Università degli studi di Modena e Reggio Emilia

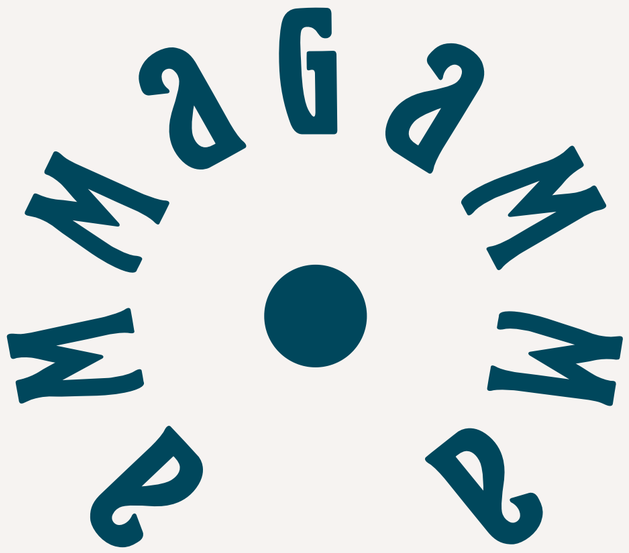
Dipartimento di Scienze Matematiche, Fisiche e Informatiche
Corso di Laurea in Informatica

Progetto, sviluppo e orchestrazione di pipeline ETL tramite Apache Airflow in esecuzione su servizi Cloud

RELATORI
Prof. Riccardo Martoglia
Andrea Giannetti

LAUREANDO
Francesco Barbanti

Anno Accademico 2021/2022



Ammagamma

Società che offre strumenti di scelta attraverso lo sviluppo di soluzioni innovative di intelligenza artificiale

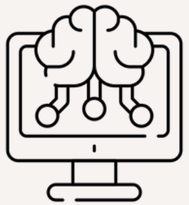
Obiettivi del tirocinio

- Progettare, orchestrare e ridistribuire le pipeline ETL che in origine eseguivano sul servizio AWS Glue su un'istanza di Apache Airflow rilasciata sull'architettura cloud di AWS
- Tutor aziendale: **Andrea Giannetti**

Indice

IL CASO DI STUDIO

All'interno della collaborazione in corso da svariati anni con un noto quotidiano economico politico finanziario italiano, Ammagamma ha sviluppato:



Modelli di intelligenza artificiale

Predizione del tasso di abbandono sui loro principali abbonamenti.

4 Pipeline ETL

In grado di trasformare, predire e valutare il tasso di abbandono mensile ed annuale su due perimetri distinti

Architettura cloud

Utilizzo dei servizi cloud di AWS per memorizzare i dati, schedulare le pipeline, definire i flussi di lavoro ed eseguire i job

IL CASO DI STUDIO - PIPELINE ETL

Le pipeline ETL prese in esame sono 4 e si possono suddividere in 2 famiglie:

Tutte le pipeline condividono la stessa struttura:

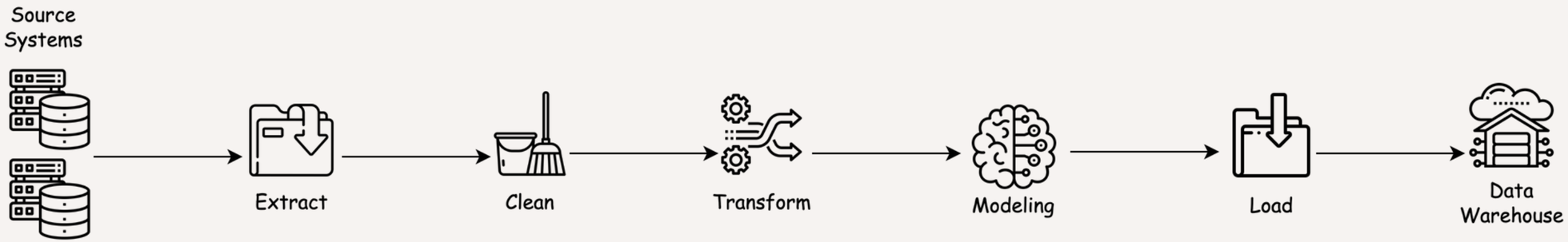
- Layer di **trasformation**: preparazione dei dati da passare al modello di intelligenza artificiale e calcolo dei KPI
- Layer di **modeling**: calcolo del *churn* di un utente
- Layer di **presentation**: elaborazione dei dati per rispettare gli schemi predefiniti e verifica della consistenza e dell'accuratezza delle predizioni

IL CASO DI STUDIO - OBIETTIVI

La mole di dati in aumento ha richiesto alcune modifiche alle pipeline attualmente in essere, che hanno **difficoltà a scalare** oltre un certo limite.

