Guida operativa per la riproduzione del progetto.

Nella cartella **my-data** devono essere presenti:

* Il flow di nifi, *flow.xml.gz*
* I file per la configurazione di hbas*e hbase-site.xml, core-site.xml*
* Cartella *meteo* e cartella *aria,* contenenti i file csv dei dati
* Gli script python

Per prima cosa va sistemato il container di **jupyter-lab**. Accedendo al container vanno copiati gli script dalla cartella ‘/data/my-data/’ alla cartella ‘/home/jovyan/work’. A questo punto va fatto girare lo script *Reference\_tables* che produrrà i file csv con le info dei sensori di meteo e aria per la prima fase di arricchimento in nifi.

Si tornerà nel container di *jupyter-lab* per lo streaming dei dati.

*A questo punto va sistemato***hbase***, i file di configurazione nella cartella servono a nifi per configurare la connessione e sono già nel posto giusto. Serve però creare la tabella in cui verranno salvati i dati meteo per l’arricchimento in streaming dei dati del flusso dell’aria.*

*Si accede al container ‘hbase-master’* , si lancia il comando ‘hbase shell’ e si crea la tabella col comando ‘create ‘meteo’,’misurameteo’ ’. Tabella con un’unica column family.

Adesso è il momento di **Nifi** , va caricato il flow in modo da poterlo usare. Dopo essere acceduti al container si lancia il comando ‘cp /data/my-data/flow.xml.gz -t ./conf/ ’ .

Per vedere l’interfaccia di Nifi ci si collega da browser al localhost porta 8080 (<http://localhost:8080/nifi/>).

Una volta attivati tutti i nodi di nifi basterà lanciare all’interno del container di jupyter-lab gli script Kafka\_producer\_aria e Kafka\_producer\_meteo, avendo cura di selezionare l’anno da usare.

Se si usano gli script il comando è :

* python 3 Kafka\_producer\_meteo.py ‘2019’

per far partire lo stream meteo del 2019. Lo stesso vale per l’aria e per altri anni.

Se invece si usa il notebook bisognerà collegarsi alla UI di jupyter e lanciare i notebook modificando l’anno tra le variabili.

**NOTA**: Sono state caricate sia la versione script, che la versione notebook perchè l’ultima permette di mostrare e spiegare meglio il procedimento che ha portato alla selezione preliminare dei sensori e allo streaming, mentre lo script è quello utilizzato per produrre il dataset effettivo e che risulta in un uso più efficiente sulla VM.

**NOTA2**: I dati del 2020 dell’aria sono disponibili solo tramite API, di conseguenza è stato caricato uno script apposta per i dati aria 2020, e gli stessi non sono presenti nella cartella aria.

**Procedura MongoDB 1.5 [Finale]**

1. Esportare i file JSON da mongo con i comandi seguenti da shell:

mongoexport --collection=rawdata2020 --db=progetto --out=rawdata2020new.json

mongoexport --collection=rawdata2019 --db=progetto --out=rawdata2019new.json

mongoexport --collection=rawdata2018 --db=progetto --out=rawdata2018new.json

mongoexport --collection=rawdata2017 --db=progetto --out=rawdata2017new.json

mongoexport --collection=rawdata2016 --db=progetto --out=rawdata2016new.json

Importare le collezioni da shell eseguire i seguenti comandi:

mongoimport --db progetto --collection rawdata --drop --file "/path/rawdata2020new.json"

mongoimport --db progetto --collection rawdata --file "/path/rawdata2019new.json"

mongoimport --db progetto --collection rawdata --file "/path/rawdata2018new.json"

mongoimport --db progetto --collection rawdata --file "/path/rawdata2017new.json"

mongoimport --db progetto --collection rawdata --file "/path/rawdata2016new.json"

1. Accedere alla shell con:

mongo

1. Accedere al db d’interesse:

use progetto

1. Modificare le date in formato ISO:

db.rawdata.aggregate([{ $addFields: {"Date": {$dateFromString : {dateString: '$Data',"format": "%d/%m/%Y %H:%M:%S", "timezone" : "Europe/Rome"}}}}]).forEach(function(x){db.rawdata.save(x)})

questo procedimento richiederà parecchio tempo se non disponete di un pc particolarmente potente.

1. Ora che le date sono sistemate rimuoviamo i documenti che non sono di nostro interesse selezionando le date dal luglio a dicembre.

db.rawdata.remove({"Date" : {$gte : new Date(2020, 07, 01), $lte : new Date(2020, 12, 31)}})

db.rawdata.remove({"Date" : {$gte : new Date(2019, 07, 01), $lte : new Date(2019, 12, 31)}})

db.rawdata.remove({"Date" : {$gte : new Date(2018, 07, 01), $lte : new Date(2018, 12, 31)}})

db.rawdata.remove({"Date" : {$gte : new Date(2017, 07, 01), $lte : new Date(2017, 12, 31)}})

db.rawdata.remove({"Date" : {$gte : new Date(2016, 07, 01), $lte : new Date(2016, 12, 31)}})

1. aggiungere campo luogo in modo da avere un indirizzo con ‘paese’+(provincia), aggiungere campo SensorIdAddress per viz

db.rawdata.aggregate([{$addFields:{'luogo': {$concat: ["$comune" ," (", "$provincia", ")"]}}}]).forEach(function(x){db.rawdata.save(x)})

db.rawdata.aggregate([{$addFields:{'luogo': {$SensorIdAddress: ["$IdStazione", "$luogo",]}}}]).forEach(function(x){db.rawdata.save(x)})

questo procedimento richiederà parecchio tempo se non disponete di un pc particolarmente potente.

1. Creare un campo unico per la geolocalizzazione

db.rawdata.aggregate( [ { $addFields: { "GeoLoc": { "type": "Point", coordinates: [ { $toDecimal: "$lng"}, { $toDecimal: "$lat"}]}}}]).forEach(function(x){db.rawdata.save(x)})

con il seguente codice viene indicizzata anche la

db.rawdata.createIndex( { GeoLoc: "2dsphere" } )

questo procedimento richiederà parecchio tempo se non disponete di un pc particolarmente potente.

1. Droppare i campi non più necessari.

db.rawdata.updateMany({}, {$unset : {"idOperatore" : 1}})

db.rawdata.updateMany({}, {$unset : {"Stato" : 1}})

db.rawdata.updateMany({}, {$unset : {"Data" : 1}})

1. Cambiare i nomi dei vari campi con:

db.rawdata.updateMany( {}, { $rename: { "IdSensore": "SensorId", "nometiposensore": "SensorType", "Valore": "Value", "unitamisura": "UnitOfMeasure", "idstazione" : "StationId", "quota": "Altitude", "meteo\_station": "MeteoStationId", "luogo": "Address", "Precipitazioni": "Precipitations" } } )

1. Uscire da mongo con:

exit

1. Esportare il db con

mongoexport --collection=rawdata --db=progetto --out=finaldb.json

1. Se necessario, mportare il file finale con:

mongoimport --db progetto --collection finaldata --drop --file "/path/finaldb"

1. Una volta effettuato l’accesso al db (vedi punti 2 e 3) eseguire le query per valutare la performance del db

db.rawdata.find( {GeoLoc: {$geoWithin: { $centerSphere: [ [ 9.17785942554474, 45.46844904627238 ], 0.002585599467882236 ]}},SensorType: 'Biossido di Azoto',SensorId: {$in: [ '5601', '5603', '5798', '5549', '6705', '5534', '5845', '5853', '9924', '9969', '5551', '5574', '5577', '5595', '5790', '5805', '5809', '5826', '5828', '5834', '5842', '5844', '5847', '6701', '6859', '6862', '6877', '9802', '9877', '9928']},Value: {$gte: 10,$lt: 20},StationId: {$in: [ '572', '574', '576', '598', '600', '627', '542', '546', '548', '583', '661']},Date: { $gte: ISODate('2020-01-01T00:00:00.000Z'), $lte : ISODate('2020-06-30T07:23:59.000Z')} }).explain("executionStats")

db.rawdata.find({$or: [{ GeoLoc: { $geoWithin: { $centerSphere: [ [ 9.17785942554474, 45.46844904627238 ], 0.002585599467882236 ] } }},{ GeoLoc: { $geoWithin: { $geometry: { type: 'Polygon', coordinates: [ [ [ 9.011621475219728, 45.2637627853291 ], [ 9.820210933685305, 45.3111555858443 ], [ 9.640738964080812, 45.65650362767964 ], [ 8.77985715866089, 45.64615959527059 ], [ 9.011621475219728, 45.2637627853291 ] ] ] } } }}],SensorType: 'Biossido di Azoto',Value: {$gte: 10,$lt: 20},Date: {$gte: ISODate('2020-01-01T12:30:00.000Z'), $lte: ISODate('2020-12-31T23:59:00.000Z')},Altitude: {$in: [ '114', '207', '206', '80', '191', '118']},}).explain("executionStats")