Lab 09

IOT Hub (Python)

Cscie63 Big Data Analytics

Harvard Extension School

IoT Hub

- IoT Hub is used to track devices and messages such as heart monitors, etc. through an Azure Service that is part of their Internet of Things Services
  - Provides reliable and secure bi-directional communications between millions of Internet of Things (IoT) devices and a back end as part of Azure Services.
  - Offers reliable device-to-cloud and cloud-to-device hyper-scale messaging.
  - Enables secure communications using per-device security credentials and access control.
  - Includes device libraries: .NET Java Node.js Python

IoT Hub Python Demo Overview

- This demo shows using AZ's portal, AZ CLI & Python SDKs to:
  - Create an IoT Hub.
  - Register 1 device to the IoT Hub's Identity Registry.
  - Create a simulated device app in Azure's portal to connect to the IoT Hub and simulates sending telemetry messages periodically from the device via the MQTT protocol to the IoT Hub; and
  - Display the received telemetry messages in a tool called IoT Hub Explorer.

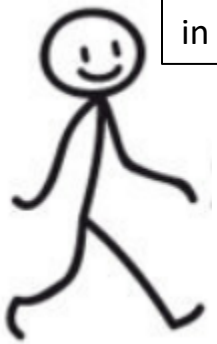
WALK THRU of Demo

Create IoT Hub in AZ CLI or Portal

Register Device in Python

Create 'simulated App' in Python – send telemetry Messages

Receive telemetry messages in IoT Hub Explorer Tool



Demo: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-python-getstarted

My Development Environment

- Windows 10
- Python 3.6.2

```
c:\Users\dhoward>python --version
Python 3.6.2 :: Anaconda, Inc.
```

- AZ CLI 2.0.20
  - C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>az --version
    azure-cli (2.0.20)

Windows

Windows 10
Home

Azure CLI 2.0

Python 3.6.2

Release Date: 2017-07-17

ⓘ Note

The *pip* packages for `azure-iothub-service-client` and `azure-iothub-device-client` are currently available only for Windows OS. For Linux/Mac OS, please refer to the Linux and Mac OS-specific sections on the Prepare your development environment for Python post.

https://github.com/Azure/azure-iot-sdk-python/blob/master/doc/python-devbox-setup.md

Prerequisites

Windows 10

- Python 2.x or 3.x. Use the 32-bit or 64-bit installation as required by your setup. When prompted during the installation, make sure to add Python to your platform-specific environment variable. If you are using Python 2.x, you may need to install or upgrade *pip*, the Python package management system.

- If you are using Windows OS, then Visual C++ redistributable package to allow the use of native DLLs from Python.

- Node.js 4.0 or later. Make sure to use the 32-bit or 64-bit installation as required by your setup. This is needed to install the IoT Hub Explorer tool.

Steps

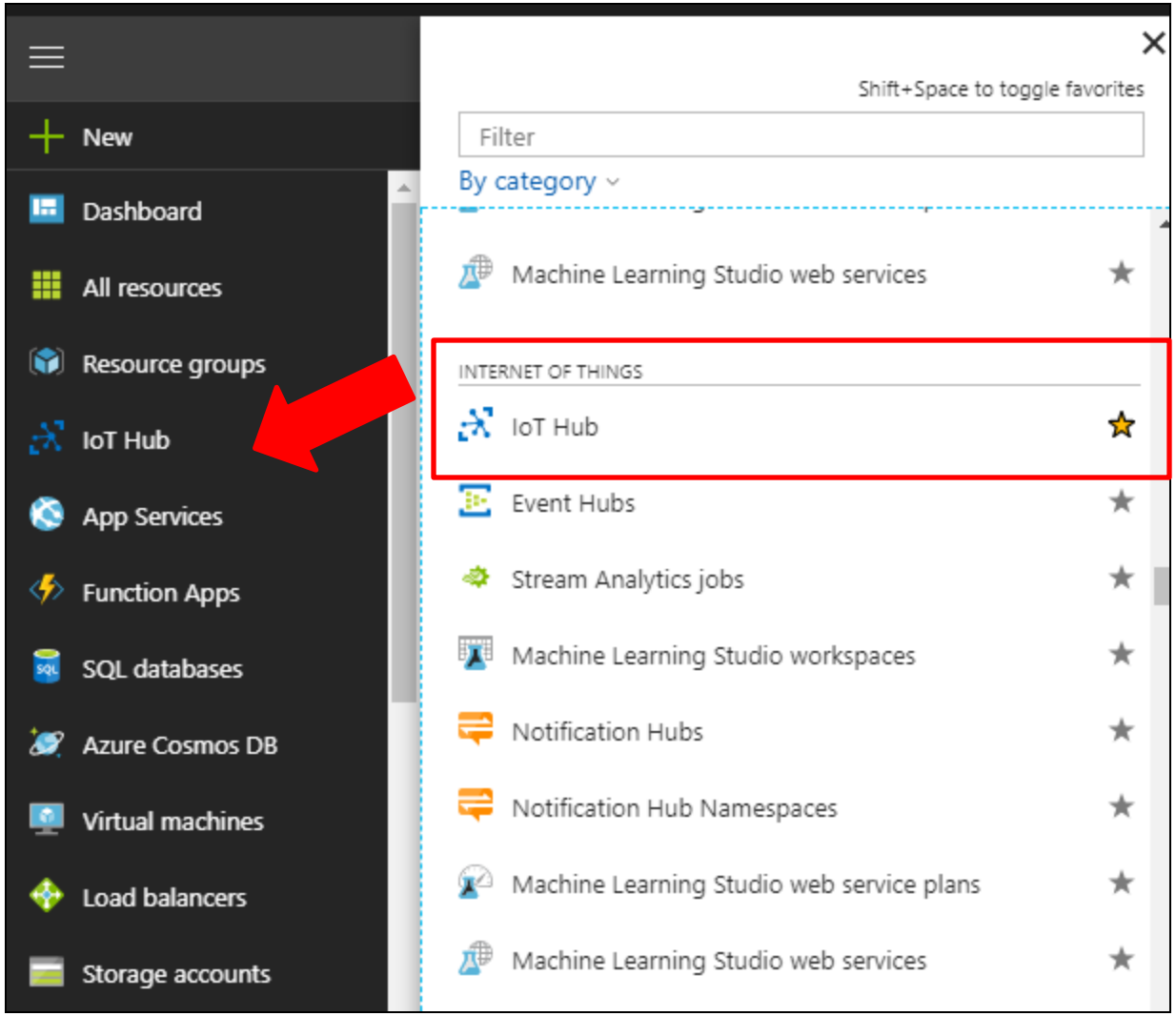
1. Portal or AZ CLI or Power Shell: Create IoT Hub and obtain name, access policy for the Connection string
2. Pip Install: IoT Hub Service Client SDK
3. Python: Register a Device in the Identity Registry
4. Pip Install: azure-iothub-device-client SDK
5. Python: Create an app to simulate a device via an app to send messages to the IoT Hub
6. Install the IoT Hub Explorer
7. Run the IoT Hub Explorer tool to receive messages from the 'Simulated App'/Device

Check my AZ Subscription

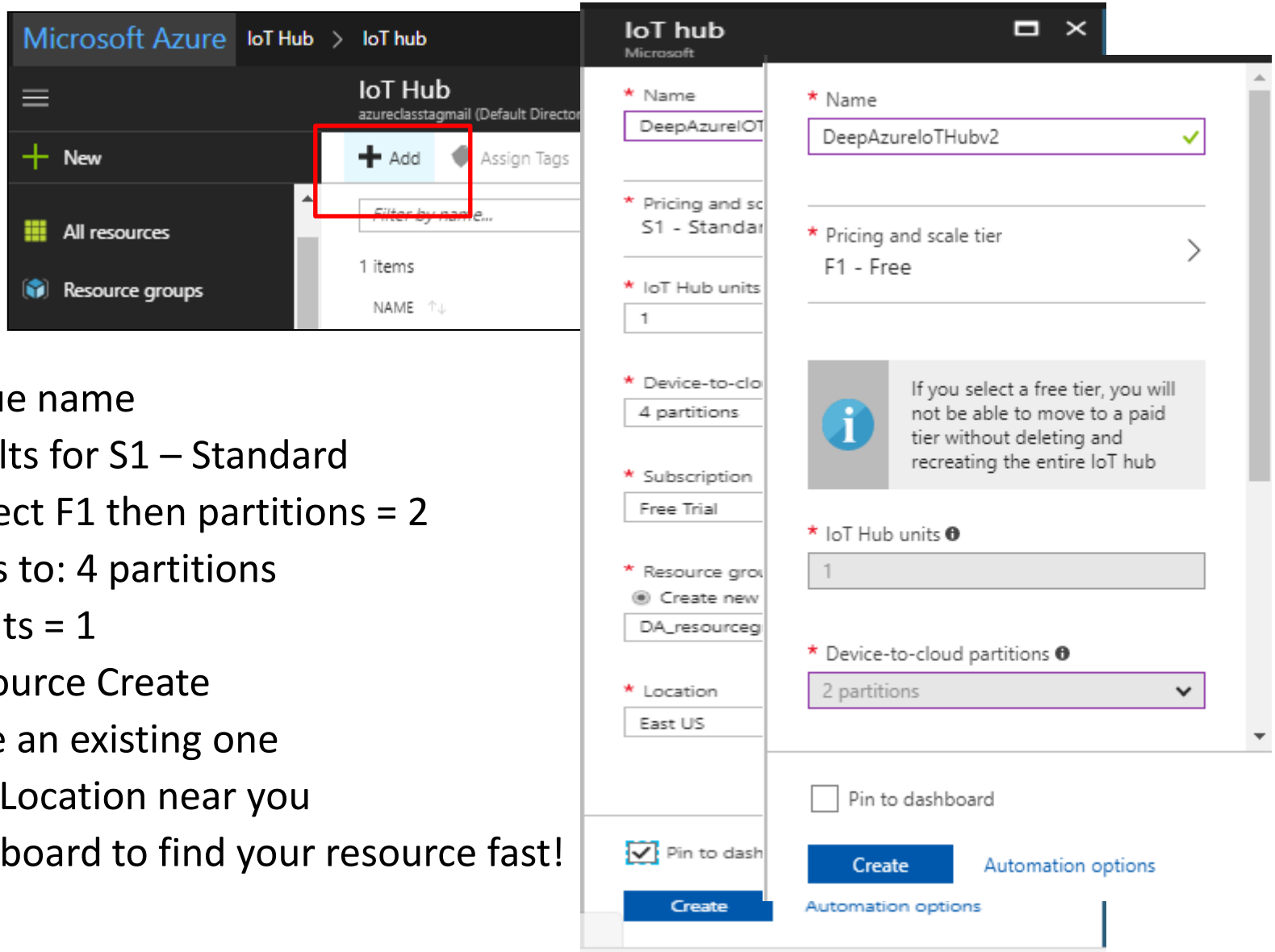
- Since I just opened another free AZ subscription I needed to run> az login to access my new subscription
- Check my default subscription: az account list

```
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>az account list
[
  {
    "cloudName": "AzureCloud",
    "id": "b92c0f6e-486f-4ae9-96af-218ba438580f",
    "isDefault": false,
    "name": "Free Trial",
    "state": "Enabled",
    "tenantId": "2cc424bd-7c70-4ef2-a8a6-e908829fc5d5",
    "user": {
      "name": "gopatsdiane@gmail.com",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "id": "1c17afd3-8b14-43d8-a867-0b5a8bc3b058",
    "isDefault": true,
    "name": "Free Trial",
    "state": "Enabled",
    "tenantId": "483429dc-8beb-463b-a9d8-4b15bc46a3d5",
    "user": {
      "name": "azureclassta@gmail.com",
      "type": "user"
    }
```

