

<https://www.halvorsen.blog>



# Python MQTT, SQL Server and Microsoft Azure

Hans-Petter Halvorsen

# Contents

- Introduction
- MQTT
- SQL Server
- Python and SQL Server
- Microsoft Azure
- Databases in Microsoft Azure
- Code Examples

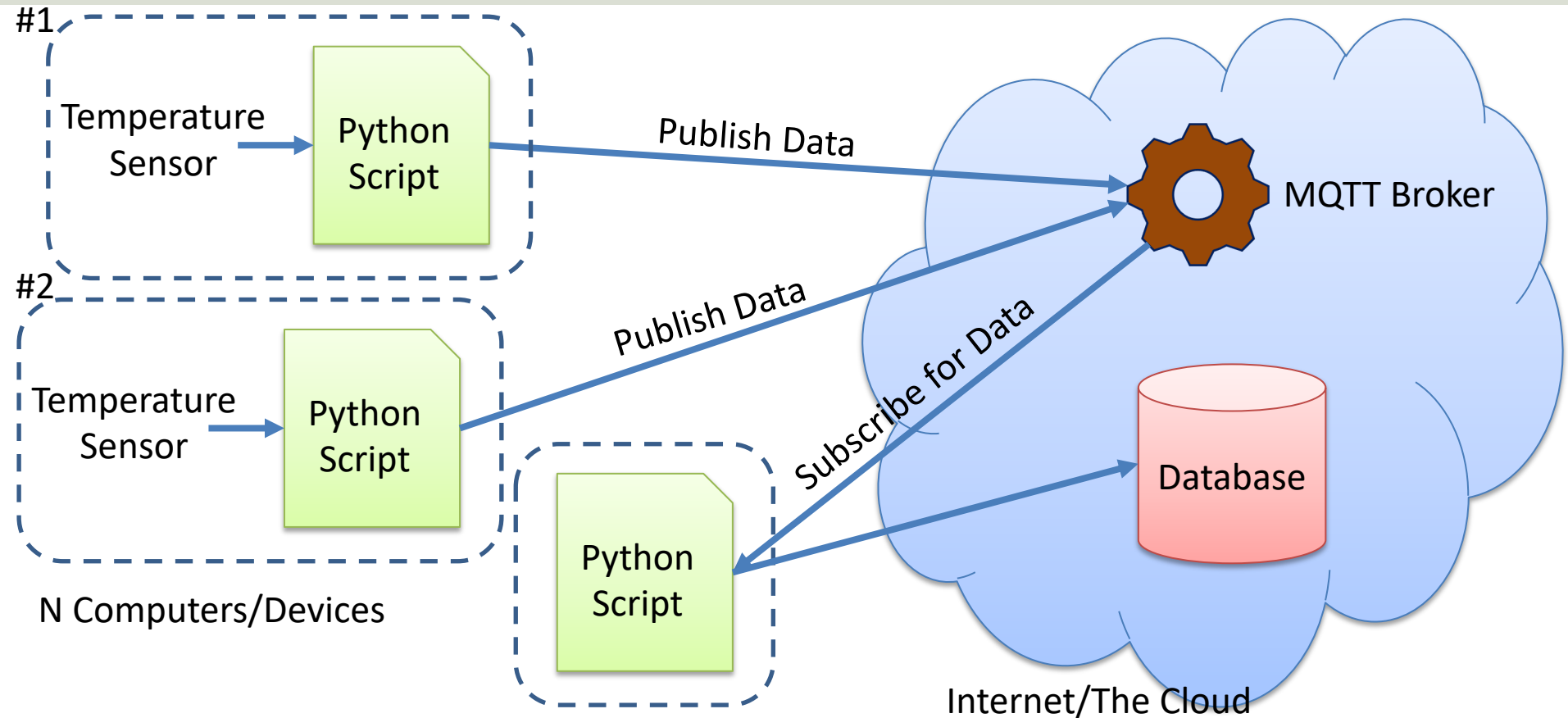


# Introduction

# Introduction

- We have N Computers/Devices that read data from different Sensors, e.g., Temperature Sensors, etc.
- The Data from all the Sensors should be stored in a Database located on the Internet/The Cloud
- Problem:
  - These Computers/Devices have no access to the Database
  - In order to get access to the Database, we need to open the Firewall for each of those Computers/Devices, and that is of course not recommended due to security issues, and it can be hundreds or thousands of computers
- Solution:
  - We use MQTT as the middle tier. MQTT is “Internet-friendly” because it uses standard HTTP
  - The Computers/Devices send Data to a MQTT Broker
  - Then a dedicated Computer (that has access to the Database) Subscribe on the Data and Forward the Data to the Database

