ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

TESI DI LAUREA MAGISTRALE
in
Sistemi Distribuiti

Progetto e Valutazione di Pipeline di Big Data Analytics per Progetti di Sostenibilità

CANDIDATO
Lorenzo Piazza

RELATORE
Chiar.mo Prof. Paolo Bellavista

CORRELATORE
Dott. Ing. Alberto Cavalucci

Anno Accademico 2019/2020
Sessione IV

Sommario

01 Introduzione: Contesto e obiettivi

Agenda 2030 per lo Sviluppo Sostenibile

- Sottoscritta nel 2015 dai 193 Paesi membri dell'ONU
- 17 obiettivi per lo Sviluppo Sostenibile

"Rafforzare i mezzi di attuazione e
rinnovare il partenariato mondiale per lo sviluppo sostenibile"

(Agenda 2030, Goal 17 «Partnership for the Goals»)

Il progetto Hera SDG

- Nasce come progetto condiviso tra le aziende Hera – Conad – Camst
- *Smart Sustainable Community* con modello a ricompensa
- Perseguire gli Obiettivi di Sviluppo Sostenibile (SDG) dell'Agenda 2030

PARTNER	COMPORTAMENTO INCENTIVATO
GRUPPO HERA	Acquisto Energia Elettrica da fonti rinnovabili CO_2 risparmiata da Diario dei consumi Energia Elettrica Acquisto GAS metano CO_2 free CO_2 risparmiata da Diario dei consumi GAS metano Autolettura consumo gas Autolettura consumo acqua Segnalazione fuga acqua stradale Invio elettronico della bolletta Download my Hera Iscrizione ai servizi on line Attivazione alert autolettura
CONAD	Acquisto prodotti sostenibili recupero bottiglie di plastica per filiera bottle to bottle
CAMST	Acquisto piatti/menù sostenibili (carbon e water footprint + aspetti nutrizionali) Acquisto piatti last minute (take away) Raccolta differenziata on site sulle frazioni di rifiuto generate dal pasto

Obiettivi di Tesi

- Realizzare una Pipeline di Big Data Analytics che affianchi e arricchisca il progetto Hera SDG
- Supporto per decisioni *Data Driven*
- Tre compiti principali:

1. Raccolta e memorizzazione dei dati generati dalla Community
2. Data Visualization
3. ML analytics (e.g. *customer segmentation*)

