FluentBit

Instal·lació manual a partir del codi font i posada en marxa inicial

FluentBit està als repositoris oficials de les distribucions importants però no en la seva darrera versió, així que preferirem compilar el seu codi font per tal d'instal·lar-lo "a mà"; això és molt senzill: només cal executar les següents comandes:

sudo dnf install git cmake flex bison gcc gcc-c++ systemd-devel libyaml-devel openssl-devel (a Fedora)
sudo apt install git cmake flex bison gcc g++ libsystemd-dev libyaml-dev libssl-dev (a Ubuntu)
git clone https://github.com/fluent/fluent-bit
cd fluent-bit/build
cmake ../
make
sudo make install

Un problema d'haver instal·lar FluentBit a partir del seu codi font és que aquest pot no venir amb cap fitxer "*.service" corresponent al servei Systemd corresponent, amb la qual cosa és possible que l'haguem de crear a mà. En concret, podem crear el fitxer "**/etc/systemd/system/fluent-bit.service**" així...:

[Unit]
Description=FluentBit
After=network-online.target
Requires=network-online.target
[Service]
ExecStart=/usr/local/bin/fluent-bit -c /usr/local/etc/fluent-bit/fluent-bit.conf
[Install]
WantedBy=multi-user.target

NOTA: Tal com es pot veure, en realitat el servei FluetBit ve implementat pel binari f*luent-bit*, el qual només necessita un paràmetre per funcionar, *-c*, el qual serveix per indicar el fitxer de configuració que farà servir (la sintaxi i significat del qual estudiarem properament). En tot cas, estem suposant que la carpeta "/usr/local/bin" (que és on el binari s'ubicarà automàticament en fer les instruccions d'instal·lació anteriors) està dins del PATH del sistema

...i finalment, iniciar (i activar) el servei executant la comanda *sudo systemctl start fluent-bit && sudo systemctl enable fluent-bit*

Per defecte la configuració de FluentBit un cop instal·lat de la forma acabada de descriure, està establerta per a recollir mètriques de l'ús de la CPU i enviar-les a la sortida estàndard cada 10 segons (és a dir, si el servei funciona en segon pla, les enviarà llavors al Journald). Per tant, una manera de comprovar que el servei funcioni és simplement fer *journalctl -f -u fluent-bit*

NOTA: A https://docs.fluentbit.io/manual/installation/getting-started-with-fluent-bit teniu més informació sobre altres mètodes d'instal·lació alternatius de FluentBit

Posada en marxa inicial a partir de la imatge Docker oficial

Si optem per fer servir la imatge Docker oficial de FluentBit, encara és més fàcil; només cal executar les següents comandes per implementar i posar en marxa un servidor FluentBit, respectivament:

touch $HOME/fluent-bit.conf
podman create --name mifluentbit -v $HOME/fluent-bit.conf:/fluent-bit/etc/fluent-bit.conf:Z cr.fluentbit.io/fluent/fluent-bit:latest
podman start mifluentbit

NOTA: El paràmetre *-v* serveix per mapejar un fitxer ubicat a la màquina amfitriona (que en aquest cas hem fet que sigui "$HOME/fluent-bit.conf", tot i que podria estar ubicat i anomenar-se de qualsevol altra manera) com arxiu de configuració del servidor FluentBit (el qual, a l'interior de la imatge està ubicat dins de la carpeta "/fluent-bit/etc" (el paràmetre -Z és per a què SELinux no doni problemes en sistemes Fedora). D'aquesta manera, només haurem de modificar el fitxer "extern" (sense haver de ser administrador!) per quan vulguem que el contenidor FluentBit, un cop reiniciat, es comporti d'una forma diferent.

NOTA: Recordeu que podrem fer servir qualsevol altra comanda típica de Docker/Podman com ara, entre d'altres:

podman stop mifluentbit : Atura el contenidor indicat
podman rm mifluentbit : Elimina el contenidor indicat
podman ps -a : Mostra la llista de tots els contenidors definits al sistema i el seu estat (en marxa, aturat, etc)
podman logs mifluentbit : Mostra els logs del contenidor indicat. Molt útil per veure els possibles errors (si no s'iniciés bé, per exemple) i també per veure la sortida de FluentBit si aquesta està definida com a "stdout" (perquè en estar funcionant en segon pla, aquesta sortida és redireccionada als logs directament)
podman image ls : Mostra la llista d'imatges descarregades al sistema
podman image rm cr.fluentbit.io/fluent/fluent-bit : Elimina la imatge indicada

També és possible executar un contenidor FluentBit efímer en primer pla; la gràcia d'això és que llavors podem indicar els paràmetres del binari *fluentbit* (com *-i*, *-o*, *-f*, *-c* etc...els anirem coneixent properament) directament al final de la comanda *docker/podman*, així:

podman run --rm -ti cr.fluentbit.io/fluent/fluent-bit:latest -i cpu -o stdout -f 1

Configuració

* Si hem fet la instal·lació de FluentBit "manual" seguint els passos marcats a l'apartat anterior, el seus arxius de configuració (perquè n'hi ha més d´ún) es trobaran ubicats a la carpeta "**/usr/local/etc/fluent-bit**" D'entre tots, l'arxiu més important, i que és el que ara mateix estudiarem, és "**fluent-bit.conf**".

* Si hem optat per la implementació via contenidors Docker/Podman, tal com hem realitzat el mapejat (i s'explica en una "nota" anterior), el fitxer de configuració corresponent s'anomenarà igual, "fluent-bit.conf", però estarà ubicat a la nostra pròpia carpeta personal.

En qualsevol cas, dins d'aquest arxiu poden aparèixer una o més de les següents seccions: *[SERVICE]*, *[INPUT]*, *[FILTER]* i *[OUTPUT]*, el significat de les quals de seguida estudiarem. Sota cada secció ha d'aparèixer com a mínim el nom d'una directiva (case-insensitive) amb el seu valor corresponent escrit a continuació (el qual pot ser un número, una cadena, un array o un map/hash/diccionari, segons la directiva en qüestió). Cada directiva cal escriure-la en una línia diferent amb la particularitat d'haver-se d'escriure identades respecte el títol de la secció a la què pertanyen (veurem exemples tot seguit). Sota les seccions també poden aparèixer comentaris, que són línies (també identades) començant per #.

* La secció *[SERVICE]* serveix per agrupar les directives que defineixen propietats general del propi servei Fluentbit com a tal. La llista completa de directives reconegudes en aquesta secció (i en les altres) es pot consultar a https://docs.fluentbit.io/manual/administration/configuring-fluent-bit/classic-mode/configuration-file tot i que al propi fitxer de configuració proporcionat per defecte ja n'apareixen indicades la majoria (amb la seva corresponent explicació). Només pot haver una secció [SERVICE]

* La secció *[INPUT]* serveix per agrupar les directives que defineixen un origen de dades (associat a un "plugin" de tipus "input"). Segons el tipus d'origen indicat, les directives a especificar aquí podran ser diferents però, en tot cas, sempre n'hi haurà d'haver com a mínim dues: la directiva ***Name*** (per indicar el nom del "plugin" de tipus "input" a utilitzar) i la directiva ***Tag*** (per indicar l'etiqueta que s'assignarà a tots els documents provinents d'aquest origen en concret). Hi pot haver més d'una secció [INPUT]

* La secció *[FILTER]* serveix per agrupar les directives que defineixen un filtre (associat a un "plugin" de tipus "filter") a aplicar als documents l'etiqueta dels quals concordin amb la directiva *Match/Match_Regex*. Segons el tipus de filtre indicat, les directives a especificar aquí podran ser diferents però, en tot cas, sempre n'hi haurà d'haver com a mínim dues: la directiva ***Name*** (per indicar el nom del "plugin" de tipus "filter" a utilitzar) i la directiva ***Match*** o bé ***Match_Regex*** (per indicar el patró a concordar amb les etiquetes dels documents entrants; en el primer cas són cadenes case-sensitive que suporten el comodí * i en el segon cas són expressions regulars; en el cas d'especificar les dues, *Match_Regex* té preferència). Hi pot haver més d'una secció [FILTER]

* La secció *[OUTPUT]* serveix per agrupar les directives que defineixen un destí de dades (associat a un "plugin" de tipus "output") a aplicar als documents l'etiqueta dels quals concordin amb la directiva *Match/Match_Regex*. Segons el tipus de destí indicat, les directives a especificar aquí podran ser diferents però, en tot cas, sempre n'hi haurà d'haver com a mínim dues: la directiva ***Name*** (per indicar el nom del "plugin" de tipus "filter" a utilitzar) i la directiva ***Match*** o bé ***Match_Regex*** (per indicar el patró a concordar amb les etiquetes dels documents entrants; en el primer cas són cadenes case-sensitive que suporten el comodí * i en el segon cas són expressions regulars; en el cas d'especificar les dues, *Match_Regex* té preferència). Hi pot haver més d'una secció [OUTPUT]

En resum: internament, cada nou línia de registre (o mètrica) que FluentBit obtingui de les entrades definides a la secció *[INPUT]* serà reconegut com un nou "document" (també a vegades anomenat "event"). A cada document entrant FluentBit li assignarà una etiqueta, que no és més que una cadena interna que s'utilitzarà al següent pas (concretament, a la secció *[FILTER]* i/o *[OUTPUT]*) per decidir per quin filtre/s i/o sortida/es haurà de passar el document en qüestió per ser processat i/o enrutat. La forma de decidir-ho és comprovant que hi hagi una concordància entre el valor de l'etiqueta del document en qüestió i el valor de la directiva *Match* (o bé *Match_Regex*) que estigui definit al filtre o sortida en qüestió: si n'hi ha, el document llavors serà processat/enrutat per aquell filtre/sortida concreta; si no, no (i si no n'hi ha cap més filtre/sortida per provar, llavors el document serà eliminat).

NOTA: Tal com hem dit, les etiquetes s'assignen manualment a l'entrada amb la directiva *Tag* però si no s'especifiqués cap etiqueta, Fluent Bit automàticament n'assignarà el nom de la instància del plugin d'entrada des d'on es va generar el document com la seva etiqueta

NOTA: A partir de què el document ha entrat per un plugin "input", per raons de rendiment FluentBit el representa internament amb el format "MessagePack" (https://msgpack.org), el qual es podria definir com una "versió binària" de JSON amb afegits, però això no és un detall que ara per ara ens hagi de preocupar i podem obviar-lo. Només cal saber que, internament, cada document sempre tindrà dos components (en forma d'array): *[TIMESTAMP, MESSAGE]* El "timestamp" representa l'hora en què es va crear un document (cada document conté una marca de temps associada, la qual és sempre un nombre enter fraccional numèric amb el format: SECONDS.NANOSECONDS. El "message" dependrà del tipus d'entrada utilitzada, a més dels eventuals filtres aplicats.