# System Overview



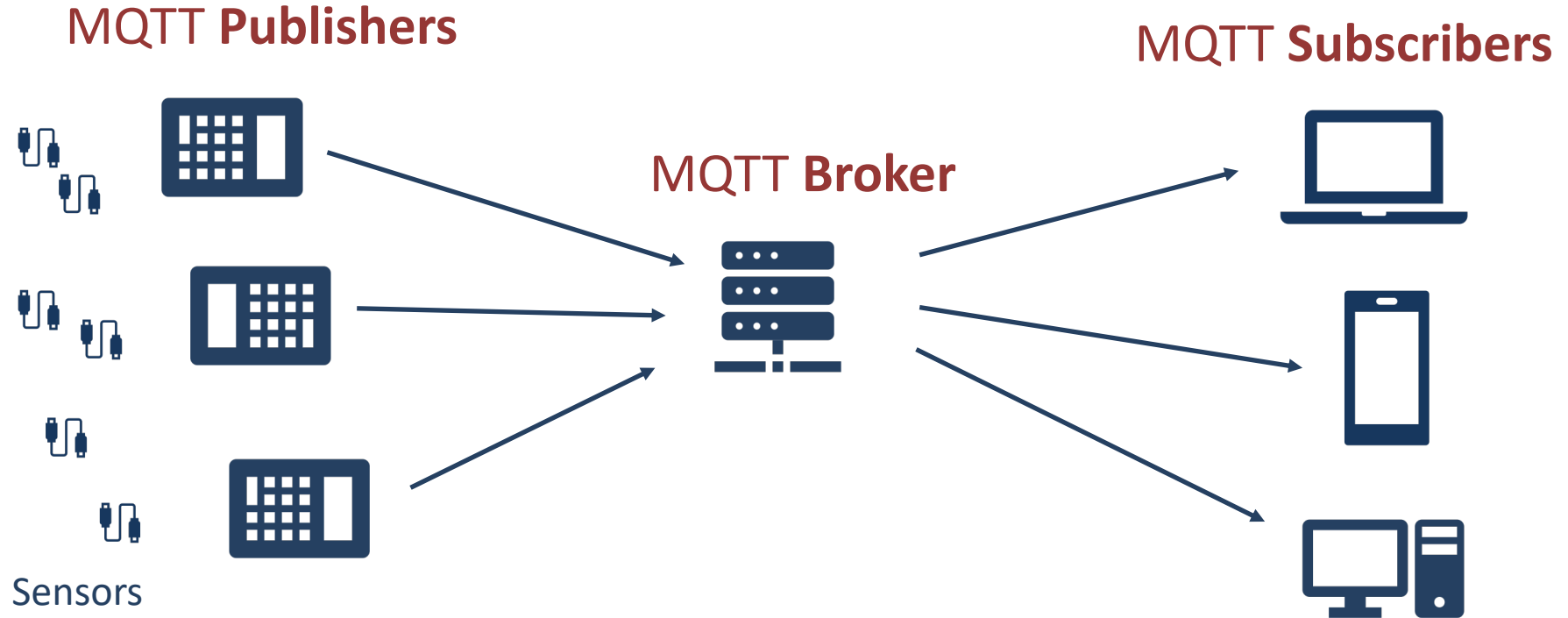


# MQTT

# MQTT

- MQTT is a Communication Protocol popular in Internet of Things (IoT) Applications
- <https://mqtt.org>
- You can use or implement MQTT in all the most popular Programming environments
- MQTT can be used on all the popular platforms like Windows, macOS, Linux, Arduino, Raspberry Pi
- You can use an existing API, or you can implement and use the MQTT protocol from scratch
- We will Python in this Tutorial

# MQTT Scenario





# MQTT Topics

- Data in MQTT are Published to Topics
- Topics are made up of one or more topic levels, separated by a forward slash

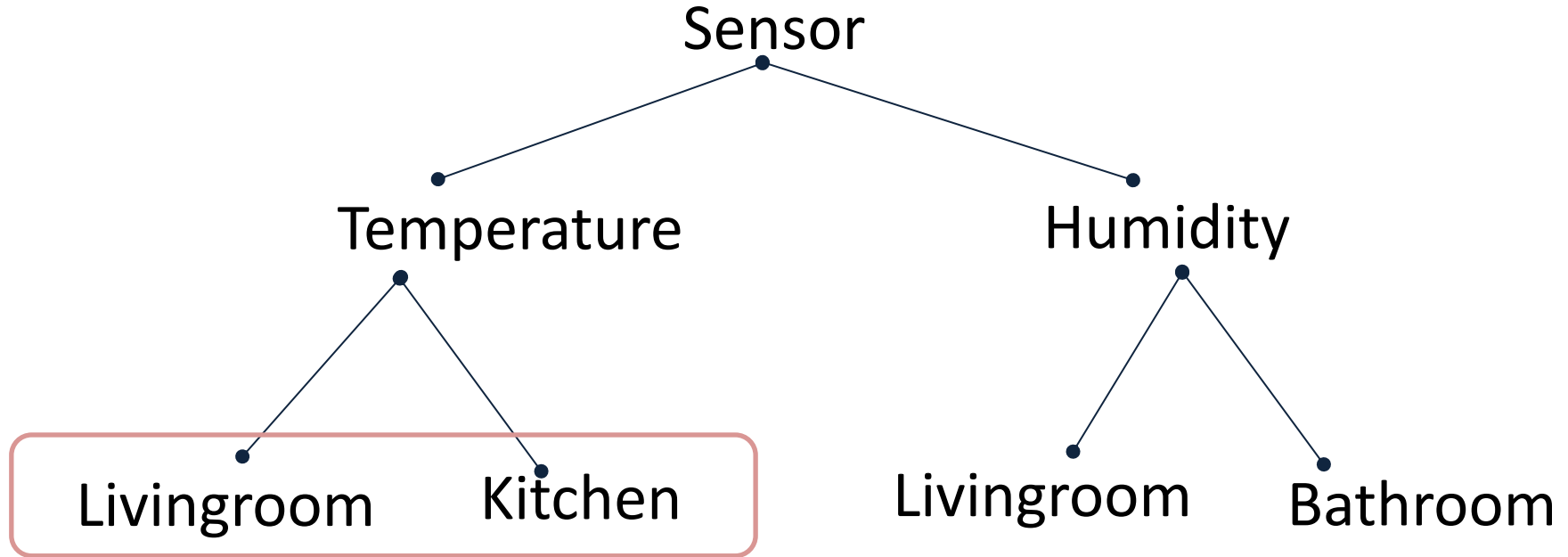
Example:

*Sensor/Temperature/Kitchen*

- Topics are used to organize the data
- Topics are case sensitive
- Topics don't have to be pre-registered at the broker

