



UNIVERSITÀ DI PISA

Master Universitario di I Livello in Cybersecurity



CyberIntelligence: esercitazione su NiFi

Tiziano Fagni

IIT,CNR

tiziano.fagni@iit.cnr.it

Soluzione docker Nifi + ES

<https://github.com/tizfa/cyber-intelligence-2022>

Per l'installazione, seguite le istruzioni disponibili sul file README.md

La soluzione tramite Docker Desktop *dovrebbe* funzionare senza problemi su qualsiasi piattaforma supportata (Apple Intel, Apple Silicon, Windows, Linux)

Prima di cominciare...

- La documentazione di NiFi è disponibile su <https://nifi.apache.org/docs.html>
- La documentazione di NiFi Expression Language è disponibile su <https://nifi.apache.org/docs/nifi-docs/html/expression-language-guide.html>
- Google è vostro amico!
-e ovviamente anche i vostri docenti sono sempre disponibili!

Dopo che il software Nifi è stato lanciato in esecuzione tramite docker potete accedere al pannello di controllo di Nifi andando su <https://localhost:8443/nifi/>

Username: **user**

Password: **cyberintelligence**

Prima di cominciare (continua)...

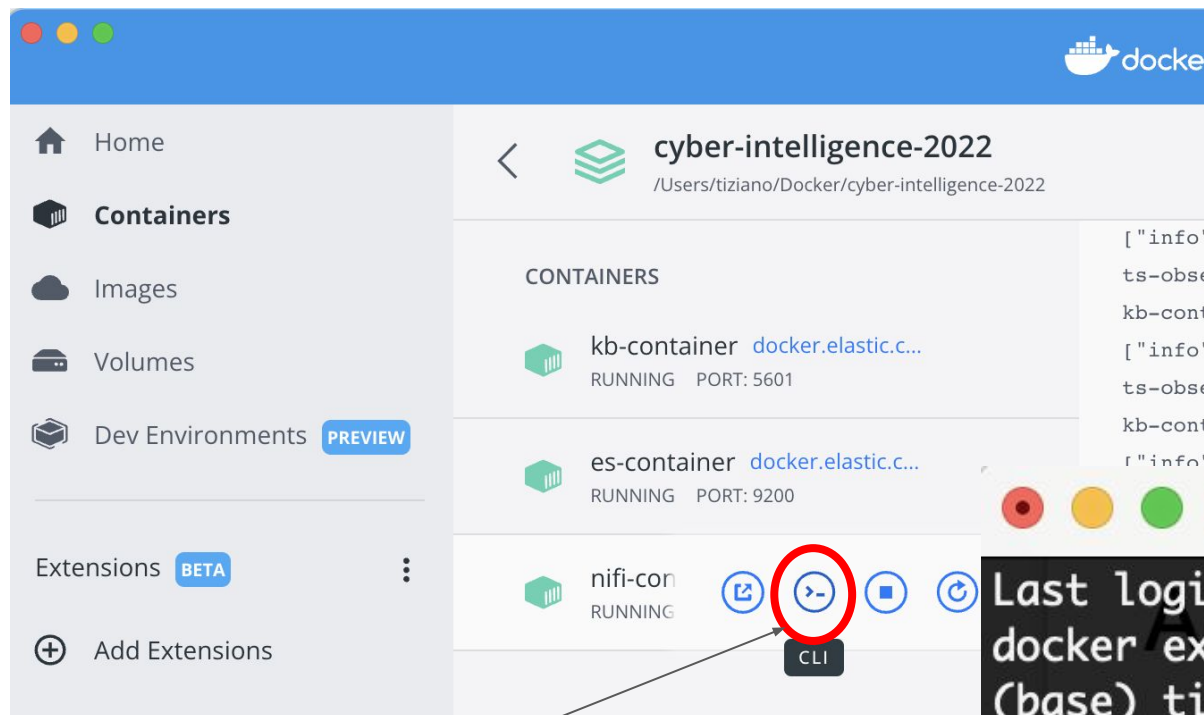
Soluzione docker composta da 3 container:

- **es-container**: VM per server Elasticsearch
- **kb-container**: VM per Kibana
- **nifi-container**: VM per Nifi

La cartella **cint** che si trova nella directory root della soluzione docker viene montata ed è condivisa con il path **/home/cint** del container “nifi-container”.