El següent exemple de fitxer de configuració mostra com recollir mètriques de la CPU i enviar-les (cada cinc segons, gràcies a la directiva ***Flush***) a la sortida estàndard:

```
[SERVICE]
   Flush     5
   #Si FluentBit és gestionat per Systemd, es recomana que la línia Daemon valgui off
   Daemon    off
   Log_Level debug
[INPUT]
   Name  cpu
   Tag   my_cpu
[OUTPUT]
   Name  stdout
   Match my*cpu
```

NOTA: Es pot dividir el fitxer de configuració en diversos fitxers petits (per comoditat) i "cridar-los" des del fitxer principal amb la línia *@INCLUDE unfitxer.conf* (si s'indica una ruta relativa, es pren des de la ubicació del fitxer principal). Aquesta línia cal indicar-la al principi del fitxer principal, fora de qualsevol secció (i per tant, no anirà indentada). Es permet l'ús del comodí *. Tingueu en compte que, independentment de l'ordre d'inclusió, sempre es respectarà el següent ordre en les seccions: Service, Input, Filter, Output

NOTA: La línia *@SET nomVariable=valor* defineix variables que es podran usar dins del fitxer de configuració a partir de llavors. Aquesta línia cal indicar-la al principi del fitxer principal, fora de qualsevol secció (i per tant, no anirà identada).

```
@SET my_input=cpu
@SET my_output=stdout
[SERVICE]
   Flush 1
[INPUT]
   Name ${my_input}
[OUTPUT]
   Name ${my_output}
```

NOTA: També es poden utilitzar dins del fitxer de configuració els valors que tinguin en el moment d'iniciar el servei Fluentbit les variables d'entorn indicades mitjançant la sintaxi següent (és case-sensitive): *${NOMVARIABLE}* Per exemple, després d'executar en un terminal la comanda *export MY_OUTPUT=stdout* podríem iniciar el servei Fluentbit amb la següent configuració:

```
[SERVICE]
   Flush      1
[INPUT]
   Name cpu
   Tag  cpu.local
[OUTPUT]
   Name  ${MY_OUTPUT}
   Match cpu.*
```

NOTA: En les darreres versions de Fluent-Bit s'està experimentant amb un format alternatiu pel seu arxiu de configuració; concretament, el format YAML, el qual es troba documentat (tot i que de moment no se n'aconsella el seu ús en producció) a https://docs.fluentbit.io/manual/administration/configuring-fluent-bit/yaml/configuration-file

Plugins de tipus "Input"

FluentBit pot rebre dades des de molts tipus d'orígens; tot depèn de quin/s "plugin"/s de tipus "input" tingui configurat/s. A continuació mencionarem els més comuns, amb les seves opcions més importants:

* Plugin "**Tail**" (https://docs.fluentbit.io/manual/pipeline/inputs/tail): Obté un nou document per cada nova línia que aparegui al final del/s fitxer/s desitjat/s, la ruta absoluta dels qual/s haurà d'indicar-se a la directiva ***Path*** (la qual pot aparèixer vàries vegades si es volen observar diversos fitxers, tot i que d'altra banda també cal saber que s'admet el comodí *). Aquest és el plugin que s'empra per monitoritzar la informació apareguda als fitxers de "log" de qualsevol programa que els guardi en un fitxer de text (com per exemple, el fitxer "access.log" de l'Apache, etc). Concretament, aquest "plugin" generarà per cada línia nova detectada en els fitxers monitoritzats un document JSON amb el format següent: *[tempsUNIX, { "log" => "contingut_de_la_nova_linia"}]*

Un exemple de configuració on s'usa aquest "plugin" (a més d'un "plugin" de tipus "Output" anomenat "Stdout", del qual parlarem de seguida i cap "plugin" de tipus "Filter") és el següent (on no s'ha especificat cap "tag" perquè la directiva *Match* indicada fa ús del comodí *, que arreplega qualsevol missatge):

```
[INPUT]
   Name  tail
   Path  /var/log/*.log
[OUTPUT]
   Name  stdout
   Match  *
```

NOTA: És molt convenient afegir sempre en la configuració d'aquest "plugin" la directiva ***Db*** (la qual tindrà com a valor associat la ruta absoluta d'un fitxer qualsevol). Si no s'indica aquesta directiva, cada cop que es reinicïi FluentBit, aquest començarà a llegir a partir de la nova línia que s'hi afegeixi al final del fitxer "log" en qüestió però totes les línies que hi poguessin haver-se guardat al fitxer mentre FluentBit no estava encès no, perquè FluentBit no sap quina va ser la darrera línia llegida en l'arranc anterior...a no ser que s'usi la directiva Db; el fitxer indicat en aquesta directiva justament representa la "base de dades" que conté l'estat del cursor de lectura de FluentBit en cadascun dels fitxers monitoritzats (el que se'n diu l'"offset"). D'aquesta manera FluentBit pot saber en tot moment quina és la darrera línia llegida i, per tant, no perdre's cap tot i estar aturat en algun moment. De fet, en realitat, el fitxer indicat per la directiva Db és una base de dades en format SQLite3, així que pot ser inspeccionada (sempre que FluentBit estigui apagat, per evitar corrupcions de dades) amb la comanda *sqlite3 fitxer.db* i, un cop dins, fent *.headers on ; .mode column ; .width 5 32 12 12 10 ; select * from in_tail_files;*

NOTA: Altres directives interessants d'aquest "plugin" són, per exemple, ***Read_from_Head*** (que per defecte és *false*, però si valgués *true* fa que, si no hi ha cap cursor *Db* definit, la lectura del/s fitxer/s indicat/s es realitzi des del seu començament) o ***Exit_On_Eof*** (que per defecte és *false*, però si valgués *true* fa que Fluentbit no continuï monitoritzant les futures línies que apareguin en el fitxer indicat sinó que llegeixi fins el final del fitxer el contingut que hi hagi en aquest moment i prou; això és útil per realitzar entrades massives de dades), entre altres.

* Plugin "**Systemd**" (https://docs.fluentbit.io/manual/pipeline/inputs/systemd): Obté un nou document per cada nova línia que aparegui al final del Journal del sistema. D'entre les directives pròpies d'aquest "plugin", podem destacar ***Systemd_Filter***, el qual permet especificar una parella clau-valor pròpia del Journal per tal d'obtenir només les línies corresponents a aquesta (per exemple, per només obtenir les línies del journal generades pel servei Cups podríem establir la parella *_SYSTEMD_UNIT=cups.service* Aquesta directiva es pot indicar més d'una vegada; per defecte, la concatenació de les diverses directives *Systemd_Filter* existents es tractarà com un "O" lògic; si es vol tractar-les com un "I" lògic, llavors caldrà afegir a més la directiva ***Systemd_Filter_Type=And*** Aquest "plugin" generarà per cada event del Journal detectat un document JSON amb el format següent: *[tempsUNIX, { "_NOM_CAMP1JOURNAL" => "valor1", "_NOM_CAMP2JOURNAL" => "valor2"}]*

Un exemple de configuració on s'usa aquest "plugin" (a més d'un "plugin" de tipus "Output" anomenat "Stdout", del qual parlarem de seguida i cap "plugin" de tipus "Filter") és el següent (on no s'ha especificat cap "tag" perquè la directiva *Match* indicada fa ús del comodí *, que arreplega qualsevol missatge):

```
[INPUT]
   Name   systemd
   Systemd_Filter  _SYSTEMD_UNIT=cups.service
[OUTPUT]
   Name   stdout
   Match  *
```

NOTA: És molt convenient afegir sempre en la configuració d'aquest "plugin" la directiva ***Db*** (la qual tindrà com a valor associat la ruta absoluta d'un fitxer qualsevol). Si no s'indica aquesta directiva, cada cop que es reinicïi FluentBit començarà a llegir des del principi del Journal perquè no sabrà quina va ser la darrera línia llegida en l'arrancanterior; precisament el fitxer indicat a la directiva *Db* representa la "base de dades" que conté l'estat del cursorde lectura de FluentBit en el Journal del sistema (el que se'n diu l'"offset"). De fet, en realitat, el fitxer indicat per la directiva Db és una base de dades en format SQLite3, així que pot ser inspeccionada (sempre que FluentBit estigui apagat, per evitar corrupcions de dades) amb la comanda *sqlite3 fitxer.db* i, un cop dins, fent *.headers on ; .mode column ; .width 5 32 12 12 10 ; select * from in_tail_files;*

NOTA: Un altre "plugin" de tipus "Input" que pot ser útil si l'anterior no ens és còmode per segons quins tipus de missatges és "**kmsg**" (https://docs.fluentbit.io/manual/pipeline/inputs/kernel-logs)

També són força útils els següents "plugins":

* Plugin "**Stdin**" (https://docs.fluentbit.io/manual/pipeline/inputs/standard-input): Obté un nou document per cada nova línia que aparegui a l'entrada estàndard. Aquest "plugin" sobretot és útil quan s'executa FluentBit en primer pla, des del terminal, i la sortida de l'origen de dades se li entuba. No obstant, aquest origen de dades ha d'enviar les dades formatades en JSON per a què aquest "plugin" les reconegui i generi llavors el document JSON pertinent, que tindrà el format següent:
[tempsUNIX, { clau1 => valor1, clau2 => valor2, clau3 => valor3 }]

* Plugin "**Exec**" (https://docs.fluentbit.io/manual/pipeline/inputs/exec): Obté un nou document per cada nova línia que aparegui a la sortida estàndard de la comanda especificada mitjançant la directiva ***Command***, la qual s'executarà automàticament cada tants segons com s'indiqui a la directiva ***Interval_Sec*** Aquest "plugin" generarà per cada línia un document JSON amb el format següent:
[tempsUNIX, { "exec" => "sortidaLinia1"}]

NOTA: Un altre "plugin" de tipus "Input" que pot ser útil si l'anterior no ens és còmode per quan volguem executar la comanda *cat* amb determinats fitxers és "**Head**" (https://docs.fluentbit.io/manual/pipeline/inputs/head) el qual com a directives més destacades té ***File*** (per indicar l'arxiu d'on obtenir les línies), ***Lines (***per indicar el nombre de línies a obtenir) i ***Interval_Sec*** (per indicar cada quants segons es repetirà l'obtenció de les línies)

Per fer proves ràpides són interessants també els següents "plugins":

* Plugin "**Dummy**" (https://docs.fluentbit.io/manual/pipeline/inputs/dummy): Genera nous documents, amb un contingut JSON personalitzable mitjançant la directiva ***Dummy*** (si no s'indica, aquest és *{"message" => "dummy"}* ***)*** La quantitat de documents generats per segons ve donada per la directiva ***Rate*** i la quantitat total per la directiva ***Samples*** (si aquesta no s'indica, la quantitat és infinita). Aquest "plugin" generarà per defecte documents JSON amb el format següent: *[tempsUNIX, { "message" => "dummy"}]*

* Plugin "**Random**" (https://docs.fluentbit.io/manual/pipeline/inputs/random): Genera nous documents, amb un contingut numèric aleatori. El període (en segons) entre generació i generació de documents ve donat per la directiva ***Interval_Sec*** i la quantitat total per la directiva ***Samples*** (si aquesta no s'indica, la quantitat és infinita). Aquest "plugin" generarà documents JSON amb el format següent: *[tempsUNIX, { "rand_value" => valorAleatori}]*