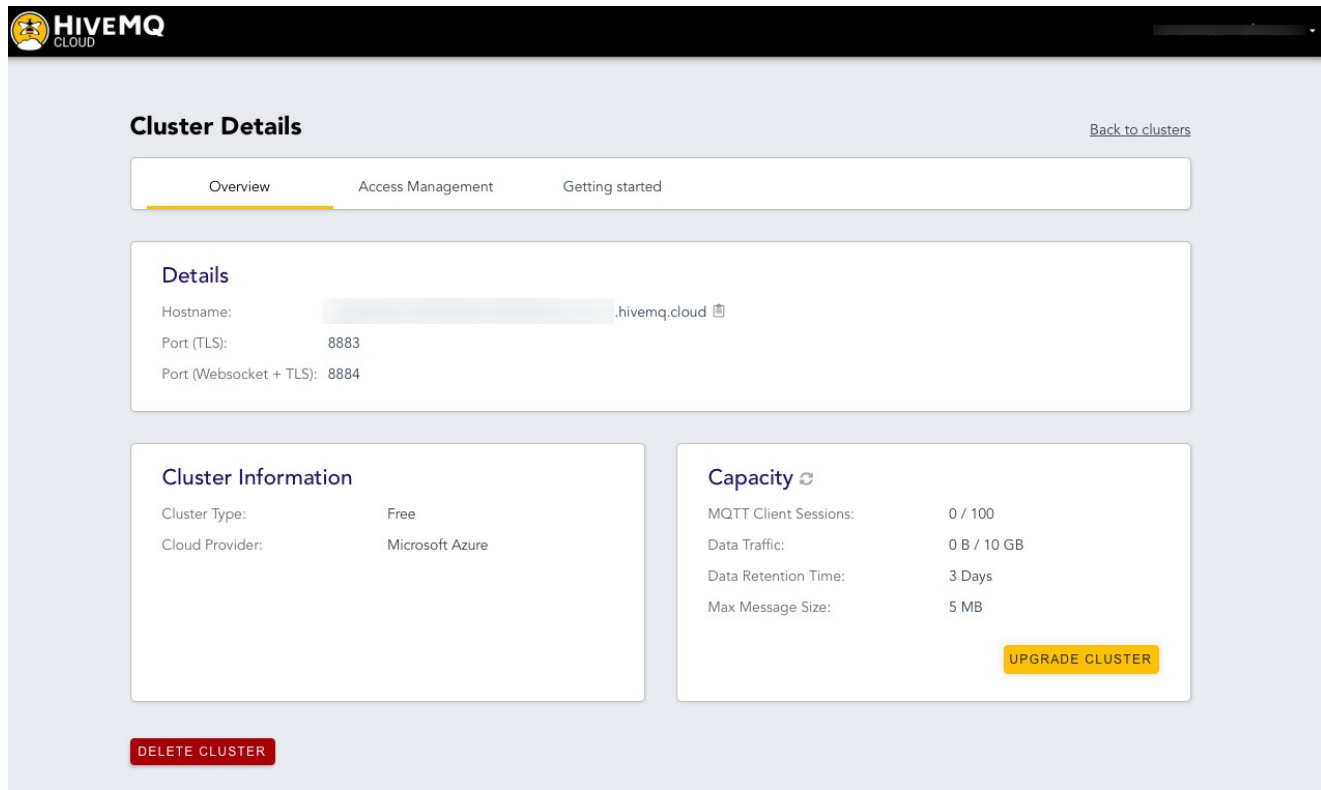
# Subscribe on Topics - Wildcards

Wildcards: *Sensor/Temperature/#*



# HiveMQ Cloud

<https://www.hivemq.com>



The screenshot displays the HiveMQ Cloud interface. At the top left is the HiveMQ Cloud logo. The main heading is "Cluster Details" with a "Back to clusters" link on the right. Below this is a navigation bar with three tabs: "Overview" (selected), "Access Management", and "Getting started".

The "Details" section contains the following information:

- Hostname: [redacted].hivemq.cloud
- Port (TLS): 8883
- Port (Websocket + TLS): 8884

The "Cluster Information" section shows:

- Cluster Type: Free
- Cloud Provider: Microsoft Azure

The "Capacity" section shows:

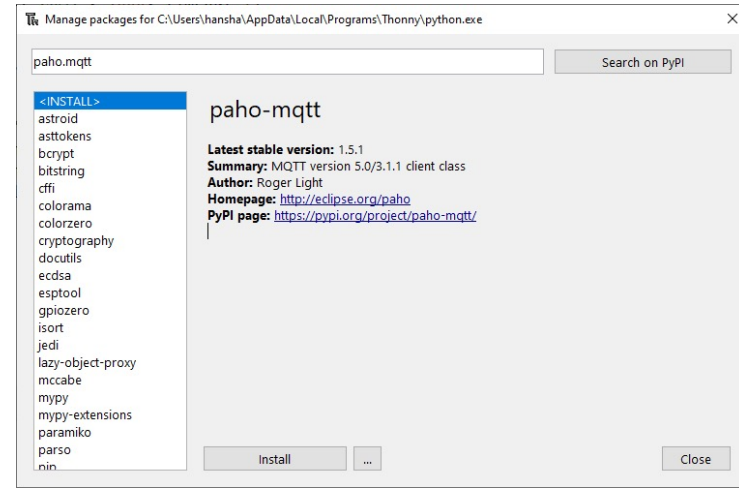
- MQTT Client Sessions: 0 / 100
- Data Traffic: 0 B / 10 GB
- Data Retention Time: 3 Days
- Max Message Size: 5 MB

At the bottom of the capacity section is a yellow "UPGRADE CLUSTER" button. At the bottom left of the dashboard is a red "DELETE CLUSTER" button.

# Using MQTT in Python

- The most used MQTT Python Library is paho-mqtt
- We need to install the paho-mqtt Python Library using pip

We need to install the paho-mqtt Python Library. You can use pip, or as here, the Thonny Python Editor has an easy way to install Python Libraries from a GUI





# SQL Server

# Database Systems

- Oracle
- MySQL
- MariaDB
- Sybase
- Microsoft Access
- Microsoft SQL Server
- ... (we have hundreds different Database Systems)

# SQL Server

- SQL Server consists of a Database Engine and a Management Studio.
- The Database Engine has no graphical interface - it is just a service running in the background of your computer (preferable on the server).
- The Management Studio is graphical tool for configuring and viewing the information in the database. It can be installed on the server or on the client (or both).

# SQL Server

- SQL Server Express
  - Free version of SQL Server that has all we need for the exercises in this Tutorial
- SQL Server Express consist of 2 parts (separate installation packages):
  - SQL Server Express
  - SQL Server Management Studio (SSMS) – This software can be used to create Databases, create Tables, Insert/Retrieve or Modify Data, etc.
- SQL Server Express Installation:  
<https://youtu.be/hhhggAlUYo8>



# SQL Server Management Studio

1 Your SQL Server

2 Your Database

3 Your Tables

4 Write your Query here

5 The result from your Query

Microsoft SQL Server Management Studio

File Edit View Query Debug Tools Window Community Help

Object Explorer

SQLQuery1.sql - P...SCHOOL (sa (52))\*

```
select * from SCHOOL
```

	SchoolId	SchoolName	Description	Address	Phone	PostCode	PostAddress
1	1	TUC	The best school	Telemark	NULL	NULL	NULL
2	2	MIT	OK School	USA	NULL	NULL	NULL
3	3	NTNU	The second best school	Trondheim	NULL	NULL	NULL
4	4	University of Oslo	The third best school	Oslo	NULL	NULL	NULL

Results Messages

Query executed successfully. | PC88235\DEVELOPMENT (10.50 ... | sa (52) | SCHOOL | 00:00:00 | 4 rows

Ready | Ln 1 | Col 21 | Ch 21 | INS

Properties

Current connection parameters

Aggregate Status

Connection fi

Elapsed time 00:00:00.0270016

Finish time 20.03.2012 08:28:15

Name PC88235\DEVELOPMENT

Rows returned 4

Start time 20.03.2012 08:28:15

State Open

Connection

Connection n PC88235\DEVELOPMENT

Connection Details

Connection e 00:00:00.0270016

Connection fi 20.03.2012 08:28:15

Connection n 4

Connection s 20.03.2012 08:28:15

Connection s Open

Display name PC88235\DEVELOPMENT

Login name sa

Server name PC88235\DEVELOPMENT

Server version 10.50.1600

Session Tracir

SPID 52

Name

The name of the connection.



