Elastic Stack

Introducció a la suite "Elastic Stack":

Acabem de veure com Sysdig i Falco poden ser una combinació perfecta per implementar un HIDS (un "Host Intrusion Detection System"). Aquests tipus de programes permeten tenir una visibilitat en temps real de tots els events de seguretat que estan passant a un determinat sistema i alertar sobre comportaments que es detectin fora d'un patró reconegut com correcte. En aquest sentit, podríem utilitzar Sysdig + Falco (també journalctl!!) per observar en temps real els logs de determinats programes, per comprovar la integritat de fitxers i carpetes (i notar canvis sospitosos), per detectar de "rootkits", per adonar-se de canvis en el funcionament del serveis, espai de disc, fitxers de contrasenyes, ports oberts, etc, etc.

Però ¿què passa si tenim diversos sistemes a monitoritzar? No podem estar al davant de la pantalla de cadascun a la vegada per observar els missatges que Sysdig/Falco generen. A més, instal.lar el Falco a cada sistema monitoritzat tampoc és una sol.lució gaire elegant: és molt més òptim tenir instal.lat només el Sysdig a cada màquina monitoritzada i llavors implementar algun sistema d'enviament per xarxa de la informació per fer-la arribar a un únic Falco centralitzat que recollís ell sol els events dels diferents sistemes. D'aquesta manera, tot es tindria molt més endreçat, amb un únic sistema d'alertes que recolliria tots els events dels sistemes remots.

Seguint amb aquesta idea, podríem fins i tot emmagatzemar en disc la informació recollida dels diferents sistemes monitoritzats per tenir-ne un històric i, per què no, visualitzar-la en forma de gràfiques estadístiques per fer-ne un estudi més exhaustiu i a la vegada còmode de totes aquestes dades.

Per implementar aquesta sol.lució integral de *recollida de dades → enviament → emmagatzematge → visualització* es poden fer servir diferents eines genèriques. La que estudiarem a continuació es diu "Elastic Stack" (https://www.elastic.co) i és una de les més utilitzades a l'hora de recollir logs (del tipus que siguin) de diferents fonts i emmagatzemar-los per tal de accedir a ells de forma gràfica còmodament des d'un navegador, que és just el que volem. Concretament, "Elastic Stack" està format pels diferents components, instal.lables de forma independent però que normalment treballen junts:

*"**FileBeat**" (https://www.elastic.co/products/beats/filebeat) : Aquest programa fa dues coses:

1) "Absorbeix" les dades d'un determinat origen local indicat, com pot ser les darreres línies que es vagin afegint a un/s determinat/s fitxer/s de log (a mode de *tail -f*), les darreres línies mostrades pel Journald, la sortida estàndar d'una determinada comanda, etc, etc, etc. i ...

2) ...aquestes dades les reenvia -per la xarxa- a un destí central que pot ser o bé LogStash o bé ElasticSearch (o un altre programa similar compatible)

NOTA: No només existeix "FileBeat" per "absorbir dades d'un origen. Depenent del tipus de dades és millor utilitzar un altre tipus de "recolector". Per exemple, si en comptes de missatges de log (és a dir, en general, cadenes) volem recollir dades numériques que es corresponen a mètriques de funcionament del sistema (com ara la temperatura de la CPU, l'espai lliure de disc, l'espai lliure de memòria, la quantitat de connexions establertes, etc) és més recomanable utilitzar el programa "MetricBeat" (https://www.elastic.co/products/beats/metricbeat). En canvi, si es volen obtenir mètriques de paquets de xarxa és més recomanable usar el programa "PacketBeat" (https://www.elastic.co/products/beats/packetbeat), etc. Per veure la llista completa de diferents tipus de "Beats" es pot consultar https://www.elastic.co/products/beats All "Beats" offers "at-least-once" guarantees, so you never lose a log line, and they use a back-pressure sensitive protocol, so it won't overload your pipeline.

*"**LogStash**" (https://www.elastic.co/products/logstash): Aquest programa també es pot fer servir per "absorbir" dades d'un origen local i, per tant, es podria prescindir llavors de l'ús de "FileBeat". No obstant, consumeix molt més recursos així que en principi és preferible delegar aquesta tasca a "FileBeat". On brilla "LogStash" és funcionant en un sistema central on es rebin tots els logs absorbits per "FileBeat" (o rebuts de molts altres origens diferents, en aquest sentit existeix moltíssima flexibilitat a l'hora de definir els "inputs" de "LogStash") per recollir-los de forma controlada en un únic punt (el que se'n diu "agregació" dels logs) i fer-ne dues coses més:

1) "Parsejar" i/o filtrar determinats logs segons diferents criteris que s'hagin especificat i ...

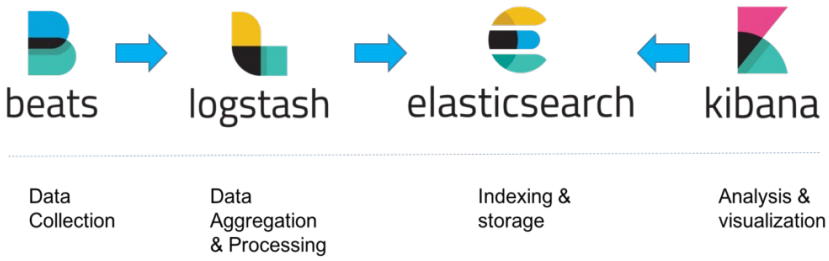
2) ...els logs ja "netejats" i filtrats els envia -per la xarxa o en local- a ElasticSearch (o a qualsevol altre tipus de destí que "LogStash" reconegui, que són uns quants)

NOTA: Seria possible que "FileBeat" enviés els logs directament a ElasticSearch sense passar per LogStach però no és gaire recomanable en relació a la "netedat" de les dades emmagatzemades. En aquest sentit, per distingir entre "FileBeat" y "LogStash" és interessant aquest paràgraf: *"Since Filebeat ships data in JSON format, Elasticsearch should be able to parse the timestamp and message fields without too much hassle. Not only that, Filebeat also supports an several modules tied to specific origins that can handle initial processing and parsing. However, as of yet, advanced log enhancement -adding context to the log messages by parsing them up into separate fields, filtering out unwanted bits of data and enriching others- cannot be handled without Logstash. Moreover, for a number of reasons, and especially in medium- and large-sized environments, you will not want to have each Filebeat agent installed on a host sending off data directly into Elasticsearch: if Elasticsearch is temporarily unavailable, back pressure to disk is not always a good solution as files can get rotated and deleted. Ideally, you would like to control the amount of indexing connections and having too many may result in a high bulk queue, bad responsiveness, and timeouts. So, in most cases, you will be using both Filebeat and Logstash"*

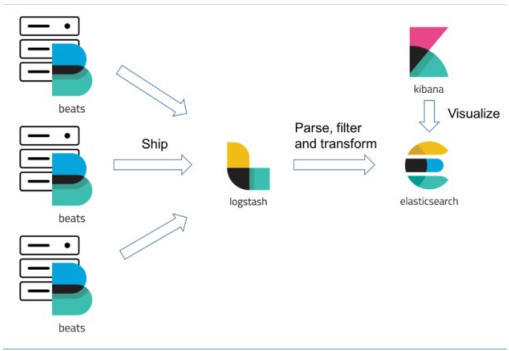
*"**ElasticSearch**" (https://www.elastic.co/products/elasticsearch): Aquest programa és el responsable d'emmagatzemar en disc les dades recollides de "LogStash" o "FileBeat". Està optimitzat per gestionar cadenes (la majoria de logs són d'aquest tipus) i la seva especialitat és la de realitzar recerques (d'aquí el seu nom) perquè indexa tot el contingut que guarda.

*"**Kibana**" (https://www.elastic.co/products/kibana) : Aquest programa és un servidor web que ofereix un panel de control molt gràfic i visual que serveix per mostrar diferents perspectives de les dades emmagatzemades a un servidor ElasticSearch.

El següent diagrama il.lustra l'acabat d'explicar:



O també aquest, on es veu que els "FileBeat" estan funcionant a màquines diferents (els sistemes a monitoritzar) i "LogStash", "ElasticSearch" i "Kibana" poden estar funcionant a una altra màquina única (o no):



Addició dels repositoris propis d'Elastic per instal.lar els diferents programes de la "suite":

*A Ubuntu cal executar les següents comandes en totes les màquines on volguem instal.lar qualsevol dels programes que formen la suite Elastic ("FileBeat", "LogStash", "ElasticSearch", "Kibana", etc):

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-6.x.list
sudo apt update
```

A partir d'aquí, a les màquines a monitoritzar es podrà instal.lar FileBeat amb la comanda...

```
sudo apt install filebeat
```

... i a la màquina "aglutinadora" de la informació recollida dels diversos "FileBeats" posats en marxa, es podrà instal.lar LogStash, ElasticSearch i,(opcionalment) Kibana -prèvia instal.lació del Java Runtime Environment-, amb les comandes:

```
sudo apt install openjdk-8-jre-headless
sudo apt install logstash
sudo apt install elasticsearch
sudo apt install kibana
```

NOTA: S'escull el paquet "openjdk-8-jre-headless" en comptes del metapaquet "default-jre-headless" perquè aquest darrer instal.la la versió 11 de Java, i Elastic actualment encara no és compatible amb aquesta versió

NOTA: Cada programa dels tres anteriors es podrien instal.lar en una màquina diferent encara que el més normal és que LogStash i ElasticSearch s'instal.lin al mateix ordinador i, habitualment, Kibana també.

*A Fedora cal executar les següents comandes (com a "root") en totes les màquines on volguem instal.lar qualsevol dels programes que formen la suite Elastic ("FileBeat", "LogStash", "ElasticSearch", "Kibana", etc)

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
cat >> /etc/yum.repos.d/elk.repo <<'EOF'
  [elasticsearch-7.x]
  name=Elasticsearch repository for 7.x packages
  baseurl=https://artifacts.elastic.co/packages/7.x/yum
  gpgcheck=1
  gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
  enabled=1
  autorefresh=1
  type=rpm-md
  EOF
```

NOTA: L'expressió <<'EOF' significa "agafa tot el que ve a continuació com a entrada fins que arribis a la línia que conté només la paraula EOF" (de End Of File, però podria ser qualsevol altra cadena)

A partir d'aquí, a les màquines a monitoritzar es podrà instal.lar FileBeat amb la comanda...

```
dnf install filebeat
```

... i a la màquina "aglutinadora" de la informació recollida dels diversos "FileBeats" posats en marxa, es podrà instal.lar LogStash, ElasticSearch i, (opcionalment) Kibana -prèvia instal.lació del Java Runtime Environment-, amb les comandes:

```
dnf install java-1.8.0-openjdk-headless
dnf install logstash
dnf install elasticsearch
dnf install kibana
```

NOTA: S'escull el paquet "java-1.8.0-openjdk-headless" en comptes del paquet "java-11-openjdk-headless" perquè aquest darrer instal.la la versió 11 de Java, i Elastic actualment encara no és compatible amb aquesta versió

NOTA: Cada programa dels tres anteriors es podrien instal.lar en una màquina diferent encara que el més normal és que LogStash i ElasticSearch s'instal.lin al mateix ordinador i, habitualment, Kibana també.

Configuració de "FileBeat":

L'arxiu de configuració de "FileBeat" és "/etc/filebeat/filebeat.yml" i és de tipus YAML.

NOTA: Les rutes i noms dels arxius de configuració dels altres "Beats" són similars: tenim "/etc/metricbeat/metricbeat.yml" per "MetricBeat", "/etc/packetbeat/packetbeat.yml" per "PacketBeat", etc . El seu contingut també és força semblant.

Un exemple de possible contingut d'aquest arxiu podria ser el següent:

```
filebeat.inputs:
- type: log
  paths:
    - /var/log/*.log
output.logstash:
  hosts: ["127.0.0.1:5044"]
```

*Entrades

La secció "`filebeat.inputs`" és on es defineixen l'entrada de dades que es vol monitoritzar. El tipus d'una entrada es defineix a la seva clau "`type`" (i, de fet, dins del fitxer de configuració pot haver definides més d'una secció "filebeat.inputs", generalment associades a entrades de diferents tipus). La clau "type" pot tenir alguns dels següents valors:

*"**log**": Indica que l'entrada de dades prové del contingut d'un (o més) fitxers de registre textual. Els "inputs" d'aquest tipus han de disposar d'una (o més) subsecció/ns "`paths`" que serveix/en per indicar precisament la ruta (o rutes, si fem servir comodins) dels diferents fitxers de log a monitoritzar. Altres parelles claus<->valor (al mateix nivell que "type" o "paths") que es poden indicar són:

`enabled: true` #Per defecte ja val "true", així que no caldria explicitar-ho si volem que funcioni

`tags: ["unaetiqueta","unaaltra"]` #Representa una llista de valor que s'afegiran de forma extra a tots els registres de dades rebuts abans de reenviar-los al destí. Les tags són útils per filtrar o també per agrupar orígens.

```
fields:
   app: "myapp"
   id: "574"
```
#Representa una conjunt de parelles clau<->valor personalitzades que s'afegiran a la resta de dades de cadascun dels registres rebuts, abans de reenviar-los al destí. Els camps són útils per identificar fluxos de dades d'entrada. Els seus valors poden ser individuals, arrays, diccionaris o qualsevol combinació dels anteriors.

NOTA: Per defecte totes les parelles apareixeran agrupades en el registre de sortida dins d'una subsecció anomenada "fields". Si es vol que apareguin "tal qual" com si fossin parelles "originals", cal establir a més l'opció `fields_under_root: true` Si es donés el cas que el nom d'alguna clau personalitzada fos el mateix que el d'alguna altra clau original, el valor que "guanya" és el de la clau personalitzada

`exclude_lines: ['^ERR']` #Representa una llista d'expressions regulars que concordaran amb les línies que es vol que FileBeat exclogui de reenviar-les al destí (per defecte no n'exclou cap). La llista d'expressions regulars admeses es pot trobar a https://www.elastic.co/guide/en/beats/filebeat/current/regexp-support.html
Notar també que es recomana escriure les expressions regulars entre cometes simples per compatibilitat amb el processador YAML

`include_lines: ['^WARN','pepito']` #Representa una llista d'expressions regulars que concordaran amb les línies que es vol que FileBeat reenvïi al destí (excloent la resta; per defecte reenvia totes). Si s'indiqués tant "exclude_lines" com "include_lines", "FileBeat" executa primer "include_lines" i després "exclude_lines" independentment de l'ordre en què s'escriguin a l'arxiu YAML

`exclude_files: ['\.gz$']` #Representa una llista d'expressions regulars que concordaran amb els fitxers ubicats dins de les carpetes indicades a la subsecció "path" que es volen ignorar (és a dir, que no es volen monitoritzar). Per defecte no s'ignora cap.

Es pot consultar https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-log.html per conèixer totes les subseccions o claus d'aquest tipus d'"inputs".

