https://www.halvorsen.blog

Python MQTT, SQL Server and Microsoft Azure

Hans-Petter Halvorsen

Contents

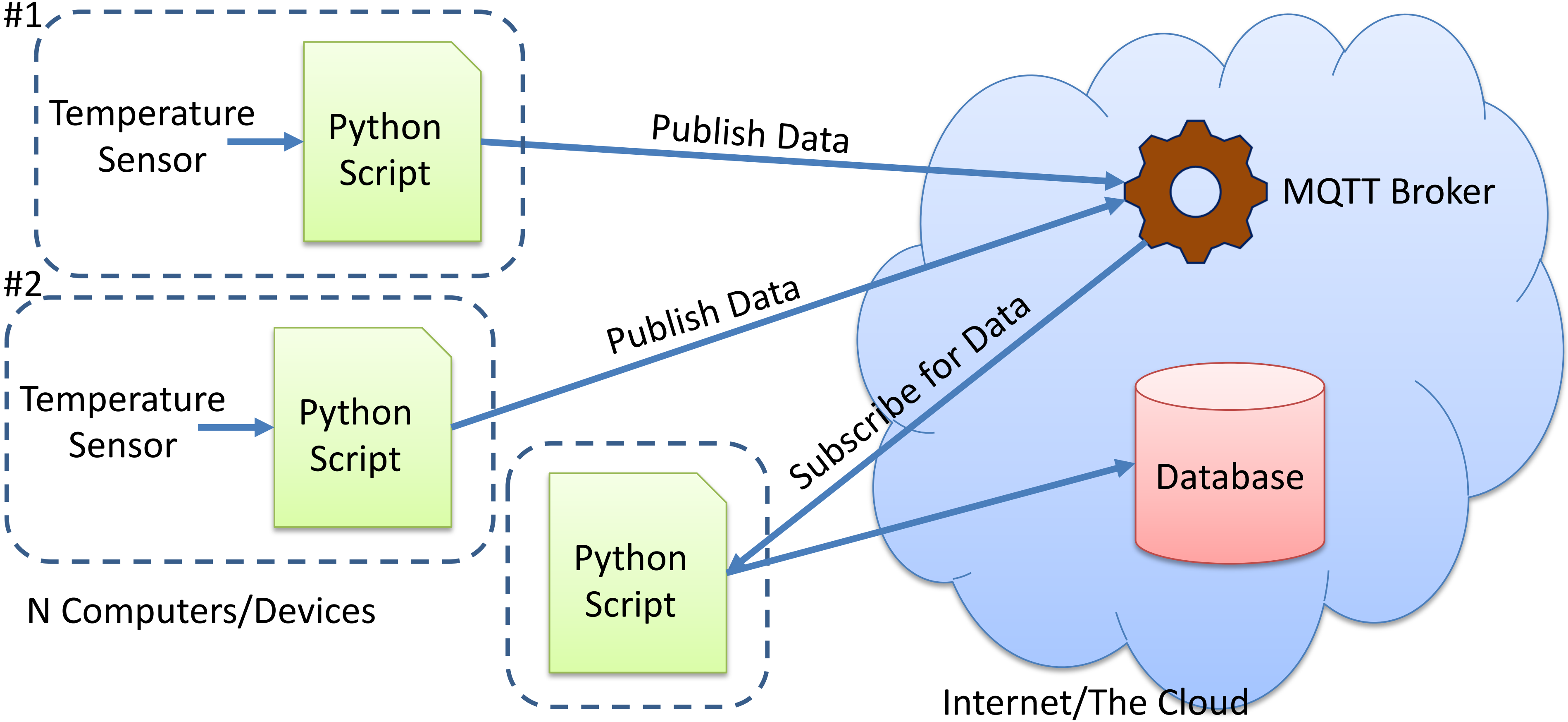
- Introduction
- MQTT
- SQL Server
- Python and SQL Server
- Microsoft Azure
- Databases in Microsoft Azure
- Code Examples