# Python and SQL Server

Hans-Petter Halvorsen

[Table of Contents](#)

# Python

- Python is a fairly old Programming Language (1991) compared to many other Programming Languages like C# (2000), Swift (2014), Java (1995), PHP (1995).
- Python has during the last 10 years become more and more popular.
- Today, Python has become one of the most popular Programming Languages.

Software used in this Tutorial:

- Anaconda Distribution (Python + most used Libraries/Packages are included)
- Spyder Python editor (included with Anaconda Distribution)

# Python Drivers for SQL Server

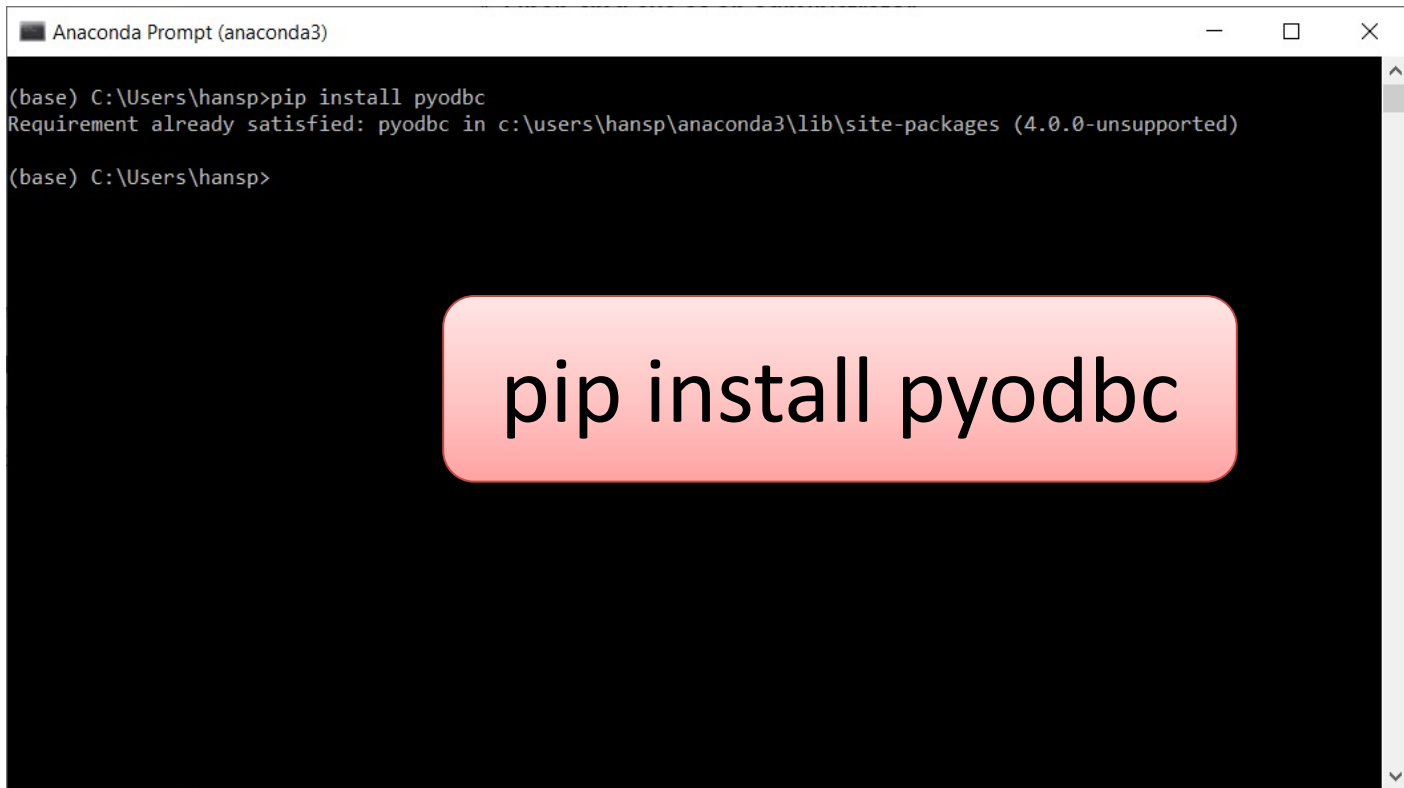
- There are several python SQL drivers available:
  - pyodbc
  - pymssql
- These Drivers are not made made Microsoft but the Python Community.
- However, Microsoft places its testing efforts and its confidence in **pyodbc** driver.
- Microsoft contributes to the pyODBC open-source community and is an active participant in the repository at GitHub

<https://docs.microsoft.com/sql/connect/python/python-driver-for-sql-server>

# pyodbc

- pyodbc is an open-source Python module that can access ODBC databases, e.g., SQL Server
- <https://pypi.org/project/pyodbc/>
- Installation: `pip install pyodbc`

# pyodbc




```
Anaconda Prompt (anaconda3)
(base) C:\Users\hansp>pip install pyodbc
Requirement already satisfied: pyodbc in c:\users\hansp\anaconda3\lib\site-packages (4.0.0-unsupported)
(base) C:\Users\hansp>
```

pip install pyodbc

# Connect to Database from Python

The newest and  
recommend driver



```
import pyodbc

driver = "{ODBC Driver 17 for SQL Server}"
server = "xxxxxx"
database = "xxxxxx"
username = "xxxxxx"
password = "xxxxxx"
conn = pyodbc.connect("DRIVER=" + driver
                      + ";SERVER=" + server
                      + ";DATABASE=" + database
                      + ";UID=" + username
                      + ";PWD=" + password )
```

# Connect to Database from Python

Example:

```
import pyodbc
```

```
driver = "{ODBC Driver 17 for SQL Server}"
```

```
server = "TESTPC\\SQLEXPRESS"
```

```
database = "BOOKSTORE"
```

```
username = "sa"
```

```
password = "Test123"
```

```
conn = pyodbc.connect("DRIVER=" + driver  
                        + ";SERVER=" + server  
                        + ";DATABASE=" + database  
                        + ";UID=" + username  
                        + ";PWD=" + password )
```

Server Name

If Server is on your local PC,  
you can use LOCALHOST

Instance Name (you can have  
multiple instances of SQL Server  
on the same computer)

Here is the built-in "sa" user (System Administrator) used to connect to the Database. In general, you should use another user than the sa user. The sa user is used here for simplicity. You can easily create a new user in SQL Server Management Studio





# Microsoft Azure

# Microsoft Azure

- Microsoft Azure is a Cloud Platform from Microsoft
- You could say it is “Windows running in the Cloud”
- Here you can host Databases, Web Applications, Virtual Machines, etc.
- Azure Portal:  
<https://portal.azure.com>



# Databases in Microsoft Azure

Hans-Petter Halvorsen

[Table of Contents](#)

# Configure Database in Azure

Home >

## SQL databases

Default Directory

[+ Create](#) [Reservations](#) [Manage view](#) [Refresh](#) [Export to CSV](#) [Open query](#) [Assign tags](#)

Filter for any field...