FluentBit no només és capaç d'obtenir dades provinents de "logs" sinó també pot obtenir mètriques del sistema on està funcionant. En aquest sentit, els següents "plugins" poden ser útils per obtenir dades sobre el funcionament de la pròpia màquina:

* Plugin "**CPU**" (https://docs.fluentbit.io/manual/pipeline/inputs/cpu-metrics): Genera nous documents contenint mètriques de l'ús de la CPU (en %) per part de tot el sistema (o bé d'un procés en particular si s'indica el seu PID a la directiva ***PID***). El període (en segons) entre obtenció i obtenció de documents ve donat per la directiva ***Interval_Sec*** Concretament, s'obtenen dades pel % d'ús en mode usuari, en mode kernel i de CPU total, tant a nivell de CPU global com per cada "core" individualment. Aquest "plugin" generarà documents JSON amb el format següent: *[tempsUNIX, {"cpu_p"=>x, "user_p"=>x, "system_p"=>x, "cpu0.p_cpu"=>x, "cpu0.p_user"=>x, "cpu0.p_system"=>x, "cpu1.p_cpu"=>x, "cpu1.p_user"=>x, "cpu1.p_system"=>x}]*

* Plugin "**Mem**" (https://docs.fluentbit.io/manual/pipeline/inputs/memory-metrics): Genera nous documents contenint mètriques de l'ús de la memòria per part del sistema. El període (en segons) entre obtenció i obtenció de documents ve donat per la directiva ***Interval_Sec*** Aquest "plugin" generarà documents JSON amb el format següent: *[tempsUNIX, {"Mem.total"=>x, "Mem.used"=>x, "Mem.free"=>x, "Swap.total"=>x, "Swap.used"=>x, "Swap.free"=>x}]*

  NOTA: Si es vol saber l'ús de la memòria que en fa un procés en particular, cal utilitzar llavors el plugin "**Proc**" (https://docs.fluentbit.io/manual/pipeline/inputs/process)

* Plugin "**Netif**" (https://docs.fluentbit.io/manual/pipeline/inputs/network-io-metrics): Genera nous documents contenint mètriques del tràfic rebut i enviat per totes les tarjes de xarxa del sistema (o bé d'una tarjeta en particular si s'indica el seu nom a la directiva ***Interface***). El període (en segons) entre obtenció i obtenció de documents ve donat per la directiva ***Interval_Sec*** Aquest "plugin" generarà documents JSON amb el format següent: *[tempsUNIX, {"nomTarja.rx.bytes"=>x, "nom.rx.packets"=>x, "nom.rx.errors"=>x, "nom.tx.bytes"=>x, "nom.tx.packets"=>x, "nom.tx.errors"=>x}]*

* Plugin "**Disk**" (https://docs.fluentbit.io/manual/pipeline/inputs/disk-io-metrics): Genera nous documents contenint mètriques del tràfic llegit i escrit en tots els discos del sistema (o bé d'un disc en particular si s'indica el seu nom a la directiva ***Dev_Name***). El període (en segons) entre obtenció i obtenció de documents ve donat per la directiva ***Interval_Sec*** Aquest "plugin" generarà documents JSON amb el format següent: *[tempsUNIX, {"read_size"=>x, "write_size"=>x}]*

NOTA: "**Thermal**" (https://docs.fluentbit.io/manual/pipeline/inputs/thermal) és un altre "plugin" interessant

Cal destacar, finalment, que FluentBit pot rebre dades d'orígens remots gràcies a diferents "plugins" de tipus "Input", els quals permeten posar a l'escolta un servidor embegut ja preparat per rebre aquestes dades per la xarxa (i emprant certs protocols concrets). Això permet agregar de forma centralitzada dades provinents de diferents origens en un sol sistema FluentBit (el qual, al seu torn, les podria reenviar a un altre destí emprant el "plugin" de tipus "Output" adient (que de seguida estudiarem). Podem destacar-ne:

* Plugin "**HTTP**" (https://docs.fluentbit.io/manual/pipeline/inputs/http): Obre el port indicat a la directiva ***Port*** (per defecte, 9880) i el vincula a la adreça IP indicada a la directiva ***Host*** (per defecte, "0.0.0.0") per tal de permetre rebre dades JSON de l'exterior transportades via HTTP. Aquest "plugin" permet etiquetatge dinàmic: això significa que es poden enviar documents amb etiquetes diferents al mateix port (i un cop rebuts ja s'enrutaran pel filtre o sortida adient segons concordi). Per adjuntar una etiqueta determinada a un missatge cal indicar-la al final de la URL de la petició (si no s'indica, s'utilitzarà automàticament l'etiqueta "http.*nºInput*"). Per exemple, si utilitzem *curl* podem enviar una dada etiquetada com "app.log" així: *curl -d @fitxerQueConteDades.json -X POST -H "Content-Type: application/json" http://ip.flu.ent.bit:9880/app.log* (on el fitxer *.json ha de contenir cada objecte JSON en una línia diferent, objectes que el "plugin" generarà en sengles documents diferents adjuntant-lis a cadascun el temps UNIX de la rebuda.

  NOTA: El plugin "**MQTT**" (https://docs.fluentbit.io/manual/pipeline/inputs/mqtt) és similar a l'anterior però empra el protocol MQTT enlloc de HTTP (un client adient per connectar-hi podria ser *mosquitto*, doncs) . Un altre plugin interessant és "**Forward**" (https://docs.fluentbit.io/manual/pipeline/inputs/forward), el qual implementa un protocol propi de FluentBit i el seu "cosí-germà" FluentD optimitzat per la comunicació entre diferents nodes que executin aquests programes. Si, no obstant, no es vol utilitzar cap protocol de nivell d'aplicació, sempre es pot utilitzar el plugin "**TCP**" (https://docs.fluentbit.io/manual/pipeline/inputs/tcp), el qual permet treballar directament al nivell de transport i és capaç de reconéixer tant missatges formatats en JSON com en format lliure; en aquest cas, un client adient per connectar-hi podria ser *netcat*

NOTA: Hi ha més "plugins" de tipus "Input". La llista completa es troba a https://docs.fluentbit.io/manual/pipeline/inputs

"Parsers" pel plugin *Tail* (i *Exec*)

Hi ha certs "plugins" de tipus "Input" que, per la seva natura, solen emetre missatges desestructurats. És el cas dels "plugins" *Tail* o *Exec* per exemple. Si és el aquest el cas, és possible afegir una directiva sota la seva secció *[INPUT]* corresponent anomenada ***Parser*** indicant el nom d'un "parser" existent dins del fitxer **parsers.conf** (ubicat a la carpeta "**/usr/local/etc/fluent-bit**" -o, si es tracta d'un contenidor Docker/Podman, a la carpeta interna "**/fluent-bit/etc**"-). Aquest "parser" serà l'encarregat de "repassar" la forma de les línies rebudes i generar-ne, a partir del seu reconeixement, diversos camps estructurats per tal de generar el document final (abans de passar pels plugins de tipus "Filter" i "Output") amb aquests camps en lloc de les dades originals. D'aquesta manera, el document processat tindrà una estructura coneguda de parelles claus<-> valor concretes i fàcilment indexable, cercable i manipulable.

NOTA: Els "plugins" *Stdin* i *Http* podrien ser altres candidats a fer servir "parsers" però en aquest cas no tenen implementada la funcionalitat de fer-los servir, així que estarien obligats a rebre dades JSON sí o sí. No obstant, en aquests casos en què els plugins de tipus "Input" no poden utilitzar "parsers", existeix la possibilitat d'usar un plugin de tipus "Filter" anomenat precisament "parser" que permet justament això: utilitzar el "parser" desitjat per realitzar les mateixes tasques de "repassada" i "divisió" de documents (però en aquest segon "stage")

NOTA: Amb la directiva ***Parsers_File*** de la secció [SERVICE] es pot indicar la ruta d'un altre fitxer extra de definició de "parsers" (o més d'un perquè aquesta directiva es pot indicar varis cops) si no es vol canviar el fitxer "parsers.conf" original

Destacarem els "parsers" ja predefinits que podem usar en qualsevol entrada on els necessitem:

* Parser anomenat "**json**" (https://docs.fluentbit.io/manual/pipeline/parsers/json): D'una entrada tal com *{"key1": 12345, "key2": "abc", "time": "2006-07-28T13:22:04Z"}* n'obté un document tal com *[1154103724, {"key1"=>12345, "key2"=>"abc"}]* Cal tenir en compte que per defecte aquest "parser" interpreta que el valor del camp que s'anomeni "time" serà assimilat al temps Unix del document (valor que, per cert, a més ha d'estar en el format indicat a l'exemple). Si això no és així, caldrà dissenyar un altre "parser" personalitzat a mà, també de tipus "json" però amb un altre nom i indicant els valors adient als camps de configuració *Time_Key* i *Time_Format* (veure més avall)

* Parser anomenat "**logfmt**" (https://docs.fluentbit.io/manual/pipeline/parsers/logfmt): D'una entrada tal com *key1=val1 key2=val2* (és a dir, en format "Logfmt", https://brandur.org/logfmt) n'obté un document tal com *[1540936693, {"key1"=>"val1", "key2"=>"val2"}]* Cal tenir en compte, però, que aquest "parser", tot i estar reconegut per FluentBit, no està definit en el fitxer "parsers.conf", així que caldria afegir-lo a mà.

* Parsers basats en expr. reg (https://docs.fluentbit.io/manual/pipeline/parsers/regular-expression): A l'arxiu "parsers.conf" n'hi ha uns quants definits, com per exemple els anomenats "**apache2**", "**apache_error**", "**nginx**", "**mysql_error**", etc. En general, tots ells es basen en el motor d'expressions generals Onigmo (https://github.com/k-takata/Onigmo), que és una llibreria desenvolupada en el llenguatge de programació Ruby (per veure'ns les seves especificitats, es pot practicar amb https://rubular.com)

A l'apartat següent, quan estudiem el filtre "parser", veurem com crear "parsers" personalitzats a mà, tant de tipus "json" com, sobre tot, basats en expressions regulars.

Plugins de tipus "Filter"

Un cop rebudes les dades de l'entrada que sigui, FluentBit té la capacitat de processar-les abans de reenviar-les a un altre destí. Per "processar" volem dir realitzar un munt d'operacions sobre les dades per tal de transformar-les de la manera que ens sigui més útil: podem eliminar camps concrets dels documents JSON o afegir-ne de nous, podem eliminar documents sencers segons un cert criteri, podem canviar el nom o el valor de camps concrets, podem "parsejar" el document per tal de localitzar i separar dades concretes del document i dividir-lo en trossos, etc, etc. Per decidir quins documents reben el processament de quins filtres, s'utilitza la seva etiqueta respectiva i les directives *Match/Match_Regex* pertinents.