02 Progetto e Realizzazione della Pipeline

Requisiti

1

Adattabilità ed estensibilità

2

Scalabilità

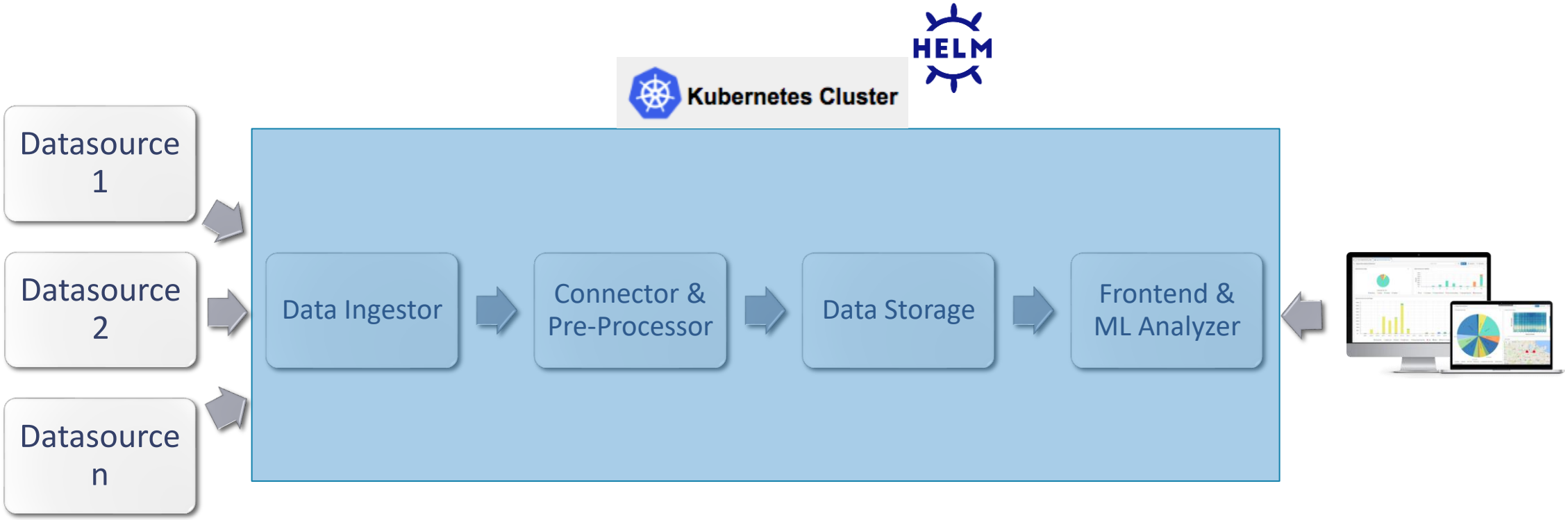
3

Robustezza e Resilienza dei dati

4

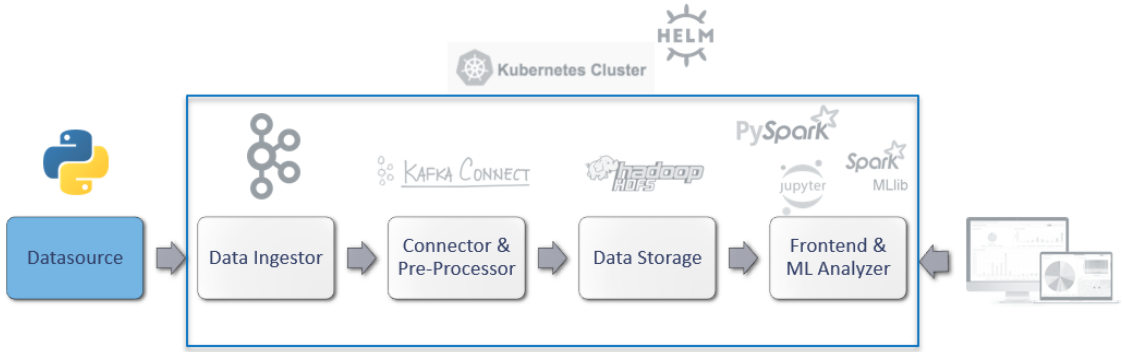
Open Source

Architettura della Pipeline



Componenti - (Mock) Datasource

- Python Kafka Producer
- Genera in modo pseudo-casuale 3 tipi di eventi:



Utente

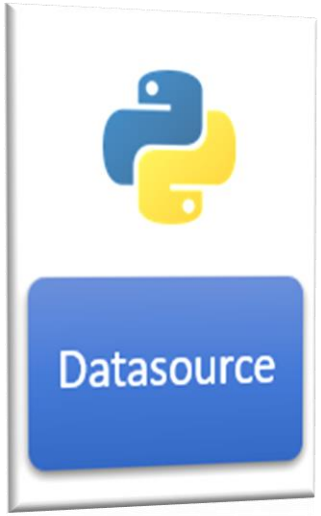
	id_utente	Sesso	Data di Nascita	Eta	Provincia
0	HE_6	M	1996-04-27	24	MO

Comportamento

comportamento	id_utente	Partner_erogante	reward(tk)
Invio elettronico della bolletta	CO_459	HERA	1.75

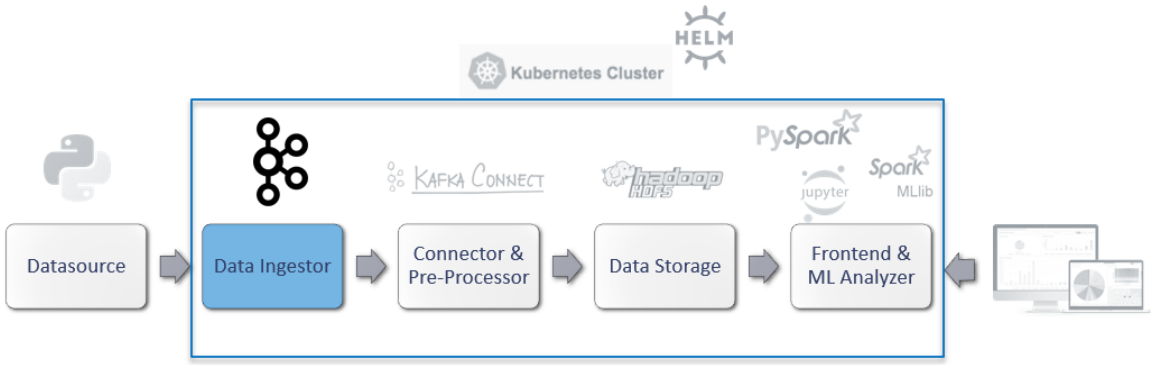
Premio

premio	id_utente	Partner_erogante	prezzo(tk)
Buono Spesa 5€ Conad	HE_123	CONAD	5



Componenti (2) - Data Ingestor

- Tecnologia utilizzata: Apache Kafka
- Cluster di *n = 2* broker (*n* configurabile)

