Proof of Concept for a Data Streaming Application in Azure IoT Hub

Johnny Meder
Santeri Österlund

2019 Laurea

Laurea University of Applied Sciences

Proof of Concept for a Data Streaming Application in Azure IoT Hub

Johnny Meder, Santeri Österlund
Business IT
Bachelor's Thesis
November, 2019 2019

Laurea University of Applied Sciences **Abstract**
Business IT
Degree Programme in Business Information Technology

Santeri Österlund

Proof of Concept for a Data Streaming Application in Azure IoT Hub

Year 20202019 Pages 52

The thesis examines the concept of applying an IoT solution for streaming data, i.e. how to continuously move data from a device directly to a cloud service for further analysis and storage.

More specifically, in this thesis a RuuviTag Environmental sensor is used to collect air temperature, humidity and pressure data. The sensor is connected to a Raspberry Pi 4 computer which functions as an edge device that sends the data directly to Azure IoT Hub on a continuous cycle. The data is then streamed within Azure Cloud Service to an SQL database. The goal of this thesis is to first prove the concept by creating a working solution and to create detailed instructions on how such a process was and can be done.

Generally, in IT, a solution that works is often the one that gets used even if it is not the most optimal one. This thesis focuses on trying to discover an easy-to-use solution, and for this reason two methods were combined to cut unnecessary material and produce a flexible solution.

The thesis was commissioned by the partner company Avanade as they desired a Proof of Concept solution and instructions for moving gathered sensor data into a cloud server.

Keywords: Azure, Cloud, IoT, Data, Proof of Concept.

Laurea-Ammattikorkeakoulu **Tiivistelmä**
Business IT
Tradenomi Tietojenkäsittely (AMK)

Santeri Österlund

Soveltuvuusselvitys reaaliaikaiselle datan lähetys- ja analysointisovellukselle Azure IoT Hub-palvelussa

Vuosi 20202019 Sivumäärä 52

Opinnäytetyössä selvitetään IoT ratkaisun lisääminen datankeruuprosessiin, eli kuinka yhtäjaksoisesti siirtää tietoa laitteesta suoraan pilvipalveluun analyysia tai varastointia varten.

Tässä opinnäytetyössä on käytetty RuuviTag ympäristösensoria ilmanlämpötilan, -kosteuden ja -paineen datan keräykseen. Sensori oli yhdistetty Raspberry Pi 4 tietokoneeseen, joka toimi edge-laitteena lähettämällä datan yhtäjaksoisella prosessilla eteenpäin Azure IoT Hub pilvipalveluun, jossa kerätty data ohjattiin datavirtauksella SQL-tietokantaan. Opinnäytetyön toimeksiantajana toimi IT konsultointiyritys Avanade, jonka pyynnöstä opinnäytetyön tarkoituksena on selvittää prosessin käytännönsoveltuvuus luomalla tähän toimiva ratkaisu ja tarkat ohjeet prosessin toistamiseen.

Yleisesti ottaen IT-alalla käytetään ratkaisuja, jotka toimivat, vaikkeivat ne olisi erityisen optimaalisia. Tässä opinnäytetyössä tavoitellaan helppokäyttöistä ratkaisua, josta syystä kaksi menettelytapaa on yhdistetty ja epäolennaista aineistoa on leikattu, luoden joustavan ratkaisun.

Asiasanat: Azure, Pilvi, IoT, Data, Soveltuvuusselvitys.

Table of Contents

1 Introduction 7
2 Terminology 8
3 Ethics 9
4 Feasibility study / proof of concept 9
5 Edge Devices and Field Gateways 10
5.1 Raspberry Pi 11
6 RuuviTag Environmental sensor 12
7 IoT 13
7.1 The cloud 14
7.2 Big Data 14
7.3 Microsoft IoT Hub 14
8 Data storage 15
8.1 Types of data 15
8.2 SQL database 16
8.3 Azure Cosmos DB 16
8.4 Azure Blob storage 17
8.5 Azure Data Lake Storage 17
8.6 Storage Options 18
9 Project Overview 19
9.1 Project Timetable 19
9.2 Project Plan 20
9.3 Project implementation 23
9.4 Project results 23
10 Step by Step instructions 24
10.1 Raspberry Pi installation and setup 24
10.2 Communication between RuuviTag and Raspberry Pi 25
10.3 Azure subscription 28
10.4 Creating a resource group 28
10.5 Creating the IoT Hub 29
10.6 Security of Azure IoT Hub 31
10.7 Initiating data transmission 31
10.8 Setting up a storage account 33
10.9 Setting up the SQL server 35
10.10 Creating an SQL database 38
10.11 Integrating a Stream Analytics job to the SQL database 42
10.12 Accessing database 43

11 Contemplation and suggestions 47
References 49
Figures 51
Tables 52

1 Introduction

This thesis was done in partnership with Avanade, who commissioned a proof of concept study for moving data to Azure cloud services. Avanade is a joint venture between Microsoft and Accenture, a leading provider of digital and cloud services, business solutions and design-led experiences, specializing in the digital workplace. Avanade builds and deploys solutions for their thousands of clients and client-partners by combining technology, business and industry expertise with a focus on technology from Microsoft (Avanade 2019).

This proof of concept thesis will focus on the practical elements involving Microsoft's Azure IoT Hub and effectively consists of two parts:

First part is completing the proof of concept by using a RuuviTag environmental sensor linked to a Raspberry Pi 4 computer to gather and send data to the IoT Hub and with Azure's resources, forward that data to an SQL database for further analysis. The focus of this thesis is also placed in the Azure cloud service environment. This project is done entirely under the limited "free subscription" option Microsoft offers for their Azure service, which means that this project has a strict timetable once the IoT Hub is set up and due to the nature of the free subscription, we are limited in our options for moving and storing data in the cloud.

The second part is compiling an extensive and easy-to-understand instructions to function as a blueprint for similar future projects as was agreed to for the partner company Avanade. The project will be thus split into ten plus one parts, with the final part not directly relating to the project's feasibility, but rather testing to see if it did work. Each part of the instruction will contain visualizations to help with understanding how to complete the project.

To make it easy to understand to as wide an audience as possible, the terminology, equipment and services used whilst completing this Proof Of Concept will be explained in detail, with the ultimate goal being that anyone with even limited experience of IT services in general, can complete the same tasks as were done for this project. Even parts where there the project requires using a programming language, should be completable for anyone.

2 Terminology

Python: A popular general-purpose programming language.

SQL: Structured Query Language, a standard language for databases.

Request Unit: A unit to calculate the needs of your system based on the need of Memory, CPU and IOPS usage percentage.

IOPS: Input/Output Operations Per Second

OS: Operating system.

Linux: Unix-based open source operating system

Debian: Linux based operating system, made by Free Software

Raspbian: Operating system based on Debian, made for the Raspberry Pi

IP address: Address given to each device in a network

SD card: Secure Digital card, a small form factor memory card

HDMI: High-Definition Multimedia Interface, audio / video interface

USB: Universal Serial Bus, connection standard between devices

Power BI: Analytics program / service by Microsoft

Database: A collection of data held in one file or linked files

Terminal: Text-based interface used for typing commands

SSH: Secure Shell, network protocol for operating devices remotely

Bluetooth: Wireless technology used for exchanging data between devices

JSON: JavaScript Object Notation, an open-standard file or data interchange format that uses human-readable text to transmit data objects

Unicode: Universal character encoding standard for computer systems

UTF-8: 8-bit Unicode Transformation Format, a variable width character encoding capable of encoding all 1,112,064 valid code points in Unicode using one to four 8-bit bytes. That is partially compatible with earlier systems and the most common method of encoding in use today

3 Ethics

We have been given sources of information and resources from Avanade, some of which is part of their internal communication that must be handled with care.

To assure we achieve as ethical an approach as possible, we will act with complete transparency with the liaison company and provide them with necessary contracts and ask for permission to use their information at any point in this thesis. All parties involved in this thesis will also sign a partnership document during the project.

Data security was a focus point that had few rudimentary solutions after we considered the type of data and the solutions already in place from the Azure side of things.

The RuuviTag sensor should not be considered for sensitive projects, because anyone with a device that can have a Bluetooth connection can intercept the data it collects and emits. The easiest solution to this specific problem is to choose a different sensor that does not simply transmit its data for everything in range.

Security between the Raspberry Pi 4 and Azure IoT Hub was achieved by simply using the unique key generated in the service when a resource is created. This means the data that was sent could only be interpreted by the receiver if they have the key.

Security is a pivotal feature and Azure offers a lot of different solutions for different needs, and it’s difficult to determine, outside the basic security measures that were taken for a small project such as this, what steps should be taken specifically for different projects. In short, we considered the data security options Azure offers, but since the data collected would be restricted to simple temperature, humidity, pressure and acceleration types of data, and because it would be collected from public and unconcealed locations, it wouldn’t be necessary to spend resources beyond the security measures Azure has as default. In effect, for this project, security meant keeping the authentication keys secure by keeping the only existing copies of them in the Azure service and the Raspberry Pi 4 device transmission coding, neither of which is accessible without their respective high security level passwords.

4 Feasibility study / proof of concept

The first result from a google search gives a definition for proof of concept from Wikipedia as follows: “Proof of concept is a realization of a certain method or idea in order to demonstrate its feasibility, or a demonstration in principle with the aim of verifying that some concept or theory has practical potential. A proof of concept is usually small and may or may not be complete.” Which is an apt, albeit very condensed definition for a proof of concept study.

Other sources such as TechTarget (Rouse 2018) offer similar definitions for proof of concept studies, that a proof of concept is an endeavour where the focus of the work is to determine

whether the concept is feasible or verify that it works as intended. The entrepreneur article by Singaram and Jain (2018) Define proof of concept as an exercise to test a design idea or assumption with the goal of demonstrating its functionality and to verify that the concept or theory can work in practice and be achievable in development. An article in My Accounting Course (2019) also offers a similar definition, though specifies it a little by stating that the point of a proof of concept study is to prove a concept in small scale for future application of it in large scale. Fisch et al. (2015) define a proof of concept study as a trial with its main criteria being to fulfil the goal set for the concept and success of a PoC study is dependent on how well it achieved those goals.

Drawing from these sources as well as the generally accepted meaning of a proof of concept study, it is, in short: a project that is conducted with certain goals and methods in mind for the purpose of proving said goals can be achieved with said methods.

For example, this thesis has the goal of transmitting data from a source to an IoT Hub and from there to a database for further analysis, with methods ranging from using a sensor for data collection, a Raspberry Pi 4 for an edge device and the Azure cloud service for the cloud resources. The success of this project will be measured by streamlining the methods and then presenting them in a way that is as easy to understand as possible to be replicable for as many people as possible for any similar project concerning the same premise.

5 Edge Devices and Field Gateways

An edge device, or a field gateway, is a device or a general-purpose software that has effectively the job of acting as a communications enabler, a local device (such as the sensor) control system and a hub for local data processing. (Microsoft EdX 2019).

The Raspberry Pi 4 computer acts as the field gateway for the RuuviTag sensor, as it processes the data that the sensor collects and determines what gets sent to the Azure IoT Hub. The Raspberry Pi also requests the permissions and fills the required primary and secondary key data for the messages sent for authentication purposes. A device or a program is necessary for this, unless the device for data collection can by itself connect directly to the cloud service.

5.1 Raspberry Pi

Figure 1: Raspberry Pi 4 Model B (Raspberry Pi 2019b).

The Raspberry Pi is a small 88x58x19.5mm-sized single-board computer that has the following specifications:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 1GB, 2GB or 4GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage

- 5V DC via USB-C connector (minimum 3A*)
- 5V DC via GPIO header (minimum 3A*)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)
- Operating temperature: 0 – 50 degrees C ambient

It can be used as a desktop computer for various projects and it's best used with the Raspbian operating system, as it is designed for the device. (Raspberry Pi 2019a).

In short, the Raspberry Pi 4 is, at the time of making this project, the latest in a line of small but powerful computers that can function exceptionally well as a substitute for desktop computers for projects such as this, because it offers performance equivalent to entry level x86 PC systems (Raspberry Pi 2019b).

6 RuuviTag Environmental sensor

Figure 2: RuuviTag Sensor (RuuviTag 2019).

RuuviTag is a small circular open source Bluetooth sensor that's waterproof and can collect temperature, humidity, pressure and motion data. The sensor is immediately activated once a small plastic strip, that's between the battery and the sensor's power source, is taken off and it has a very long-lasting battery life.

A sensor is a device that collects a specific type of data, such as temperature, pressure and humidity, from its surrounding environment and one that has an output, like a signal for Bluetooth connection, whereupon the data can be then received and processed by the user. A sensor can be connected to other devices that manage the communication with a network or work as the endpoint for the collected data.

According to TechTarget (Rouse 2016) a smart sensor is a device that would, complementary to its regular sensor functions, also have built-in computing resources that allow the device itself to process its input data before transmitting it to the next IoT architecture node.

7 IoT

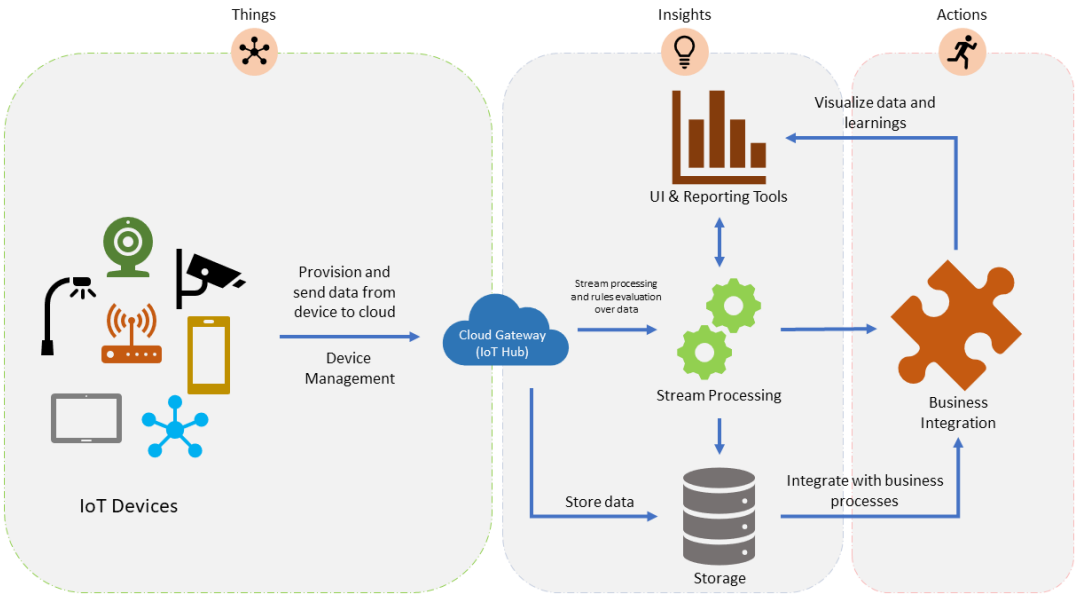

Figure 3: IoT (edX 2019)

Internet of things is, as its name implies, about any device, such as mobile phones, refrigerators, watches, even a sauna's thermometer, connected to the internet. Internet of things, or IoT, has gained a whole lot of traction, especially in recent years and according to Familiar (2015, 133) they estimate that by 2020 over 250 billion devices connected will be connected to the interned, of which mobile phones are only a fraction. B.K. Tripathy (2018, 44) argues, that IoT will be the next big step in the Internet evolution.