* Plugin "**Grep**" (https://docs.fluentbit.io/manual/pipeline/filters/grep): Si s'usa la directiva ***Exclude nomClau valor***, elimina tots els documents rebuts pel filtre (gràcies a la directiva *Match/Match_Regex*) que tinguin el valor indicat (en forma d'expressió regular) en la clau JSON indicada. Si, en canvi, s'usa la directiva ***Regex nomClau valor*** fa al revés: manté només (eliminant la resta) els documents rebuts que tinguin el valor indicat (en forma d'expressió regular) en la clau JSON indicada.

  NOTA: En el cas que la clau JSON sigui anidada, la sintaxi per indicar el seu nom serà **$nomClau1["nomClau2"]["nomClau3"]**, etc

* Plugin "**Record_modifier**" (https://docs.fluentbit.io/manual/pipeline/filters/record-modifier): Si s'usa la directiva ***Record nomClau valor***, afegeix la clau indicada amb el valor indicat a tots els documents rebuts pel filtre (gràcies a la directiva *Match/Match_Regex*). Si s'usa la directiva ***Remove_key nomClau*** s'eliminarà la parella *clau<->valor* amb el nom de clau indicat de tot els documents rebuts pel filtre; es pot especificar més d'un cop aquesta directiva per així eliminar vàries parelles *clau<->valor*. Si s'usa, en canvi, la directiva ***Allowlist_key nomClau*** s'eliminarà, de tots els documents rebuts pel filtre, la parella *clau<->valor* que no sigui la indicada; es pot especificar més d'un cop aquesta directiva per així especificar explícitament les diverses parelles *clau<->valor* que es vulguin mantenir.

* Plugin "**Modify**" (https://docs.fluentbit.io/manual/pipeline/filters/modify): Si s'usa la directiva ***Add nomClau valor***, afegeix la clau indicada amb el valor indicat a tots els documents rebuts pel filtre (gràcies a la directiva *Match/Match_Regex*); en aquest sentit aquesta directiva seria equivalent a l'anomenada *Record* del plugin "Record_modifier"; la diferència de la directiva anterior amb la directiva ***Set nomClau valor*** és que amb *Add*, si la clau indicada ja existís, no passaria res però amb *Set*, es sobreescriuria el valor preexistent. Si s'usa la directiva ***Remove nomClau***, en canvi, s'eliminarà la parella *clau<->valor* amb el nom de clau indicat de tot els documents rebuts pel filtre (es pot especificar més d'un cop aquesta directiva per així eliminar vàries parelles *clau<->valor* o també usar la directiva ***Remove_wildcard nomClau***, que permet l'ús del comodí * a l'hora d'indicar

el nom de la clau, o la directiva ***Remove_regex nomClau***, que permet l'ús d'expressions regulars a l'hora d'indicar el nom de la clau); en aquest sentit aquesta directiva seria equivalent a l'anomenada *Remove_key* del plugin "Record_modifier". Si s'usa la directiva ***Rename nomClauAntic nomClauNou*** s'estarà renombrant la clau indicada pel nou nom indicat; la diferència de la directiva anterior amb la directiva ***HardRename nomClauAntic nomClauNou*** és que amb *Rename*, si la nom nou de la clau indicada ja existís, no passaria res però amb *HardRename*, es sobreescriuria. Si s'usa la directiva ***Copy nomClau nomClauNou*** s'estarà copiant el parell *clau<->valor* corresponent a la clau "nomClau" en un altre parell del mateix document, on la seva clau tindrà el nom "nomClauNou". La diferència de la directiva anterior amb la directiva ***HardCopy nomClau nomClauNou*** és que amb *Copy*, si la clau "còpia" indicada ja existís, no passaria res però amb *HardCopy*, s'hi sobreescriuria el valor preexistent. Múltiples regles es poden definir, les quals seran processades en ordre. Cada regla operarà sobre el resultat obtingut de la regla immediatament precedent.

NOTA: Una funcionalitat molt interessant d'aquest "plugin" és la capacitat de realitzar les modificacions indicades sempre i quan es compleixin determinades condicions indicades mitjançant la directiva ***Condition una_condicio***, escrita abans de les directives descrites al paràgraf anterior (també poden escriure's diverses directives *Condition* , cadascuna en una línia diferent, però, en tot cas, totes elles hauran de retornar *true* per a què les directives *Add/Set/Remove/Rename/Copy/etc* -escrites sempre a continuació, tal com es pot veure a la configuració d'exemple següent- s'executin). Exemples de condicions que es poden indicar són:

Key_exists nomClau : Retorna *true* si la clau indicada existeix
Key_does_not_exist nomClau : Retorna *true* si la clau indicada NO existeix
A_key_matches exprReg : Retorna *true* si existeix alguna clau amb un nom concordant amb l'expr. reg. indicada
No_key_matches exprReg : Retorna *true* si NO existeix cap clau amb un nom " amb l'expr. reg. indicada
Key_value_equals nomClau valor : Retorna *true* si la clau indicada té el valor indicat
Key_value_does_not_equal nomClau valor : Retorna *true* si la clau indicada NO té el valor indicat
Key_value_matches nomClau exprReg : Retorna *true* si la clau indicada té un valor que concorda amb l'expressió regular indicada
Key_value_does_not_match nomClau exprReg : Retorna *true* si la clau indicada té un valor que NO concorda amb l'expressió regular indicada

```
[INPUT]
   Name mem
   Tag  mem.local
   Interval_Sec 1

[FILTER]
   Name    modify
   Match   mem.*
   Condition Key_Does_Not_Exist cpustats
   Condition Key_Exists Mem.used
   Set cpustats UNKNOWN

[FILTER]
   Name    modify
   Match   mem.*
   Condition Key_Value_Does_Not_Equal cpustats KNOWN
   Add sourcetype memstats

[FILTER]
   Name    modify
   Match   mem.*
   Rename Mem.free MEMFREE

[FILTER]
   Name    modify
   Match   mem.*
   Condition Key_Value_Equals cpustats UNKNOWN
   Remove_wildcard Mem
   Remove_wildcard Swap
   Add cpustats_more STILL_UNKNOWN

[OUTPUT]
   Name   stdout
   Match  *
```

* Plugin "**rewrite_tag**" (https://docs.fluentbit.io/manual/pipeline/filters/rewrite-tag): Reetiqueta tots els documents rebuts pel filtre (gràcies a la directiva *Match/Match_Regex*) amb una nova etiqueta indicada amb la directiva ***Rule $nomClau exprReg novaEtiqueta keep*** ; els valors *$nomClau exprReg* permeten filtrar els documents que es reetiquetaran, que seran només aquells que tinguin una clau amb el nom indicat amb un valor que concordi amb l'expressió regular indicada; el valor *novaEtiqueta* indica la nova etiqueta (la qual pot estar composta d'una cadena qualsevol i/o l'etiqueta antiga -indicada com *$TAG-* i/o alguna variable d'entorn -indicada com *${nomVariable}-*, etc) i *keep* és en realitat un valor booleà (i, per tant, pot valer *true* o *false*; en el primer cas el document amb l'etiqueta antiga es manté un cop s'ha fet la còpia del mateix amb la nova etiqueta (i, per tant, continua el seu processament "com si no hagués passat res") i en el segon cas, s'elimina. Múltiples regles es poden definir, les quals seran processades en ordre fins que alguna concordi.

*Plugin "**Parser**" (https://docs.fluentbit.io/manual/pipeline/filters/parser): Permet utilitzar "parsers" (o bé predefinits o bé creats per nosaltres mateixos a partir d'expressions regulars definides manualment) per tal de dividir un document amb contingut desestructurat en diversos camps. Les seves directives més importants són ***Key_Name nomCamp*** (per indicar el camp desestructurat a "parsejar" del document que entri al filtre) i ***Parser nomParser*** (per indicar el nom del "parser" a utilitzar sobre ell). A més es poden afegir les directives ***Preserve_key true*** (per indicar que es vol mantenir el camp original *Key_Name* al document; per defecte s'elimina) i/o ***Reserve_data true*** (per indicar que es vol mantenir la resta de camps originals del document; per defecte s'eliminen tots)

NOTA: Per tal de crear un "parser" personalitzat, al fitxer "parsers.conf" oficial (o bé al fitxer indicat a la directiva *Parsers_file* de la secció [SERVICE] del fitxer de configuració de Fluent-Bit), cal afegir una secció titulada *[PARSER]* amb les següents directives: ***Name nomParser*** , ***Format { json | logfmt | regex }*** , ***Time_Key nomCampUsatComTimestamp*** , ***Time_Format formatDelCampAnterior*** (es poden indicar els valors descrits a *man strptime*) i, en el cas d'un parser de tipus "regex", ***Regex exprReg*** (on ja hem dit que l'expressió regular indicada ha de ser compatible amb https://rubular.com) i, en el cas de parsers de tipus "regex" i "logfmt", ***Type camp1:tipus1 camp2:tipus2*** (per indicar el tipus dels diferents camps generats després del parsejament; els tipus poden ser "string" -per defecte-, "integer", "bool", "float" i "hex").

NOTA: Altres "plugins" de tipus "Filter" interessants a destacar són el plugin "**GeoIP**" (el qual afegeix un camp amb informació geolocalitzada a partir d'algun valor de tipus adreça IP present prèviament en algun camp concret, https://docs.fluentbit.io/manual/pipeline/filters/geoip2-filter), el plugin "**Lua**" (el qual permet desenvolupar els nostres propis "plugins", https://docs.fluentbit.io/manual/pipeline/filters/lua) o el plugin de depuració "**Expect**" (el qual permet comprovar si l'estructura dels documents respon a l'esperat observant determinades característiques, com per exemple si conté o no una determinada clau, si una clau conté o no determinat valor, si un valor és *null* o no, etc, https://docs.fluentbit.io/manual/pipeline/filters/expect). A https://docs.fluentbit.io/manual/pipeline/filters es troba la llista completa.