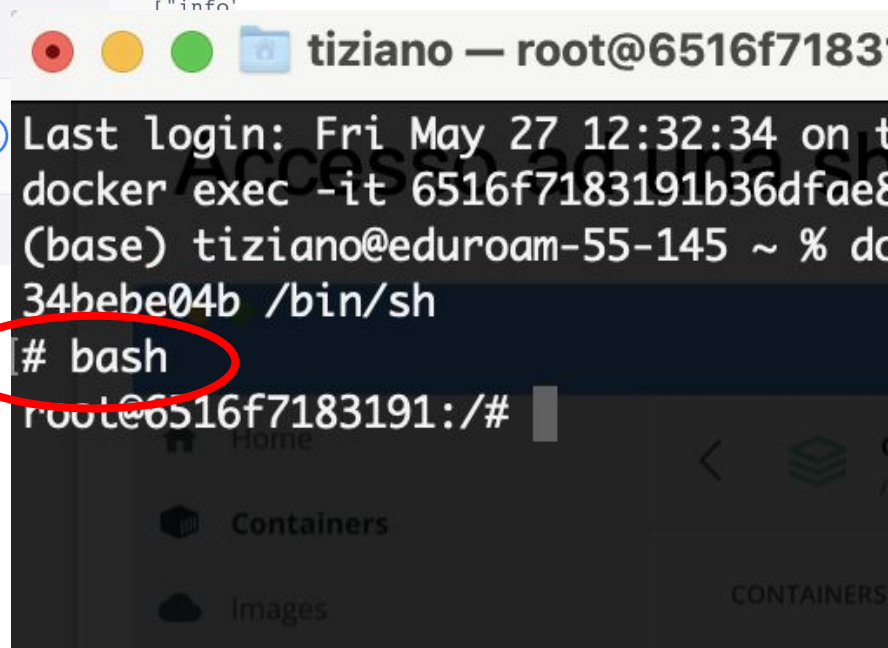
Dentro **cint** trovate le slide di documentazione (cartella *documentazione*), tutti gli script necessari al funzionamento degli esercizi proposti (cartella *esercitazione*) e l'output che viene generato da Nifi tutte le volte che eseguite un esercizio (cartella *test*).

Accesso ad una shell di un container



Premere qui per aprire un terminale sul container desiderato

Eeguire il comando **bash** sulla shell



Esercizio 1

In questo esercizio impareremo a usare e configurare i seguenti processori:

1. GetHTTP per accedere senza autenticazione ai dati forniti da un web service.
2. EvaluateJSONPath per catturare alcune informazioni dal JSON di input.
3. ReplaceText per generare un FlowFile con contenuto customizzato.

Obiettivo: Periodicamente ogni 10 secondi, ottenere e salvare su due cartelle distinte il prezzo attuale in dollari e euro delle criptovalute Bitcoin e Ethereum.

Per ottenere il prezzo corrente, si utilizzerà il Web service di CryptoCompare:

<https://www.cryptocompare.com/api/>

Esempio di richiesta/risposta del Web service

Esempio di richiesta per risolvere l'esercizio:

<https://min-api.cryptocompare.com/data/pricemultifull?fsyms=BTC,ETH&tsyms=USD,EUR>

Estratto JSON di risposta:

```
"RAW": {  
  "BTC": {  
    "USD": {  
      "TYPE": "5",  
      "MARKET": "CCCAGG",  
      "FROMSYMBOL": "BTC",  
      "TOSYMBOL": "USD",  
      "FLAGS": "4",  
      "PRICE": 6758.39,  
      "LASTVOLUME": 1.82501163,  
      ...  
    },  
    "EUR": {  
      "TYPE": "5",  
      "MARKET": "CCCAGG",
```

NiFi: il processore GetHTTP

Scopo: Permette di interrogare un server HTTP, utile per l'accesso a Web service di tipo REST.

