Monitoratge amb TSDBs

INTRODUCCIÓ:

Time Series Data

Time series data can be defined as data points indexed by their temporal order, where the distance between two data points may or may not be equal. If the frequency at which data points are taken is constant (e.g., sampling the data every 10 ms) then the series is called a discrete time data series.

In computer systems, all user data can potentially be represented as time series data, as all stored information has a time component that can provide different metrics in different scenarios. For example, Twitter, Facebook, and LinkedIn have data on the user's registration date, as well as the dates and times at which various actions were performed (tweet or article posted, activity liked, etc.). Other examples are data collected from physical sensors (temperature, humity, etc) or from performance's computer monitoring tools (RAM use, CPU use, disk space use, network bandwith use, etc) and/or specific service monitoring tools (apache2 status, mysql status, etc).

Having data arrive at a higher frequency can create challenges, including having to handle a greater number of write requests per second and needing to store all the data. One sensor, with a sampling frequency of 30 requests per second and a payload of 1KB, can generate 86MB of information each day, meaning 100 sensors would create a data load of 8GB per day. Querying and aggregating such a large amount of data to extract useful information is another issue to be considered.

Eines

No todos los programas que tratan datos de tipo "series de tiempo" son iguales: algunos tan solo se encargan de recopilar los datos en sí; otros se encargan de recibir por la red esos datos para recopilarlos y/o guardarlos de forma permanente en una base de datos específica (las llamadas Time Series BD) y otros se encargan de ofrecer esos datos gráficamente en forma de "dashboard" editable con gráficas y colorines.

Históricamente, las primeras TSBDs estaban mayormente basadas en un formato de datos llamado RRD, manejado principalmente por una herramienta central llamada **RRDTool** (https://oss.oetiker.ch/rrdtool) , la cual integra tanto la base de datos en sí (de tipo "Round Robin", de ahí el nombre) como diversos comandos de terminal para su manejo e incluso también un graficador, pudiéndose utilizar diversos recolectores, como por ejemplo:

Collectd (https://collectd.org)
Collectl (http://collectl.sourceforge.net)
MRTG (http://oss.oetiker.ch/mrtg) -Eina més específica per obtenir dades de tipus SNMP-
SmokePing (http://oss.oetiker.ch/smokeping) -Eina més específica per obtenir dades de latència de xarxa-

En el caso de no querer utilizar el graficador que RRDTool incorpora, también se podrían utilizar otros graficadores especializados externos, como:

GnuPlot (http://www.gnuplot.info)
Drraw (http://web.taranis.org/drraw)

O bien optar por soluciones "todo-en-uno", las cuales incorporan un recolector de datos y un graficador propio convenientemente configurados con el motor RRDTool para funcionar de forma unificada. Ejemplos son:

Cacti (http://www.cacti.net)
Munin (http://munin-monitoring.org)
Monit (https://mmonit.com/monit)
Monitorix (http://www.monitorix.org)

Por otro lado, existe una segunda generación de herramientas de monitoraje que van más allá e incorporan todo un ecosistema de plugins que multiplican la versatilidad y funcionalidad de las herramientas RDDTool iniciales. En este sentido se tratan de completos "packs integrales" con multitud de opciones, tanto a nivel de recolección de datos como de visualización (y también de alertas/notificaciones, disparadas al detectar valores fuera de los umbrales previamente definidos para determinados datos). El "abanderado" de esta segunda generación es **Nagios** (https://www.nagios.org/downloads/nagios-core) pero hay muchos más:

Zennoss (http://www.zenoss.org)
Icinga (https://www.icinga.com/download)
Zabbix (http://www.zabbix.com/download)
OpenNMS (https://www.opennms.org/en/install)
LibreNMS (http://www.librenms.org)
PandoraFMS (https://pandorafms.org/en/features/free-download-monitoring-software)
Observium (http://www.observium.org/docs)
Centreon (https://www.centreon.com)
Bosun (http://bosun.org). Especializado en alertas

Se pueden consultar más en https://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems

No obstante, nosotros estudiaremos "la tercera generación", en la cual se vuelve a la filosofía de "divide y vencerás" (es decir, utilizar una herramienta especializada independiente para cada cosa -recolectar, agregar/almacenar, visualizar- en vez de una macro aplicación "todo-en-una", teniendo entonces que preocuparse, eso sí, de configurarlas para que se comuniquen entre sí correctamente pero ganando a cambio versatilidad y "desacople"). En este sentido, volvemos a encontrarnos con recolectores (que, insistimos, no proporcionan persistencia ya que para ello deberán conectarse a una TSDB)...:

NetData (https://my-netdata.io)
Performance Co-Pilot (http://pcp.io) Incluye un gestor de TSDB propio básico y también un alertador
Sensu (https://sensuapp.org)
Statsd (https://github.com/etsy/statsd)
Diamond (https://github.com/python-diamond/Diamond)

...con los gestores de base de datos de serie de tiempo propiamente dichos...:

Graphite (https://graphiteapp.org). Formado por un aglutinador de datos provenientes de diversos recolectores llamado Carbon y un gestor de BD propiamente dicho llamado Whisper, además de un graficador propio llamado Graphite-web.
InfluxDB (https://www.influxdata.com). Este gestor de BD puede venir acompañado (aunque no tiene por qué) de un recolector de datos desarrollado por el mismo equipo llamado Telegraph y un graficador también propio llamado Chronograf, además, si se desea, de un gestor de alertas llamado Kapacitor. A la suite completa se le suele llamar TICK.
ElasticSearch (https://www.elastic.co). Este gestor de BD (especializado en la indexación y búsqueda de texto -de ahí el nombre-, por lo que es una herramienta muy interesante para el manejo de logs) puede venir acompañado (aunque no tiene por qué) de un recolector de datos desarrollado por el mismo equipo llamado Beats, un aglutinador/parseador/limpiador de datos provenientes de diferentes orígenes llamado Logstash y un graficador también propio llamado Kibana.
Graylog (https://www.graylog.org). Alternativa completa (recolector+aglutinador+almacenador+graficador)
Prometheus (https://prometheus.io) . Alternativa sin recolector (aglutinador+almacenador+graficador)
OpenTSDB (http://opentsdb.net). Necesita HBase para funcionar
KairosDB (https://kairosdb.github.io). Necesita Cassandra para funcionar
Cyanite (http://cyanite.io). Necesita Cassandra para funcionar
Gnocchi (https://gnocchi.xyz). Parte de OpenStack

...y con graficadores que muestran al usuario -via navegador- los datos allí almacenados:

Grafana (http://grafana.org)
Graphsky (http://www.graphsky.org)

Cubism.js (http://square.github.io/cubism)

Se pueden consultar más graficadores en https://github.com/obazoud/awesome-dashboard
Y también (hay más en general) https://github.com/graphite-project/graphite-web/blob/master/docs/tools.rst

También existen herramientas específicas para generar y enviar alertas a partir de determinadas condiciones detectadas en los datos recopilados, como por ejemplo:

Seyren (https://github.com/scobal/seyren)
Cabot (https://github.com/arachnys/cabot)

Incluso existen servicios SaaS (es decir, servicios online) que ofrecen el almacenamiento, gestión de alertas y visualización de los datos que hayamos obtenido mediante nuestros recolectores, de forma que no tengamos que preocuparnos de la gestión de dichos datos. Ejemplos son:

https://www.hostedgraphite.com
https://cloud.influxdata.com
https://www.elastic.co/cloud
http://pnda.io

Per saber más sobre las similitudes y diferencias técnicas entre las herramientas anteriores (y muchas otras) recomiendo la visita a http://db-engines.com

RECOL·LECTORS:

Collectd:

1.-En una màquina virtual Debian amb la tarja de xarxa en mode "adaptador pont" instal.la el recol.lector Collectd des dels repositoris oficials de la distribució i, un cop fet això, edita el seu fitxer de configuració (/etc/collectd/collectd.conf) per tal de:

> NOTA: Si volguessis instal.lar-ho des del repositori que ofereix el propi projecte Collectd, només caldria seguir els pasos indicats a https://collectd.org/download.shtml#debian

a) Establir l'interval general de recollida de dades a 5 segons (directiva *Interval*) i establir també un nom manual a la màquina, que serà el que s'enviarà amb les dades recollides (directiva *Hostname*)

b) Activar (recorda, descomentant si cal la línia *LoadPlugin nomPlugin*) i configurar adientment (recorda, descomentant i editant si cal les directives incloses dins de la secció *<Plugin nomPlugin>...</Plugin>*) els següents plugins, encarregats de definir el destí que tindran les dades a recollir:

csv (https://collectd.org/wiki/index.php/Plugin:CSV)
Per escriure els valors absoluts de totes les dades recollides dins de la carpeta "/var/lib/collectd/csv"

network (https://collectd.org/wiki/index.php/Plugin:Network)
Per enviar els valors absoluts de les dades a un servidor emmagatzemador de tipus Influx (que en aquest cas no serà remot sinò que estarà escoltant a la IP loopback i que posarem en marxa als propers exercicis). Recorda que el servidor Influx rep les dades al port 25826 (UDP)

> IMPORTANT: Recorda que després de modificar el fitxer de configuració, ja sigui per activar/desactivar o reconfigurar un plugin o per qualsevol altre motiu, hauràs de reiniciar el servei Collectd per tal de posar en marxa el recol.lector amb la nova configuració, així: *sudo systemctl start collectd*

c) Entendre per a què serveixen, activar i configurar adientment els següents plugins, encarregats d'obtenir diferents tipus de dades:

> IMPORTANT: Per veure les directives particulars de cadascun dels plugins, has de consultar https://collectd.org/documentation/manpages/collectd.conf.5.shtml
>
> IMPORTANT: També hauràs de consultar, a la pàgina informativa de cada plugin indicada a sota, si té algun tipus de dependència extra més enllà del que ofereix el kernel (crides al sistema, informació guardada a les carpetes /proc i/o /sys, etc)
>
> IMPORTANT: Activa els plugins un a un i comprova, gràcies a tenir el plugin CSV activat, que les dades es comencen a guardar. Pots visualitzar les dades obtingudes mitjançant la importació del fitxer adient dins d'un full de càlcul o bé amb eines online com ara https://plot.ly/create o http://app.rawgraphs.io En qualsevol cas, seria bo fer-ne una captura.

load (https://collectd.org/wiki/index.php/Plugin:Load)
Per saber la càrrega de treball

uptime (https://collectd.org/wiki/index.php/Plugin:Uptime)
Per saber el tems actual, màxim i mínim de funcionament de la màquina

df (https://collectd.org/wiki/index.php/Plugin:DF)
Per saber el % d'espai ocupat només a la partició /dev/sda1

disk (https://collectd.org/wiki/index.php/Plugin:Disk)
Per saber les estadístiques "Disk operations" i "Disk time per operation" només del disk /dev/sda

cpu (https://collectd.org/wiki/index.php/Cpu)
Per saber el % de temps "System", "User", "IOWait" i "Idle" de la CPU

processes (https://collectd.org/wiki/index.php/Plugin:Processes)
Per saber la quantitat de procesos "running", "sleeping", "stopped", etc a més del seu ús de memòria -RSS value- i el número de fitxers oberts

memory (https://collectd.org/wiki/index.php/Plugin:Memory)
Per saber la quantitat de memòria RAM usada, catxejada i lliure

interface (https://collectd.org/wiki/index.php/Plugin:Interface)
Per saber les estadístiques "Bits/s", "Packets/s" i "Errors/s" de només la tarja enp0s3

tcpconns (https://collectd.org/wiki/index.php/Plugin:TCPConns)
Per saber l'estat de les connexions en els ports que es mantenen a l'escolta

conntrack (https://collectd.org/wiki/index.php/Plugin:ConnTrack)
Per saber l'ocupació a la RAM de la taula de conexions

ping (https://collectd.org/wiki/index.php/Plugin:Ping)
Per saber la latència respecte l'ordinador "www.fbi.gov"

battery (https://collectd.org/wiki/index.php/Plugin:Battery)
Per saber l'estat de la càrrega de la bateria -en unitats d'Amperi-hora-

users (https://collectd.org/wiki/index.php/Plugin:Users)
Per contar el número d'usuaris amb sessió iniciada al sistema

tail (https://collectd.org/wiki/index.php/Plugin:Tail)
Per contar inicis de sessió SSH invàlids...consulteu els exemples mostrats a la documentació

NOTA: Les línies que comencen amb ## es corresponen a plugins no compilats (i per tant, no disponibles)