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

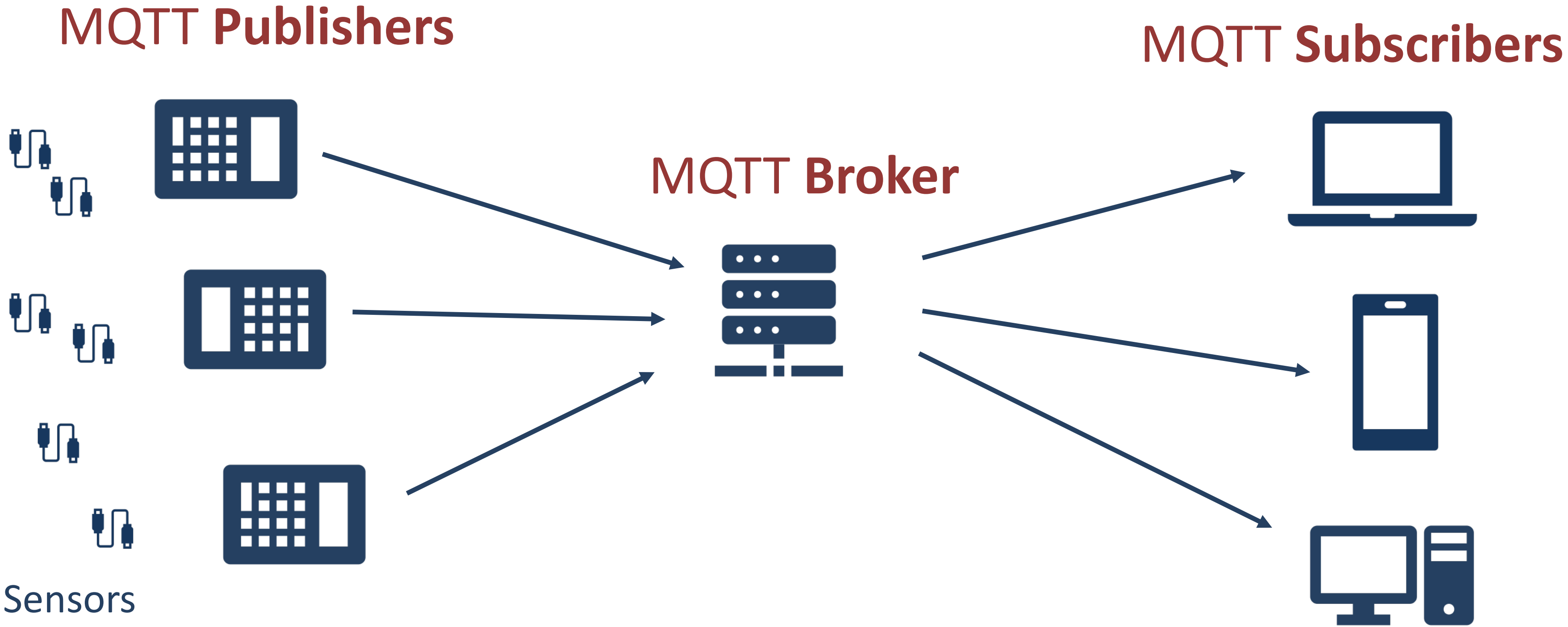
Hans-Petter Halvorsen

Table of Contents

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

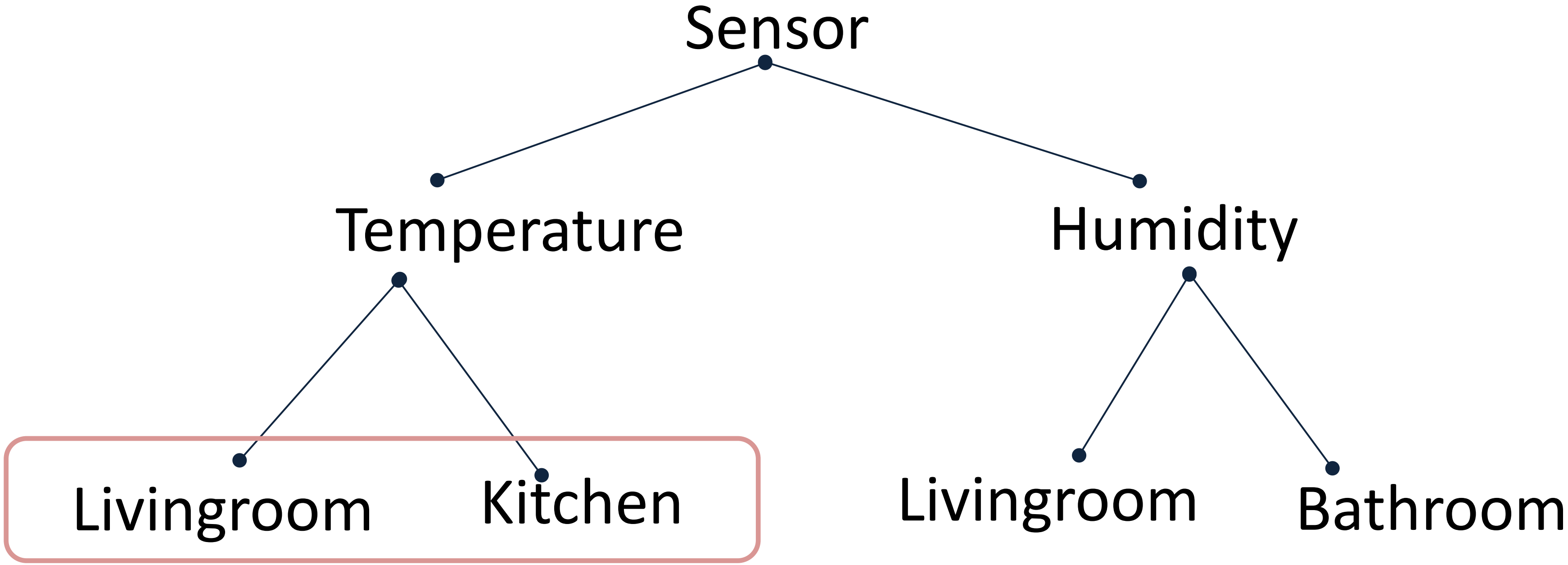
Example:

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

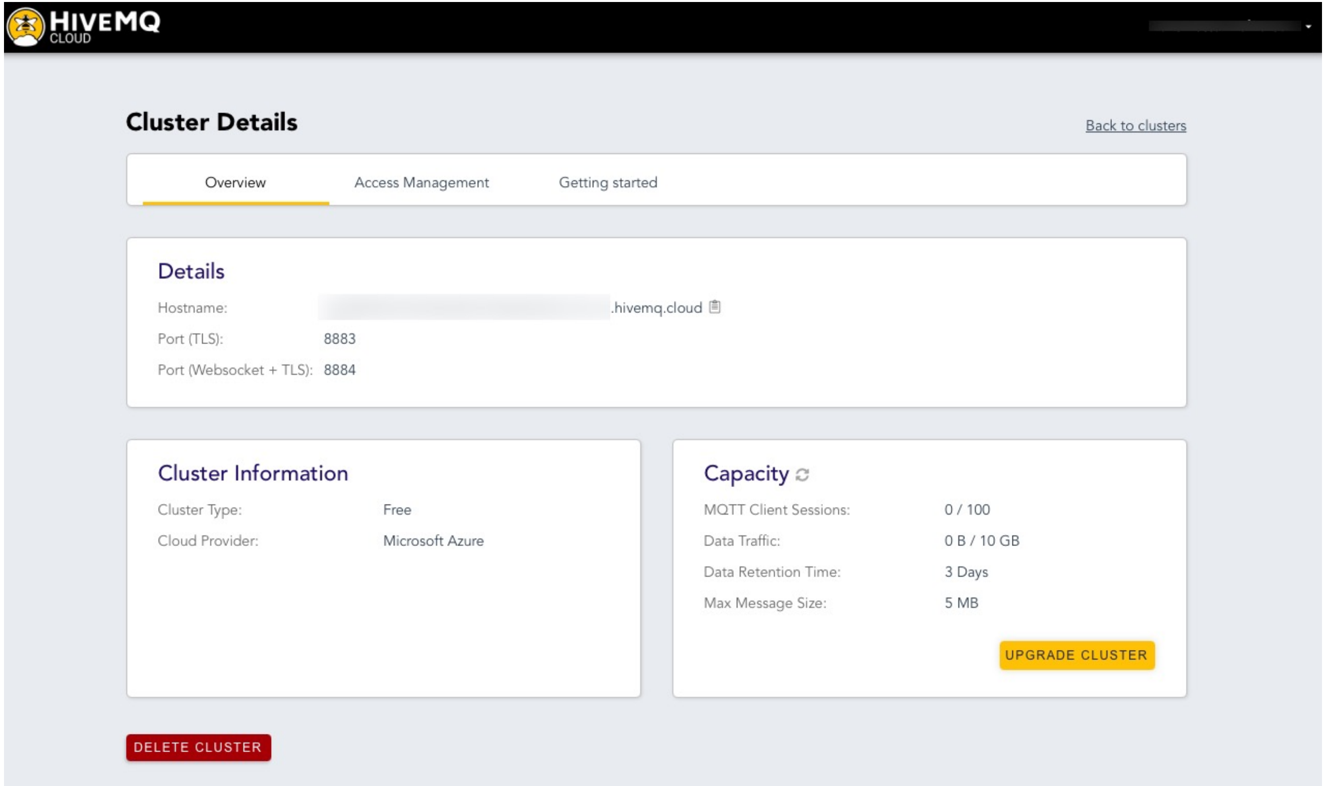
Subscribe on Topics - Wildcards

Wildcards: Sensor/Temperature/#



HiveMQ Cloud

https://www.hivemq.com



Cluster Details

Details

Hostname: [redacted].hivemq.cloud

Port (TLS): 8883

Port (Websocket + TLS): 8884

Cluster Information

Cluster Type: Free

Cloud Provider: Microsoft Azure

Capacity

MQTT Client Sessions: 0 / 100

Data Traffic: 0 B / 10 GB

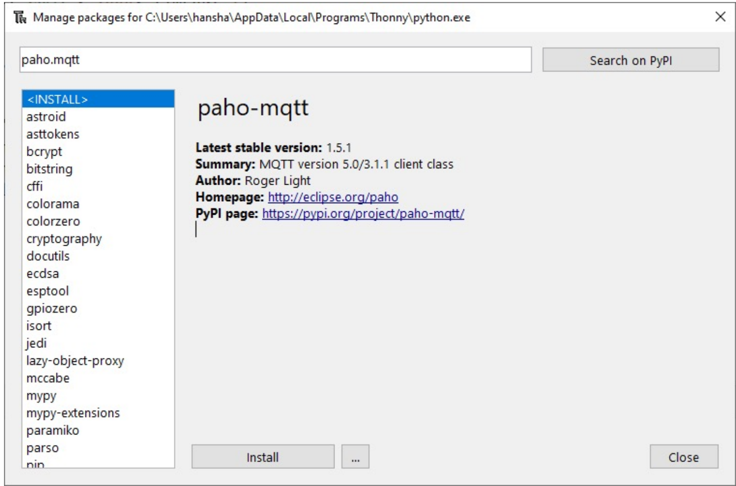
Data Retention Time: 3 Days

Max Message Size: 5 MB

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



Manage packages for C:\Users\hansha\AppData\Local\Programs\Thonny\python.exe

paho.mqtt

paho-mqtt

Latest stable version: 1.5.1
Summary: MQTT version 5.0/3.1.1 client class
Author: Roger Light
Homepage: http://eclipse.org/paho
PyPI page: https://pypi.org/project/paho-mqtt/

SQL Server

Database Systems

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

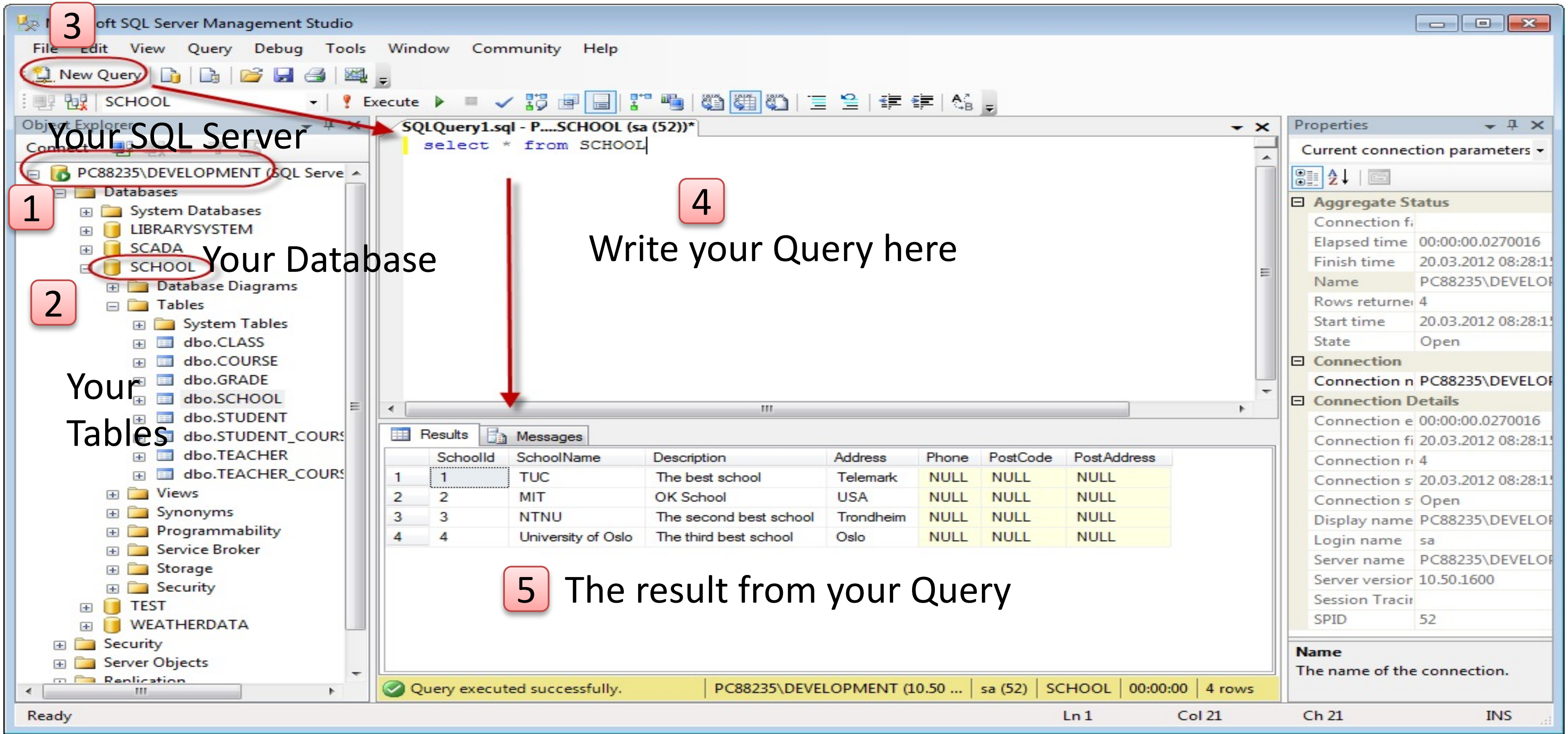
SQL Server

- 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

```
select * from SCHOOL
```

5 The result from your Query

	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

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

Aggregate Status	
Connection fi	
Elapsed time	00:00:00.0270016
Finish time	20.03.2012 08:28:1!
Name	PC88235\DEVELOI
Rows returne	4
Start time	20.03.2012 08:28:1!
State	Open
Connection	
Connection n	PC88235\DEVELOI
Connection Details	
Connection e	00:00:00.0270016
Connection fi	20.03.2012 08:28:1!
Connection r	4
Connection s	20.03.2012 08:28:1!
Connection s	Open
Display name	PC88235\DEVELOI
Login name	sa
Server name	PC88235\DEVELOI
Server versior	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