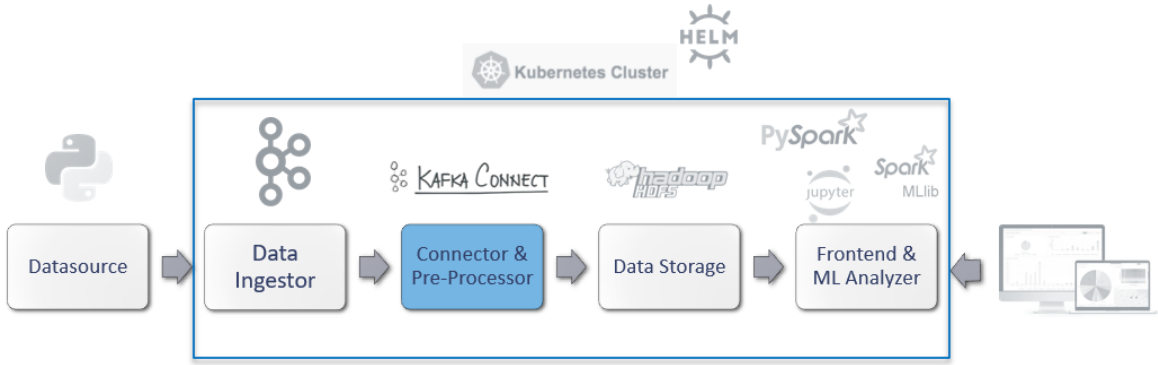


- Topic *partizionate*, *replicate* e *distribuite* su cui vengono pubblicati i record

- Requisiti soddisfatti dalla scelta tecnologica:
  - ✓ Resilienza
  - ✓ Scalabilità
  - ✓ Alto throughput
  - ✓ Open Source

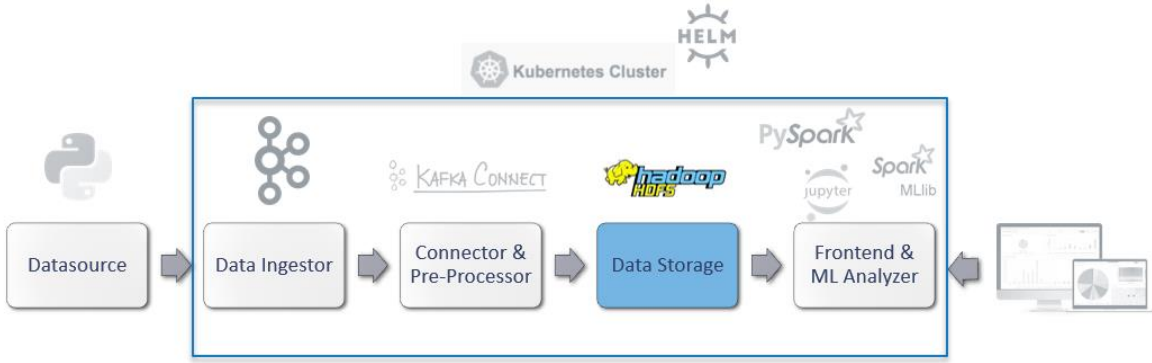
Componenti (3) - Connector & Pre-Processor

- Tecnologia utilizzata: Kafka Connect
- Worker che eseguono in modalità distribuita
- Leggono i record dalle topic Kafka e in modalità *batch* le scrivono sul Data Storage
- Operazioni di Pre-Processing (e.g. filtraggio)
- Requisiti soddisfatti dalla scelta tecnologica:
  - ✓ Adattabilità
  - ✓ Scalabilità



Componenti (4) - Data Storage

- Tecnologia utilizzata: HDFS
- Cluster di *n = 1* NameNode e *m = 2* DataNode
- Record memorizzati su file in formato JSON
- File *distribuiti* tra i DataNode e *replicati* con *replicationFactor* = 2 *(valore configurabile)*