Add the IOT Hub Service in Portal display



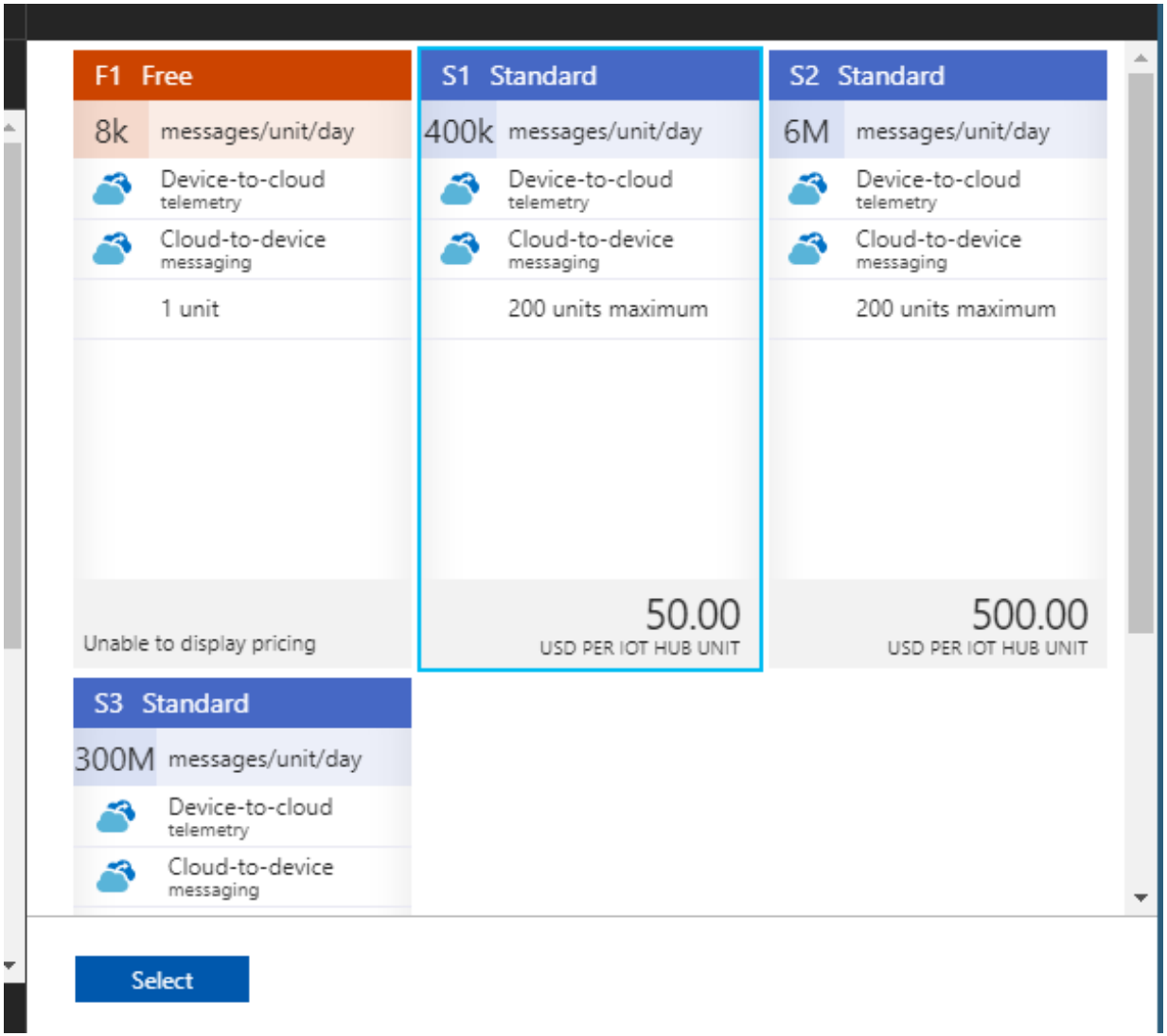
Create an IOT Hub



Steps:

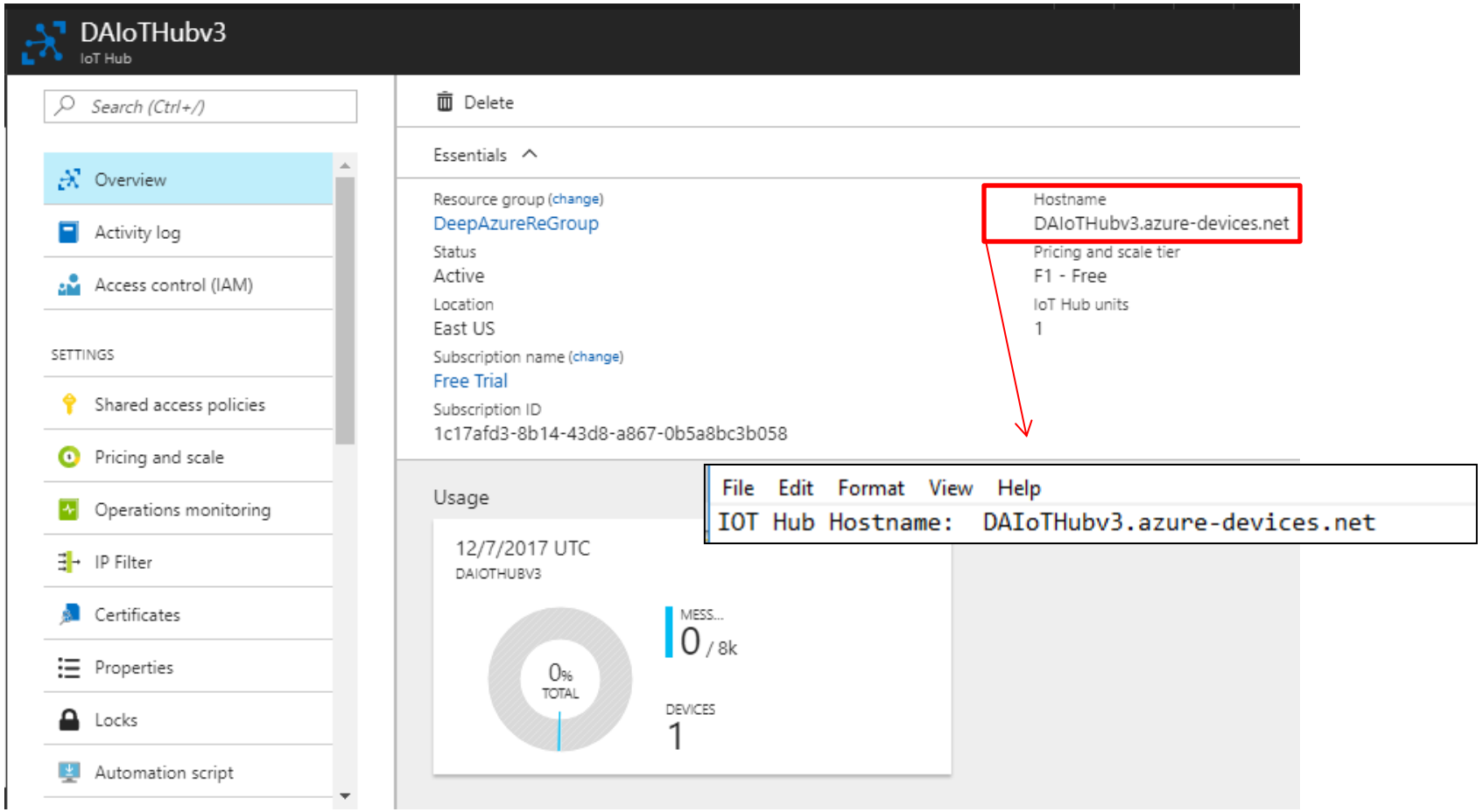
- Enter unique name
- Keep defaults for S1 – Standard
- Note: If select F1 then partitions = 2
- Std defaults to: 4 partitions
- IoT Hub units = 1
- Create Resource Create
  or use an existing one
- Create in a Location near you
- Pin to Dashboard to find your resource fast!