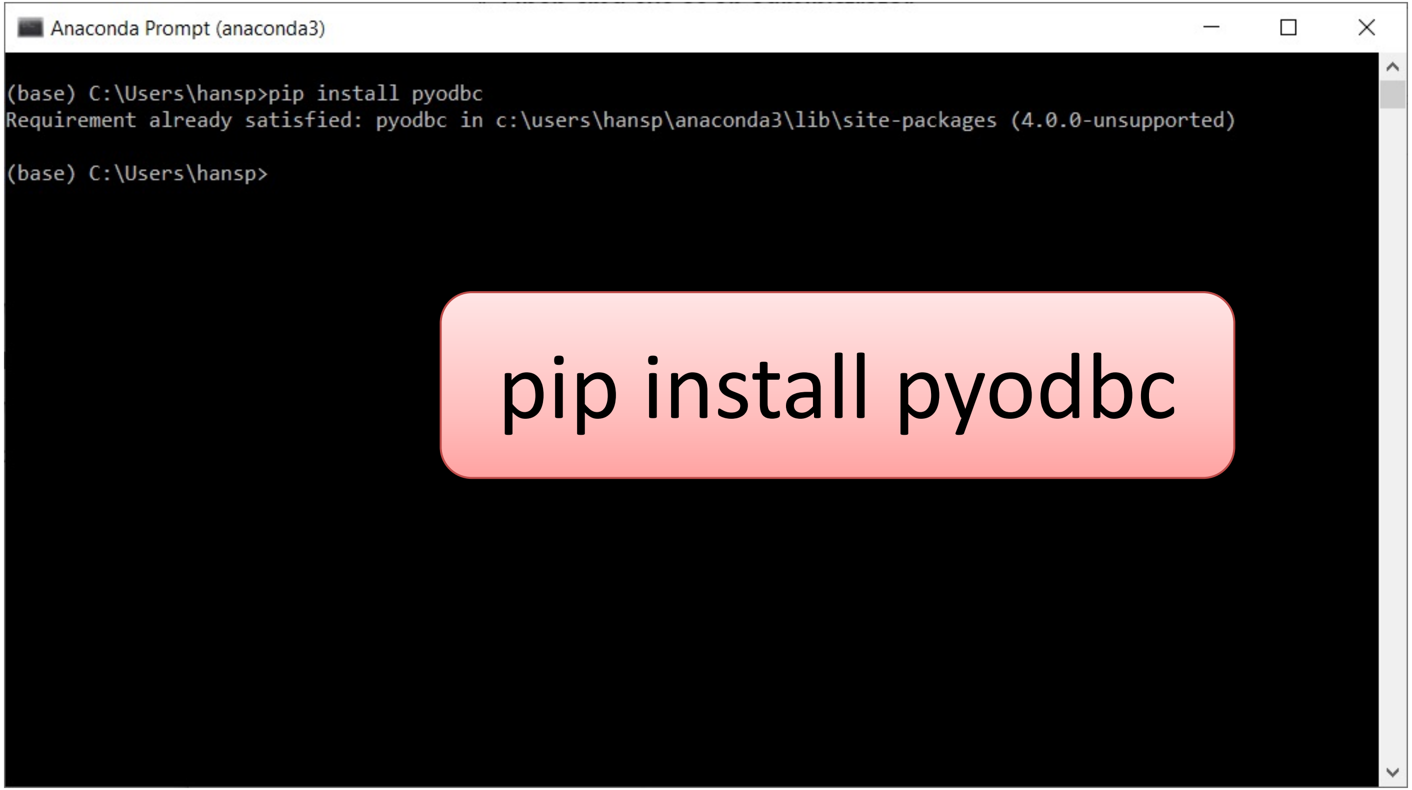
- There are several python SQL drivers available:
  - pyodbc
  - pymssql
- These Drivers are not made made Microsoft but the Python Community.
- However, Microsoft places its testing efforts and its confidence in pyodbc driver.
- Microsoft contributes to the pyODBC open-source community and is an active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



```
(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 = "xxxxx"
username = "xxxxx"
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

Table of Contents

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

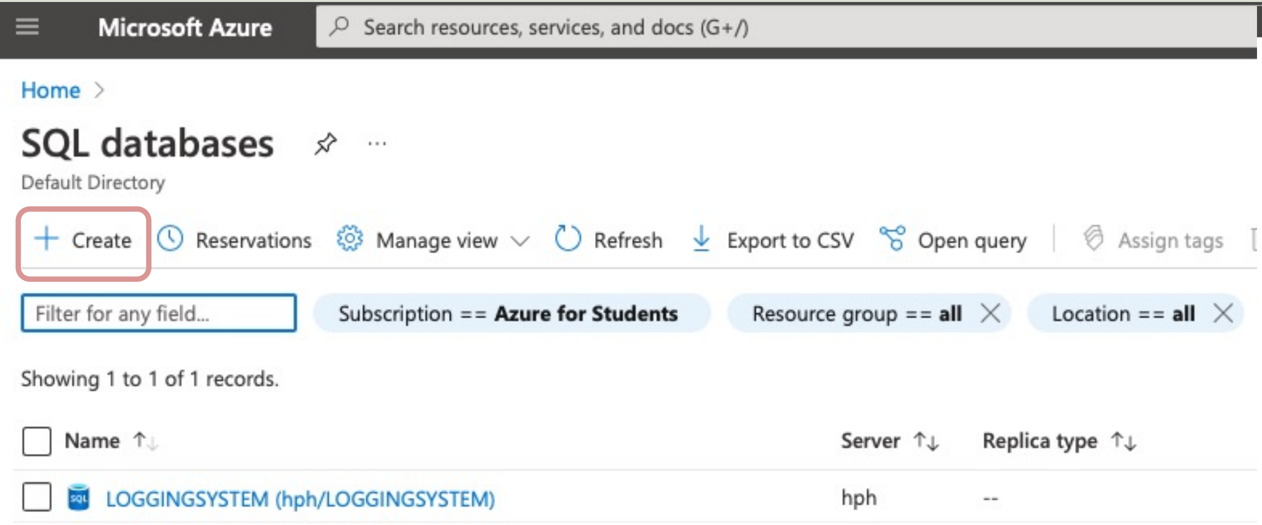
https://www.halvorsen.blog

Databases in Microsoft Azure

Hans-Petter Halvorsen

Table of Contents

Configure Database in Azure



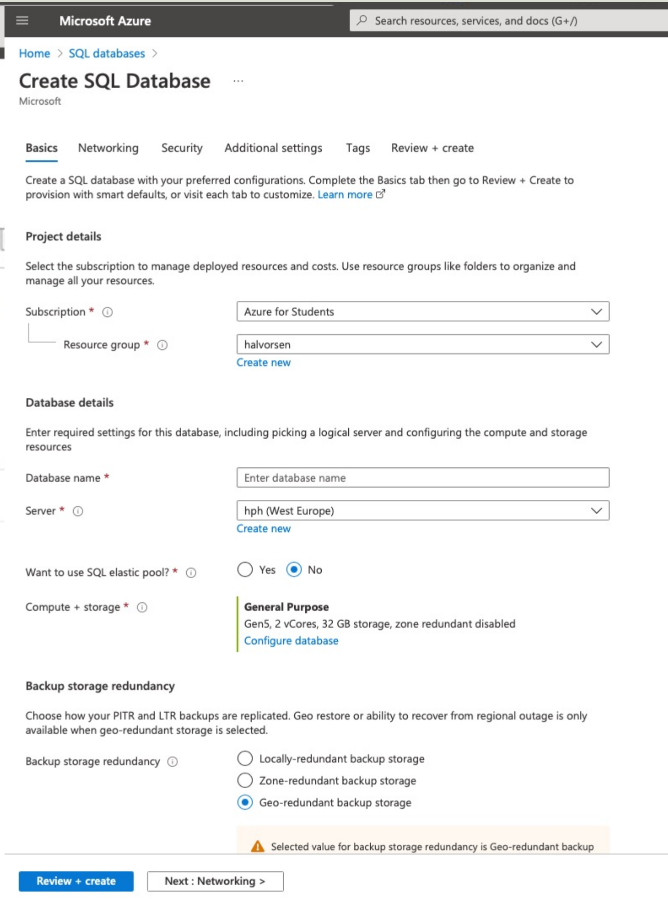
SQL databases

Default Directory

Subscription == **Azure for Students** | Resource group == **all** | Location == **all**