Plugins de tipus "Output"

FluentBit pot reenviar les dades ja obtingudes i processades cap a molts tipus de destins; tot depèn de quin/s "plugin"/s de tipus "output" tingui configurat/s. A continuació mencionarem els més comuns, amb les seves opcions més importants:

* Plugin "**Stdout**" (https://docs.fluentbit.io/manual/pipeline/outputs/standard-output): Envia a la sortida estàndard el document rebut per alguna entrada (i, opcionalment, processat per algun filtre) on se li hagi assignat una etiqueta que concordi amb el valor *Match/Match_Regex* definit en la secció [OUTPUT] en qüestió. Per cada document, a més d'ell en sí (en el format *[tempsUNIX, {...}]*) es mostrarà la seva etiqueta (abans del document en sí, en el format *etiqueta:*) i un codi numèric (al principi de la línia, entre claudàtors) que representa l'ordre del document dins de la remesa de cada "flush". De tota manera, el format concret en què aquest document es mostrarà a pantalla es pot definir mitjançant la directiva ***Format***, la qual pot tenir, entre d'altres, els valors: ***msgpack*** (per defecte), ***json*** (tots els documents enviats en un mateix "flush" s'agrupen en un array de documents), ***json_lines*** (cada document d'un mateix "flush" s'envia separat de la resta en línies diferents) i ***json_stream*** (cada document d'un mateix "flush" s'envia separat de la resta però tots en la mateixa línia, un rera l'altre)

* Plugin "**File**" (https://docs.fluentbit.io/manual/pipeline/outputs/file): Envia a un fitxer (el nom del qual s'ha d'indicar mitjançant la directiva ***File*** i la ruta absoluta de la carpeta on s'ubica s'ha d'indicar mitjançant la directiva ***Path***) el document rebut per alguna entrada (i, opcionalment, processat per algun filtre) on se li hagi assignat una etiqueta que concordi amb el valor *Match/Match_Regex* definit en la secció [OUTPUT] en qüestió. El format en què aquest document s'imprimirà es pot definir mitjançant la directiva ***Format***, la qual pot tenir, entre d'altres, els valors...:

  * ***out_file*** : les línies apareixeran així: *tag: [time, {"key1":"value1", "key2":"value2"}]*
  * ***plain*** : les línies apareixeran així: *{"key1":"value1", "key2":"value2"}*
  * ***csv*** : les línies apareixeran així: *time[delimiter]"value1"[delimiter]"value2"* , on el caràcter delimitador vindrà definit per la directiva extra ***Delimiter*** (per defecte és la coma)
  * ***template*** : les línies apareixeran amb un format personalitzat (per més informació, mireu https://docs.fluentbit.io/manual/pipeline/outputs/file#template-format)

* Plugin "**Null**" (https://docs.fluentbit.io/manual/pipeline/outputs/null): Elimina el document rebut amb l'etiqueta concordant amb el valor *Match/Match_Regex* definit a la secció [OUTPUT] en qüestió

El més normal, però, és que FluentBit reenvïi les dades a destins remots gràcies a diferents "plugins" de tipus "Output", els quals permeten connectar-s'hi a la xarxa emprant diferents protocols. La intenció més habitual en aquests casos és voler connectar amb algun tipus de servidor de base de dades (relacional o no) per tal d'emmagatzemar-hi les dades, ja recollides i processades per Fluentbit. En aquest sentit, podem destacar els següents "plugins" d'aquest tipus:

* Plugin "**HTTP**" (https://docs.fluentbit.io/manual/pipeline/outputs/http): Envia (fent servir el mètode POST) a un servidor HTTP remot (l'adreça IP del qual s'ha d'indicar mitjançant la directiva ***Host,*** el port de destí amb la directiva ***Port*** i, opcionalment, el path de la URL de destí, amb la directiva ***Uri***) el document rebut per alguna entrada (i, probablement, processat per algun filtre) on se li hagi assignat una etiqueta que concordi amb el valor *Match/Match_Regex* definit en la secció [OUTPUT] en qüestió. Aquest document pot ser enviat en diversos formats però es recomana que sigui JSON; per això caldrà indicar-ho així mitjançant la directiva ***Format***. Òbviament, el servidor HTTP de destí haurà d'estar programat per rebre dades via POST i saber-les interpretar; un servidor de prova per fer experiments podria ser https://public.requestbin.com/r Per tal d'interaccionar amb un servidor HTTPS, en https://docs.fluentbit.io/manual/pipeline/outputs/tcp-and-tls#tls-configuration-parameters (o també https://docs.fluentbit.io/manual/administration/security) es poden veure els paràmetres TLS necessaris a afegir, els quals són bàsicament establir ***Tls*** a ***On*** i, a més, establir ***Tls.verify*** a ***Off*** (si el servidor remot té un certificat autosignat) o bé indicar la ruta del certificat arrel d'una CA amb ***Tls.ca_file*** (si el servidor remot té un certificat signat per aquesta CA en qüestió).

* Plugin "**opensearch**" (https://docs.fluentbit.io/manual/pipeline/outputs/opensearch): Envia a un servidor OpenSearch remot (l'adreça IP del qual s'ha d'indicar mitjançant la directiva ***Host***, el port de destí amb la directiva ***Port***, el nom de l'índex de destí, amb la directiva ***Index*** i el nom de l'usuari i contrasenya a emprar amb les directives ***HTTP_User*** i ***HTTP_Passwd*** respectivament) el document rebut per alguna entrada (i, probablement, processat per algun filtre) on se li hagi assignat una etiqueta que concordi amb el valor *Match/Match_Regex* definit en la secció [OUTPUT] en qüestió. Per tal d'interaccionar via HTTPS (el servidor OpenSearch funciona per defecte d'aquesta manera), en https://docs.fluentbit.io/manual/administration/security es poden veure els possible paràmetres TLS a afegir, els quals són bàsicament establir ***Tls*** a ***On*** i establir ***Tls.verify*** a ***Off*** (si el servidor remot té un certificat autosignat) o bé indicar la ruta del certificat arrel d'una CA amb ***Tls.ca_file*** (si el servidor remot té un certificat signat per aquesta CA en qüestió). Una altra directiva útil és ***Current_Time_Index***, la qual fa que si val ***On*** s'utilitzi el temps actual com el temps a guardar en l'índex (en lloc del temps indicat al document original, que és el que passa per defecte, en valer *Off).*

Posem un exemple: suposant que el contingut de la secció *[SERVICE]* és idèntic al que és proporcionat pel fitxer "fluent-bit.conf" que ve de sèrie amb la instal·lació de FluentBit (on, per exemple, la directiva *Flush* val 5), un contingut tal com el següent servirà per enviar (cada 5 segons, doncs) a un servidor OpenSearch local els percentatges d'ús de CPU obtinguts cada segon (per tant, s'enviaran 5 documents de cop cada vegada) i gravar-los en un índex anomenat "fluent_bit":

```
[INPUT]
    Name cpu
    Interval_Sec 1
[OUTPUT]
    Name opensearch
    Host 127.0.0.1
    Port 9200
    Index fluent_bit
    HTTP_User admin
    HTTP_Passwd Un4C0ntraSs3ny4C0mplicad4_
    Tls on
    Tls.verify off
```

NOTA: Un altre plugin de tipus "output" interessant és "**Forward**" (https://docs.fluentbit.io/manual/pipeline/outputs/forward) , el qual implementa un protocol propi de FluentBit i el seu "cosí-germà" FluentD optimitzat per la comunicació entre diferents nodes que executin aquests programes. Si, no obstant, no es vol utilitzar cap protocol de nivell d'aplicació, sempre es pot utilitzar el plugin "**TCP**" (https://docs.fluentbit.io/manual/pipeline/outputs/tcp-and-tls) , el qual permet treballar directament al nivell de transport; en aquest cas, un servidor adient on connectar-hi podria ser *netcat* . Un altre plugin de tipus "output" interessant és **PostgreSQL** (https://docs.fluentbit.io/manual/pipeline/outputs/postgresql) , el qual permet connectar a un altre tipus de servidor d'almacenament, en aquest cas un servidor de bases de dades relacional anomenat PostgreSQL. En tot cas, hi ha molts més "plugins" de tipus "output" (la majoria ja associats a diferents productes privatius). Podeu consultar la llista completa a https://docs.fluentbit.io/manual/pipeline/outputs

NOTA: En el cas de què el destí no respongui (perquè no s'hi pugui arribar per un problema a la xarxa, perquè estigui apagat, etc), FluentBit pot ésser configurat per definir quants reintents d'enviament pot fer, durant quant de temps ho seguirà intentant, etc. A https://docs.fluentbit.io/manual/administration/scheduling-and-retries estan documentades totes aquestes opcions. Ho deixarem com exercici d'ampliació. D'altra banda, en qualsevol "plugin" de tipus "output" que enviï dades a un host remot (és a dir, on intervingui la xarxa) es poden afegir diverses directives específiques de gestió de tràfic a més baix nivell, descrites a https://docs.fluentbit.io/manual/administration/networking

NOTA: En el cas de tenir diversos destins remots equivalents (per tal de proporcionar redundància i realitzar balanceig de càrrega), a https://docs.fluentbit.io/manual/administration/configuring-fluent-bit/classic-mode/upstream-servers s'explica com configurar FluentBit per a què gaudeixi de la possibilitat de contactar amb diversos servidors remots com si fossin un. Ho deixarem com exercici d'ampliació

Fluent-Bit com a comanda de terminal

Existeix la possibilitat, molt útil quan volem fer proves ràpides, d'executar FluentBit com a comanda de terminal en primer pla -o com a contenidor efímer, tal com ja hem vist- (i aturar-lo amb CTRL+C) en lloc de com a servei/contenidor en segon pla. Normalment en aquest casos, emprarem el plugin de tipus "output" *stdout* per veure el resultat, el qual l'indicarem amb el paràmetre pertinent. També indicarem, a més, els paràmetres associats a la resta d'elements necessaris per construir el "pipeline": plugins de tipus "input" (amb les seves corresponents característiques), els filtres (també amb les seves corresponents característiques), etc. De fet, amb aquests paràmetres no caldrà fer servir cap fitxer de configuració (el qual, de fet, només s'utilitza si s'empra el paràmetre *-c*), així que per construir "pipelines" ràpides aquesta manera de executar FluentBit és molt interessant. Els paràmetres que ens seran útils en aquest context són:

-i nomPluginInput : Obvi
-F nomFiltre : Obvi
-o nomPluginOutput : Obvi
-p directiva=valor : Aquest paràmetre cal repetir-ho tantes vegades com directives diferents volguem indicar. Segons si apareix darrera del paràmetre *-i*, *-F* o *-o*, s'entendrà que la directiva en qüestió correspon a un plugin "input", a un filtre o a un plugin "output", respectivament
-t valor : Equivalent a *-p Tag=valor*. Si no s'indica, els documents entrats no tindran tag
-m valor : Equivalent a *-p Match=valor*. Si no s'indica, és el mateix que escriure *-m "*"*
-f nº : Equivalent a la directiva *Flush* de la secció [SERVICE]
-R /ruta/fitxer.conf : Equivalent a la directiva *Parsers_file* de la secció [SERVICE]
-h : Mostra l'ajuda de la comanda

NOTA: Recordem d'aturar abans el servei FluentBit si volem executar FluentBit en primer pla, per evitar interferències

Per exemple, la següent comanda envia cada segon a la sortida estàndard totes les noves línies que hagin aparegut al final dels fitxers *.log ubicats a "/var/log/mariadb" des de la darrera execució de FluentBit (aquesta comanda cal executar-la com administrador per poder llegir l'arxiu "*.log"):

sudo fluent-bit -f 1 -i tail -p path=/var/log/mariadb/.log -p db=/opt/logsmaria.db -o stdout -m "*"*

La següent comanda, variant de l'anterior, només envia les línies que continguin la paraula "error":

sudo fluent-bit -f 1 -i tail -p path=/var/log/mariadb/.log -p db=/opt/logsmaria.db* ***-F grep -p "regex=log error"*** *-o stdout*