Costs for Creating an IoT Hub



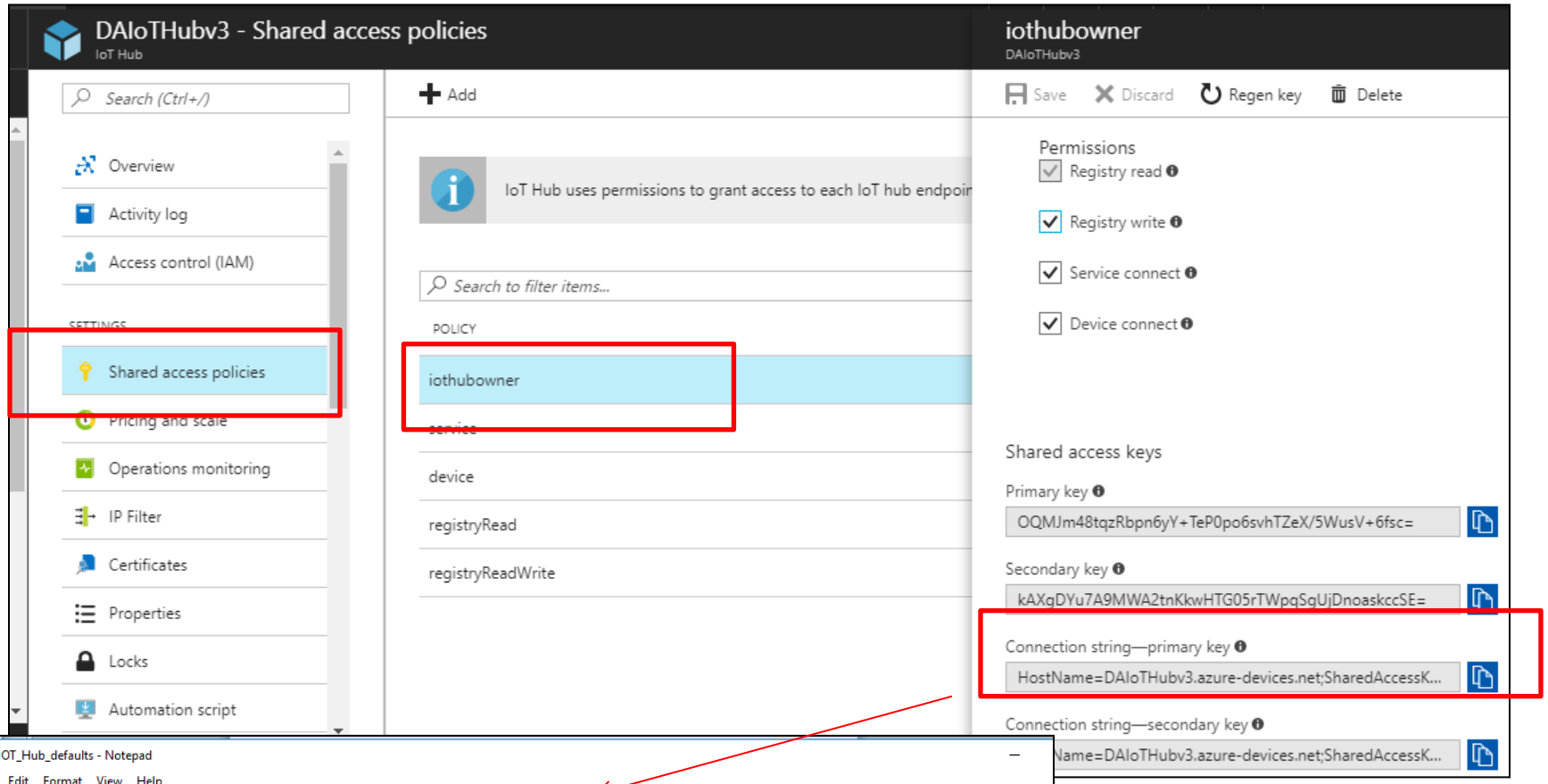
Obtain the Hostname

- Obtain the Hostname
- Save it in Notepad



Obtain the IOT Hub Connection String

- Go to Shared access Policies
- Select iothubowner
- Copy the IoT Hub Connection string to Notepad



AZ CLI IoT

- Manage Internet of Things (IoT) assets using AZ CLI (PREVIEW)
- Documentation:

https://docs.microsoft.com/en-us/cli/azure/iot?view=azure-cli-latest

1. Check for available Resource Groups:

>az group list

OR

>az group list --query "[?location=='eastus']"

2. Create IoT Hub

Azure CLI

```
az iot hub create --name
                  --resource-group
                  [--location]
                  [--sku {F1, S1, S2, S3}]
                  [--unit]
```

>az iot hub create --name DAIoTHubv3 --resource-group DeepAzureReGroup --location eastus --sku F1 --unit 1

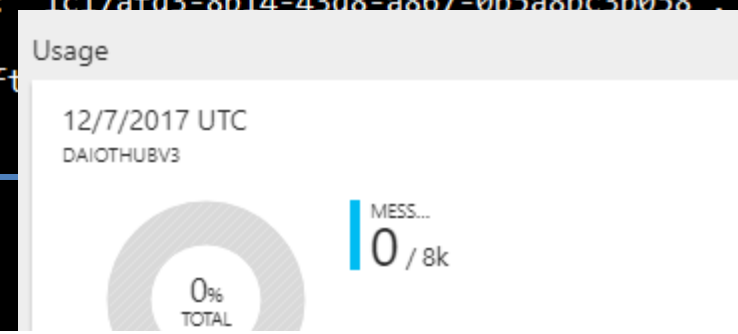
AZ create IoT Hub


```
{- Finished ..
  "etag": "AAAAAAFZygU=",
  "id": "/subscriptions/1c17afd3-8b14-43d8-a867-0b5a8bc3b058/resourceGroups/DeepAzureReGroup/providers/Microsoft.Devices/IotH
ubs/DAIoTHubv3",
  "location": "eastus",
  "name": "DAIoTHubv3",
  "properties": {
    "authorizationPolicies": null,
    "cloudToDevice": {
      "defaultTtlAsIso8601": "1:00:00",
      "feedback": {
        "lockDurationAsIso8601": "0:01:00",
        "maxDeliveryCount": 10,
        "ttlAsIso8601": "1:00:00"
      },
      "maxDeliveryCount": 10
    },
    "comments": null,
    "enableFileUploadNotifications": false,
    "eventHubEndpoints": {
      "events": {
        "endpoint": "sb://ihsuprodblres075dednamespace.servicebus.windows.net/",
        "partitionCount": 2,
        "partitionIds": [
          "0",
          "1"
        ],
        "path": "iothub-ehub-daiothubv3-288196-f83dfcabcd",
        "retentionTimeInDays": 1
      },
```

AZ create IoT Hub - continued


```
    "operationsMonitoringEvents": {
      "endpoint": "sb://ihsuprodblres076dedname
      "partitionCount": 2,
      "partitionIds": [
        "0",
        "1"
      ],
      "path": "iothub-ehub-daiothubv3-288196-35
      "retentionTimeInDays": 1
    }
  },
  "features": "None",
  "hostName": "DAIoTHubv3.azure-devices.net",
  "ipFilterRules": [],
  "messagingEndpoints": {
    "fileNotifications": {
      "lockDurationAsIso8601": "0:01:00",
      "maxDeliveryCount": 10,
      "ttlAsIso8601": "1:00:00"
    }
  },
  "operationsMonitoringProperties": {
    "events": {
      "C2DCommands": "None",
      "Connections": "None",
      "DeviceIdentityOperations": "None",
      "DeviceTelemetry": "None",
      "FileUploadOperations": "Non
      "None": "None",
      "Routes": "None"
    }
  },
```

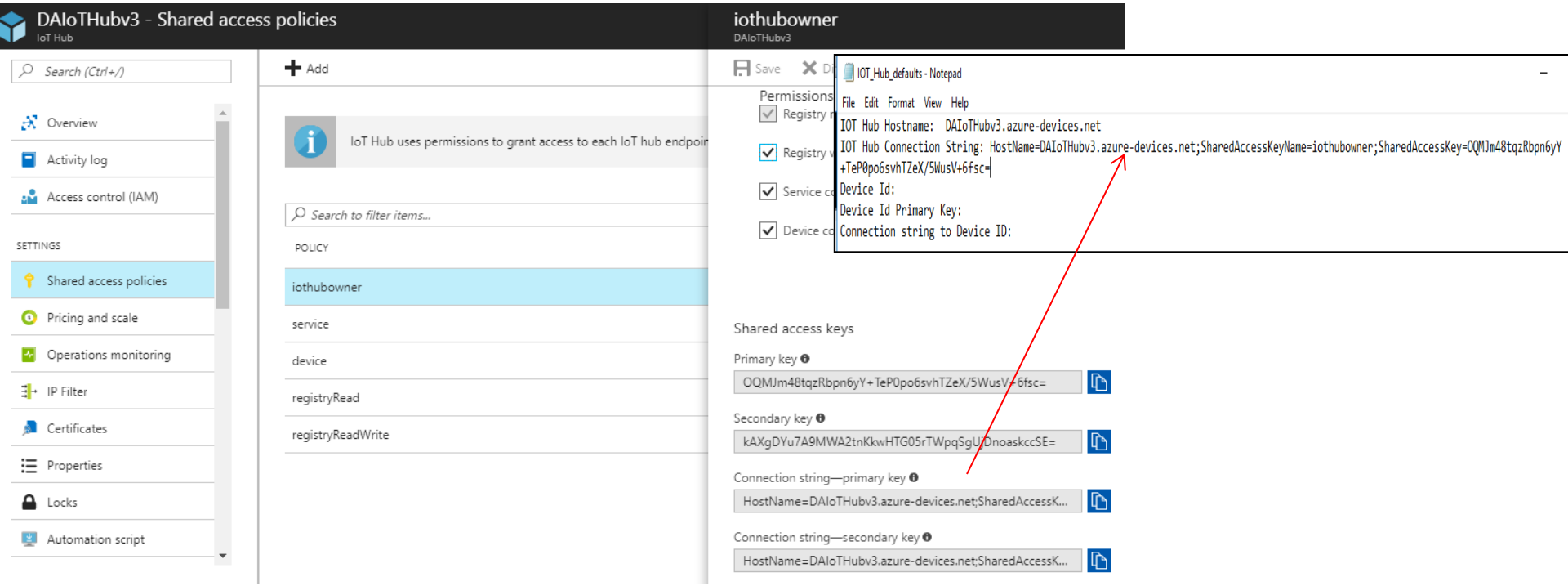
```
    "provisioningState": "Succeeded",
    "storageEndpoints": {
      "$default": {
        "connectionString": "",
        "containerName": "",
        "sasTtlAsIso8601": "1:00:00"
      }
    }
  },
  "resourceGroup": "DeepAzureReGroup",
  "resourcegroup": "DeepAzureReGroup",
  "sku": {
    "capacity": 1,
    "name": "F1",
    "tier": "Free"
  },
  "subscriptionid": "1c17afd3-8b14-43d8-a867-0b5a8bc3b058",
  "tags": {},
  "type": "Microsoft
}
```

HostName=DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=OQMJm48tqzRbpn6yY+TeP0po6svhTZeX/5WusV+6fsc=

AZ IoT Hub Connection string

```
>az iot hub show-connection-string --name DAIoTHubv3
{
  "connectionString": "HostName=DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=OQMJm48tqzRbpn6yY+TeP0po6svhTZeX/5WusV+6fsc="
}
```