- Requisiti soddisfatti dalla scelta tecnologica:
  - ✓ Resilienza
  - ✓ Scalabilità
  - ✓ Alto throughput
  - ✓ Open Source

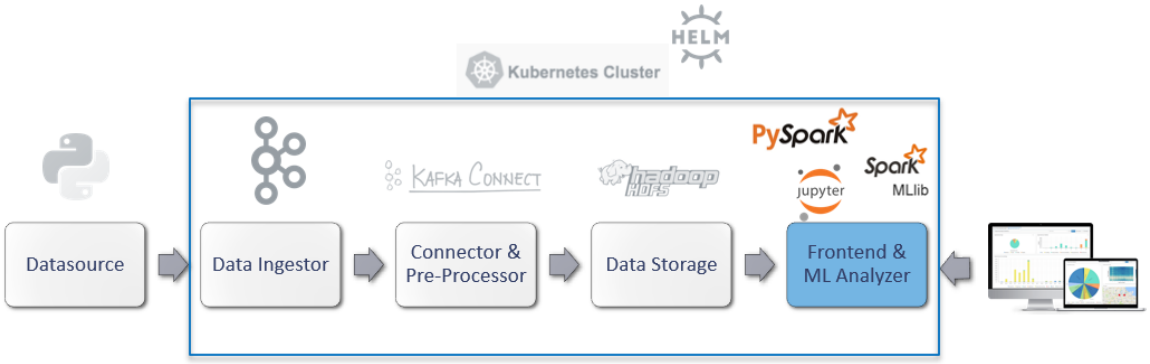
Componenti (5) - Frontend & ML Analyzer

- Tecnologie utilizzate:

  Frontend: Jupyter Lab

  Processamento e ML: Apache Spark

  PySpark, Spark MLlib, SparkSQL

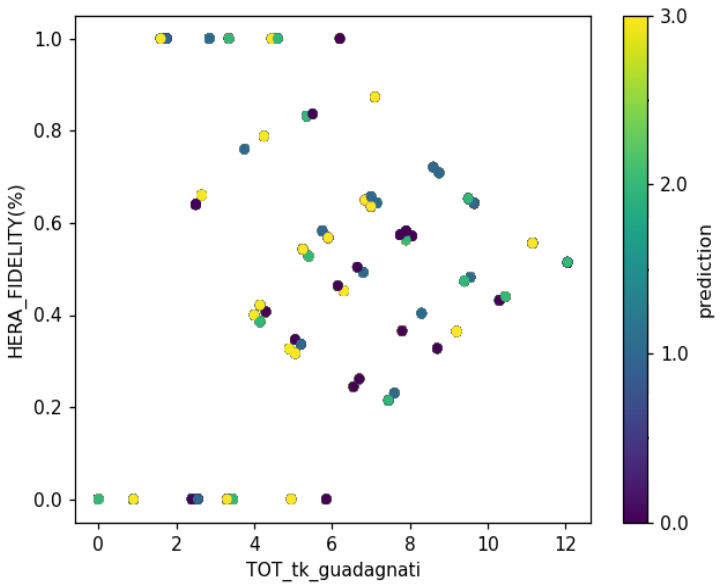


- Requisiti soddisfatti dalle scelte tecnologiche:
  - ✓ Estensibilità
  - ✓ Resilienza
  - ✓ Scalabilità
  - ✓ Velocità di processamento
  - ✓ Open Source

Componenti (6) - Frontend & ML Analyzer

Feature offerte:

1. Creazione Spark Context
2. Lettura Dati da HDFS
3. Processamento e preparazione al clustering
4. Data Visualization
5. ML Clustering (K-Means || ed Elbow Method)