La següent comanda envia cada segon a la pantalla les noves línies rebudes al Journal generades per la comanda *sudo*:

fluent-bit -f 1 -i systemd -p systemd_filter="_COMM=sudo" -p db=~/logssudo.db -o stdout

Igualment, si volem obtenir (per defecte és cada 5 segons) les línies rebudes al Journal per part del sistema Audit corresponents a l'event USER_START, podem fer:

fluent-bit -i systemd -p systemd_filter="_AUDIT_TYPE_NAME=USER_START" -p db=~/logsaudit.db -o stdout

Per provar el plugin d'entrada "stdin" podem escriure el següent script, que anomenarem "test.sh" (i al qual li assignarem permisos d'execució), i tot seguit executar la comanda: *./test.sh | fluent-bit -i stdin -o stdout*

```
#!/bin/bash
while [[ true ]]; do
      echo -n '{"clau": "un valor"}' ;   sleep 1
done
```

D'altra banda, un exemple d´us del "plugin" d'entrada "head" pot ser el següent, on cada 2 segons es llegeix la 1ª línia de l'arxiu */proc/uptime* (però es mostraran a pantalla els resultats obtinguts cada 5, *-f 5*):

fluent-bit -i head -p file=/proc/uptime -p lines=1 -p interval_sec=2 -o stdout

Si volem obtenir una línia ubicada enmig d'algun fitxer, haurem d'obtenir primer amb el plugin "head" totes les línies des del començament fins aquesta línia en qüestió tenint en compte d'assignar a la seva directiva *Split_line* el valor *true* per tal de crear al document sortint una parella *clau<->valor* diferent per cadascuna de les línies obtingudes amb els noms de clau "line0", "line1", etc...(si no, totes les línies obtingudes estaran col·lapsades en un únic registre) i, tot seguit, eliminar tots els camps "lineX" que no ens interessin (per quedar-nos al corresponent a la línia desitjada) mitjançant el filtre "Record_modifier" i la seva directiva *Allowlist_key* És a dir, si volem quedar-nos amb l'evolució cada 5 segons de la freqüència d'ús de la CPU (línia nº7 del fitxer "/proc/cpuinfo"), haurem de fer el següent (el paràmetre *-m* en filtres és obligatori):

fluent-bit -i head -p file=/proc/cpuinfo -p lines=7 -p interval_sec=5 -p split_line=true -F record_modifier
-p allowlist_key=line6 -m "" -o stdout*

Un altre exemple de manipulacions de camps per part del filtre "Record_modifier" en els documents rebuts per una determinada entrada (en aquest cas, la del plugin "mem") és el següent:

fluent-bit -i mem -F record_modifier -p "record=hostname ${HOSTNAME}" -p remove_key=Swap. -m "*" -o stdout*

D'altra banda, com exemple d'entrada provinent de la xarxa podem provar la següent comanda juntament amb l'execució, des d'una màquina remota, de la comanda *ncat ip.serv.fluent.bit 5555* (es pot comprovar, a la sortida de FluentBit, com aquest només acceptarà, dels missatges rebuts, només els que estiguin en format JSON):

fluent-bit -i tcp -t unaetiqueta -p host=0.0.0.0 -p port=5555 -o stdout

En comptes de servidor TCP, tal com s'ha comentat en apartats anteriors, es pot posar en marxa un servidor HTTP amb una comanda similar a la següent (en aquest cas, el client remot podria ser la comanda *curl* i caldria que enviés els missatges d'aquesta manera: *curl -d '{"unaclau":"unvalor"}' -X POST -H "Content-Type: application/json" http://ip.serv.fluent.bit:5555/unaetiqueta*):

fluent-bit -i http -t unaetiqueta -p host=0.0.0.0 -p port=5555 -o stdout

En el cantó contrari, podem volem enviar dades ja rebudes i processades per FluentBit a un destí remot per a que s'hi emmagatzemin. Molts cops aquest servidor remot serà un de tipus OpenSearch; en aquest cas, haurem de fer servir una comanda similar a la següent (la qual, a més envia la sortida a la pantalla)...

fluent-bit -i cpu -t cpu -o opensearch -p host=ip.serv.Open.Search -p port=9200 -p index=unindex -p http_user=admin
-p http_passwd=Un4C0ntraSs3ny4C0mplicad4_ -p tls=on -p tls.verify=off -m "" -o stdout -m cpu*

NOTA: Per provar si l'enviament de dades anterior s'ha realitzat correctament, podem realitzar una consulta simple a l'índex en qüestió (el qual, si no existís en executar la comanda anterior, es crearia en aquell moment) per exemple,executant aquesta comanda *curl -k -u "admin:Un4C0ntraSs3ny4C0mplicad4_" https://ip.serv.Open.Search:9200/_cat/indices* (en el proper PDF estudiarem en profunditat el seu significat)

EXERCICIS:

1.-a) Arrenca una màquina virtual (Ubuntu o Fedora, tant és, però que sigui de tipus "Server" per a què no consumeixi tanta RAM) que tingui la seva tarja de xarxa en mode "adaptador pont" i mínim 4GB de RAM i segueix els passos indicats a la teoria per instal·lar-hi manualment el FluentBit. No cal que creis de moment cap fitxer de configuració ni cap fitxer ".service"

b) Digues què fan les següents comandes, paràmetre a paràmetre (i prova-les per comprovar-ho):

NOTA: En el que féssim servir contenidors efímers, caldria substituir el nom de la comanda indicada ("*fluent-bit*") per aquesta altra: "*podman run --rm -ti cr.fluentbit.io/fluent/fluent-bit:latest*" No els farem servir perquè a l'hora de treballar amb fitxers (d'entrada o de sortida) és farragós haver d'estar mapejant-los tota l'estona al sistema de fitxers amfitrió, a més que tots els apartats relacionats amb l'ús del Systemd del sistema com a entrada no funcionarien

fluent-bit -f 1 -i random -p interval_sec=2 -p samples=8 -o stdout <-Fixa´t cada quants segons apareixen els docs
fluent-bit -f 1 -i random -p interval_sec=2 -p samples=8 -o stdout -p format=json
fluent-bit -f 1 -i random -p interval_sec=2 -p samples=8 -o stdout -p format=json_lines
fluent-bit -f 1 -i random -p interval_sec=2 -p samples=8 -o stdout -p format=json_stream
fluent-bit -f 2 -i dummy -p dummy='{"hola":"adeu"}' -p rate=3 -p samples=8 -o stdout <-Fixa't en quants docs/s

c) Digues què fan les següents comandes, paràmetre a paràmetre (i prova-les per comprovar-ho):

fluent-bit -i netif -t eth -p interface=enp0s3 -p interval_sec=1
-i netif -t lo -p interface=lo -p interval_sec=2
-o file -m eth -p path=$HOME -p file=eth.txt -p format=plain*
-o stdout -m lo -p format=msgpack*

fluent-bit -i cpu -t cpu
-i mem -t mem -F record_modifier -p "record=hostname ${HOSTNAME}" -p remove_key=swap. -m mem*
-o file -m cpu -p path=$HOME -p file=cpu.txt -p format=csv
-o stdout -m mem

d) Digues què fan les següents comandes, paràmetre a paràmetre (i prova-les per comprovar-ho):

fluent-bit -i head -p file=/proc/uptime -p lines=1 -p interval_sec=2 -o stdout -p format=json_lines

fluent-bit -i head -p file=/proc/cpuinfo -p lines=7 -p interval_sec=2 -p split_line=true
-F record_modifier -p allowlist_key=line6 -m "" -o stdout*

e) Digues què fan la següent comanda, paràmetre a paràmetre (fixa't en l'ús del filtre "record_modifier")...

fluent-bit -f 1 -i systemd -p systemd_filter="_COMM=sudo" -p db=$HOME/logssudo.db
-F record_modifier -p allowlist_key=syslog_timestamp -p allowlist_key=priority -p allowlist_key=_cmdline
-p allowlist_key=_uid -p allowlist_key=_hostname -p allowlist_key=message -m ""*
-o stdout -p format=json_lines

...i prova-la per comprovar-ho executant *sudo fdisk -l* en un altre terminal (pots posar la contrasenya bé o malament, com vulguis; simplement veuràs missatges diferents a la consola de FluentBit).

eII) Digues què fan la següent comanda (similar a l'anterior però amb l'afegit del filtre "grep"), paràmetre a paràmetre...

fluent-bit -f 1 -i systemd -p systemd_filter="_COMM=sudo" -p db=$HOME/logssudo.db
-F record_modifier -p allowlist_key=message -m ""*
-F grep -p "regex=MESSAGE incorrect.password.attempts" -m ""*
-o stdout -p format=json_lines

...i prova-la per comprovar-ho executant *sudo fdisk -l* en un altre terminal (posant tant la contrasenya bé com malament per veure les diferències en els missatges mostrats a la consola de FluentBit).

> NOTA: Recorda que a la web https://rubular.com pots comprovar si una determinada expressió regular concorda amb una determinada línia. Aquesta web és molt pràctica per fer proves amb diferents expressions regulars fins trobar l'expressió regular desitjada

f) Digues què fan la següent comanda, paràmetre a paràmetre...

fluent-bit -i systemd -p systemd_filter="_AUDIT_TYPE_NAME=USER_START" -p db=$HOME/logsaudit.db
-F record_modifier -p allowlist_key=audit_field_exe -p allowlist_key=audit_field_res
-p allowlist_key=message -m ""*
-o stdout -p format=json_lines

...i prova-la per comprovar-ho executant *su -l usuari* en un altre terminal i també iniciant sessió en un terminal virtual (en tots dos casos pots posar la contrasenya bé o malament, com vulguis; simplement veuràs missatges diferents a la consola de FluentBit).

fII) ¿Què es veurà si a la comanda anterior afegim el següent filtre (després del filtre "record_modifier" ja existent) i tornem a provar d'iniciar sessió executant *su -l usuari* o bé en un terminal virtual?

-F grep -p "regex=AUDIT_FIELD_EXE /usr/sbin/sshd" -m ""*

¿I, si en lloc del filtre anterior, afegim aquest altre (i tornem a provar d'executar *su -l usuari* o bé loguejar-nos en un terminal virtual)? Prova tant el cas en què escrius correctament la contrasenya com el cas en que no.