Subscription == **Azure for Students**

Resource group == **all**

Location == **all**

Showing 1 to 1 of 1 records.

<input type="checkbox"/> Name ↑↓	Server ↑↓	Replica type ↑↓
<input type="checkbox"/>  LOGGINGSYSTEM (hph/LOGGINGSYSTEM)	hph	--

Home > SQL databases >

## Create SQL Database

Microsoft

[Basics](#) [Networking](#) [Security](#) [Additional settings](#) [Tags](#) [Review + create](#)

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. [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 \*   
Resource group \*   
[Create new](#)

### Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name \*   
Server \*   
[Create new](#)

Want to use SQL elastic pool? \*  Yes  No

Compute + storage \* **General Purpose**  
Gen5, 2 vCores, 32 GB storage, zone redundant disabled  
[Configure database](#)

### Backup storage redundancy

Choose how your PITR and LTR backups are replicated. Geo restore or ability to recover from regional outage is only available when geo-redundant storage is selected.

Backup storage redundancy  Locally-redundant backup storage  
 Zone-redundant backup storage  
 Geo-redundant backup storage

 Selected value for backup storage redundancy is Geo-redundant backup

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

# Create Table

We will use SQL Server Management Studio and connect to the Azure Database:

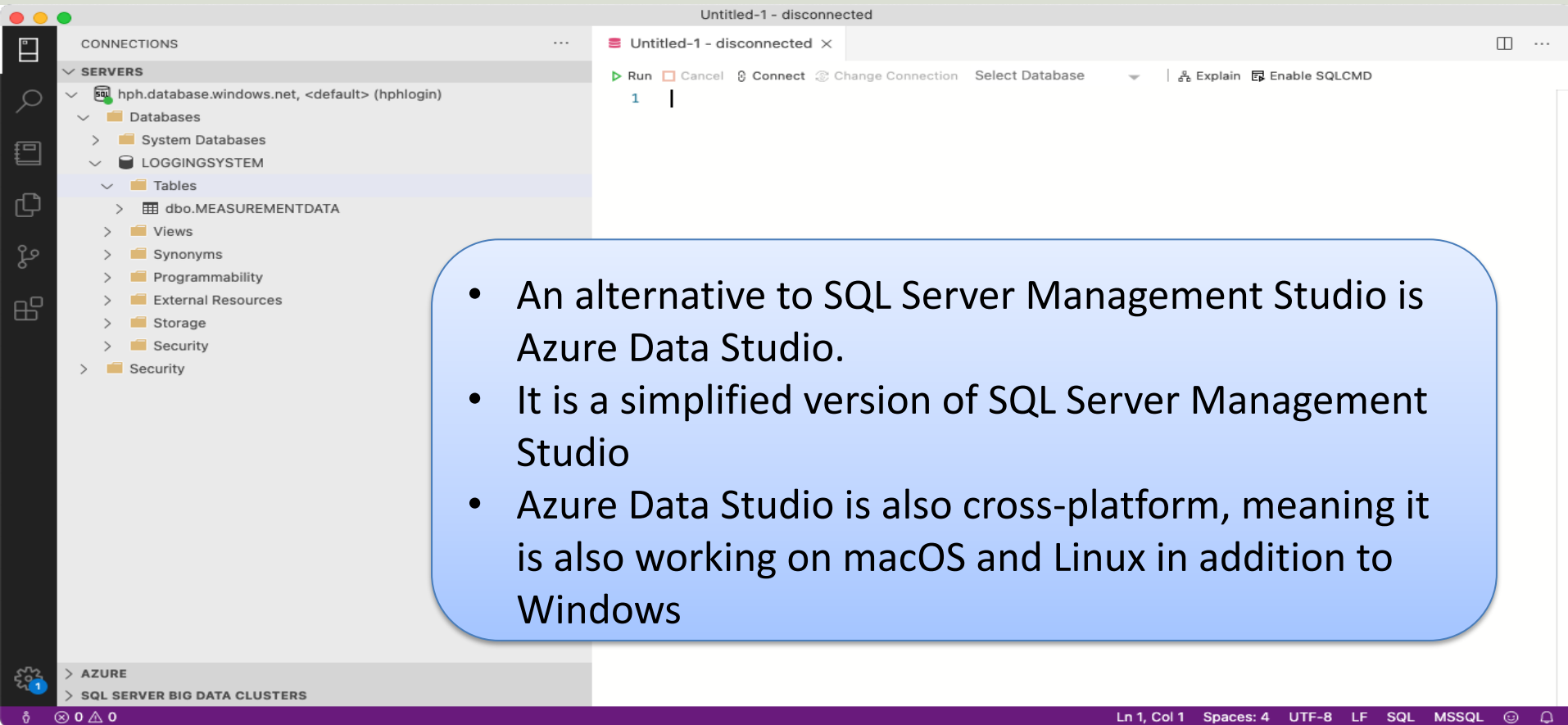
The screenshot displays the Microsoft SQL Server Management Studio (SSMS) interface. The main window title is "SQLQuery3.sql - hph.database.windows.net.LOGGINGSYSTEM (hphlogin (90))\* - Microsoft SQL Server Management Studio". The Object Explorer on the left shows the server hierarchy for "hph.database.windows.net (SQL Server 12.0.2000.8 - hphlogin)", with the "dbo.MEASUREMENTDATA" table selected. The SQL query editor on the right contains the following SQL code:

```
CREATE TABLE [MEASUREMENTDATA]
(
    [MeasurementId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
    [SensorName] [varchar](50) NOT NULL,
    [MeasurementValue] float NOT NULL,
    [MeasurementDateTime] datetime NOT NULL
)
GO
```

In the foreground, a "Connect to Server" dialog box is open, titled "SQL Server". The "Server type" is set to "Database Engine". The "Server name" is "hph.database.windows.net", "Authentication" is "SQL Server Authentication", and the "Login" is "hphlogin". The "Remember password" checkbox is checked. The "Connect" button is highlighted.