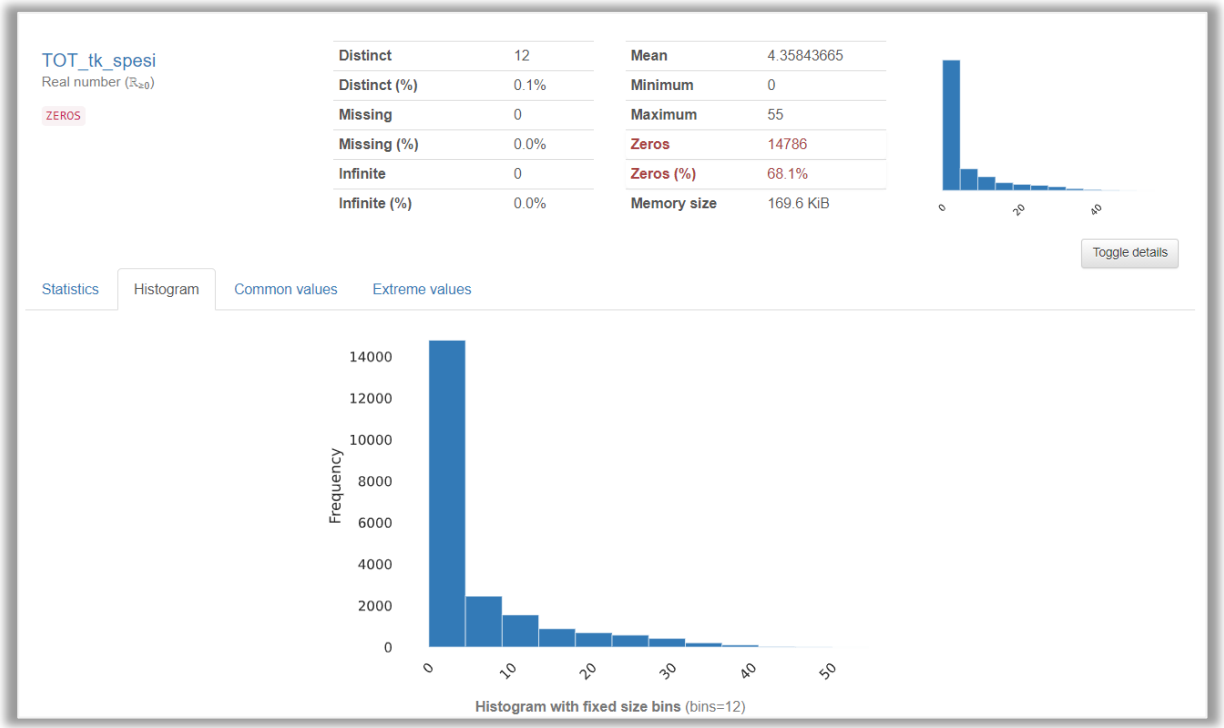


Figura A – Un esempio di Insight prodotto dalla libreria Pandas_profiling.

Figura B – Uno scatter plot ottenuto a seguito della Customer Segmentation.
I diversi colori rappresentano l'appartenenza degli utenti a diversi segmenti.

03 Deploy e Performance

Pod Affinity e Anti-Affinity


```
130   # ensure the scheduler does not co-locate replicas (brokers) on a single node.
131   affinity:
132     podAntiAffinity:
133       preferredDuringSchedulingIgnoredDuringExecution:
134       - weight: 1
135         podAffinityTerm:
136           labelSelector:
137             matchExpressions:
138             - key: app.kubernetes.io/component
139               operator: In
140               values:
141               - kafka
142           topologyKey: "kubernetes.io/hostname"
```

Figura A – Regola di Pod AntiAffinity per il componente Data Ingestor.

- Regole K8s con cui è possibile esprimere dei vincoli sul deploy dei componenti

```
$ kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP             NODE
my-hdfs-datanode-0                  1/1     Running   32         33h   10.244.1.15    piazza-2
my-hdfs-datanode-1                  1/1     Running   23         33h   10.244.0.15    piazza-1
my-hdfs-namenode-0                  1/1     Running   50         33h   10.244.1.14    piazza-2
my-hdfs-shell-84768f96cc-zlvkh      1/1     Running   0          33h   10.244.1.13    piazza-2
my-kafka-0                          1/1     Running   0          13m   10.244.0.19    piazza-1
my-kafka-1                          1/1     Running   0          13m   10.244.1.22    piazza-2
my-kafka-connect-69488b47bd-tn2jl   1/1     Running   0          12m   10.244.1.23    piazza-2
```

Figura B – Risultato del deploy.