AZ IoT Hub Access Policies

>az iot hub policy list --hub-name DAIoTHubv3 --output json

```
C:\Users\dhoward>az iot hub policy list --hub-name DAIoTHubv3 --output json
[
  {
    "keyName": "iothubowner",
    "primaryKey": "OQMJm48tqzRbpn6yY+TeP0po6svhTZeX/5WusV+6fsc=",
    "rights": "RegistryWrite, ServiceConnect, DeviceConnect",
    "secondaryKey": "kAXgDYu7A9MWA2tnKkwHTG05rTWpqSgUjDnoaskccSE="
  },
  {
    "keyName": "service",
    "primaryKey": "9kYQqG/drLRQeG9Wb06orAQNhZFt+F7bhkjn+TFuZ2o=",
    "rights": "ServiceConnect",
    "secondaryKey": "fiFujEPeLPeZ+aBSXHuNcbJbZOtfFjRslg3KnEbHeqs="
  },
  {
    "keyName": "device",
    "primaryKey": "RSc6W1X9KOvyaXNjLGmIziSGVHHU8nXYFU1bXm+w+Cc=",
    "rights": "DeviceConnect",
    "secondaryKey": "C1Fz6FQreh/T3U+Ov2fziajbtRWPvpv0ruhlnG9gYb8="
  },
  {
    "keyName": "registryRead",
    "primaryKey": "JvA+2DLaifE2Ueto/IpwJHLLs1WDS8tejRBZVbpW2uA=",
    "rights": "RegistryRead",
    "secondaryKey": "MWNxa2LzHDwGVGO2/jvMrG5KuDslTds1oKB1gUTdy2U="
  },
  {
    "keyName": "registryReadWrite",
    "primaryKey": "XJS8XyG9LaxAXxl0F3dwL1VVcph7gXvxs3rrdsGClSI=",
    "rights": "RegistryWrite",
    "secondaryKey": "pn6o5cHxzoGLovqTTul7ZpTsIZR7ThJh3N1oC2iwP34="
  }
]
```

Steps

✓ Portal or AZ CLI or Power Shell: Create IoT Hub and obtain name, access policy for the Connection string

2. Pip Install: IoT Hub Service Client SDK
3. Python: Register a Device in the Identity Registry
4. Pip Install: azure-iothub-device-client SDK
5. Python: Create an app to simulate a device via an app to send messages to the IoT Hub
6. Install the IoT Hub Explorer
7. Run the IoT Hub Explorer tool to receive messages from the 'Simulated App'/Device

WALK THRU of Demo

Create IoT Hub in AZ CLI or Portal

Register Device in Python

Create 'simulated App' in Python – send telemetry Messages

Receive telemetry messages in IoT Hub Explorer Tool

Install Azure IoT Hub Service SDK for Python

- In command window run:

  >pip install azure-iothub-service-client

Command Prompt

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\dhoward>pip install azure-iothub-service-client
Collecting azure-iothub-service-client
  Downloading azure_iothub_service_client-1.1.0.20-py3-none-win_amd64.whl (276kB)
    100% |                                        | 276kB 1.2MB/s
Installing collected packages: azure-iothub-service-client
Successfully installed azure-iothub-service-client-1.1.0.20
```

- If azure-iothub-service-client is already installed then you will receive this message:

```
C:\Users\dhoward>pip install azure-iothub-service-client
Requirement already satisfied: azure-iothub-service-client in c:\users\dhoward\anaconda3_64bit\lib\site-packages
```

IoT Hub's Identity Registry

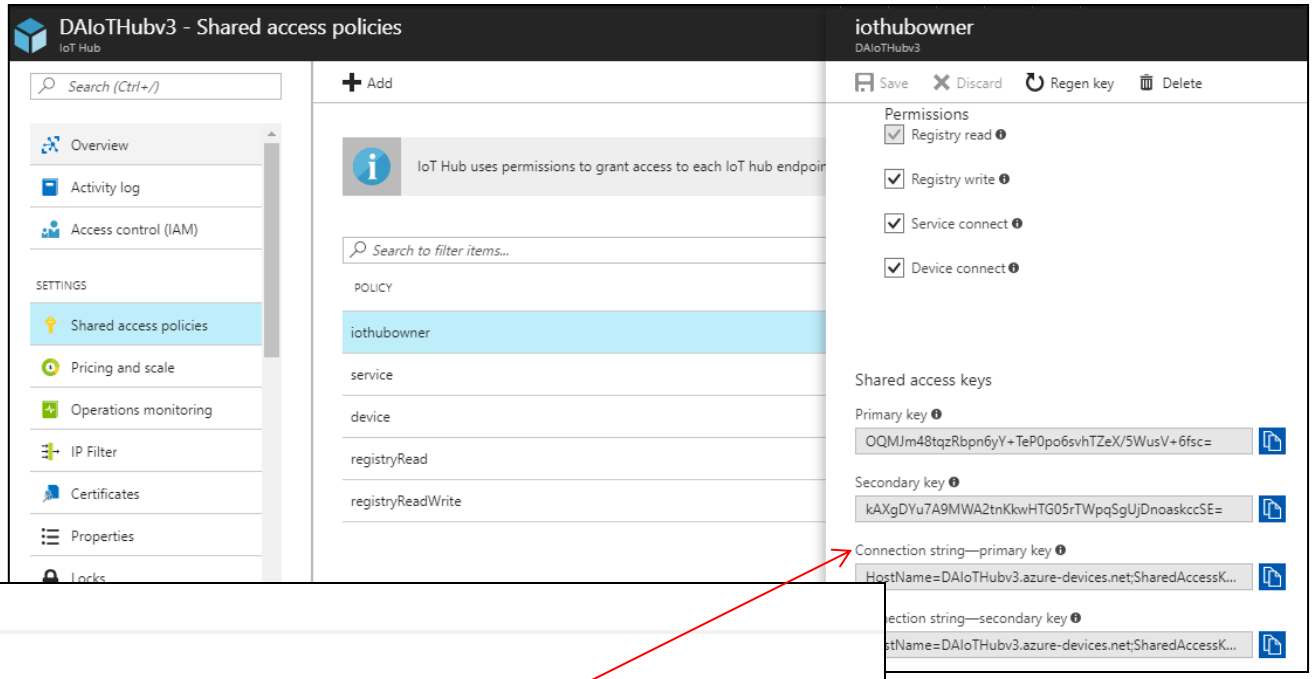
- For any device to connect to Azure's IoT Hub it must be recognized within Azure's Identity Registry.
- The Identity Registry stores information about the devices permitted to connect to that specific IoT Hub.
- New devices need to be added to the Identity Registry.
  - A unique Device ID & a key will be created for each device you register for authentication.
  - This identifies your device for sending data from 'device-to-cloud' messages to the IoT Hub.
- To register a Device you can use: Python, .NET, Portal, PowerShell, AZ CLI

Identity Registry operations:

Create device identity
Update device identity
Retrieve device identity by ID
Delete device identity
List up to 1000 identities
Export all identities to Azure blob storage
Import identities from Azure blob storage

Register a Device in AZ's Identity Registry

- This is rather simple to do in Python!
  - Add the IoT Hub Connection string from the Shared access policies
  - Add your new device name in the DEVICE_ID variable in Python code.
  - Run the code



File Edit Format View Help

```
import sys
import iothub_service_client
from iothub_service_client import IoTHubRegistryManager, IoTHubRegistryManagerAuthMethod
from iothub_service_client import IoTHubDeviceStatus, IoTHubError

# Connection to IOT Hub that was created in Portal or AZ CLI
CONNECTION_STRING = "HostName=DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=OQMJm48tqzRbpn6yY
+TeP0po6svhTZeX/5WusV+6fsc="
DEVICE_ID = "DianesFakeDevice"
```

Your new device name

Python: Imports and Connection String

CreateDeviceIdentity.py

- Add your Connection String & Define your Device Name

CreateDeviceIdentityv2 - Notepad

File Edit Format View Help

Add your Connection String here

```
import sys
import iothub_service_client
from iothub_service_client import IoTHubRegistryManager, IoTHubRegistryManagerAuthMethod
from iothub_service_client import IoTHubDeviceStatus, IoTHubError

# Connection to IOT Hub that was created in Portal or AZ CLI
CONNECTION_STRING = "HostName=DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=OQMJm48tqzRbpn6yY
+TeP0po6svhTZeX/5WusV+6fsc="
DEVICE_ID = "DianesFakeDevice"

# Print statements of metadata of the IOT Hub (note: you can compare this to the Portal IOT Hub metadata)
def print_device_info(title, iothub_device):
    print ( title + ":" )
    print ( "iothubDevice.deviceId                    = {0}".format(iothub_device.deviceId) )
    print ( "iothubDevice.primaryKey                  = {0}".format(iothub_device.primaryKey) )
    print ( "iothubDevice.secondaryKey                = {0}".format(iothub_device.secondaryKey) )
    print ( "iothubDevice.connectionState              = {0}".format(iothub_device.connectionState) )
    print ( "iothubDevice.status                       = {0}".format(iothub_device.status) )
    print ( "iothubDevice.lastActivityTime             = {0}".format(iothub_device.lastActivityTime) )
    print ( "iothubDevice.cloudToDeviceMessageCount    = {0}".format(iothub_device.cloudToDeviceMessageCount) )
    print ( "iothubDevice.isManaged                    = {0}".format(iothub_device.isManaged) )
    print ( "iothubDevice.authMethod                   = {0}".format(iothub_device.authMethod) )
    print ( "" )
```

Add your Device Name

Python: Print Statements

CreateDeviceIdentity.py

- Primarily prints metadata from the IoT Hub and Identity Registry after the device is created

```
# Print statements of metadata of the IOT Hub (note: you can compare this to the Portal IOT Hub
metadata)
def print_device_info(title, iothub_device):
    print ( title + ":" )
    print ( "iothubDevice.deviceId                    = {0}".format(iothub_device.deviceId) )
    print ( "iothubDevice.primaryKey                  = {0}".format(iothub_device.primaryKey) )
    print ( "iothubDevice.secondaryKey                = {0}".format(iothub_device.secondaryKey)
)
    print ( "iothubDevice.connectionState             = {0}".format
(iothub_device.connectionState) )
    print ( "iothubDevice.status                      = {0}".format(iothub_device.status) )
    print ( "iothubDevice.lastActivityTime             = {0}".format
(iothub_device.lastActivityTime) )
    print ( "iothubDevice.cloudToDeviceMessageCount    = {0}".format
(iothub_device.cloudToDeviceMessageCount) )
    print ( "iothubDevice.isManaged                   = {0}".format(iothub_device.isManaged) )
    print ( "iothubDevice.authMethod                  = {0}".format(iothub_device.authMethod) )
    print ( "" )
```

Python: Function to create & register a Device & Main

CreateDeviceIdentity.py

- Just one Function!
- Create & Register your Device in the Identity Registry

CreateDeviceIdentityv2 - Notepad

File Edit Format View Help

