**RKM Architecture:**

Basically, this tool helps to find or search content in various document format ex: pdf / doc / excel / txt / ppt /note and etc. Here we have Total 3 layers to connect with Destination repository or file containers.

List of Layers,

1. RKM Front End Tool
2. ELK Engine
3. Connectors or Receivers – File Watcher

RKM Front End Tool:

It does search content / edit or update content / delete content. How it does search? It retrieves all alert, I mean email alert from (tower), once it receives email alert, it does list in the grid. The grid contains mainly content with 2 Action events. Action event 1, it does search from ELK repository, action event 2, it does update on repository location either in NAS / SQL DB.

ElasticSearch:

Elasticsearch is the repository that holds all content in JSON format. <https://www.elastic.co/what-is/elasticsearch>

It holds and convert all document into meta data. How it converts as meta data?. What is ELK index? It holds all meta data in shred format. What is shred? Consider shred is the container, it has primary and secondary (replica). It builds internally, but we need to configure how much we required. 100% search will not disturb primary it does search content in secondary replica. How does it synchronize, it uses inbuild flash copy to do synchronize data between primary and secondary replica? It has component called pipeline, what is pipeline? Consider this pipeline behave like transporter, which we need to define based on our requirement. Basically, we need to define required parameters and their format. Once this is done while transporting data via pipeline, we need to use pipeline in post command. So that it sends the data to ELK indices.

Each Content Source(VLMS, NAS) has its own index.

File Watcher:

This File watcher does main role, it performs or retrieve details from Destination source ex: NAS / Database. There 2 different components play main role here. This component written in JAVA code.

* Component1: Reading content from physical NAS path, it does convert the actual content into base64 and it passes to ELK, ELK can read base64 format. During this time, it does entry in MYSQL database (RKM DB), it can hold file properties and also it contains MD5 converted string. Why MD5 algorithm used in this flow? It helps to identify modified content from that actual / Physical NAS path. And it does avoid un modified content while do pipeline transport. And also this File watcher used recursive logic to read nested child folders. And also it does file delete in ELK / RKM

DB, if the file is missing from Source repository.

Pre-requests and Logic:

Pre-Req: Configure NAS path as network drive in Windows / Ensure index and pipeline got created.

Flow:

1. Read Content from Mapped path using recursive logic
2. Ensure we are converting only file not folder
3. Ensure File converted to base 64
4. Ensure file not process already using our RKM DB
5. Ensure file processed but modified using MD5 string in our RKM DB
6. Convert the file to Base64
7. Post Base64 with required fileproperties into Elastic search(ELK) using specific index via specific pipeline
8. Ensure File pushed to ELK Successfully and place an entry in MYSQL DB, continue same process with files and folders until file count ends

* Component2: Read content from SQL DB, it does convert the actual filestream content into base64 and it passes to ELK, ELK can read base64 format. As I defined in component 1, here also we need to define MD5 algorithm to identify modify content.

Pre-requests and Logic:

Pre-Req: Access to VLMS SQL Server

Flow:

1. Run SQL query against VLMS DB
2. Ensure Filestream converted to base 64
3. Ensure file not process already using our RKM DB
4. Ensure file processed but modified using MD5 string in our RKM DB
5. Convert the file to Base64
6. Post Base64 with required fileproperties into Elastic search(ELK) using specific index via specific pipeline
7. Ensure File pushed to ELK Successfully and place an entry in MYSQL DB, continue same process with files and folders until file count ends

Note:

* Development Tools
  + HTML / JAVA / Springboot / Elastic Search (ELK) / ELK Pipeline / ELK Attachment / ELK Logstash / MYSQL

**Architecture diagram:**

ELK Engine: (Index / Shred (Primary / Secondary (Replica) / Logstash / Pipeline (attachment)) – META DATA

R

R

Alert Display

Search Display

Modify / Update search content

User Module

File Watcher 1: (Read NAS content) uses pipeline, attachment, recursive, MD5, Base64 and RKM DB

File Watcher 2: (Read MS SQL DB) uses pipeline, attachment, MD5, Base64, Logstash and RKM DB

RKM DB: It holds File properties / Audit details / Email Alerts / MD5 String

NAS Path

VLMS DB

R

R

R/W

R/W

R/W

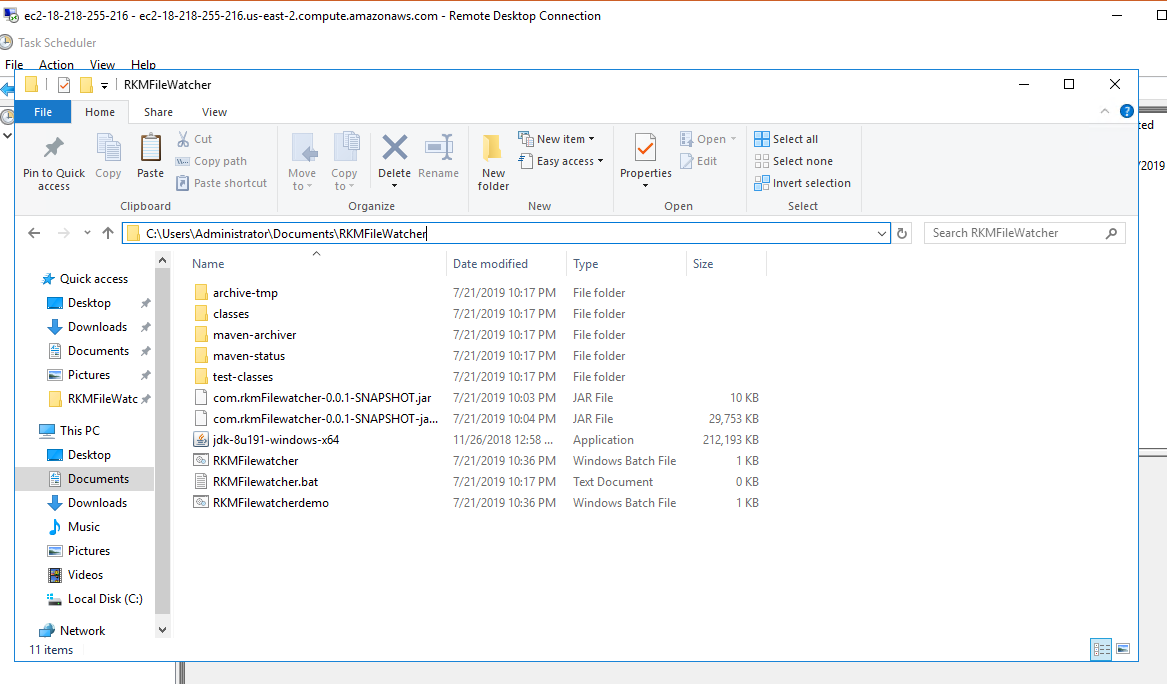
R/W

R/W

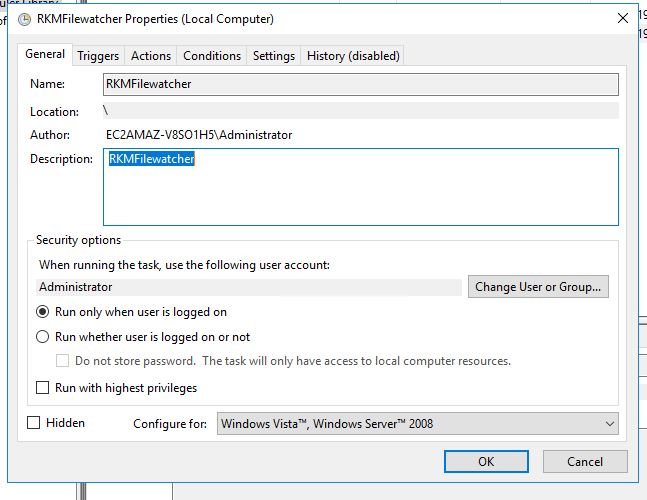
R/W

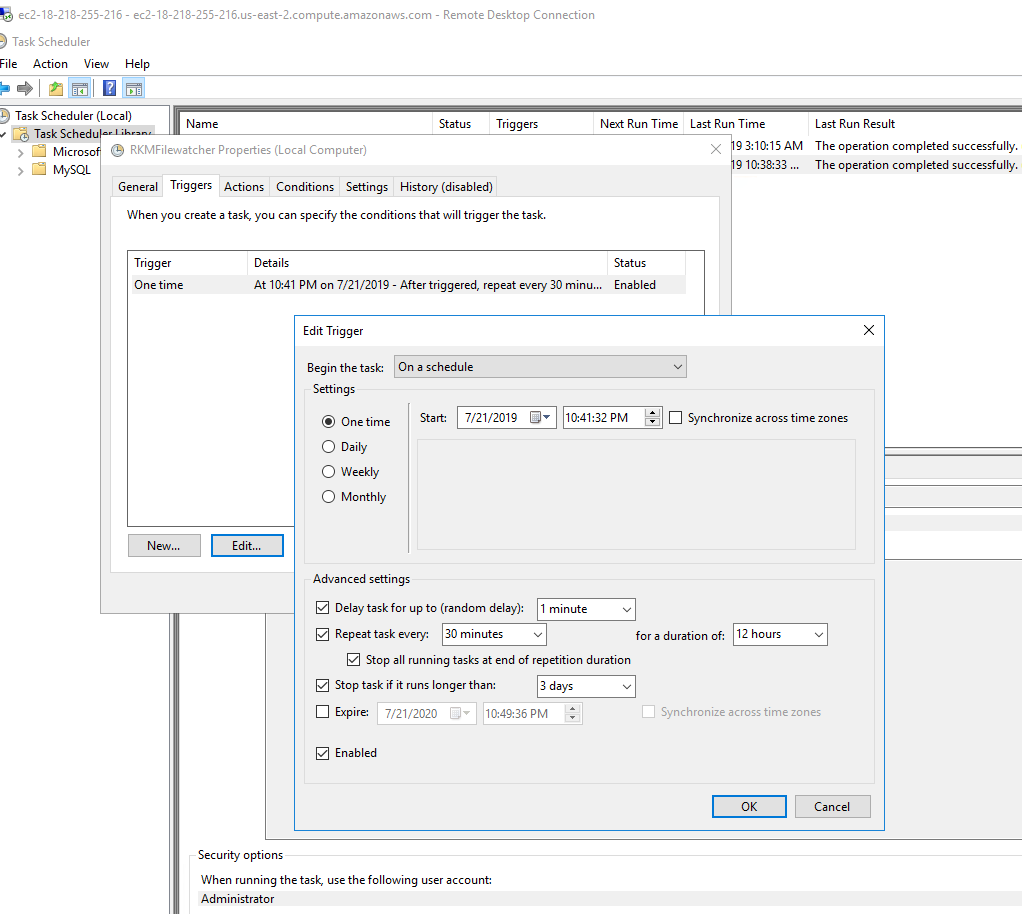
***How to Deploy Package for Filewatcher?***

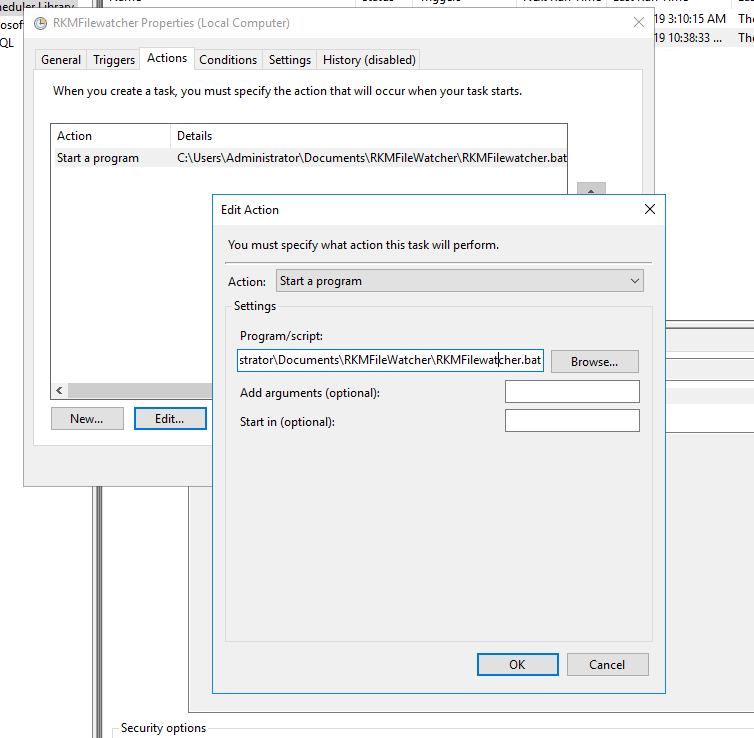
Step 1: Deploy our package on Windows (ensure jdk 1.8 installed)



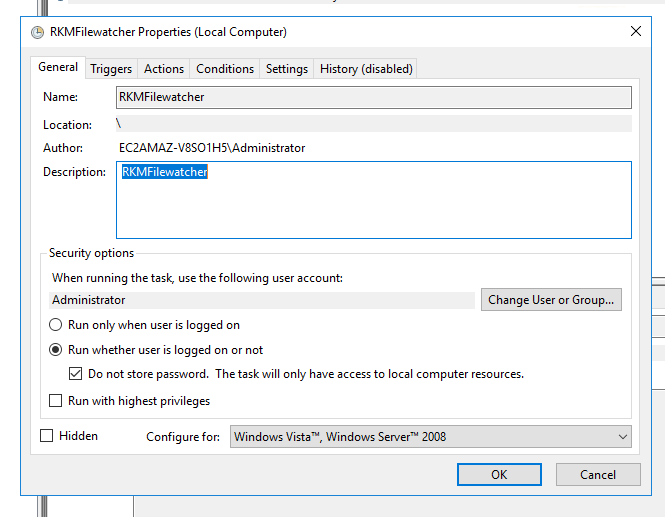
Step 2: GOTO Task scheduler create as showed below





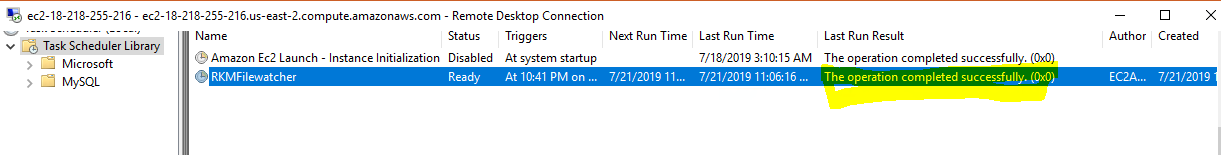


Step 3: Ensure you removed logged on user access as like below,

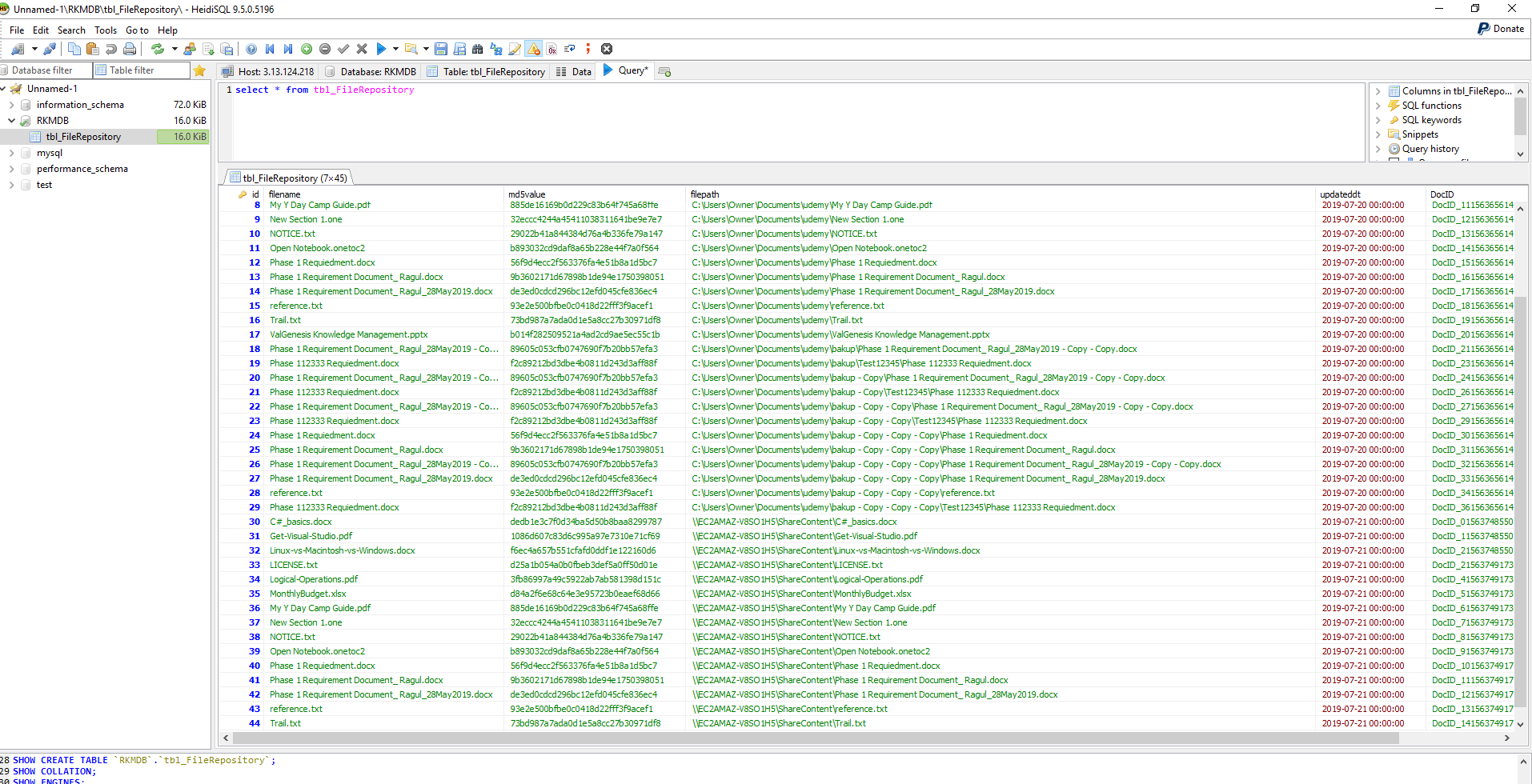


Step 4: Right Click and Run

Results:



Database entry:



Actual Execution result in command prompt at AWS – windows server:



Elastic Search Configuration:

**Step 1:**

Elastic Search can install on Windows using online download zip file [Elastic Search](https://www.elastic.co/downloads/elasticsearch) (5.2.2).

Pre-requisites need to install JDK JAVA 9 [Java 8](http://www.oracle.com/technetwork/java/javase/downloads/index.html).

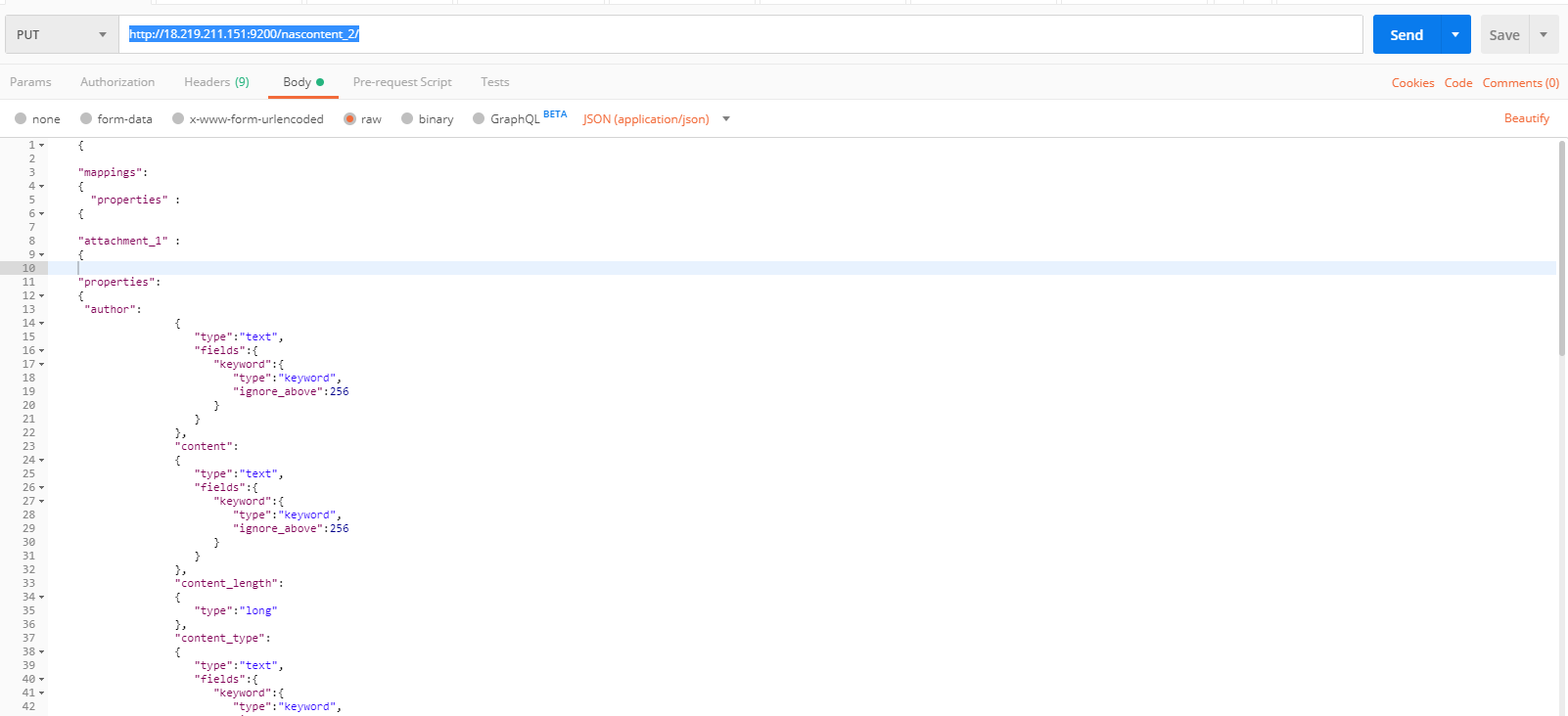
Steps to install ELK:

Please follow [Elastic Search Installation Guide](https://www.elastic.co/guide/en/elasticsearch/reference/5.2/windows.html) on windows. Please follow [Elastic search Installation Guide (Linux)](https://tecadmin.net/setup-elasticsearch-on-ubuntu/) on Linux Ubuntu.

**Step 2:**

In Elastic Search, how to create Index / pipeline / attachment?

* Install Postman tool
* Do Postman put command similar to below screen shot



**Detail create json for index:**

{

"mappings":

{

"properties" :

{

"attachment\_1" :

{

"properties":

{

"author":

{

"type":"text",

"fields":{

"keyword":{

"type":"keyword",

"ignore\_above":256

}

}

},

"content":

{

"type":"text",

"fields":{

"keyword":{

"type":"keyword",

"ignore\_above":256

}

}

},

"content\_length":

{

"type":"long"

},

"content\_type":

{

"type":"text",

"fields":{

"keyword":{

"type":"keyword",

"ignore\_above":256

}

}

},

"date":

{

"type":"date"

},

"language":

{

"type":"text",

"fields":{

"keyword":{

"type":"keyword",

"ignore\_above":256

}

}

},

"title":

{

"type":"text",

"fields":{

"keyword":{

"type":"keyword",

"ignore\_above":256

}

}

}

}

},

"data" :

{

"type":"text",

"fields":{

"keyword":{

"type":"keyword",

"ignore\_above":256

}

}

},

"filepath":{

"type":"text",

"fields":{

"keyword":{

"type":"keyword",

"ignore\_above":256

}

}

},

"filename":{

"type":"text",

"fields":{

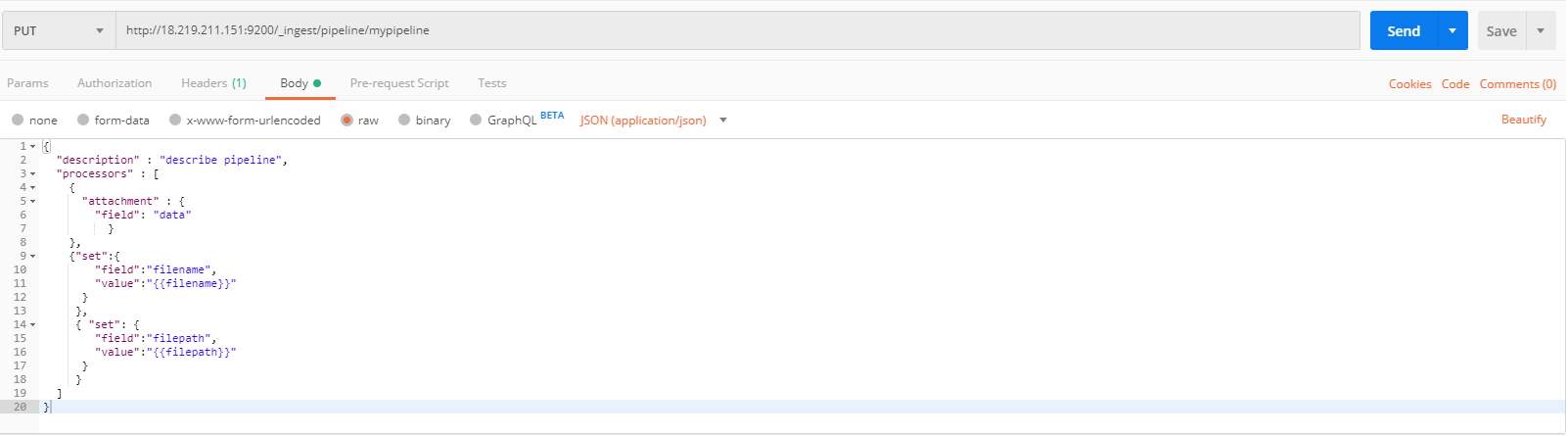
"keyword":{

"type":"keyword",

"ignore\_above":256

}}}}}}

How to create Pipeline?



**Detail create json for pipeline:**

{

"description" : "describe pipeline",

"processors" : [

{

"attachment" : {

"field": "data"

}

},

{"set":{

"field":"filename",

"value":"{{filename}}"

}

},

{ "set": {

"field":"filepath",

"value":"{{filepath}}"

}

}

]

}