Auto Scaling del "Pre-Processor & Connector"

- Scaling up & down automatico del componente «Pre-Processor & Connector» basandosi sul suo utilizzo di CPU
- Utilizzo di un Horizontal Pod Autoscaler (HPA)

```
Lorenzo@DESKTOP-QAC8QT9 MINGW64 ~/Documents/LORI/UNIVERSITA'/Magistrale/Tesi/HeraSDG-Big
$ kubectl top pods my-kafka-connect-cdc56bf8-wtvzw
NAME                                CPU(cores)   MEMORY(bytes)
my-kafka-connect-cdc56bf8-wtvzw     21m          1307Mi

Lorenzo@DESKTOP-QAC8QT9 MINGW64 ~/Documents/LORI/UNIVERSITA'/Magistrale/Tesi/HeraSDG-Big
$ kubectl get hpa
NAME              REFERENCE                         TARGETS    MINPODS   MAXPODS   REPLICAS
my-kafka-connect  Deployment/my-kafka-connect       10%/80%    1         3         1
```

Figura A – Il Pod gestito dall'HPA presenta un consumo di CPU sottosoglia

```
Lorenzo@DESKTOP-QAC8QT9 MINGW64 ~/Documents/LORI/UNIVERSITA'/Magistrale/Tesi/HeraSDG-Big
$ kubectl get hpa
NAME              REFERENCE                         TARGETS    MINPODS   MAXPODS   REPLICAS
my-kafka-connect  Deployment/my-kafka-connect       149%/80%   1         3         1

Lorenzo@DESKTOP-QAC8QT9 MINGW64 ~/Documents/LORI/UNIVERSITA'/Magistrale/Tesi/HeraSDG-Big
$ kubectl get pods
NAME                                READY   STATUS     RESTARTS   AGE
kafkacat-debugger                   1/1     Running    0          6d2h
my-hdfs-datanode-0                  1/1     Running    32         7d12h
my-hdfs-datanode-1                  1/1     Running    31         7d12h
my-hdfs-namenode-0                  1/1     Running    52         7d12h
my-hdfs-shell-84768f96cc-zlvkh      1/1     Running    0          7d12h
my-jupyter-jupyter-0                1/1     Running    0          3d10h
my-kafka-0                          1/1     Running    0          63m
my-kafka-1                          1/1     Running    1          63m
my-kafka-connect-cdc56bf8-lrvdz     0/1     Init:0/1   0          18s
my-kafka-connect-cdc56bf8-wtvzw     1/1     Running    0          19m
```

Figura B – L'utilizzo di CPU al 149% causa la creazione di un secondo my-kafka-connect

Spark on K8s: Dynamic Resource Allocation

- Creazione di ulteriori executor Spark quando il carico aumenta
- Rilascio delle risorse dopo un periodo di inattività
- Gestione efficiente delle risorse per operazioni di analisi dei dati *on-demand*