Showing 1 to 1 of 1 records.

Name	Server	Replica type
LOGGINGSYSTEM (hph/LOGGINGSYSTEM)	hph	--



Create SQL Database

Microsoft

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 * : Azure for Students

Resource group * : halvorsen

Database details

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

Database name * : Enter database name

Server * : hph (West Europe)

Want to use SQL elastic pool? * : ○ Yes ● No

Compute + storage * : **General Purpose**
Gen5, 2 vCores, 32 GB storage, zone redundant disabled

Backup storage redundancy

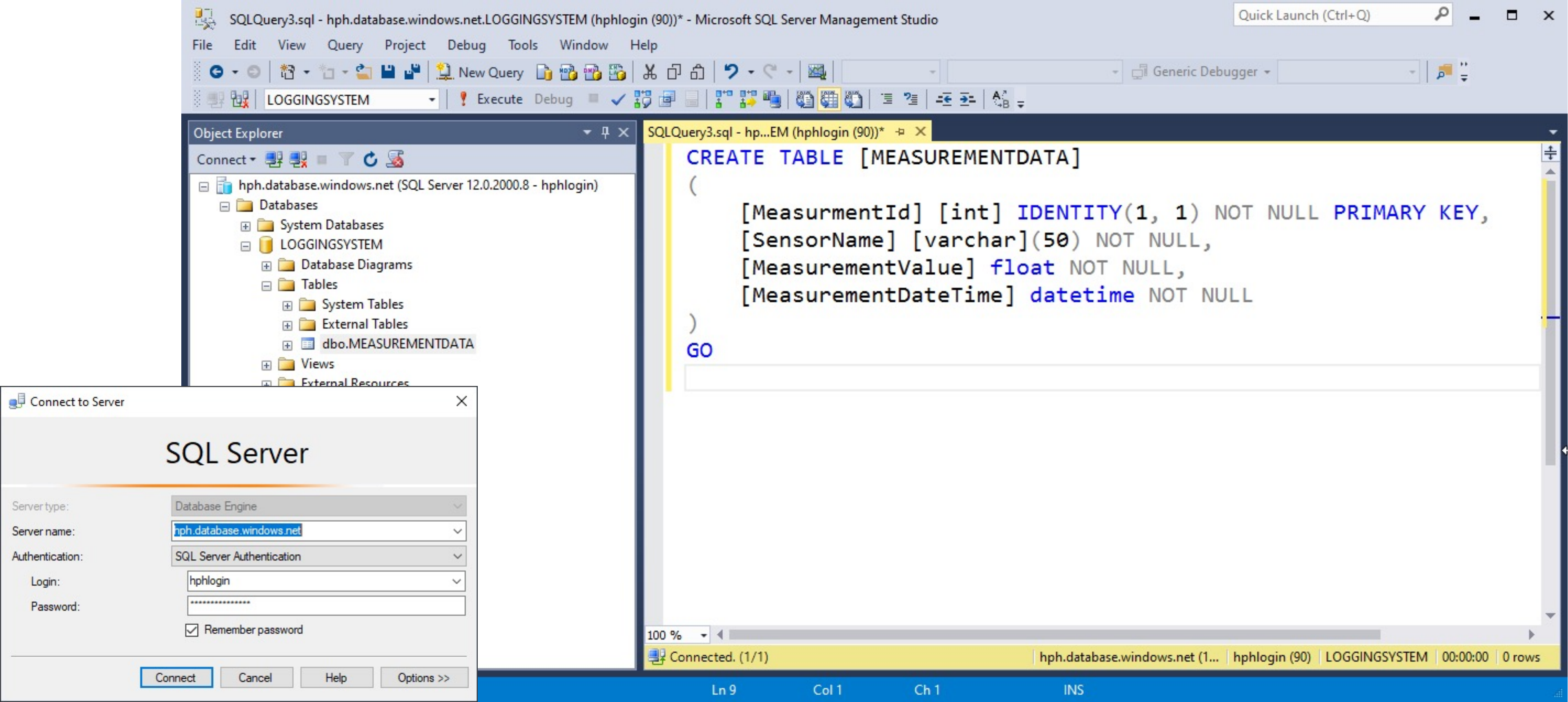
Choose how your PITR and LTR backups are replicated. Geo restore or ability to recover from regional outage is only available when geo-redundant storage is selected.

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

Create Table

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



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