```
# Function to create the device using the Registry Manager.
def iothub_createdevice():
    try:
        iothub_registry_manager = IoTHubRegistryManager(CONNECTION_STRING)
        auth_method = IoTHubRegistryManagerAuthMethod.SHARED_PRIVATE_KEY
        new_device = iothub_registry_manager.create_device(DEVICE_ID, "", "", auth_method)
        print_device_info("CreateDevice", new_device)

    except IoTHubError as iothub_error:
        print ( "Unexpected error {0}".format(iothub_error) )
        return
    except KeyboardInterrupt:
        print ( "iothub_createdevice stopped" )

# Main function
if __name__ == '__main__':
    print ( "" )
    print ( "Python {0}".format(sys.version) )
    print ( "Creating device using the Azure IoT Hub Service SDK for Python" )
    print ( "" )
    print ( "    Connection string = {0}".format(CONNECTION_STRING) )
    print ( "    Device ID         = {0}".format(DEVICE_ID) )

    iothub_createdevice()
```

Run CreateDeviceIdentity.py

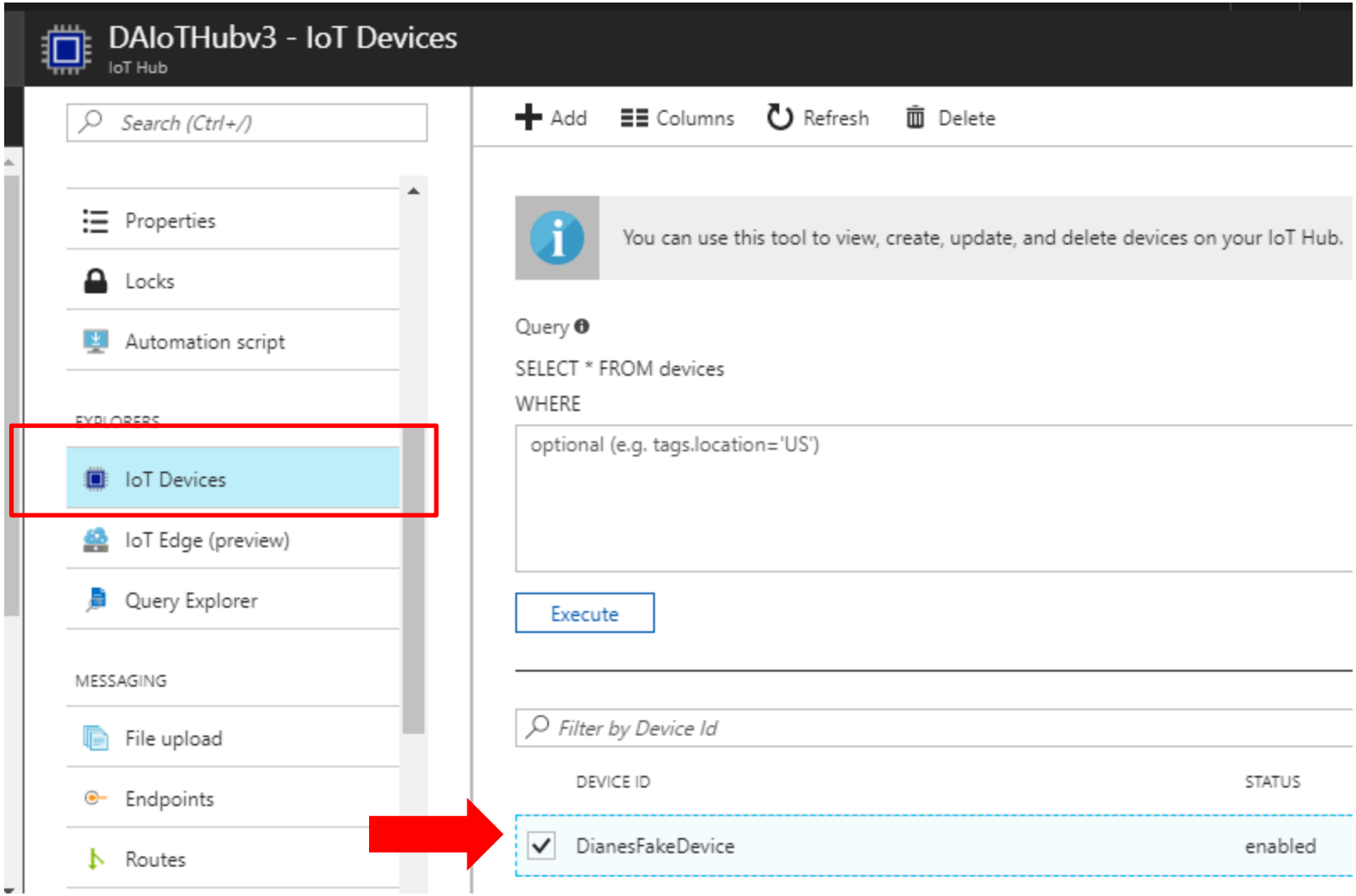
>python createdeviceidentity.py

```
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>python createdeviceidentityv3.py

Python 3.6.2 |Anaconda, Inc.| (default, Sep 19 2017, 08:03:39) [MSC v.1900 64 bit (AMD64
)]
Creating device using the Azure IoT Hub Service SDK for Python

    Connection string = HostName=DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothub
owner;SharedAccessKey=OQMJm48tqzRbpn6yY+TeP0po6svhTZeX/5WusV+6fsc=
    Device ID         = DianesFakeDevice3
CreateDevice:
iothubDevice.deviceId                    = DianesFakeDevice3
iothubDevice.primaryKey                  = i00pzhb4iRgqRAxLCPveRzxjVCO7cjLivpVrcYlFXlw=
iothubDevice.secondaryKey                = nK/ssQPQ9iPl5uFT+3WTybME7dVdhubjnIVGgK/KCW4=
iothubDevice.connectionState             = DISCONNECTED
iothubDevice.status                      = ENABLED
iothubDevice.lastActivityTime            = 0001-01-01T00:00:00
iothubDevice.cloudToDeviceMessageCount   = 0
iothubDevice.isManaged                   = False
iothubDevice.authMethod                  = SHARED_PRIVATE_KEY
```

Check AZ Portal to see if your Device is Registered



OR Check your Device using AZ CLI

Use:

>az iot device list --hub-name yourIoTHubname

```
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>az iot device list --hub-name DeepAzureIOTHub
[
  {
    "authentication": {
      "symmetricKey": {
        "primaryKey": "kkYF7ZgwXrZQ5LE8UAEaeWvtychyxuVU1+KYdtHt2Es=",
        "secondaryKey": "Z19PIdb17PljK5Av46Yk1m2WSfdiBh/qw3WOCbzeyR8="
      },
      "x509Thumbprint": {
        "primaryThumbprint": null,
        "secondaryThumbprint": null
      }
    },
    "cloudToDeviceMessageCount": 0,
    "connectionState": "Disconnected",
    "connectionStateUpdatedTime": "0001-01-01T00:00:00",
    "deviceId": "DianesPythonDevice",
    "etag": "ODY1Mzc5OTIz",
    "generationId": "636481275457080526",
    "lastActivityTime": "0001-01-01T00:00:00",
    "status": "enabled",
    "statusReason": null,
    "statusUpdatedTime": "0001-01-01T00:00:00"
  }
```

Steps

- ✓ Portal or AZ CLI or Power Shell: Create IoT Hub and obtain name, access policy for the Connection string
- ✓ Pip Install: IoT Hub Service Client SDK
- ✓ Python: Register a Device in the Identity Registry

4. Pip Install: azure-iothub-device-client SDK
5. Python: Create an app to simulate a device via an app to send messages to the IoT Hub
6. Install the IoT Hub Explorer
7. Run the IoT Hub Explorer tool to receive messages from the 'Simulated App'/Device

WALK THRU of Demo

Create IoT Hub in AZ CLI or Portal

Register Device in Python

Create 'simulated App' in Python – send telemetry Messages

Receive telemetry messages in IoT Hub Explorer Tool

Install Python SDK to Create a Simulated App

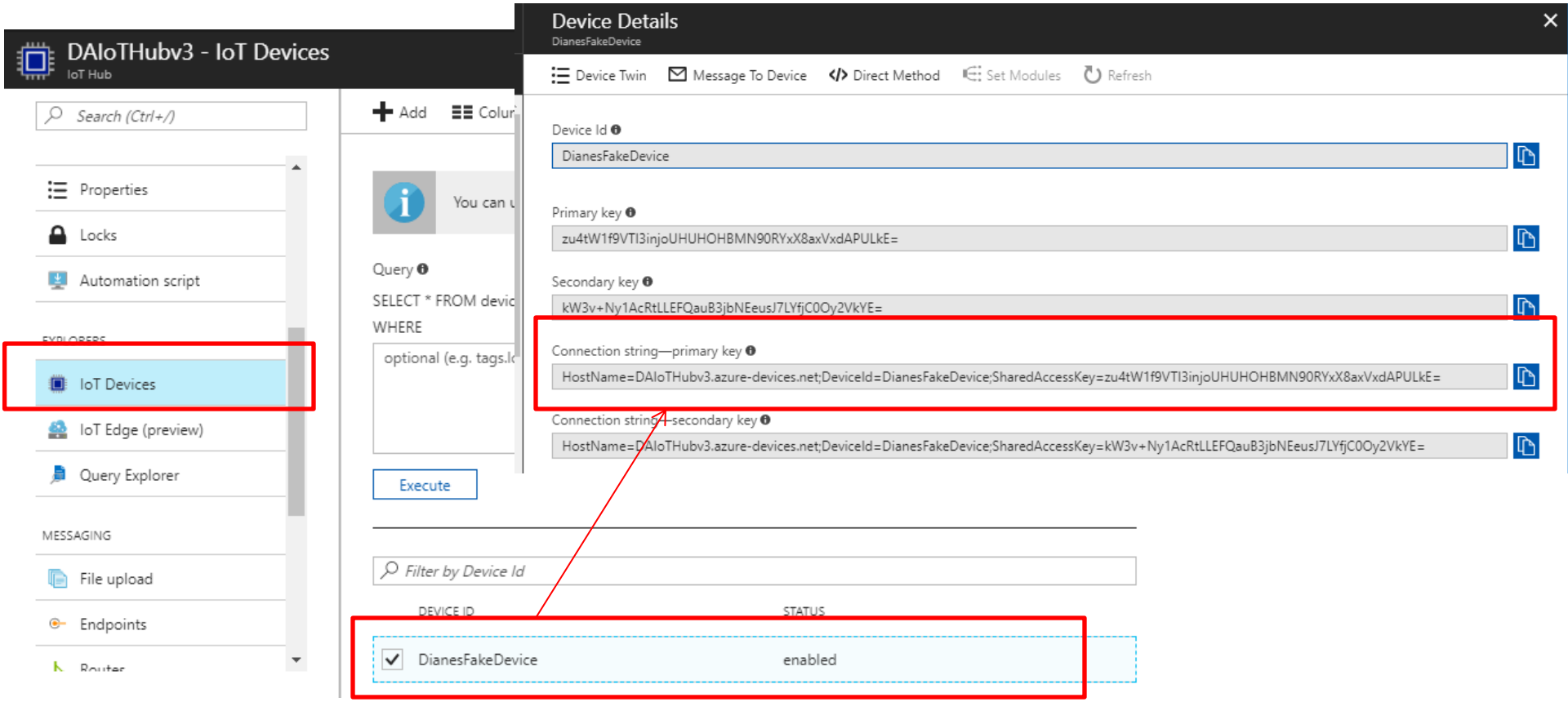
- Sends ‘device-to-cloud’ telemetry messages to the IoT Hub.
- First need to install Python SDK azure-iothub-device-client

>pip install azure-iothub-device-client