Gli obiettivi del progetto di tirocinio definiti sono i seguenti:

1. costruzione e schedulazione di una **pipeline ETL** scalabile che estragga, aggreghi e prepari il dato per modelli predittivi e ne appenda il risultato all'output finale
2. **refactoring** del codice esistente
3. orchestrazione tramite di **Apache Airflow**
4. **confronto delle prestazioni** tra la pipeline attualmente in produzione e quella rimaneggiata.



Indice

TECNOLOGIE UTILIZZATE

Amazon Web Services

Piattaforma di cloud computing proprietaria di Amazon

Apache Airflow

Piattaforma di schedulazione e monitoraggio di workflows batch-oriented

Altri stumenti

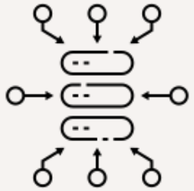
- Docker
- Python
- Pandas
- SQL
- AWS CDK
- Bitbucket
- ....

Indice

CODE REFACTORING

Creazione di asset comuni

Per aumentare la **manutenibilità** e ridurre la **dimensione** del codice, sono stati creati **asset comuni** alle pipeline per raggruppare logiche comuni ai flussi ETL.



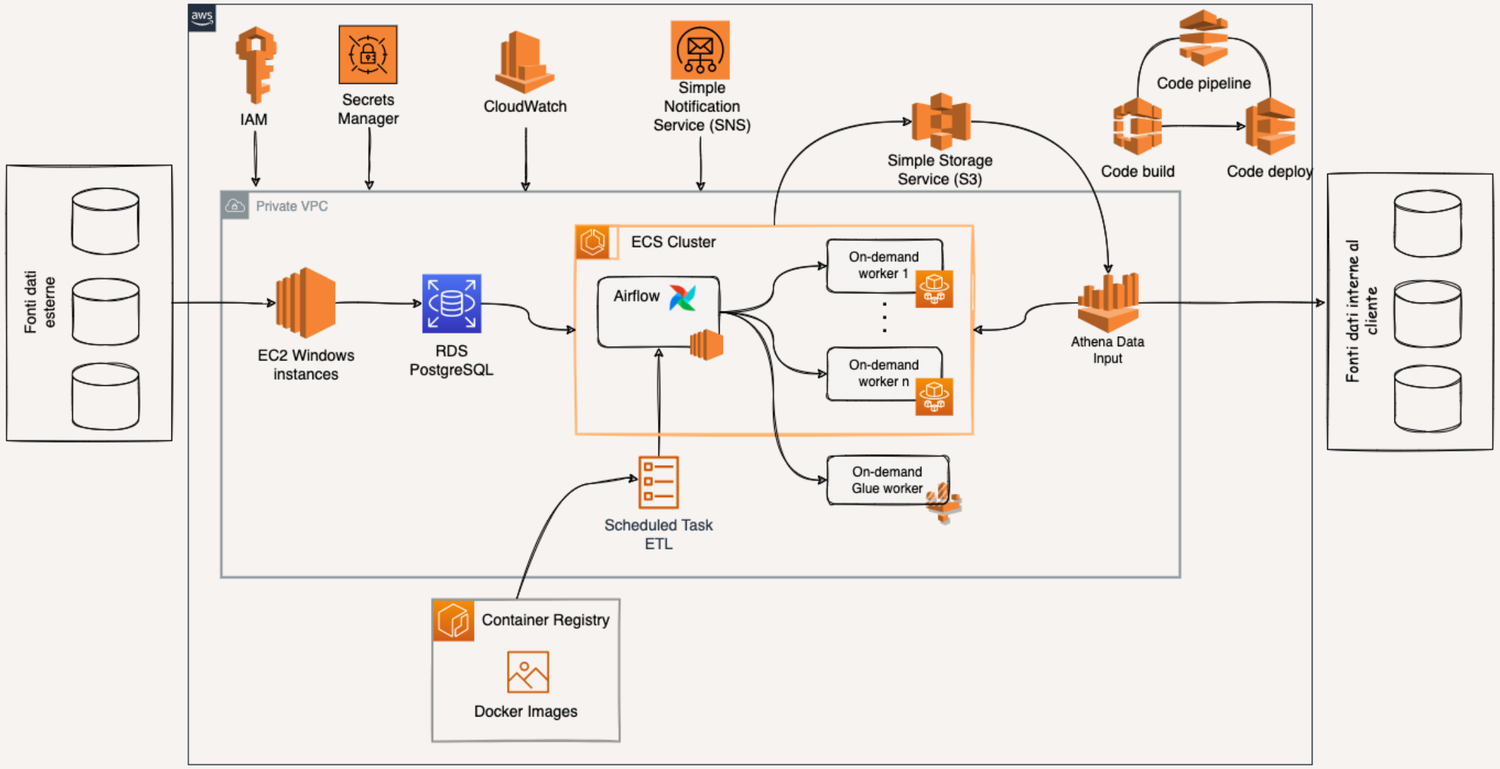
Aggregazione delle pipeline ETL

- **Aggregazione** delle pipeline con due livelli temporali: **mensile** e **annuale**
- Unione della pipeline di tipo *QV* e quella standard relativa allo stesso lasso temporale
- Parametrizzazione dei job

Creazione ambiente di sviluppo e produzione

Lo scopo di questa **separazione** è di permettere ai programmatori di sviluppare e rilasciare aggiornamenti dell'applicazione senza intaccare l'attuale versione in uso sugli utenti finali

ARCHITETTURA DELLE PIPELINE RIVISITE



Principali caratteristiche

- Architettura **cloud**
- Piattaforma **AWS**
- Presenza di un cluster ECS con:
  - Airflow in esecuzione su una macchina **EC2**
  - Creazione di istanze **Fargate** al bisogno per eseguire i job delle pipeline
- Utilizzo del **Container Registry** di AWS per memorizzare e versionare le immagini dei container Docker