At the bottom of the SSMS window, the status bar shows "Connected. (1/1)" and "hph.database.windows.net (1... hphlogin (90) LOGGINGSYSTEM | 00:00:00 | 0 rows". The status bar at the very bottom of the image shows "Ln 9 Col 1 Ch 1 INS".

# Azure Data Studio



The screenshot displays the Azure Data Studio interface. On the left, the 'CONNECTIONS' pane shows a tree view under 'SERVERS' for 'hph.database.windows.net, <default> (hphlogin)'. The tree includes folders for 'Databases', 'System Databases', 'LOGGINGSYSTEM', 'Tables', 'Views', 'Synonyms', 'Programmability', 'External Resources', 'Storage', 'Security', and 'Security'. The 'Tables' folder is expanded, showing 'dbo.MEASUREMENTDATA'. The main editor area shows a query editor with a single line of code: '1 |'. The status bar at the bottom indicates 'Ln 1, Col 1 Spaces: 4 UTF-8 LF SQL MSSQL'.

- An alternative to SQL Server Management Studio is Azure Data Studio.
- It is a simplified version of SQL Server Management Studio
- Azure Data Studio is also cross-platform, meaning it is also working on macOS and Linux in addition to Windows

# Azure Query Editor

The screenshot displays the Azure Query Editor interface for a SQL database named LOGGINGSYSTEM (hph/LOGGINGSYSTEM). The interface is divided into several sections:

- Navigation and Search:** At the top, there is a breadcrumb trail: Home > SQL databases > LOGGINGSYSTEM (hph/LOGGINGSYSTEM). Below this, the page title is "LOGGINGSYSTEM (hph/LOGGINGSYSTEM) | Query editor (preview)". A search bar (Search (Cmd+)) and navigation links (Login, New Query, Open query, Feedback) are present.
- Left Sidebar (Navigation):** This sidebar contains various options for the database:
  - SQL databases: + Create, Reservations
  - Filter for any field...
  - Name ↑
  - LOGGINGSYSTEM (hph/LOGGINGSYSTEM) ...
  - Overview
  - Activity log
  - Tags
  - Diagnose and solve problems
  - Quick start
  - Query editor (preview) (highlighted)
  - Power Platform
    - Power BI
    - Power Apps
    - Power Automate
  - Settings
    - Compute + storage
    - Connection strings
    - Properties
    - Locks
  - Data management
    - Replicas
    - Sync to other databases
  - Integrations
    - Stream analytics (preview)
    - Add Azure Search
- Object Explorer (Center):** Shows the database structure for LOGGINGSYSTEM (hphlogin). A message states: "Showing limited object explorer here. For full capability please open SSDT." The tree view includes:
  - Tables
  - Views
  - Stored Procedures
- Query Editor (Right):** Labeled "Query 1", it features a toolbar with "Run", "Cancel query", "Save query", "Export data as", and "Show only Editor". Below the toolbar is a text area for writing queries, currently containing the number "1".
- Results and Messages (Bottom Right):** Includes tabs for "Results" and "Messages", and a search bar: "Search to filter items..."

A 3.alternative is the Query Editor  
in the Microsoft Azure Portal

# Firewall

We need to give access to the computers running the Python Scripts

The screenshot shows the Microsoft Azure portal interface. At the top, there is a search bar and navigation links. The main content area is divided into a left sidebar with navigation options like 'Overview', 'Activity log', and 'Tags', and a main panel. The main panel shows the 'LOGGINGSYSTEM (hph/LOGGINGSYSTEM)' resource, which is an SQL database. A red box highlights the 'Set server firewall' button in the top action bar. Below this, the 'Firewall settings' page is visible, showing various configuration options: 'Deny public network access' is unchecked, 'Minimum TLS Version' is set to 1.2, 'Connection Policy' is set to Default, and 'Allow Azure services and resources to access this server' is set to No. At the bottom, there is a table for 'Client IP address' with columns for 'Rule name', 'Start IP', and 'End IP', and a table with three empty input fields.

Microsoft Azure

Search resources, services, and docs (G+)

Home >

LOGGINGSYSTEM (hph/LOGGINGSYSTEM) SQL database

Search (Cmd+/) <<

Copy Restore Export Set server firewall Delete Connect with... Feedback

Overview

Activity log

Tags

Diagnose and solve problems

Quick start

Query editor (preview)

Essentials

Resource group (Move) : halvorsen

Status : Online

Location : West Europe

Subscription (Move) : Azure for Students

Subscription ID : 3c6a9d07-b932-4d

Tags (Edit) : Click here to add t

Microsoft Azure

Search resources, services, and docs (G+)

Home > LOGGINGSYSTEM (hph/LOGGINGSYSTEM) >

Firewall settings hph (SQL server)

