1. **Crear máquina de Ubuntu con VirtualBox:**
    1. Link de descarga: https://ubuntu.com/download/desktop
    2. Tutorial de instalación: https://brb.nci.nih.gov/seqtools/installUbuntu.html

2. **Instalación de herramientas y prueba con el tutorial de "weather":**
(Link de la página: https://medium.com/forsk-labs/real-time-weather-analysis-using-kafka-and-elk-pipeline-a849eb27017a)

    1. Instalar **Java** según el tutorial

    2. Instalar el paquete **Kafka + Zookeeper** según tutorial, modificando el enlace de descarga por el siguiente: http://ftp.nluug.nl/internet/apache/kafka/2.2.1/kafka_2.11-2.2.1.tgz (Mover la carpeta a */etc/init.d*, creando la carpeta *kafka*)

    3. Probar a arrancar primero Zookeeper y luego Kafka, en ventanas diferentes:

        cd /etc/init.d/kafka
        bin/zookeeper-server-start.sh config/zookeeper.properties

        bin/kafka-server-start.sh config/server.properties

    4. Crear el topic "weather" en Kafka según tutorial

        *bin/kafka-topics.sh --create *
        *--zookeeper localhost:2181 *
        *--replication-factor 1 *
        *--partitions 1 *
        --topic weather

    5. Instalar **Python** y "confluent":
        a. Python:
            $ sudo apt-get update
            $ sudo apt-get install python3.6

        b. "confluent-kafka":
            $ sudo apt install python3-pip
            $ sudo pip3 install confluent-kafka

    6. Crear archivo en **Python**, con el editor **Gedit**, "*gedit producer.py*"

        from confluent_kafka import Producer
        import json
        import requests
        import time
        p = Producer({'bootstrap.servers': 'localhost:9092'})
        while True:
        response1 =
        requests.get("http://api.openweathermap.org/data/2.5/weather?q=London,uk&APPID=89d3481e151103e02e7803f338233b6c")
        p.produce('weather', key='london', value=response1.text)

Ejecutar como: *python3 producer.py*

7. Instalar **ELK** y ubicar las carpeta en /etc/init.d (descargar tar.gz y seguir tutorial):
    a. Link de descarga **Logstah**: https://www.elastic.co/downloads/logstash
    b. Link de descarga **Elasticsearch**: https://www.elastic.co/downloads/elasticsearch
    c. Link de descarga **Kibana**: https://www.elastic.co/downloads/kibana

8. Iniciar Logstash como:

    cd /etc/init.d/logstash

    bin/logstash --path.settings /etc/init.d/logstash/config -e 'input { kafka { bootstrap_servers => "localhost:9092" topics =>"weather" } } filter { json { source =>"message" } } output { elasticsearch { hosts =>["localhost:9200"] index => "weather" } stdout {codec => rubydebug } }'

9. Iniciar Elasticsearch y Kibana:
    a. Elasticsearch:
    cd /etc/init.d/elasticsearch
    ./bin/elasticsearch

    (Comprobar su funcionamiento en http://localhost:9200/ y visualizar el "index" "weather")

    b. Kibana:
    cd /etc/init.d/kibana
    ./bin/kibana

    (Comprobar su funcionamiento en http://localhost:5601/ y cargar datos creando el "Index Pattern")

3. Probar herramientas con los datos de tráfico de Santander (Punto actualizado en el apartado 4.)

En este caso solamente se modifican los pasos 4, 6 y 8.

1. Se crea un nuevo topic en Kafka: "trafico" (paso 4).

2. Crear un nuevo archivo de Python: "trafico.py", modificando la dirección web (paso 6):

    http://datos.santander.es/api/rest/datasets/mediciones.json?items=467
    p.produce('trafico', key='trafico', value=response1.text)
    time.sleep(300)

    (Añadimos un time.sleep de un minuto, para reducir el número de llamadas a la API)

3. Se modifica el comando para ejecutar Logstash (paso 8):

    topics =>"trafico"

    index => "trafico"

4. Actualización. Modificación de los valores del archivo json, a *float*.

1. Para ello descargamos los archivos de la API, en formato csv, y creamos un archivo .py para llevar a cabo la transformación de csv a json.

```
import json
import os

with open("mediciones.csv",'r') as f:
    # read lines, strip newlines, split at ,
    lines = [ x.strip('\n').split(',') for x in f.readlines()]


listDic = []
for lineIndex in range(1,len(lines)):
    row = lines[lineIndex]     # get data row
    row[2] = float(row[2])     # convert data
    row[3] = float(row[3])     # convert data
    row[4] = float(row[4])     # convert data
    row[5] = float(row[5])     # convert data
    row[6] = float(row[6])     # convert data

    # zip to tuples of (key,value) and append to result list of dicts
    listDic.append( dict( zip(lines[0],row)))


with open("test_json.json", 'w') as json_file_ind:
    json_file_ind.write('{"resources":[\n')

    for row  in listDic:
        json_file_ind.write(json.dumps(row,sort_keys=False,
                        indent=4, separators=(',', ': ')))
        json_file_ind.write(',\n')

x = open("test_json.json").read()
os.remove("test_json.json")


with open("test_json.json", 'w') as json_file_ind:
    json_file_ind.write(x[:-2])
    json_file_ind.write(']}\n')
```

2. Obtenido el json, se carga a Kafka con el comando:

```
jq -rc . test_json.json | bin/kafka-console-producer.sh --broker-list localhost:9092 --
topic  trafico
```

3. Antes de cargar los datos se debe eliminar cualquier *topic* creado anteriormente con el mismo nombre:

```
bin/kafka-topics.sh --zookeeper localhost:2181 --delete --topic trafico
```

4. A continuación, se inicia Logstash y se lanzan los datos a Elasticsearch con el comando:

   bin/logstash --path.settings /etc/init.d/logstash/config -e 'input { kafka { bootstrap_servers => "localhost:9092" topics =>"trafico" } } filter { json { source =>"message" } } output { elasticsearch { hosts =>["localhost:9200"] index => "trafico" } stdout {codec => rubydebug } }'

5. Creación de índice para datos de ubicación de los sensores

Se descarga el archivo SHP del enlace: http://datos.santander.es/dataset/?id=datos-trafico y se convierte a CSV (Se borran los paréntesis de "RefName" y "Text" para cargarlo en Logstash).

1. Se crea un archivo .json en la carpeta /etc/init.d/logstash para usar como template:
   gedit sensores_template.json
   Dentro del archivo:

```
{
 "index_patterns": ["sens*"],
 "order": 1,
 "settings": {
  "index.number_of_shards": 1
 },
 "mappings": {
    "properties": {
     "location": {
      "type": "geo_point"
      },
         "RefName": {
      "type": "float"
         },
         "Elevation": {
      "type": "float"
         },
         "Text": {
      "type": "float"
         }
    }
  }
}
```

2. Se crea un archivo .conf para cargar con Logstash:
   gedit sensores_conf.conf

```
input {
        file {
    path => "/home/selataboada/Data/mygeodata/Coordenadas.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
```

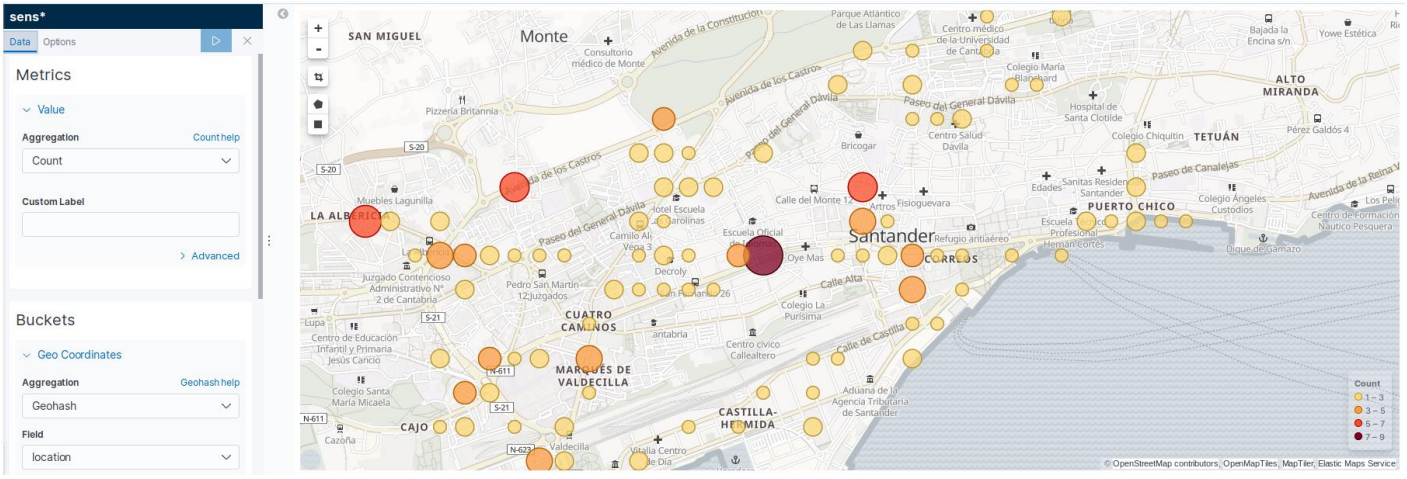
```
            }
}

filter {
    csv {
            separator => ","
            columns => [ "Y", "X" , "Elevation" , "RefName", "Text" ]
             }
    mutate {
       remove_field => [ "message", "host", "@timestamp", "@version" ]
              }
    mutate {
       convert => { "Y" => "float" }
       convert => { "X" => "float" }
       convert => { "Elevation" => "float" }
       convert => { "RefName" => "float" }
       convert => { "Text" => "float" }
          }

    mutate {
       rename => {
          "X" => "[location][lon]"
          "Y" => "[location][lat]"
         }
        }
}
output {
   elasticsearch {
    hosts=>"localhost:9200"
    index => "sensores"
    template => "sensores_template.json"
    }
stdout {}
}
```

3. Ejecutar con el comando:

   bin/logstash --path.settings /etc/init.d/logstash/config -f sensores_conf.conf

4. Para cualquier modificación en los archivos .json o .conf, se debe eliminar primero el índice creado anteriormente con el comando:

   curl -XDELETE localhost:9200/sensores



1. Ejemplo de visualización de la ubicación de sensores en Kibana.