- Database **PostgreSQL** per salvare i log e le informazioni di schedulazione delle pipeline

Indice

RISULTATI CODE REFACTORING

Valutazione della bontà del code refactoring sulla base di:

Halstead complexity

- Metrica statica
- Indipendenza del software dalla sua esecuzione su una specifica piattaforma
- Codice come sequenza di *token*, dove ogni token può essere un operatore o un operando
- La complessità del codice dipende da:
  - numero di operatori e operandi distinti
  - numero totale di operatori

RISULTATI CODE REFACTORING

Valutazione della bontà del code refactoring sulla base di:

Halstead complexity

- Metrica statica
- Indipendenza del software dalla sua esecuzione su una specifica piattaforma
- Codice come sequenza di *token*, dove ogni token può essere un operatore o un operando
- La complessità del codice dipende da:
  - numero di operatori e operandi distinti
  - numero totale di operatori

Cyclomatic complexity

- Metrica quantitativa
- Quanto più è elevato il numero di percorsi linearmente indipendenti all'interno di un modulo quanto più il codice è complesso
- La complessità deriva dal *control flow graph* e dipende da:
  - numero di archi del grafo
  - numero di nodi del grafo
  - numero di componenti connesse

RISULTATI CODE REFACTORING

Valutazione della bontà del code refactoring sulla base di:

Cognitive complexity

- Introdotta da SonarSource nel 2017
- Si basa sulle strutture presenti nel *control flow graph* del codice sorgente
- Tre regole fondamentali:
  - Ignora le strutture che consentono di abbreviare più dichiarazioni in una sola
  - Incrementa di 1 la complessità per ogni interruzione del flusso lineare del codice
  - Incrementa la complessità quando le strutture di interruzione del flusso sono annidate

RISULTATI CODE REFACTORING

Valutazione della bontà del code refactoring sulla base di:

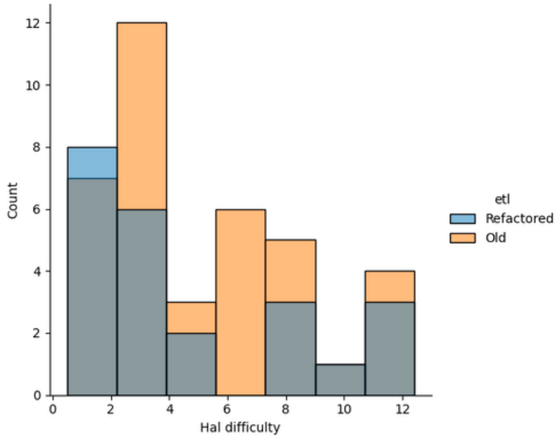
Cognitive complexity

- Introdotta da SonarSource nel 2017
- Si basa sulle strutture presenti nel *control flow graph* del codice sorgente
- Tre regole fondamentali:
  - Ignora le strutture che consentono di abbreviare più dichiarazioni in una sola
  - Incrementa di 1 la complessità per ogni interruzione del flusso lineare del codice
  - Incrementa la complessità quando le strutture di interruzione del flusso sono annidate

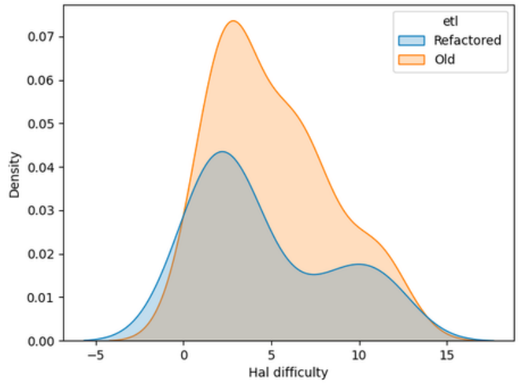
Metriche standard

- Linee di codice
- Blocchi di istruzioni
- Numero di funzioni
- Percentuale di blocchi duplicati
- ....

RISULTATI CODE REFACTORING- HALSTEAD COMPLEXITY



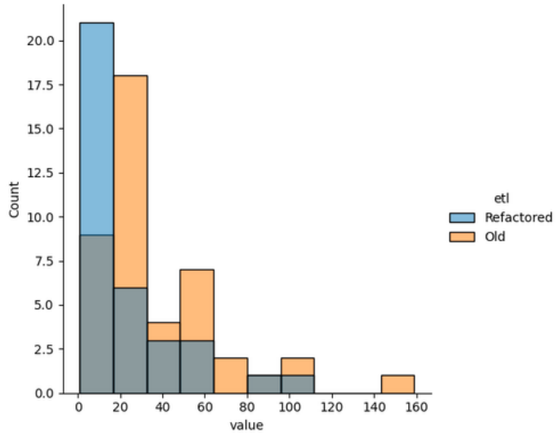
Distribuzione del valore di difficoltà di Halstead delle pipeline ETL



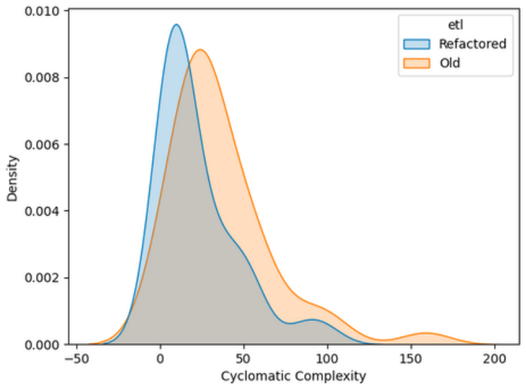
Kernel density estimation del valore di difficoltà di Halstead delle pipeline ETL

- **Riduzione** del numero di script a difficoltà elevata
- Valore di complessità nel punto di densità massima:
  - ETL rivisitata: **2.17**
  - ETL non rivisitata: **2.86**

RISULTATI CODE REFACTORING- CYCLOMATIC COMPLEXITY



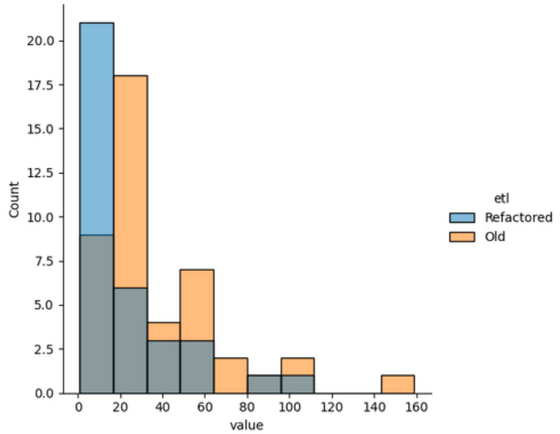
Distribuzione del valore di cyclomatic complexity delle pipeline ETL