An important aspect of IoT is that each device connected to the internet will collect data via sensors and transmit it to the cloud. The amount of data is currently measured in zettabytes, a trillion gigabytes or to put a number on it, 10000000000000000000 bytes, and the amount is only going to grow as more devices are connected and terms such as big data become necessary. Even a single, relatively small IoT service could consist of thousands of devices, each sending numerous messages per second, for a total of hundreds of thousands to millions a day. And most important about this data collection from these devices is that it generates real time data feeds for real time analysis for any number of goals, including business decisions. (Familiar 2015, 133).

7.1 The cloud

The cloud or cloud computing refers to information technology devices or software offered as a flexible online solution. Hardware and its upkeep, software and their upkeep, updating and safety is handled by the service provider. The services and their use are left to the consumer and company. This can also be referred as Utility computing. This means that companies and consumers do not need to own expensive servers for all their computing needs. (Salo 2010, 20)

7.2 Big Data

The amount of data collected is too much for traditional methods of data collection, analysis and storage to handle, Big Data is an answer to all that. As Minelly, Chambers, et al. (2012, 29) put it, it took a combination of things in the computing world for big data to truly emerge. Information technology became cheaper, mobile computing emerged, social networking and of course, cloud computing was developed. From these came out the three V's of data: Volume, Velocity and Variety. There is an abundance of data (Volume) that is generated and moved at a mind-boggling rate (Velocity) and the data is generated by pretty much anything from thermometers to car engine monitors for all kinds of data (Variety).

Big Data is already changing the world (Marr 2015, 9) and the more data we generate, the more it can be analysed for better solutions, which can simply mean getting smarter. The most important take away from big data is not the actual amounts of data, but rather, the constantly developing ways that data can be harnessed for use, so much so that Marr (2015, 11) considers big data as an inaccurate term to describe the phenomenon, favouring "Smart Data" instead.

7.3 Microsoft IoT Hub

Microsoft's Azure IoT Hub service enables bi-directional communication between IoT device and Azure (Microsoft Azure 2019). This is the environment used to connect the sensor to the cloud.

Familiar (2015) explains in his book Microservices, IoT and Azure, that the service was released for general availability in 2015 and that it enables three main resources for IoT use; device-to-cloud and cloud-to-device hyper scale messaging, per-device security credentials and access control and device libraries.

The IoT Hub offers these device connections with varying numbers, from single digit devices able to be connected to the hub to thousands at a time. An important factor in cloud services is security, because sending millions of messages between your devices and the cloud, each message sent and received need to be authenticated, for obvious reasons such as preventing

hijacking. The pricing should always be kept in mind, because running the service can cause costs to creep up rather quickly.

The project done in this thesis will not delve into the maximum capacity of the IoT Hub, as the number of devices is just one and the messages-per-day is just shy of a thousand.

8 Data storage

Simply put, data needs to be stored for later use and choosing the best data storage options for an IoT solution is a must. Storing data allows for creating visualisations and having historical data also enables better analysis.

Azure cloud service offers multiple different storage options: Azure SQL database, Azure Cosmos database, Azure blob storage, Azure data lake storage, Azure files, Azure Queue and disc storage. There is an important distinction between types of data when considering storage options, as some storage types rely on the data being suitable for them.

8.1 Types of data

Azure Storage devised to take three primary types of data: Structured data, Semi-structured data and Unstructured data (Microsoft 2019b).

Structured data keeps to a certain composition, meaning the data can be stored in tables with rows and columns. In other words, because structured data depends on keys to designate how different rows relate to each other, it is also referred to as relational data. Since all the data is in the same format, structured data is straightforward because it can be easily entered, queried and analysed. The sensor data used in the project is structured data.

Semi-structured data isn't as straightforward as structured data, because it doesn't conform cleanly into tables, rows and columns and instead uses keys or tags to systemize the data into a hierarchy. Because of this, semi-structured data is also known as non-relational or NoSQL-data.

Unstructured data, as its name implies, does not have any designated structure to it and the lack of structure means that effectively any kind of data can be held as unstructured data from PDF documents to JSON files and even video content. Azure offers Blob storage for exactly this type of data and it's becoming increasingly popular because of how unrestricted by storage structures it is.

Further, databases are designed with different tiers of data in mind and Microsoft (2019b) defines these tiers as such:

Hot - Optimized for storing data that is accessed frequently.

Cool - Optimized for storing data that is infrequently accessed and stored for at least 30 days.

Archive - Optimized for storing data that is rarely accessed and stored for at least 180 days with flexible latency requirements (on the order of hours).

8.2 SQL database

SQL database is a relational database that uses Structured Query Language and it's a high performance and reliable type of storage but requires the collected data to be structured. SQL database is very simple to use, because everything needed to effectively fully use the database are the standard commands "Select", Drop", "Update", "Delete", "Create", and "Insert".

Microsoft offers Azure Database Migration Service and Microsoft Data Migration Assistant for moving one's own SQL databases to the Azure cloud service (Microsoft 2019b).

This type of database was selected for the project, because the collected sensor data was structured and because we were familiar with this type of storage and could modify queries when needed. It is also arguably the easiest to use type of data storage.

An SQL server costs between $991,78 to $1490,53 a month to upkeep on general purpose settings and upwards of $4000 with business critical settings, with storage costing at $0 at the minimum 32GB and $923,68 at the maximum of 8TB making it a very expensive choice for a database for small projects, but a relatively cost-effective choice for mid-range projects (Microsoft 2019c).

8.3 Azure Cosmos DB

Azure\s Cosmos DB is a multi-model database service that supports unstructured data and lets you build responsive and always on applications for constantly changing data. Cosmos DB also allows for a scalable throughput, which means that the amount of data that passes through it can be scaled to one's needs.

Azure Cosmos DB supports SQL, MongoDB, Cassandra, Tables and Gremlin, which effectively means that it can function with any type of data, depending which of the supported databases you start out with and only those types of databases.

Cosmos DB pricing is scalable, costing $0,008 an hour per 100 RU/s (Request Units per second) and $0,250 per GB of storage per month, meaning it woud be a very cheap option for a small scale project, the bare minimum cost for this database is only €23,61 a month. Cosmos DB can be scaled up to support millions of request units per second and thousands of terabytes of storage capacity, with the price scaling up accordingly (Microsoft 2019c).

8.4 Azure Blob storage

Azure blob storage is a highly scalable and can use unstructured data and as such, it is not restricted in the types of data it can store. Effectively, Blobs work with apps like files on a disk for reading and writing data and can manage massive amounts of data in thousands of simultaneous uploads. Blobs are a kind of jack of all trades when it comes to data storage, so it is a desirable form of storage for any project. The versatility of blob storage makes it a prime candidate for any project to use as their data storage method (Microsoft 2019b).

	Premium*	Hot*	Cool*	Archive*
First 50 terabyte (TB) / month	$0,15	$0,0184	$0,01	$0,00099
Next 450 TB / Month	$0,15	$0,0177	$0,01	$0,00099
Over 500 TB / Month	$0,15	$0,0170	$0,01	$0,00099

Table 1: Azure Blob Storage pricing (Microsoft 2019b)

The pricing on blob storage is dependent on the type as well as volume of the data stored. Overall, blob storage is a cheap method for storing data.

8.5 Azure Data Lake Storage

Azure Data Lake Storage is a scalable database that works with structured and unstructured data (Microsoft 2019b). Azure Data lakes costs $0,04 per GB of storage, $0,004 per 10000 read transactions and $0,05 per 10000 write transactions a month and allows for scaling upwards indefinitely (Microsoft 2019c). A simple project with the minimum settings can run for a month for as little as $0,09. The scalability and pricing make it an extremely cost-effective storage option. You can create a blob storage directly into a data lake and stream your data there.

8.6 Storage Options

	Structured Data	Semi-structured Data	Unstructured Data
SQL High to Mid cost	Suitable	Unsuitable	Unsuitable
Azure Cosmos Low to mid cost	Suitable	Suitable	Suitable
Azure Blob Very low cost	Suitable	Suitable	Suitable
Azure Data Lake Low cost	Suitable	Suitable	Suitable

Table 2: Data storage type, suitability and cost comparison

Prices and suitability are a major factor when considering an IoT solution. The best option for a large-scale would almost certainly be the Azure Blob storage option, however, for this project the data was stored in an SQL table, because it was readily available in the free subscription and because it is a familiar type of storage for beginners.

9 Project Overview

The main purpose of this project is to prove a working concept by connecting a sensor, in this case the RuuviTag environmental sensor, via Microsoft Azure IoT Hub service to an available cloud service of choice. To accomplish this task, we will be utilizing a Raspberry Pi computer to process the sensor's captured data to the cloud provided by our liaison company Avanade.

9.1 Project Timetable

Project planning: 1.9.-30.9.

Background research and timetable planning, which includes completing edX courses related to IoT.

Project implementation: 21.10.-1.11.

23.10. We received the Raspberry Pi 4 and RuuviTag sensor.

25.10. We received corporate accounts for Azure, but due to some unfortunate complications, we could still only use the free subscription version of the service, limiting our time with the environment to just one month.

28.10. First communication achieved between Raspberry Pi, RuuviTag and Azure IoT Hub.

29.10. Communication is automated to happen once per minute. Attempting to create an SQL table in Azure, but instructions are unclear on this part.

30.10. SQL server is created, and the SQL database is formed.

31.10. Attempting to directly get the data to transmit from the device to a database.

1.11. Azure Stream Analytics is selected as the method for data transfer.

4.11. Unable to get the data queries to function, everything seems to work but data tables remain empty.

5.11. Managed to integrate a Stream Analytics job to the SQL database. Data now flows from the IoT Hub to the Database without issues.

Project reporting: 9.11.-

9.11. Project has been completed, reporting begins.

9.2 Project Plan

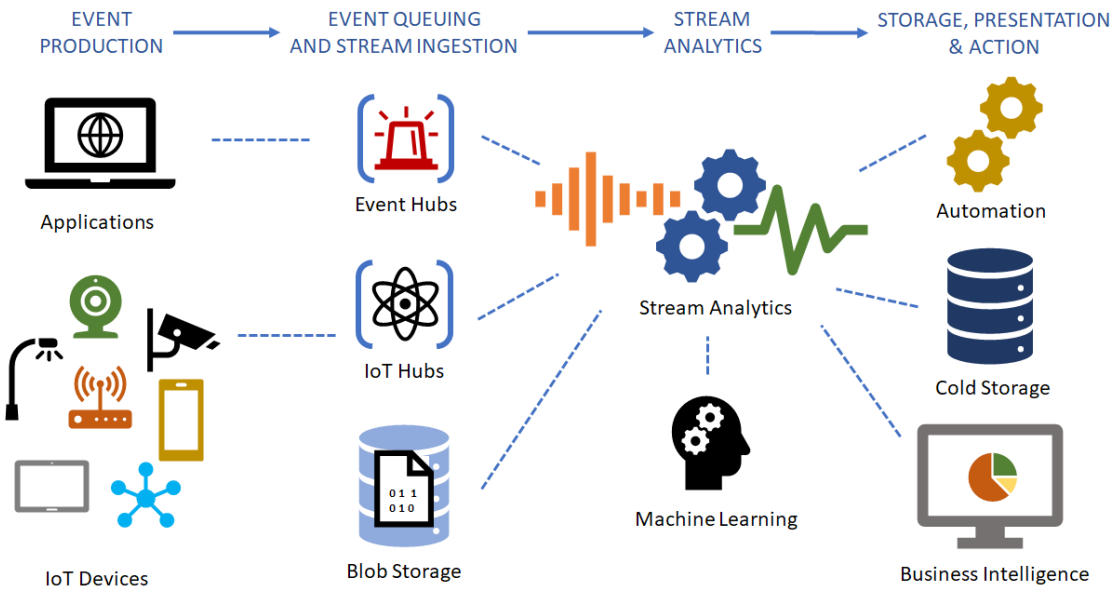

Figure 4: IoT in use (Microsoft EdX 2019)

The general idea for this project can be seen from figure 4, starting with event production and going through the phases in the same fashion from event queuing to Stream Analytics and finally storage and presentation.

IoT devices have already been used as pollution control devices (Pal 2017, 42) and this project could essentially be used in a very similar way, though this is more focused on the Azure IoT Hub end of things. As such, it is partially irrelevant for proof of concept what specific devices were used, for any device with a data output and a computer would suffice.

The first step for the project was to set up the RuuviTag environmental sensor and the Raspberry Pi 4 as a local computer for data collection. Once the sensor is ready, and the Raspberry Pi has the necessary programs to run the data collection and obtain the required message transmission permits, we can begin to transmit data through the IoT Hub to a database.

A large portion of the project is centred around the functionality of the Microsoft Azure cloud service, where we use the following services:

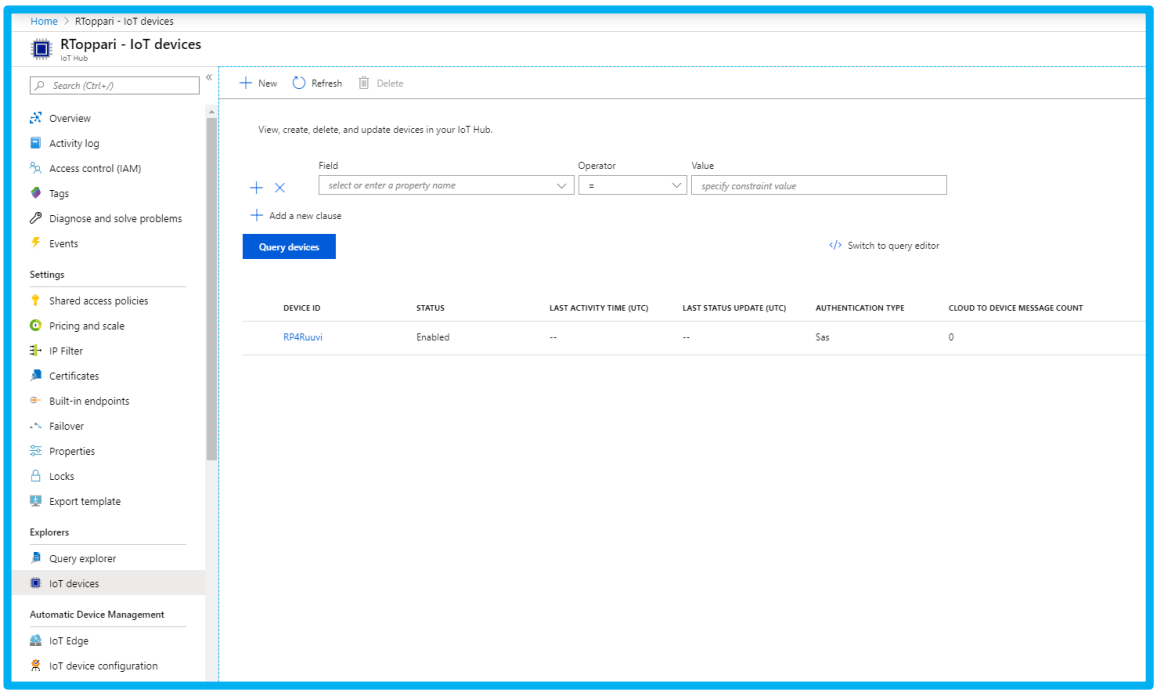

Figure 5: IoT Hub (Microsoft)

IoT Hub: This is the service that allows us to connect the sensor to the cloud, effectively what it does is function as an entry point for moving the data inside the Azure cloud service. We can observe all the devices we've linked to the cloud with this service, which for the purpose of this project will only be a single sensor named "RP4Ruuvi".

