# ***SABD-PROJECT1***

## Introduzione

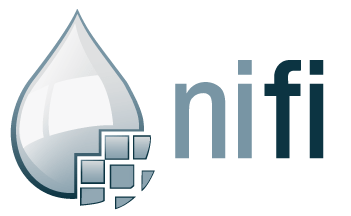
L’obbiettivo di questo report è quello di presentare l’implementazione ed i risultati ottenuti nella realizzazione del progetto di Sistemi e Architetture per i Big Data dell’anno 2020-2021.

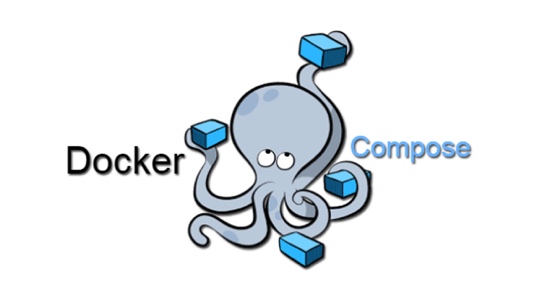
Per lo sviluppo di questo progetto è stato utilizzato il framework Spark per analizzare i dati presi dalla repository github delle vaccinazioni in Italia, per contrastare il diffondersi della pandemia dovuta al virus Covid-19, in modo tale da poter rispondere alle seguenti domande:

* Query1: calcolare della media delle vaccinazioni per un generico centro per ciascuna regione e ciascun mese;
* Query2: individuare per ciascun mese e ciascuna fascia d’età le prime cinque regioni per le quali si prevedeva il maggior numero di vaccinati di sesso femminile;
* Query3: stimare per ciascuna regione il numero totale di somministrazioni effettuate il 1 giugno 2021 e classificare tali risultati attraverso un algoritmo di clustering K-means e Bisecting K-means.

Il codice che è stato realizzato può essere reperito in due versioni differenti: la prima all’interno del package src/main/java/queries e la seconda all’interno del package src/main/java/sql\_queries, nella quale è stata utilizzata la versione SQL di Spark.

## Architettura del sistema



### Docker compose

Docker compose è uno strumento per la definizione e l’esecuzione di applicazioni Docker multi-containers. Si utilizza un file “docker-compose.yml” per configurare i servizi necessari all’esecuzione del progetto in modo tale che questi possano eseguire contemporaneamente e in maniera isolata. Con un solo comando si è in grado di creare e avviare tutti i servizi specificati all’interno della configurazione, avendo in più la possibilità di specificare il numero di spark-worker e datanode che si vuole andare ad utilizzare.

Nel nostro caso particolare, i containers che andiamo ad istanziare sono:

* Spark-master/spark-worker
* Namenode/Datanode
* Redis
* Nifi

### Apache Spark

Apache Spark è un framework open source per il processamento dei Big Data, che mette a disposizione API di alto livello per numerosi linguaggi di programmazione e tool di alto livello come SparkSQL.

La scelta di utilizzare tale framework rispetto a MapReduce è stata dettata dalla dimensione del dataset, poiché essendo abbastanza piccolo è possibile contenerlo tutto in memoria traendo dei vantaggi nel processamento rispetto al salvataggio su disco. Inoltre, si è deciso di utilizzarlo per il maggior numero di funzioni messe a disposizione rispetto alle sole due map e reduce.

### Apache Nifi

Apache NiFi è un tool affidabile e potente progettato per automatizzare il flusso di dati tra i sistemi software. Per tali ragioni, nel nostro progetto è stato utilizzato per andare a gestire il routing dei dati fra i conteiners istanziati: prendere i dati da Github e metterli su HDFS, prendere i risultati delle query da HDFS per caricarli su Redis.

La scelta di tale sistema è stata dettata dall’interfaccia grafica semplice e intuitiva e dalla buona documentazione reperibile online.

### Apache Hadoop

Apache Hadoop è un framework open source che permette il processamento distribuito di grandi quantità di dati all’interno di cluster di computer utilizzando un modello di programmazione semplice.

Nel nostro progetto tutti i dati per la configurazione di Spark vengono memorizzati grazie al suo utilizzo.

### Redis

Redis è un key-value store open source residente in memoria, adatto per la memorizzazione veloce dei dati. Nel nostro progetto è stato utilizzato per esportare tutti i dati delle query.