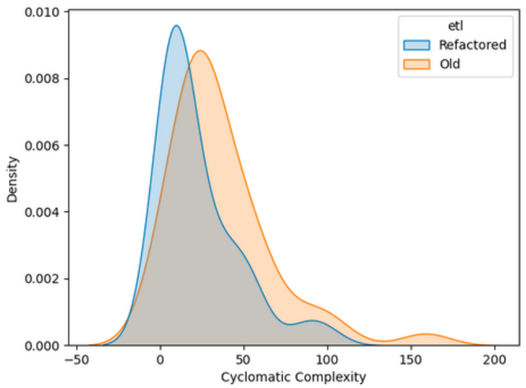
Kernel density estimation del valore di cyclomatic complexity delle pipeline ETL

- **Riduzione** del numero di script a difficoltà elevata
- Valore di complessità nel punto di densità massima:
  - ETL rivisitata: **21**
  - ETL non rivisitata: **36**
- **Eliminazione** del job con difficoltà maggiore di **140**

RISULTATI CODE REFACTORING- CYCLOMATIC COMPLEXITY



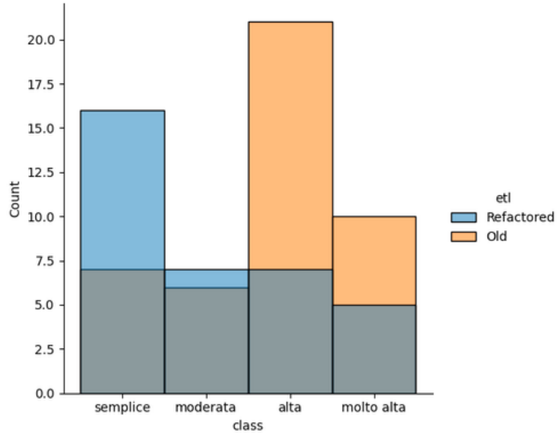
Distribuzione del valore di cyclomatic complexity delle pipeline ETL



Kernel density estimation del valore di cyclomatic complexity delle pipeline ETL

- **Riduzione** del numero di script a difficoltà elevata
- Valore di complessità nel punto di densità massima:
  - ETL rivisitata: **21**
  - ETL non rivisitata: **36**
- **Eliminazione** del job con difficoltà maggiore di **140**

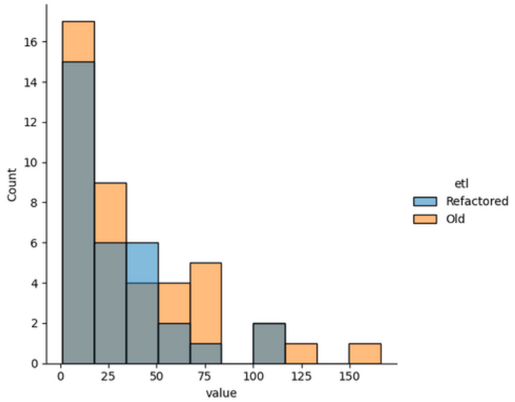
- **ETL rivisitata** mappa la maggior parte della complessità nella classe ***semplice***
- **ETL non rivisitata** contiene molti job a complessità ***alta***



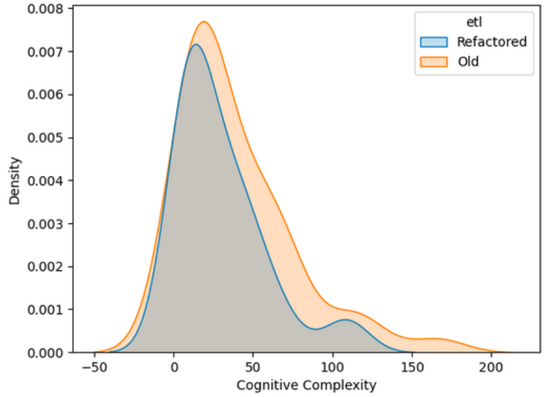
Distribuzione delle classi di complessità ciclomatica delle pipeline etl