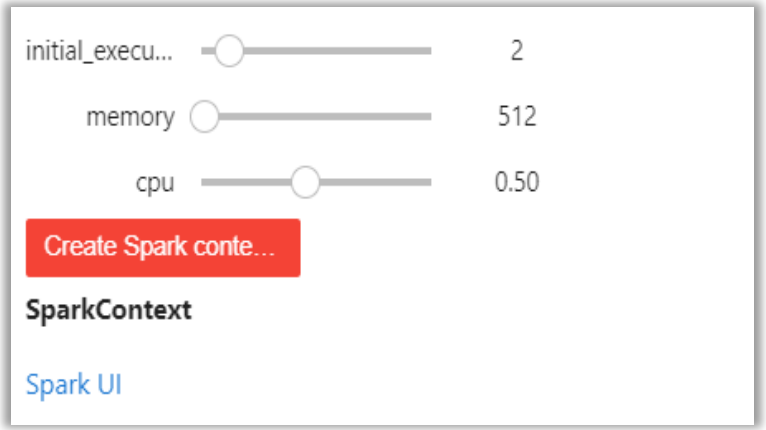


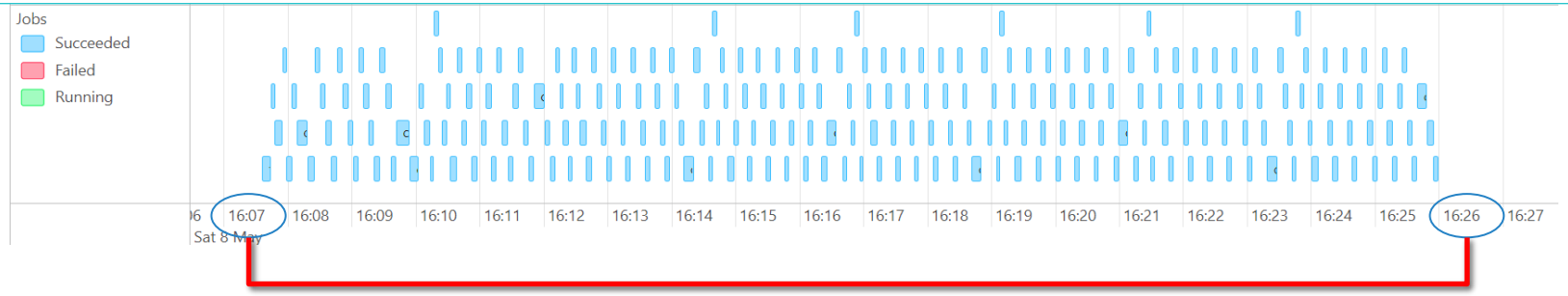
Figura A – Widget del Frontend con cui richiedere la creazione degli executor Spark iniziali

```
################ DYNAMIC ALLOCATION ################
conf.set("spark.dynamicAllocation.enabled", "true")
conf.set("spark.dynamicAllocation.shuffleTracking.enabled", "true")
# remove policy
conf.set("spark.dynamicAllocation.executorIdleTimeout", "1800s")
conf.set("spark.dynamicAllocation.cachedExecutorIdleTimeout", "3600s")
# request policy:
conf.set("spark.dynamicAllocation.initialExecutors", initial_executor)
conf.set("spark.dynamicAllocation.minExecutors", 1)
conf.set("spark.dynamicAllocation.maxExecutors", 4)
conf.set("spark.dynamicAllocation.schedulerBacklogTimeout", "1s")
conf.set("spark.dynamicAllocation.sustainedSchedulerBacklogTimeout", "1s")
```

Figura B – Codice di configurazione della Dynamic Resource Allocation

Spark on K8s: Dynamic Resource Allocation (2)

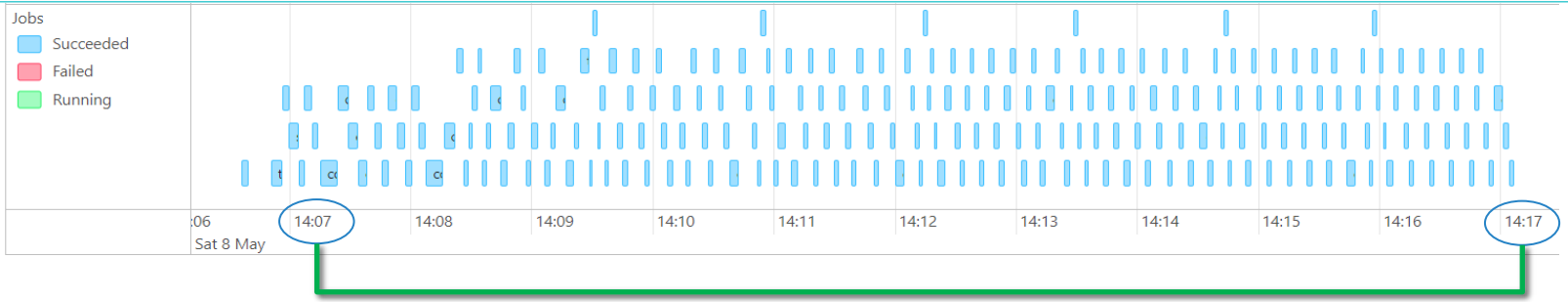
ML Clustering (K-Means con Elbow Method) condotto con **1 executor Spark**



Dynamic Allocation disabilitata ✖

~19 min.

ML Clustering (K-Means con Elbow Method) condotto con **1 executor Spark iniziale e scaling fino a 5**



Dynamic Allocation abilitata ✓

~10 min.

Figura - Risultati del test osservati con la Spark UI.
I riquadri azzurri rappresentano i Job Spark che sono stati eseguiti.

04 Conclusioni e Prospettive future

Conclusioni e sviluppi futuri

- Soddisfatti i principali requisiti per garantire una gestione del dato efficiente ed efficace
- Robusta ad eventuali cambi nei requisiti del progetto Hera SDG
- Pipeline come importante base di partenza per la realizzazione di ulteriori soluzioni

- Sviluppi futuri:
    - Necessari ulteriori test con carichi adeguati prima di dichiarare la pipeline *production ready*
    - Aggiunta di ulteriori modelli di ML
    - Automatizzazione dei processi decisionali

Grazie per l’attenzione!