Configure Processor

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Required field

Property	Value
URL	https://min-api.cryptocompare.com/data/pricemultifull?fsyms=B
Filename	btce\${now():toNumber()}
SSL Context Service	StandardSSLContextService
Username	No value set
Password	No value set
Connection Timeout	30 sec
Data Timeout	30 sec
User Agent	No value set
Accept Content-Type	No value set
Follow Redirects	false
Redirect Cookie Policy	default
Proxy Host	No value set
Proxy Port	No value set












Importante configurare il contesto SSL per accesso a servizi in https

NiFi: il processore GetHTTP, configurazione SSL

1. Creare un controller di tipo StandardSSLContextService.
2. Configurarli correttamente.
3. Abilitarlo.

GENERAL

CONTROLLER SERVICES

Name	Type	Bundle	State	Scope
 JettyWebSocketServer	JettyWebSocketServer 1.5.0	org.apache.nifi - nifi-websocket-s...	 Disabled	Cyberintelligence
 JettyWebSocketServer	JettyWebSocketServer 1.5.0	org.apache.nifi - nifi-websocket-s...	 Disabled	Cyberintelligence
 StandardHttpContextMap	StandardHttpContextMap 1.5.0	org.apache.nifi - nifi-http-context-...	 Enabled	Cyberintelligence
 StandardHttpContextMap	StandardHttpContextMap 1.5.0	org.apache.nifi - nifi-http-context-...	 Disabled	Cyberintelligence
 StandardRestrictedSSLContextS...	StandardRestrictedSSLContextS...	org.apache.nifi - nifi-ssl-context-s...	 Enabled	Cyberintelligence
 StandardSSLContextService	StandardSSLContextService 1.5.0	org.apache.nifi - nifi-ssl-context-s...	 Enabled	BitcoinPrice

+

Abilita o disabilita il controller

Per configurare correttamente il controller fare riferimento a questa guida:

<https://community.hortonworks.com/questions/9509/connecting-to-datasift-https-api-using-nifi.html>

Controller Service Details

SETTINGS

PROPERTIES

COMMENTS

Required field

Property	Value
Keystore Filename	 No value set
Keystore Password	 No value set
Key Password	 No value set
Keystore Type	 No value set
Truststore Filename	 /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Co...
Truststore Password	 Sensitive value set
Truststore Type	 JKS
TLS Protocol	 TLS

Configurazione contesto SSL

Configure Controller Service | StandardSSLContextService 1.16.0

SETTINGS
PROPERTIES
COMMENTS

Required field
☒
☐

Property	Value
Keystore Filename	No value set
Keystore Password	No value set
Key Password	No value set
Keystore Type	No value set
Truststore Filename	/usr/lib/jvm/default-java/lib/security/cacerts
Truststore Password	Sensitive value set
Truststore Type	JKS
TLS Protocol	TLS

CANCEL
APPLY

Usare questa configurazione quando usate la VM Linux. La password di default della JVM da usare nella proprietà Truststore Password è “changeit”.

Esercizio 1: traccia su come realizzare il dataflow

1. Configurate il processore GetHttp per accedere al WebService tramite l'URL indicato.
2. Impostate il processore GetHttp affinché sia schedulato ogni 10 secondi.
3. Processate i JSON provenienti dal Web service tramite il processore EvaluateJSONPath ed inserire dei nuovi attributi con il prezzo di BTC e ETH.
4. Splittate il flusso in due sottoflussi, ognuno corrispondente a ciascuna criptovaluta.
 - a. Usate il processore ReplaceText per riscrivere i dati nel formato voluto, ad esempio CSV.
 - b. Scrivete i dati su una cartella di output tramite il processore PutFile.

Esercizio 2

Problemi con soluzione Esercizio 1:

- Ogni flusso ha il suo processore PutFile
- Ogni file su disco contiene una sola misurazione.

Obiettivo: modificare il flusso di Esercizio 1 affinché da ciascun sottoflusso escano FlowFile contenenti 5 misurazioni ciascuno e ognuno di questo sia salvato su una opportuna cartella sulla base del valore dell'attributo "CryptoCurrency".

Nuovi processori:

- UpdateAttribute per aggiungere un nuovo attributo.
- MergeContent per unire i contenuti

Esercizio 2: traccia su come realizzare il dataflow

1. Partire da una copia dell'Esercizio 1.
2. In ciascun sottoflusso, dopo avere generato il FlowFile con formato custom, andare a inserire mediante il processore UpdateAttribute un nuovo attributo "CryptoCurrency" contenente il valore "BTC" o "ETH".
3. Nello stesso sottoflusso, utilizzare il processore MergeContent per unire 5 FlowFile differenti in un nuovo FlowFile. Il processore MergeContent deve essere impostato a:
 - "Merge Strategy" = "Bin-Packing algorithm"
 - "Merge Format" = "Binary concatenation"
 - "Delimiter strategy" = "Text"
 - "Demarcator" = newline (premere Shift + Invio)
4. Utilizzare una unica istanza del processore PutFile per ricevere i FlowFile aggregati dai 2 sottoflussi e sfruttare l'attributo "CryptoCurrency" per scrivere i dati in cartelle diverse.