```
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>pip install azure-iothub-device-client
Collecting azure-iothub-device-client
  Downloading azure_iothub_device_client-1.1.28.0-py3-none-win_amd64.whl (422kB)
    100% |                                        | 430kB 859kB/s
Installing collected packages: azure-iothub-device-client
Successfully installed azure-iothub-device-client-1.1.28.0
```

Obtain your connection string of your Device

- Go to IoT Devices, select your Device Name and copy the Connection String



Python: Imports and Set up Connection Info

SimulatedDevice.py

- Add your Device ID Connection String.
- Add your Device ID
- Use the default MQTT transport protocol

SimulatedDevicev2 - Notepad

File Edit Format View Help

```
import random
import time
import sys
import iothub_client
from iothub_client import IoTHubClient, IoTHubClientError, IoTHubTransportProvider,
IoTHubClientResult
from iothub_client import IoTHubMessage, IoTHubMessageDispositionResult, IoTHubError,
DeviceMethodReturnValue

# String containing Hostname, Device Id & Device Key
CONNECTION_STRING = "HostName=DAIoTHubv3.azure-
devices.net;DeviceId=DianesFakeDevice;SharedAccessKey=zu4tW1f9VTI3injoUHUHOHBMN90RYxX8axVxdAPULk
E="

# choose HTTP, AMQP or MQTT as transport protocol
PROTOCOL = IoTHubTransportProvider.MQTT
MESSAGE_TIMEOUT = 10000
AVG_WIND_SPEED = 10.0
SEND_CALLBACKS = 0
MSG_TXT = "{\"deviceId\": \"DianesFakeDevice\",\"windSpeed\": %.2f}"
```

Python: Functions

SimulatedDevice.py
3 Functions in total.
2 Functions below:

- Send confirmation msgs & Initialize IoT Hub Client, set timing for telemetry messages.

SimulatedDevicev2 - Notepad

File Edit Format View Help

```
# Send the confirmation callback
def send_confirmation_callback(message, result, user_context):
    global SEND_CALLBACKS
    print ( "Confirmation[%d] received for message with result = %s" % (user_context, result) )
    map_properties = message.properties()
    print ( "    message_id: %s" % message.message_id )
    print ( "    correlation_id: %s" % message.correlation_id )
    key_value_pair = map_properties.get_internals()
    print ( "    Properties: %s" % key_value_pair )
    SEND_CALLBACKS += 1
    print ( "    Total calls confirmed: %d" % SEND_CALLBACKS )

# Initialize the client
def iothub_client_init():
    # prepare iothub client
    client = IoTHubClient(CONNECTION_STRING, PROTOCOL)
    # set the time until a message times out
    client.set_option("messageTimeout", MESSAGE_TIMEOUT)
    client.set_option("logtrace", 0)
    client.set_option("product_info", "HappyPath_Simulated-Python")
    return client
```

Python: One more Function

SimulatedDevice.py

- Send message from simulated AP to the IoT Hub

SimulatedDevicev2 - Notepad

File Edit Format View Help

```
# format and send a message from your simulated device to your IoT hub.
def iothub_client_telemetry_sample_run():

    try:
        client = iothub_client_init()
        print ( "IoT Hub device sending periodic messages, press Ctrl-C to exit" )
        message_counter = 0

        while True:
            msg_txt_formatted = MSG_TXT % (AVG_WIND_SPEED + (random.random() * 4 + 2))
            # messages can be encoded as string or bytearray
            if (message_counter & 1) == 1:
                message = IoTHubMessage(bytearray(msg_txt_formatted, 'utf8'))
            else:
                message = IoTHubMessage(msg_txt_formatted)
            # optional: assign ids
            message.message_id = "message_%d" % message_counter
            message.correlation_id = "correlation_%d" % message_counter
            # optional: assign properties
            prop_map = message.properties()
            prop_text = "PropMsg_%d" % message_counter
            prop_map.add("Property", prop_text)

            client.send_event_async(message, send_confirmation_callback, message_counter)
            print ( "IoTHubClient.send_event_async accepted message [%d] for transmission to IoT
Hub." % message_counter )

            status = client.get_send_status()
            print ( "Send status: %s" % status )
            time.sleep(30)

            status = client.get_send_status()
            print ( "Send status: %s" % status )
```

Python: Main

SimulatedDevice.py

- Main Function to connect to the IoT Hub and send telemetry messages via MQTT.

```
# main function
if __name__ == '__main__':
    print ( "Simulating a device using the Azure IoT Hub Device SDK for Python" )
    print ( "    Protocol %s" % PROTOCOL )
    print ( "    Connection string=%s" % CONNECTION_STRING )