*"**stdin**": Indica que l'entrada de dades prové de l'entrada estàndar (útil quan FileBeat s'executa en primer pla per depurar el seu comportament, rebent les dades d'una canonada, per exemple). No es pot combinar amb cap altre "input". Parelles claus<->valor que es poden indicar són les conegudes `enabled, tags, fields, exclude_lines` i `include_lines` , entre d'altres. Per conèixer totes les claus d'aquests "'inputs", https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-stdin.html

*"**udp**": Indica que l'entrada de dades prové d'un equip remot. Els "inputs" d'aquest tipus han de disposar d'una subsecció "host" que serveix per indicar la *IP:port* local per on s'escoltaran les dades que arribin. Altres parelles claus<->valor ja conegudes que es poden indicar són: `enabled, tags` i `fields,` entre d'altres. Per conèixer totes les claus d'aquest tipus d'"inputs" consulta https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-udp.html

*"**tcp**": Indica que l'entrada de dades prové d'un equip remot. Els "inputs" d'aquest tipus han de disposar d'una subsecció "host" que serveix per indicar la *IP:port* local per on s'escoltaran les dades que arribin. Altres parelles claus<->valor ja conegudes que es poden indicar són: `enabled, tags` i `fields,` entre d'altres (com ara `timeout` o `ssl`). Per conèixer totes les claus d'aquest tipus d'"inputs" consulta https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-input-udp.html

There are several configuration options are used to close the FileBeat harvester after a certain criteria or time. Closing the harvester means closing the file handler. If a file is updated after the harvester is closed, the file will be picked up again after **`scan_frequency`** option has elapsed (this option has a numerical value which indicates how often -in seconds- the harvester checks for new files in the specified paths). However, if the file is moved or deleted while the harvester is closed, Filebeat will not be able to pick up the file again, and any data that the harvester hasn't read will be lost. Some of these configuration options are:

`close_inactive`: ***`5m`*** Filebeat closes the file handle if a file has not been harvested for the specified duration. The counter for the defined period starts when the last log line was read by the harvester. If the closed file changes again, a new harvester is started and the latest changes will be picked up after scan_frequency has elapsed. We recommended that you set close_inactive to a value that is larger than the least frequent updates to your log files. For example, if your log files get updated every few seconds, you can safely set close_inactive to 1m. Setting close_inactive to a lower value means that file handles are closed sooner. However this has the side effect that new log lines are not sent in near real time if the harvester is closed. You can use time strings like 2h (2 hours) and 5m (5 minutes). The default is 5m.

`close_removed : true` This option is enabled by default. If it is, Filebeat closes the harvester when a file is removed. Normally a file should only be removed after it's inactive for the time specified by `close_inactive`.

`close_eof : false` This option is disabled by default. When it is enabled, Filebeat closes a file as soon as the end of a file is reached. This is useful when your files are only written once and not updated from time to time.

*Sortides

La secció "output.*xxx*" informa a "FileBeat" del destí on ha de reenviar les dades rebudes. Aquest reenviament, però, no només és de les dades en sí sinó que aquestes s'acompanyen de certes metadades, tot plegat dins d'un objecte JSON anomenat "event". Concretament, un "event" típic disposa dels següents elements més importants:

*Una clau anomenada "**@timestamp**", el valor de la qual representa la data i hora (en format ISO) en la qual la dada ha sigut recollida per FileBeat (que no té perquè coincidir amb la data i hora de la generació real de l'event...veure més avall)

*Un objecte anomenat "**@metadata**" que bàsicament inclou dues claus: "**beat**" i "**version**". El valor de la primera és el nom del Beat generador de l'event (per nosaltres valdrà "filebeat", doncs) i el valor de la segona és la versió d'aquest Beat. En el cas de què FileBeat estigui configurat per enviar les dades recollides directaent a ElasticSearch sense passar per LogStash, aquests dos valors són importants perquè combinats s'usen per indicar el nom base de l'index del servidor ElasticSearch on s'emmagatzemarà l'event (veure més avall).

*Una objecte anomenat "**input**" que bàsicament inclou la clau "**type**", el valor de la qual indica el tipus d'"input" utilitzat ("log", "stdin",...)

*Una clau anomenada "**source**", el valor de la qual seria la ruta del fitxer d'on s'ha obtingut l'event (en el cas de què el tipus d'entrada fos "log"; en els altres casos aquesta clau no té valor).

*Una clau anomenada "**message**", el valor de la qual és el contingut del missatge recollit pròpiament dit (el "payload")

*Una objecte anomenat "**beat**" que bàsicament inclou dues claus: "**version**" i "**name**" (o "**hostname**") . El valor de la primera torna a ser la versió del Beat utilitzat i el de la segona el nom de la màquina on aquest Beat s'està executant.

*Una objecte anomenat "**host**" que bàsicament inclou tres claus: "**name**" (nom de la màquina on aquest Beat s'està executant), "**containerized**" (si aquesta màquina és un contenidor o no) i "**architecture**" (l'arquitectura de la màquina: x86_64, etc), i un objecte intern anomenat "**os**" que inclou al seu torn tres claus més: "**platform**" (nom de la distribució Linux), "**family**" (família genèrica a la que pertany aquesta distribució) i "**version**" (versió concreta detectada d'aquesta distribució). Cal dir, però, que la majoria d'aquestes claus només estaran presents si s'usa el "prospector" anomenat *add_host_metadata* (veure més avall; per defecte s'usa).

A l'exemple del principi d'aquest apartat vam escollir com a destí dels events un servidor LogStash gràcies a la secció "`output.logstash`". Tal i com es pot veure allà, l'única opció obligatòria és "`hosts`", el valor de la qual és un llista de valors "IP:nºport" corresponents als possibles servidors LogStash a connectar-s'hi (on "nºport" és el d'escolta del servidor LogStash, que per defecte és el nº 5044). Si s'indica més d'un servidor (i si l'opció de *loadbalance* és *false*, cosa que per defecte ja ho és), s'usarà un únic servidor escollit a l'atzar i, només si no respongués, s'escollirà un altre de la llista també a l'atzar. Es poden consultar a https://www.elastic.co/guide/en/beats/filebeat/current/logstash-output.html totes les opcions que proporciona aquest tipus de sortida. Entre elles podem destacar:

```
enabled: true
```
#Per defecte ja val "true", així que no caldria explicitar-ho si volem que funcioni

```
index: "hola"
```
#Indica a LogStash el nom base de l'index ElasticSearch on s'escriuran els events. El seu valor per defecte, en el cas de fer servir FileBeat, és "filebeat" (valor obtingut a partir de l'objecte @metadata). Aquest nom base sempre ve seguit del número de versió del Beat usat (és a dir, del valor del camp *%{[beat.version]}*) i la data en format YYYY.MM.DD (és a dir, del valor *%{+yyyy.MM.dd}*).
Un índex ElasticSearch es pot entendre com una "taula" d'una BD relacional o un "document" a una BD NoSQL

```
timeout: nº
```
#Indica el número de segons a esperar a una resposta del servidor LogStash abans de considerar trencada la comunicació. El seu valor per defecte és 30

```
loadbalance: true
```
#Si val "true" i s'indiquen múltiples servidors LogStash a la línia "hosts", FileBeat enviarà els events a aquests servidors balancejant la càrrega. Per defecte val "false"

Però també es poden definir altres tipus de sortides. Per exemple, es podria definir directament com a destí un servidor ElasticSearch si no es necessités un processament addicional indicant una secció "`output.elasticsearch`". Dins d'aquesta secció, l'única opció obligatòria és "`hosts`", el valor de la qual és un llista de valors "IP:nºport" corresponents als possibles servidors ElasticSearch a connectar-s'hi (on "nºport" és el d'escolta del servidor ElasticSearch, que per defecte és el nº 9200). Si s'indica més d'un servidor, FileBeat enviarà els events a aquests servidors balancejant la càrrega. Es poden consultar a https://www.elastic.co/guide/en/beats/filebeat/current/elasticsearch-output.html totes les opcions que proporciona aquest tipus de sortida. Entre elles podem destacar:

```
enabled: true
```
#Per defecte ja val "true", així que no caldria explicitar-ho si volem que funcioni

```
index: "hola"
```
#Indica el nom base de l'index ElasticSearch on s'escriuran els events si FileBeats està configurat per escriure-hi-els directament . El seu valor per defecte, en el cas de fer servir FileBeat, és "filebeat". Aquest nom base sempre ve seguit del número de versió del Beat usat (és a dir, del valor del camp *%{[beat.version]}*) i la data en format YYYY.MM.DD (és a dir, del valor *%{+yyyy.MM.dd}*). Un índex ElasticSearch es pot entendre com una taula d'una BD relacional o un "document" a una BD NoSQL

```
timeout: nº
```
#Indica el número de segons a esperar a una resposta del servidor LogStash abans de considerar trencada la comunicació. El seu valor per defecte és 90

```
protocol: http
```
#Indica el protocol que farà servir FileBeat per contactar amb el servidor ElasticSearch. Només hi ha dos valors possibles: "http" (per defecte) i "https". De fet, totes les interaccions que es fan amb un servidor ElasticSearch per disseny són de tipus API HTTP (REST)

```
schema: /ruta
```
#Indica la ruta de la URL (és a dir, tot el que ve després d'indicar-hi el nºport) que farà servir FileBeat com a base de les peticions HTTP (REST). Per exemple, si volem contactar amb un servidor ElasticSearch amb IP 192.168.1.1 escoltant al port 9200 mitjançant el protocol HTTP i tenim la línia *schema: /hola* , la URL on FileBeat realitzarà les peticions HTTP serà "http://192.168.1.1:9200/hola" Per defecte el valor d'aquesta opció és "/"

```
parameters:
   unaclau: un valor
   unaaltraclau: unaltrevalor
```
#Indica un conjunt de paràmetres que es passaran dins la URL usada, per realitzar operacions amb els índexs

```
headers:
   unaclau: un valor
   unaaltraclau: unaltrevalor
```
#Indica un conjunt de capçaleres personalitzades a incloure en la petició feta al servidor ElasticSearch

També es pot definir com a sortida, encara que no sigui molt habitual, un fitxer ubicat localment al mateix sistema on s'estigui executant FileBeat mitjançant una secció anomenada "`output.file`". Tots els events recollits es guardaran, línia a línia, en forma d'objecte JSON. Dins d'aquesta secció, l'única opció obligatòria és "`path`", el valor de la qual serà la ruta de la carpeta escollida on es guardara/n el/s fitxer/s. A https://www.elastic.co/guide/en/beats/filebeat/current/file-output.html es poden consultar totes les opcions que proporciona aquest tipus de sortida. Entre elles podem destacar:

```
enabled: true
```
#Per defecte ja val "true", així que no caldria explicitar-ho si volem que funcioni

```
filename: results.txt
```
#Nom del/s fitxer/s guardats. El valor per defecte és el nom del Beat, seguit d'un número que indica l'ordre de rotació (veure següent opció). És a dir, en el nostre cas: "filebeat", "filebeat.1", "filebeat.2", etc

```
rotate_every_kb: nº
```
#Indica el tamany màxim (en KB) de cada fitxer guardat. Quan s'arriba a aquest tamany, automàticament es fa una rotació del fitxer, convertint-se "filebeat" en "filebeat.1", "filebeat.1" en "filebeat.2", etc. El seu valor per defecte és 10240KB

```
number_of_files: nº
```
#Indica el número màxim de fitxers guardats. Quan s'arriba a aquest número, el fitxer més antic serà esborrat automàticament. El seu valor pot estar entre 2 i 1024 i el per defecte és 7.

Finalment, si FileBeat s'executa en primer pla, es poden enviar les dades recollides al terminal de la pròpia màquina on s'està executant "FileBeat" mitjançant la secció "`output.console`". En aquest cas, es pot escollir veure els events en format JSON (per defecte) o bé en forma de cadena formatada. En el primer cas (on s'utilitza, encara que no s'expliciti, el "codec" "json") podríem fer simplement...:

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

...i en el segon cas (on hem d'explicitar l'ús del "codec" "format") podríem fer, per exemple, el següent per veure només el timestamp (camp que FileBeat sempre genera per cada event i que té el nom de clau "@timestamp") i el valor d'un camp anomenat "pepito":

```
output.console:
  codec.format:
      string: '%{[@timestamp]} %{[pepito]}'
```

NOTA: L'ús del "codec.format" també és possible dins de la secció "output.file" (la qual, per defecte, també fa servir el còdec JSON).

Altres directives generals (és a dir, escrites al mateix nivell que "filebeat.inputs" o "output.*xxx*") són un altre cop `tags` i `fields` i també la parella `name: "nom"` , la qual serveix per identificar la màquina origen de les dades (si n'hi hagués diverses possibles): si no s'especifica s'usarà el *hostname*

Es poden indicar valors de variables d'entorn dins del fitxer de configuració de FileBeat. Només cal indicar-los amb la sintaxis: `${NOMVARIABLE}` Si la variable no estigués definida, el valor a substituïr seria el valor buit, a no ser que s'hagi definit un valor per defecte, així: `${NOMVARIABLE:valor_per_defecte}`

*"Processors"

While not as powerful and robust as Logstash, Filebeat can apply basic processing and data enhancements to log data before forwarding it to the destination of your choice. You can decode JSON strings, drop specific fields, add various metadata, and more. This can be done by writting a "processors" subsection (globally or inside desired "filebeat.inputs" sections), which defines one (or more) specific processes to apply to input data. Each process can be applied optionally if respective "`when`" conditional statement is used (if not present, process will always be applied). You can find listed all possible conditions in https://www.elastic.co/guide/en/beats/filebeat/current/defining-processors.html#conditions. The general pattern is like this:

```
- type: <input_type>
  processors:
  - <processor_name>:
      when:
        <condition>
      <parameters>
...
```

Some interesting "processors" (you can find all the existing ones in https://www.elastic.co/guide/en/beats/filebeat/current/defining-processors.html#processors) are:

"**drop_fields**" : specifies which fields to drop before resend them to output (if a certain -and optional- condition is fulfilled; if it's missing, the specified fields are always dropped). Its pattern configuration is:

```
processors:
 - drop_fields:
       when:
         condition
       fields: ["aKey","anotherKey",...]
```

Below is an example using this processor for (in this case, unconditionally) dropping some fields (in this case, from Apache access logs):

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - "/var/log/apache2/access.log"
  fields:
    apache: true
  processors:
  - drop_fields:
      fields: ["verb","id"]
output.logstash:
  hosts: ["127.0.0.1:5044"]
```

"**drop_events**" : It drops the entire event if the associated condition is fulfilled. The condition is mandatory, because without one, all the events are dropped. Its pattern configuration is:

```
processors:
 - drop_event:
       when:
         condition
```

"**add_locale**" : It enriches each event with the machine's time zone offset from UTC via a new "beat.timezone" value. Its configuration should be:

```
processors:
  - add_locale: ~
```

"**add_host_metadata**" : It enriches each event adding an extra "host" object containing relevant metadata aobut the origin host, like its architecture or OS. If its option *netinfo.enabled* is true, besides, this "host" object will contain too the origin machine's IP and MAC addresses. So its configuration should be:

```
processors:
  - add_host_metadata:
       netinfo.enabled: true
```

Other interesting "processor" is, for instance, "**rename**" (for renaming received fields' keys before resending) or "add_host_metadata", entre molts d'altres.

*Final

Un cop ja configurat "FileBeat", ja es pot posar en marxa per començar a recol.lectar la informació definida: *sudo systemctl start filebeat && sudo systemctl enable filebeat*

D'altra banda, podreu trobar tota la documentació oficial de "FileBeat" a https://www.elastic.co/guide/en/beats/filebeat/current

EXERCICIS:

1.-a) Arrenca dues màquines virtuals qualssevol amb les seves respectives tarjetes de xarxa en mode "adaptador pont". En una d'elles (l'anomenarem "MaquinaClient") instal.la Sysdig i FileBeat i en l'altra (l'anomenarem "MaquinaServidora") instal.la LogStash, ElasticSearch i Kibana. Seguidament, apaga "MaquinaServidora" (de moment no la farem servir).

b) Edita el fitxer de configuració de FileBeat de "MaquinaClient" per tal de què tingui un únic "input" activat que sigui de tipus "stdin" i un únic "output" activat que sigui de tipus "console" (funcionant amb el "codec" JSON). Un cop gravat aquest fitxer, assegura't de què a "MaquinaClient" no estigui funcionant el servei FileBeat (perquè l'executarem interactivament des del terminal al punt següent).

c) Concretament, executa la comanda *echo Hola | /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml* ¿Què veus a pantalla?
NOTA: El binari filebeat no s'instal.la en cap carpeta dins del PATH sinò que es troba dins de "/usr/share/filebeat/bin"; és per això que cal indicar-la sencera. D'altra banda, el paràmetre -c serveix per indicar l'arxiu de configuració que es farà servir en aquesta execució concreta del binari.

d) Ara canvia l'"output" activat de FileBeat per a què sigui de tipus "file". Concretament, fes que els events es guardin a l'arxiu "/var/filebeat". Seguidament executa la comanda *sysdig -p "User:%user.name Proc: %proc.name File:%fd.name" "evt.type=openat" | /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml* i espera uns quants segons. Finalment, pulsa CTRL+C. ¿Quina informació és la que recull Sysdig? ¿Quin és el contingut de l'arxiu /var/filebeat?

e) Executa ara la comanda *journalctl -f -p info | /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml* i espera uns quants segons. Torna a pulsar CTRL+C i torna a observar (el final d)el contingut de l'arxiu /var/filebeat. ¿Què veus ara? ¿Recordes per a què servien els paràmetre -f i -p de *journalctl*?

Configuració de "LogStash":

One of the things that makes Logstash so powerful is its ability to aggregate logs and events from various sources. Using more than 50 **input** plugins for different platforms, databases and applications, Logstash can be defined to collect data from these sources, process them with several filters and send them to other systems for storage and analysis. The most common inputs used are "beats", "file", "http", "tcp", "udp" and "stdin" but you can ingest data from plenty of other sources (Redis, AQMP, Kafka, etc, etc). If you do not define an input, Logstash will automatically create a stdin input

As with the inputs, Logstash supports a number of **output** plugins that enable you to push your data to various locations, services, and technologies. You can store events using outputs such as File, CSV, and S3, convert them into messages with IMAP, RabbitMQ and SQS, or send them to various services like ElasticSearch, Redis, MongoDB, Kafka, IRC or even online ones, like HipChat or PagerDuty, etc, etc. The number of combinations of inputs and outputs in Logstash makes it a really versatile event transformer. If you do not define an output, Logstash will automatically create a stdout output.

The **filter** stage seats between input and output ones and it's the responsible of parsing, filtering, transforming and enrichying the incoming data before outputting it . This is where you'll find most of Logstash's value.

So every Logstash configuration file will generally have three sections: inputs, outputs and filters, following this structure (each of these sections will contain zero or more plugin configurations, and there can be multiple blocks):

```
input { }
filter { }
output { }
```

Logstash finds its configuration files inside the "/etc/logstash/conf.d" folder and, although they can be named either way, they must have the ".conf" extension. Each configuration file should be related to some kind of monitored source application (for instance, you could have an "apache.conf" file, a "mysql.conf" file, etc). Each of those files will contain one "input" section, one "filter" section and one "output" section to perform its function. It's important to know, though, that Logstash reads the all files located in "/etc/logstash/conf.d" in lexical order.

> NOTA: En realitat, la configuració interna del propi programa LogStash com a tal es troba a l'arxiu /etc/logstash/logstash.yml. Allà podem trobar línies com" path.config" (que indica la ruta de la carpeta on trobar els fitxers .conf descrits al paràgraf anterior -per defecte val "/etc/logstash/conf.d"-), "path.logs" (que indica la ruta dels arxius de registre generats pel propi programa -per defecte val "/var/log/logstash"-), etc. En general, els valors per defecte ja són prou bons així que no els modificarem (de fet, molts d'ells es poden alterar directament via paràmetres del binari)

*Input config

A typical input section can look like this:

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

This tells Logstash to open the "**beats**" input plugin on port 5044 to receive incoming data from there. Opcionalment, a més de l'opció `port` (obligatòria), es pot indicar l'opció `host` , la qual tindrà com a valor una cadena que representa la IP per on escoltarà LogStash si el sistema on s'està executant tingués més d'una IP asssignada (per defecte escolta per la "0.0.0.0", que vol dir per totes). També és interessant l'opció `client_inactivity_timeout` , la qual té com a valor el número de segons que LogStash s'esperarà sense rebre dades d'un Beat determinat per tancar la connexió. Per conèixer totes les opcions que ofereix aquest plugin, es pot consultar https://www.elastic.co/guide/en/logstash/current/plugins-inputs-beats.html

És possible fer, gràcies a un plugin d'entrada de LogStash anomenat "**file**" (https://www.elastic.co/guide/en/logstash/current/plugins-inputs-file.html) que sigui el propi Logstash qui obtingui directament les dades de fitxers de registres locals (per defecte les llegeix des del final com FileBeat però això es pot canviar, i les espera en format JSON -concretament un objecte per línia- perquè el "codec" emprat per defecte per aquest plugin és aquest: https://www.elastic.co/guide/en/logstash/current/plugins-codecs-json.html). Però aquesta forma de funcionar "suplanta" la funcionalitat de "FileBeat" i això vol dir que s'hauria d'instal.lar LogStash directament a la màquina monitoritzada, cosa que en principi aquí no farem.

> NOTA: Si les dades d'entrada no tinguessin format JSON i no s'hagués especificat cap "codec" (és a dir, que LogStash estigués utilitzant el codec JSON, automàticament aquest codec no interpretarà les dades d'entrada i les tractarà com a text pla simplement.

També és possible fer que LogStash obtingui els events directament de l'entrada estàndar (en aquest cas també estaríem suplantant la funcionalitat de FileBeat) amb el plugin d'entrada anomenat "**stdin**" (https://www.elastic.co/guide/en/logstash/current/plugins-inputs-stdin.html) , el qual pot especificar-se sense cap paràmetre (és a dir, simplement així: `input { stdin { } }`). No obstant, aquest plugin està dissenyat per funcionar preferentment si LogStash s'executa en primer pla (obtenint així l'entrada a través d'una canonada); si, en canvi, executem LogStash com a servei en segon pla (que serà el més habitual amb diferència), la seva implementació es fa molt complicada i és per això que la descartarem.

Altres plugins d'entrada interessants són "**http**" (per rebre events via HTTP, https://www.elastic.co/guide/en/logstash/current/plugins-inputs-http.html), "**tcp**" (per rebre events via TCP, https://www.elastic.co/guide/en/logstash/current/plugins-inputs-tcp.html), "**udp**" (per rebre events via UDP, https://www.elastic.co/guide/en/logstash/current/plugins-inputs-udp.html) o també "**exec**" (per executar periòdicament una comanda concreta i capturar la seva sortida com a event -concretament, al seu camp "message"-, https://www.elastic.co/guide/en/logstash/current/plugins-inputs-exec.html). En qualsevol cas, per veure la llista completa de tots els "input" plugins amb les seves respectives opcions, consulteu https://www.elastic.co/guide/en/logstash/current/input-plugins.html Però ens centrarem només en les entrades provinents de FileBeat.

*Filter config

The "filter" section is where you transform your data into something that's newer and easier to work with. Before studying different filter plugins, though, you should know that:

> **a)** The syntax to access to the value of a simple field (already getched by a input plugin), wherever it is inside a "filter" or "output" section, is `[fieldname]`, although if you are referring to a top-level field, you can omit the `[]` and simply use `fieldname`.
>
> **b)** The syntax to access to the value of a nested field is the full path to that field:
> `[top-level field][nested field]`.
>
> **c)** Anyway, if you want to put a field value into a string, you must specify it like this:
> %{[fieldname]} , instead.
>
> NOTA: Similarly, you can convert the timestamp in the @timestamp field into a string like this:
> `%{+yyyy.MM.dd.HH.mm.ss}` or any other format compatible with
> http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html

Another aspect you should know is the conditionals syntax. When you only want to filter or output an event under certain conditions, you can use a conditional, which looks and acts the same way they do in programming languages. You can use `if`, `else if` and/or `else` statements, and they can be nested without limit. The conditional syntax is:

```
if EXPRESSION {
  ...
} else if EXPRESSION {
  ...
} else {
  ...
}
```

What's an expression? Comparison tests, boolean logic, and so on. You can use the following comparison operators: related to equality (`==`, `!=`, `<`, `>`, `<=`, `>=`), related to regexps -that's is: used to check a pattern on the right against a string value on the left- (`=~`, `!~`) and related to inclusion in a list (`in`, `not in`). You can use also these boolean operators to concatenate expressions (`and`, `or`, `nand`, `xor`) and the negation operator: `!` Note regular expressions must be enclosed between slashes ("/") and string between quotes (").

Here are a few examples of filters that accomplish different goals (per veure la llista completa de tots els "filter" plugins i les seves respectives opcions, consulteu https://www.elastic.co/guide/en/logstash/current/filter-plugins.html):

"**date**" : Many logging systems emit a timestamp. But by default, the @timestamp of the event is when it was *ingested*, which could be seconds, hours, or even days later of the emission of the log. This plugin parses that emission timestamp and sets the @timestamp of the event to be that emission time. Its most important option is "match", which can be used like this:

```
filter {
   date {
      match => [ "fieldwhichhasemissiontimevalue", "MMM dd-yyyy HH:mm:ss"]
   }
}
```

where second value of this array is the format in which field (indicated as first value) must be parsed. Possibles characters to put inside this second value are: "yyyy" (full year number), "M" (month number with minimal possible digits -for instance, 1 for January and 12 for December-), "MM" (month number, zero-padded if needed -for instance, 01 for January and 12 for December-), "MMM" (abbreviated month text -for instance, "Jan" for January, if locale is English-), "d" (day number with minimal possible digits), "dd" (day number, zero-padded if needed), "H" (hour number with minimal possible digits), "HH" (hour number, zero-paded if needed), "m" (minute number with minimal possible digits), "mm" (minute number, zero-padded if needed), "s" (second number with minimal possible digits), "ss" (second number, zero-padded if needed), "S" (tenths of a second), "SS" (hundredths of a second), "SSS" (thousandths of a second). Note this second value can be alternatively the string "ISO8601" (in this case the first value should be parsed as any valid ISO8601 timestamp, such as "2019-04-19T03:33:01.103Z") or the string "UNIX" (in this case the first value should be an integer representing the number of seconds from 1-1-1970 to reach the desired time)

Another interesting option of *date* filter is `target => "alternativefieldto@timestamp"` which tells to store the emited timestamp into the given target field instead of @timestamp field (which is the one updated by default). For more information about the options of this filter, see https://www.elastic.co/guide/en/logstash/current/plugins-filters-date.html

"**geoip**" : This plugin looks up IP addresses, derives geographic location information from them and adds that location information to logs. Its most important configuration options are `source => "fieldcontainingtheIPaddress"` and `target => "newfieldwhereegeoipdatewillbestored"` See https://www.elastic.co/guide/en/logstash/current/plugins-filters-geoip.html for more information about the options of this filter.

"**csv**" : This filter takes an event field (by default, "message" field but this can change with the `source => "fieldwithcsvdata"` option) containing CSV data, parses it, and stores it as individual fields (whose names can optionally be specified with the `columns =>`

`["1rstfieldname","2ndfieldname",...]` option following the order they appear in the csv; if this option is not specified, the default field names will be "column1", "column2", etc). This filter can also parse data with any separator, not just commas (default separator) using the `separator => "customseparatorcharacter"` option. For more information about the options of this filter, see https://www.elastic.co/guide/en/logstash/current/plugins-filters-csv.html

"**json**" : This filter takes an event field (specified with the `source => "fieldwithjsondata"` option) containing JSON data, parses it, and stores every key<->value pair as individual fields in event's root. By using the `target => "newfield"` option, however, all these individual fields will be put inside the specified "box field" instead of hanging directly from event's root. See see https://www.elastic.co/guide/en/logstash/current/plugins-filters-json.html for more information about the options of this filter,

"**mutate**" : This filters allows you to perform general mutations on fields: you can rename, remove, replace, and modify fields in your events. For instance, `add_field => { "fieldname" => "fieldvalue" }` adds a new field (with indicated value) to event, `add_tag => [ "tagname" ]` adds a new tag to event, `remove_field => [ "fieldname" ]` removes the indicated field/s from event, `remove_tag => [ "tagname" ]` removes the indicated tag/s from event, `uppercase => ["fieldname"]` uppercases the value of indicated field (there is `lowercase` also), `rename => ["oldfieldname","newfieldname"]` renames the indicated field (overwritting the new field'svalue if it already exists), `split => ["fieldname","separatorcharacter"]` splits the value of field indicated in first place in several values of an array using as a separator the character indicated in second place, `gsub => ["fieldname","regexpr","string"]` substitutes all ocurrences of the regular expression indicated in second place with string indicated in third place inside field indicated in first place's value (which must be of string type) -for instance, *gsub => ["path", "/", "_", "name", "[\\?#-]",""]* replaces all forward slashes with underscores inside "path" field's value and replaces all backslashes, question mark, hashes and minus with anything inside "name" field's value- , etc, etc. There are many other possibilities (`join, coerce, copy, strip, update, capitalize...`), all listed in https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html

You can control the order of mutate actions by using separate mutate blocks, one below another. For instance, next example change the initial value of "hostname" field from "name.domain.tld" to only "NAME":

```
filter {
    mutate {
        split => ["hostname", "."]
        add_field => { "shortHostname" => "%{hostname[0]}" }
    }
    mutate {
        rename => ["shortHostname", "hostname" ]
         uppercase => ["hostname"]
    }
}
```

In next example if the "program" field, has the value "metrics_fetcher", then this event is tagged with "metrics" value. This tag could be used in a later filter plugin to further enrich the data.