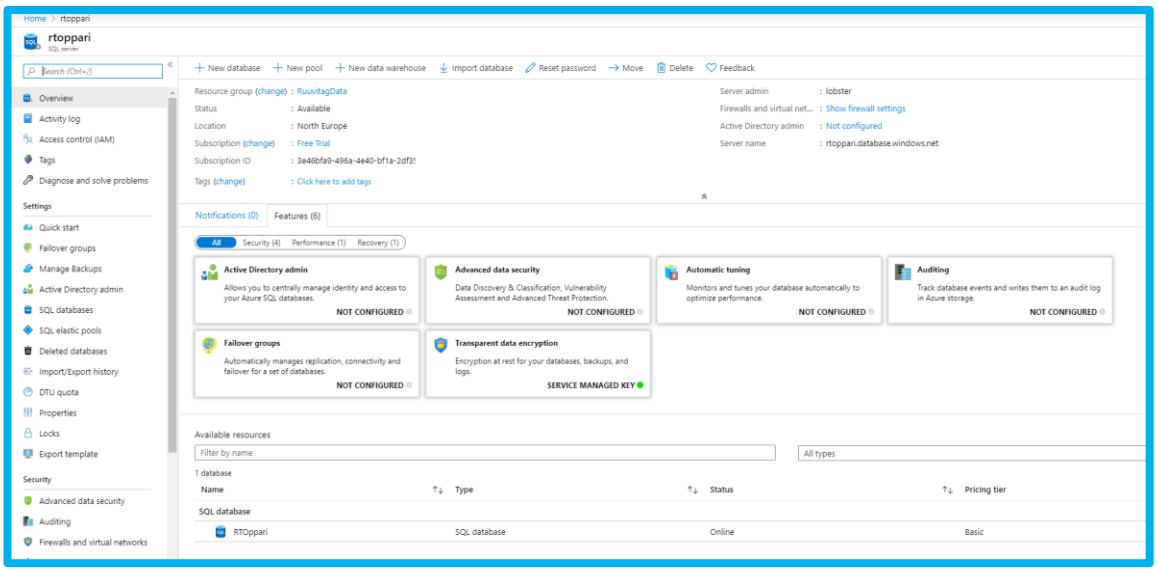

Figure 6; SQL Server (Microsoft)

SQL Server: In order to create an SQL database, we also need to host an SQL server. We can view the SQL databases hosted in the server at the lower part of the overview tab. Most

importantly, this service also allows configuring security settings, such as the admin login and handing out permissions to specific IP addresses.

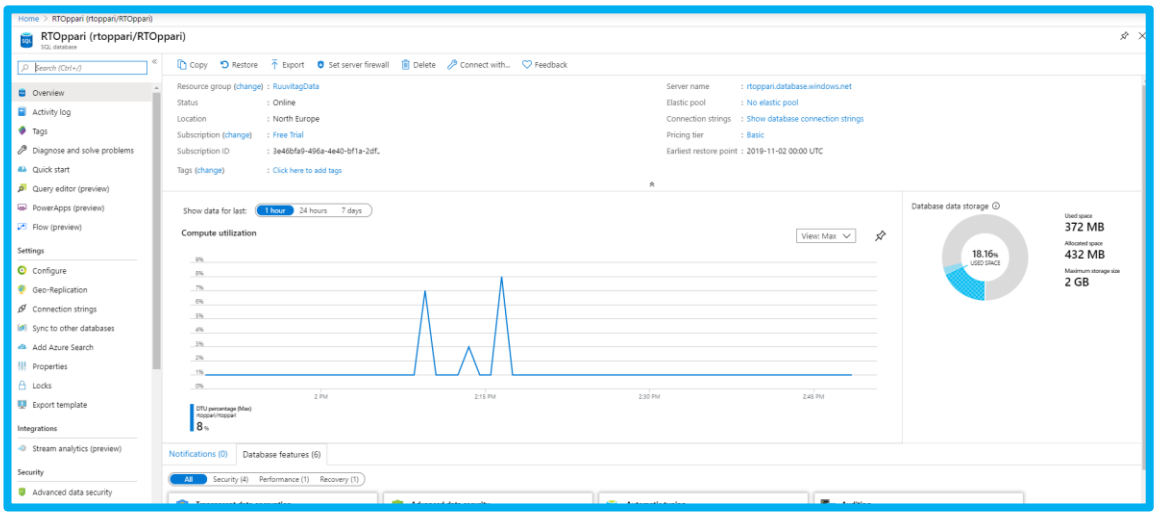

Figure 7: SQL Database (Microsoft)

SQL Database: For data storage, we chose to use an SQL data format, which required an SQL database. The Azure service allows us to see data usage, pricing tier among other things, but most importantly, there is a quick option for integrating and creating a Stream Analytics job for this specific database.

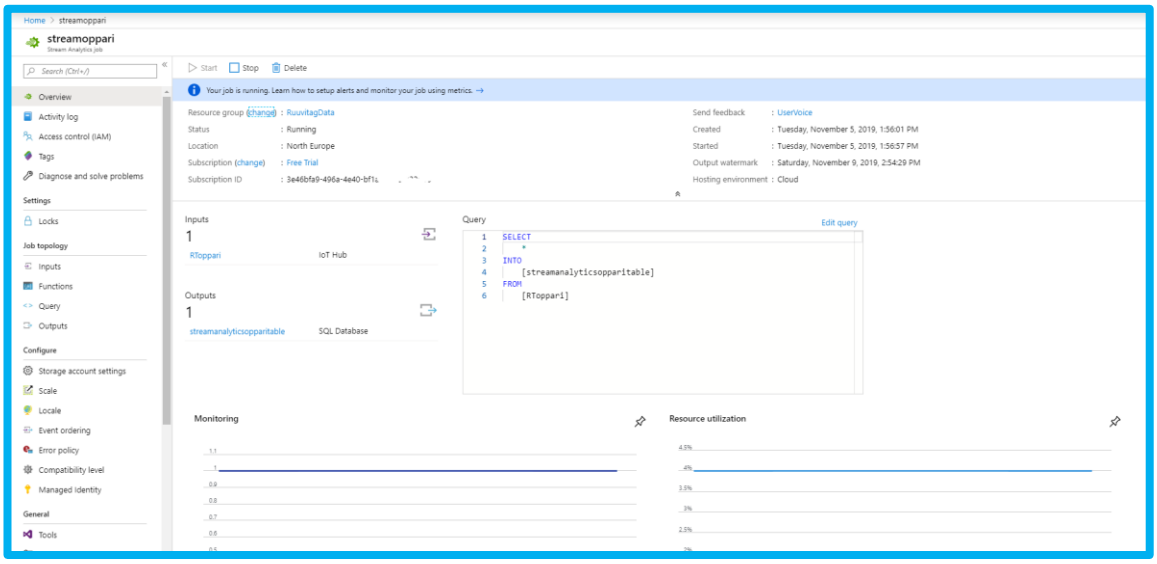

Figure 8: Azure Stream Analytics Job (Microsoft)

Stream Analytics Job: this service allows connecting the IoT Hub with a database. For this project, the SQL query used is very simple, because we wanted to gather all the data our device sent to the IoT Hub. Because the service runs an SQL query, it is possible to create a

more specific query which leaves out unwanted collected data or even have the data be ordered in any specific way.

Power BI: Finally, for the visualisation and to test that the data had for sure been transmitted from the Hub to the database, we used Power BI for retrieving and processing it. Power BI was chosen because it can directly retrieve the data from Azure and is very apt for processing large amounts of data.

Due to having to use the free subscription, we were limited in the amount of programs we had access to.

9.3 Project implementation

The project was completed with resources provided by Avanade and within the free tier subscription limitations.

9.4 Project results

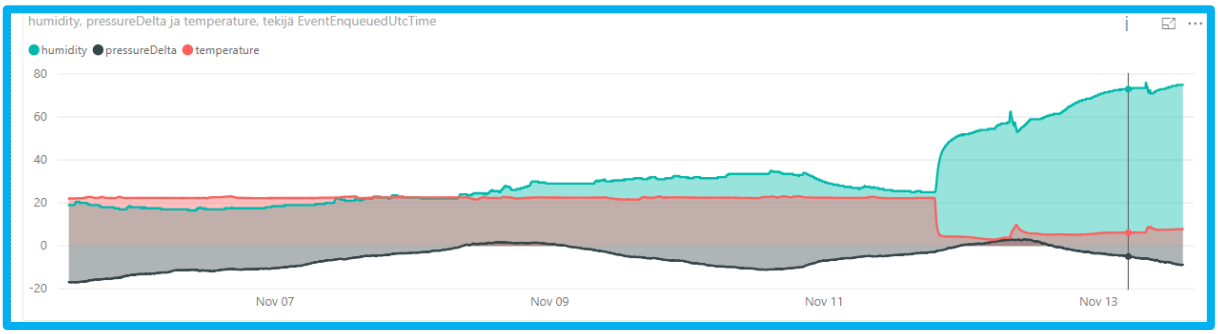

Figure 9: Collected data

Once the data had been moved from the RuuviTag device through the Iot Hub to the SQL database, we retrieved it with Power BI, a program which allows the creation of visualizations for large amounts of data, which in this case was over 12000 lines of data.

The sensor was placed indoors to gather data for 6 days, after which it was taken outdoors.

10 Step by Step instructions

The goal of this thesis is to provide a concise and easy-to-understand instructions on how to connect a sensor to the IoT Hub, essentially to move data from a physical component to a cloud. This section is dedicated to the instructions for that purpose. We used Figure 3 as a rough basis for the project plan, starting with the device, moving to IoT Hub, then to Stream Analytics and finally data storage.

10.1 Raspberry Pi installation and setup

While this project could be done with any model of Raspberry Pi, we have access to the newest version 4. We chose and recommend using Raspbian as our operating system, as it is very user friendly and has all the features we need and is optimized for the Raspberry Pi. This can be downloaded from the official website of Raspberry Pi as a zip file.

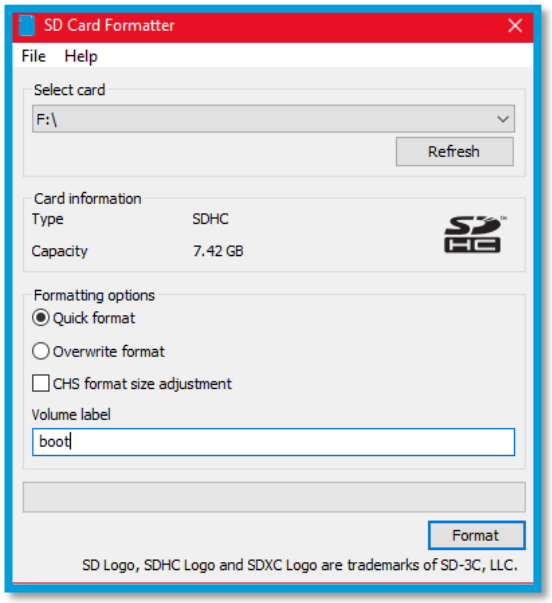

Figure 10: SD Card formatter by SD Card Association

The Pi uses a SD card as its hard drive, so a SD card reader is necessary to copy over the operating system from a different computer. We also needed to format the SD card in FAT. We used the SD Card Formatter program made by the SD Association for this as it was recommended on the official Raspberry Pi website. Format your card using the SD Card Formatter or any similar program. Use the default settings and name the volume label as boot.

Raspberry Pi has a great tool for creating an image for the disc called NOOBS – New Out Of the Box Software that we also used to install the operating system onto the Pi. You need to extract and copy over the Raspbian files from the zip file onto the SD card.

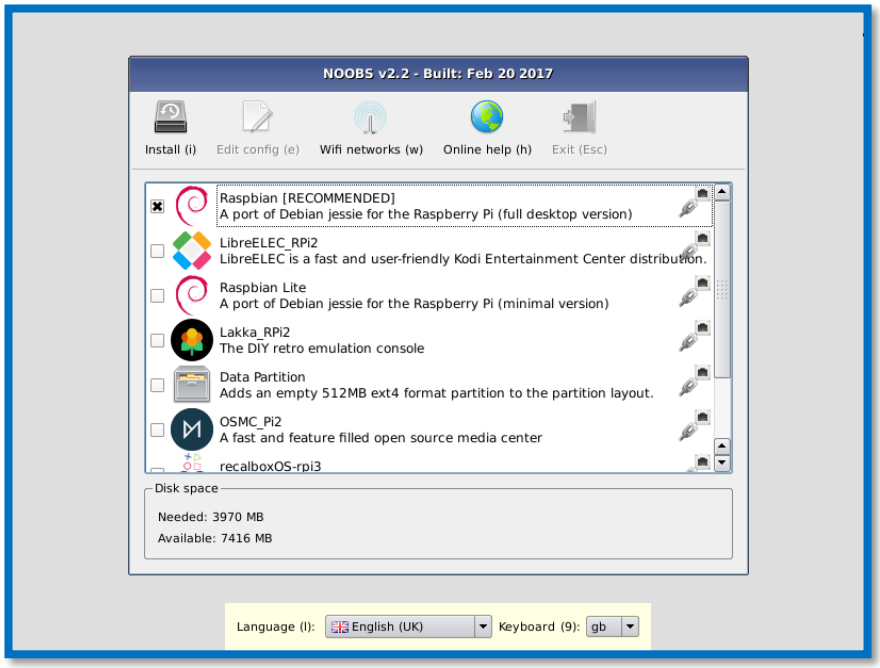

Figure 11: NOOBS installer

After the card had the Raspbian files copied over, insert the SD card into the SD card slot and connect a monitor over HDMI and a USB keyboard and mouse. Powering up the Pi with the card will open a setup screen. Select Raspbian and the correct language, time zone and keyboard layout settings and continue. The installer will install your operating system onto the Raspberry Pi and the SD card. You will be asked to create a username and password, make sure you remember these.

