University of Padua

Department of Information Engineering

Master Degree in ICT for Internet and Multimedia

Leveraging cloud and machine learning technologies for the development of an IOT knowledge base

Master thesis

Relator
Prof. Lorenzo Vangelista

Master Candidate
Alessandro Discalzi
ID 2088235

Academic Year 2023-2024

Summary

This document describes the work done during the 750-hours final project at 221e S.r.l.
The project's goal is to architect and develop a cloud-based system capable of ingesting and processing data from heterogeneous IoT sensors so that a knowledge database can be built.
The system must be designed to be scalable and fault-tolerant, and it must be platform-agnostic.
This document is going to describe the company, the idea behind the project, the work done and an assessment of what I developed and learned during my internship.

"If the past is just dust
Then the future could be our dream"

— Lorna Shore

Acknowledgements

Prof. Lorenzo Vangelista, my thesis supervisor, deserves my deepest gratitude for his exceptional support and guidance throughout the completion of this research.
My family, for their encouragement and understanding throughout this academic endeavour, has my heartfelt thanks.
I am truly grateful to Luca Perosa, Bledar Gogaj, Marco Lionello, and all my peers at SCAI ITEC, for their unwavering support when I made the decision to pursue a Master's degree.
I extend my sincere appreciation to PhD. Roberto Bortoletto, my company tutor, and all my colleagues in 221e for their invaluable support and guidance throughout my final project.
Last but not least, I want to give a shoutout to all my friends for having my back and just being there through high and lows. Your friendship means a lot to me, and I appreciate the support and good times we've shared.

Padova, October 2024 Alessandro Discalzi

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Company

221e S.r.l.[1] is an Italian company based in Abano Terme (PD). The company, founded in 2012, is a leading supplier of IoT solution providing a wide range of products, both hardware and software, for industrial and wereable applications. The great experience and know how of the company's team, allows to provide clients with the best hardware solution for their needs, which can be enhached via the company's software platform or via a custom solution.

Figure 1.1: 221e's logo

1.2 Idea

The idea behind the project is to create a new cloud-based software platform for the company's IoT devices. The platform needs to ingest data from a variety of IoT devices, process it and create a knowledge database that can be used to provide insights and predictions to the end user.

1.3 Thesis outline

The second chapter describes the high level requirements.

The third chapter describes the technologies taken into consideration for the project.

The fourth chapter describes how the solution has been implemented.

The fifth chapter assess the results of the project.

[1] *221e S.r.l.* URL: `https://www.221e.com/`.

The last chapter describes the conclusion of the project and the future work.

Chapter 2

Objectives and requirements

In this chapter are discussed the main objectives and requirements.

2.1 Cloud Infrastructure

The system must be able to ingest, store and process large amount of IoT data leveraging the power of at least two cloud providers. Furthermore, the final product must be architectured to be cloud agnostic, meaning that it must be able to run on any cloud provider. This is important because the system must be able to run on different cloud providers to avoid vendor lock-in and to enable the customer to choose the best cloud provider for their needs.
The system will be developed using two of the following providers: Aruba Cloud[1], Amazon Web Services[2] and Microsoft Azure[3] mainly because of the already developed experience with them, both by the company and by the author of this thesis.

2.2 Data collection

The system must be able to ingest data from online devices and on-premise data sources. Online devices are devices that are connected to the internet and can send data to the cloud via MQTT protocol. On-premise data source are offline files that are stored on a local machine and must be uploaded to the cloud. The system must provide a way to upload these files to the cloud.

2.3 Data processing

The system must be able to preprocess the data before storage. The preprocessing of the data includes data validation, data cleaning and data transformation, this can be done in cloud or on-premise. The system must be also able to process data after storage. The processing of the data includes data analysis and machine learning operations. The goal of the data processing is to extract useful information from the data and to

[1] *Aruba Cloud.* URL: `https://www.arubacloud.com/`.
[2] *Amazon Web Services.* URL: `https://aws.amazon.com/`.
[3] *Microsoft Azure.* URL: `https://azure.microsoft.com/`.

provide insights to the customer.

2.4 Security

Security is a major concern for the system. In certain scenarios, the data collected could be sensitive and thus must be protected both in transit and at rest.