-F grep -p "regex=AUDIT_FIELD_RES failed" -m ""*

Ara combina els dos filtres anteriors (tal com es mostra a la comanda següent) per tal d'obtenir només els missatges corresponents només als inicis de sessió fallits per SSH . Comprova que és així executant *su -l usuari* i escrivint la contrasenya bé i malament i, d'altra banda, accedint des d'un client SSH i escrivint també la contrasenya bé i malament:

fluent-bit -i systemd -p systemd_filter="_AUDIT_TYPE_NAME=USER_START" -p db=$HOME/logsaudit.db
-F record_modifier -p allowlist_key=audit_field_exe -p allowlist_key=audit_field_res
-p allowlist_key=message -m ""*
-F grep -p "regex=AUDIT_FIELD_EXE /usr/sbin/sshd" -m ""*
-F grep -p "regex=AUDIT_FIELD_RES failed" -m ""*
-o stdout -p format=json_lines

g) Digues què fan la següent comanda, paràmetre a paràmetre...

fluent-bit -i tcp -p host=127.0.0.1 -p port=5555 -o stdout

...i prova-la per comprovar-ho executant (en un altre terminal): *nc 127.0.0.1 5555* i escrivint-hi qualsevol document JSON a la consola del netcat (com per exemple: *{"hola":"adeu"}*)

gII) Digues què fan la següent comanda, paràmetre a paràmetre...

fluent-bit -i http -p host=127.0.0.1 -p port=9999 -o stdout

...i prova-la per comprovar-ho executant (en un altre terminal): *curl -X POST -d '{"hola":"adeu"}' -H "Content-Type:application/json" http://127.0.0.1:9999/unaetiqueta*

> NOTA: FluentBit, a diferència d'altres programes similars, no disposa (encara) de cap plugin de tipus "HTTP poller" (o, el que és el mateix, d'un client HTTP integrat que obtingui dades d'entrada demanades a servidors HTTP externs -normalment a intervals periòdics-)

h) Digues què fa la següent comanda, paràmetre a paràmetre (fixa't en l'ús del filtre "modify").....

fluent-bit -i systemd -t fireFedora -p systemd_filter="UNIT=firewalld.service" -p db=$HOME/logsfirewallF.db
-i systemd -t fireUbuntu -p systemd_filter="UNIT=ufw.service" -p db=$HOME/logsfirewallU.db
-F modify -p "set=distro Fedora" -m fireFedora -F modify -p "set=distro Ubuntu" -m fireUbuntu
-F record_modifier -p allowlist_key=job_type -p allowlist_key=message -p allowlist_key=distro -m ""*
-o http -p host=https://yyyyy.x.pipedream.net -p port=443 -p tls=on -p tls.verify=off -p format=json -m ""*
-o stdout -p format=json_lines -m fireFedora

> NOTA: El "yyyyy" marcat en lila anterior representa una cadena aleatòria que s'haurà generat automàticament en haver anat un cop prèviament a https://public.requestbin.com/r

...i per comprovar-ho reinicia (des d'un altre terminal) el servei "firewalld" (a Fedora) o "ufw" (a Ubuntu) i tot seguit vés amb un navegador a la URL *https://yyyyy.x.pipedream.net* ¿Què veus a la columna mostrada a l'esquerra de la finestra (que representa la llista de dades obtingudes provinents de les peticions rebudes)? ¿I a la consola de FluentBit?

2.-a) A la màquina virtual de treball de l'exercici anterior instal·la ara el paquet "apache2" (a Ubuntu) o "httpd" (a Fedora). Tot seguit, crea un arxiu anomenat "index.html" sota la carpeta "/var/www/html" amb el següent contingut: *<html><body>Hola</body></html>* i finalment posa en marxa el servei.

aII) Realitza unes quantes peticions contra aquest servidor HTTP mitjançant "curl" (les pots fer des de la màquina real si vols); unes quantes han de retornar una resposta amb codi 200 i unes altres han de retornar una resposta amb codi 404 (això ho pots fer executant, respectivament, les comandes *curl http://ip.serv.web* i *curl http://ip.serv.web/asdf*)

El registre de les peticions anteriors es guarda a l'arxiu "/var/log/httpd/access_log" (a Fedora) o "/var/log/apache2/access.log" (a Ubuntu). el qual està format, en la seva configuració per defecte, per línies amb els següents camps (d'esquerra a dreta, i separats per un espai):

* IP del client que ha fet la petició
* Un guió (correspon a una dada actualment obsoleta)
* El nom d'usuari (en el cas que la petició HTTP fos autenticada; si no, apareixerà un altre guió)
* La data quan s'ha fet la petició
* La petició en sí (en el format "*<METODE> <RUTAiQUERYSTRING> <VERSIO>*")
* El codi numèric de la resposta
* La mida de la resposta (sense compta capçaleres) en bytes
* El valor de la capçalera de client "Referer" (si no es troba disponible, apareixerà un altre guió)
* El valor de la capçalera de client "User-Agent".

aIII) Confirma l'explicat al paràgraf anterior observant el contingut del fitxer de registre de peticions del teu servidor Apache

b) Executa la següent comanda (canvia la ruta de l'arxiu monitoritzat si calgués). ¿Què veus a la pantalla en tornar a fer peticions (amb resposta 200 i també 404) amb *curl*?

sudo fluent-bit -f 1 -i tail -p path=/var/log/httpd/access_log -p db=/opt/logsapache.db -o stdout

NOTA: Cal executar aquesta comanda amb *sudo* per poder tenir permisos de lectura sobre el fitxer monitoritzat

bII) ¿I si executes aquesta altra comanda (simplement hem afegit un filtre a la comanda anterior), què veus ara a la pantalla en tornar a fer peticions amb resposta 200 i també 404?

sudo fluent-bit -f 1 -i tail -p path=/var/log/httpd/access_log -p db=/opt/logsapache.db
-F grep -p "regex=log 404" -m "" -o stdout*

Les comandes FluentBit anteriors tenen un inconvenient, que és que treballen directament amb cadenes desestructurades, cosa que fa que sigui difícil i fràgil processar, cercar, etc la informació allà continguda. És molt adient que les dades d'entrada es puguin estructurar de forma estàndard i fixa tant bon punt travessin un "plugin" Input, i per això farem servir el filtre anomenat "Parser". Concretament, al següent exercici aconseguirem passar de tenir un camp anomenat "log" amb un contingut com aquest...:

192.168.2.20 - - [28/Jul/2021:10:27:10 -0300] "GET /cgi-bin/try/ HTTP/1.0" 200 3395

...a una estructura de camps tal com aquesta:

```
{
 "host":    "192.168.2.20",
 "user":    "-",
 "method":  "GET",
 "path":    "/cgi-bin/try/",
 "code":    "200",
 "size":    "3395",
 "referer": "",
 "agent":   ""
}
```

En aquest sentit, tenim la sort que FluentBit ja disposa d'un "parser" predefinit que entén el format dels logs del servidor Apache i, per tant, només haurem de fer-lo servir i ja està. Concretament, si posem en marxa FluentBit com a servei utilitzant l'arxiu de configuració "fluent-bit.conf" predeterminat, gràcies a la directiva *Parsers_file* allà present, el nom concret (en aquest cas, "apache2") i la definició corresponent (mitjançant una expressió regular) dels "parsers" disponibles per defecte en una instal·lació de FluentBit s'anirà a buscar en l'arxiu "parsers.conf", ubicat (si hem instal·lat FluentBit a partir de les seves fonts), sota la carpeta "/usr/local/etc/fluent-bit". Si posem en marxa FluentBit com una comanda de terminal, en canvi, caldrà indicar explícitament la ruta de l'arxiu de definició dels "parsers" a utilitzar mitjançant el paràmetre *-R*, tal com ja hem dit a la teoria i com farem al següent apartat

c) Executa la següent comanda (simplement hem substituit un filtre per un altre). ¿Què veus ara a la pantalla en tornar a fer peticions (amb resposta 200 i també 404)?

sudo fluent-bit -f 1 -R /usr/local/etc/fluent-bit/parsers.conf -i tail -p path=/var/log/httpd/access_log
-p db=/opt/logsapache.db -F parser -p parser=apache2 -p key_name=log -m "" -o stdout*

És possible (tot i que no és habitual), canviar el format dels logs generats pel servidor Apache2 per a què s'escriguin en format JSON. Al següent apartat farem aquest canvi per tal de mostrar com recolliria en aquest cas Fluent-Bit aquest tipus de missatges, ja que són molt comuns en altres tipus de logs.

d) Primer canvia la configuració del servidor Apache2: concretament, substitueix la línia que comença per "LogFormat" i que acaba per "combined" de l'arxiu "/etc/httpd/conf/httpd.conf" (a Fedora) o de l'arxiu "/etc/apache2/sites-available/000-default.conf" (a Ubuntu) per aquesta, i tot seguit reinicia el servei...:

LogFormat "{ \"time\":\"%{%s}t\", \"host\":\"%h\", \"method\":\"%m\", \"path\":\"%U\", \"querystring\":\"%q\", \"code\":\"%>s\", \"size\":\"%b\", \"referer\":\"%{Referer}i\", \"userAgent\":\"%{User-Agent}i\"}" combined

> NOTA: Teniu una referència del significat de cadascun dels símbols indicats a la línia anterior en la documentació de l'Apache que tracta sobre la personalització dels seus logs: https://httpd.apache.org/docs/2.4/mod/mod_log_config.html
> NOTA: Fixeu-vos que ha sigut necessari escapar totes i cadascuna de les cometes que envolten els noms de les claus JSON i els seus respectius valors.

...i comprova com, en fer alguna petició al servidor Apache2 el nou contingut del fitxer de registre de peticions apareix en format JSON contenint els camps indicats a la directiva anterior ("time", "host", "method", "path", "querystring", "code", "size", "referer" i "userAgent").

dII) Executa la següent comanda, la qual, com pots veure, no fa servir cap "parser" sobre el valor del camp "log" (que, recordem, representa el contingut de cada línia nova apareguda al final de l'arxiu de registre monitoritzat pel "plugin" *tail*). A pantalla hauràs de veure, doncs, que, tot i que ara aquest contingut, com hem vist, està en format JSON, FluentBit no l'interpreta com a tal (i per tant, no identifica els possibles camps individuals). Comprova-ho.

sudo fluent-bit -f 1 -i tail -p path=/var/log/httpd/access_log -p db=/opt/logsapache.db -o stdout

dIII) Executa ara la següent comanda, la qual fa servir el parser anomenat "json" (ja predefinit dins de l'arxiu "parsers.conf" del Fluent-Bit) sobre el mateix camp "log" que hem fet servir a l'apartat anterior. ¿Què veus ara a la pantalla en tornar a fer peticions i per què?

sudo fluent-bit -f 1 -R /usr/local/etc/fluent-bit/parsers.conf -i tail -p path=/var/log/httpd/access_log
-p db=/opt/logsapache.db ***-F parser -p parser=json -p key_name=log*** *-m '*' -o stdout*

> NOTA: És important que el camp guardat a l'arxiu de registre de l'Apache2 que conté la dada temporal que indica quan es va rebre la petició s'anomeni "time" (tal com hem fet) i que aquest camp tingui el format de "Unix time" (també anomenat "Epoch time"), tal com hem fet també indicant el modificador "%s" entre claus davant de la dada %t. Això és perquè per defecte, el "parser" JSON del Fluent-bit reconeix com a camp de referència temporal el que tingui aquest nom i el seu valor estigui en aquest format. Si això no pogués ser personalitzat, hauríem de crear llavors un "parser" personalitzat de tipus "Json" igualment però on hauríem d'indicar el nom del camp temporal i el format de temps esperat (el qual està documentat a *man strptime*) amb les directives ***Time_Key*** i ***Time_Format*** respectivament, tal com ja s'ha explicat a la teoria (per exemple, així):
>
> ```
> [PARSER]
>    Name elmeuparserjsonpersonalitzat
>    Format json
>    Time_Key nomCampTemps
>    Time_Format %Y-%m-%dT%TZ
> ```