10.2 Communication between RuuviTag and Raspberry Pi

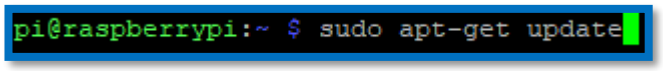

Figure 12: Updating the Raspberry Pie

Starting off you should open the terminal and update the Raspberry Pi with the command: ‘sudo apt-get update’

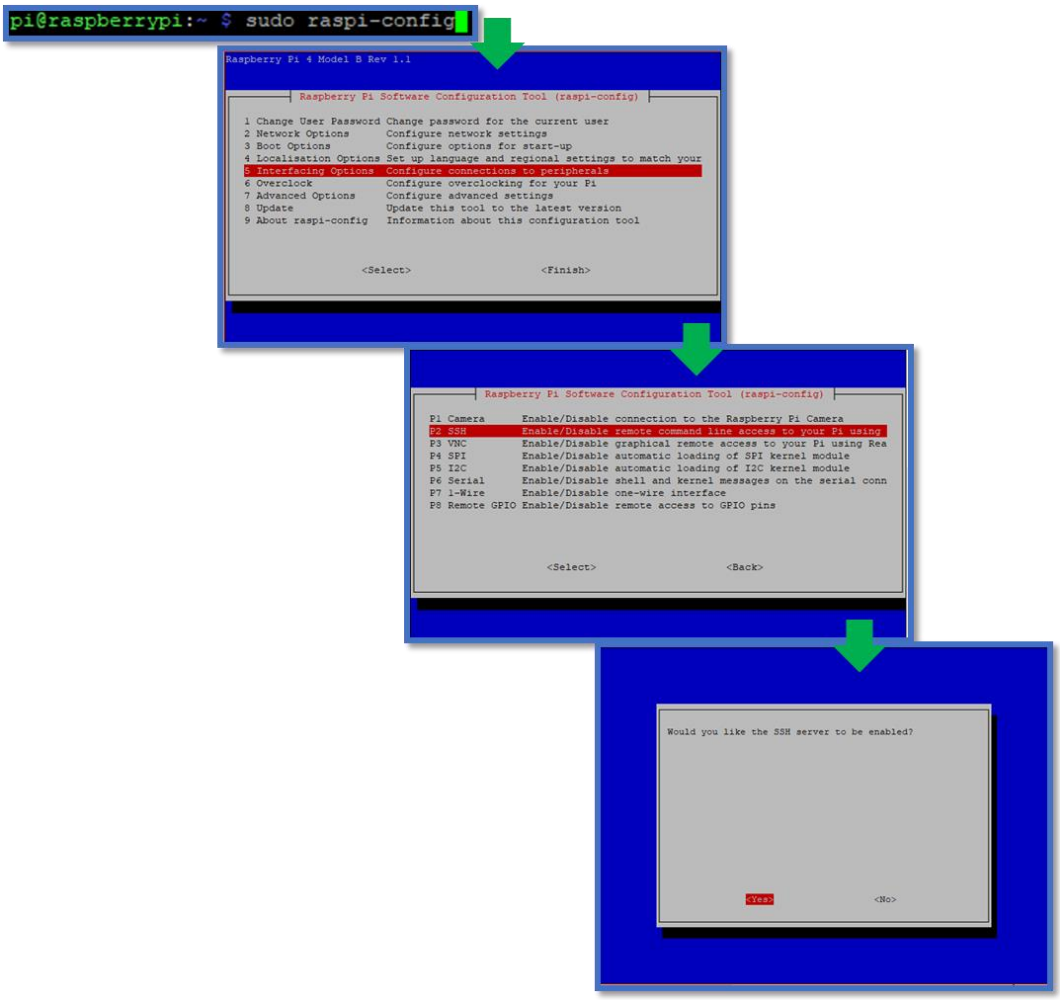

Figure 13: Enabling SSH

We also recommend enabling SSH for later access to the Raspberry Pi with a remote desktop. You can do this by writing ‘sudo raspi-config’ in the terminal window.

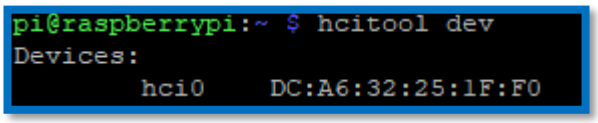

Figure 14: Testing the Bluetooth

Next up you need to start up the RuuviTag and make sure your Raspberry Pi can see it by writing ‘hcitool dev’ onto the terminal window. It should show up as hci0 or hci1

At this point updating the Python version from 3 to either 3.6 or 3.7 can be done and is recommended but not necessary. We use python 3.7 in this example.

```
pi@raspberrypi:~ $ sudo apt-get install bluez-hcidump
```

Figure 15: Installing bluez-hcidump

In order to read and write data from the device you need to install bluez-hcidump. Bluez is a Linux Bluetooth system and it enables the Raspberry Pi to communicate with older and low energy Bluetooth devices and is already installed with Raspbian. Bluez-hcidump allows us to print data from Bluetooth devices to the terminal. Installing it with the terminal window by writing ‘sudo apt-get install bluez-hcidump’.

```
pi@raspberrypi:~ $ sudo pip3 install --upgrade setuptools
```

Figure 16: Upgrading setuptools

There is a ready-made library for a command line utility using python that enables us to communicate and manage the data coming from the RuuviTag. In order to install this, you need to first update your setuptools with the command ‘sudo pip3 install --upgrade setuptools’.

```
pi@raspberrypi:~ $ pip3 install --user ruuvitag-sensor
```

Figure 17: Installing the RuuviTag sensor library

Installing the actual library is done with the command ‘pip3 install --user RuuviTag sensor’

```
pi@raspberrypi:~ $ alias ruuvitag='python3 ~/.local/lib/python3.7/site-packages/
ruuvitag_sensor'
You have new mail in /var/mail/pi
pi@raspberrypi:~ $ ruuvitag -f
Finding RuuviTags. Stop with Ctrl+C.
Start receiving broadcasts (device hci0)
C4:B7:32:86:CC:1C
{'data_format': 3, 'humidity': 67.5, 'temperature': 8.31, 'pressure': 1014.73, '
acceleration': 1041.8114032779638, 'acceleration_x': -13, 'acceleration_y': -39,
 'acceleration_z': 1041, 'battery': 3073}
```

Figure 18: Creating an alias and testing

In order to make accessing the utility easier, create an alias with the following command ‘alias RuuviTag='python3 ~/.local/lib/python3.7/site-packages/RuuviTag_sensor'’. Note what version of python you are using and update it to the command if necessary. Then you can test if you are receiving the right data with the command ‘RuuviTag -f’. Note down the MAC address of your RuuviTag too, as you will need it in the next step.

10.3 Azure subscription

Anyone can complete this project with the free subscription, however, one must consider that it enables the free products for one month and you have generally restricted access to the service, i.e. you cannot use and create some database formats.

What also needs to be kept in mind, is that when selecting a subscription, it requests your banking information, which means that the subscription is permanently tied to that account and that you can't renew a free subscription again with it.

10.4 Creating a resource group

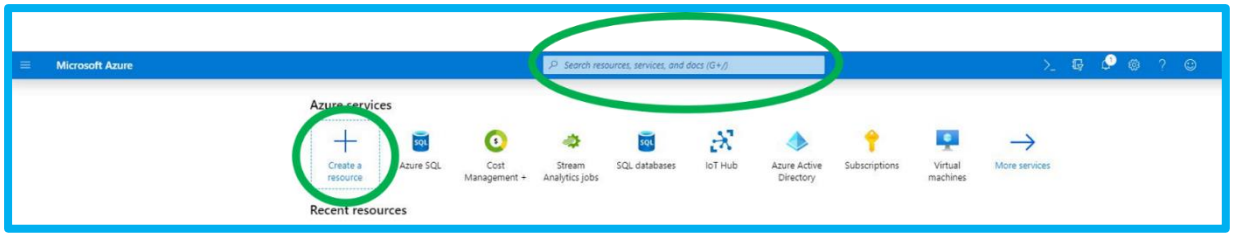

Figure 19: Azure search (Microsoft Azure)

You can see from figure 22 where to add resources by either directly clicking the "create a resource" and looking through the list, or by using the search bar for "resource group".

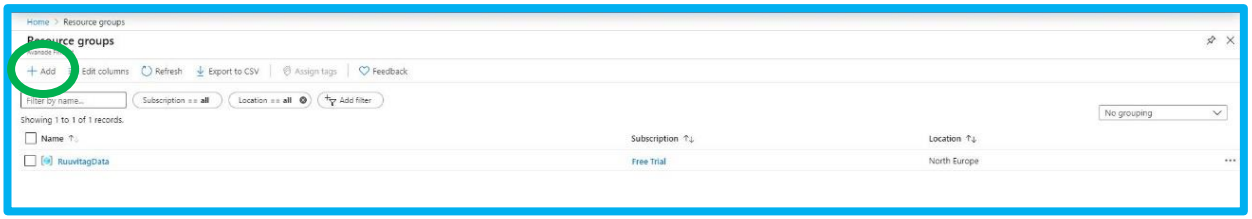

Figure 20: Adding a new resource group (Microsoft Azure)

Add a resource group, all the other Azure resources will be in this resource group.

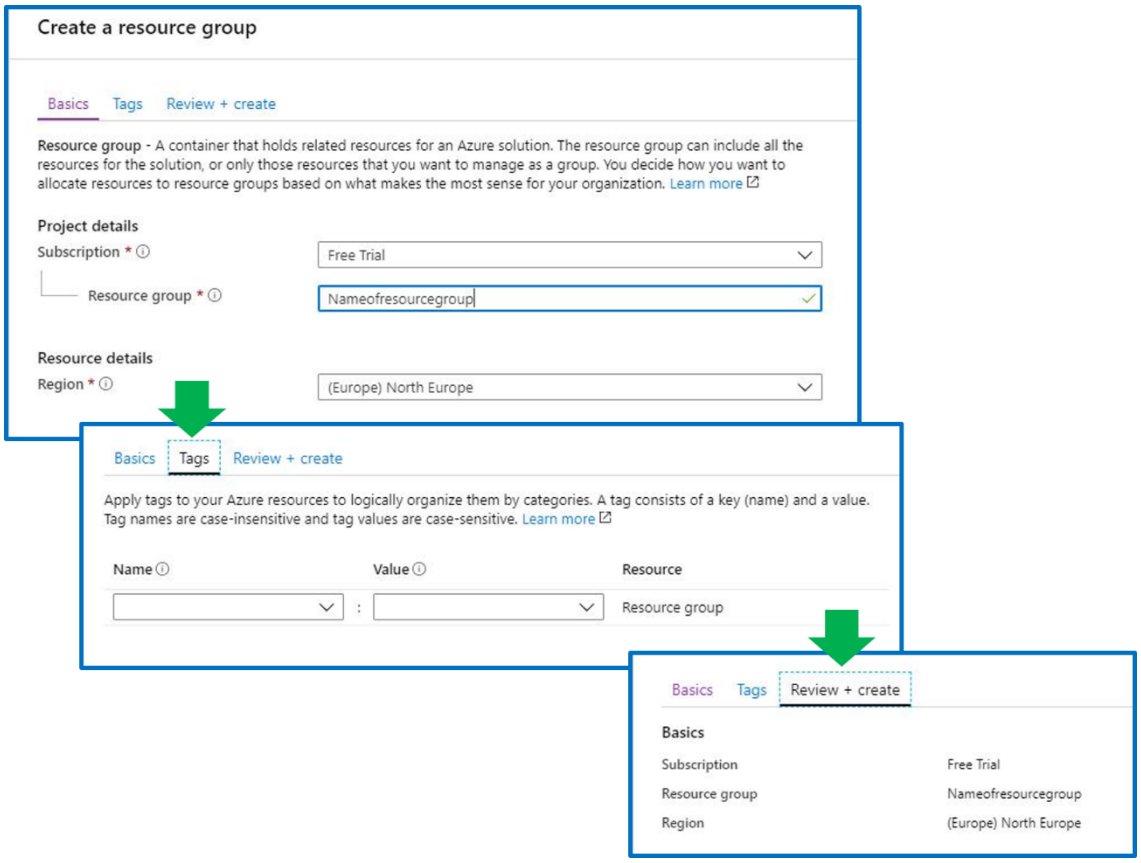

Figure 21: creating a resource group (Microsoft Azure)

On the basics menu, the subscription should automatically be filled with whatever subscription you are using, if it's empty, select the subscription you're using and name your resource group and select a location that best suits you.

There is no need for tags in this project, so you can move on to review + create and finalize the creation.

10.5 Creating the IoT Hub

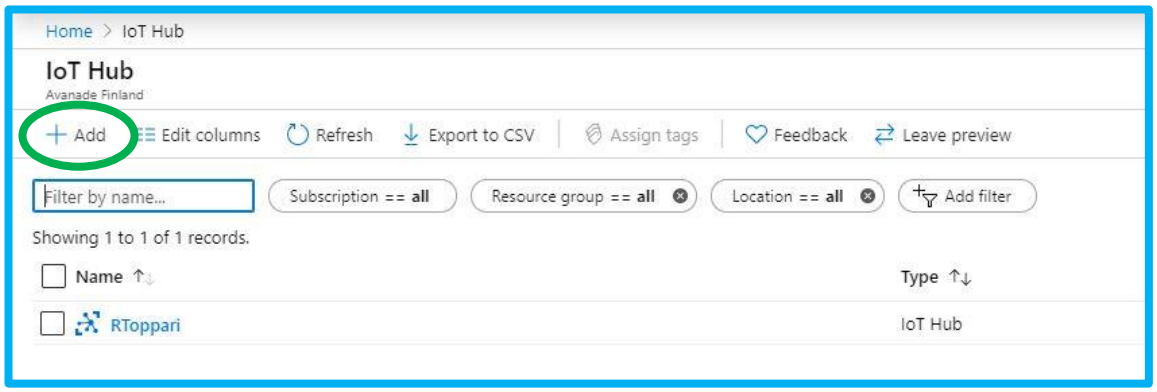

Figure 22: Adding a new Hub (Microsoft Azure)

Start by clicking the “add” button as seen in figure 25, this will open the IoT Hub creation menu.

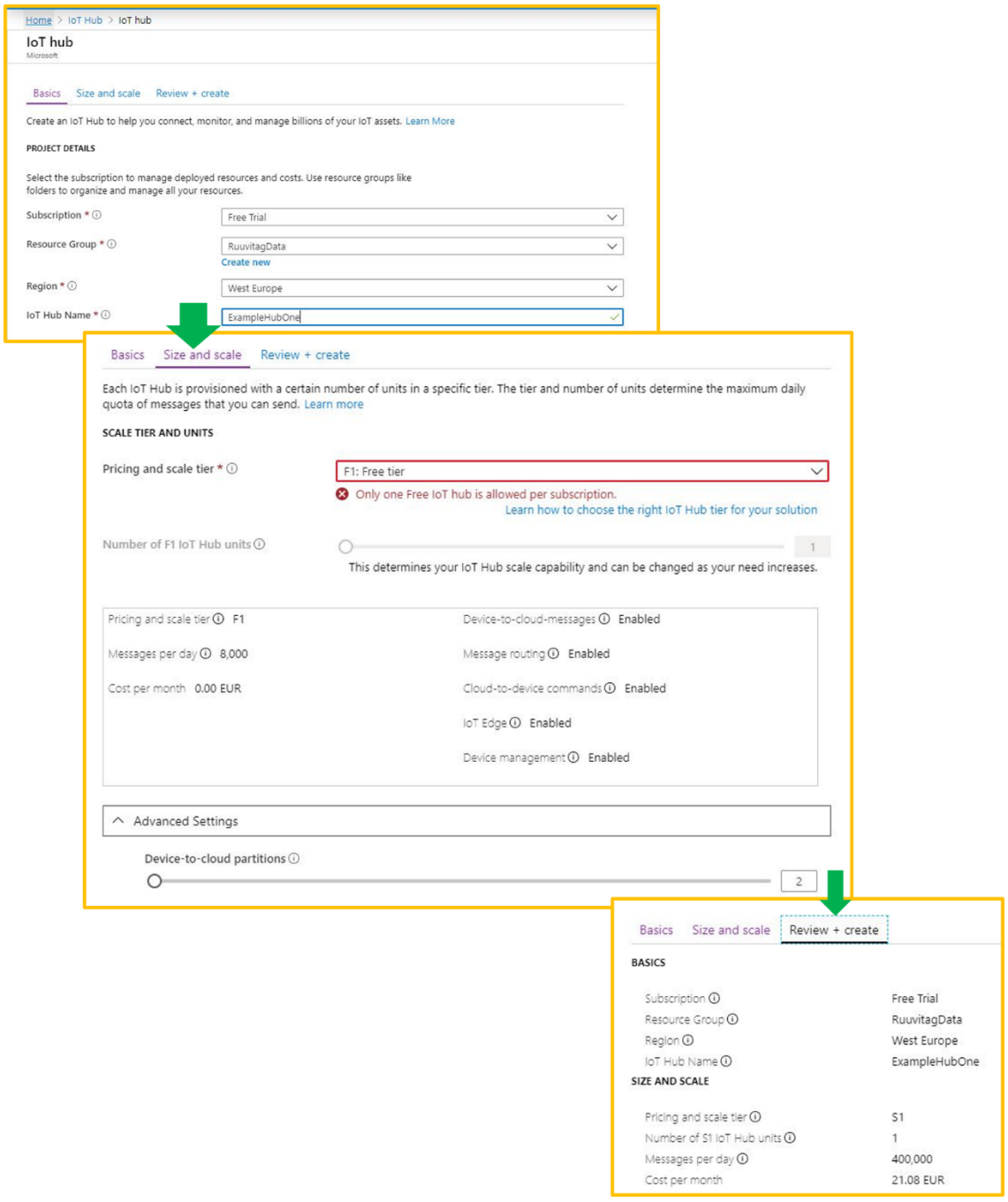

Figure 23: IoT Hub creation (Microsoft Azure)

Creating the IoT Hub requires filling out this form, the subscription should be automatically filled with whatever subscription you’re using. Select the resource group you created earlier in these instructions for that, select a region that best suits your location and then name your IoT Hub with a distinct name to avoid mixing it up with other resources you’ll be creating.