2.4.1 Security in transit

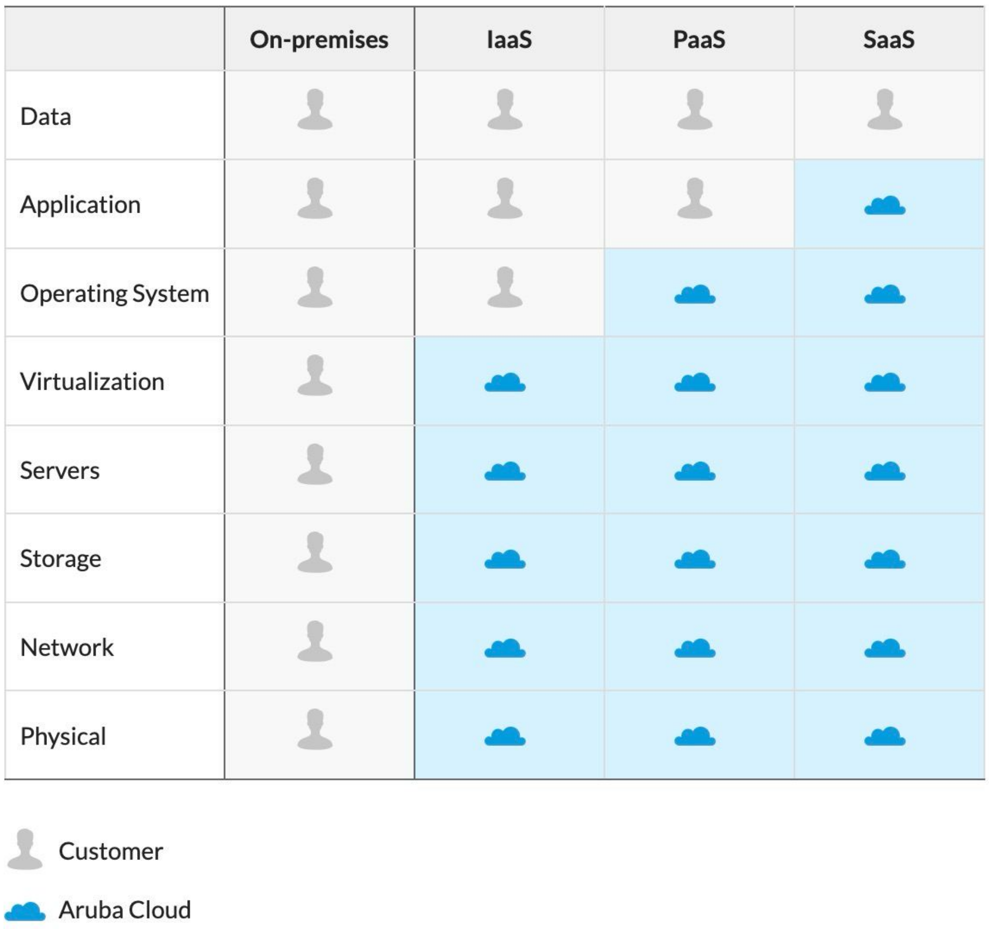
Data in transit are transfered using MQTT protocol which is not encrypted by default. The system must provide a way to encrypt and secure the data in transit. MQTT brokers however supports authentication and authorization through certificates as well as TLS/SSL encryption. Using a broker that supports these features is a must.

2.4.2 Security at rest

Data at rest is stored in cloud storage. The cloud storage must provide a way to encrypt the data at rest. The system must also provide a way to manage the encryption keys. Each cloud service provider taken into account provide a way to encrypt data at rest and manage the encryption keys.
Furthermore, each cloud provider uses a shared responsibility model for security.

Aruba Shared Responsibility Model

Aruba Shared Responsibility Model[4] is a model that defines the responsibilities of Aruba and the customer for security. Aruba is responsible for the security of the cloud, while the customer is responsible for security in the cloud.
The responsibilities of the customer and Aruba varies depending on the service, as shown in the following table.



	On-premises	IaaS	PaaS	SaaS
Data	👤	👤	👤	👤
Application	👤	👤	👤	☁
Operating System	👤	👤	☁	☁
Virtualization	👤	☁	☁	☁
Servers	👤	☁	☁	☁
Storage	👤	☁	☁	☁
Network	👤	☁	☁	☁
Physical	👤	☁	☁	☁

👤	Customer
☁	Aruba Cloud

Figure 2.1: Aruba Shared Responsibility Model

[4] *Aruba Shared Responsibility Model.* URL: `https://kb.arubacloud.com/en/computing/use-and-technology/shared-responsibility-model.aspx`.

AWS Shared Responsibility Model

AWS Shared Responsibility Model[5] is a model that defines the responsibilities of AWS and the customer for security. AWS is responsible for the security of the cloud, while the customer is responsible for security in the cloud.
It's important to keep in mind that services in services catalogued as *Infrastructure as a Service* (IaaS) the customer is responsible for the security of the operating system and the applications, while in fully managed services the customer is responsible only for the security of the data.

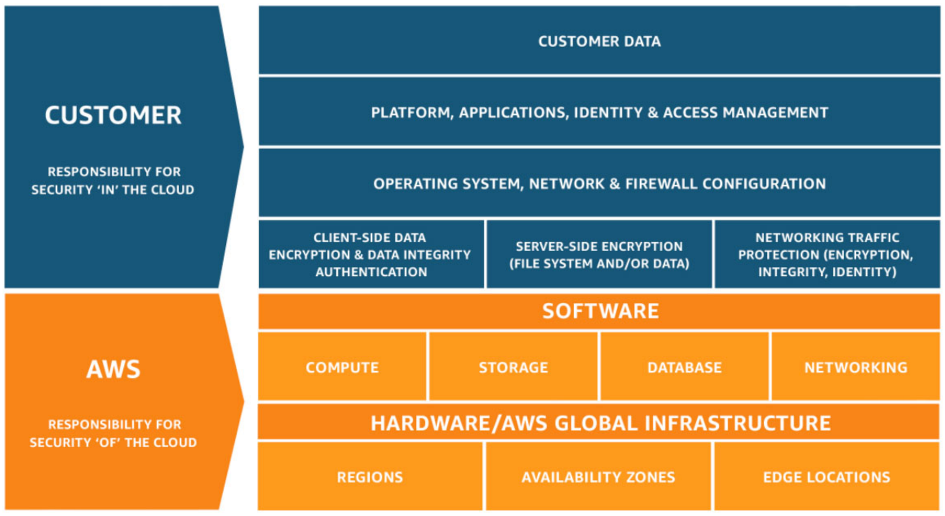
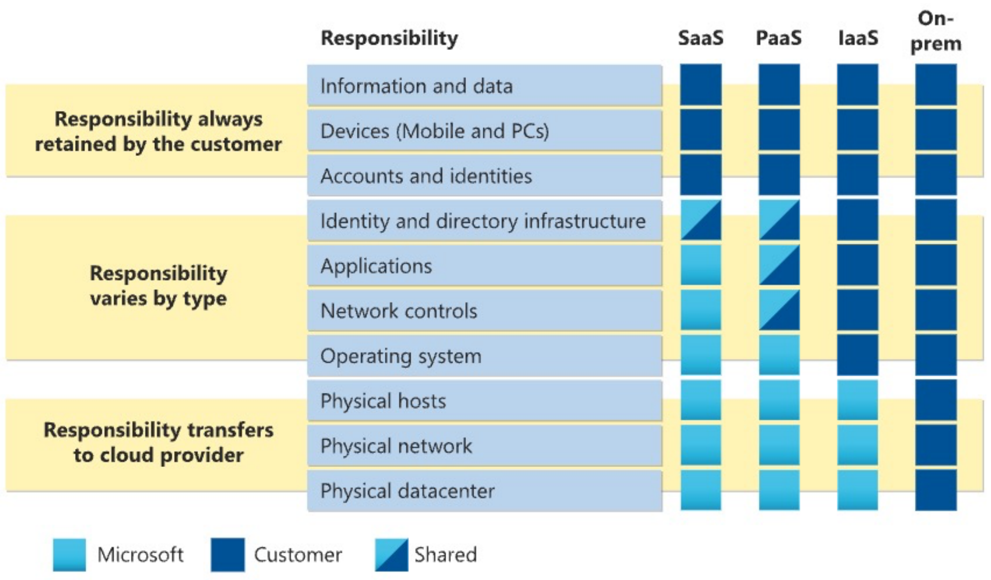