[Qui](#) trovate maggiori dettagli sul processore MergeContent

Esercizio 3

Nell'esercizio impareremo a sfruttare i ProcessGroup per realizzare componenti riutilizzabili in contesti differenti.

Obiettivo: Scrivere un componente riutilizzabile che dato in input uno stream di FlowFile ritorni un uscita uno stream di FlowFile il cui contenuto è stato compresso solo se la dimensione del contenuto in input era superiore a 1000000 bytes.

Possibile caso di uso reale: Prendo i file da una directory di input, li passo a questo componente e i file risultanti (compressi o meno) vengono scritti in una directory di output.

Esercizio 3: traccia su come realizzare il dataflow

1. Definire un nuovo componente chiamato “ZipBigFile”
 - a. Creare un nuovo ProcessGroup contenente una porta di Input per prendere i dati in ingresso ed una porta di Output per restituire i dati in uscita.
 - b. Sui FlowFile in ingresso sfruttare il processore RouteOnAttribute sul valore dell’attributo “fileSize” per creare due possibili sottoflussi (“BigFile” e “SmallFile”) su cui distribuire i FlowFile.
 - c. Nel sottoflusso “BigFile” sfruttare il processore CompressContent per comprimere il contenuto del FlowFile mediante il compressore “gzip”.
 - d. Far confluire nella porta di Output definita i FlowFile dei due sottoflussi.
2. Provare ad utilizzare il componente definito collegando un processore GetFile per prendere i dati da una cartella di input e un processore PutFile per scrivere i file risultanti su disco.

Dati di esempio sono disponibili sulla cartella

“/home/cint/esercitazione/nifi/Esercizio3/” della macchina virtuale.

Impostazione account developer su Twitter

Per accedere all'API di Twitter è **necessario registrarsi come developer verificati** e quindi creare una app Twitter contenente le credenziali che sarà utilizzata all'interno dei prossimi esercizi.

Le guide su come registrarsi a Twitter come utente e developer sono disponibili su

<https://github.com/tizfa/cyber-intelligence-2022/tree/main/cint/documentazione/Twitter%20API>

Twitter API: 1.1 vs 2.0

Improvements of v2.0

- A consistent design across endpoints
- The ability specify which fields and objects return in the response payload
- New and more detailed data objects
- Receive and filter data with new contextual information powered by Tweet annotations
- Access to new metrics
- Easily identify and filter for conversations that belong to a reply thread
- Advanced functionality and increased access to data for academic researchers
- Easily return counts of Tweets that match a query
- Simplified JSON response objects by removing deprecated fields and modernizing labels

Twitter API: solutions

Essential

With Essential access, you can now get access to Twitter API v2 quickly and for free!

- Retrieve 500,000 Tweets per month
- 1 Project per account
- 1 App environment per Project
- No access to standard v1.1, premium v1.1, or enterprise

- Available for all without asking any specific infos!
- Only V2 access

Elevated

With Elevated access, you can get free, additional access to endpoints and data, as well as additional App environments.

- Retrieve 2 million Tweets per month
- 1 Project per account
- 3 App environments per Project
- Access to standard v1.1, premium v1.1, and enterprise

- Specify details for a project, quite hard to pass verification!
- V1.1 and V2 access

Academic Research

If you qualify for our Academic Research access level, you can get access to even more data and advanced search endpoints.

- Retrieve 10 million Tweets per month
- Access to full-archive search and full-archive Tweet counts
- Access to advanced search operators

- Need a detailed research project and to belong to academic institution
- V1.1 and V2 access

Data format: v1.1 vs v2.0

	v1.1 structure	v2 structure
Default	<pre>{ ~all tweet object fields~, "entities": { "hashtags": [], "symbols": [], "user_mentions": [], "urls": [], "media": [] }, "extended_entities": {}, "user": {}, "place": {}, "retweeted_status/quoted_status" }</pre>	<pre>{ "data": [{ "id", "text" }] }</pre>

All fields by default

With defined
field and
expansion
parameters

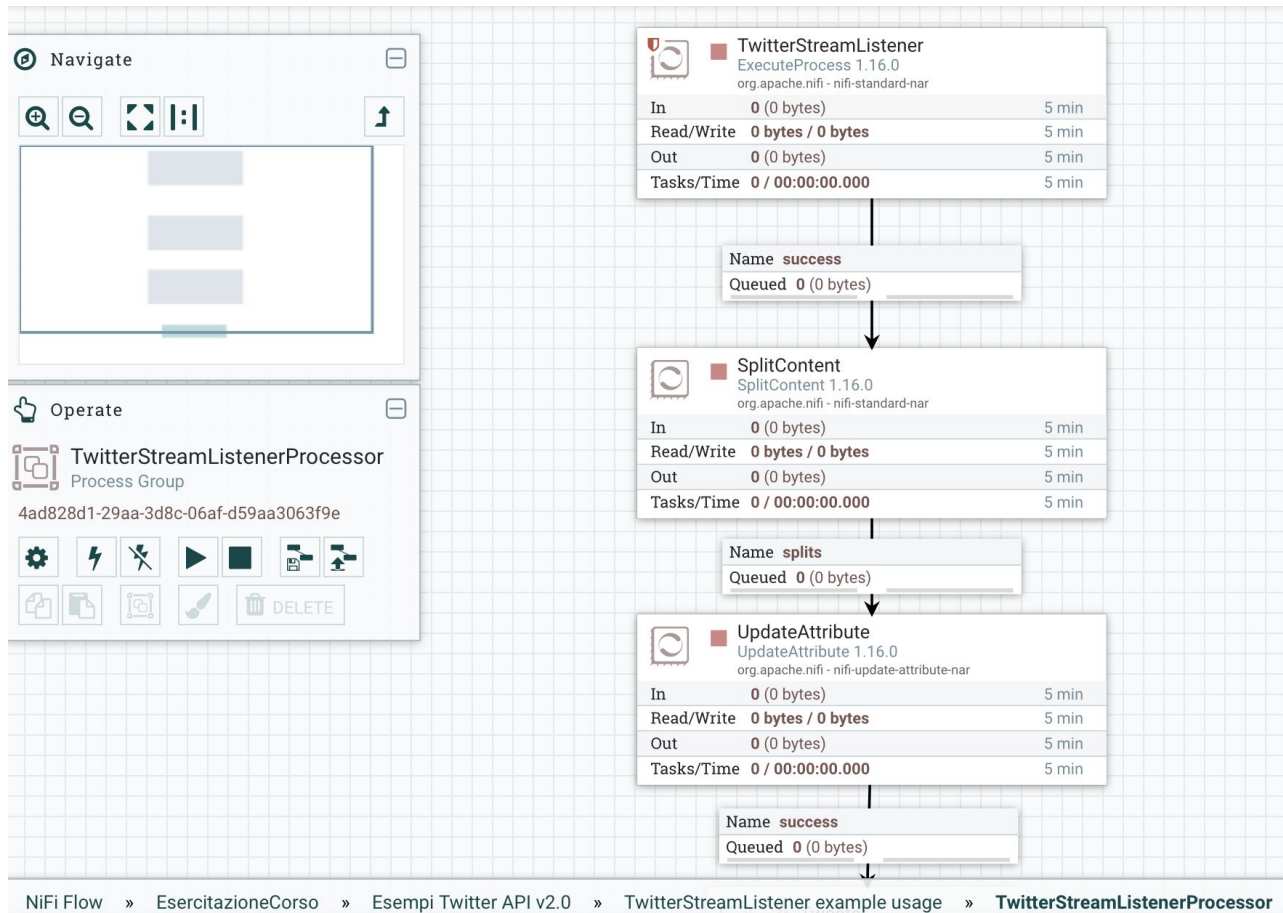
Minimal info for
primary object

Additional info requested with
fields or **expansions**
parameters

```
{
  "data": [{
    ~tweet object fields~,
    "entities": {
      "hashtags": [],
      "cashtags": [],
      "mentions": [],
      "urls": [],
    },
    "attachments": {
      "media_keys": [],
      "poll_ids": []
    }
  }],
  "includes": [
    "tweets": [~tweet objects~],
    "users": [~user objects~],
    "media": [~media objects~],
    "places": [~place object~],
    "polls": [~poll object~]
  ],
  "matching_rules": []
}
```

[Live example!](#)

Analisi processore TwitterStreamListener per API 2.0



E relativo script Python:

https://github.com/tizfa/cyber-intelligence-2022/blob/main/cint/esercitazione/twitter_stream_listener/twitter_stream_listener.py

Esercizio 4

Nell'esercizio sfrutteremo i processori:

- GetTwitter per ottenere un flusso di tweet tramite la Stream API di Twitter.
- ExecuteStreamCommand per eseguire uno script Python custom.

Obiettivo: Mettersi in ascolto delle keyword “(conte OR salvini OR meloni OR letta OR berlusconi)” dei tweet in lingua italiana e processarli tramite lo script CheckPoliticians_v2.0.py (disponibile nella cartella Esercizio4). Lo script arricchisce i FlowFile con la polarità del sentiment di ciascun tweet ed individua al suo interno i nomi dei politici di cui si parla. Es: `"politicians": {"polarity": "-1", "names": ["Salvini"]}`. Sulla base del valore “politicians.polarity” si scrivono i FlowFile finali in tre cartelle distinte.

Esercizio 4: traccia su come realizzare il dataflow.

Stavolta provate a pensarci un attimo da soli!! :-)

- Per il processore custom `TwitterStreamListener` usate il vostro bearer token che vi sarete generati dal sito Twitter.
- Nel componente `ExecuteStreamCommand` ricordate di utilizzare come comando di Python `python3`.
- A parte questi due nuovi componenti, potete usare componenti che conoscete già per risolvere l'esercizio.

Dopo guardiamo insieme la soluzione, compreso il codice dello script.

Esercizio 5

Scaricamento delle pagine linkate nei tweets

Estrarre il contenuto delle pagine Web linkate nei tweet (utilizzare lo script `WebPageContentExtractor.py` presente nella cartella `Esercizio5`). Lo script estrae, se disponibile, il primo URL buono dal tweet, scarica la pagina Web, estrae titolo e contenuto dell'articolo, e arricchisce il JSON originale con queste informazioni. Es:

```
"webpage" :  
{  
  "url": "https://t.co/6wixF7WHyq",  
  "title": "Notizia su Twitter...",  
  "content": "Ora sei..."  
}
```

Esercizio 5 (2)

Obiettivo

Mettersi in ascolto delle keyword “e,allora” dei tweet in lingua italiana e processarli tramite lo script `WebPageContentExtractor.py` . Sui FlowFile arricchiti, identificare quelli che hanno associato un contenuto Web da quelli che non hanno contenuti Web associati, scrivendo i JSON finali in cartelle separate.

Verificare che la coda di collegamento tra il processore `TwitterStreamListener` e il processore `ExecuteStreamCommand` (che esegue lo script) si riempie molto velocemente e trovare un modo per velocizzare il processamento (suggerimento: provare a variare il numero di istanze di processamento dello script).

Maggiori informazioni su come Nifi ottimizza latenza e throughput:

<https://community.cloudera.com/t5/Community-Articles/Understanding-NiFi-processor-s-quot-Run-Duration-quot/ta-p/248921>

Esercizio 6

Obiettivo

Riadattare i dataflow degli Esercizi 4 e 5 in modo tale da incapsularli in 2 componenti riutilizzabili. Sfruttando questi 2 nuovi componenti, scrivere un dataflow facente le seguenti cose:

1. Si mette in ascolto sulla query "(ucraina OR putin OR zelenski OR zelensky OR zelensky) -is:retweet -is:reply -is:quote lang:it".
2. I tweet recuperati vengono processati e arricchiti dal componente "Esercizio 4".
3. I tweet arricchiti dal componente "Esercizio 4" (solo quelli con polarità positiva o negativa) sono quindi processati dal componente "Esercizio 5".
4. I tweet provenienti da componente "Esercizio 5" e aventi link a contenuto Web sono memorizzati su Elasticsearch nell'indice "esercizio6".
5. I tweet aventi polarità neutra oppure senza link a contenuto Web sono raccolti e fatti morire da un processore ad hoc (a tale scopo usare processore UpdateAttribute con autoterminazione)

Esercizio 6: traccia visuale

- Sfruttare i ProcessGroup, le Input e le Output Port per riorganizzare gli esercizi già svolti
- Per ES 7.x è necessario ricordarsi di impostare la proprietà type a “_doc”