Selecting the size and scale is easy if you're using the free trial, everything is filled automatically. If you're using another subscription, you can then select the best options for your subscription and budget. The device-to-cloud partitions setting effectively means how many devices you can connect to your IoT Hub.

Finally, you can review and on the bottom part of the page click the "create" button to create your new IoT Hub.

10.6 Security of Azure IoT Hub

Creating an IoT hub generates a unique key that is long and complex enough that it cannot be cracked or brute forced trough. This along with the data security provided by the Azure services means that your data is secure and inaccessible to any attackers.

Security features can be selected in Azure depending on your specific needs and security options are usually offered at most instances when creating a data transferring job and each option has more information on its features.

10.7 Initiating data transmission

```
import DeviceClient
import datetime, requests
from ruuvitag_sensor.ruuvi import RuuviTagSensor
import json

# START: Azure IoT Hub settings
KEY = "DEVICEPRIMARYKEY";
HUB = "AZUREIOTHUBNAME";
DEVICE_NAME = "DEVICENAME";
# END: Azure IoT Hub settings

macs = ['RUUVITAG:MAC:ADDRESS:HERE']
timeout_in_sec = 5
datas = RuuviTagSensor.get_data_for_sensors(macs, timeout_in_sec)

device = DeviceClient.DeviceClient(HUB, DEVICE_NAME, KEY)
device.create_sas(600)

weatherdata = datas['RUUVITAG:MAC:ADDRESS:HERE']
encode_weatherdata = json.dumps(weatherdata, indent=1).encode('utf-8')

print(encode_weatherdata)

# Device to Cloud
print(device.send(encode_weatherdata))
```

Figure 24: Code to send data to the IoT Hub (Roine 2019)

In order to set up the Raspberry Pi to send the data to your IoT Hub, we need to create a Python script that pulls the data from the RuuviTag, encodes it from JSON to UTF-8 and send it to your IoT Hub. Name this script something appropriate, in this example we have named it datapull2.py. You get the necessary Azure IoT Hub settings from your Azure IoT Hub. You can copy this piece of code by double-clicking it.

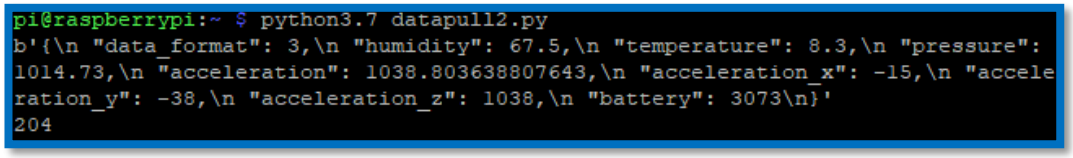

Figure 25: Testing the sending of data to IoT Hub

To test that the code works, you can run a command in the terminal 'python3.7 datapull2.py'.

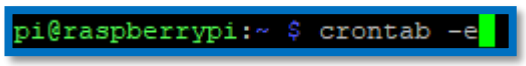

Figure 26: Making a crontab job

In order to automate this, we will use crontab. Crontab is a file that schedules entries to be run at specific times. In order to create a crontab job using the terminal, use the command 'crontab -e'.

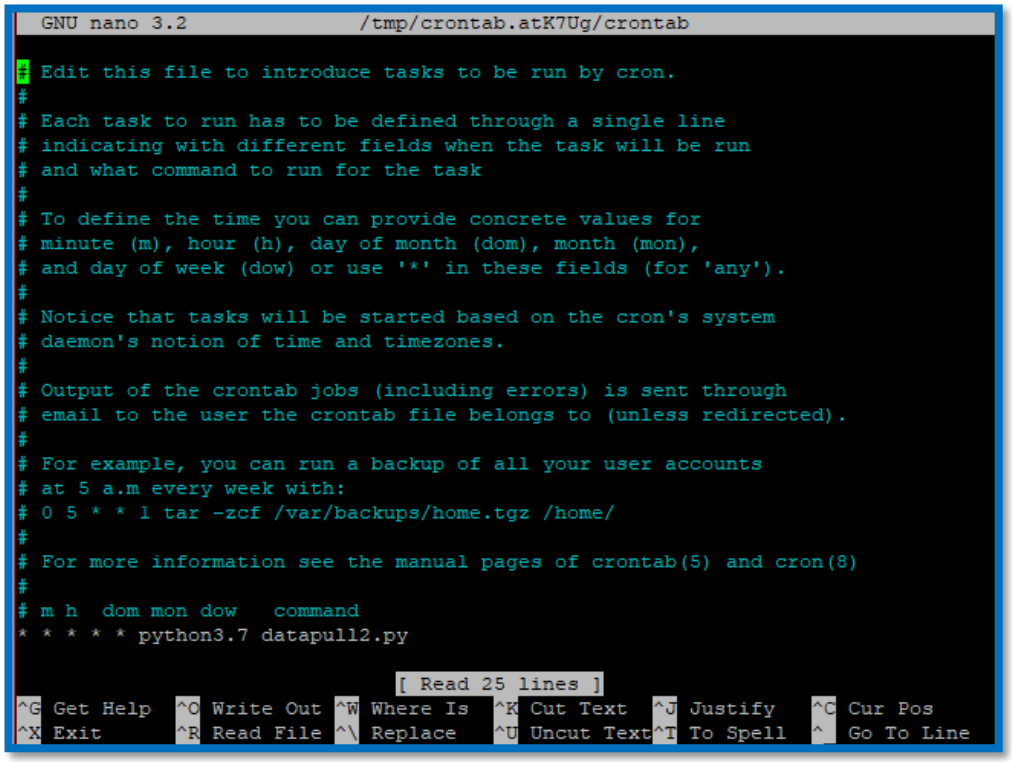

Figure 27: Editing the crontab job

This opens a text editor in the terminal in order to write your crontab job. In order to run our Python script every minute, we want the write the following code '* * * * * python3.7 data-pull2.py'. After you have written your code save it by using the button combination CTRL+O and then enter. Close the prompt by hitting CTRL+X.

10.8 Setting up a storage account

At this point in time, it is also possible to deviate from these instructions, and create a storage resource other than an SQL server and database in Azure. Simply click the "Create a resource" in your Azure home page and search for storage accounts.

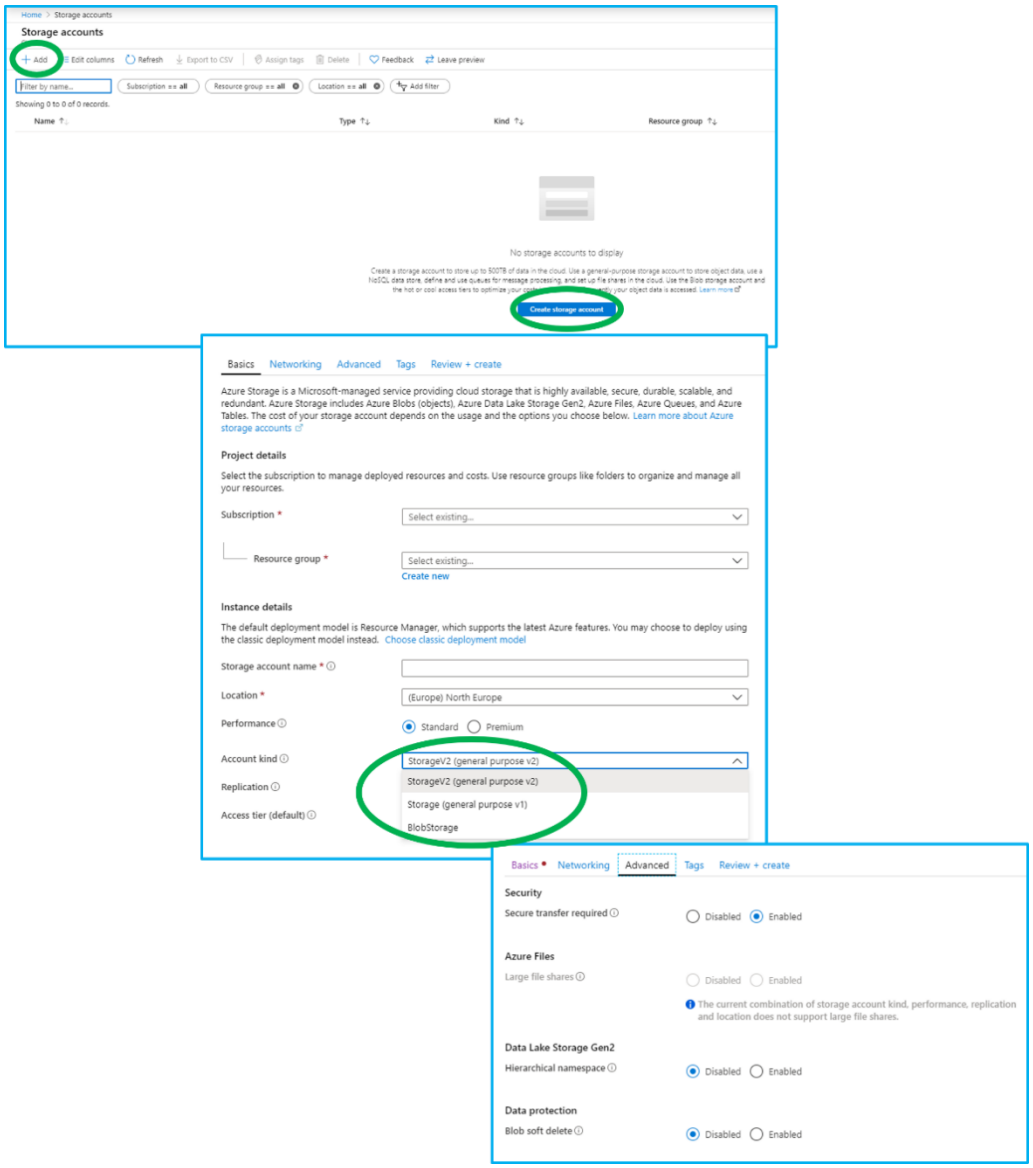

Figure 28: Creating a storage account (Microsoft Azure)

Simply click “add” or “create storage account” and select the desired type of storage unit for your project. You can either leave everything else on default or select the options you want.

10.9 Setting up the SQL server

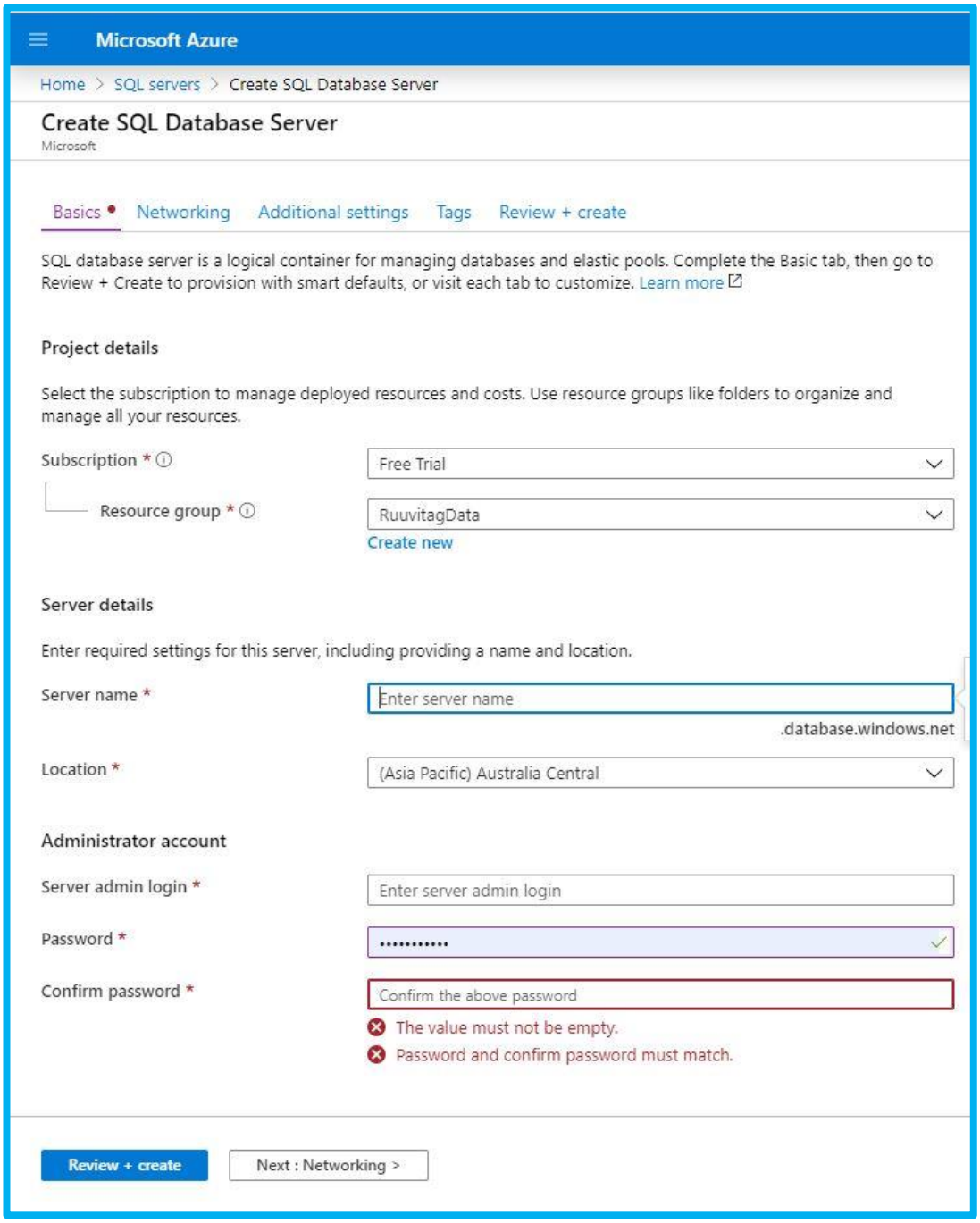

Figure 29: SQL server basics (Microsoft Azure)

Select the subscription you're using and then the resource group you created earlier.

For server details, name your server something distinct, so it won't get accidentally mixed up with other resources you create for this project. Select a location that best suits you.