CUSTOMER
RESPONSIBILITY FOR SECURITY 'IN' THE CLOUD
CUSTOMER DATA
PLATFORM, APPLICATIONS, IDENTITY & ACCESS MANAGEMENT
OPERATING SYSTEM, NETWORK & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION (FILE SYSTEM AND/OR DATA)
NETWORKING TRAFFIC PROTECTION (ENCRYPTION, INTEGRITY, IDENTITY)
AWS
RESPONSIBILITY FOR SECURITY 'OF' THE CLOUD
SOFTWARE
COMPUTE
STORAGE
DATABASE
NETWORKING
HARDWARE/AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS

Figure 2.2: AWS Shared Responsibility Model

[5] *AWS Shared Responsibility Model.* URL: `https://aws.amazon.com/compliance/shared-responsibility-model/`.

Azure Shared Responsibility Model

Azure Shared Responsibility Model[6] is a model that defines the responsibilities of Microsoft and the customer for security. The responsibility varies depending on wheter the service is *Software as a Service* (SaaS), *Platform as a Service* (PaaS), *Infrastructure as a Service* (IaaS) or on premise. Regardless of the deployment type, the customer always detain data, endpoints account and access management responsibilities.



	Responsibility	SaaS	PaaS	IaaS	On-prem
Responsibility always retained by the customer	Information and data	■	■	■	■
	Devices (Mobile and PCs)	■	■	■	■
	Accounts and identities	■	■	■	■
Responsibility varies by type	Identity and directory infrastructure	◩	◩	■	■
	Applications	░	◩	■	■
	Network controls	░	◩	■	■
	Operating system	░	░	■	■
Responsibility transfers to cloud provider	Physical hosts	░	░	░	■
	Physical network	░	░	░	■
	Physical datacenter	░	░	░	■

░ Microsoft ■ Customer ◩ Shared

Figure 2.3: Azure Shared Responsibility Model

2.5 Scalability

The system must be able to scale horizontally and vertically. The system must be able to handle a large amount of data and must be able to automatically scale to handle more data. Fully managed services and autoscaling properties offered by the cloud providers can be used to handle the scaling of the system.

[6] *Azure Shared Responsibility Model.* URL: `https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility`.

Chapter 3

Technologies

In this chapter it's reported the study made on the various technologies taken into account to develop the project.

3.1 Cloud

This section describes the services offered by Aruba Cloud[1], Amazon Web Services[2] and Microsoft Azure[3], exploring their features, Pros: and Cons:. Services features are better described in the respective offical Aruba Cloud[4], AWS[5] and Azure[6] documentation, while Pros: and Cons: are based on the author's and the company experience as well as other users reviews that can be found in TrustRadius[7] website.
These services are among the most used cloud services in the world, offering a wide range of services that can be used to develop the project. It's also important to mention that these providers were chosen right away because of the already developed experience with them, both in the company and in the author of this thesis.

3.1.1 Aruba Cloud

Aruba Cloud is a cloud service provider that offers a wide range of services like virtual machines, object storage, databases, and managed Kubernetes.

Bare Metal

Aruba Cloud Bare Metal is a service that provides dedicated servers with high performance and reliability.
Pros:

- High performance

[1] *Aruba Cloud.*
[2] *Amazon Web Services.*
[3] *Microsoft Azure.*
[4] *Aruba Documentation.* URL: `https://kb.arubacloud.com/en/home.aspx`.
[5] *AWS Documentation.* URL: `https://docs.aws.amazon.com/`.
[6] *Azure Documentation.* URL: `https://docs.microsoft.com/en-us/azure/`.
[7] *Trust Radius.* URL: `https://trustradius.com/`.