Valore	Complessità
1 - 10	Semplice
11 - 20	Moderata
21 - 50	Alta
> 50	Molto alta

RISULTATI CODE REFACTORING- COGNITIVE COMPLEXITY



Distribuzione del valore di cognitive complexity delle pipeline ETL



Kernel density estimation del valore di cognitive complexity delle pipeline ETL

- Risultati analoghi alla cyclomatic complexity

RISULTATI CODE REFACTORING- METRICHE STANDARD

Numero di linee di codice

- ETL non rielaborata: **26586**
- ETL rielaborata: **9264**

↓ **- 65.2%**

Numero di file

- ETL non rielaborata: **101**
- ETL rielaborata: **64**

↓ **- 36.6 %**

Classi

- ETL non rielaborata: **14**
- ETL rielaborata: **8**

↓ **- 42.9 %**

Blocchi di istruzioni

- ETL non rielaborata: **12791**
- ETL rielaborata: **4328**

↓ **- 66.2%**

Percentuale di linee commentate

- ETL non rielaborata: **6.9%**
- ETL rielaborata: **12.9%**

↑ **+ 6%**

Blocchi duplicati

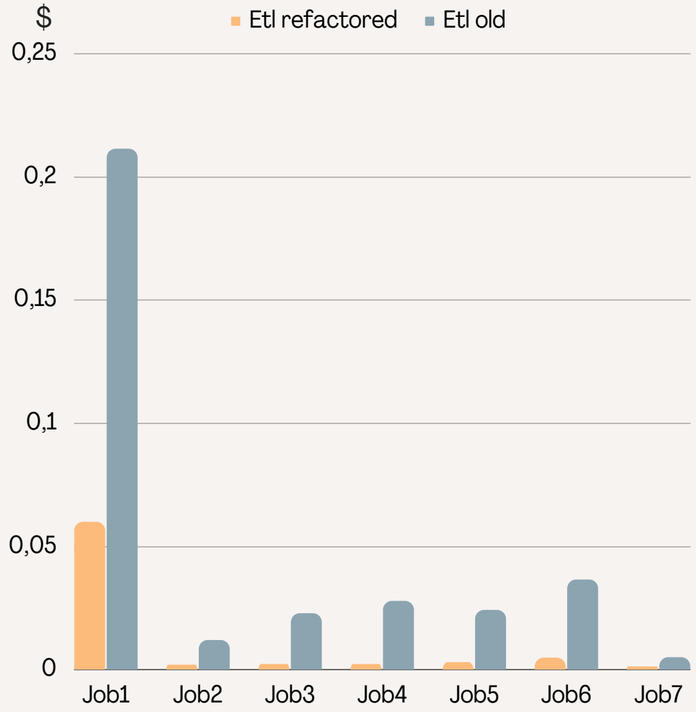
- ETL non rielaborata: **418**
- ETL rielaborata: **40**

↓ **-90.4 %**

RISULTATI - ETL MENSILE

Costo computazionale

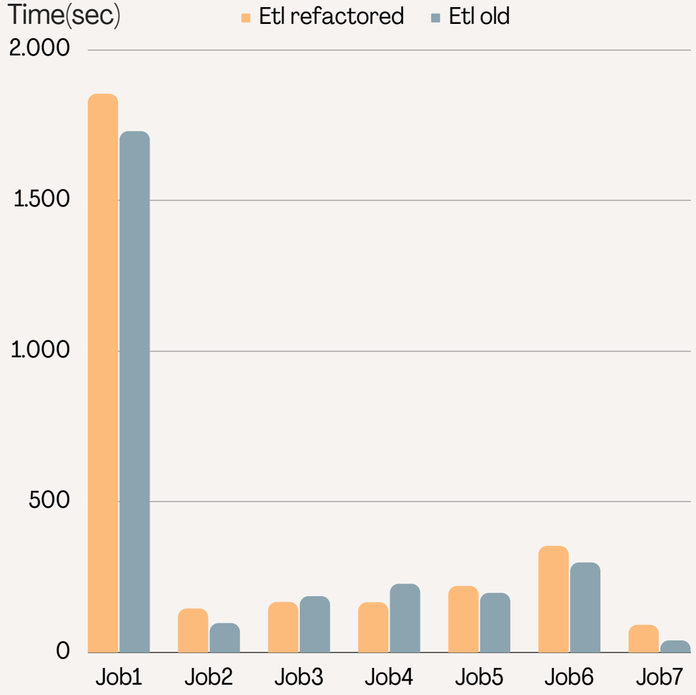
Riduzione del costo di esecuzione della pipeline ETL del **22.3** %