Creating the administrator account is extremely important for this project, because it's what you'll need to prove your credentials when you try to access the server and any databases on it. Make sure you don't lose these.

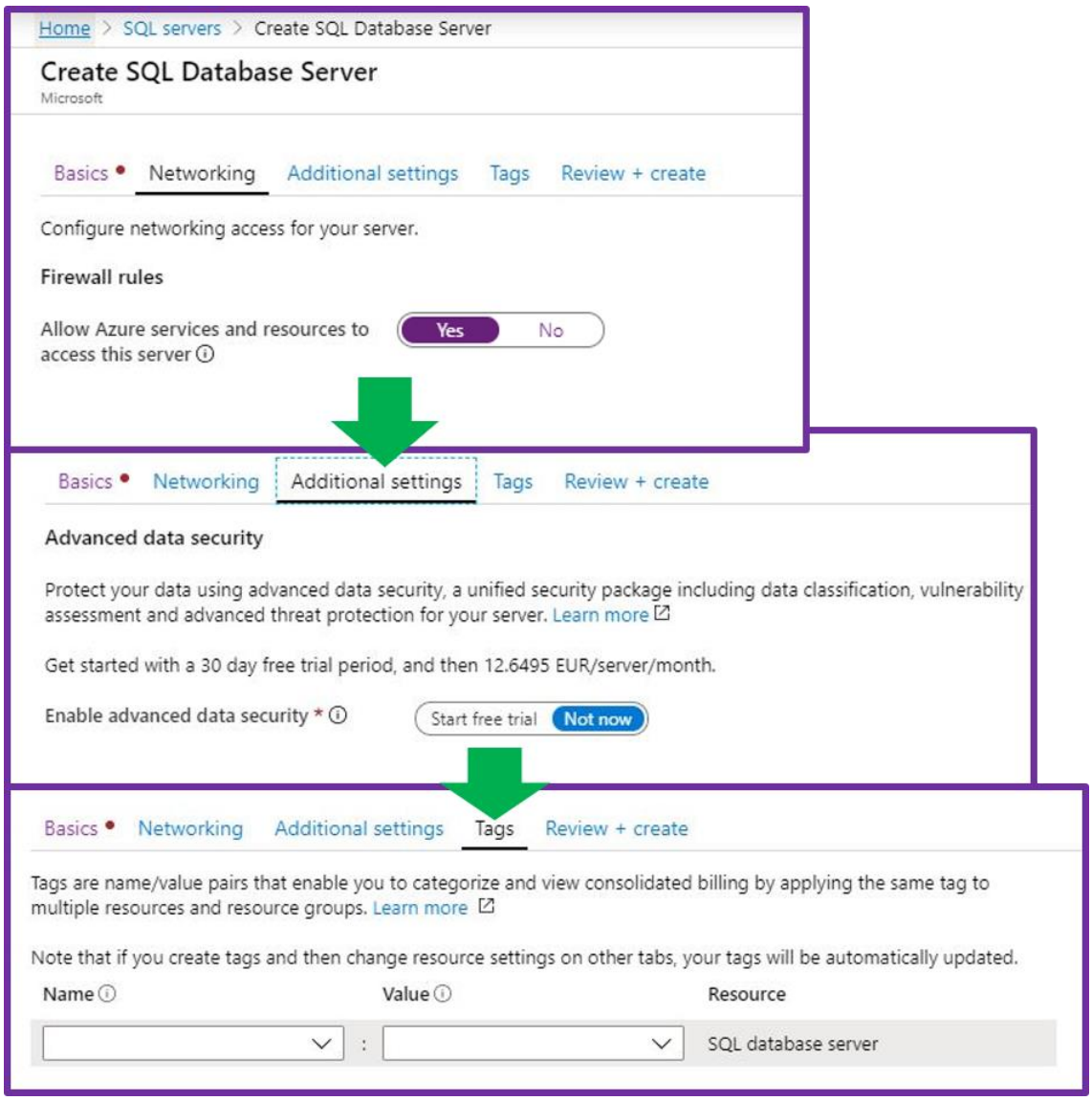

Figure 30: SQL server additional settings (Microsoft Azure)

For networking, allow Azure services and resources to access the server, the other resources need access for this project to function.

Enabling advanced data security isn't important for a project like the one done for this thesis, so it isn't necessary to use it. For longer lasting projects with more sensitive information, security is never a bad idea.

Since we're doing a very straightforward project, adding tags isn't that important. For larger projects, adding tags should be considered.

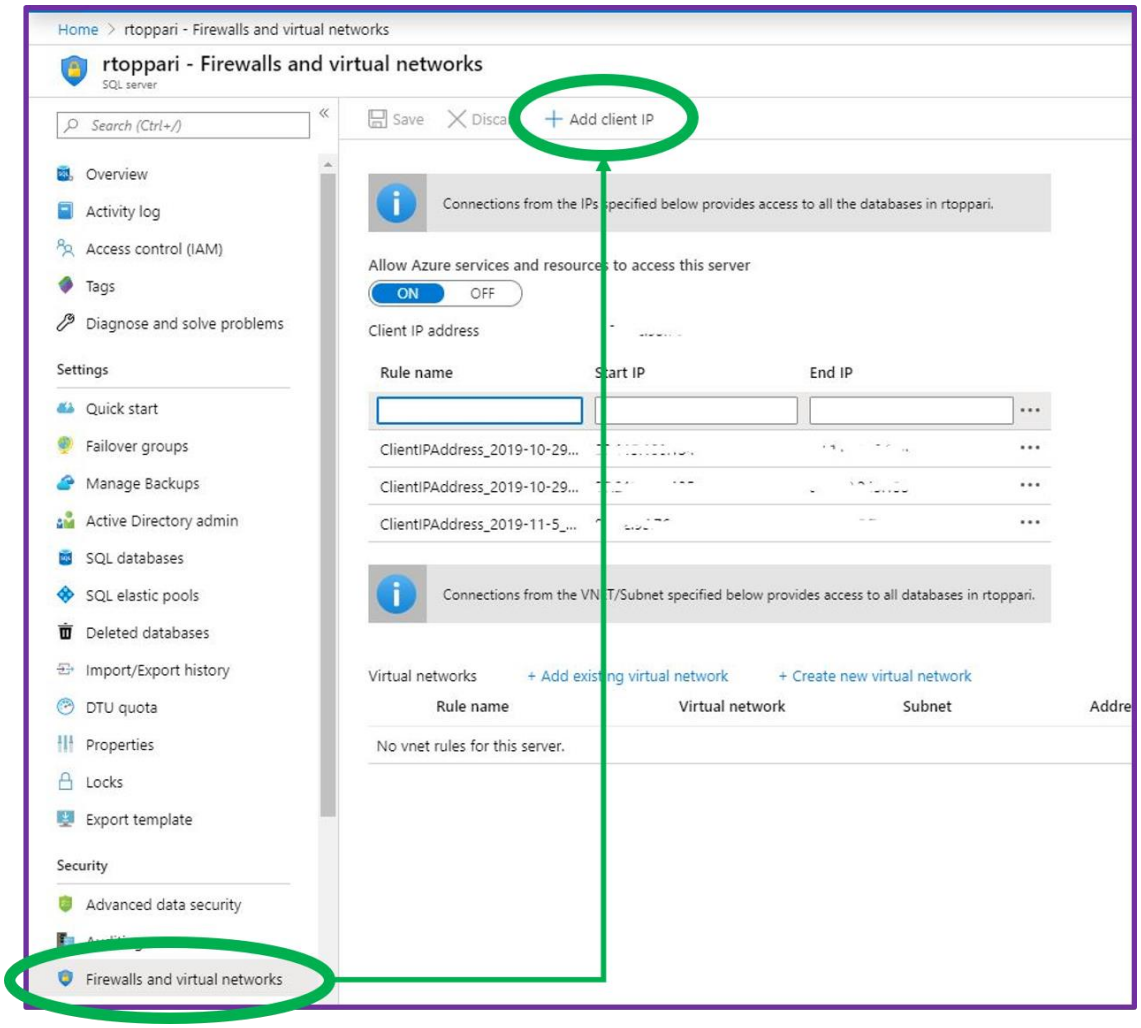

Figure 31: Server Permissions (Microsoft Azure)

Finally, you must open the Firewalls and virtual networks from the list (Figure 29) and add your client IP to the list of permitted IP addresses. You must do this for every IP address you wish to access this database with.

10.10 Creating an SQL database

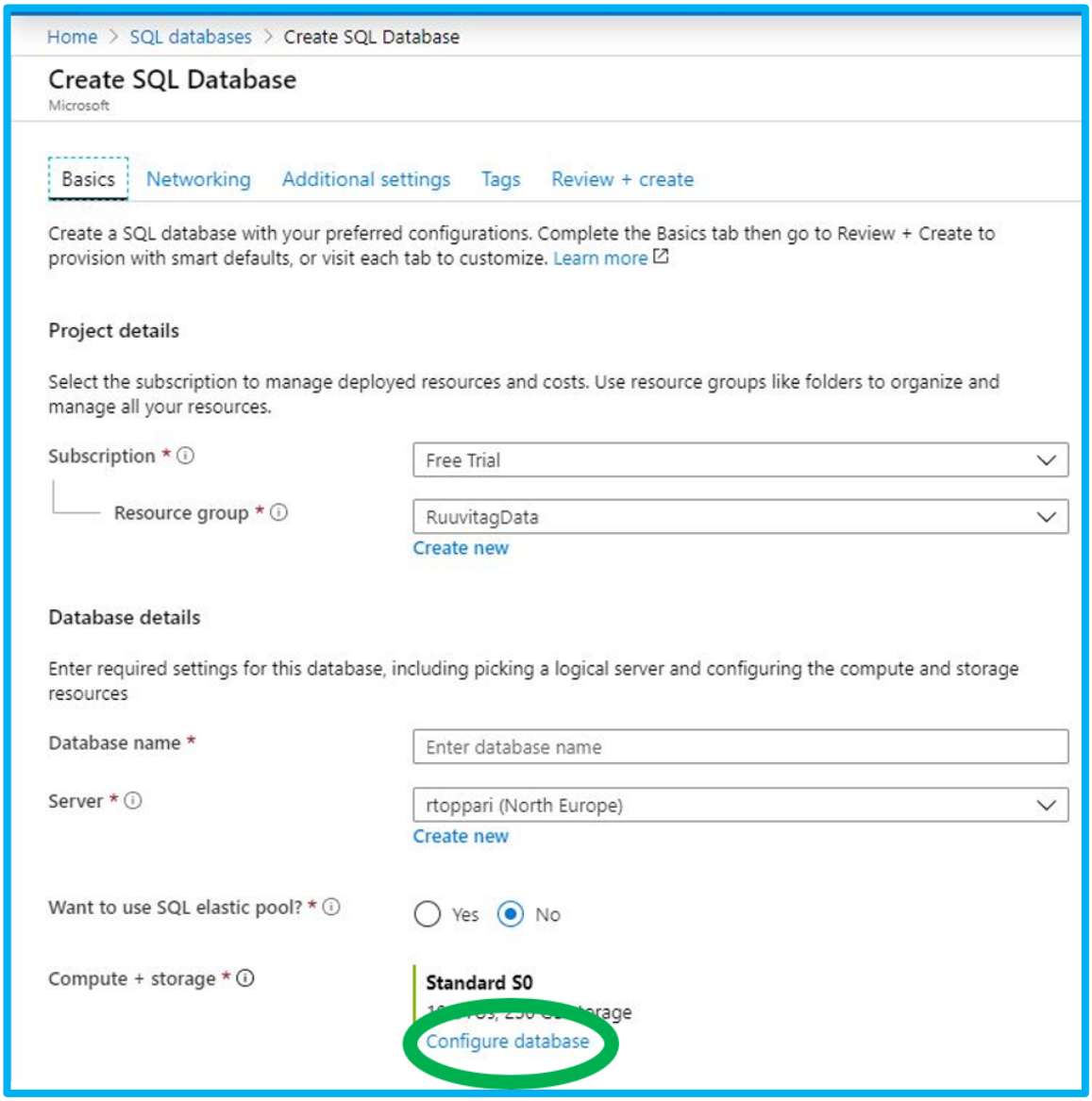

Figure 32: SQL database basics (Microsoft Azure)

Just like with other Azure resources, simply look for “SQL database” in the “add resource” menu or search for it directly.

Set the subscription as whatever one you’re using, this project was using the free subscription, which is enough for 1-2 device project. The resource group should automatically have selected the one you created earlier, if you have multiple resource groups, select the one you want this database to use.

Name the database and select the server you created earlier. There is also no need for the use of SQL elastic pool for this project, however, if you plan to use more than one database, this option is worth exploring.

It is very important to open the “configure database” as highlighted in the green circle, otherwise Azure will select “standard” as your default setting, which costs 12,65€ a month and allows storing 250GB of data, which is unnecessary for something like this project.

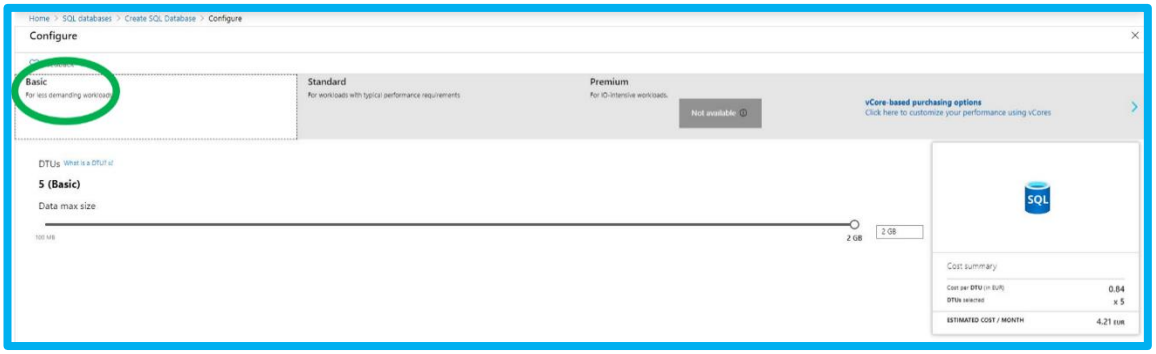

Figure 33: SQL database configuration (Microsoft Azure)

You can switch database types in the configuration page, select the “basic” option, it is more than enough for this project. If you plan to create a project with a much heavier workload, choose a configuration model that best suits it.

Once the configuration is set and the other fields filled, move on to networking.

