end customers.

References:

1. **Deploying Azure Functions:**
<https://learn.microsoft.com/en-us/azure/azure-functions/create-first-function-vs-code-java>
2. **Data Bricks configuration: -**
https://microsoftlearning.github.io/DP-203-Data-Engineer/Instructions/Labs/LAB_11_stream_with_azure_databricks.html
- 2.1 <https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/data/stream-processing-databricks>
3. **Connecting to Power BI:** <https://learn.microsoft.com/en-us/azure/databricks/partners/bi/power-bi>