Confronto dei costi computazionali dei job della pipeline ETL mensile rivisitata e non rivisitata

Tempo di esecuzione

Aumento del tempo di esecuzione pari al **7.9** %



Confronto del tempo di esecuzione dei job della pipeline ETL mensile rivisitata e non rivisitata

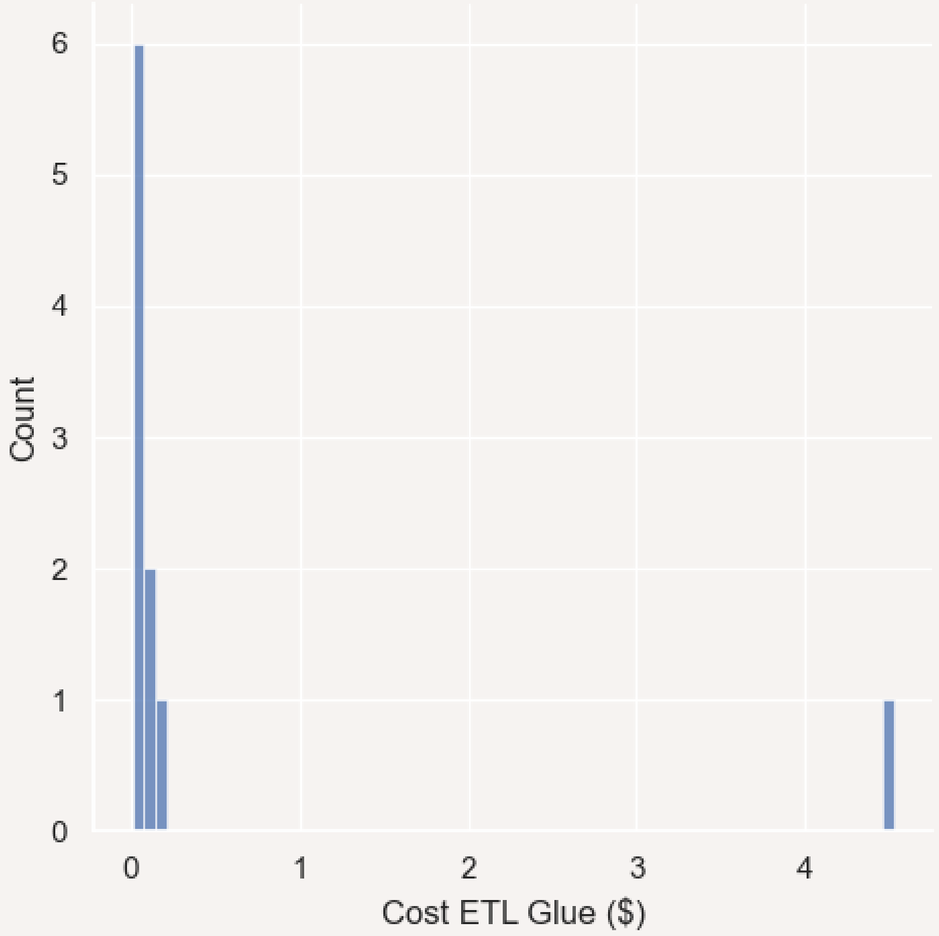
RISULTATI - ETL ANNUALE

Costo computazionale

- **Aumento** del costo di esecuzione della pipeline ETL del **1.3** %
- Dopo aver **parallelizzato** il secondo job di più oneroso, il costo di esecuzione è maggiore del costo di esecuzione della pipeline non rivisitata del **1.28** %