Server type: Database Engine

Server name: hph.database.windows.net

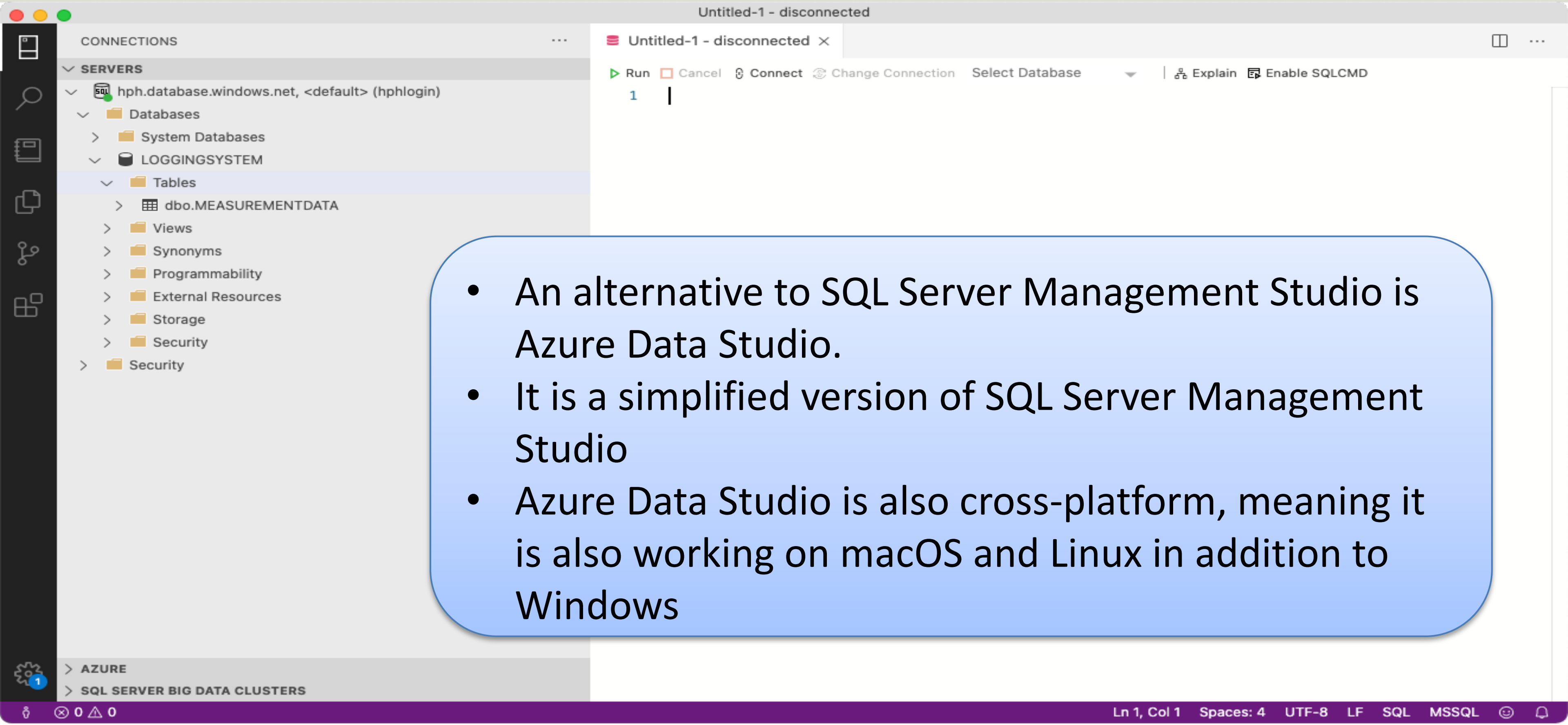
Authentication: SQL Server Authentication

Login: hphlogin

Password: ***************

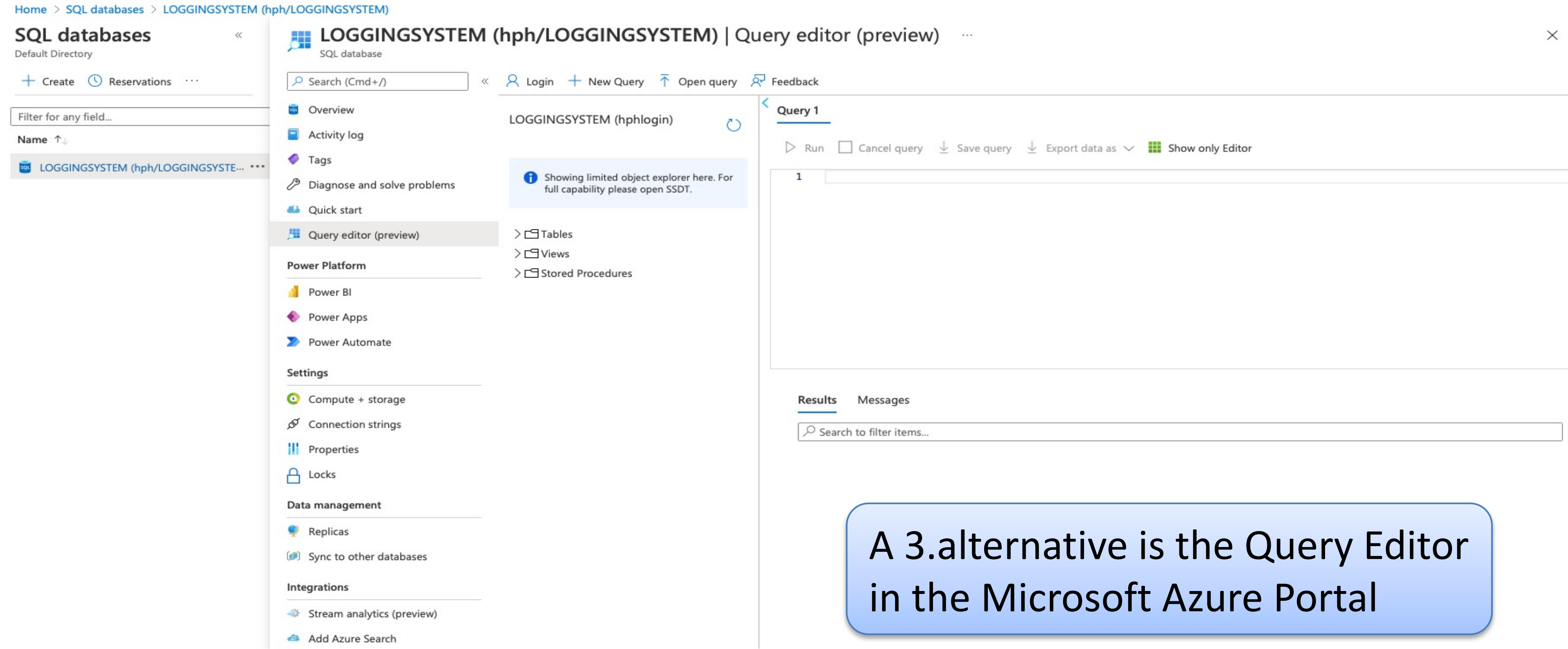
☑ Remember password

Azure Data Studio



- 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

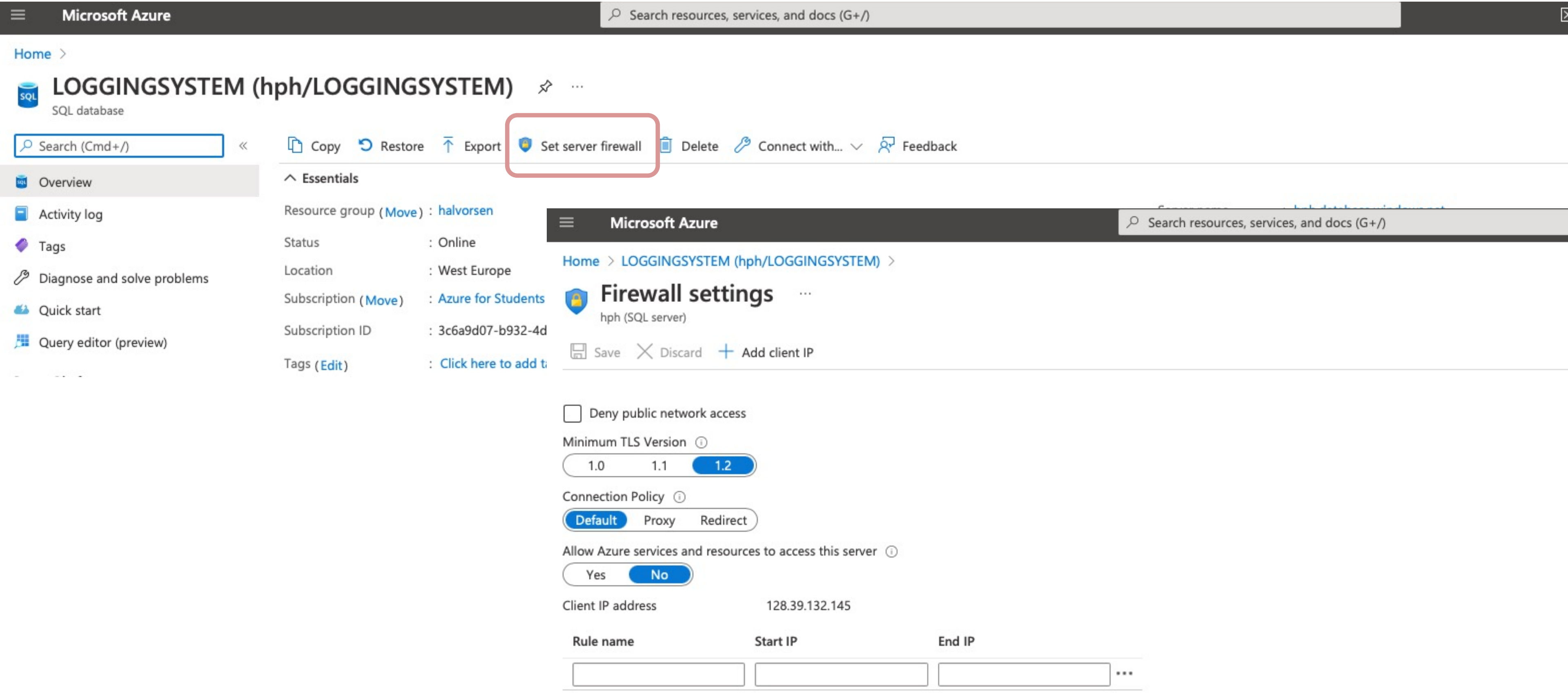


Showing limited object explorer here. For full capability please open SSDT.

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



LOGGINGSYSTEM (hph/LOGGINGSYSTEM)

SQL database

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) >
Save
Discard
Add client IP

Firewall settings

hph (SQL server)

□ Deny public network access

Minimum TLS Version: 1.0 1.1 [✓ 1.2]

Connection Policy: [✓ Default] Proxy Redirect

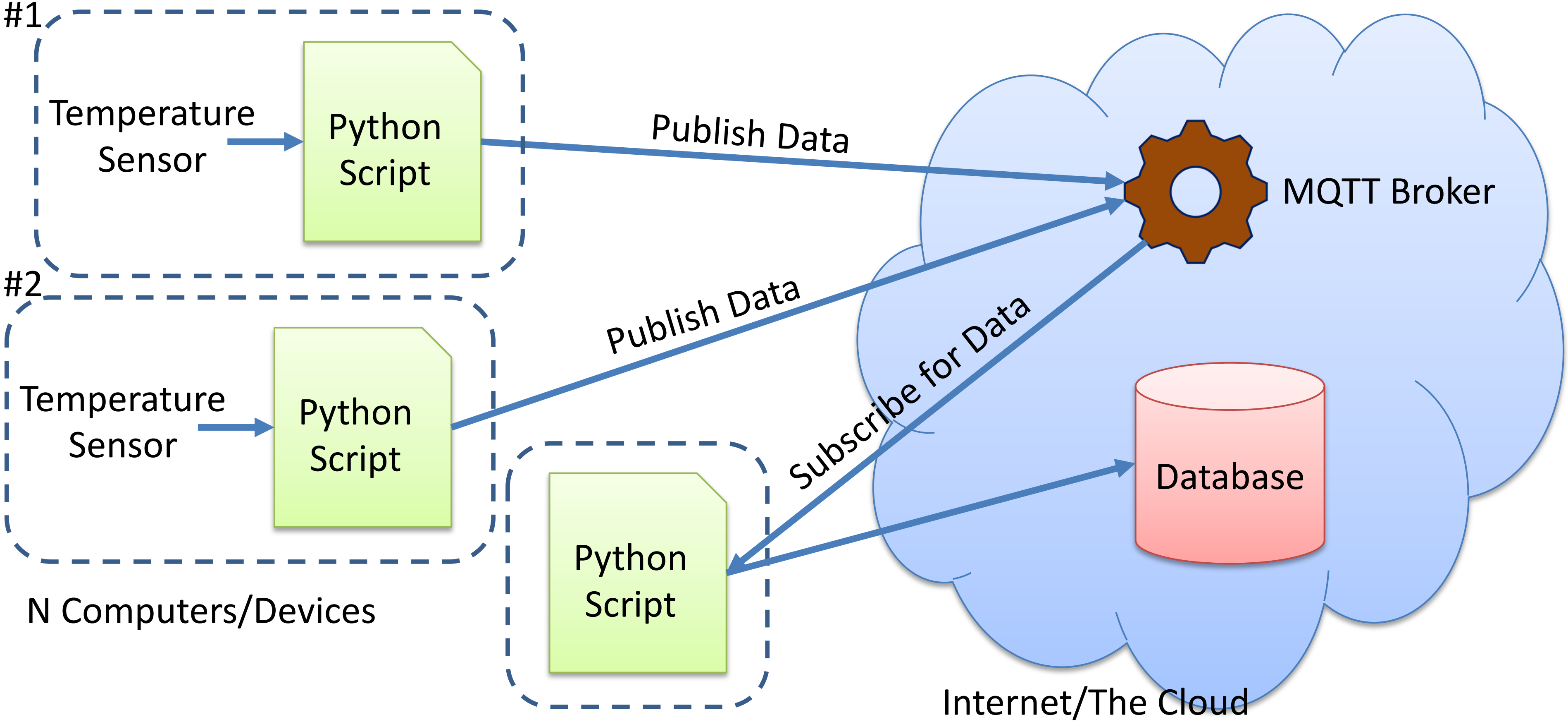
Allow Azure services and resources to access this server: Yes [✓ No]

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