NOTA: El plugin "Syslog" (https://collectd.org/wiki/index.php/Plugin:SysLog) no s'hauria de comentar mai perquè envia al registre del sistema els missatges de log del propi dimoni Collectd. Un mòdul alternatiu seria "LogFile" (https://collectd.org/wiki/index.php/Plugin:LogFile) , el qual envia en canvi aquests missatges de log al fitxer que s'hi indiqui (per exemple, "/var/log/collectd.log")

NOTA: La llista completa de plugins oficials es troba a https://collectd.org/wiki/index.php/Table_of_Plugins

d) Segueix els passos indicats a https://blog.dbrgn.ch/2017/3/10/write-a-collectd-python-plugin/ per tal de crear el teu propi plugin Collectd (gràcies a https://collectd.org/wiki/index.php/Plugin:Python) que llegeixi en temps real la temperatura de la teva CPU. No cal que entenguis cada línia del codi Python proporcionat però sí la idea general. Prova que funcioni.

NOTA: Això mateix es podria aconseguir fent servir el plugin "sensors", però no tindria la mateixa gràcia

e) Segueix els passos indicats a https://github.com/mbachry/collectd-systemd per tal d'afegir i activar el plugin "collectd-systemd" a la llista de plugins activats al seu sistema Collectd. Prova que funcioni.

f) ¿Què mostra la pàgina https://collectd.org/wiki/index.php/Category:Callback_write?

Netdata:

2.-a) A la mateixa màquina de la pràctica 1 instal.la el recol.lector Netdata. Com que no està als repositoris oficials, hauràs de seguir els passos indicats a https://github.com/firehol/netdata/wiki/Installation , els quals es resumeixen en executar (com a root) la següent comanda, la qual s'encarrega de descarregar, compilar i instal.lar automàticament tots els paquets necessaris segons la distribució actual:

bash <(curl -Ss https://my-netdata.io/kickstart.sh) all

A diferència de Collectd, Netdata incorpora un servidor web/visualitzador en temps real de les dades recollides (les quals per defecte són una sel.lecció de les més habituals: ús de CPU, de memòria, de disc, de xarxa, etc però això ho canviarem) ja plenament funcional sense haver de configurar res. Aquestes dades es mantenen a la memòria RAM però no tenen cap persistència.

b) Un cop hagis comprovat que el servei Netdata estigui encés (*sudo systemctl status netdata*), obre un navegador a la teva màquina real i apunta a la direcció http://ip.maq.virtual:19999 i fes una captura de pantalla. Respon a les següents preguntes:

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_cpu ?

-Llegeix l'explicació que acompanya la gràfica i, a partir d'aquí, digues per què és important monitoritzar els valors de "iowait" i "softirq"?

-Quina diferència hi ha entre la gràfica anterior i l'anomenada "Utilization" mostrada a http://ip.maq.virtual:19999/#menu_cpu_submenu_utilization ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_ram ?

-Què mostra la gràfica anomenada "System" mostrada a http://ip.maq.virtual:19999/#menu_mem_submenu_system ?

-Què mostra la gràfica anomenada "Kernel" mostrada a http://ip.maq.virtual:19999/#menu_mem_submenu_kernel ? Què vol dir memòria "dirty" ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_disk ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_disk_submenu_sda ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_network ?

-Què mostra la gràfica anomenada "Sockets" mostrada a http://ip.maq.virtual:19999/#menu_ipv4_submenu_sockets ?

-Què mostra la gràfica anomenada "Packets" mostrada a http://ip.maq.virtual:19999/#menu_ipv4_submenu_packets ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_netfilter_submenu_conntrack ?

-Què mostra la gràfica http://ip.maq.virtual:19999#menu_net_submenu_enp0s3 ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_load ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_processes ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_entropy ?

-Què mostra la gràfica http://ip.maq.virtual:19999/#menu_system_submenu_uptime ?

La configuració de Netdata es pot veure a la direcció http://ip.maq.virtual:19999/netdata.conf. Físicament es troba al fitxer /etc/netdata/netdata.conf. Aquest fitxer té les directives organitzades per diferents seccions. Les seccions (i dins d'elles, les directives) més rellevants són:

Secció [global]

**hostname* : Estableix el nom de la màquina que serà enviat juntament amb les dades recollides

**update every* : Defineix l'nterval de recollida de dades (en segons). Implica també l'interval de refresc de les gràfiques

**history* : Defineix el número de segons que es mantindrà les dades a la memòria RAM abans d'esborrar-se. El valor de 3600 (és a dir, 1h de retenció) implica un ús de 15MB de RAM; 2h implica 30MB, 4h implica 60MB, i així)

Secció [web]

**default port* : Defineix el port per defecte on se serviran les dades via HTTP. També existeix la directiva *bind to* , la qual permet definir el port sinó també restringir per quina IP (o quines si s'especifica més d'una, separades per espais) se serviran aquestes dades.

