

✅ Create Azure Function Project in VS Code (Palette Method)

📦 Prerequisites

Before you start, make sure you have:

- [VS Code](#)
 - [.NET 8 SDK](#)
 - Azure Functions Core Tools (auto-installed with the Azure Functions extension)
 - These VS Code extensions:
 - ✅ **Azure Functions**
 - ✅ **C#**
-

🚀 Steps to Create Project via Command Palette

◆ 1. Open Command Palette

Press Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac)

◆ 2. Search and Select:

sql

CopyEdit

Azure Functions: Create New Project...

◆ 3. Follow the Prompts:

Prompt	Your Answer
Select folder	Choose or create a new folder
Language	C# (.NET Isolated)
.NET runtime version	net8.0 (or your preferred version)
Template	Event Hub trigger
Function name	StoreTelemetryFunction
Namespace	(press Enter)
Event Hub name	messages/events (IoT Hub default)

Prompt

Your Answer

Setting name for connection IoTHubConnection

Open in new window?

Up to you

☒ VS Code will scaffold your full project, including StoreTelemetryFunction.cs.

Next Steps

1. Add Your Telemetry Model

Create a file TelemetryModel.cs:

csharp

CopyEdit

```
public class TelemetryModel
{
    public string DeviceId { get; set; }
    public double Temperature { get; set; }
    public double Humidity { get; set; }
    public DateTime Timestamp { get; set; }
}
```

2. Install SQL Client

Press Ctrl+Shift+P → “.NET: Add NuGet Package” → enter:

CopyEdit

Microsoft.Data.SqlClient

3. Update Function Code to Insert into SQL

Replace the function body with the code from earlier (ask if you want it again).

4. Update local.settings.json

json

CopyEdit

```
{
```

```
"IsEncrypted": false,
"Values": {
  "AzureWebJobsStorage": "UseDevelopmentStorage=true",
  "FUNCTIONS_WORKER_RUNTIME": "dotnet-isolated",
  "IoTHubConnection": "<Event Hub-compatible string from IoT Hub>",
  "SqlConnectionString": "<Your Azure SQL DB connection string>"
}
}
```

Run It

Press F5 or open Command Palette and choose:

pgsql

CopyEdit

Debug: Start Debugging

It will listen for IoT Hub events and write telemetry to your SQL DB.

Done!

You've created an Azure Function project **entirely through the VS Code UI**, with:

- Event Hub trigger (for IoT Hub)
 - C# isolated worker
 - SQL database insert support
-

Would you like a **GitHub-ready export**, or help setting up **deployment to Azure**?

You said:

Update Function Code to Insert into SQL

ChatGPT said:

Here's how to **update your Azure Function code** in VS Code to **insert IoT telemetry into Azure SQL Database**.

What You're Updating

The default code in `StoreTelemetryFunction.cs` (created by VS Code) reads messages from IoT Hub. You will update it to:

- Parse the incoming JSON message
 - Insert data into SQL using `Microsoft.Data.SqlClient`
-

Step-by-Step Code Update

1. Add using statements

At the top of `StoreTelemetryFunction.cs`:

```
csharp
CopyEdit
using Microsoft.Azure.Functions.Worker;
using Microsoft.Extensions.Logging;
using Microsoft.Data.SqlClient;
using System.Text.Json;
using System.Threading.Tasks;
```

2. Create or Update the Model

Add this in a separate file called `TelemetryModel.cs`:

```
csharp
CopyEdit
public class TelemetryModel
{
    public string DeviceId { get; set; }
    public double Temperature { get; set; }
    public double Humidity { get; set; }
    public DateTime Timestamp { get; set; }
}
```

◆ 3. Update the Function Body

Replace the contents of StoreTelemetryFunction.cs with this:

csharp

CopyEdit

```
public class StoreTelemetryFunction
{
    private readonly ILogger _logger;
    private readonly string _sqlConnectionString;

    public StoreTelemetryFunction(ILoggerFactory loggerFactory)
    {
        _logger = loggerFactory.CreateLogger<StoreTelemetryFunction>();
        _sqlConnectionString = Environment.GetEnvironmentVariable("SqlConnectionString");
    }

    [Function("StoreTelemetryFunction")]
    public async Task Run(
        [EventHubTrigger("messages/events", Connection = "IoTHubConnection")] string message)
    {
        _logger.LogInformation($"Received message: {message}");

        try
        {
            var telemetry = JsonSerializer.Deserialize<TelemetryModel>(message);

            using var conn = new SqlConnection(_sqlConnectionString);
            await conn.OpenAsync();

            var query = @"
```

```
INSERT INTO Telemetry (DeviceId, Temperature, Humidity, Timestamp)
VALUES (@DeviceId, @Temperature, @Humidity, @Timestamp);
```

```
using var cmd = new SqlCommand(query, conn);
cmd.Parameters.AddWithValue("@DeviceId", telemetry.DeviceId);
cmd.Parameters.AddWithValue("@Temperature", telemetry.Temperature);
cmd.Parameters.AddWithValue("@Humidity", telemetry.Humidity);
cmd.Parameters.AddWithValue("@Timestamp", telemetry.Timestamp);

await cmd.ExecuteNonQueryAsync();
}
catch (Exception ex)
{
    _logger.LogError($"Error inserting telemetry: {ex.Message}");
}
}
}
```

◆ 4. Ensure Your SQL Table Exists

Run this on your Azure SQL Database:

sql

CopyEdit

```
CREATE TABLE Telemetry (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    DeviceId NVARCHAR(100),
    Temperature FLOAT,
    Humidity FLOAT,
    Timestamp DATETIME
);
```

◆ 5. Set Up local.settings.json

In your project root:

json

CopyEdit

```
{  
  "IsEncrypted": false,  
  "Values": {  
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",  
    "FUNCTIONS_WORKER_RUNTIME": "dotnet-isolated",  
    "IoTHubConnection": "<Event Hub-compatible connection string>",  
    "SqlConnectionString": "<Azure SQL DB connection string>"  
  }  
}
```

▶ Test It

- Run the function: F5 in VS Code
- Send a telemetry JSON message to IoT Hub
- Check Azure SQL DB → table Telemetry for new rows

Need help testing with a simulated device or setting up deployment? Let me know — I can walk you through that too.