Save Discard Add client IP

Deny public network access

Minimum TLS Version ⓘ

1.0 1.1 1.2

Connection Policy ⓘ

Default Proxy Redirect

Allow Azure services and resources to access this server ⓘ

Yes No

Client IP address 128.39.132.145

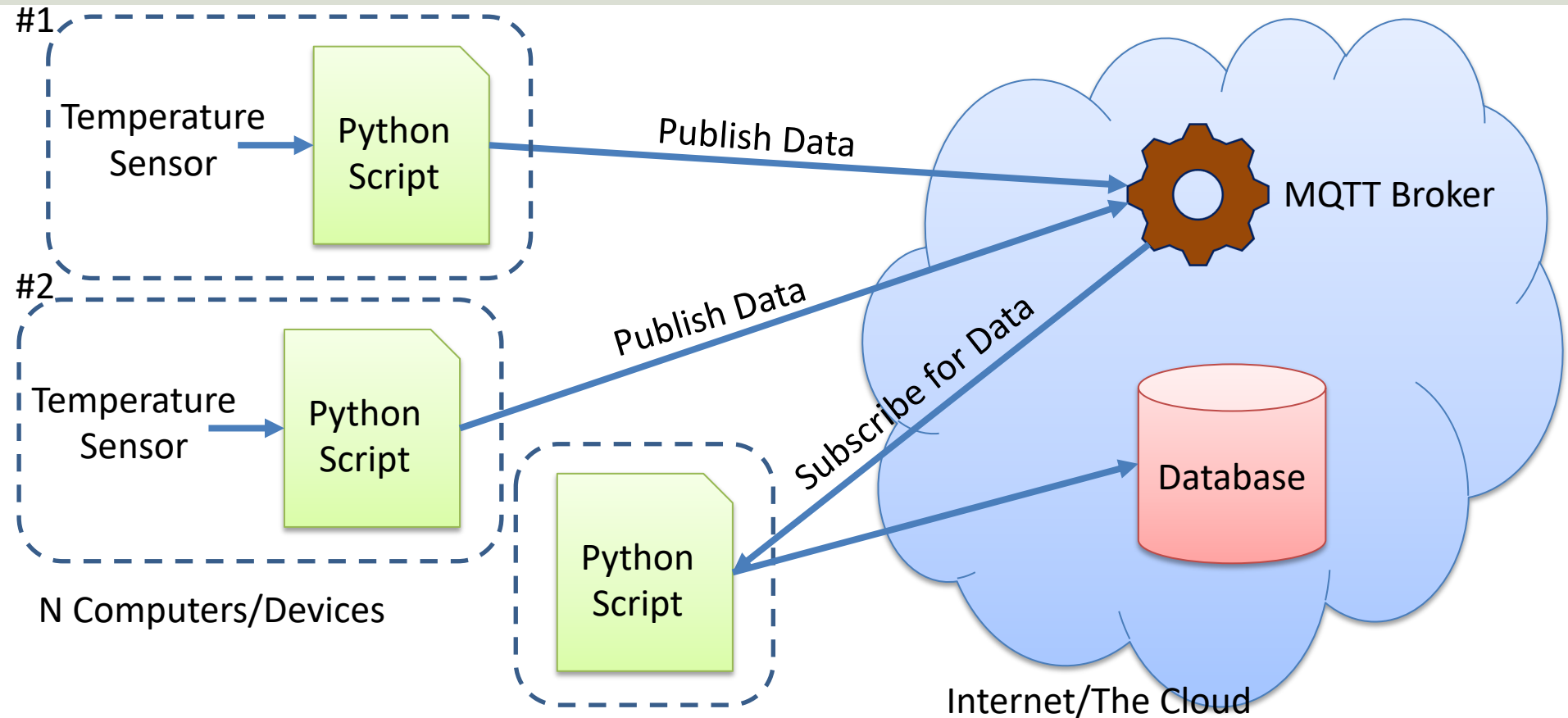
Rule name	Start IP	End IP





# Code Examples

# System Overview



# MQTT Broker

The MQTT Broker Data is put into a Python File called “broker.py”:

broker.py

```
def GetBroker () :  
    brokerAddress = "xxxxxxxxxx.s2.eu.hivemq.cloud"  
    userName = "xxxxxx"  
    passWord = "xxxxxx"  
  
    return brokerAddress, userName, passWord
```

# Connection String

The Connection string has been put in a separate Python File called “database.py”:

database.py

```
def GetConnectionStringAzure () :  
    driver = "{ODBC Driver 17 for SQL Server}"  
    server = "xxx.database.windows.net"  
    database = "LOGGINGSYSTEM"  
    username = "xxxxxxx"  
    password = "xxxxxxx"  
  
    connectionString = "DRIVER=" + driver + ";SERVER=" + server + ";DATABASE=" +  
        + database + ";UID=" + username + ";PWD=" + password  
  
    return connectionString
```

## Publish Temperature1 to HiveMQ Cloud.py

```
import paho.mqtt.client as mqtt
import random
import time
import broker

#MQTT Settings
brokerAddress, userName, passWord = broker.GetBroker()
topic = "Sensor/Temperature/Livingroom"

min = 20
max = 30

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))

# create the client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)

# Publish Temperature Data
wait = 20
while True:
    data = random.randint(min, max)
    print(data)
    client.publish(topic, data)
    time.sleep(wait)
```

## Publish Temperature2 to HiveMQ Cloud.py

```
import paho.mqtt.client as mqtt
import random
import time
import broker

#MQTT Settings
brokerAddress, userName, passWord = broker.GetBroker()
topic = "Sensor/Temperature/Kitchen"

min = 20
max = 30

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print("Received message: " + msg.topic + " -> " + msg.payload.decode("utf-8"))

# create the client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)

# Publish Temperature Data
wait = 20
while True:
    data = random.randint(min, max)
    print(data)
    client.publish(topic, data)
    time.sleep(wait)
```

# Subscribe on Topic in HiveMQ Cloud and Insert Data in SQL Server.py

```
import paho.mqtt.client as mqtt
import pyodbc
from datetime import datetime
import broker
import database
```

```
#MQTT Settings
brokerAddress, userName, passWord = broker.GetBroker()
subscribeTopic = "Sensor/Temperature/#"
```

```
# Connect to Database
connectionString = database.GetConnectionStringAzure()
conn = pyodbc.connect(connectionString)
cursor = conn.cursor()
```

```
# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected successfully")
    else:
        print("Connect returned result code: " + str(rc))
```

```
# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    topic = msg.topic
    measurementValue = msg.payload.decode("utf-8")
    SaveToDatabase(topic, measurementValue)
```

```
def SaveToDatabase(topic, measurementValue):
    print(topic + " " + measurementValue)
```

```
#Find Date and Time
now = datetime.now()
datetimeformat = "%Y-%m-%d %H:%M:%S"
measurementDateTime = now.strftime(datetimeformat)
```