**allow connections from* : Indica una llista d'IPs permeses (separades per espais) des d'on es podran visualitzar les gràfiques via HTTP. Es permet el símbol "*" com a comodí i el símbol "!" com a negador. Existeix una altra directiva més específica que restringeix la visualització de l'arxiu netdata.conf via HTTP anomenada *allow netdata.conf from* (i que segueix la mateixa sintaxi)
NOTA: Netdata per ara no admet ni autenticació d'usuaris ni peticions HTTPS. Per aconseguir ambdues coses, caldria "amagar" Netdata darrera d'un servidor web que si ho admetés. D'això se'n diu muntar un servidor web com a "proxy revers". A https://github.com/firehol/netdata/wiki/Running-behind-apache es troba una guia de com fer-ho per Apache

**disconnect idle clients after seconds* : Defineix el número de segons després dels quals es deconnectaran els clients webs que no estiguin actius

Secció [plugins]

Conté la llista de plugins que permeten la recollida de dades (si el seu valor és "yes" voldrà dir que estan activats). El plugin més important és "proc", el qual és el responsable d'obtenir la majoria de dades estudiades a l'apartat anterior. Si es vol deshabilitar l'obtenció d'alguna dada concreta oferida per aquest plugin sense deshabilitar el plugin sencer, això es pot fer dins de la secció genèrica *[plugin:proc]* (la qual conté la llista concreta de fitxers del kernel (de tipus /proc i /sys) d'on s'obtenen la majoria de dades) i/o també dins de seccions específiques *[plugin:proc:xxx]* si s'escaiés.
NOTA: De fet, qualsevol plugin té una secció genèrica anomenada [plugin:nomPlugin] on es configura el seu comportament

Un altre plugin important és "python.d", el qual permet utilitzar plugins propis programats en Python (i ubicats dins de la carpeta "/usr/libexec/netdata/python.d")

Un tutorial genèric sobre com escriure plugins propis fent servir qualsevol llenguatge de programació (com Bash, Node.js oPython) es troba a https://github.com/firehol/netdata/wiki/External-Plugins
No obstant, a https://github.com/firehol/netdata/wiki/How-to-write-new-module es troba un tutorial concret per Python.

La llista oficial de plugins, principalment relacionats amb programes de tercers es troba a https://github.com/firehol/netdata/wiki/Add-more-charts-to-netdata

Secció [health]

Aquesta secció serveix per establir el comportament de les alertes (prèviament definides en forma de sengles fitxers *.conf dins de la carpeta /etc/netdata/health.d...alguns d'ells es troben explicats a https://github.com/firehol/netdata/wiki/health-configuration-examples). Les directives més rellevants són:

**enabled* : Si val "no" les alertes no estaran disponibles

**script to execute on alarm* : Indica la ruta de l'script que s'executarà en detectar qualsevol de les alertes definides (pot consistir en l'enviament d'un correu, d'un missatge de Telegram o l'execució d'un determinat programa, etc)

**run at least every seconds* : Indica cada quants segons es comprovarà l'estat de les alertes definides

Secció [backend]

Aquesta secció serveix per definir el servidor (remot o no) emmagatzemador on s'enviaran les dades recollides. Ho estudiarem més endavant, quan introduïm l'InfluxDB però ja podem dir quines són les directives més importants:

**enabled* : Si val "no" l'enviament a l'emmagatzemador no estarà disponible

**destination* : El valor d'aquesta directiva ha de ser una llista (separada per espais) d'ítems amb la forma "*protocol:IP:port*" (sense cometes) on "protocol" pot ser "udp" o "tcp", la IP serà la d'una màquina emmagatzemadora i el port serà per on aquesta escoltarà (en el cas de l'InfluxDB sol ser el 25826/UDP). Netdata will use the first available item to send the metrics.