Tempo di esecuzione

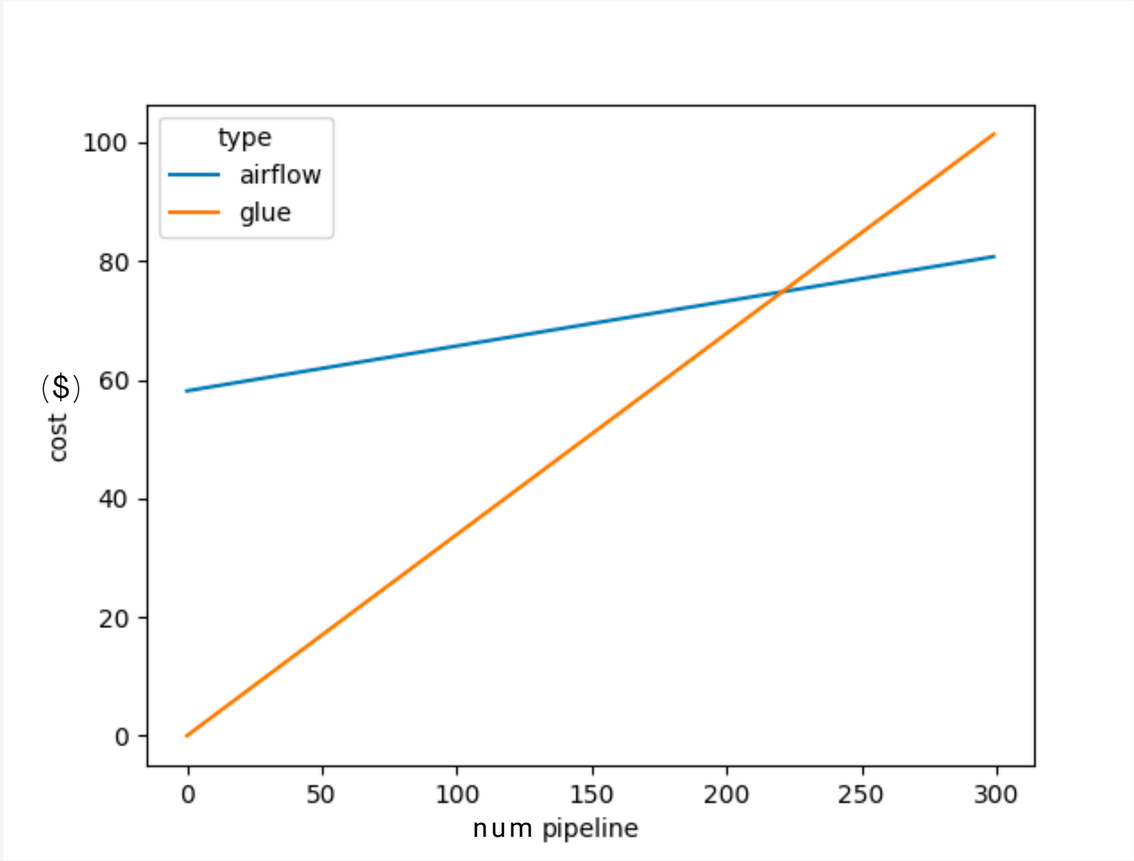
Riduzione del tempo di esecuzione pari al **15.8** %



Distribuzione del costo dei job della pipeline in esecuzione su Airflow

RISULTATI - COSTO DEI SERVIZI AWS

Quando conviene utilizzare l'architettura proposta in questo elaborato?



Servizio	Istanza	Costo all'ora
AWS RDS	1 vCPU e 1 GB di RAM	0.076 $
AWS EC2	4 vCPU e 16 GB di RAM	0.1664 $

Costo dei servizi (escluso Fargate)

- Numero di pipeline dopo il quale si ha un risparmio economico: **220**

RISULTATI - METRICHE NON QUANTITATIVE

Benefici code refactoring

- Creazione **ambiente di sviluppo**
- **Aggregazione** delle pipeline su due livelli temporali. In questo modo si è passati da gestire quattro pipeline differenti a due
- Creazione di **asset** comuni alle due pipeline

Benefici Apache Airflow

- E' altamente **dinamico**
- E' altamente **estendibile**
- E' altamente **scalabile**
- Progetto open source
- Permette il **versionamento dei DAG**
- Interfaccia grafica ricca di informazione
- E' **agnostico** rispetto gli strumenti
- Possibilità di **aumentare le capacità computazionali** scegliendo tra tante configurazioni possibili
- Monitoraggio delle schedulazioni più **granulare** rispetto a Glue

Benefici IAC

- **Versionamento dell'architettura**
- Configurazione dell'ambiente più rapida
- Maggiore **trasparenza**
- **Semplifica** la creazione di ambienti di test e di produzione
- Automatizzazione dei processi manuali

Indice

SVILUPPI FUTURI

- Utilizzo di Apache Spark come motore di elaborazione per la pipeline annuale
- Considerare la migrazione di ulteriori pipeline su Airflow prima di rilasciare le pipeline considerate in questo elaborato

Grazie per l'attenzione