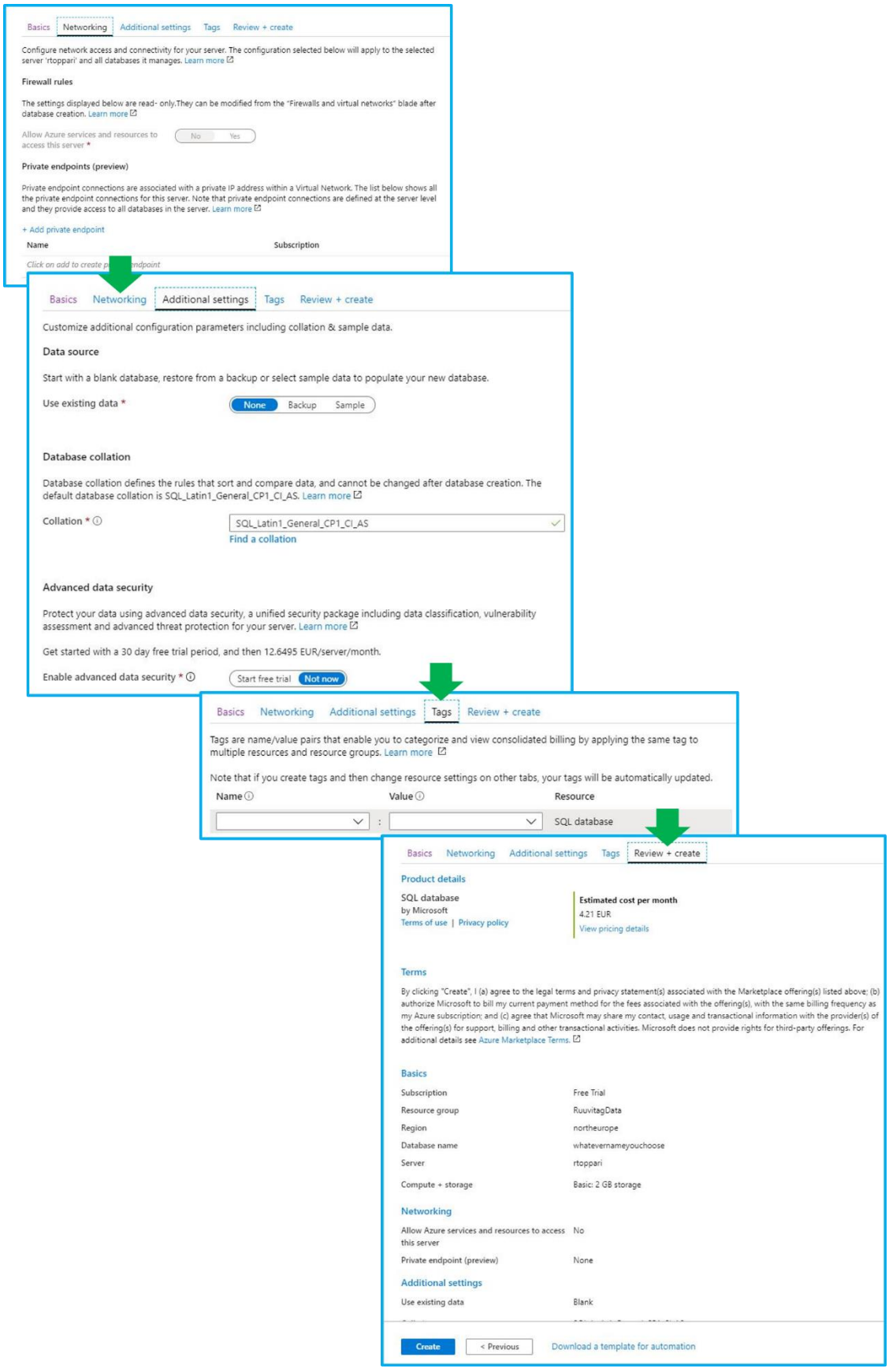

Figure 34: SQL database additional settings (Microsoft Azure)

You can’t change anything on the networking page, and you don’t need a private endpoint if you plan to use Azure to access the database.

You can select “none” for the data source, since we’re starting fresh with data collection. If you have backups or samples you want to add to the database, you can select those, but their use is not covered in this project.

Database collation has the default “SQL_Latin1_General_CP1_CI_AS” set and it works for this project. For security, there is no need for advanced security if you’re not running anything sensitive.

10.11 Integrating a Stream Analytics job to the SQL database

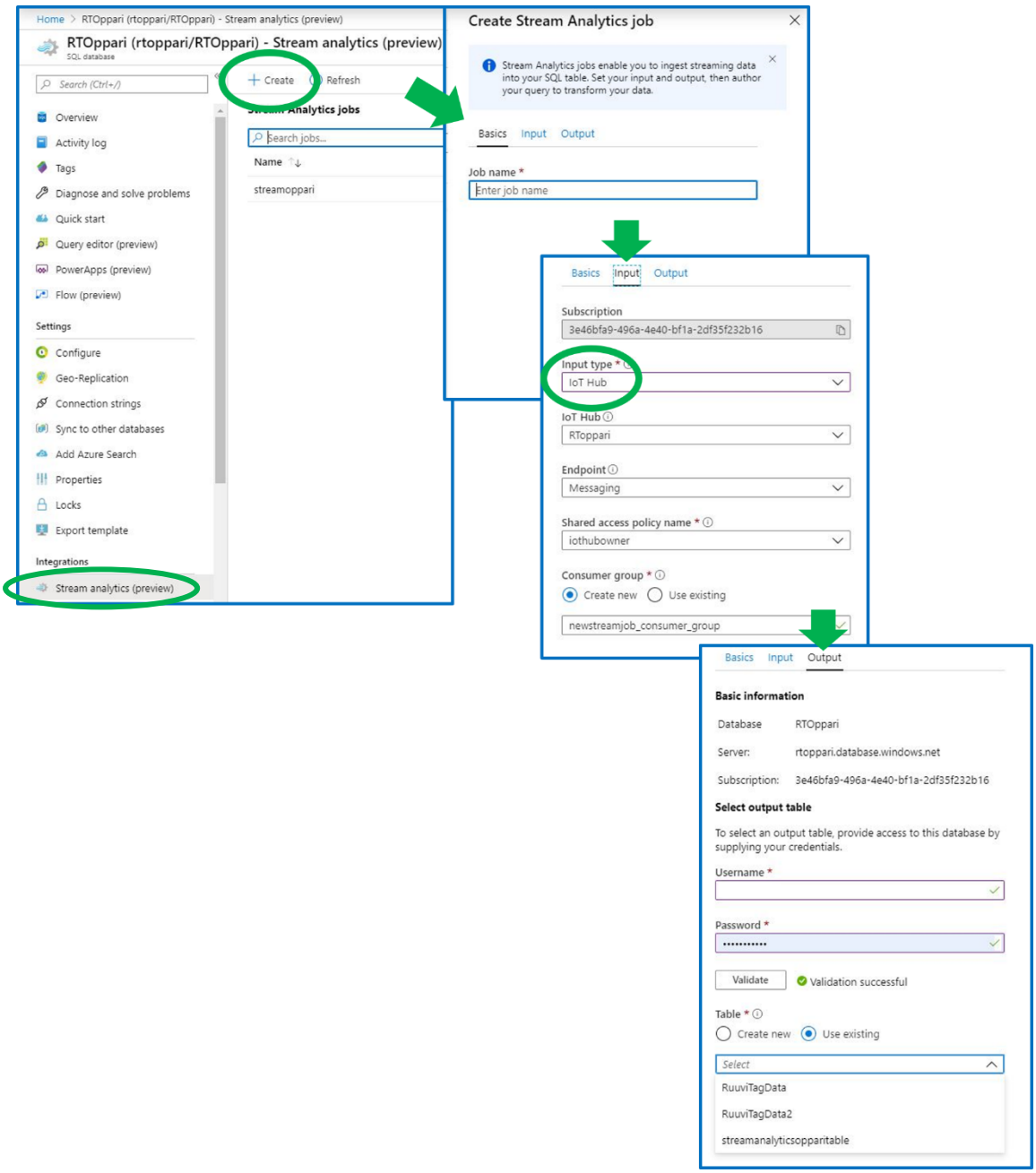

Figure 35: Stream Analytics job integration (Microsoft Azure).

First, go to the SQL database in Azure and find the “Stream Analytics (preview), highlighted in the picture above in a green circle, once you’ve clicked that it will allow you to create a new Stream Analytics job and when you click “create”, it will open a menu on the right side of the screen.

After you’ve named your Stream Analytics job, select “IoT Hub” as your input type and it should automatically insert the specific IoT Hub to the field below it, unless you have

multiple IoT Hubs to choose from. Keep the endpoint as “messaging”, since the Raspberry Pi you set up earlier will be messaging the IoT Hub. “Shared access policy name” is “iothubowner” by default, which works for this project.

For output, it should already have the information for the SQL database you opened when you started creating the Stream Analytics job and the username and password it is requesting at this stage are the ones you set for the SQL server. After you’ve been validated, you can either use an existing table, or simply select the “create new” and make a new table where the data will be stored.

10.12 Accessing database

Since we want to make sure the data has indeed been transmitted from the sensor to the database, we’ll use Power BI.

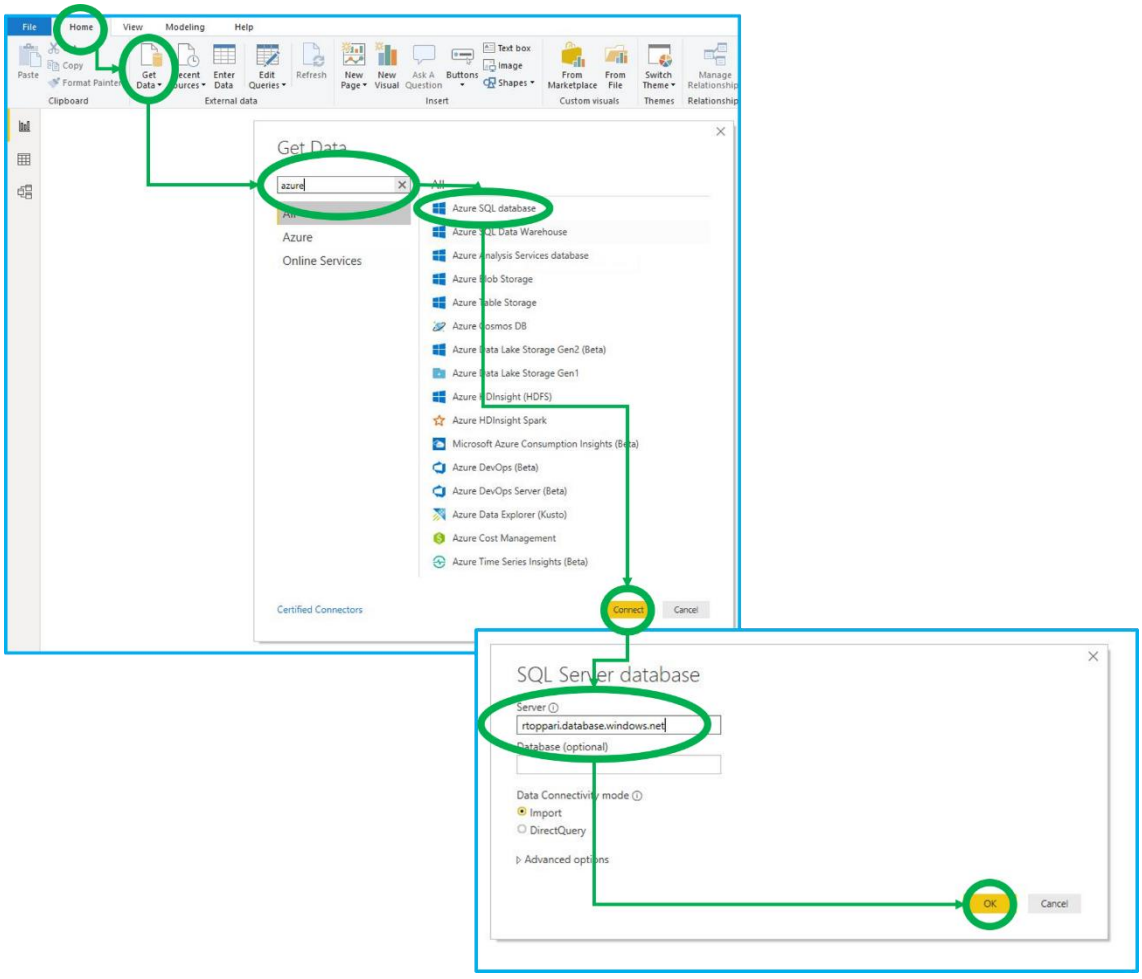

Figure 36: Getting data (Power BI)

On the home tab of Power BI, select “get data” by clicking it, which opens the get data menu. Search for Azure or select it from the list and then select “Azure SQL Database”, which then opens the connection window.

Fill the server with the link you can find in your Azure SQL database overview page.

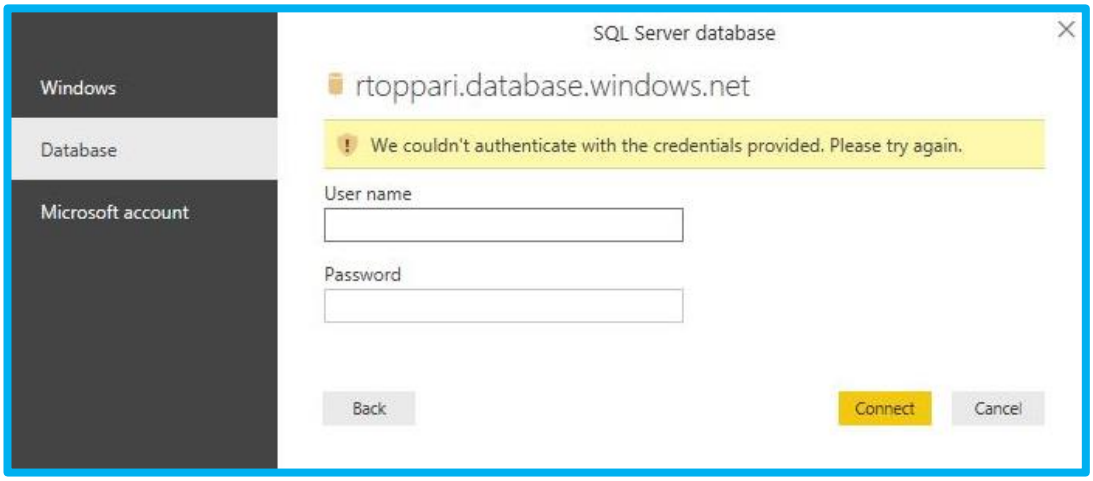

Figure 37: Admin access (Power BI)

Trying to connect to the database prompts an authentication window to pop up, you need to insert the admin log in information to continue.

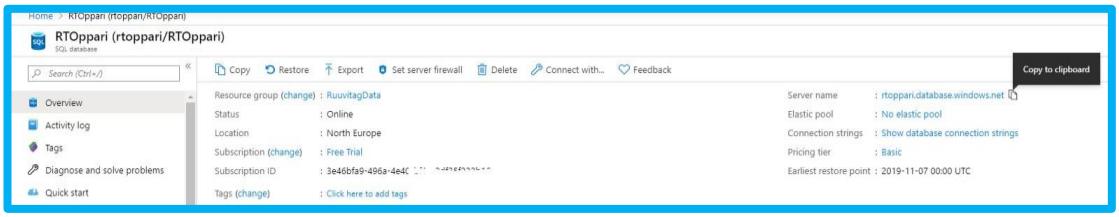

Figure 38: Finding the SQL databse server (Microsoft Azure)

You can directly copy the server link from the Azure Service. On first time connection. You must have added your IP address to the list of permitted IP addresses as instructed when creating the SQL server to access the database. First time logging in to the database, Power BI might also require you to log in with the SQL server’s administrator credentials.

You can either choose to simply import the current data in the database or choose “DirectQuery” which then connects to the database and allows for live feed on the data, but for the purposes of this project, simply importing whatever is currently in the database is enough.

