Pirelli Challenge

Domanda 1:

After some discussions with Juanito, it was decided that the project will start with the creation of a unified data platform on the cloud with the purpose of being the single point of access to all the data from the company. The goal is to come up with a proposal for taking the data into the cloud.

a. You are asked to focus on the data storage components only. Which type of systems would you use and why (distributed filesystem, object storage, RDBMS, No-SQL, DWH, etc.)? How would you provide and manage access to the Maria and the kitchen workers?
b. You are now discussing an architecture proposal for the whole system that starts from the source systems up to the storage components you defined before. What constraints or requirements would you take into account while evaluating the proposal? What types of proposals would you avoid? You might give examples of a high level architecture while writing your answers. Remember to justify them.

Consider the following requirements while working on your answers:

- The impact on the existing applications developed by the stores' workers and Maria should be minimal.
- A team of data scientists will be joining the company to start working on models to support the decision making processes, so don't forget about their potential needs!

Assunzioni:

- AWS come cloud provider
- Gli store possiedono dei server dotati di connettività internet
- Il DWH centralizzato risiede su un server dotato di connettività internet

Risposta punto a:

A livello di storage si deve necessariamente considerare la situazione di partenza, ovvero la presenza di un RDBMS in ciascuno store e di un DWH presso la casa di Juan. Entrambe sono quindi tecnologie di tipo relazionale.

La soluzione più semplice sembrerebbe riportare un sistema relazionale anche in Cloud prevedendo l'utilizzo di appositi sistemi di migrazione dati (Change Data Capture o tool nativi AWS che permettono la migrazione – si veda risposta successiva).

Questa soluzione però, visti i programmi di espansione ed eventuali use case sui dati raccolti non sarebbe scalabile e comporterebbe notevole effort a livello di gestione (un unico tool relazionale o più istanze? Come gestire possibili richieste di data scientist in futuro? etc.).

Probabilmente la soluzione migliore (oltre che quella favorita dallo stato dell'arte) è l'utilizzo di un object store come S3.

In questo modo sarebbe possibile convogliare in uno o più bucket (a seconda del partizionamento che si vuole mantenere) tutti i dati derivanti dai vari store distribuiti in città (e, successivamente cross città). Tale soluzione permetterebbe inoltre di portare facilmente i dati all'interno di una

struttura relazionale (AWS Redshift) utilizzando servizi come Redshift Spectrum su esigenza o di permetterne la fruizione tramite Athena per un accesso semplice e SQL-like al dato.

In aggiunta, a seconda del formato con cui si considera di immagazzinare i dati (e.g. csv, parquet, etc.) è possibile:

- rispondere con efficacia e tempestività alle eventuali richieste dei data scientist sull'accesso al dato
- ottimizzare l'utilizzo dello spazio di storage a seconda delle tecniche di compressione e cifratura del dato messe a disposizione nativamente da AWS

Sfruttando poi le policy di retention è possibile ottimizzare i costi prevedendo lo spostamento dei dati tra i diversi Tier di S3 nel corso del tempo.

Per quanto riguarda l'accesso al dato:

- a livello di sicurezza, è possibile utilizzare il servizio IAM dove andranno censiti i vari utenti/ruoli per l'accesso al dato
- a livello di fruizione, tramite Athena (SQL-like) o con delle dashboard immediate costruite su un servizio come QuickSight

Per quanto riguarda l'impatto sulle applicazioni esistenti:

- nell'ipotesi in cui si mantenga il RDBMS in ogni store e il DWH presso la sede centrale, allora non verrebbero impattate minimamente
- nell'ipotesi in cui, invece, si migri completamente verso il Cloud, allora le applicazioni potrebbero essere migrate su ambiente AWS utilizzando servizi come EKS o simili, o semplicemente venendo deployate su delle EC2. Anche in questo caso l'impatto sarebbe minimo: a partire dall'object store di aws è possibile caricare i dati su un DWH centralizzato come Redshift e a quel punto si tratterebbe solo di configurare la nuova sorgente dati e 'aggiustare' eventuali differenze di dialetto SQL.