**buffer on failures* : Nº d'intents d'enviament de dades on aquestes s'emmagatzemaran en un buffer intern de Netdata abans de descartar-les (si el destí no es trobés i per tant, es considerés invàlid). El temps (en milisegons) que s'espera Netdata a trobar el destí en cada enviament abans de donar-lo per invàlid ve donat per la directiva *timeout ms*

**type* : Netdata admet diferents tipus de formats a l'hora d'enviar les dades a l'emmagatzemador: el format "graphite" (basat en text pla i utilitzat per Graphite i InfluxDB entre altres), el format "opentsdb" (utilitzat per OpenTSDB i altres), el format "json" (genèric) o bé el format "prometheus" (específic d'aquest emmagatzemador concret). En el cas del primer format (que és el que farem servir), les mètriques són enviades amb la forma "prefix.hostname.chart.dimension" on "prefix" es configura a la línia *prefix* (explicada a sota), "hostname" es configura a la línia "hostname" (explicada a sota), "chart" es correspon al nom d'una gràfica/taula (per exemple, "memory") i "dimension" a una de les dades/columnes possibles que podria tenir aquesta gràfica/taula (per exemple, "free" o "used", etc).

**prefix* : Indica el valor del prefix present en les mètriques a enviar a l'emmagatzemador

**hostname* : Indica el nom de la màquina present en les mètriques a enviar a l'emmagatzemador (si no s'indica cap, s'utilitzarà el hostname especificat a la secció global)

**update every* : Indica cada quants segons s'enviarà a l'emmagatzemador la suma/mitjana de les dades recollides durant aquest interval. Since netdata collects thousands of metrics per server per second, which would easily congest any backend server when several netdata servers are sending data to it, netdata allows sending metrics at a lower frequency.

*_data source_ : Pot tenir tres valors: "sum", "average" o "as collected".

sum: the sum of the interpolated values shown on the netdata graphs is sent to the backend. So, if netdata is configured to send data to the backend every 10 seconds, the sum of the 10 values shown on the netdata charts will be used.

average: the average of the interpolated values shown on the netdata graphs is sent to the backend. So, if netdata is configured to send data to the backend server every 10 seconds, the average of the 10 values shown on the netdata charts will be used.

as collected: the latest collected value is sent to the backend. This means that if netdata is configured to send data to the backend every 10 seconds, only 1 out of 10 values will appear at the backend server. The values are sent exactly as collected, before any multipliers or dividers applied and before any interpolation.

*_send charts matching_ : Includes one or more space separated patterns (optionally with * and/or ! symbols).The patterns are checked against both chart id and chart name allowing us to filter which charts will be sent to the backend. So to match all charts named "apps.*" except charts ending in "*reads", use "!*reads apps.*" (the order is important: the first pattern matching the chart id or the chart name will be used - positive or negative).

*_send hosts matching_ : Includes one or more space separated patterns (optionally with * and/or ! symbols). The patterns are checked against the hostname, allowing us to filter which hosts will be sent to the backend when this netdata is a central netdata aggregating multiple hosts.

*_host tags_ : Defines tags (as a list of TAG=VALUE) that should be appended on all metrics for the given host. Note these are currently only sent to opentsdb and prometheus. Please use the appropriate format for each time-series db.

Per conèixer la llista completa de directives i els seus possibles valors, consulteu: https://github.com/firehol/netdata/wiki/Configuration

c) Configura Netdata per a què només tingui activat el plugin corresponent a les gràfiques "Load" i "Uptime", les quals només s'hauran d'actualitzar cada 10 segons Comprova-ho visitant de nou el dashboard web, el qual ara, a més, haurà d'estar accessible a través del port 8080 en comptes del 19999.

d) ¿Què explica aquest article: https://github.com/firehol/netdata/wiki/General-Info---charts.d ?

e) ¿Què explica aquest article: https://github.com/firehol/netdata/wiki/Custom-Dashboards ?