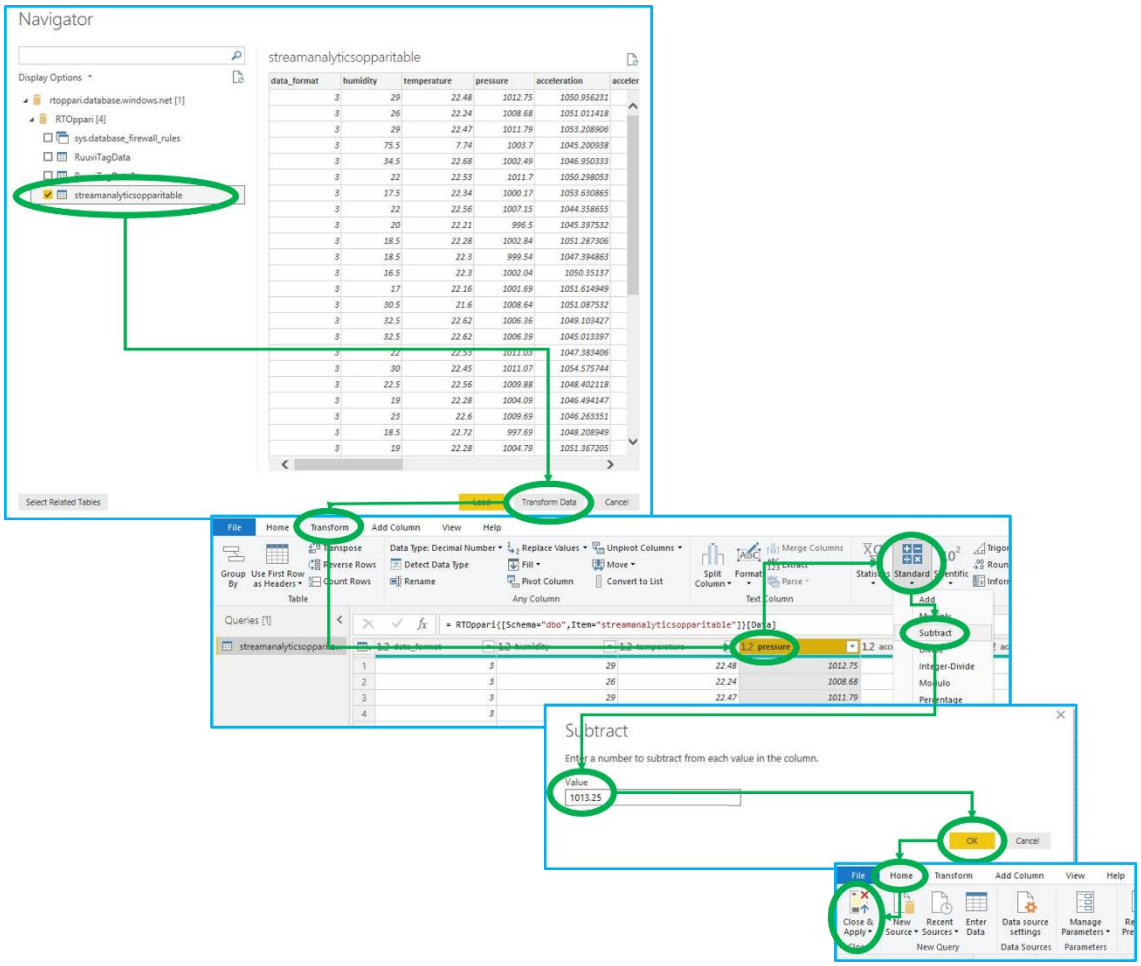

Figure 39: selecting and transforming data (Power BI)

Once you have accessed the database, select the table you want to import. Before importing, modify the tables by selecting “transform data”.

This directly opens the data tables in Power BI and allows you to modify the query. First, select the “transform” tab and then double-click the “pressure” column header and change it to “PressureDelta” and while still having the column selected, look for the “standard” option and select “Subtract” from it.

Write “1013.25” or “1013,25” (depending on your language options) into the field, it’s the standard air pressure in millibars and doing this allows creating a visualization showing the pressure changes from normal and enables tagging it with temperature and humidity.

Once the subtraction is done, select the “home tab” and close and apply the changes, which will then also automatically open the database content in Power BI.

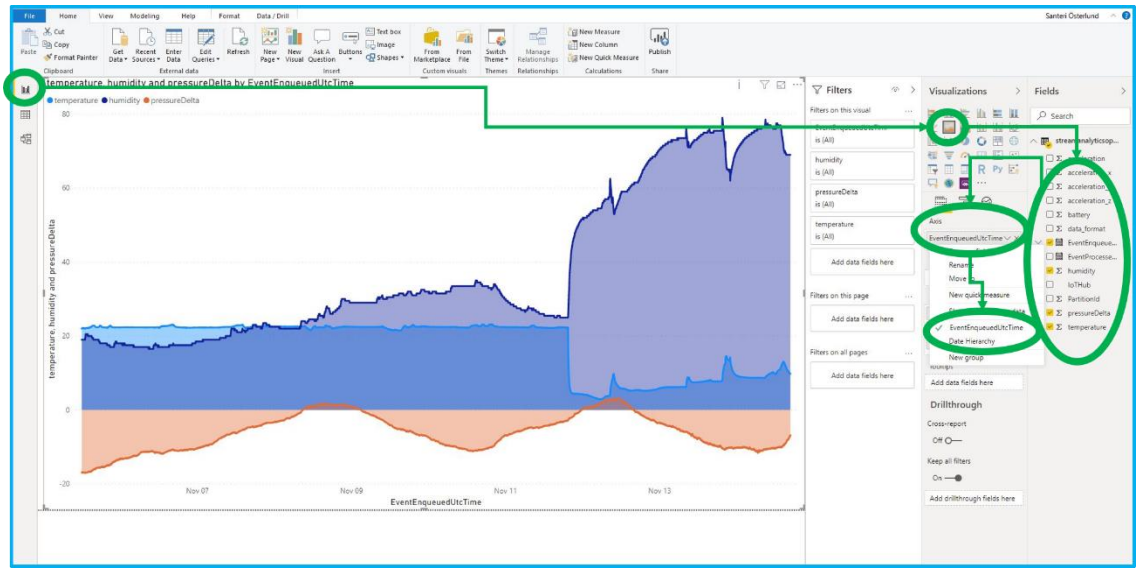

Figure 40: creating a visualization (Power BI)

On the leftmost part of the screen, select “report” and then click anywhere on the white area on the centre of the screen.

On the right, are the “visualizations” and “Fields” tabs, select “area chart” and tag “EventEnqueuedUtcTime”, “humidity”, “PressureDelta” and “Temperature”, they should automatically go into the correct fields, however, if they do not, place the “EventEnqueuedUtcTime” into Axis with the rest going into the value area.

Right click or click the small downward arrow on the “EventEnqueuedUtcTime” in the Axis spot, it opens a dropdown menu, it has Date Hierarchy selected by default, select EventEnqueuedUtcTime instead.

This is a simple visualization of the data collected by the RuuviTag environmental sensor.

11 Contemplation and suggestions

We combined the data gathering of a low end and accessible platform like the Raspberry Pie and a RuuviTag sensor with the Azure IoT hub and the platform in general.

Our solution makes the IoT solution flexible and easy to use, while being cost effective to the needs of the individual. It is flexible to the hardware you wish to use and the output format of your data. It is easy to use thanks to the tools available by the Azure platform and the platform allows you to scale it to your needs, so you do not need to pay more than you use.

The resources used for this project were somewhat limited, because of the subscription tier being “free” that restricted access to some of the features of Azure cloud resources, these limitations included things like the maximum capacity of device-to-cloud and cloud-to-device communication and database options such as blob storage. The Azure resources are easy to grasp once one gets to tinker with them, and since they are all in the Azure cloud service, despite this project not using every resource there in particular, adjusting the project with those resources should pose no problem.

A major issue when it came to using the Azure resources was the pricing and it wasn’t clear on every resource what their price was going to be, especially the Stream Analytics job, which ended up costing a lion’s share of the free subscription’s 170€ allocated funds and we had to end the project sooner than we’d have liked. Running out of funds also meant that the SQL server would go offline, cutting us off from accessing it.

The concept of the project was proven to such a capacity, that it can be used for a multitude of purposes. Any device that has a data output could have that linked to the cloud service and given the user’s programming skills, could mean creating complex IoT solutions ranging from simple data collection to outright simultaneous bilateral communication between the cloud and thousands of devices on a global scale. This has great meaning for both private and business applications, and once an IoT Hub is created and IoT devices created within, there are plenty of options available in the Azure cloud service for analysing and processing that data by simply creating a resource and linking it to the database you’re collecting your data to or starting a Stream Analytics job directly from the IoT Hub to the resource.

Furthermore, it is highly recommended for individuals looking to using Azure cloud service to look at the cost calculator offered in their website (Microsoft 2019c), as the costs can creep up depending on usage.

The project was largely based on the information created by other users of the service. This collection of different instructions and tutorials was first examined thoroughly and then applied in practice in what effectively boils down to “whatever sticks” method of elimination.

After this process, the result was an amalgamation of different instructions and two major methods, that should be easier to understand and use than other instructions available.

References

Printed sources

Arnkil, R., Spangar, T., Jokinen, E. 2007. Hyvä vertaisoppiminen kuntatyön arjessa: toteutettavuusanalyysi hyvien käytäntöjen välittämisestä. Helsinki: Suomen Kuntaliitto.

Familiar, B. 2015. Microservices, IoT and Azure. New York: Apress L. P.

Pal, A. 2017 IoT: Technical challenges and solutions. Boston: Artechouse.

Probst, D. & Schuster, P. 2016. Concepts of Proof in Mathematics, Philosophy, and Computer Science. De Gruyter.

Salo, I. 2010. Cloud computing palvelut verkossa. Porvoo: Bookwell

Slama, D., Puhlmann, F., Morrish, J. & Bhatnagar, R. 2015. Enterprise IoT. Sebastopol, CA: O'Reilly Media.

Tripathy, B. & Anuradha, J. 2018. Internet of things (IoT): technologies, applications, challenges and solutions. Boca Raton: Taylor & Francis, CRC Press.

Electronic sources

Fisch, R.; Jones, I., Jones, J., Kerman, J., Rosenkranz, Gerd K., et al. 2015. Bayesian Design of Proof-of-Concept Trials. Referenced 15.11.2019. https://search-proquest-com.nelli.laurea.fi/docview/1645918614

Microsoft 2015. Competing in the digital age of manufacturing. Referenced 23.9.2019. https://info.microsoft.com/rs/157-GQE-382/images/Industrial%20IoT_Whitepaper.pdf?aliId=865134805

Microsoft EdX 2019. Getting Started with the Internet of Things (IoT). Referenced 1.11.2019. https://courses.edx.org/courses/course-v1:Microsoft+DEV296x+2T2019/course/

Minelli, M., Chambers, M., Dhiraj, A. & Rajaram, D. 2012. Big Data, Big Analytics. http://ebookcentral.proquest.com/lib/laurea/detail.action?docID=818100

My Accounting Course 2019. What is Proof of Concept (POC)?. Referenced 14.11.2019. https://www.myaccountingcourse.com/accounting-dictionaryA /proof-of-concept

Raspberry Pi 2019a. Raspberry Pi 4 Computer Model B Product Brief. Referenced 1.11.2019. https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf

Roine, J. 2019. Building a Raspberry Pi 4-based weather monitoring solution using RuuviTag sensors, Azure IoT Hub and Azure Functions. Referenced 10.11.2019. https://jussi-roine.com/2019/07/building-a-raspberry-pi-4-based-weather-monitoring-solution-using-RuuviTag-sensors-Azure-iot-hub-and-Azure-functions/

Microsoft Azure 2018. Connect Raspberry Pi online simulator to Azure IoT Hub (Node.js). Referenced 10.11.2019. https://docs.microsoft.com/en-us/Azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started

Rouse, M. 2016. Definition: Smart Sensor. TechTarget IoT Agenda. Referenced 9.11.2019. https://internetofthingsagenda.techtarget.com/definition/smart-sensor

Rouse, M. 2018. Proof Of Concept (POC). TechTarget. Referenced 15.11.2019. https://searchcio.techtarget.com/definition/proof-of-concept-POC

Singaram, M & Jain, P. 2018. What is the Difference between Proof of Concept and Prototype? Entrepreneur India. Referenced 13.11.2019. https://www.entrepreneur.com/article/307454

Raspberry Pi 2019b. Referenced 10.11.2019. https://www.raspberrypi.org/

Ruuvi 2019. Referenced 11.11.2019. https://ruuvi.com/

Microsoft Azure 2019. Azure IoT Hub. Referenced 15.11.2019. https://Azure.microsoft.com/en-gb/services/iot-hub/

Marr, B. 2015. Big Data. Referenced 18.11.2019. https://laurea.finna.fi/Record/nelli01.3710000000335307

Microsoft 2019c. Pricing Calculator. Referenced 10.12.2019. https://Azure.microsoft.com/en-us/pricing/calculator/

Raspberry Pi. n.d. Raspberry Pi Documentation. Referenced 1.11.2019. https://www.raspberrypi.org/documentation/

Microsoft 2019b. Azure Storage Documentation. Referenced 10.12.2019. https://docs.microsoft.com/en-us/Azure/storage/

Unpublished sources

Avanade 2019. Avanade Overview. Referenced 20.11.2019.

Figures

Figure 1: Raspberry Pi 4 Model B (Raspberry Pi 2019b). 11

Figure 2: RuuviTag Sensor (RuuviTag 2019). 12

Figure 3: IoT (edX 2019) 13

Figure 4: IoT in use (Microsoft EdX 2019) 20

Figure 5: IoT Hub (Microsoft) 21

Figure 6; SQL Server (Microsoft) 21

Figure 7: SQL Database (Microsoft) 22

Figure 8: Azure Stream Analytics Job (Microsoft) 22

Figure 9: Collected data 23

Figure 10: SD Card formatter by SD Card Association 24

Figure 11: NOOBS installer 25

Figure 12: Updating the Raspberry Pie 25

Figure 13: Enabling SSH 26

Figure 14: Testing the Bluetooth 26

Figure 15: Installing bluez-hcidump 27

Figure 16: Upgrading setuptools 27

Figure 17: Installing the RuuviTag sensor library 27

Figure 18: Creating an alias and testing 27

Figure 19: Azure search (Microsoft Azure) 28

Figure 20: Adding a new resource group (Microsoft Azure) 28

Figure 21: creating a resource group (Microsoft Azure) 29

Figure 22: Adding a new Hub (Microsoft Azure) 29

Figure 23: IoT Hub creation (Microsoft Azure) 30

Figure 24: Code to send data to the IoT Hub (Roine 2019) 31

Figure 25: Testing the sending of data to IoT Hub 32

Figure 26: Making a crontab job 32

Figure 27: Editing the crontab job 33

Figure 28: Creating a storage account (Microsoft Azure) 34

Figure 29: SQL server basics (Microsoft Azure) 35

Figure 30: SQL server additional settings (Microsoft Azure) 36

Figure 31: Server Permissions (Microsoft Azure) 37

Figure 32: SQL database basics (Microsoft Azure) 38

Figure 33: SQL database configuration (Microsoft Azure) 39

Figure 34: SQL database additional settings (Microsoft Azure) 40

Figure 35: Stream Analytics job integration (Microsoft Azure) 42

Figure 36: Getting data (Power BI) 43

Figure 37: Admin access (Power BI) 44

Figure 38: Finding the SQL databse server (Microsoft Azure) 44

Figure 39: selecting and transforming data (Power BI) 45

Figure 40: creating a visualization (Power BI) 46

Tables

Table 1: Azure Blob Storage pricing (Microsoft 2019b) 17

Table 2: Data storage type, suitability and cost comparison 18