    iothub_client_telemetry_sample_run()
```

Don’t run SimulatedDevice.py just yet!

Steps

- ✓ Portal or AZ CLI or Power Shell: Create IoT Hub and obtain name, access policy for the Connection string
- ✓ Pip Install: IoT Hub Service Client SDK
- ✓ Python: Register a Device in the Identity Registry
- ✓ Pip Install: azure-iothub-device-client SDK
- ✓ Python: Create an app to simulate a device via an app to send messages to the IoT Hub

6. Install the IoT Hub Explorer
7. Run the IoT Hub Explorer tool to receive messages from the 'Simulated App'/Device

WALK THRU of Demo

Create IoT Hub in AZ CLI or Portal

Register Device in Python

Create 'simulated App' in Python – send telemetry Messages

Receive telemetry messages in IoT Hub Explorer Tool

IoT Hub Explorer tool

- IoT Hub Explorer is a tool that allows you to explore and test Azure IoT Hub features.
- You use an Event Hubs-compatible endpoint exposed by the IoT Hub, which reads the device-to-cloud messages **BUT Event Hubs does not support reading telemetry in Python yet.**
- Other options: create a Node.js or a .NET Event Hubs-based console app to read the device-to-cloud messages from IoT Hub.
- Or use the IoT Hub Explorer tool to read the device telemetry messages.
- Use npm to install the IoT Hub Explorer tool.

Check if npm is already installed:

>npm -v

'npm' is not recognized as an internal or external command,

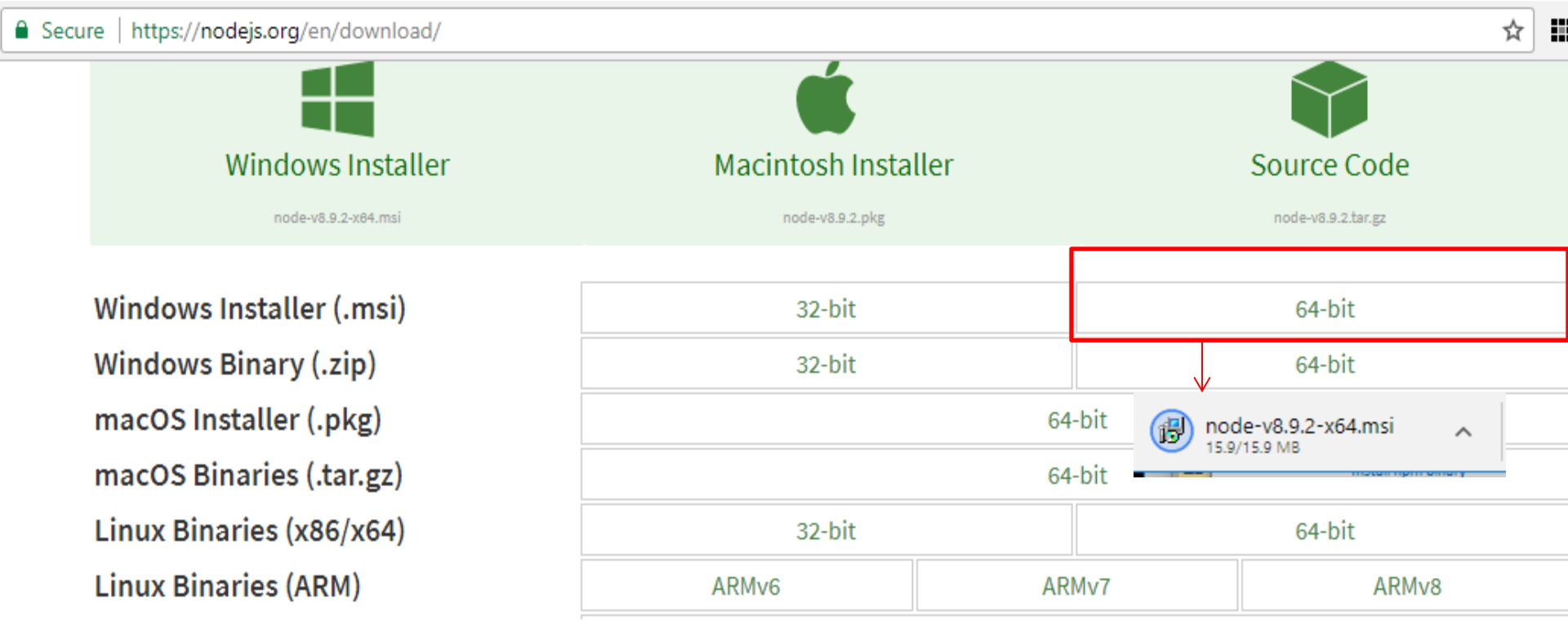
operable program or batch file.

Most likely npm is not installed in your Windows 10 or Windows 7 version.

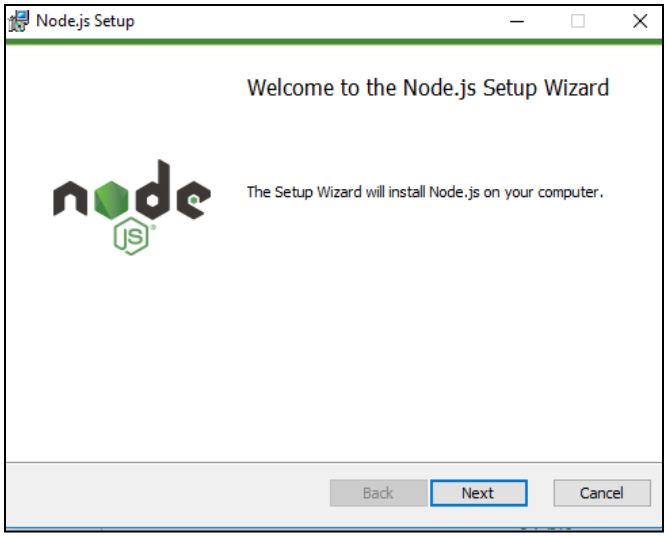
Install npm binary

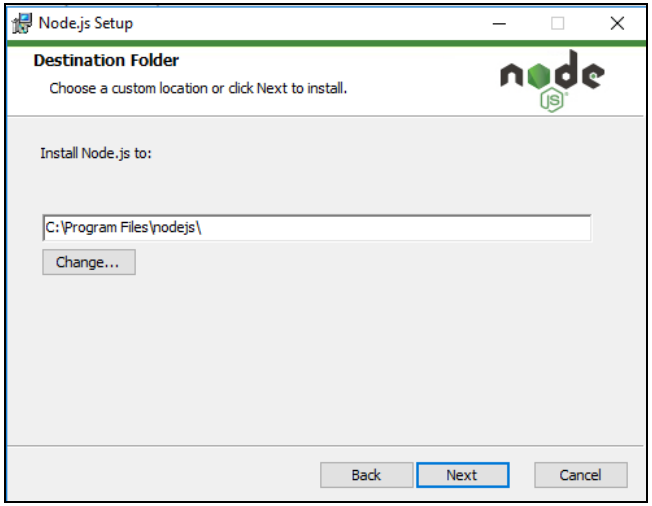
- Latest npm is built into node.js
- Download node.js

https://nodejs.org/en/download/ for Windows, MAC or Linux

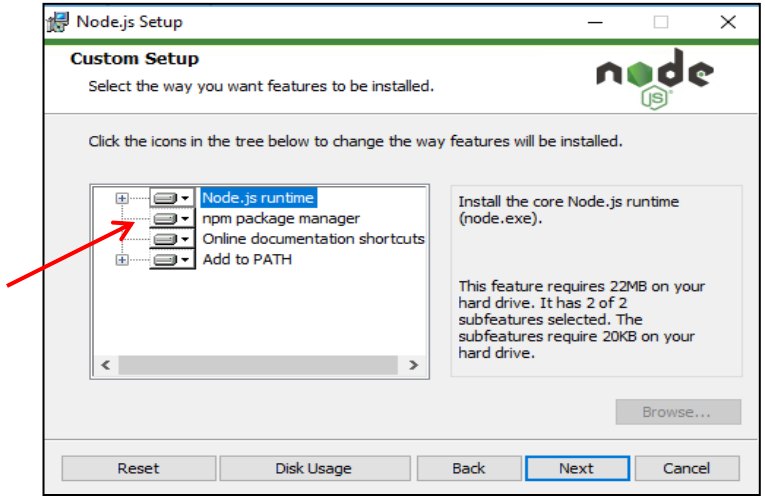


Install node.js

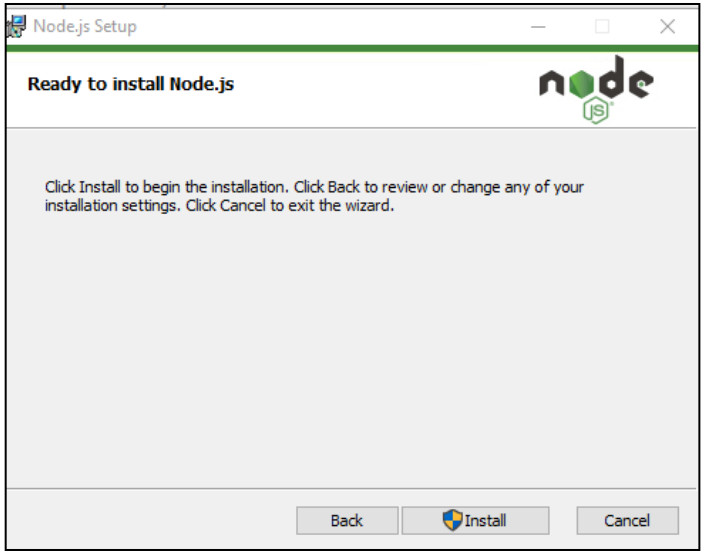


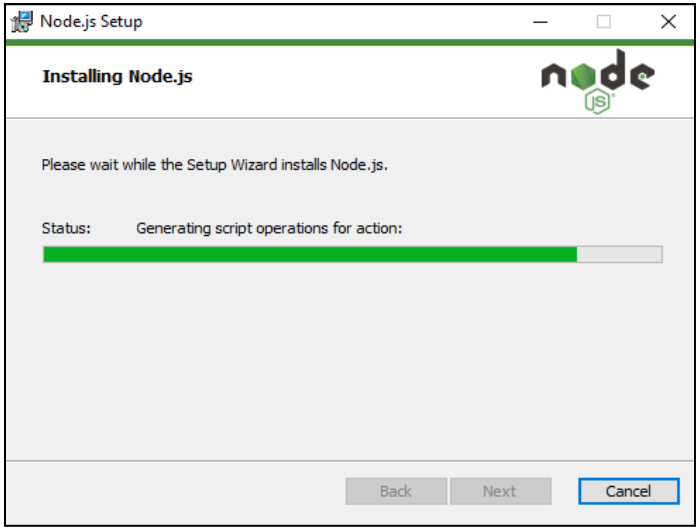


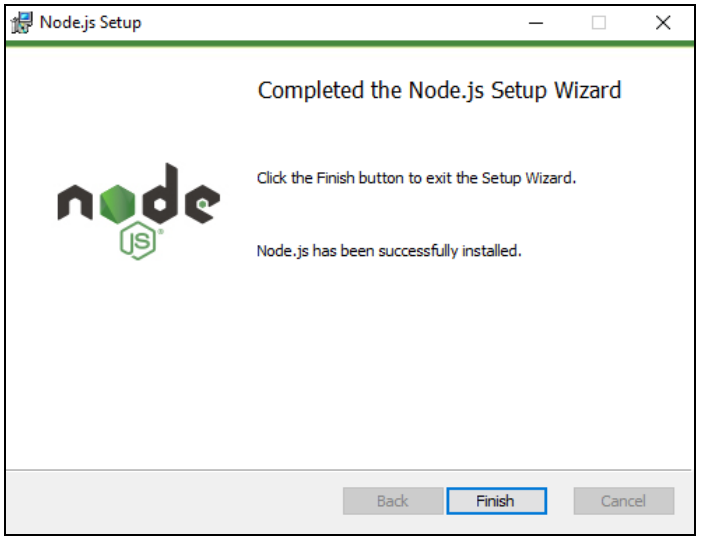
Notice that npm will be installed



Install node.js - continued







This installation set up env variables for npm. Now check if npm is installed:

C:\Users\dhoward>npm -v
5.5.1

Install the IoT Hub Explorer tool

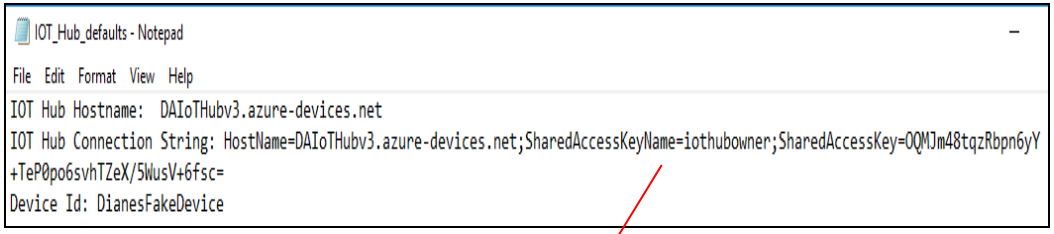
>npm install -g iothub-explorer

```
C:\Users\dhoward>npm install -g iothub-explorer
npm WARN deprecated crypto@0.0.3: This package is no longer supported. It's now a built-in Node module. If you've depende
d on crypto, you should switch to the one that's built-in.
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
C:\Users\dhoward\AppData\Roaming\npm\iothub-explorer -> C:\Users\dhoward\AppData\Roaming\npm\node_modules\iothub-explorer
\iothub-explorer.js
+ iothub-explorer@1.1.19
added 209 packages in 57.445s
```

Start the IoT Hub Explorer

- This is our simulator to receive messages from our Device thru our AZ IoT Hub that we created.
- To start the IoT Hub Explorer enter:

> iothub-explorer monitor-events yourdevicename --login "[IoTHub connection string]"



My example:

>>iothub-explorer monitor-events DianesFakeDevice --login "HostName=DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=OQMJm48tqzRbpn6yY+TeP0po6svhTZeX/5WusV+6fsc="

```
C:\Users\dhoward>iothub-explorer monitor-events DianesPythonDevice --login "H
ostName=DeepAzureIOTHub.azure-devices.net;SharedAccessKeyName=iothubowner;Sha
redAccessKey=du/8Hb08oASbnKuYhPlEsterCglztLVCAX6/w0VWlvQ="

Monitoring events from device DianesPythonDevice...
```

Run the App Simulator

>python simulateddevice.py

```
C:\Windows\System32\cmd.exe - python simulateddevicev2.py
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>python simulateddevicev2.py
Simulating a device using the Azure IoT Hub Device SDK for Python
    Protocol MQTT
    Connection string=HostName=DAIoTHubv3.azure-devices.net;DeviceId=DianesFakeDevice;SharedAccessKey=z
u4tW1f9VTI3injoUHUHOHBMN90RYxX8axVxdAPULkE=
Info: IoT Hub SDK for C, version 1.1.28
IoT Hub device sending periodic messages, press Ctrl-C to exit
IoTHubClient.send_event_async accepted message [0] for transmission to IoT Hub.
Send status: BUSY
Confirmation[0] received for message with result = OK
    message_id: message_0
    correlation_id: correlation_0
    Properties: {'Property': 'PropMsg_0'}
    Total calls confirmed: 1
Send status: IDLE
IoTHubClient.send_event_async accepted message [1] for transmission to IoT Hub.
Send status: BUSY
Confirmation[1] received for message with result = OK
    message_id: message_1
    correlation_id: correlation_1
    Properties: {'Property': 'PropMsg_1'}
    Total calls confirmed: 2
Send status: IDLE
IoTHubClient.send_event_async accepted message [2] for transmission to IoT Hub.
Send status: BUSY
Confirmation[2] received for message with result = OK
```

Output

C:\Windows\System32\cmd.exe - python simulateddevicev2.py