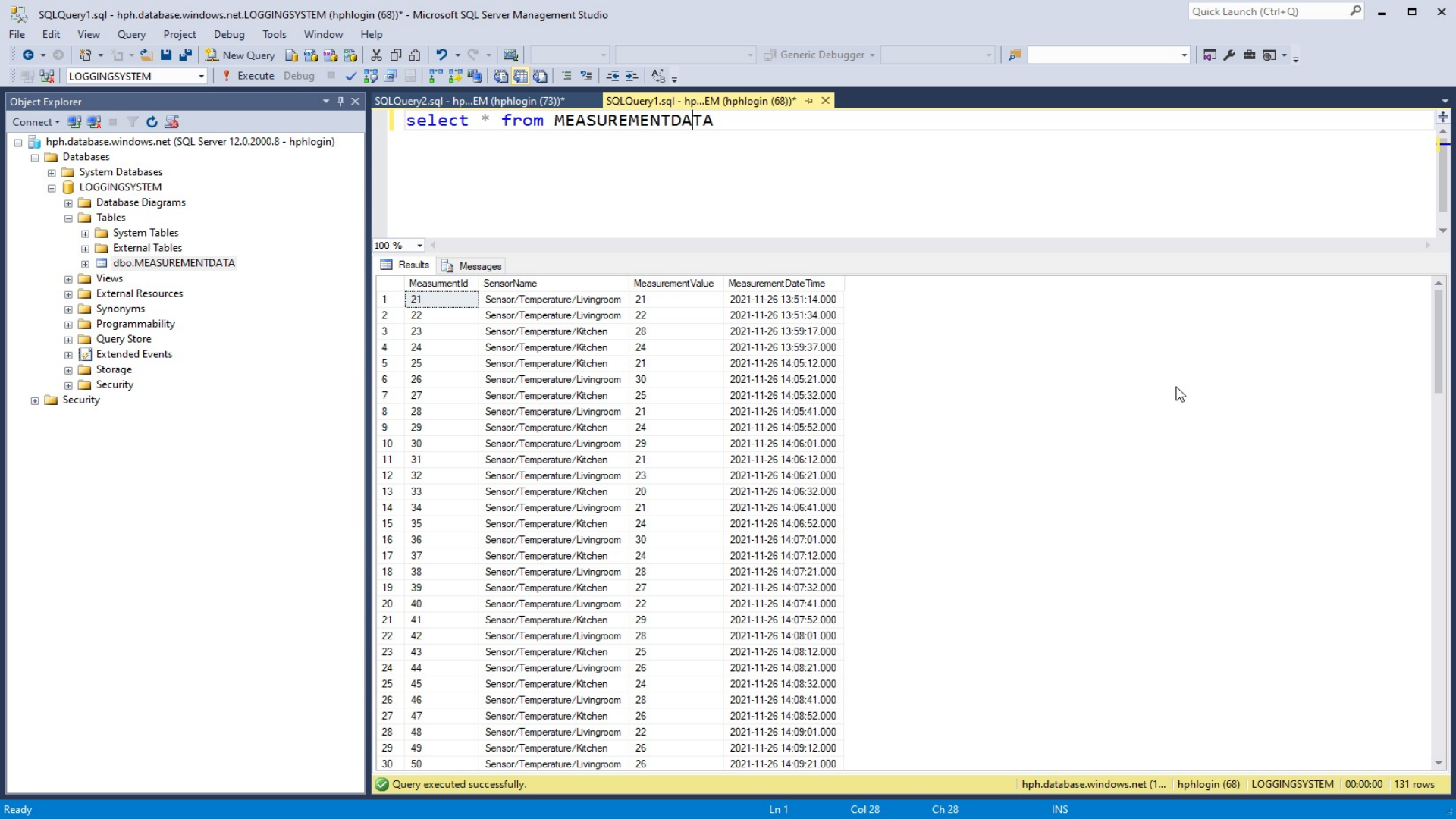
```
# Insert Data into Database
query = "INSERT INTO MEASUREMENTDATA (SensorName, MeasurementValue, MeasurementDateTime) VALUES (?, ?, ?)"
sensorName = topic
parameters = sensorName, measurementValue, measurementDateTime
cursor.execute(query, parameters)
cursor.commit()
```

```
# Create the MQTT client
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
```

```
client.tls_set(tls_version=mqtt.ssl.PROTOCOL_TLS)
client.username_pw_set(userName, passWord)
client.connect(brokerAddress, 8883)
```

```
client.subscribe(subscribeTopic)
```

```
client.loop_forever()
```



Object Explorer

- Connect
- hph.database.windows.net (SQL Server 12.0.2000.8 - hphlogin)
  - Databases
    - System Databases
    - LOGGINGSYSTEM
      - Database Diagrams
      - Tables
        - System Tables
        - External Tables
        - dbo.MEASUREMENTDATA
      - Views
      - External Resources
      - Synonyms
      - Programmability
      - Query Store
      - Extended Events
      - Storage
      - Security
    - Security

```
SQLQuery2.sql - hp...EM (hphlogin (73))* SQLQuery1.sql - hp...EM (hphlogin (68))* - X  
select * from MEASUREMENTDATA
```

100 %

Results Messages

MeasurementId	SensorName	MeasurementValue	MeasurementDateTime
1	Sensor/Temperature/Livingroom	21	2021-11-26 13:51:14.000
2	Sensor/Temperature/Livingroom	22	2021-11-26 13:51:34.000
3	Sensor/Temperature/Kitchen	28	2021-11-26 13:59:17.000
4	Sensor/Temperature/Kitchen	24	2021-11-26 13:59:37.000
5	Sensor/Temperature/Kitchen	21	2021-11-26 14:05:12.000
6	Sensor/Temperature/Livingroom	30	2021-11-26 14:05:21.000
7	Sensor/Temperature/Kitchen	25	2021-11-26 14:05:32.000
8	Sensor/Temperature/Livingroom	21	2021-11-26 14:05:41.000
9	Sensor/Temperature/Kitchen	24	2021-11-26 14:05:52.000
10	Sensor/Temperature/Livingroom	29	2021-11-26 14:06:01.000
11	Sensor/Temperature/Kitchen	21	2021-11-26 14:06:12.000
12	Sensor/Temperature/Livingroom	23	2021-11-26 14:06:21.000
13	Sensor/Temperature/Kitchen	20	2021-11-26 14:06:32.000
14	Sensor/Temperature/Livingroom	21	2021-11-26 14:06:41.000
15	Sensor/Temperature/Kitchen	24	2021-11-26 14:06:52.000
16	Sensor/Temperature/Livingroom	30	2021-11-26 14:07:01.000
17	Sensor/Temperature/Kitchen	24	2021-11-26 14:07:12.000
18	Sensor/Temperature/Livingroom	28	2021-11-26 14:07:21.000
19	Sensor/Temperature/Kitchen	27	2021-11-26 14:07:32.000
20	Sensor/Temperature/Livingroom	22	2021-11-26 14:07:41.000
21	Sensor/Temperature/Kitchen	29	2021-11-26 14:07:52.000
22	Sensor/Temperature/Livingroom	28	2021-11-26 14:08:01.000
23	Sensor/Temperature/Kitchen	25	2021-11-26 14:08:12.000
24	Sensor/Temperature/Livingroom	26	2021-11-26 14:08:21.000
25	Sensor/Temperature/Kitchen	24	2021-11-26 14:08:32.000
26	Sensor/Temperature/Livingroom	28	2021-11-26 14:08:41.000
27	Sensor/Temperature/Kitchen	26	2021-11-26 14:08:52.000
28	Sensor/Temperature/Livingroom	22	2021-11-26 14:09:01.000
29	Sensor/Temperature/Kitchen	26	2021-11-26 14:09:12.000
30	Sensor/Temperature/Livingroom	26	2021-11-26 14:09:21.000



# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