```python
def GetConnectionStringAzure():
    driver = "{ODBC Driver 17 for SQL Server}"
    server = "xxx.database.windows.net"
    database = "LOGGINGSYSTEM"
    username = "xxxxxx"
    password = "xxxxxxx"

    connectionString = "DRIVER=" + driver + ";SERVER=" + server + ";DATABASE="
                 + database + ";UID=" + username + ";PWD=" + password

    return connectionString
```

Publish Temperature1 to HiveMQ Cloud.py

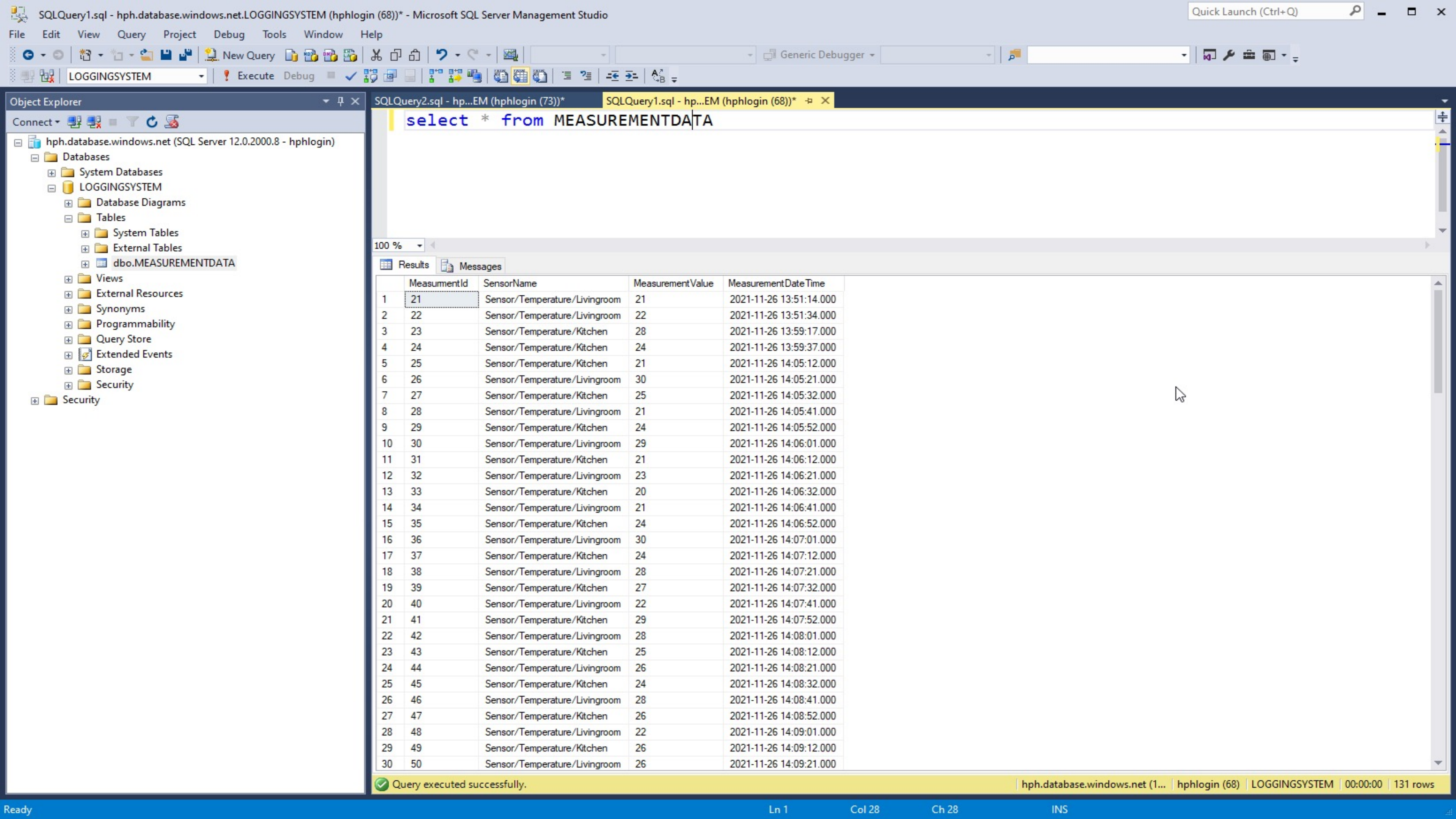
```python
import paho.mqtt.client as mqtt
import random
import time
import broker