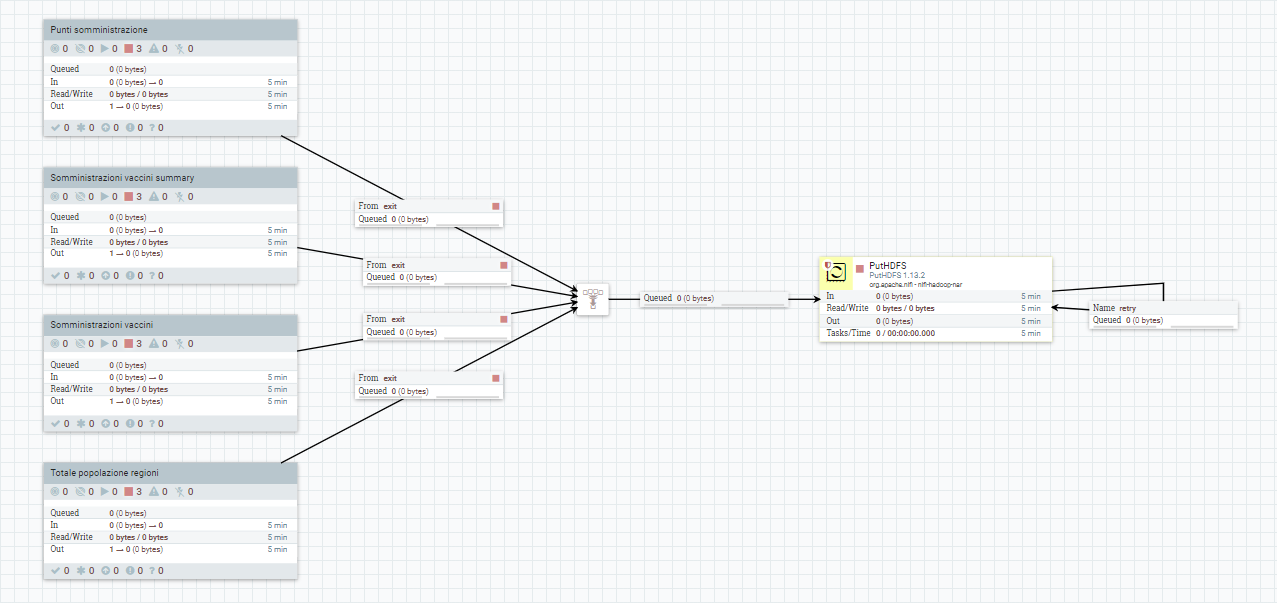
## Implementazione

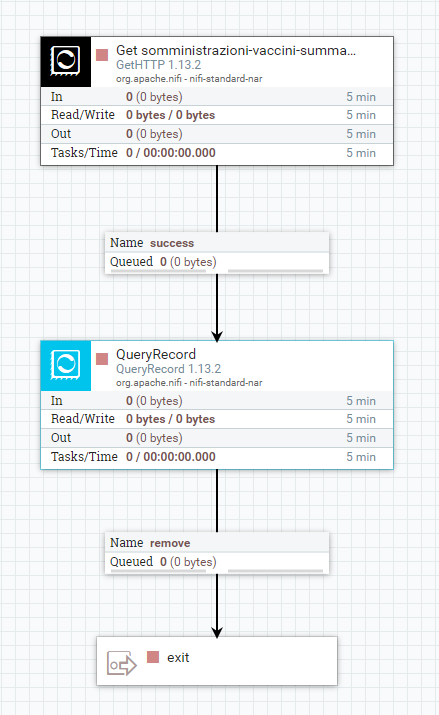
### Query

### Nifi

Per l’implementazione di tale framework sono stati utilizzati due templates.

#### **Input.xml**

Prende i dati dalla repository github e fa l’injection in hdfs. È stato creato un process group per ogni singolo file che deve essere importato. In questo viene creato un flow file che è stato convertito da formato csv a formato parquet e sul quale viene ad essere realizzata una “query” per effettuare il pre-processamento dei dati.



Come esempio di pre-processamento dei dati riportiamo l’immagine relativa al file “somministrazioni-vaccini-summary.csv” nel quale è stato affettuato anche un ordinamento dei dati in base alla data, come richiesto dalla specifica del progetto.

#### **Redis.xml**

Prende i dati da hdfs, dove sono stati precedentemente memorizzati i risultati delle query, e li salva su redis.

Immagine che contiene testo, parete, interni, toeletta

Descrizione generata automaticamente

Immagine che contiene testo, screenshot, interni, parecchi

Descrizione generata automaticamente

Nel particolare nel process group di redis quello che si fa è:

* Prendere i risultati delle query
* Cambia il nome dei risultati ponendo davanti il numero della query corrispondente
* Mette davanti al nome anche la data odierna

## Deployment

Come specificato precedentemente per sviluppare il progetto è stato utilizzato Docker-compose, è possibile quindi avviare l’applicazione andando ad eseguire contemporaneamente tutti i servizi, in maniera isolata e specificando anche il numero di worker nodes che si vuole, rendendo il tutto più scalabile. Per fare ciò viene utilizzato il comando

docker-compose up --scale spark-worker=3 --scale datanode=4

e.g., cluster con 3 spark worker e 4 hdfs datanode.

Al primo deployment del cluster, inoltre, è necessario, per poter andare ad utilizzare nifi, importare i templates, sopra citati, memorizzati nel folder /nifi/templates.

Submit-query

Per effettuare il submit delle query è stato realizzato uno script che prende come parametri il numero della query che si intende andare ad eseguire e, nel caso della query 3 e sql\_query3, l’algoritmo da utilizzare e il numero di cluster. Per tanto è sufficiente eseguire i seguenti comandi:

* mvn package: crea il file jar
* sh submit\_query.sh <numero della query> <eventuali parametri>

## Test

I test sono stati svolti utilizzando un processore intel core i-7 3770. È stata presa in considerazione un’architettura composta da due datanode hdfs e numero variabile di spark worker da 1 a un massimo di 4. Per le due versioni della query 3 è stato, inoltre, testato anche il caso in cui il cluster variasse da 2 a un massimo di 5.

Di seguito vengono riportati i risultati dei test.

Dai grafici riportati quello che possiamo vedere è che all’aumentare dei nodi worker in esecuzione il tempo di processamento aumenta. Ciò potrebbe essere dovuto al fatto che il dataset che stiamo analizzando, nonostante sia trattato come un dataset di grandi dimensioni, alla fine non risulta essere effettivamente così.