


Audit Access Expires Jun 18, 2019

You lose all access to this course, including your progress, on Jun 18, 2019.

Activities

Activities

Instructions for the activities that must be performed during this challenge are presented below. You will need to complete all the activities in this topic and submit the noted deliverables (each earmarked by " DELIVERABLE" below) for submitting your work for grading.

Architecture

The following activities establish the architecture of the solution. You will use the deliverables from this section as a specification to create the solution in the subsequent sections.

1. Use the site below to obtain your Student ID. You can use your user name for edX (i.e. **first_last**) or some ID. It does not have to be an email, etc.

User Registration

EdxId



dennislamcv

UId

00434C67

[Contact us](#) [Privacy & cookies](#) [Terms of use](#) [Trademarks](#) [About](#)

Note: Write down your ID - you will need it throughout the course. If you need to retrieve it, entering the same edX ID above will return the same code.

2. Review requirements as outlined in the previous topic titled "Scenario". The tasks below should fulfill all the requirements of the scenario.
3.  **DELIVERABLE:** Create an Architecture Diagram for a solution that satisfies the requirements (use PowerPoint or Visio) and save it as **WeatherStationArchitecture-[YOUR STUDENT ID]** in the **Lab1** folder within your GitHub repository (ensure you add, commit and push your changes).
 - List all systems and subsystems chosen for the solution.
 - Explain why the subsystem was chosen.
4.  **DELIVERABLE:** Create a threat model and note any key factors that drive configuration of the solution.

- Save the threat model as **WeatherStationThreatModel-[YOUR STUDENT ID]** in the **Lab1** folder within your GitHub repository (ensure you add, commit and push your changes)

Tip: Review the following materials:

- **Course:** *DEV301x IoT Architecture Design and Business Planning*
- **Module:** *Understanding the Azure IoT Reference Architecture*
- **Lab:** *Reference Architecture SubSystems and Security*
- **Topics:** *Threat Modeling the Azure IoT Reference Architecture and Microsoft Threat Modeling Tool 2016 Review*

Provisioning the Azure Resources

Provision the Azure Resources you specified within the Architecture Diagram. Utilize the following naming conventions:

- All resources to be located within a single resource group: **IoTCapstoneRG**
- All resources will follow a naming convention of:

`iot-[resource mnemonic]-[studentid]`

For example:

`iot-hub-12345678`

Here is a list of resources and the required mnemonics which should be used **unless** an explicit name for a resource is provided:

Resource	Mnemonic
Azure Function	func
Azure Logic App	logic
Azure SQL Server	ss

Resource	Mnemonic
Azure Stream Analytics	asa
Blob Storage	blob
Cosmos DB	cosmos
Data Lake Analytics	dla
Data Lake Storage	dls
Device Provisioning Service	dps
IoT Hub	hub
Machine Learning	ml
Power BI	pbi
Time Series Insights	tsi
Web App	app

1. Create a resource group named: **IoTCapstoneRG**.
2. Create all resources identified in the architecture diagram you created earlier.
 - The resources must be created within the **IoTCapstoneRG** resource group.
 - Use the naming convention as specified above.
 - Configure the resources to meet the requirements.

Important: You will not receive credit for your work if you do not follow the naming conventions.

Preparing the Device and Connecting to Azure

1. Configure Device and connect to Azure

Tip: Review the following materials:

- **Course:** *DEV325x Introduction to Device Programming for IoT: C Edition*
- **Module:** *Data and Device Inputs*
- **Lab:** *Configure the MXChip Development Environment*

and:

- **Course:** *DEV297x IoT Device Configuration and Communication: C Edition*
- **Module:** *Manage Your Devices*
- **Lab:** *Automating Device Configuration and Management*
- Connect the device to WiFi
- Configure the device development environment.

Tip: Reference [Getting Started with the MXChip IoT DevKit](#) for guidance.

2. Update the device software

- Import the [MXChip.IoT.Capstone.Library](#) into the Arduino development environment as detailed here - [Importing a .zip Library](#)
- Extract the Weather Station Simulator source code from the [WeatherStationSimulator.zip](#) file.
 - You will need to update the paths in the `includePath` property within the **c_cpp_properties.json** file to your local paths in order to compile and upload the application to the device. Use the paths included in the file for guidance on where to look for your local paths.
 - You must update the Student ID constant value with your ID in **device.ino**:

```
enum WindSpeedStatus
{
    Normal,
    Strong,
    Dangerous
};

// Update Student ID
static const char *_studentId = "ABCDEFGHGIJK";
static bool _isConnected = false;
static uint64_t _sendIntervalMs = 30 * 1000;
static uint64_t _lastSentMs = 0;
```

Configure Azure Services, Connect Your Device, and Send Data


1. Configure Device Twin and connect the device to IoT Hub

- On the IoT Hub device settings, configure the device twin by adding `windSpeedStatus` and `sendFrequencySeconds` to the **desired** properties as shown below:

```
"properties": {
  "desired": {
    "windSpeedStatus": "Normal",
    "sendFrequencySeconds": 5,
    "$metadata": {
      "$lastUpdated": "2019-01-01T02:19:36.7248",
    },
    "$version": 1
  }
},
```

Tip: Review the following materials:

- **Course:** *DEV297x IoT Device Configuration and Communication: C Edition*
- **Module** *Implement Device Communications*

