Azure IoT Cloud Workshop

Labs

Microsoft

Lab 1:

Deploy remote monitoring solution and do a walk-through

Lab 1 Steps

1. Go to http://www.azureiotsuite.com and create a new RM solution (preview). *[Provision the solution in North Europe.]*
2. Download the open source code from github: https://github.com/Azure/azure-iot-pcs-remote-monitoring-dotnet and extract it onto your computer.
3. Do a walk-through of the deployed RM on the Azure portal.
4. Open the solution in Visual Studio (Code) and do a walk-through.

Lab 2:

Create a physical device and connect it to the Remote Monitoring solution

Lab 2 Steps

1. Add a new physical device in the Remote Monitoring solution.
2. Create the physical device [Ubuntu VM, with nodejs & npm]: https://docs.microsoft.com/en-us/azure/iot-suite/iot-suite-connecting-devices-node
3. Run the physical device and see the result in the Remote Monitoring solution.
4. Use iothub-explorer [https://github.com/azure/iothub-explorer] to see the events sent to IoT-Hub
5. *Extra*: customize the Remote Monitoring solution and deploy it using CLI and Docker Hub: https://github.com/Azure/azure-iot-pcs-remote-monitoring-dotnet/wiki/Developer-Reference-Guide#customization.

Lab 3:
Individual Device Provisioning

Lab 3 Steps

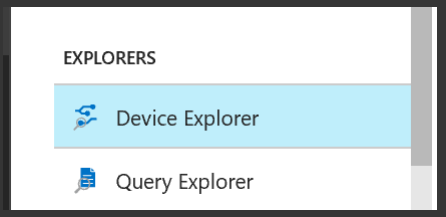
1. Clone the Azure IoT SDK C to your PC:
   git clone https://github.com/Azure/azure-iot-sdk-c.git --recursive
2. Create the Visual Studio solution for the provisioning client (you need cmake on your PC).
   cd azure-iot-sdk-c
   mkdir cmake
   cd cmake
   cmake -Duse_prov_client:BOOL=ON ..
3. Create the X.509 Cert file
   Use: https://docs.microsoft.com/en-us/azure/iot-dps/quick-create-simulated-device-x509#create-a-device-enrollment-entry-in-the-device-provisioning-service
4. Simulate first boot sequence for the device
   Use: https://docs.microsoft.com/en-us/azure/iot-dps/quick-create-simulated-device-x509#simulate-first-boot-sequence-for-the-device

Lab 4:

Working with device twins.

Lab 4 Steps

1. Open the Azure Portal and browse to your IoT Hub
2. Select a simulated device in the Device Explorer



3. Open the device twin

4. Add latitude & longitude to the desired properties

```
6     "desired": {
7       "Latitude": 40.343432,
8       "Longitude": 8.3334,
9       "$metadata": {
```

Lab 4 Steps [2]

5. Open the node.js file of your device in bash
6. Add code in your device to react to changes of the device twin

```
twin.on('properties.desired', function (delta) {
  // Handle desired properties set by solution
  console.log('Received new desired properties:');
  console.log(JSON.stringify(delta));
  // update location
  reportedProperties.Latitude = delta.Latitude;
  reportedProperties.Longitude = delta.Longitude;
  // Send updated properties
  twin.properties.reported.update(reportedProperties, function (err) {
    if (err) throw err;
    console.log('twin state reported');
  });
});
```

7. Start your device and change the values in the device twin
8. Open the Remote Monitoring Solution to see the change

Lab 5:

Creating a IoT Hub Job.

Lab 5 Steps

1. Adjust your "physical" device and add a direct method "ReactOnJob"
   - Add "ReactOnJob" to SupportedMethods
   - Add the client.onDeviceMethod for "ReactOnJob"
   - Implement the defined method to update a property.
2. Create a nodejs job "jobService"
   - Adjust the steps in https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-schedule-jobs#schedule-jobs-for-calling-a-direct-method-and-updating-a-device-twins-properties to represent your device and direct method.
3. Run the physical device and device job on 2 separate bash windows to see the output.
4. Use https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-schedule-jobs as your guideline

Lab 6:

Deploy an IoT Edge device.

Lab 6 Steps

1. Create a Ubuntu VM in your resource group to act a an IoT Edge.
2. Deploy and run the IoT Edge using the Linux quick start: https://docs.microsoft.com/en-us/azure/iot-edge/quickstart-linux.
3. Deploy Azure Function as an IoT Edge module: https://docs.microsoft.com/en-us/azure/iot-edge/tutorial-deploy-function.
4. Extra: Create a custom C# module: https://docs.microsoft.com/en-us/azure/iot-edge/tutorial-csharp-module.

Lab 7:

Add Time Series Insights to the Remote Monitoring solution.

Lab 7 Steps

- Create a new Time Series Insights environment in the Azure portal https://docs.microsoft.com/en-us/azure/time-series-insights/time-series-insights-get-started
- Create the IoT Hub event source for your Time Series Insights environment using the Azure portal https://docs.microsoft.com/en-us/azure/time-series-insights/time-series-insights-how-to-add-an-event-source-iothub
- Access your Time Series Insight environment https://insights.timeseries.azure.com/
- Have a look at the Temperature by Device Id. Add pressure to the time series.

Lab 8:

Add a logic app for Enterprise Integration.

Lab 8 Steps

1. Open the Azure Portal and add a Logic App using the following walk-through: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-monitoring-notifications-with-azure-logic-apps
   a) Create a service bus
   b) Add an endpoint and routing rule
      Query string: temperature > ??. (whatever you want to use as trigger)
   c) Create and configure the Logic App
   d) Test it adjusting your “physical” device

```
function generateRandomIncrement() {
    return ((Math.random() * 5) - 1);
}
```