```
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>python simulateddevicev2.py
Simulating a device using the Azure IoT Hub Device SDK for Python
    Protocol MQTT
    Connection string=HostName=DAIoTHubv3.azure-devices.net;DeviceId=DianesFakeDevice;SharedAccessKey=z
u4tW1f9VTI3injoUHUHOHBMN90RYxX8axVxdAPULkE=
Info: IoT Hub SDK for C, version 1.1.28
IoT Hub device sending periodic messages, press Ctrl-C to exit
IoTHubClient.send_event_async accepted message [0] for transmission to IoT Hub.
Send status: BUSY
Confirmation[0] received for message with result = OK
    message_id: message_0
    correlation_id: correlation_0
    Properties: {'Property': 'PropMsg_0'}
    Total calls confirmed: 1
Send status: IDLE
IoTHubClient.send_event_async accepted message [1]
Send status: BUSY
Confirmation[1] received for message with result =
    message_id: message_1
    correlation_id: correlation_1
    Properties: {'Property': 'PropMsg_1'}
    Total calls confirmed: 2
Send status: IDLE
IoTHubClient.send_event_async accepted message [2]
Send status: BUSY
Confirmation[2] received for message with result =
```

IoT Explorer tool: receipt of messages

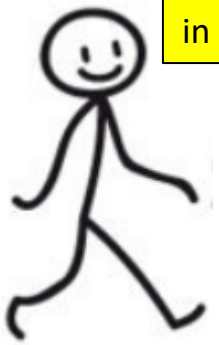
```
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>iothub-explorer monitor-events DianesFakeDevice --login "HostName=
DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=OQMJm48tqzRbpn6yY+TeP0po6svhTZeX/5WusV+6fsc
="

Monitoring events from device DianesFakeDevice...
==== From: DianesFakeDevice ====
{
  "deviceId": "DianesFakeDevice",
  "windSpeed": 13.13
}
---- application properties ----
{
  "Property": "PropMsg_0"
}
====================
==== From: DianesFakeDevice ====
{
  "deviceId": "DianesFakeDevice",
  "windSpeed": 12.64
}
---- application properties ----
{
  "Property": "PropMsg_1"
}
====================
==== From: DianesFakeDevice ====
{
  "deviceId": "DianesFakeDevice",
  "windSpeed": 12.47
}
---- application properties ----
{
  "Property": "PropMsg_2"
```

Steps

- ✓ Portal or AZ CLI or Power Shell: Create IoT Hub and obtain name, access policy for the Connection string
- ✓ Pip Install: IoT Hub Service Client SDK
- ✓ Python: Register a Device in the Identity Registry
- ✓ Pip Install: azure-iothub-device-client SDK
- ✓ Python: Create an app to simulate a device via an app to send messages to the IoT Hub
- ✓ Install the IoT Hub Explorer
- ✓ Run the IoT Hub Explorer tool to receive messages from the 'Simulated App'/Device

WALK THRU of Demo



Create IoT Hub in AZ CLI or Portal

Register Device in Python

Create 'simulated App' in Python – send telemetry Messages

Receive telemetry messages in IoT Hub Explorer Tool

Summary

Demonstrated:

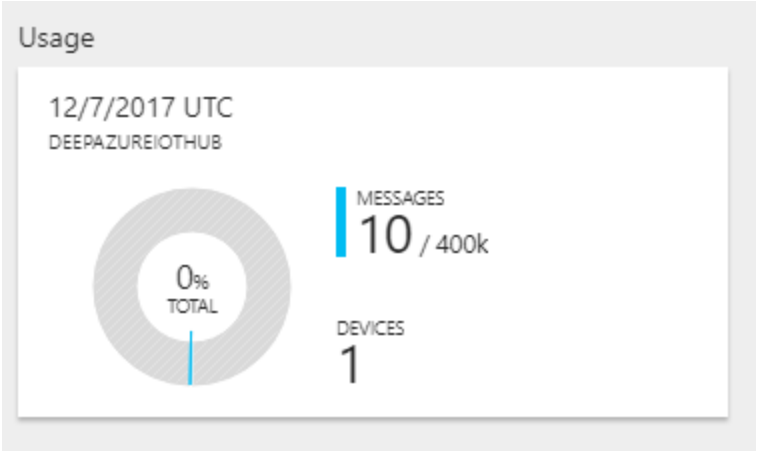
- how to set up an IoT Hub
- register one device to the IoT Hub's Identity Registry
- create a simulated device app in Azure's portal to connect to the IoT Hub and simulates sending telemetry messages periodically from the device via the MQTT protocol to the IoT Hub and
- Display received telemetry messages in a tool called IoT Hub Explorer.

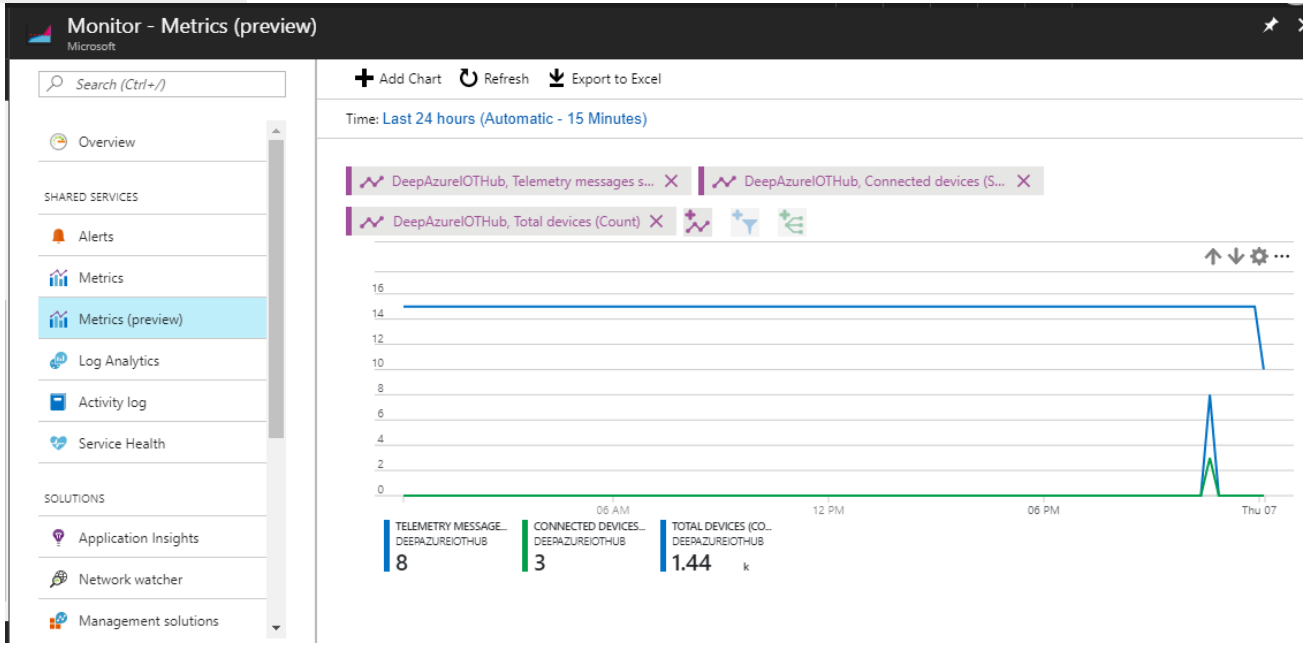
Excellent skeleton to use for setting up an IoT Hub, devices and testing messages via the MQTT protocol.

Caveats:

1. Python IoT Hub Service Client SDK had missing module error with Python 2.7
2. Python SDK IoT Hub Service Client & azure-iothub-device-client only available in Windows. Unsure if it works with Windows 7. There was a workaround page provided for Mac and Linux.
3. AZ's Event Hubs does not support reading telemetry in Python yet. Stay tuned!

Metrics of Successful Delivery/Receipt of Messages





Error: When creating a device in the Identity Registry

```
C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>python CreateDeviceIdentity.py

Python 3.6.2 |Anaconda, Inc.| (default, Sep 19 2017, 08:03:39) [MSC v.1900 64 bit (AMD64)]
Creating device using the Azure IoT Hub Service SDK for Python

    Connection string = [DeepAzureIOTHub.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=du/8Hb08oAS
uYhPlEsterCglztLVCAX6/w0VWlvQ=]
    Device ID         = DianesPythonDevice
Error: Time:Tue Dec  5 20:24:44 2017 File:C:\Users\Administrator\Documents\WindowsPowerShell\Modules\aziotsdk_pytools\
\c\iothub_service_client\src\iothub_service_client_auth.c Func:IoTHubServiceClientAuth_CreateFromConnectionString Line
 Couldn't find HostName in connection string
Unexpected error IoTHubRegistryManager.IoTHubRegistryManager, IoTHubRegistryManagerResult.ERROR
```

Due to invalid Hostname.

Fix: You don't need the []. Also check that you copied the entire line from the Shared Access Policies Connection String Primary key (not the secondary key).

Error: When running SimulatedDevice.py

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\dhoward>cd C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09

C:\Users\dhoward\Documents\personal\Deep Azure\Lab 09>python SimulatedDevice.py
  File "SimulatedDevice.py", line 9
    CONNECTION_STRING = "HostName=DeepAzureIOTHub.azure-devices.net;DeviceID="DianesPythonDevice";SharedA
ccessKey=kkYF7ZgwXrZQ5LE8UAEaeWvtychyxuVU1+KYdtHt2Es="
                                                                                                        ^
SyntaxError: invalid syntax
```

Due to: Invalid Connection string.
Fix: Go to your Hub then in Explorers panel select IOT Devices then Device Details.
Copy the Connection String.
Replace the string in the code for the Connection String. Do not use [].

My Notepad defaults

IOT_Hub_defaults - Notepad

File Edit Format View Help

```
IOT Hub Hostname:  DAIoTHubv3.azure-devices.net
IOT Hub Connection String: HostName=DAIoTHubv3.azure-devices.net;SharedAccessKeyName=iothubowner;SharedAccessKey=OQMJm48tqzRbpn6yY
+TeP0po6svhTZeX/5WusV+6fsc=
Device Id: DianesFakeDevice
```