No obstant, en la majoria de vegades, els formats dels diferents logs que vulguem monitoritzar no tindran un format que FluentBit el pugui reconèixer per defecte, així que haurem de crear-nos el nostre propi "parser" a mà (normalment fent ús d'expressions regulars) dins del fitxer "parsers.conf" mencionat. Com fer això és el que veurem al proper exercici

3.-a) Crea (a la màquina virtual) un arxiu dins de la teva carpeta personal anomenat "custom-parsers.conf" amb el següent contingut (¡estudia l'expressió regular per tal d'entendre-la perfectament, i també el format de data (*man strptime*)!):

```
[PARSER]
   Name pepito
   Format regex
   Regex ^\[(?<data>[^ \]]+)\] (?<paraula>[^ ]+) (?<sencer>[^ ]+) (?<dec>[^ ]+) (?<bool>.+)$
   Time_Key data
   Time_Format %Y-%m-%dT%TZ
```

NOTA1: L'expressió *(?<nomCamp>...)* -on els punts suspensius representen qualsevol expressió regular que s'hi vulgui indicar- fa que "s'extregui" la cadena concordant amb l'expressió regular donada i s'assigni com a valor d'un nou camp anomenat "nomCamp".
NOTA2: En el cas de voler "extreure" una cadena concordant però no voler assignar-la a cap nou camp (això pot ser interessant per anar trossejant les dades originals sense haver de guardar cada tros obtingut, que pot no tenir rellevància, en un camp nou), l'expressió a usar ha de ser *(?:...)*
NOTA: Entre cada grup marcat pels parèntesis hi ha d'haver un espai en blanc per tal de que es reconeguin correctament
NOTA: No cal que tots els "parsers" es defineixin a partir d'expressions regulars (tot i que és el més habitual amb diferència): altres "formats" possibles per un "parser" personalitzat són "json" o "logfmt", tal com ja s'ha dit a la teoria

b) Executa la següent comanda. ¿Què es guarda a l'arxiu "out.txt"? ¿Per què?

```
fluent-bit -R "$HOME/custom-parsers.conf" -i dummy -p dummy='{"linia":"[2022-1-11T23:35:05Z] hola 123 3.5 true"}'
        -F parser -p parser=pepito -p key_name=linia -m "*" -o file -p path=$HOME -p file=out.txt -p format=plain
```

c) ¿Què passa ara si comentes al fitxer de definició del "parser" *pepito*, les directives *Time_Key* i *Time_Format* i tornes a provar la comanda anterior? ¿Per què?

NOTA: La següent expressió regular és la que defineix, dins del fitxer "parsers.conf" oficial, el comportament del "parser" anomenat "apache2" utilitzat a l'exercici anterior. Sabent ja el que hem après, no hauria de ser difícil interpretar-la.
^(?<host>[^]) [^]* (?<user>[^]*) \[(?<time>[^\]]*)\] "(?<method>\S+)(?: +(?<path>[^]*) +\S*)?" (?<code>[^]*) (?<size>[^]*)(?: "(?<referer>[^\"]*)" "(?<agent>.*)")?$*

3II.-a) Afegeix al fitxer "custom-parsers.conf", creat a l'exercici anterior, un nou "parser" tal com aquest:

```
[PARSER]
   Name pepito2
   Format logfmt
```

b) Executa la següent comanda. ¿Quina informació obtens a pantalla i per què té aquest format? (pots forçar la generació de més registres si inicies sessió al sistema d'alguna forma: *gdm, login, su, ssh...*)

```
fluent-bit -R custom-parsers.conf -i systemd -p systemd_filter="_AUDIT_TYPE_NAME=USER_START"
      -F parser -p parser=pepito2 -p key_name=MESSAGE -m "*"
      -F record_modifier -p allowlist_key=exe -p allowlist_key=res -p allowlist_key=auid -p allowlist_key=addr -m "*"
      -o stdout -p format=json_lines
```

c) Executa la següent comanda. ¿Quina informació obtens a pantalla i per què té aquest format? (pots forçar la generació de més registres si reinicies algun servei qualsevol vàries vegades)

```
fluent-bit -R custom-parsers.conf -i systemd -p systemd_filter="_AUDIT_TYPE_NAME=SERVICE_START"
       -p systemd_filter="_AUDIT_TYPE_NAME=SERVICE_STOP"
       -F parser -p parser=pepito2 -p key_name=MESSAGE -m "*" -p reserve_data=true
      -F record_modifier -p allowlist_key=_AUDIT_TYPE_NAME -p allowlist_key=AUDIT_FIELD_UNIT
      -p allowlist_key=res  -m "*" -o stdout -p format=json_lines
```

El servidors SSH també generen diversos missatges de registre interessants (en aquest cas es guarden directament al Journal). A continuació estudiarem diverses expressions regulars que podrem fer servir per detectar determinats patrons de missatges relacionats amb l'intent d'inicis de sessió SSH.

3III.-a) Vés a https://rubular.com i comprova si (i raona per què) el patró següent... :

^(?<month>\w+) (?<month_day>\d+) (?<hour>[^]+) (?:[^]+) sshd\[(?:\d+)\]: Disconnected from (?<client_ip>[^]+) port (?:\d+)$

... serveix per reconèixer línies com la següent, i en cas que sí, digues quins són els camps "extrets" i per què

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

NOTA: Pots ajudar-te consultant la referència d'exemples https://digitalfortress.tech/tips/top-15-commonly-used-regex (on també hi ha un breu resum del significat dels símbols més comuns en expressions regulars, com ara \w, \d, \s, etc

b) Si tinguéssim una expressió regular com la següent...

^(?<month>\w+) [ ] (?<month_day>\d+) (?<hour>[^]+) (?:[^]+) sshd\[(?:\d+)\]: (?<status>[^]+) password for (?<user>[^]+) from (?<client_ip>[^]+) port (?:[^]+) ssh2$*

...digues, fent servir de nou https://www.rubular.com, si serviria per registrar missatges com els següents i per què; en cas que sí, digues a més quins serien els camps "extrets" (amb els seus possibles valors) i per què

Dec 12 13:15:23 localhost sshd[1387]: Failed password for testuser from 192.168.0.102 port 60004 ssh2
Dec 12 13:08:30 localhost sshd[1338]: Accepted password for testuser from 192.168.0.102 port 59958 ssh2

Squid (http://www.squid-cache.org) és un servidor proxy-catxé HTTP/S. Aquest tipus de programes serveixen per reduir l'ús de l'ample de banda utilitzat pels ordinadors d'una LAN gràcies a funcionar com un node central per on passen (i on es guarden a la memòria cau) totes les seves peticions; també poden fer-se servir com a "Gran Hermano" i, mitjançant llistes negres, punt central de censura. El format dels logs de l'Squid on es registren els intents d'accés a les diferents webs d'Internet per part dels ordinadors d'una LAN es troba documentat a https://wiki.squid-cache.org/Features/LogFormat però podeu trobar una representació gràfica a continuació...

i un parell de logs de mostra tot seguit:

1524206424.034 19395 207.96.0.0 TCP_MISS/304 15363 GET http://elastic.co/android-chrome-192x192.gif - DIRECT/10.0.5.120 -
1524206424.145 106 207.96.0.0 TCP_HIT/200 68247 GET http://elastic.co/guide/en/logstash/current/images/logstash.gif - NONE/- image/gif

3IV.-Després de llegir la teoria anterior digues si un filtre de tipus "parser" basat en la següent expressió regular (on el símbol "\" s'usa per escapar)....

^(?<timestamp>[^]+) [ ]+ (?<duration>[^]+) (?<client_address>[^]+) (?<cache_result>[^]+)\/(?<status_code>[^]+) (?<bytes>[^]+) (?<request_method>[^]+) (?<url>[^]+) (?<user>[^]+) (?<hierarchy_code>[^]+)\/(?<server>[^]+) (?<content_type>.+)$

... ens permetria generar un document JSON amb els noms dels camps que es detallen a continuació (el valors concrets mostrats són irrellevants), i si no, digues per què

{
"timestamp" => "1524206424.034", "duration" => "19395", "client_address" => "207.96.0.0",
"cache_result" => "TCP_MISS", "status_code" => "304", "bytes" => "15363", "request_method" => "GET",
"url" => "http://elastic.co/android-chrome-192x192.gif", "user" => "", "hierarchy_code" => "DIRECT"
"server" => "10.0.5.120", "content_type" => "-",
}

NOTA: Pots utilitzar https://rubular.com per ajudar-te

A continuació repetirem alguns apartats del primer exercici (concretament, el e) i el eII)) però ara farem servir, per indicar les entrades, filtres i sortides, un arxiu de configuració enlloc de paràmetres individuals de la comanda. Aquesta forma és la més habitual, de fet, a l'hora d'executar Fluent-Bit com a dimoni, tal com es pot veure amb *systemctl cat fluent-bit*

4.-a) Crea un arxiu anomenat "prova.conf" amb el següent contingut i tot seguit executa la comanda *fluent-bit -c prova.conf*. A continuació, en un altre terminal prova d'executar per exemple la comanda *sudo fdisk -l* (pots posar la contrasenya bé o malament, com vulguis). ¿Quins missatges veus a la consola de FluentBit?

```
[SERVICE]
  Flush     1
  Daemon off

[INPUT]
  Name systemd
  Systemd_filter   _COMM=sudo
  Db   ${HOME}/logssudo.db

[FILTER]
  Name    record_modifier
  Match   *
  Allowlist_key   syslog_timestamp
  Allowlist_key   priority
  Allowlist_key   _cmdline
  Allowlist_key   _uid
  Allowlist_key   _hostname
  Allowlist_key   message

[OUTPUT]
  Name   stdout
  Match  *
  Format json_lines
```

> NOTA: Recordeu que en el cas d'executar Fluent-Bit (o qualsevol altre programa) com a dimoni, la sortida estàndard es redireccionarà al Journal, així que per veure els missatges emesos cal utilitzar la comanda *journalctl*

aII) Modifica l'arxiu "prova.conf" anterior per a què ara tingui el següent contingut i tot seguit torna a executar la comanda *fluent-bit -c prova.conf*. A continuació, en un altre terminal prova d'executar de nou la comanda *sudo fdisk -l* (posa ara la contrasenya malament tres cops). ¿Quins missatges veus ara exclusivament a la consola de FluentBit?

```
[SERVICE]
  Flush     1
  Daemon off

[INPUT]
  Name systemd
  Systemd_filter   _COMM=sudo
  Db   ${HOME}/logssudo.db

[FILTER]
  Name    record_modifier
  Match   *
  Allowlist_key   message

[FILTER]
  Name    grep
  Match   *
  Regex   MESSAGE incorrect.password.attempts

[OUTPUT]
  Name   stdout
  Match  *
```