```
filter {
   if [program] == "metrics_fetcher" {
         mutate {
          add_tag => [ "metrics" ]
         }
   }
}
```

"**kv**" : This filter turns strings like "key1=value1 key2=value2" present in "message" field (or in another field if it is indicated with the `source => "anotherfield"` option) into individual pair of key<->value you can perform operations on. If other character different from "=" is used to bind key with value, it can be specified with the `value_split => "separatorcharacter"` option. Moreover, by using the `target => "boxfield"` option, however, all these individual fields will be put inside the specified "box field" instead of hanging directly from event's root. In https://www.elastic.co/guide/en/logstash/current/plugins-filters-kv.html there are many other options listed (like `default_keys, exclude_keys, include_keys, field_split,` etc)
For example, if you have a log message which contains "ip=1.2.3.4 error=REFUSED", you could parse those automatically to obtain the fields {ip:1.2.3.4, error:REFUSED} simply by configuring:

```
filter { kv { } }
```

Another exemple where conditionals are used is the following one, which runs only if "is_a_metric" is in the list of tags. These new keys are placed as sub-fields of the "metrics" field, allowing, for instance the text "*pages_per_second=42 faults=0*" (value of "message" field) to become metrics.pages_per_second = 42 and metrics.faults = 0 key<->value pairs on the event:

```
filter {
   if "is_a_metric" in [tags] {
     kv {
       source => "message"
       target => "metrics"
     }
   }
}
```

"**grok**" : This filter tries to match a line against a regular expression, map specific parts of the line into dedicated fields, and perform actions based on this mapping. In other words, if you need to translate strings like "The accounting backup failed" into something that will pass `if [backup_status] == 'failed'`, this will do it. So, in summary, it allows you to do is give structure to unstructured logs.

Here is the basic syntax format for a Logstash grok filter: %{PATTERN:FieldName} This will match the predefined pattern and map it to a specific field (which could exist previously or not). There are many built-in patterns that are supported out-of-the-box by Logstash for filtering items such as words, numbers, and dates (the full list of supported patterns can be found here: https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns).
Examples of predefined patterns are: INT, NUMBER, WORD, USERNAME, EMAILADDRESS, IP, HOSTNAME, IPORHOST (equivalent a "ip" o bé a "hostname"), HOSTPORT (unió d'"iporhost" + ":" + número sencer), MAC, URIPROTO, URIHOST, URIPATH, URIPARAM, URIPATHPARAM (unió de les dues anteriors), URI (unió d'"uriproto", "urihost" i "uripathparam"), YEAR, MONTHNUM, MONTHDAY, HOUR, MINUTE, SECOND, TIME (combinació d'"hour", "minute" i "second" amb ":"), DATE (combinació de "year", "monthnum" i "monthday" tant en format americà com europeu), PATH, etc.

The main option of grok filter is `match => { "sourcefieldname" => "%{PATTERN:destinationfieldName}" }` For example, the following will match an existing value in any place inside the "message" field for the given pattern, and if a match is found will add the field "duration"to the event with the captured value:

```
filter {
 grok { match => { "message" => "Dur: %{NUMBER:duration}" } }
}
```

If you need to match the pattern only if it is in the beginning of field's value, you can precede pattern expression with the "^" symbol and if you need to match the pattern only if it is in the end, you can finish pattern expressin with the "$" symbol. For instance, next example will only match lines which are EXACTLY "Dur: nº" without anything before or after:

```
filter {
 grok { match => { "message" => "^Dur: %{NUMBER:duration}$" } }
}
```

If you need to match multiple patterns against a single field without forcing an order between them inside the inspected field, the value can be an array of patterns:

```
filter {
 grok {
   match => { "message" => [ "Dur: %{NUMBER:duration}","Speed: %{NUMBER:speed}" ]}
 }
}
```

Take this random log message for example: "*2016-07-11T23:56:42.000+00:00 INFO [MySecretApp.com.Transaction.Manager]:Starting transaction for session-464410bf-37bf-475a-afc0-498e0199f008*" The grok pattern we will use looks like this (note claudators are scaped):

```
filter {
 grok {  match => { "message" =>"%{TIMESTAMP_ISO8601:timestamp}
%{LOGLEVEL:log-level} \[%{DATA:class}\]:%{GREEDYDATA:message}" }
  }
}
```

After processing, the log message will be parsed as follows

```
{
 "timestamp" => "2016-07-11T23:56:42.000+00:00",
 "log-level" =>"INFO",
 "class" =>"MySecretApp.com.Transaction.Manager"
 "message" => "Starting transaction for session-464410bf-37bf-475a-afc0-498e0199f008"
 }
```

This is how Elasticsearch indexes the log message. Sorted out in this format, the log message has been broken up into logically-named fields which can be then queried, analyzed and visualized more easily.

NOTA: Logstash has a nice feature when it fails to find the desired pattern in a log line: it tags this line with `_grokparsefailure` value, which can be useful for debugging. For instance, we'll tell it to write events that didn't match to an external file that we can check at will:

```
output {
  if "_grokparsefailure" in [tags] {
       file { "path" => "/home/vagrant/grok_failures.txt" }
  } else {
         stdout { codec => rubydebug }
  }
}
```

If you cannot find the pattern you need, you can write your own custom pattern: since grok is essentially based upon a combination of regular expressions, you can also create your own regex-based grok filter. The reference of valid regular expression symbols is https://github.com/kkos/oniguruma/blob/master/doc/RE For example (?<pepe>\d\d-\d\d-\d\d) will match the regular expression of 22-22-22 (or any other digit) to the value of field "pepe". To help in the creation of custom regular expression you can use online wizards like https://grokdebug.herokuapp.com or http://grokconstructor.appspot.com/

For more information about the options of this filter, see
https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

NOTA: There is plugin called "**dissect**" (https://www.elastic.co/guide/en/logstash/current/plugins-filters-dissect.html) which extract values from logs in a similar way than grok but without using regular expressions, so it can be faster than grok when events are structured

*Output config

LogStash would like you to send it all into ElasticSearch, but anything that can accept a JSON document, or the datastructure it represents, can be an output. In fact, events can be sent to multiple outputs. Per veure la llista completa de tots els "output" plugins amb les seves respectives opcions, consulteu https://www.elastic.co/guide/en/logstash/current/output-plugins.html

If we want LogStash to send it all into ElasticSearch, we must use ElasticSearch output, like this (if several hosts are specified in the array LogStash will load balance requests across them)

```
output {
    elasticsearch { hosts => ["localhost:9200"] }
}
```

In https://www.elastic.co/guide/en/logstash/current/plugins-outputs-elasticsearch.html are listed all options belonging to this type of output. We will talk about them in next chapter .

Other interesting outputs are:

*"**stdout**" (https://www.elastic.co/guide/en/logstash/current/plugins-outputs-stdout.html) Many times this output (which only works if LogStash is running in foreground) is used without options, like this: `output { stdout { } }`. Anyway, its most used option is `codec` to customize the output (by default the codec used is JSON). Possible values of this option are the following ones (but there are much more, see https://www.elastic.co/guide/en/logstash/current/codec-plugins.html):

```
codec => rubydebug #A pretty and coloured output

codec => line { format => "Hello: %{message}"} #A custom-formatted output
```

NOTA: As we've already seen, Codecs can be used in both inputs and outputs. Input codecs provide a convenient way to decode your data before it enters the input. Output codecs provide a convenient way to encode your data before it leaves the output. As we will use FileBeats as input, the default input code (which is JSON) is fine for us because is emitted events by FileBeats are formatted in JSON, so we won't play many time with input codecs. But with output codecs we will do because we will try several different output in exercices (ElasticSearch, stdout, file...)

NOTA: Only the "rubydebug" codec allows you to show the contents of the @metadata field if it configured like this: `stdout { codec => rubydebug { metadata => true } }` Make use of the @metadata field any time you need a temporary field but do not want it to be in the final output. For instance:

```
input { stdin { } }
filter {
  mutate { add_field => { "show" => "This data will be shown" } }
  mutate { add_field => { "[@metadata][test]" => "Hello" } }
  mutate { add_field => { "[@metadata][no_show]" => "This data will not be shown" } }
}
output {
  if [@metadata][test] == "Hello" {
     stdout { codec => rubydebug }
  }
}
```

*"**file**" (https://www.elastic.co/guide/en/logstash/current/plugins-outputs-file.html) This ouput provides the same `codec` option as "stdout" but also it requires a mandatory option `path => "/path/filename"` (where event fields can be used to form the file name)

*"**csv**" (https://www.elastic.co/guide/en/logstash/current/plugins-outputs-csv.html) This output is very similar to "file" but it has one more mandatory option: `fields => ["fieldname", "[nested][field]",...]` which specifies the field names from the event that should be written to the CSV file. Fields are written to the CSV in the same order as the array. If a field does not exist on the event, an empty string will be written. Another interesting option is the object `csv_options =>`

`{"col_sep" => "\t"}` which specifies several keys for building the customized CSV output, like "col_sep" to indicate the separator character (by default is the colon).

*"**exec**" (https://www.elastic.co/guide/en/logstash/current/plugins-outputs-exec.html) This output will run a specific command for each event received. You can pass fields from the event to the child process. For example:

```
output {
  if [type] == "abuse" {
    exec {
      command => "iptables -A INPUT -s %{clientip} -j DROP"
      quite => false #If false, it displays the result of the command to the terminal
    }
  }
}
```

Altres plugins de sortida interessants són "**http**" (per enviar events -per defecte en format JSON si no es canvia el "codec"- via HTTP/S, https://www.elastic.co/guide/en/logstash/current/plugins-outputs-http.html), "**tcp**" (per enviar events -per defecte en format JSON si no es canvia el "codec"- via TCP, https://www.elastic.co/guide/en/logstash/current/plugins-outputs-tcp.html), "**udp**" (per enviar events -per defecte en format JSON si no es canvia el "codec"- via UDP, https://www.elastic.co/guide/en/logstash/current/plugins-outputs-udp.html) , "**pipe**" (per enviar events -per defecte en format JSON si no es canvia el "codec"- a l'entrada estàndar del programa que s'indiqui), "**email**" (per enviar correus per cada event rebut, ja sigui mitjançant el sistema integrat que porta LogStash o bé, si es configura, a través d'algun servidor SMTP local com ara Postfix, etc, https://www.elastic.co/guide/en/logstash/current/plugins-outputs-email.html), "influxdb" (per enviar events a un servidor Influx, https://www.elastic.co/guide/en/logstash/current/plugins-outputs-influxdb.html), etc. Qualsevol d'aquests plugins es poden utilitzar incondicionalment o bé segons condicions if/else determinades.

*Final

Un cop ja configurat "LogStash", ja es pot posar en marxa per començar a processar la informació rebuda dels "inputs" definits" i reenviar-la als "outputs" definits: *sudo systemctl start logstash && sudo systemctl enable logstash*

D'altra banda, podreu trobar tota la documentació oficial de "LogStash" a https://www.elastic.co/guide/en/logstash/current/index.html i un tutorial força complet i entenedor a https://www.elastic.co/blog/a-practical-introduction-to-logstash

EXERCICIS:

1.-a) Edita el fitxer de configuració de FileBeat de "MaquinaClient" per tal de què tingui un únic "input" activat que sigui de tipus "stdin" i un únic "output" activat que sigui de tipus "logstash". Un cop gravat aquest fitxer, comprova que el servei FileBeat no estigui iniciat i seguidament executa la comanda *(while true; do echo Hola; sleep 2; done) | sudo /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml* ¿Què creus que està fent aquesta comanda? No l'aturis

b) Crea ara un fitxer de configuració de LogStash de "MaquinaServidora" per a què tingui aquest contingut:

input { beats { port => 5044 } }
output { stdout { codec => rubydebug } }

Un cop gravat aquest fitxer, comprova que el servei LogStash no estigui iniciat, que les regles del tallafoc no interfereixin (fes sudo *iptables -F* per si de cas) i seguidament executa la comanda *sudo -u logstash /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/nom_fitxer.conf* ¿Què veus a pantalla? Un cop vist, para aquest Logstash interactiu pulsant CTRL+D

NOTA: El binari logstash no s'instal.la en cap carpeta dins del PATH sinò que es troba dins de "/usr/share/logstash/bin"; és per això que cal indicar-la sencera. D'altra banda, el paràmetre -f serveix per indicar l'arxiu de configuració que es farà servir en aquesta execució concreta del binari.

NOTA: Un altre paràmetre interessant del binari logstash és -e, el qual serveix per indicar una configuració "al vol" sense haver de fer servir cap fitxer de configuració concret. Això va bé per fer proves ràpides. Per exemple: */usr/share/logstash/bin/logstash -e "input { stdin { } output { stdout { } }"*

c) Modifica el contingut del fitxer de configuració anterior per a què ara tot allò rebut de FileBeat en comptes de mostrar-ho per pantalla, ho gravi en el fitxer "/var/pepe" (que ha de tenir permisos 666!). Un cop feta aquesta modificació, inicia ara el servei Logstash (i per tant, no interactiu) amb *sudo systemctl start logstash* Espera't uns segons per executar *sudo tail -f /var/pepe*

NOTA: Si a /var/pepe no apareix res executa les següents comandes com a root i reinicia el servei *logstash: chown -R logstash:root /etc/logstash/conf.d &&* chmod 0750 /etc/logstash/conf.d && chmod 0640 /etc/logstash/conf.d/*

2.-a) Ja hem vist a la teoria que LogStash (executat directament des del terminal com a l'apartat b) de l'exercici anterior) és capaç de llegir de l'entrada estàndar (via canonada). Sabent això, ¿quina comanda seria una bona candidata per a redireccionar la seva sortida a LogStash sabent que aquest té la configuració següent? ¿Quin serà el valor del camp "message" a la sortida mostrada per LogStash? ¿I el del camp "maquina"? ¿I el del camp "geoip"?

```
input { stdin {} }
filter {
    grok { match => {"message" => "%{IP:maquina}"} }
    geoip { source => "maquina" }
}
output { stdout { codec => rubydebug } }
```

b) Suposant que a "MàquinaServidora" executes la següent comanda (sense que el servei LogStash estigui funcionant): *sysdig -t a "not(proc.name = sysdig)" | sudo /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/sysdig.conf* (fitxa't en el significat dels paràmetres del Sysdig!) i que el contingut del fitxer "sysdig.conf" és aquest, ¿què veus a pantalla? ¿Quina tasca fa l'if i els plugins *kv* i *mutate* que hi ha dins?

```
input { stdin { } }
filter {
  grok {
    match => { "message" => "^%{NUMBER:num} %{NUMBER:time} %{INT:cpu} %{NOTSPACE:procname} %{NOTSPACE:tid}
(?<direction>[<>]) %{WORD:event} %{DATA:args}$" }
  }
  if [args] {
    kv { source => "args" }
    mutate {  remove_field => ["args"]  }
  }
}
output {  stdout { codec => rubydebug } }
```

c) Vés a https://grokdebug.herokuapp.com i comprova, amb les opcions "single" i "named captures only" activades, si el patró següent...:

```
%{MONTH:[eventossh][mes]} %{MONTHDAY:[eventossh][dia]} %{TIME:[eventossh][hora]}
%{IPORHOST:[eventossh][maquina]} sshd(?:\[%{POSINT:[eventossh][pid]}\])?: %{GREEDYDATA:[eventossh][mensaje]}
```

... serveix per reconèixer cada camp de línies com aquesta:

```
Dec 12 12:32:58 localhost sshd[4161]: Disconnected from 10.10.0.13 port 55769
```

Configuració de "ElasticSearch":

L'arxiu de configuració d'"ElasticSearch" és "/etc/elasticsearch/elasticsearch.yml" i és de tipus YAML. Allà es poden editar diferents dades com ara el nom del servidor (directiva "node.name"), la IP (directiva "network.host") i port d'escolta (directiva "http.port", que per defecte val 9200), les rutes on es guardaran les dades, les rutes dels fitxers de registre del programa, els números d'afinament d'ús de memòria, etc però en general els valors per defecte ja estan prou bé.

> NOTA: Many times you will want to restrict outside access to your Elasticsearch instance to prevent outsiders from reading your data or shutting down your Elasticsearch cluster through the REST API. To achieve this you should uncomment the line "network.host" and replace its value with "localhost". Then, to get Elasticsearch server from remote clients you should implement a reverse web proxy (like Nginx or Apache) to redirect remote requests to local Elasticsearch instance.

Un cop instal.lat "ElasticSearch", ja es pot posar en marxa per començar a indexar i emmagatzemar la informació rebuda de "LogStash": *sudo systemctl start elasticsearch && sudo systemctl enable elasticsearch* A continuació, es pot realitzar una consulta a http://127.0.0.1:9200 bé mitjançant el navegador o bé mitjançant *curl* per veure que efectivament, ElasticSearch respon. És a dir, en executar...:

curl http://127.0.0.1:9200

...haurem de veure metadades informatives sobre el propi servidor ElasticSearch en forma d'objecte JSON.

Les dades que emmagatzemarà ElasticSearch poden provenir de múltiples orígens: directament de Beats, processats des de LogStash, etc. En aquest apartat suposarem aquest darrer cas, així que haurem de tenir la configuració del "output" de LogStash establerta de la següent manera:

```
output {
    elasticsearch { hosts => ["localhost:9200"] }
}
```

Índices

The process of adding data to Elasticsearch is called "indexing." This is because when you feed data into Elasticsearch, the data is placed into Apache Lucene (https://lucene.apache.org/core) indexes. This makes sense because Elasticsearch uses the Lucene indexes to store and retrieve its data. Although you do not need to know a lot about Lucene, it does help to know how it works when you start getting serious with Elasticsearch.

Un índice ElasticSearch es asimilable en concepto al de una tabla donde sus filas/registros se llaman "documentos" JSON y se corresponden a cada uno de los eventos recibidos del exterior. La gran diferencia con las tablas relacionales es que los índices no tienen estructura rígida: cada documento puede disponer de diferentes columnas/campos (es decir, parejas clave←>valor) independientemente de los demás documentos almacenados en el índice.

Todos los eventos provenientes de LogStash, en el momento de enviarse a ElasticSearch se autoasignan a un índice llamado por defecto "logstash-%{+YYYY.MM.dd}". Lo que quiere decir que todos los eventos, de entrada, se almacenan clasificados por día. Esto se puede cambiar indicando otro valor para la propiedad "index" del output de Logstash diferente del usado por defecto, así por ejemplo (en este caso, se crean nuevos índices a cada hora indicando en su nombre que los datos provienen de tal FileBeat con tal versión):

```
output {
    elasticsearch { hosts => ["127.0.0.1:9200"] }
    index => "pepito-%{[@metadata][beat]}-%{[@metadata][version]}-%{+HH}"
}
```

Consultas GET

Para ver la lista de los índices actualmente almacenados en ElasticSearch se puede ejecutar el siguiente comando tal cual:

curl http://127.0.0.1:9200/_cat/indices

o, si se quiere observar en modo verboso (esto es, mostrando el título de cada dato):

curl http://127.0.0.1:9200/_cat/*indices?v*

Es interesante fijarse, entre la información mostrada con el comando anterior, en la columna "docs.count", que indica la cantidad de documentos que tiene cada índice, además de otras como "docs.deleted", "store.size", etc.

NOTA: Otra consulta que podemos hacer es *curl http://127.0.0.1*:9200/_cat , la cual nos ofrece una lista de otras categorías de objetos que podemos explorar, como "count", "health", "aliases", etc.

Para ver el contenido almacenado dentro de un determinado índice se puede ejecutar el siguiente comando:

curl http://127.0.0.1:9200/nombre-indice/_search

aunque para verlo de forma más cómoda se suele utilizar un parámetro modificador así:

curl http://127.0.0.1:9200/nombre-indice/_search?pretty

NOTA: Se puede usar el comodín * en el nombre del índice para buscar en más de uno o incluso escribir el nombre especial "_all" para indicar todos los índices existentes. También se pueden indicar varios nombres de índices concretos, separados por comas, para buscar en ellos en concreto (así: *http://127.0.0.1:9200/nombre-indice1,nombre-indice2/_search*)

De todo el objeto JSON obtenido de la consulta anterior, lo que más nos interesará habitualmente es el objeto "hits", el cual tiene como elementos la propiedad "total" indicando el número total de registros/documentos recuperados -no necesariamente en orden- y, sobre todo, el array "hits", que contiene cada uno de dichos registros/documentos, los cuales son asimismo objetos que contienen, entre otros, los siguientes tres elementos importantes:

*El campo "_id" : Representa el identificador único de cada registro/documento. Los registros/documentso recopilados a través de LogStash poseen un identificador aleatorio alfanumérico generado automáticamente.

*El campo "_type" : Representa el "tipo" arbitrario del registro/documento. Puede tener cualquier valor (por ejemplo, "usuarios" o "proveedores", etc, aunque por defecto los registros obtenidos desde Logstash siempre tienen el tipo "doc") y su utilidad radica en categorizar el registro/documento para agruparlo con otros de su mismo "tipo". De hecho, es habitual escribir una consulta que restringela búsqueda a solamente registros/documentos de un determinado tipo, así:
curl http://127.0.0.1:9200/nombre-indice/tipo/_search

*El subobjeto "_source": Se corresponde con el documento original indexado proveniente, en este caso, de Logstash (dentro del cual, por tanto, podemos encontrar los ya conocidos campos "source" (indicando el origen del dato -por ejemplo, si es de un fichero, su ruta-), "beat" (la versión del Beat usado), "@timestamp"...y, por encima del resto, "message" (conteniendo el mensaje original recopilado por Beat).

NOTA: El elemento "took" del objeto general obtenido de una búsqueda con la orden *_search* indica el tiempo en milisegundos que tardó la búsqueda; el elemento "timed_out" indica si se sobrepasó el timeout,

Si sólo quisiéramos obtener un determinado registro/documento, se puede realizar una búsqueda por su identificador, así:

*curl "http://127.0.0.1:9200/nombre-indice/tipo/_search?pretty&**q=_id:valorIdentificador**"*

NOTA: Fijarse en el uso de las comillas para no confundir al terminal debido a la introducción en la URL del símbolo &

De hecho, aunque el identificador garantiza que solo haya un registro/documento concordante con la búsqueda, se puede realizar búsquedas por cualquier campo, de igual manera:

*curl "http://127.0.0.1:9200/nombre-indice/tipo/_search?pretty&**q=nombreCampo:valorCampo**"*

NOTA: En verdad, el valor del parámetro *q* puede ser tan solo una palabra (*q=palabra*), en cuyo caso la búsqueda se realizará por todos los campos existentes. También se admiten comodines (* y ?)

Una consulta que también podemos realizar de forma alternativa para obtener un determinado registro/documento es indicando directamente su URL completa de forma explícita, sin necesidad de realizar ninguna búsqueda, así...:

*curl http://127.0.0.1:9200/nombre-indice/tipo/**valorIdentificador**?pretty*

...aunque si sólo deseamos ver el contenido de su subobjeto "_source" lo podemos indicar así:

*curl http://127.0.0.1:9200/nombre-indice/tipo/valorIdentificador/**_source**?pretty*

Para saber más acciones que se pueden escribir en la URI para realizar búsquedas, consultar https://www.elastic.co/guide/en/elasticsearch/reference/current/search-uri-request.html

NOTA: Hay que saber que existen muchísimas más maneras de realizar búsquedas más allá de las acciones escritas en la URI (incluyendo operadores lógicos, expresiones regulares, expresiones "fuzziness"...); ElasticSearch ofrece un idioma propio ("Query DSL") con sus propias reglas sintácticas que permiten realizar al detalle cualquier tipo de consulta. Pero estudiar esto más en profundidad nos llevaría demasiado fuera de nuestros objetivos.

Consultas PUT, POST y DELETE

Tal como se puede deducir del apartado anterior, ElasticSearch ofrece una REST API via HTTP para interaccionar con los datos almacenados en él. Y no solo para consultar (que es lo que se ha estado haciendo en los ejemplos anteriores mediante peticiones GET -el tipo de petición por defecto usado por *curl*-) sino también, mediante peticiones de tipo POST, PUT o DELETE, para manipular, añadir o borrar datos, respectivamente.

Por ejemplo, para crear un nuevo registro/documento con un determinado identificador (a elegir como se desee siempre que no exista ya, leer más abajo) de un determinado tipo (no hace falta que exista previamente) en un determinado índice, se deberá realizar una consulta similar a esta (donde, tal como se ve, el cuerpo del dato a almacenar puede ser cualquiera pero ha de ser de tipo JSON para respetar el formato del subobjeto "_source" en el que se integrará) :

```
curl -X PUT http://127.0.0.1:9200/nombre-indice/tipo/idElegido -H "Content-Type:application/json"
        -d '{ "campo1":"valor1", "campo2":12345, "campo3":"2018-01-25 12:34:56" }'
```

En el caso de indicar un identificador de registro/documento que ya existiera, la petición lo PUT sobreescribirá por completo, pudiéndose usar, pues, para modificaciones completas. Pero si esta funcionalidad no la queremos por agresiva, podemos añadir /_create al final de la URI anterior (es decir, así: *http://127.0.0.1:9200/nombre-indice/tipo/idElegido/_create*) para que nos devuelva un error en el caso de encontrarse ya un identificador igual al indicado.

Otra manera tanto de crear como de actualizar un registro/documento es mediante POST. Si se trata de crear un nuevo registro/documento, la diferencia más importante entre ambos métodos es que mediante POST no se ha de indicar ningún identificador sino que se ha de delegar en ElasticSearch para que nos lo asigne él automáticament. Así pues, la sentencia podría ser similar a:

```
curl -X POST http://127.0.0.1:9200/nombre-indice/tipo  -H "Content-Type:application/json"
        -d  '{ "campo1":"valor1",  "campo2":12345,  "campo3":"2018-01-25 12:34:56" }'
```

Pero donde POST brilla más es a la hora de modificar un registro/documento ya existente, ya que es mucho más fino que PUT al permitir la edición, adición o borrado de campos individuales. Para ello, no obstante, debemos hacer uso de la orden /_update al final de la URL y del objeto especial "doc", que actúa como actualizador. Por ejemplo,para modificar el valor del "campo1" del registro/documento generado con la sentencia PUT anterior para que ahora valga "nuevovalor1" deberíamos ejecutar:

```
curl -X POST http://127.0.0.1:9200/nombre-indice/tipo/idElegido/_update -H "Content-Type:application/json"
        -d  '{ "doc" : {"campo1":"nuevovalor1"} }'
```

De la misma manera se podría añadir una nueva pareja clave<->valor no existente previamente o también eliminar una pareja clave<->valor concreta, aunque en este último caso la sintaxis de la consulta es algo diferente porque se ha de utilizar un objeto especial diferente llamado "script" (notar también el escape de las comillas dobles):

```
curl -X POST http://127.0.0.1:9200/nombre-indice/tipo/idElegido/_update -H "Content-Type:application/json"
        -d  '{ "script" : "ctx._source.remove(\"campo1\")" }'
```

Finalmente, para eliminar un determinado registro/documento tan solo hace falta lanzar una consulta DELETE indicando el identificador deseado, así:

```
curl -X DELETE http://127.0.0.1:9200/nombre-indice/tipo/valorId
```

Igualmente, se puede eliminar un índice entero lanzando la consulta...:

```
curl -X DELETE http://127.0.0.1:9200/nombre-indice
```

...y todos los índices mediante:

```
curl -X DELETE http://127.0.0.1:9200/_all
```

Se puede saber de una manera muy rápida si un determinado registro/documento existe observando el código de respuesta HTTP (200 o 404, según si es sí o si es no) a una consulta realizada mediante el método HEAD, así:

```
curl -I http://127.0.0.1:9200/nombre-indice/tipo/valorId
```

Configuració de "Kibana":

L'arxiu de configuració de Kibana és "/etc/kibana/kibana.yml". Allà haurem d'editar les següents opcions...

#Port on escolta les peticions dels clients web (navegadors).
#Ja té aquest valor per defecte, no cal canviar res
server.port: 5601
#IP de la tarja de xarxa per on rebrà aquestes peticions (pot valer 0.0.0.0 si es vol indicar totes)
#Per defecte val "localhost" i, per tant, no escolta per la xarxa
server.host: "IP.tarja.Adapt.pont"
#Servidor ElasticSearch on Kibana haurà de connectar per obtenir les dades que mostra.
#Ja té aquest valor per defecte, que ja ens va bé
elasticsearch.url: "http://127.0.0.1:9200"

...i finalment iniciar el servei executant (com a root): *systemctl start kibana*. Si tot va bé, en obrir un navegador (a la màquina real, per exemple) i escriure http://ip.serv.Kibana:5601 hauria d'aparèixer el panell web de Kibana. Tingueu en compte, no obstant, que aquesta instal.lació és molt bàsica i no incorpora cap autenticació ni està xifrada amb HTTPS, així que encara caldria aprofundir una mica més en la seva configuració per adaptar-la a un ús més real de l'eina.

EXERCICIS:

1.-Arrenca la màquina virtual servidora dels darrers exercicis i instal.la-hi, si no ho has fet ja, l'ElasticSearch, assegurant-te primer que tingui com a mínim 3GB de RAM i la seva tarja de xarxa en mode "adaptador-pont". A partir d'aquí, inicia el servei ElasticSearch (*sudo systemctl start elasticsearch*) i...:

a) Comprova amb *curl* que el servidor ElasticSearch respon

b) Crea amb *curl* (concretament, amb una petició PUT) un nou registre/document amb identificador "Manolo" i de tipus "amic" dins d'un índex anomenat "persones" (en no existir aquest índex, es crearà automàticament) que estigui format per dos camps JSON: un anomenat "nom" amb el valor de cadena que tu vulguis i un altre anomenat "edat" amb el valor numèric que tu vulguis també.

c) Comprova amb *curl* la llista d'índexs existents per tal de comprovar que l'índex anterior s'ha creat bé

d) Realitza una consulta amb *curl* que mostri el contingut del subobjecte "_source" corresponent al registre/document acabat de crear

e) Afegeix amb *curl* un nou registre/document a aquest índex també de tipus "amic" però ara amb els camps "nom", "edat" i "telèfon" (i amb els valors que tu vulguis)

NOTA: A les darreres versions d'ElasticSearch no es poden afegir a un índex documents/registres de tipus diferents

f) Fes una recerca amb *curl* de només els registres/documents que siguin de tipus "amic" (que són tots, de fet)

2.-a) Per no haver de tenir dues màquines virtuals (client i servidor) funcionant a la vegada, instal.la FileBeat a la mateixa màquina que estàs fent servir ara (la servidora, on ja n'hi ha el LogStash i el ElasticSearch). Configura aquest FileBeat per a què ara (només) obtingui les dades d'entrada del contingut que vagi apareixent dins d'un fitxer anomenat "/var/log/pepe.log" i les reenvïi al LogStash de la mateixa màquina. Inicia el servei (*sudo systemctl start filebeat*)

b) Configura el LogStash de la màquina servidora per a què rebi les dades del FileBeat local i les envïi al servidor ElasticSearch local també, sense fer servir cap filtre. Inicia el servei (*sudo systemctl start logstash*)

NOTA: Fixa't que en iniciar el servei LogStash no ha calgut indicar el seu arxiu de configuració perquè per defecte, n'agafa tots els que hi hagi ubicats dins de la carpeta /etc/logstash/conf.d . Això és així gràcies a la invocació concreta al binari LogStash que es realitza a través del fitxer /etc/systemd/system/logstash.service , la qual utilitza el paràmetre --path.settings justament per això: per indicar no un fitxer de configuració concret sinó una carpeta on es llegiran tots els arxius de configuració que allà hi hagin.

c) Executa la comanda *echo $RANDOM >> /var/log/pepe.log* varis cops i seguidament comprova amb *curl* la llista d'índexs existents per tal de comprovar que s'ha creat un nou índex anomenat "logstash-yyyy-mm-dd" amb tants registres/documents com cops hagis executat la comanda anterior). Fes una recerca de tots aquests registres/documents i observa el seu respectiu valor del subojecte "_source" per confirmar que efectivament és el valor correcte.

NOTA: La comanda *echo $RANDOM > /var/log/pepe.log* no és recollida gaire bé per FileBeat: en "resetejar" cada cop el fitxer per començar de nou les escriptures des del principi, el cursor del FileBeat es "despista" i no sap tornar a aquest principi. Això és perquè els fitxers "log", que són les fonts d'informació principals de FileBeat no tenen aquest comportament.

3.- a) Instal.la, si no ho has fet ja, Kibana a la mateixa màquina que has anat fent servir al llarg dels exercicis anteriors. Configura'l tal com s'explica a la teoria i engega'l. Seguidament accedeix-hi a través del navegador de la teva màquina real. Un cop dins de la pantalla principal de Kibana, vés al link "Discover" i allà introdueix "logstash-*" com a "index pattern" per tal de què Kibana sàpiga que ha de recollir (només) les dades dels índexs emmagatzemats a ElasticSearch que quadrin amb aquest patró (si has seguit els exercicis anteriors, hi hauria d'haver un índex amb aquestes característiques). Defineix @timestamp com a filtre de temps.

NOTA: Si es volgués tornar a repetir aquest pas en el futur, caldria anar al link "Management"->"Index patterns"

b) Un cop agregat l'"index pattern", torna a anar al link "Discover" per obtenir en format de dues columnes les dades dels camps @timestamp i "message". Si d'entrada Kibana t'avisa de què no tens dades en el rang de temps actual, hauràs de modificar-lo al botonet que apareix a la cantonada de dalt a la dreta amb la icona del rellotge.

NOTA: Pots fer servir el quadre de búsqueda que apareix a dalt de la pàgina "Discover" per escriure el mateix que si fos el paràmetre *q* d'una consulta GET a ElasticSearch. Igual que llavors, les recerques dels noms de camps i els seus valors són case-insensitive; es poden escriure cometes per indicar una recerca exacta; es poden fer servir els comodins * i ? ; es poden indicar rangs de valors per un camp amb sintaxi *[x TO y]* o *{x TO y}* (si s'escriu entre [] éls extrems del rang s'inclouen i si s'escriu entre {} els extrems no s'inclouen) ; es poden indicar operadors booleans com AND, OR o NOT, etc, etc . Alternativament, per tal d'escriure filtres de forma més fàcil, es pot utilitzar botó/assistent "Add a filter" que apareix a la banda esquerra sobre la llista de camps.

Kibana visualizations are based on aggregations performed by Elasticsearch. Kibana simply supplies the UI to use these aggregations for defining the different dimensions on display in visualizations. There are two types of aggregations: Metric aggregations and Bucket aggregations. Bucket aggregations groups documents together in one "bucket" according to your logic and requirements, while the Metric aggregations are used to calculate a value (average, count, sum, min, max, etc) for each bucket based on the documents inside the bucket. Each visualization type presents buckets and their values in different ways. So in a pie chart for example, the number of slices is defined by the Buckets aggregation while the size of the slice is defined by the Metric aggregation. Kibana supports quite a large number of Elasticsearch aggregation types, each with specific configuration options and field type limitations. For example, Histogram and Date Histogram Bucket aggregations will only work on integers, the Min and Max Metric aggregations will only work on number or date fields while the Unique Count Metric aggregation works on any field type, and so on.

c) Vés al link "Visualize" per crear una gràfica de punts (per això s'ha d'escollir la visualització "Line" i després personalitzar la seva aparença a la pestanya "Metrics & Axes") que mostri en l'eix de les X un "bucket" de tipus "Date Histogram" amb el camp @timestamp com a referència amb un interval de 10 segons i en l'eix de les Y la suma de tots els valors del camp "message" per aquest interval . Guarda aquesta visualització. Podeu veure la documentació oficial sobre aquest tipus de visualització aquí: https://www.elastic.co/guide/en/kibana/current/xy-chart.html

NOTA: ¡Atenció! Per a què funcioni aquest apartat cal que els valors del camp "message" siguin emmagatzemats per ElasticSearch (i, com a conseqüència, reconeguts per Kibana) com a números (i no com a cadenes, que és el que passa per

defecte). Una manera senzilla d'aconseguir-ho és mitjançant l'opció "convert" del filtre "mutate" de LogStash (https://www.elastic.co/guide/en/logstash/current/plugins-filters-mutate.html#plugins-filters-mutate-convert). Per tant, hauràs de modificar adientment la configuració de LogStash abans de fer res

d) Afegeix la visualització anterior a un "Dashboard" per a què es vegi a pantalla completa. Per més informació pots consultar https://www.elastic.co/guide/en/kibana/current/dashboard-getting-started.html

Integració de Falco + Elastic Stack com a HIDS + SIEM

A **HIDS** is a Host Intrusion Detection software that monitors a critical host system for malicious activity or policy violations. It examines events on a computer on your network rather than the traffic that passes around the system. A HIDS agent utilizes various detection methods including; log monitoring, rootkit detection (via checksum verification methods), file integrity monitoring, etc and can alert security personnel of specific events and configuration changes. Naturally, if you have more than one HIDS host on your network, you don't want to have to log into each one in order to get feedback. So, a distributed HIDS system needs to include a centralized control module, called SIEM. Look for a system that encrypts communications between host agents and the central monitor (the SIEM system).

A **NIDS** is a Network Intrusion Detection System that monitors traffic on your network infraestructure for malicious activity or policy violations. As such, a typical NIDS has to include a packet sniffer in order to gather network traffic for analysis and uses a predefined set of "bad behaviour" rules to match them with collected packets to identify suspicious traffic. Any detected activity or violation is typically reported either to security personnel or collected centrally using a SIEM system.

A **SIEM** (Security Information Event Management) is a system that provides real-time analysis of security alerts generated by network hardware and applications, by correlating events from critical assets to provide actionable threat intelligence. A properly configured SIEM can help detect threats and decrease the amount of time a malicious adversary may be in/on a network.

1.-A partir de l'ús de Falco i la suite ELK es pot implementar un HIDS amb SIEM integrat. Això és el que farem en aquest exercici.

a) A la màquina on ja tens instal.lat dels exercicis anteriors FileBeat + LogStash + ElasticSearch + Kibana, ara instal.la-hi Falco (recorda, només cal que executis aquesta comanda per fer-ho: *curl -s https://s3.amazonaws.com/download.draios.com/stable/install-falco | sudo bash*

b) Comprova que Falco ja estigui en marxa (*systemctl status falco*) i que les seves regles per defecte ja funcionin. Això últim ho pots comprovar executant algun event sospitós (com ara *sudo touch /bin/virus* o bé *sudo touch /etc/fstab*) i observant seguidament la sortida de la comanda *journalctl -u falco -p err*

> NOTA: Estem suposant que la configuració de Falco és la per defecte, on gràcies a tenir activada la línia *syslog_output* de l'arxiu de configuració "falco.yaml" totes les incidències de seguretat són enviades al registre-journal del sistema. Si no fos el cas, caldria editar aquesta línia convenientment
> NOTA: Recordeu que el format concret del missatge enviat al Journal per cada event detectat es podria modificar a la línia "output" de la regla corresponent, definida a l'arxiu "falco_rules.yaml"
> NOTA: Fixeu-vos que no cal instal.lar explícitament Sysdig per a què Falco funcioni: el mateix motor recol.lector de crides al sistema ve incorporat als dos programes... Sysdig en realitat només és una interfície interactiva, així que ara no el necessitarem per res.

A partir d'aquí, la idea seria que, en gravar el missatge d'error al Journald, hi hagués algun tipus de dimoni Beat "subscrit" al Journald que detectés l'aparició d'aquest nou missatge per tal de reenviar-lo a un servidor LogStash/ElasticSearch. Inicialment hom podria pensar en fer servir FileBeat, però aquest Beat no és capaç de llegir directament del Journald degut a què aquest és un registre binari i no pas de tipus "log" (és a dir, de tipus text, que és com FileBeat se l'espera). Afortunadament, existeix un Beat similar a FileBeat que justament fa el que volme: "rastrejar" en temps real els missatges del Journal; l'anomenat JournalBeat (https://github.com/elastic/beats/tree/master/journalbeat)

> NOTA: Algú podria pensar que igualment es podria fer servir FileBeat si establim la seva configuració per a què obtingui les dades de "stdin" i modifiquem la secció "program_output" de Falco per a què quedi així:
>
> ```
> program_output:
>   enabled: true
>   keep_alive: false
>   program: sudo /usr/share/filebeat/bin/filebeat -c /etc/filebeat/filebeat.yml
> ```

No obstant, si es prova això, en provocar diversos events sospitosos es veuria que ElasticSearch només rep **un sol** missatge, corresponent al primer event sospitós provocat (contant després del reinici del servidor Falco), sigui quin sigui el número total d'events sospitosos que hagis provocat. Això és perquè la secció "program_output" de Falco està pensada per cridar a comandes que s'executen d'un sol cop i paren, no per executar un binari indefinidament, com fa FileBeat. El primer cop que Falco detecta un event, crida a FileBeat i l'entrega el missatge però aquest FileBeat es queda funcionant; això fa que el segon cop que Falco detecti algun event, vulgui cridar a FileBeat i el que passa és que s'executa un altre binari mentre el primer encara hi és, provocant que el missatge acabi no entregant-se a cap. El tercer event posarà en marxa un tercer FileBeat "a sobre" dels altres dos, i així. Això es pot veure clarament amb eines com *top* o *htop*

En principi una sol.lució podria ser limitar la duració de l'execució de FileBeat per tal de què, en passar un cert temps (curt, per exemple un segon) sense rebre cap dada, s'auto-apagui; això és el que en teoria fa l'opció filebeat.shutdown_timeout: 1s de la secció general del fitxer "filebeat.yml", però no funciona, els diversos FileBeats es mantenen encesos indefinidament igualment. Una altra sol.lució podria ser, en canvi, establir a "true" l'opció "keep_alive" de la secció "program_output" de Falco; això el que fa és executar només un sol FileBeat la primera vegada i deixar-lo funcionant per rebre els següents missatges. En principi hauria de ser l'opció ideal però per alguna raó (segurament deguda a com està programat FileBeat) no funciona: el FileBeat (únic, en aquest cas) es posa en marxa a la vegada que Falco (és a dir, abans de generar-se cap missatge) però quan Falco comença a enviar algun missatge, el FileBeat ja no se n'entera.

2.-a) Instal.la el paquet "journalbeat" (*apt/dnf install journalbeat*) i escriu una configuració similar a la següent dins del fitxer "/etc/journalbeat/journalbeat.yml":

```
journalbeat.inputs:
- paths: []
  seek: cursor
  include_matches: ["process.name=falco"]
output.logstash:
    hosts: ["127.0.0.1:5044"]
```

NOTA: La llista de característiques/filtres reconegudes per JournalBeat que es poden afegir dins de l'array "include_matches" per tal de reenviar a l'output indicat només els missatges de Journal que quadrin amb aquestes (totes en conjunt) és a https://www.elastic.co/guide/en/beats/journalbeat/6.x/configuration-journalbeat-options.html#translated-fields Entre elles podem destacar "**process.uid**", "**process.pid**", "**systemd.unit**", "**syslog.priority**", "**message**" o "**host.name**". Si no hi hagués disponible alguna característica concreta que necessitéssim, també podem afegir com elements de la llista directament els noms dels camps Journal tal qual, així: "_COMM=falco", per exemple. En qualsevol cas, no s'admeten ni comodins ni expressions regulars en els valors indicats.

NOTA: Per més informació sobre les possibilitats de configuració que ofereix JournalBeat, consulta https://www.elastic.co/guide/en/beats/journalbeat/current/journalbeat-configuration.html

b) Inicia el servei JournalBeat (*systemctl start journalbeat*) i assegura't a més que, si tens instal.lat el dimoni FileBeat d'exercicis anteriors, que estigui apagat (ja no el farem servir més).

c) Provoca algun event sospitós a la màquina virtual (com ara els mateixos *sudo touch /bin/virus* o *sudo touch /etc/fstab*) i seguidament comprova que els missatges d'avís corresponents generats per Falco hagin arribat efectivament a ElasticSearch (executant simplement *curl http://127.0.0.1:9200/_cat/indices* per veure quants són, per exemple).

NOTA: Se suposa que el servidor LogStash ja el vam deixar configurat en els exericis anteriors per a què reenviés les dades rebudes al servidor ElasticSearch local. Per tant, en principi això ja hauria d'estar fet i, així doncs, no cal modificar res.

d) Obre el navegador de la teva màquina real per accedir al servidor Kibana i observar els missatges d'error rebuts al seu panell web "Discover" (hauràs d'afegir primer un "index pattern"). Fes que en aquest panell "Discover" només es vegin dues columnes: @timestamp i "message"

Segurament, en fer l'apartat anterior veuràs molts missatge d'error provocats pel propi JournalBeat degut a què justament aquest programa modifica fitxers sobre la carpeta "/", fet que Falco identifica com a perill potencial. Per evitar aquest soroll tenim tres possibilitats: o bé modificar les regles del Falco per tal de reconèixer els fitxers ubicats a l'arrel del sistema i manipulats per JournalBeat com a vàlids o bé fer que LogStash filtri tots els missatges rebuts no desitjats o directament eliminar-los a l'origen mitjançant el "processor" "drop_event" de JournalBeat. Optarem per aquesta tercera opció:

e) Atura el servei JournalBeat i, per començar de nou, esborra l'índex "logstash-*" amb els missatges de Falco recopilats fins ara (recorda, *curl -X DELETE http://127.0.0.1:9200/logstash-**). Seguidament, torna a editar el fitxer "/etc/journalbeat/journalbeat.yml" i a l'array de "processors" afegeix-ne un de nou de tipus "drop_event". Concretament, ha de quedar així (els "processors" que ja estan activats per defecte els pots deixar com estaven o no, com vulguis):

```
processors:
  - drop_event:
      when:
        contains:
          message: "journalbeat"
```

NOTA: Per veure més condicions possibles (regexp, range, has_fields, etc) consulta *https://www.elastic.co/guide/en/beats/journalbeat/current/defining-processors.html#conditions*

NOTA: Si haguéssim volgut eliminar els missatges espuris no a JournalBeat sinó a LogStash, una manera de fer-ho hagués sigut indicant la següent secció *filter* a l'arxiu de configuració ubicat la carpeta "/etc/logstash/conf.d":

```
filter {
   if "journalbeat" in [message] {  drop {}  }
}
```

També es podria haver fet servir expressions regulars per realitzar una recerca més àmplia, així:

```
filter {
   if [message] =~ /journal.*/ {  drop {}  }
}
```

f) Inicia de nou el servei JournalBeat i torna a repetir els apartats c) i d) per veure si ara obtens uns resultats més nets. ¿Què passa quan al quadre de recerca de la zona superior del panel "Discover" de Kibana escrius l'expressió *"syslog.priority:[1 TO 3]"*? ¿Quina és la diferència entre els valors dels camps "process.name", "process.cmd" i "process.executable"? ¿I entre els dels camps "process.name" i "systemd.unit"?

g) Realitza una visualització de tipus "Pie Chart" on a l'apartat "Metrics" es mantingui "Count" com a valor de l'ítem '"Slice size" i on a l'apartat "Bucket" aparegui el valor "Terms" al quadre "Aggregation", el valor "process.name" al quadre "Field" i els valors de "Metric:Count" i "Descending:5" al quadre "Order by" de l'ítem "Split Slices". ¿Quin és el significat del diagrama resultant? Grava aquest diagrama en el dashboard.

gII) Realitza una altra visualització de tipus "Pie Chart" on a l'apartat "Metrics" es mantingui igualment "Count" com a valor de l'ítem '"Slice size" i on a l'apartat "Bucket" aparegui el valor "Histogram" al quadre "Aggregation", el valor "syslog.priority" al quadre "Field" i el valor 1 al quadre "Minimum interval" de l'ítem "Split Slices". ¿Quin és el significat del diagrama resultant? Grava aquest diagrama en el dashboard i vés-hi per observar que aparegui aquesta visualització i la realitzada a l'apartat anterior. Grava aquest dashboard.

ElastAlert:

ElastAlert (https://github.com/Yelp/elastalert) is a simple non-official framework for alerting on anomalies, spikes, or other patterns of interest from data in Elasticsearch. If you have data being written into Elasticsearch in near real time and want to be alerted when that data matches certain patterns, ElastAlert is the tool for you.

It works by combining Elasticsearch with two types of components: rule types and alerts. Elasticsearch is periodically queried and the data is passed to the rule type, which determines when a match is found. When a match occurs, it is given to one or more alerts, which take action based on the match.

Several rule types with common monitoring paradigms are included with ElastAlert. Some are:

*Match where there are at least X events in Y time (`frequency` type)
*Match when there are less than X events in Y time (`flatline` type)
*Match when the rate of events increases or decreases (`spike` type)
*Match when a certain field matches a blacklist/whitelist (`blacklist` and `whitelist` type)
*Match on any event matching a given filter (`any` type)
*Match when a field has two different values within some time (`change` type)
*Match when a never before seen term (value) appears in a field (`new_term` type)
*Match when the number of unique values for a field is above or below a threshold (`cardinality` type)
*Match when the value of an aggregation function (sum, avg, min, max) from a numeric field, calculated after a specific period of time, is above or below a threshold (`metric_aggregation` type)

Several alert types are built-in supported in ElasticAlert, too. We can highlight, among them: email, Telegram, GoogleChat, AWS SNS, Twilio, Slack, etc. Additional rule types and alerts can be easily imported or written.

Per instal.lar, configurar i posar en marxa aquest programa, cal fer els següents passos:

1.- Executar les següents comandes (observeu quan s'ha d'escriure "sudo" i quan no, és important):

sudo apt install python-pip python-dev libffi-dev libssl-dev gcc git (a Ubuntu)
sudo dnf install python2-pip python2-devel libffi-devel openssl-devel gcc git (a Fedora)
git clone https://github.com/Yelp/elastalert.git
cd elastalert
pip install -r requirements-dev.txt
pip install -r requirementstxt
sudo python2 setup.py install

2.- A continuació editar l'arxiu "config.yaml.example", ubicat dins de la carpeta "elastalert", i canviar, si s'escau, alguna de les següents opcions. Un cop gravades, cal canviar el nom d'aquest fitxer per a què sigui "config.yaml". Algunes d'aquestes opcions són:

*`rules_folder` : is the folder path where ElastAlert will attempt to load every rule configuration YAML files from. As the files in this folder change, ElastAlert will also load new rules, stop running missing rules, and restart modified rules.

*`run_every` : is how often ElastAlert will query Elasticsearch.

*`buffer_time` : is the size of the query window, stretching backwards from the time each query is run (that is: how much time ElasticAlert will seek for past data to catch and compare it to know if it must trigger an alert or not). This value is ignored for rules where `use_count_query` or `use_terms_query` is set to true.

*`es_host` : is the address of an Elasticsearch server where ElastAlert will store data about its state, queries run, alerts, and errors. Each rule may also use a different Elasticsearch host to query against.

*`es_port` : is the port corresponding to `es_host`. Each rule may also use a different Elasticsearch port to query against.

*`writeback_index` : is the name of the index in which ElastAlert will store data. We must create this index later.

*`alert_time_limit` : if an alert fails for some reason, ElasticAlert will try sending the alert until this time period has elapsed

NOTA: N'hi ha més, com ara les que gestionen la comunicació encriptada amb ElasticSearch (*use_ssl, verify_certs,client_cert,client_key,ca_certs*) o les que gestionen l'autenticació amb ell (*es_username, es_password*), etc. Per veure tota la llista es pot consultar https://elastalert.readthedocs.io/en/latest/elastalert.html#configuration

3.- ElastAlert can save information and metadata about its queries and its alerts back to Elasticsearch. This is useful for auditing, debugging, and it allows ElastAlert to restart and resume exactly where it left off. This is not required for ElastAlert to run, but highly recommended. To achieve that, you need to create an index for ElastAlert to write to by running *elastalert-create-index* Aquesta comanda a Ubuntu preguntarà diferents dades interactivament, com ara la IP i port del servidor ElasticSearch a connectar-s'hi, si s'utilitzarà SSL o no, si s'utilitzarà autenticació per usuari/contrasenya o no, la URI de l'ElasticSearch a fer servir (el valor per defecte, que és "/", ja va bé), el nom de l'index a crear (el valor proposat per defecte ja està bé) i si volem un índex de còpia de seguretat o no.

4.- Cal crear com a mínim una regla (cadascuna s'haurà d'escriure en un fitxer YAML per separat i ubicat dins de la carpeta indicada a l'opció `rules_folder`). Una regla ha de tenir diferents elements, com ara la consulta a fer, els possibles valors que disparen l'alerta, els possibles destins (poden ser més d'un) per l'alerta, etc. Alguns d'aquests elements són comuns a tots els tipus de regles i uns altres en són particulars. D'entre els generals podem destacar:

*name is the name for the rule. ElastAlert will not start if two rules share the same name.

*type: Each rule has a different predefined type which may take different parameters. For information about every possible types you can use or custom-create, see https://elastalert.readthedocs.io/en/latest/ruletypes.html#rule-types

*index: The name of the index(es) to query.

*filter is a list of Elasticsearch filters that are used to filter obtained documents in order to not to take them into account for triggering the rule in question. You can found a summarized list of them in https://elastalert.readthedocs.io/en/latest/recipes/writing_filters.html#writingfilters and https://elastalert.readthedocs.io/en/latest/recipes/writing_filters.html#common-filter-types

For instance, here we have a single term filter for documents with "some_field" matching "some_value"...:

```
filter:
- term:
    some_field: "some_value"
```

...but if you have a filter in Kibana and want to recreate it in ElastAlert, you probably want to use a query string, like this:

```
filter:
- query:
    query_string:
       query: "foo: bar AND baz: abc*"
```

If no filters are desired, it should be specified as an empty list: filter: []

*alert is a list of alert names; specified alerts will run on each match. For more information on alert types, see https://elastalert.readthedocs.io/en/latest/ruletypes.html#alerts or read grey table below. Following this element there must be another/s element/s defining the desired configuration of specified alerts: see grey table below too for more information about these anothers elements.

Per saber més opcions possibles per qualsevol tipus de regla, es pot consultar la documentació oficial a https://elastalert.readthedocs.io/en/latest/ruletypes.html#commonconfig Per exemple, una altra opció general que convé conèixer és l'opció `realert`, així:

```
realert:
  minutes: 0
```

Aquesta opció indica el temps mínim que ha de passar per a què s'emetin dos alertes de la mateixa regla: qualsevol alerta que es pugui activar dins d'aquest temps després d'una alerta idèntica anterior simplement no es generarà (encara que al registre de ElastAlert hi quedarà constància mitjançant el missatge "Ignoring match for silenced rule"). El valor per defecte d'aquest temps mínim és 1 minut.

NOTA: Les especificacions de temps es poden donar a diverses directives de l'ElastAlert. En tot cas, són les següents: "weeks", "days", "hours", "minutes" i "seconds"

Una altra opció interessant és `alert_text`, que serveix per afegir un text personalitzat a l'alerta. Però si el que es volgués no és afegir sinó directament que aquest text fos l'únic de l'alerta, llavors caldria afegir també l'opció `alert_text_type` amb el valor `alert_text_only` En el cas de què es volgués indicar el valor de certs camps dins del text personalitzat de l'alerta, caldrà indicar com a valor de l'opció `alert_text_args` un array d'aquests camps i, en el text escrit a `alert_text`, indicar la seva posició amb la sintaxis {0}, {1}, etc. Per exemple:

```
alert_text: "Something happened with {0} at {1}"
alert_text_type: alert_text_only
alert_text_args: ["username", "@timestamp"]
```

NOTA: Opcions similars a les anteriors són `alert_subject` i `alert_subject_args` , útils per les alertes que admeten "subjects", com ara el correu.

Finalment, comentar que tots els documents han de tenir un camp de tipus "timestamp". ElasticAlert per defecte farà servir el camp @timestamp pero això es pot canviar amb l'opció `timestamp_field` , si es vol. També es pot canviar el format del "timestamp", per passar del ISO8601 (per defecte) al format UNIX, mitjançant l'opció `timestamp_type`.

Some alert types

Si es vol enviar l'alerta per **mail**, cal indicar a la secció alert el valor email , així...:

```
alert:
  - email
```

...i, a més, indicar la següent opció a la línia següent: `email: ["desti1@gmail.com","desti2@gmail.com"]`

Es poden indicar altres opcions extra (cadascuna en una línia diferent)com ara:

*smtp_host: IP o nom del servidor SMTP a usar (per defecte, és "localhost"; això vol dir que hauríem de tenir un software Postfix o similar funcionant, tal com ja vam veure)

*smtp_port: port del servidor SMTP a usar (per defecte, és 25)

*smtp_ssl: Connect the SMTP server using TLS (defaults to false). If this option is not used, ElastAlert will still attempt STARTTLS.

*smtp_auth_file: The path to a file which contains SMTP authentication credentials. The path can be either absolute or relative to the given rule. It should be YAML formatted and contain two fields, `user` and `password`. If this is not present, no authentication will be attempted.

*from_addr: This sets the From header in the email. By default, the from address is "ElastAlert@" and the domain will be set by the smtp server.

*email_format: If set to `html`, the email's MIME type will be set to HTML, and HTML content should correctly render. If you use this, you need to put your own HTML into `alert_text` and use `alert_text_type: alert_text_only`.

NOTA: Options for each alerter can either defined at the top level of the YAML file, or nested within the alert name, allowing for different settings for multiple of the same alerter. For example, consider sending multiple emails, but with different 'To' and 'From' fields:

```
alert:
 - email
from_addr: "no-reply@example.com"
email: "customer@example.com"
```

versus

```
alert:
 - email:
     from_addr: "no-reply@example.com"
     email: "customer@example.com"
 - email:
     from_addr: "elastalert@example.com""
     email: "devs@example.com"
```

If multiple of the same alerter type are used, top level settings will be used as the default and inline settings will override those for each alerter.

Si, en canvi, es vol enviar el missatge d'alerta (en format JSON) via **POST** a un servidor HTTP extern, cal indicar a la secció alert el valor post , així...:

```
alert:
  - post
```

...i, a més, indicar la següent opció a la línia següent: `http_post_url: "http://url/on/enviar/missatge"`

Es poden indicar altres opcions extra (cadascuna en una línia diferent) com ara:

*http_post_payload: List of keys:values to use as the content of the POST. Example - *ip:clientip* will map the value from the "clientip" index of Elasticsearch to JSON key named "ip". If not defined, all the Elasticsearch keys will be sent.

*http_post_static_payload: Key:value pairs of static parameters to be sent, along with the Elasticsearch results. Put your authentication or other information here.

*http_post_timeout: The timeout value, in seconds, for making the post. The default is 10. If a timeout occurs, the alert will be retried next time elastalert cycles.

Si, en canvi, es vol executar una determinada **comanda** en activar-se una alerta, cal indicar a la secció alert el valor command , així...:

```
alert:
  - command
```

...i, a més, indicar a la línia següent l'opció obligatòria `command`, la qual serà un array de cadenes el primer element del qual serà la ruta del programa a executar, el segon serà el nom del seu primer paràmetre, el tercer serà el valor d'aquest paràmetre, el quart serà el nom del segon paràmetre, el cinquè serà el valor d'aquest paràmetre, etc. Si es volgués correspondre el valor d'algun paràmetre amb el d'algun camp del missatge d'alerta rebut, es pot indicar així: {nom_camp} o {nom_camp[subcamp]} A més es poden indicar altres opcions com ara:

*pipe_match_json: If true, the match will be converted to JSON and passed to stdin of the command. Note that this will cause ElastAlert to block until the command exits or sends an EOF to stdout.

*pipe_alert_text: If true, the standard alert body text will be passed to stdin of the command. Note that this will cause ElastAlert to block until the command exits or sends an EOF to stdout. It cannot be used at the same time as pipe_match_json

Per conèixer millor els diferents tipus de regles, és recomanable observar els diferents arxius-plantilla proporcionats per ElastAlert "de regal" anomenats "example_*.yaml" i ubicats dins de la carpeta "example_rules". Per exemple, si observem l'arxiu "example_frequency.yaml", es pot veure que té un contingut similar al següent, on es defineix una alerta de tipus "**frequency**" i ....:

```
name: Example rule 1
type: frequency
index: logstash-*
num_events: 50
timeframe:
    hours: 4
filter:
- term:
    some_field: "some_value"
alert:
- email
email: ["elastalert@example.com"]
```

...on el significat dels elements particulars per aquest tipus de regles són:

*num_events: This parameter is specific to frequency type and is the threshold for when an alert is triggered. That is: it means “Alert when more than num_events occur within timeframe.

*timeframe is the time period in which num_events must occur.

NOTA: La regla d'exemple anterior significa: "enviar un email a "elastalert@example.com" quan hi hagi més de 50 documents/registres amb "some_field" == "some_value" en un període de 4 hores.

A l'arxiu "example_spike.yaml", en canvi, es pot veure un cas d'alerta de tipus "**spike**", la qual, a més de tenir les opcions genèriques name, type, index, filter, alert i les corresponents al destins escollits (email,...), disposa de les opcions específiques següents:

*threshold_cur : is the minimum number of events detected in current period of time that will trigger an alert (assuming this number is also greather than reference (last) number of events * spike_height) There's also threshold_ref, which serves to force to detect at least the number of events specified to be a valid value to use in the multiplication above. In general, both threshold have the same value to ensure a minimal quantity of samples

*timeframe : is the time period in which reference (last) rate and current rate are measured to be compared

*spike_height: if its value is 3, the rule matches when the current window contains 3 times more events than the reference window

*spike_type: is the direction of the spike: "up" matches only spikes, "down" matches only troughs, "both" matches both spikes and troughs.

*field_value: when set, uses the value of the field in the document and not the number of matching documents. This is useful to monitor for example a temperature sensor and raise an alarm if the temperature grows too fast. Note that the means of the field on the reference and current windows are used to determine if the spike_height value is reached. Note also that the threshold parameters are ignored in this smode.

Un exemple de regla d'aquest tipus seria aquest:

```
name: Example rule 2
type: spike
index: logstash-*
threshold_cur: 5
timeframe:
    hours: 2
spike_height: 3
spike_type: "up"
filter:
- query:
    query_string:
       query: "some_field: some_value"
alert:
- email
email: ["elastalert@example.com"]
```

NOTA: La regla d'exemple del fitxer anterior significa: "enviar un email a "elastalert@example.com" quan en les darreres dues hores es detectin tres vegades més documents/registres coincidents amb el filtre indicat que el número d'events detectats en les dues hores anteriors

A l'arxiu "example_change.yaml", en canvi, es pot veure un cas d'alerta de tipus "**change**", la qual, a més de tenir les opcions genèriques name, type, index, filter, alert i les corresponents al destins escollits (email,...), disposa de les opcions particulars següents:

*compare_key: is the name of the field whose value will be monitored to see if it changes

*ignore_null: ignore documents without the compare_key field

*query_key: the change must occur in two documents with the same value of the field indicated here

*timeframe : to trigger an alert the value of compare_key must change in two events that are less than timeframe apart

Un exemple de regla d'aquest tipus seria aquest:

```
name: Example rule 3
type: change
index: logstash-*
compare_key: country_name
ignore_null: true
query_key: username
timeframe:
   days: 1
filter: []
```

```
alert:
- email
email: ["elastalert@example.com"]
```

NOTA: La regla d'exemple del fitxer anterior significa: "enviar un email a "elastalert@example.com" quan es detecti que dos documents amb el mateix valor pel camp "username" obtinguts en un període de 24 hores tinguin un valor diferent pel camp "country_name"

A l'arxiu "example_new_term.yaml", finalment, es pot veure un cas d'alerta de tipus "**new_term**", la qual, a més de tenir les opcions genèriques name, type, index, filter, alert i les corresponents al destins escollits (email,...), disposa de les opcions particulars següents:

*fields : array of field names to monitor

*terms_window_size: period of time when ElasticAlert will search backwards to find which values of monitored fields already exist. If they existed, no alerts will be triggered for them when they appear again

Un exemple de regla d'aquest tipus seria aquest:

```
name: Example rule 4
type: new_term
index: logstash-*
fields: ["ip_address"]
terms_window_size:
  days: 90
filter:
- term:
    username: "admin"
alert:
- email
email: ["elastalert@example.com"]
```

NOTA: La regla d'exemple del fitxer anterior significa: "enviar un email a "elastalert@example.com" quan es detecti que un nou valor apareix en el camp "ip_address" dels documents obtinguts (filtrats per usuari "admin")

Una alerta també força útil és "**any**" la qual només disposa de les opcions genèriques (name, type, index, filter, alert i les corresponents al destins escollits: email,...); ja està. Aquesta alerta s'activarà per cada document/registre que arribi a ElasticSearch (i que concordi amb el filtre indicat). Es fa servir sobretot per detectar l'existència de valors concrets en camps concrets dels documents/registres recopilats. alert.

De totes formes, hi ha força més tipus d'alertes (concretament, les llistades al principi d'aquest apartat); per veure més exemples de la resta d'aquests tipus, es pot consultar https://qbox.io/blog/integrating-elastalert-email-alerting-with-elasticsearch o https://qbox.io/blog/elastalert-telegram-alerting-elasticsearch o també https://qbox.io/blog/elasticsearch-alerting-at-scale-using-elastalert

Opcionalment, es pot executar la comanda *elastalert-test-rule ruta/fitxer_regla.yaml* per comprovar si el fitxer de regla indicat es carrega correctament. A més, aquesta informa de quantes alertes s'activarien amb la regla en qüestió consultant els documents/registres de les darreres 24 hores.

5.- Invocar a ElasticAlert. Hi ha dues maneres: o bé directament en un terminal (aquest mode ens convindrà al principi per tal d'observar el seu comportament en temps real i poder depurar possibles errors que haguem escrit dins la seva configuració) o bé com a servei Systemd (aquest mode serà el mode "de producció" un cop haguem depurat i testejat bé el seu comportament).

*Directament: executa la comanda *elastalert --verbose [--rule ruta/fitxer_regla.yml]*
NOTA: El paràmetre *--rule* serveix per carregar només la regla indicada; si no s'indica es carregaran totes les regles ubicades sota la carpeta indicada a "rules_folder". Es poden consultar tots els altres paràmetres disponibles d'aquesta comanda a https://elastalert.readthedocs.io/en/latest/elastalert.html#running-elastalert

*A través de Systemd: crea un fitxer anomenat "/lib/systemd/system/elastalert.service" amb el següent contingut...:

```
[Unit]
Description=ElastAlert Service
After=elasticsearch.service
[Service]
Type=simple
WorkingDirectory=/home/usuari/elastalert   <--Carpeta on estigui "config.yaml"
ExecStart=/usr/local/bin/elastalert --verbose
[Install]
WantedBy=multi-user.target
```

... i seguidament inicia el servei de la forma habitual: *sudo systemctl start elastalert*

3.-a) Mantenint la mateixa configuració establerta a l'exercici anterior (servei Falco funcionant i Journalbeat recollint els seus missatges de registre per enviar-los a Logstash->ElasticSearch i visualitzar-los a Kibana), instal.la ara ElastAlert i configura'l per a què es llegeixi els arxius de regles que estiguin ubicats a la carpeta "/home/usuari/elastalert/lesmevesregles" (aquesta carpeta l'hauràs de crear, òbviament) i per a què connecti al servidor ElasticSearch local. Un cop gravada aquesta configuració, fes que l'ElastAlert funcioni com a servei Systemd, però no l'engegis encara (repassa els punts "1", "2", "3" i "5" de la teoria anterior per saber com fer tot això).

b) Crea una regla ElastAlert (punt "4" de la teoria anterior) que sigui de tipus "any" que envïi a algun servidor POST de prova (com ara http://ptsv2.com o https://docs.postman-echo.com) una alerta amb el text "*Alerta FALCO - {0} : ({1})*" on {0} representa el valor del camp @timestamp i {1} el valor del camp "message" del document/registre inspeccionat. Atenció: aquestes alertes han d'estar filtrades per enviar-se només en el cas què el document/registre en qüestió tingui un valor entre 1 i 3 en el seu camp "syslog.priority"

> NOTA: En el cas de fer servir el servidor Ptsv2, la URL que s'ha d'indicar com a valor de l'opció http_post_url ha de ser del tipus http://ptsv2.com/t/lalala/post, on "lalala" representa que és el nom del "toilet" que prèviament hauràs hagut de crear al formulari web que aquest servidor t'ofereix. En el cas de fer servir el servidor Postman-echo, es deixa com a exercici.

c) Engega el servei ElastAlert i provoca algun event sospitós a la màquina virtual (com ara executar *sudo touch /bin/virus* o bé *sudo touch /etc/fstab*) i seguidament comprova que l'alerta definida a l'apartat anterior arribi al servidor indicat.

> NOTA: Si no veus que arribi cap missatge al servidor POST escollit, observa els registres d'ElastAlert (*journalctl -e -u elastalert*) per intentar esbrinar què passa. Tingues present, de totes formes, que ElastAlert no envia les alertes inmediatament que Falco les registra sinó només quan fa la consulta periódica a ElasticSearch, i aquest moment ve donat per la directiva *run_every* de l'arxiu "config.yaml" (pots observar l'enviament "en directe" executant *journalctl -f -u elastalert*) . També cal tenir en compte que encara que hagis executat vàries vegades seguides diferents events sospitosos, ElastAlert no enviarà una alerta diferent per cada event sospitós (si concorden amb la mateixa regla i han ocorregut dins del període de temps establert per la directiva *realert*) sinó que n'enviarà una de sola per tots.

d) Crea una altra regla ElastAlert de tipus "frequency" que envïi al mateix servidor POST de prova una alerta amb el text "*Alerta FALCO – S'ha superat el llindar de 10 errors per minut*" quan Falco generi 10 o més documents/registres en un minut. Atenció: afegeix a la configuració de l'alerta l'opció general `query_key: syslog.priority` per a què el compte es faci segregant pel valor d'aquest camp (és a dir, per a què no es comptin tots els documents/registres junts sinò separats per prioritat, de manera que calgui arribar a 10 missatges d'una prioritat determinada). Seguidament provoca més de 10 events sospitosos a la màquina virtual (com ara executar *sudo touch /bin/virus* o bé *sudo touch /etc/fstab* moltes vegades seguides) i comprova a continuació que l'alerta corresponent hagi arribat al servidor indicat. Finalment, torna a provocar algun event sospitós més però sense arribar a llindar de 10 per minut i comprova que, en aquest cas, no s'envia cap alerta d'aquest tipus al servidor.

e) Obre Kibana en un navegador de la màquina real i afegeix com a "index pattern" l'índex que ha generat ElastAlert amb el registre d'alertes ("elastalert_status"). Inspecciona'l a la pestanya "Discover". ¿Què veus?

f) Suposant que ElasticSearch rebés documents/registres que tinguessin un camp anomenat "ip_address" que indiqués la direcció IP de l'origen d'aquest document/registre i un camp "username" que indiqués l'usuari del sistema que va generar (en aquest origen) el document/registre en qüestió, ¿per a què creus que serviria aquesta regla? (pots consultar https://elastalert.readthedocs.io/en/latest/ruletypes.html#cardinality)

```
name: Some rule
type: cardinality
index: logstash-*
max_cardinality: 1
cardinality_field: ip_address
timeframe:
  days: 1
query_key: username
filter: []
alert:
- email
email: ["elastalert@example.com"]
```

Integració d'Audit + Elastic Stack com a HIDS + SIEM

Una alternativa a Sysdig la tenim dins del propi kernel Linux i s'anomena Audit (https://github.com/linux-audit) . Els programes d'usuari que podem "acoplar com a sondes" al mòdul Audit del kernel per recollir els events i crides detectades (tant per les regles que haguem definit com per ser d'algun tipus recollit per defecte) són varis (i és millor que només es faci servir una per no "molestar-se" entre sí); ja coneixem el socket "systemd-journald-audit.socket" i també hem vist el servei "auditd", però també tenim el paquet "auditbeat", Beat proporcionat per l'equip de desenvolupadors de la suite Elastic, el qual permet centralitzar en un servidor ElasticSearch els events recollits de múltiples màquines. Altres característiques interessants d'AuditBeat són:

> *The Linux Audit Framework can send multiple messages for a single auditable event. For example, a rename syscall causes the kernel to send eight separate messages. Each message describes a different aspect of the activity that is occurring (the syscall itself, file paths, current working directory, process title). This module will combine all of the data from each of the messages into a single event.
>
> *Messages for one event can be interleaved with messages from another event. This module will buffer the messages in order to combine related messages into a single event even if they arrive interleaved or out of order.

Cal tenir en compte que, ja que AuditBeat substituirà al dimoni Auditd i també al "socket" systemd-journald-audit.socket, és molt recomanable deshabilitar completament ambdós "competidors" i, un cop configurat AuditBeat (o cada cop que canviem la seva configuració), iniciar-lo només a ell.

La configuració d'aquest dimoni es troba a l'arxiu "/etc/auditbeat/auditbeat.yml"; allà es pot indicar, a més de la sortida dels missatges (que serà normalment un servidor LogStash), el conjunt de regles personalitzades que es faran servir per registrar els events que volguem (substituïnt, per tant, la funcionalitat de *auditctl {-a|-w}* i *auditctl -R*. Un exemple del seu contingut podria ser el següent (veure https://www.elastic.co/guide/en/beats/auditbeat/current/configuring-howto-auditbeat.html): per tenir més informació:

```
auditbeat.modules:
- module: file_integrity
  paths:
  - /usr/bin
  - /usr/sbin
  - /etc/fstab
  exclude_files:
  - '/\.git($|/)'
  recursive: true
- module: auditd
  audit_rule_files: [ "/etc/audit/rules.d/*.conf" ]
  audit_rules: |
    -a exit,always -S clock_settime -k changetime
    -w /etc/passwd -p wra -k passwd
output.logstash:
  hosts: ["127.0.0.1:5044"]
```

Tal com es pot veure, AuditBeat pot gestionar actualment dos "mòduls" interns, anomenats "file_integrity" i "auditd" (no cal que s'utilitzin els dos obligatòriament): el primer utilitza la API Inotify del kernel per assabentar-se dels events relacionats amb els canvis en fitxers i carpetes del sistema i el segon monitoritza totes les crides al sistema en general.

Respecte el mòdul "file_integrity": l'única opció obligatòria és "paths". L'opció "exclude_files" és opcional (i tal com es pot veure, té com a valor una o més expressions regulars) i l'opció "recursive" (que per defecte és *false*) indica si es monitoritzaran també les subcarpetes

NOTA: At startup "file_integrity" module will perform an initial scan of the configured files and directories to generate baseline data for the monitored paths and detect changes since the last time it was run. It uses locally persisted data in order to only send events for new or modified files.

Respecte el mòdul "auditd": tal com es pot veure, les regles a utilitzar per filtrar els events recollits per aquest darrer segueixen la mateixa sintaxis que l'utilitzada amb la comanda *auditctl* i es poden escriure o bé directament dins del propi fitxer de configuració d'AuditBeat (una per línia), com a valors de l'opció "audit_rules" o bé agafar-se de fitxer/s extern/s on hi estiguin escrites (amb el mateix format que l'arxiu "audit.rules" ja conegut) via l'opció "audit_rule_files" (la qual admet comodins).

Per conèixer més opcions dels dos mòduls que les mostrades a l'exemple anterior es pot consultar, https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-module-file_integrity.html i https:// www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-module-auditd.html Recorda, a més, que com qualsevol altre Beat, se li poden definir "tags", "fields", "processors", etc (https://www.elastic.co/guide/en/beats/auditbeat/current/configuration-general-options.html

AuditBeat, a més, proporciona unes quantes comandes útils per consultar l'estat de les coses:

**auditbeat show auditd-rules* seria similar a *auditctl -l*
**auditbeat show auditd-status* seria similar a *auditctl -s*

EXERCICIS:

1.-a) Instal.la el paquet "auditbeat" (*apt/dnf install auditbeat*) i escriu una configuració que monitoritzi qualsevol canvi que pugui haver dins de la carpeta /etc de forma recursiva (consulta la teoria) i envïi els missatges al servidor Logstash + ElasticSearch + Kibana que ja deus tenir funcionant d'exercicis anteriors.

b) Provoca algun canvi dins d'aquesta carpeta i seguidament observa al panell web de Kibana si apareix algun missatge (dins de l'índex "logstash-*" que pertoqui) corresponent a aquest canvi. Observa l'estructura interna d'aquest missatge i fixa't sobre tot en el contingut dels camps "actor", "action", "@timestamp", "how", "thing" i "result" (si el missatge provingués d'una crida al sistema detectada) o bé el camp "file" amb diferents valors com "action", "type", "path","mtime","atime","uid", "hashed", etc (si el missatge prové de la monitorització d'un fitxer o carpeta, com és el cas).

NOTA: When monitoring files and folders, events generated by AuditBeat contain file metadata and cryptographic hashes of the file contents.The file hashes can be used to identify whether a specific file is in compliance across a fleet of machines. Or the hashes could be cross-referenced with threat intelligence sources to identify potential malware.

c) Crea una visualització en Kibana que mostri quines comandes són les que generen més events. Per fer això, crea un "Pie chart" amb una agregació de tipus "Terms", escollint com a camp a agregar "auditd.summary.how" i ordenant amb la mètrica "count".

d) Modifica la configuració del LogStash per descartar tots els events que no siguin intents d'inici de sessió per SSH (de qualsevol usuari, fallits o no). Una manera d'aconseguir-ho és utilitzant, per exemple, el filtre "drop" així: *filter { if [type] == "USER_AUTH" { drop{} } }*

NOTA: Aquest filtre no es pot fer a nivell d'AuditBeat perquè ja hem explicat que encara que no hi hagi cap regla definida, hi ha alguns events que Audit els genera sí o sí per imperatiu de les agències de seguretat, i els inicis de sessió és un d'aquests.

dII) Un cop gravats alguns events d'inici de sessió SSH (fallits o no) a l'índex ElasticSearch, crea una visualització en Kibana també de tipus "Pie chart" que mostri la quantitat d'intents d'inici fallits discriminant per IP d'origen i una altra visualització "Pie chart" que mostri la quantitat d'inicis de sessió correctes discriminant per nom d'usuari.

e) Defineix una alerta a ElastAlert de tipus "frequency" que envïi un missatge POST quan detecti més de cinc intents fallits d'inici de sessió per SSH en un minut.