- Reliable
- Cost effective
- Supports multiple operating systems

Cons:

- Need to manage the whole infrastructure
- Not scalable
- Not upgradable

Virtual Private server (VPS)

Aruba Cloud VPS is a service that provides virtual servers with high performance and reliability. It uses virtualizers such as OpenStacks, VMware and Hyper-V.
Pros:

- High performance
- Reliable
- Cost effective
- Supports multiple operating systems
- Scalable
- SLA up to 99.95%

Cons:

- Guaranteed resources only in the PRO plans

Virtual Private cloud (VPC)

Aruba Cloud VPC is a service that provides a virtual network isolated from the public network. It allows for the creation of multiple subnets and the configuration of security groups. It can be used to create a whole private cloud infrastructure.
Pros:

- Isolated network
- Multiple subnets
- Security groups
- Scalable
- SLA up to 99.98%

Cons:

- Complexity
- Costly

Object Storage

Aruba Cloud Object Storage is a service that provides scalable and secure object storage. It can be used to store and retrieve large amounts of unstructured data.
Pros:

- Scalable
- Secure
- Cost effective
- Highly available
- Reservable plans or pay per use

Cons:

- Limit for traffic to the public network

Managed Kubernetes

Aruba Cloud Managed Kubernetes is a service that provides a fully managed Kubernetes cluster. It can be used to deploy, scale, and manage containerized applications.
Pros:

- Fully managed
- Scalable
- Secure
- Cost effective
- Maximum redundancy
- Minimum latency
- Rapid deployment
- Kubernetes native API

Cons:

- Complexity

3.1.2 Amazon Web Services

Batch

AWS Batch is a fully managed service that enables developers to easily and efficiently run thousands of batch and machine learning computing jobs on AWS.
Pros:

- Fully managed
- Scalable
- Cost effective

- Supports different batch processing scenarios
- Supports machine learning
- Easy to use
- Versatile

Cons:

- Not well documented

Bedrock

AWS Bedrock is a fully managed service that simplifies the deployment and management of machine learning models.
Using AWS Bedrock users can chose from a variety of pre-trained models and deploy them on the edge.
Pros:

- Fully managed
- Flexible
- Native support for Retrieval Augmented Generation (RAG) models

Cons:

- Costly
- New dependencies may introduce problems
- The chosen model may not be future proof

DynamoDB

AWS DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. Tables can store and retrieve virtually any amount of data, serving any level of request traffic. It automatically spreads the data and traffic for the table over a sufficient number of servers to handle the request capacity specified by the customer and the amount of data stored, while maintaining Cons:istent and fast performance.
Pros:

- Fully managed
- Fast and predictable performance
- Scalable
- Highly available
- NoSQL

Cons:

- Hard to make changes against bulks of records
- Need to know at prior which queries will be made

Elastic map reduce (EMR)

AWS EMR is a big data platform that simplifies the deployment and management of big data frameworks, like Apache Hadoop and Apache Spark, on AWS.
Pros:

- Fully managed
- Scalable
- Petabyte scale data processing
- Easy resources provisioning
- Reconfigurable

Cons:

- Complexity
- Costly

Glue

AWS Glue is a fully managed ETL service that enables efficient data integration on a large scale.
Pros:

- Fully managed
- Pay per use
- Scalable
- Provides a centralize metadata repository
- Supports different data sources and formats
- Can automatically discover and catalog data from various sources
- Allow for job scheduling
- Data encryption

Cons:

- Costly for high workloads
- Performance issues with large datasets
- Complexity

Greengrass

AWS Greengrass is an open source edge runtime and cloud service used to build, deploy, and manage device software. It enables the devices to process the data locally, while still using the cloud for management, analytics, and durable storage.
It also enables encryption at rest and in transit and it can also extend device functionality with AWS Lambda functions.
Pros:

- Edge computing
- Encryption at rest and in transit
- Extend device functionality with AWS Lambda functions
- ML models deployment

Cons:

- Restrained to AWS services
- Not platform agnostic
- Resource intensive for small devices
- Need a connection for the initial setup

IoT Core

AWS IoT Core is a fully managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices. It is composed of multiple services like Device Management, Device Defender, Device Advisor, and IoT Analytics and only some of them can be used during the development.
Pros:

- Composed of multiple services so only the necessary ones can be used
- Encription at rest and in transit
- Supports MQTT, HTTP, and WebSockets
- Allows for device management
- Allows for machine learning at edge
- Can trigger events thanks to custom rules

Cons:

- Not platform agnostic if installed on devices
- Lacks of integration for some devices

Kendra

AWS Kendra is a fully managed enterprise search service that allows developers to add search capabilities across various content repositories leveraging on built in connectors.
Pros:

- Fully managed
- Scalable
- Supports multiple data sources
- Easy to use and set up
- Accurate search results

Cons:

- Costly

Kinesis Data Firehose

AWS Kinesis Data Firehose is a fully managed service that simplifies the process of capturing, transforming and loading streaming data. It acts as an ETL service that can capture, transform, and load streaming data into a variety of AWS services. Additionaly it can transform raw data in column oriented data formats like Apache Parquet[8]
Pros:

- Fully managed
- Can read data from IoT core and Kinesis Data Streams
- Scalable
- Can transform data
- Can load data into different AWS services
- Supports batching based on time or size

Cons:

- Not always cost effective
- Limited transformation capabilities
- Does not support batching based on more complex rules

[8] *Apache Parquet.* URL: `https://parquet.apache.org/`.

Kinesis Data Streams

AWS Kinesis Data Stream is a fully managed service that simplify the capture, processing and loading of streaming data in real time at any scale thus enabling real-time data analytics with ease.
Pros:

- Fully managed
- Scalable
- Real-time and fast data processing
- Keeps data for 24 hours by default

Cons:

- Not always cost effective
- Limited data retention
- Limited data transformation
- Not useful for certain batch processing scenarios

Lake Formation

AWS Lake Formation is a fully managed service that simplifies the creation, security and management of data lakes. It allows for cleaning and transforming the data using machine learning.
Pros:

- Fully managed
- Scalable
- Secure
- Simplifies lake creation
- Simplifies ingestion management
- Simplifies permission management
- Provides data auditing
- Supports machine learning
- Supports data cataloging

Cons:

- Complexity
- Costly
- Not native support for all data sources

Lambda

AWS Lambda is an event driven serverless compute service that automatically manages the underlying compute resources. AWS Lambda can be used to extend other AWS services with custom logic, and to create new back-end services that can operate at AWS scale, performance, and security.
Pros:

- Fully managed
- Serverless
- Pay per use
- Scalable
- Easy to integrate with other AWS services
- Supports multiple programming languages
- Easy to deploy and maintain
- Can run parallel executions
- Low time to market
- Supports custom libraries

Cons:

- Limited execution time (15 mins)
- Limited memory
- Limited environment variables
- Maximum 1000 concurrent executions
- Cold start
- Not cost effective for high workloads

Managed Service for Apache Flink (MSF)

AWS MSF is a fully managed service that simplifies the creation and the execution of real time application using Apache Flink[9] .
Pros:

- Fully managed
- Scalable
- Supports batch and stream processing
- Real-time processing
- Large-scale data processing

Cons:

- Complexity

[9] *Apache Flink.* URL: `https://flink.apache.org/`.

Managed Streaming for Kafka (MSK)

AWS MSK is a fully managed service that simplifies the setup, the scaling and the management of Apache Kafka[10] clusters.
Pros:

- Fully managed
- Scalable
- Cost effective
- Secure
- High availability
- Easy to integrate with other AWS services

Cons:

- Local testing challenges: hard to replicate the same environment in production and locally
- Not suitable for high traffic scenarios
- Complexity

Sage Maker

AWS Sage maker is a cloud-based machine learning platform that allows developer to build, train and deploy machine learning models.
Pros:

- Fully managed
- Scalable
- Supports multiple machine learning frameworks
- Supports multiple programming languages
- Allow for easy model deployment

Cons:

- Cannot schedule training jobs
- Costly for high workloads

[10] *Apache Kafka.* URL: `https://kafka.apache.org/`.

Simple Storage Service (S3)

AWS S3 is an object storage service offering scalability, data availability, security, and performance. With S3, any amount of data can be stored and retrieved from anywhere on the web. Data is stored as objects in buckets, with each object representing a file and its metadata.
Pros:

- Scalable
- Highly available
- Secure
- Durable
- Cost effective
- No bucket size limit
- No limit to the number of objects that can be stored in a bucket
- Has different storage classes to fit frequent access, infrequent access, and long-term storage

Cons:

- Not suitable for small files
- Object size limit (5TB)
- Maximum 100 buckets per account
- Max 5GB per file upload via PUT operation

3.1.3 Microsoft Azure

Blob Storage

Azure Blob Storage is a fully managed object storage service that is highly scalable and available. It can store large amounts of unstructured data, making it suitable for a wide range of workloads.
Pros:

- Fully managed
- Scalable
- Highly available
- Secure
- Cost effective
- No limit to the number of objects that can be stored in a container
- Has different storage tiers to fit frequent access, infrequent access, and long-term storage

- Different storage options (Blob, archive, queue, file and disk)

Cons:

- Not suitable for small files
- Object size limit (4TB)
- Maximum 2PB per account in US and Europe
- Maximum 500TB per account in other regions

Cosmos DB

Azure Cosmos DB is a fully managed NO-SQL database service supporting multiple data models. It supports multiple NoSQL databases like PostgreSQL, MongoDB, and Cassandra.
Pros:

- Fully managed
- Scalable
- Multi model
- Global distribution
- Cons:istency levels based on the application needs
- Easy to use

Cons:

- Expensive
- Slow for complex queries

DataBricks

Azure Databricks is a fully managed Apache Spark-based analytics platform supporting a variety of libraries and languages.
Pros:

- Fully managed
- Scalable
- Supports multiple programming languages (Python, R, Scala, SQL, Java)
- Supports multiple libraries (Tensorflow, PyTorch, Scikit-learn, etc.)
- Open data lakehouse

Cons:

- Costly
- Complexity
- Hard to configure

Data Explorer

Azure data explorer is a fully managed, real-time and high volume data analytics service. It offers speed and low latency, being able to get quick insights from raw data.
Pros:

- Fully managed
- Scalable
- Real-time data processing
- Low latency
- Supports multiple data sources
- Supports structured, semi-structured and unstructured data
- Fast data ingestion
- Can use batch processing

Cons:

- Complexity
- Costly
- Limited capabilities for data transformation
- Hard configuration

Data Factory

Azure data factory is a fully managed cloud-based data integration service. It provides tools to orchestrate data workflows while monitoring executions.
Pros:

- Fully managed
- Scalable
- Can perfom data Analytics using Synapse

Cons:

- Complexity
- Limited transformation capabilities

Data Lake Storage

Azure Data Lake Storage is a secure and scalable data lake platform. It provides a single place to store structured and unstructured data, making it easy to perform big data analytics.
Pros:

- Fully managed

- Scalable
- Secure
- Cost effective
- Compatible with Hadoop
- Supports Python for data analytics

Cons:

- Data governance challenges

Event Grid

Azure Event Grid is a fully managed event routing service that simplifies the development of event-driven applications.
Pros:

- Fully managed
- Scalable
- Supports MQTT5
- Supports multiple event sources
- Supports multiple event handlers
- Supports multiple event types
- Supports multiple programming languages
- Supports multiple event patterns

Cons:

- Complexity
- Limitations in event storage and retention
- Costly

Event Hubs

Azure Event Hubs is a fully managed, real-time data ingestion service that is simple, secure, and scalable. It can be used to stream millions of events per second with low latency, from any source to any destination. It offers also native support for Apache Kafka, allowing user to run existing Kafka applications.
Pros:

- Fully managed
- Scalable
- Secure
- Low latency

- Supports Apache Kafka
- Schema registry: centralize repository for schema management
- Real time data processing

Cons:

- Costly
- Complexity
- Limitation in event storage
- Cons:umers need to manage their state of processing

Functions

Azure functions is a serverless compute service that enables developers tu run code in response to events without the need to manage the infrastructure.
Pros:

- Fully managed
- Pay per use
- Scalable
- Supports multiple programming languages
- Easy to deploy and maintain
- Can run parallel executions
- Low time to market
- Supports custom libraries

Cons:

- Limited execution time (10 mins)
- Cold start
- Not cost effective for high workloads

IoT Hub

Azure IoT Hub is a cloud service that serves as the bridge between IoT devices and solutions in the cloud, facilitating reliable and secure communication. It can handle and manage a large number of devices making it suitable both for small-scale and enterprise-level solutions.
Pros:

- Secure
- Supports MQTT, AMQP, and HTTP
- Allows for device management

- Allows for machine learning at edge
- Can trigger events thanks to custom rules
- Can extend device functionality with Azure Functions

Cons:

- Not platform agnostic if installed on devices
- Lacks of integration for some devices
- Not well documented
- Costly
- Does not fully support MQTT5

Machine Learning

Azure Machine Learning is a fully managed service that allows developers to build, train, and deploy machine learning models.
Pros:

- Fully managed
- Scalable
- Supports multiple machine learning frameworks
- Supports multiple programming languages
- Allow for easy model deployment
- Cost effective
- Has MLOps capabilities
- Pay as you go

Cons:

- Cost raises when training big models
- Hard to optimize

3.2 Present Solutions

In this section are presented the solutions that are currently available on the market.

3.2.1 Alleantia IoT Edge Hub

IoT Edge Hub is Alleantia[11]'s plug and play solution for the industrial IoT. It offers a wide range of features like device management, alarms and events, log management and report generation. It also supports the integration with Microsoft Azure[12].
Pros:

- Plug and play
- Device management
- Alarms and events
- Log management
- Report generation
- Integration with Microsoft Azure

Cons:

- Not platform agnostic
- Does not support Amazon Web Services[13]

3.2.2 Eclipse Kura

Eclipse Kura[14] is an open source IoT Edge Framework that serves as a platform for building IoT gateways. It's based on Java/OSGi and it provides API access to the hardware interfaces of IoT Gateways.
Pros:

- Open source
- Platform agnostic
- Allows for flexible and modular development
- API access to hardware interfaces
- Introduces AI capabilities at the edge

Cons:

- Computational complexity for small devices
- Not well documented
- No native support for cloud services

[11] *Alleantia.* URL: `https://www.alleantia.com/`.
[12] *Microsoft Azure.*
[13] *Amazon Web Services.*
[14] *Eclipse Kura.* URL: `https://eclipse.dev/kura/`.

3.2.3 Eurotech Everyware Cloud

Eurotech[15] Everyware Cloud is a cloud-based IoT Integration Platform with a microservices architecture that allows to connect, configure and manage IoT gateways and devices.
Pros:

- Cloud-based
- Microservices architecture
- Allows to connect, configure and manage IoT gateways and devices
- Supports multiple protocols
- Supports multiple cloud services

Cons:

- Last update in 2019

3.2.4 STMicrelectronics X-Cube Cloud

STMicrelectronics[16] X-Cube Cloud is a software package that enables the connection of STM32 microcontrollers to the cloud.
Pros:

- Supports multiple cloud services
- Offers generic and secure connection to the cloud
- Supports multiple protocols

Cons:

- Specific for STM32 microcontrollers
- Specific packages for each cloud service if you want to use the full potential
- The generic solution only runs on a subset of STM32 microcontrollers

3.2.5 MQTTX

MQTTX[17] is a cross-platform MQTT 5.0 client tool that can be used to publish and subscribe to MQTT messages.
Pros:

- Cross-platform
- Supports MQTT 5.0
- Connection management
- Log capabilities
- Data pipelines

Cons:

- MQTT client only

[15] *Eurotech.* URL: `https://www.eurotech.com/`.
[16] *STMicroelectronics.* URL: `https://www.st.com/content/st_com/en.html`.
[17] *MQTTX.* URL: `https://mqttx.app/`.

3.2.6 EMQX

EMQX[18] is an open source MQTT broker designed to be highly scalable.
Pros:

- Open source
- Scalable
- Supports MQTT 5.0
- Supports web sockets
- Supports multiple cloud services

Cons:

- The free version has limitations

3.3 Machine Learning at edge

This section describes the technologies that can be used to build and deploy machine learning models at the edge and the Federated Learning approach.

3.3.1 Tensorflow Lite

Tensorflow Lite[19] is the mobile and edge version of Tensorflow[20] that allows to run machine learning models on edge devices.
Pros:

- Lightweight
- Supports multiple platforms
- Easy to use
- Well documented

Cons:

- Need to pretrain the model before deploying it
- On-device training is limited to Unix-based systems

3.3.2 Tiny Engine

Tiny Engine[21][22] is a machine learning framework that allows to build, train and deploy machine learning models on edge devices.
Pros:

[18] *EMQX*. URL: `https://www.emqx.io/`.
[19] *TensorFlow Lite*. URL: `https://www.tensorflow.org/lite`.
[20] *TensorFlow*. URL: `https://www.tensorflow.org/`.
[21] *TinyEngine*. URL: `https://github.com/mit-han-lab/tinyengine`.
[22] Ji Lin et al. "On-Device Training Under 256KB Memory". In: *arXiv:2206.15472 [cs]* (2022). URL: `https://arxiv.org/abs/2206.15472`.

- Lightweight
- Supports multiple platforms
- Converts Tensorflow Lite models to C++
- Uses pre trained models

Cons:

- Need to pretrain the model before deploying it
- Better support for MCUnet[23]
- Not well documented for deployment of custom models

3.3.3 Federated Learning and Transfer Learning

Federated Learning is a machine learning approach that trains an algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging them. This approach allows for privacy preservation and data security, as well as for the possibility to train models on devices with low computational power as described in "EdgeFed: Optimized Federated Learning Based on Edge Computing"[24]. Another important approach that has been taken into account is Transfer Learning, a technique that transfer the knowledge of a model trained on a specific task to a new task, improving the performance of the model and reducing the time and resources needed to train it, as described in "Federated learning for IoT devices: Enhancing TinyML with on-board training"[25].
Pros:

- Privacy preservation
- Data security
- No need for data centralization
- Low computational power needed
- Low latency

Cons:

- Complex to implement
- Limited capabilities

[23]Ji Lin et al. "Mcunet: Tiny deep learning on iot devices". In: *Advances in Neural Information Processing Systems* 33 (2020). URL: https://arxiv.org/abs/2007.10319.
[24]Yunfan Ye et al. "EdgeFed: Optimized Federated Learning Based on Edge Computing". In: *IEEE Access* 8 (2020), pp. 209191–209198. DOI: 10.1109/ACCESS.2020.3038287.
[25]M. Ficco et al. "Federated learning for IoT devices: Enhancing TinyML with on-board training". In: *Information Fusion* 104 (2024), p. 102189. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2023.102189. URL: https://www.sciencedirect.com/science/article/pii/S1566253523005055.

Chapter 4

Methodology

Introduction

In this chapter, we will present the methodology used to develop the proposed solution. The chapter is divided into two sections: architecture, and implementation. The architecture section will present the proposed solution's architecture for the chosen cloud providers, while the implementation section will detail the steps taken to implement the solution.

4.1 Architecture

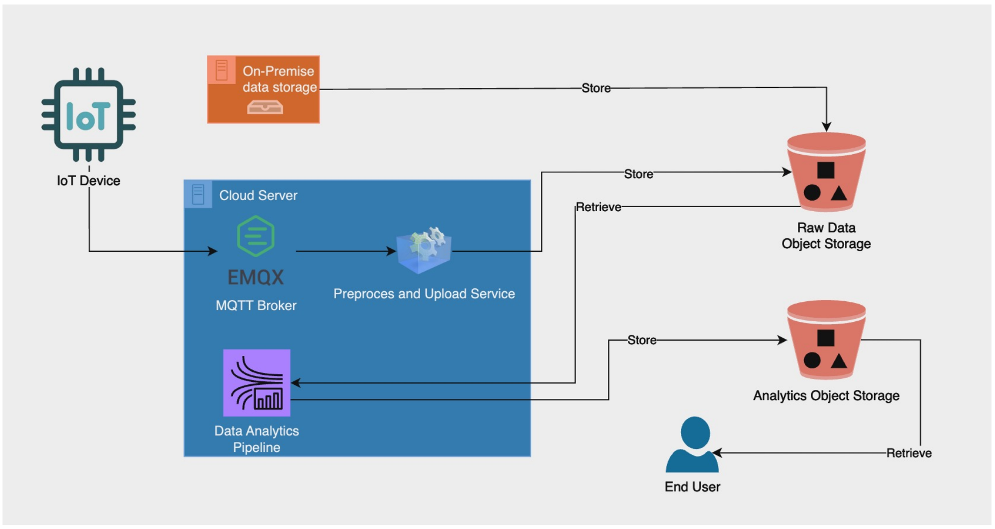


Figure 4.1: Architecture of the proposed solution

The proposed solution's architecture consists of a composition of loosely coupled microservices that work together to collect, process, and store data. The architecture is divided into three layers: the data collection layer, the data storage layer and the data analysis layer. Each layer plays a specific role in the overall system, ensuring scalability, flexibility, and resilience.

4.1.1 Data Collection Layer

The data collection layer is responsible for gathering data from IoT devices and archived data.

4.1.2 Data Collection from IoT Devices

The data collection from IoT devices utilizes the MQTT protocol, a lightweight messaging protocol designed for devices with low bandwidth and high latency. MQTT operates on a publish-subscribe model, where devices publish messages to a broker, which then forwards the messages to subscribers. This protocol is ideal for IoT devices due to its simplicity, ease of implementation, and reliability, with support for different quality of service (QoS) levels to ensure message delivery.

In our scenario, a large number of IoT devices publish data to the cloud. Each device publishes data to a specific topic managed by a cloud-based broker. The data is sent in JSON format, with each message containing a timestamp and a set of key-value pairs representing the data. Once the data is published to the broker, it undergoes preprocessing before being stored in object storage. This preprocessing involves parsing the JSON data, extracting the timestamp and key-value pairs, and converting the data to CSV format, which is less verbose than JSON. The preprocessed data is then stored in object storage, making it accessible to the data processing pipeline.

4.1.3 Data Collection from Archived Data

The data collection from archived data is done by uploading files from on-premises to the cloud via a simple script that leverages cloud API. The files, which are already in CSV format, are uploaded to the object storage, where they can be accessed by the data processing pipeline. The files contain historical data that have been collected over time in a number of different scenarios.

4.1.4 Data Storage Layer

The data storage layer is responsible for storing the raw CSV data and making it accessible to the data processing pipeline. This layer utilizes cloud-based object storage for scalability and durability. The data, always stored in CSV format, is uploaded to the object storage by the data collection layer and accessed by the data analysis layer. Once the analysis is done, the results are stored in a separate object storage bucket, making them accessible to other services or users. The object storage provides a reliable and cost-effective solution for storing large amounts of data, with built-in redundancy and scalability features.

4.1.5 Data Analysis Layer

The data analysis layer is responsible for analyzing the collected data and generating insights. This layer consists of several services that retrieve the preprocessed data from the object storage and perform various analytical tasks. The data, initially preprocessed and stored in CSV format, is accessed directly from the object storage by the microservices.

Once the data is retrieved, the microservices perform advanced analytics and machine learning tasks using custom algorithms tailored to specific requirements. These analytical processes transform the raw data into valuable insights, enabling more informed decision-making and deeper understanding of the underlying patterns and trends in the collected data. Right after the analysis is done, the results are stored in a separate object storage bucket, making them accessible to other services or users.

4.1.6 Cloud Server vs. SaaS Solutions

The choice of using a cloud server to host both the MQTT broker and the data analytics pipeline provides several advantages over SaaS solutions. While SaaS architectures have been considered, a cloud server offers more flexibility and control over the infrastructure, which is crucial for custom configurations and dependencies required by the data analytics pipeline. It is also more cost-effective for long-term projects, as SaaS solutions typically incur significant monthly fees. A cloud server allows better control over costs, with the trade-off being the need to manage infrastructure and security patches.

Furthermore, this solution is seamlessly implementable with any cloud provider since it doesn't rely on specific services unique to a particular provider. This cloud-agnostic approach allows users to choose the cloud provider that best fits their needs without altering the solution's architecture. Additionally, even though the MQTT broker and the data analytics pipeline are hosted on the same server, they remain independent, respecting the principles of a microservices architecture.

In summary, this architecture provides a robust, scalable, and flexible solution for data collection, processing, and storage, with the added benefit of being cloud-agnostic and cost-effective for long-term deployments.

4.2 Implementation

The implementation of the proposed solution involves several steps, including setting up the cloud server, configuring the MQTT broker, setting up the storage services , and developing the data processing pipeline. The implementation is divided into two main components: the data collection component and the data analysis component.

4.2.1 Data Collection Component

The data collection component is responsible for collecting data from IoT devices and archived data. This component consists of two main parts: the MQTT broker and the data upload script.

MQTT Broker

Data Upload Script

4.2.2 Data Analysis Component

Chapter 5

Results

Chapter intro

5.1 Tests

Chapter 6

Conclusion

6.1 Objectives achieved

6.2 Future developments

6.3 What I learned

6.4 Final considerations

Chapter 7

Bibliography

Article references

Ficco, M. et al. "Federated learning for IoT devices: Enhancing TinyML with on-board training". In: *Information Fusion* 104 (2024), p. 102189. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2023.102189. URL: https://www.sciencedirect.com/science/article/pii/S1566253523005055 (cit. on p. 28).

Lin, Ji et al. "Mcunet: Tiny deep learning on iot devices". In: *Advances in Neural Information Processing Systems* 33 (2020). URL: https://arxiv.org/abs/2007.10319 (cit. on p. 28).

Lin, Ji et al. "On-Device Training Under 256KB Memory". In: *arXiv:2206.15472 [cs]* (2022). URL: https://arxiv.org/abs/2206.15472 (cit. on p. 27).

Ye, Yunfan et al. "EdgeFed: Optimized Federated Learning Based on Edge Computing". In: *IEEE Access* 8 (2020), pp. 209191–209198. DOI: 10.1109/ACCESS.2020.3038287 (cit. on p. 28).

Website references

221e S.r.l. URL: https://www.221e.com/ (cit. on p. 1).

Alleantia. URL: https://www.alleantia.com/ (cit. on p. 25).

Amazon Web Services. URL: https://aws.amazon.com/ (cit. on pp. 3, 9, 25).

Apache Flink. URL: https://flink.apache.org/ (cit. on p. 17).

Apache Kafka. URL: https://kafka.apache.org/ (cit. on p. 18).

Apache Parquet. URL: https://parquet.apache.org/ (cit. on p. 15).

Aruba Cloud. URL: https://www.arubacloud.com/ (cit. on pp. 3, 9).

Aruba Documentation. URL: https://kb.arubacloud.com/en/home.aspx (cit. on p. 9).

Aruba Shared Responsibility Model. URL: https://kb.arubacloud.com/en/computing/use-and-technology/shared-responsibility-model.aspx (cit. on p. 6).

AWS Documentation. URL: `https://docs.aws.amazon.com/` (cit. on p. 9).

AWS Shared Responsibility Model. URL: `https://aws.amazon.com/compliance/shared-responsibility-model/` (cit. on p. 7).

Azure Documentation. URL: `https://docs.microsoft.com/en-us/azure/` (cit. on p. 9).

Azure Shared Responsibility Model. URL: `https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility` (cit. on p. 8).

Eclipse Kura. URL: `https://eclipse.dev/kura/` (cit. on p. 25).

EMQX. URL: `https://www.emqx.io/` (cit. on p. 27).

Eurotech. URL: `https://www.eurotech.com/` (cit. on p. 26).

Microsoft Azure. URL: `https://azure.microsoft.com/` (cit. on pp. 3, 9, 25).

MQTTX. URL: `https://mqttx.app/` (cit. on p. 26).

STMicroelectronics. URL: `https://www.st.com/content/st_com/en.html` (cit. on p. 26).

TensorFlow. URL: `https://www.tensorflow.org/` (cit. on p. 27).

TensorFlow Lite. URL: `https://www.tensorflow.org/lite` (cit. on p. 27).

TinyEngine. URL: `https://github.com/mit-han-lab/tinyengine` (cit. on p. 27).

Trust Radius. URL: `https://trustradius.com/` (cit. on p. 9).

"Te ghe da pomparghe drio"

— *Federico Gallo*

Dedicated to my family, my friends, and my mentors.