- **Lab:** *Configuring and Securing IoT Hub Devices*
 - **Topic:** *Access Device Twin Properties from the Back End*
 - Connect the device to your IoT Hub - use the name **CWF-MX-001**.
2. Deploy and configure the server side code that will update the device twin properties in accordance with the requirements.
- The source for the Azure function can be found here - [CapstoneAzFunctions.zip](#)
- Tip:** Review the following materials:
- **Course:** *DEV301x IoT Architecture Design and Business Planning*
 - **Module** *PoV and Rollout*
 - **Lab:** *Planning a PoV*
 - **Topic:** *Stream Analytics and Azure Functions* (particularly the sections "Create an Azure Function" and "Create a Stream Analytics Job Output" in this topic)
3. Stream data to the chosen storage and functions.
- Tip:** Use Azure Streaming Analytics to:
- Send data to Data Lake Storage Gen 1. Review the following materials:
 - **Course:** *DEV326x IoT Data Analytics and Storage*
 - **Module** *Getting Started with Data Lake Storage and Analytics*
 - **Lab:** *IoT Analytics and Cold Storage*
 - **Topic:** *Set up a Cold Storage Repository with Azure Data Lake Storage*
 - Send data to the Azure function.
4. Write a query in **Data Lake Analytics** that calculates the average wind speed and temperature for the last hour of telemetry received from the weather station. Submit the following:
- The data set from which the results were calculated, exported as a CSV file (data should contain at least 120 records).
 - The U-SQL that generates the results - see below for starting U-SQL.
 -  **DELIVERABLE:** The output CSV files - **one_hour_of_data.csv** and **avg_temp_and_windspeed.csv**. Save these files in the **Lab1** folder within your

GitHub repository (ensure you add, commit and push your changes).

Tip: You will need to create a U-SQL database called **IoTCapstoneDB** and register the **Newtonsoft.Json.dll** and **Microsoft.Analytics.Samples.Formats.dll** assemblies in order to be able to query the JSON telemetry. You can download the assemblies here: [JsonAssemblies.zip](#). The following script will create the database and register the assemblies - it assumes the assemblies have already been uploaded to an **Assemblies/JSON** folder in the root of your Data Lake Storage. You will need to create these folders in the root of your Data Lake Storage instance if they don't already exist.

```
CREATE DATABASE IF NOT EXISTS IoTCapstoneDB;
USE DATABASE [IoTCapstoneDB];
-- The lines below assume the DLLs have been uploaded to a Data Lake Storage
CREATE ASSEMBLY [Newtonsoft.Json] FROM @"/Assemblies/JSON/Newtonsoft.Json.dll";
CREATE ASSEMBLY [Microsoft.Analytics.Samples.Formats] FROM @"/Assemblies/JSON/Microsoft.Analytics.Samples.Formats.dll";
```

Start with the following U-SQL and add your query to assign the values to `@avgdata` (the section you need to write is noted in the code below in angle brackets <>):


```

REFERENCE ASSEMBLY IoTCapstoneDB.[Newtonsoft.Json];
REFERENCE ASSEMBLY IoTCapstoneDB.[Microsoft.Analytics.Samples.F

USING Microsoft.Analytics.Samples.Formats.Json;

DECLARE @InputPath string = "/telemetry/{date:yyyy}/{date:MM}/{

DECLARE @OutputOneHourFile string = "/output/one_hour_of_data.c
DECLARE @OutputAvgFile string = "/output/avg_temp_and_windspeed

// Extract all data and convert from JSON
@json =
EXTRACT
    date DateTime,
    EventProcessedUtcTime DateTime,
    PartitionId int,
    EventEnqueuedUtcTime DateTime,
    metadata_deviceType string,
    metadata_studentId string,
    metadata_uid string,
    telemetry_temperature double,
    telemetry_pressure double,
    telemetry_humidity double,
    telemetry_windSpeed double,
    telemetry_windDirection double,
    IoTHub_ConnectionDeviceId string
FROM
    @InputPath
USING new MultiLevelJsonExtractor(null,
    true,
    "EventProcessedUtcTime",
    "PartitionId",
    "EventEnqueuedUtcTime",
    "metadata.deviceType",
    "metadata.studentId",
    "metadata.uid",
    "telemetry.temperature",
    "telemetry.pressure",
    "telemetry.humidity",
    "telemetry.windSpeed",

```

```

        "telemetry.windDirection",
        "IoTHub.ConnectionDeviceId"
    );

// Restrict data to last hour
@lastHour =
    SELECT
        *
    FROM
        @json
    WHERE
        EventProcessedUtcTime > (DateTime.UtcNow - TimeSpan.FromHours(1))


// Output intermediate data set for grading
OUTPUT @lastHour
TO @OutputOneHourFile
USING Outputters.Csv(outputHeader:true);

// Determine the average temperature and windspeed for each IoT device
// Output should be 3 columns:
//      IoTHub_ConnectionDeviceId
//      avg_temp
//      avg_windspeed

@avgdata =
    <YOUR SELECT HERE>

// Output averaged values for assessment
OUTPUT @avgdata
TO @OutputAvgFile
USING Outputters.Csv(outputHeader:true);
...

```

5.  **DELIVERABLE:** Once you have created and configured the resources, use the **Automation script** menu option in the **Resource Group** UI to generate and **Download** an Azure Resource Management script that documents the created resources. The downloaded file will be named similar to **ExportedTemplate-IoTCapstoneRG.zip**.

- Rename this file to **ExportedTemplate-IoTCapstoneRG-**

WeatherStation-[YOUR STUDENT ID].zip and save it in the **Lab1** folder within your GitHub repository (ensure you add, commit and push your changes)