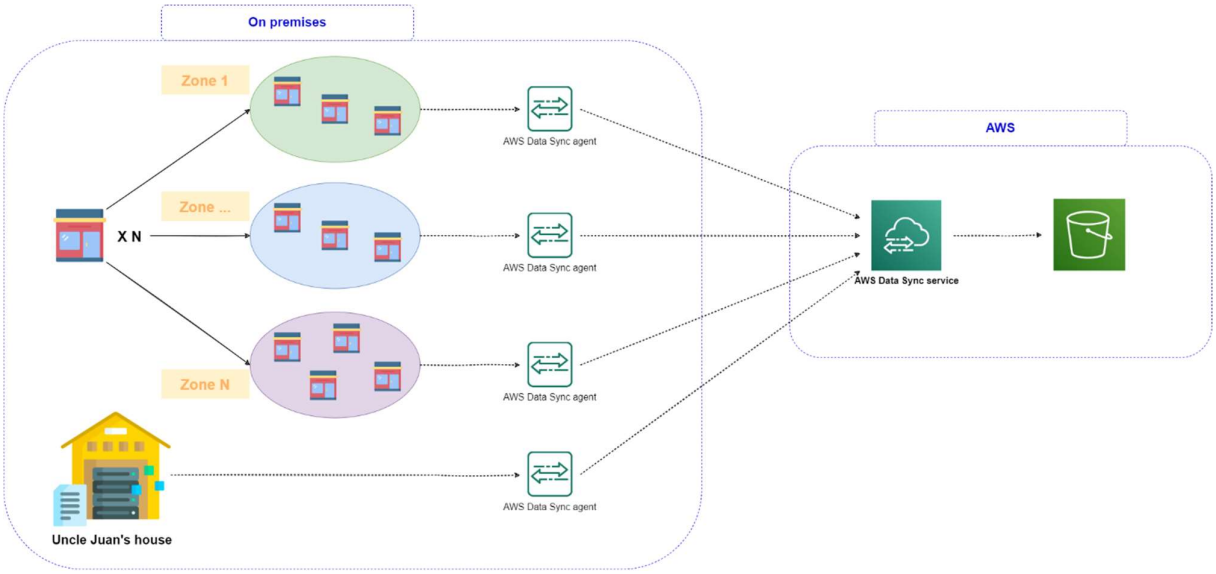
Risposta punto b:

In questo caso le opzioni percorribili sono due, che vanno di pari passo con gli ultimi due punti riportati alla risposta precedente.

In un primo caso, si potrebbe pensare di mantenere comunque i sistemi di storage locali, in modo tale da non dover modificare nulla a livello locale ma portando avanti una soluzione con una parte on-premises affiancata a una data platform in cloud.

Probabilmente a livello di costo sarebbe più onerosa come soluzione, porta in ogni caso benefici dall'off-loading dei dati e quindi eventuali applicazioni analitiche sviluppate su data platform in cloud non insisterebbero sulle stesse sorgenti dati ad oggi previste. In più a livello di data platform si potrebbero sviluppare applicazioni/modelli/dashboard a livello globale e non solo locale.

Per uno scenario di questo tipo si può pensare di utilizzare servizi di migrazione di storage come AWS DataSync o DMS (Database Migration Service); in questo secondo caso la possibilità di migrazione è legata alla tecnologia dei sistemi di storage presenti on-premises (non tutti i sistemi sono supportati dal tool).



Nel secondo caso, invece, si può pensare di non avere più gli storage locali, ma solamente la data platform centralizzata.

In questo scenario è verosimile pensare di sfruttare tool come AWS Kinesis o MSK (Managed Service for Kafka).

In tal modo si andrebbero a sviluppare delle applicazioni, deployate sui server in ogni store in grado di collezionare i dati riguardanti lavorazione e preparazione dei prodotti e anche tutta la parte di inventario e vendite.

In uno scenario di completo rinnovamento, a meno che l'architettura sia già esistente, si potrebbe pensare di equipaggiare i vari macchinari con sensoristica (IoT) e raccogliere i dati localmente inviandoli poi verso la data platform.

Utilizzando Kinesis, sarebbe possibile utilizzare sia KDF (Kinesis Data Firehose) per l'invio dei dati verso il Cloud (con destinazione S3), sia KDA (Kinesis Data Analytics) – inserito come una delle destinazioni di KDF – per fornire elaborazioni di dati e dashboard on-the-flight e consentire cosi azioni di predictive maintanance in caso di problemi rilevati.

Ovviamente tutta questa architettura, anche nel caso di utilizzo di MSK al posto di Kinesis, avrebbe un notevole esborso in termini di costi e effort inizialmente ma sul lungo periodo probabilmente consentirebbe di avere un sistema sempre aggiornato con dati in real time e fruibile da tutta la catena di Juan.