#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

```python
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

```python
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()
```



```
select * from MEASUREMENTDATA
```

	MeasumentId	SensorName	MeasurementValue	MeasurementDateTime
1	21	Sensor/Temperature/Livingroom	21	2021-11-26 13:51:14.000
2	22	Sensor/Temperature/Livingroom	22	2021-11-26 13:51:34.000
3	23	Sensor/Temperature/Kitchen	28	2021-11-26 13:59:17.000
4	24	Sensor/Temperature/Kitchen	24	2021-11-26 13:59:37.000
5	25	Sensor/Temperature/Kitchen	21	2021-11-26 14:05:12.000
6	26	Sensor/Temperature/Livingroom	30	2021-11-26 14:05:21.000
7	27	Sensor/Temperature/Kitchen	25	2021-11-26 14:05:32.000
8	28	Sensor/Temperature/Livingroom	21	2021-11-26 14:05:41.000
9	29	Sensor/Temperature/Kitchen	24	2021-11-26 14:05:52.000
10	30	Sensor/Temperature/Livingroom	29	2021-11-26 14:06:01.000
11	31	Sensor/Temperature/Kitchen	21	2021-11-26 14:06:12.000
12	32	Sensor/Temperature/Livingroom	23	2021-11-26 14:06:21.000
13	33	Sensor/Temperature/Kitchen	20	2021-11-26 14:06:32.000
14	34	Sensor/Temperature/Livingroom	21	2021-11-26 14:06:41.000
15	35	Sensor/Temperature/Kitchen	24	2021-11-26 14:06:52.000
16	36	Sensor/Temperature/Livingroom	30	2021-11-26 14:07:01.000
17	37	Sensor/Temperature/Kitchen	24	2021-11-26 14:07:12.000
18	38	Sensor/Temperature/Livingroom	28	2021-11-26 14:07:21.000
19	39	Sensor/Temperature/Kitchen	27	2021-11-26 14:07:32.000
20	40	Sensor/Temperature/Livingroom	22	2021-11-26 14:07:41.000
21	41	Sensor/Temperature/Kitchen	29	2021-11-26 14:07:52.000
22	42	Sensor/Temperature/Livingroom	28	2021-11-26 14:08:01.000
23	43	Sensor/Temperature/Kitchen	25	2021-11-26 14:08:12.000
24	44	Sensor/Temperature/Livingroom	26	2021-11-26 14:08:21.000
25	45	Sensor/Temperature/Kitchen	24	2021-11-26 14:08:32.000
26	46	Sensor/Temperature/Livingroom	28	2021-11-26 14:08:41.000
27	47	Sensor/Temperature/Kitchen	26	2021-11-26 14:08:52.000
28	48	Sensor/Temperature/Livingroom	22	2021-11-26 14:09:01.000
29	49	Sensor/Temperature/Kitchen	26	2021-11-26 14:09:12.000
30	50	Sensor/Temperature/Livingroom	26	2021-11-26 14:09:21.000

Query executed successfully. hph.database.windows.net (1... hphlogin (68) LOGGINGSYSTEM 00:00:00 131 rows

Ready Ln 1 Col 28 Ch 28 INS

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog