Logstash

Práctica

Se propone preprocesar las líneas de logs de la forma log-generator-*
Para ello deberán utilizar el grok debugger dentro de las Dev Tools de Kibana.
Un link útil para entender cómo formular los patrones es: http://grokdebug.herokuapp.com/patterns#

1. Preprocesar las líneas de logs con la siguiente estructura y convertir cada dato al tipo correspondiente:
   a.

```
INFO 2019-07-09 14:55:48 [main] a.b.t.loggenerator.LogGenerator - PERFORMANCE|1.591|SEND EMAIL|SUCCESS|Batman
```

   b.

```
INFO 2019-07-09 14:55:48 [main] a.b.t.loggenerator.LogGenerator - LOGIN|204|Superman|253.37.2.46
```

   c.

```
ERROR 2019-07-09 14:55:48 [main] a.b.t.loggenerator.LogGenerator - User can not be null. Can not fetch information from the database| java.lang.Exception: User can not be null. Can not fetch information from the database
at am.ballesteros.training.loggenerator.LogGeneratorKt.generateExceptionTrace(LogGenerator.kt:66)
at am.ballesteros.training.loggenerator.LogGeneratorKt.generateTraces(LogGenerator.kt:31)...
```

El resultado esperado luego de obtener los documentos bien conformados sería:

a. Para los de PERFORMANCE:

```
{
        level:INFO
        date: 2019-06-10T17:32:31.774Z (Aplicar date)
        thread:main
        class:a.b.t.loggenerator.LogGenerator
        logtype: PERFORMANCE
        duration:1.591
        action:SEND EMAIL
```

```
        status: SUCCESS
        user: Batman
}
```

b. Para los de LOGIN:

```
{
        level:INFO
        date: 2019-06-10T17:32:31.774Z (Aplicar date)
        thread:main
        class:a.b.t.loggenerator.LogGenerator
        logtype: LOGIN
        responseText: "Texto Correspondiente al codigo 204"
        (Aplicar Translate)
        code:204
        login true (Aplicar mutate)
        user:Superman
        ip:253.37.2.46
}
```

c. Para los de ERROR:

```
{
        level:INFO
        date: 2019-06-10T17:32:31.774Z (Aplicar date)
        thread:main
        class:a.b.t.loggenerator.LogGenerator
        logtype: JAVA_ERROR (Aplicar mutate)
        description:User can not be null
        stacktrace: Can not fetch information from the database|
                java.lang.Exception: User can not be null. Can not
                fetch information from the database
                at
                am.ballesteros.training.loggenerator.LogGeneratorKt.
                generateExceptionTrace(LogGenerator.kt:66)
                at
                am.ballesteros.training.loggenerator.LogGeneratorKt.
                generateTraces(LogGenerator.kt:31)...
}
```

2. Utilizar el filtro "Geoip" para agregar la información geográfica según la ip

Resolución

```
input {
    beats{
        port => 5044
    }
}

filter{
    grok {
        break_on_match => true
        patterns_dir => "patterns"
        match => {"message" => [ "%{WORD:level} %{TIMESTAMP_ISO8601:date}
\[%{WORD:thread}\] %{JAVACLASS:class} -
%{WORD:logtype}\|%{NUMBER:duration:float}\|%{DATA:action}\|%{WORD:status}\|%{USERNAME
:user}","%{WORD:level} %{TIMESTAMP_ISO8601:date} \[%{WORD:thread}\]
%{JAVACLASS:class} -
%{WORD:logtype}\|%{NUMBER:code:int}\|%{USERNAME:user}\|%{IP:ip}","(?m)%{WORD:level}
%{TIMESTAMP_ISO8601:date} \[%{WORD:thread}\] %{JAVACLASS:class} -
%{DATA:description}\|%{GREEDYDATA:stacktrace}"]
        }
    }

    if [logtype] == "LOGIN"{
        translate{
            field => "[code]"
            destination => "[responseText]"
            dictionary =>{
                "404" => "Usuario inexistente"
                "202" => "Ingreso satisfactorio"
                "204" => "Ingreso satisfactorio con remember me"
                "400" => "Email o contraseña erronea"
                "100" => "Codigo de respuesta para el codigo 100"
                "150" => "Codigo de respuesta para el codigo 150"
                "201" => "Codigo de respuesta para el codigo 201"
                "500" => "El servidor no se encuentra disponible en este momento"
                "250" => "Ingreso satisfactorio usuario nuevo"
            }
        }
    }

    if ![logtype] {
        mutate{
            add_field => { "logtype" => "JAVA_ERROR"}
        }
```

```
    }

    if [logtype] == "LOGIN"{
        mutate{
            add_field => {"login" => true}
        }
        if [code] > 300 {
                mutate{
                    replace => {"login" => false }
                }
            }
        }

    date{
        match => ["date", "yyyy-MM-dd HH:mm:ss"]
    }

    if [logtype] == "LOGIN"{
        geoip{
            source => "ip"
        }
    }
}

output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "log-generator-%{+YYYY.MM.dd}"
    }
}
```

Opcional

Si, adicionalmente a la práctica quisieran instalarse un entorno ELK local deberían seguir los siguientes pasos

Prerrequisitos

Descargar (https://www.elastic.co/downloads/):

- Elastic Search
- Kibana
- Logstash
- Filebeat

Configuracion FileBeat y Logstash

Como primer paso vamos a hacer una configuración de prueba, para luego poder probar nuestros filtros de logstash sin tener la necesidad de indexar cada vez que probemos.

Clonar el siguiente repositorio: https://github.com/leofs94/workshop-elastic

- En la carpeta filebeat-conf el archivo filebeat.yml es un ejemplo del archivo original de config. El que se debe modificar es **filebeat.yml**

En el repositorio se encuentran los archivos de configuración necesarios para comenzar con el workshop, un generador de logs, un archivo de logs que es el que vamos a utilizar en este workshop y la configuración de logstash (un ejemplo y uno que vamos desarrollar más adelante).

Filebeat

En cuanto a filebeat vamos a probar configurando nuestro **filebeat-conf/filebeat.yml** asignando como input nuestra carpeta de logs en "paths" (la ruta donde se van a alojar nuestros logs)

```
- type: log

  # Change to true to enable this input configuration.
  enabled: true

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - D:/Users/Usuario/Desktop/Elastic/logs/log-*.log
```

para probar si recolecta la información correctamente, comentamos todos los outputs y agregando las siguientes líneas al archivo:

```
#================================ Outputs =====================================

# Configure what output to use when sending the data collected by the beat.

output.console:
  pretty: true
```

Luego vamos a ejecutar las herramientas en el siguiente orden moviéndonos hacia la carpeta que corresponda en el shell y ejecutando:

1. Elasticsearch
    a. **.\elasticsearch**
2. Kibana
    a. **.\kibana.bat**
3. Logstash
    a. **Copiar el archivo logstash-conf/log-workshop.conf dentro de la carpeta de logstash.**
    b. **.\bin\logstash.bat -f log-generator.conf**
4. Filebeat
    a. **Copiar el archivo filebeat-conf/filebeat.yml a la carpeta de filebeat.**
    b. **.\filebeat.exe -c filebeat.yml**
    c. **Una vez abierto este programa no imprime nada por pantalla!**

Aclaración: En el caso de FileBeat y Logstash lo que estamos pasando como parámetro es el archivo de configuración, puede llamarse de otra forma mientras esté conformado correctamente.

Una vez finalizado lo anterior vamos a descargar el archivo de log y con nuestro stack de herramientas corriendo, vamos a posicionarlo dentro de nuestra carpeta (la que definimos en el parámetro path de filebeat.yml).

Si todo funciona correctamente en el shell sobre el que corremos Filebeat se deberían estar mostrando nuestros logs con estructura de objeto json, siempre y cuando se descomenten las siguientes líneas:

```
output.console:
   pretty: true
```

Y comentado:

```
output.logstash:
  # The Logstash hosts
  hosts: ["localhost:5044"]
```

Filebeat no soporta 2 o más outputs a la vez de distintos tipos.

Si Filebeat está funcionando correctamente como último paso vamos a comentar el output que agregamos anteriormente y descomentar nuestro output para logstash.

Logstash

Vamos a utilizar logstash para procesar nuestras líneas de logs y darles las estructura de un documento, pero antes que nada vamos a realizar una prueba para ver si los datos nos llegan correctamente:

Como primer paso vamos a crear un archivo de configuración con el siguiente contenido y lo vamos llamar **log-generator.conf (opcional este nombre).**

```
input {
    beats{
        port => 5044
    }
}


output {
   stdout{

   }
}
```

Una vez que tengamos esto, vamos ejecutar logstash pasándole este archivo como parámetro, y luego vamos a mover una copia de nuestro archivo de logs a la carpeta destino que indicamos en la configuración de Filebeat, si los logs empiezan a aparecer en el shell donde se ejecutó logstash es porque está funcionando correctamente y ya podemos poner manos a la obra con el preprocesamiento de nuestros datos.

Realizar la Práctica para obtener los patrones grok antes de seguir.

Una vez que lo imprima en el shell dispuesto de la manera anteriormente explicada, se deberá modificar el archivo de configuración para que la salida sea ElasticSearch.

```
output {
    elasticsearch{
        hosts => ["localhost:9200"]
        index => "log-workshop-%{+YYYY.MM.dd}"
    }
}
```

Habrá que detener Logstash y volver a ejecutar para que tome los cambios de configuración.