EMMAGATZEMADORS:

InfluxDB

3.-a) Clona la màquina usada als exercicis anteriors. Arrenca aquesta nova màquina i canvia-li el nom de la màquina amb la comanda *hostnamectl set-hostname*. Reinicia-la i comprova que ara aparegui aquest nou nom dins les directives "Hostname" dels arxius collectd.conf i netdata.conf. Si és així, tindrem en marxa dues màquines recol.lectores de dades, cadascuna amb un nom diferent.

b) Vés a https://www.influxdata.com/time-series-platform i digues quin paper realitzen els components Telegraf, InfluxDB, Chronograf i Kapacitor dins de l'ecosistema TICK

We already know that a Time Series Database (TSDB) is a database optimized for time-stamped or time series data. Time series are simply measurements or events that are tracked, monitored, downsampled, and aggregated over time. This could be server metrics, application performance monitoring, network data, IoT sensor data, events, clicks, trades in a market, and many other types of analytics data. The key difference with time series data from regular data is that you're always asking questions about it over time.
NOTA: If you want to go deeper in the theory of TSDBs, I recommend this very interesting introductory article: https://blog.timescale.com/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563

If you're familiar with relational database (RDB) management software like MySQL or PostgreSQL, tables, columns, rows and primary keys will be familiar terms to you, and SQL language will be your "lingua franca" used to manage everything through well-known declarative sentences (CREATE, SELECT, INSERT, UPDATE, DELETE, etc). Because RDBMS is a kind of software already very extended and very known, InfluxDB developers decided design the interaction offered by its product to the admin user as if it were a SQL-based DB (that is, it offers SQL-like sentences like SELECT, INSERT, CREATE, etc using a SQL-ish language called InfluxQL), although it really isn't a relational database at all!.

In this sense, in InfluxDB we could assimilate the concept of relational table to a (new) concept called "measurement". Each "measurement", like traditional tables, contains information which pertains to a specific entity (e.g "cpu" measurement contains CPU utilisation data, "memory" measurement conatins memory utilization data, etc). Each record ("row") stored inside of a measurement is known as a "point". Points are made up of the following "columns":

time : Timestamp that represents the time in which the data belonging to that row was recorded. This timestamp has this format YYYY-MM-DDTHH:MM:SS.nnnnnnnnnZ (RFC 3339)

field : Contain the actual measurement data, e.g 5% CPU utilisation. Each point must contain one or more fields. Field values are your data, so they can be strings, floats, integers, or booleans. However, remember adding a trailing "i" to the end of the field value when writing an integer. If you do not provide the i, InfluxDB will treat the field value as a float.

tags : Metadata about the data being recorded, e.g the hostname of the device whose CPU is being monitored. Each point can contain zero or more tags. Tag values are always stored as strings. Important: tags are indexed while keys are not; as a result, queries run against tags perform much better than those which are performed against keys. So:

Store data in tags if they're commonly-queried meta data
Store data in tags if you plan to use them with GROUP BY()
Store data in fields if you plan to use them with an InfluxQL function
Store data in fields if you need them to be something other than a string

InfluxDB identifies the type of the title of each column depending on if it corresponds to a field ("field key") or a tag ("tag key") , respectively.

A "field set" is the collection of field key/field value pairs on a point.
A "tag set" is the collection of tag key/tag value pairs on a point
A "serie" is a collection of data in a measurement that share a tag set (and retention policy).

c) Després d'haver llegit la teoria anterior, a qualsevol de les dues màquines virtuals que ara tens, instal.la l'emmagatemador InfluxDB. La versió dels repositoris oficials és antiga, així que recomano que segueixis els passos d'instal.lació oficials indicats a https://docs.influxdata.com/influxdb/v1.5/introduction/installation , els quals es resumeixen en executar (com a root) les següents comandes:

curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -
echo "deb https://repos.influxdata.com/ubuntu ***artful*** *stable" > /etc/apt/sources.list.d/influxdb.list*

InfluxDB's Command Line Interface (influx) is an interactive shell that comes with InfluxDB (but be careful: in official Ubuntu repository comes in a separate package called "influxdb-client"!). You can use this client to write data (manually or from a file), query data interactively via InfluxQL queries and view query output in different formats. You can execute some CLI-specific commands directly as well.

There are several arguments you can pass into influx when starting (list them with –help argument):

-host *name* : The host to which influx connects. By default, InfluxDB runs on localhost
-port *port* : The port to which influx connects. By default, InfluxDB runs on port 8086
-username *username* : The username influx uses to connect to the server.
-password *password* : The password influx uses to connect to the server. influx will prompt for a password if you leave it blank (-password ").
-database *DBname* : The database to which influx connects
-execute *command* : Execute an InfluxQL command and quit. It often needs specifying -database argument too (depending on InfluxQL command to execute)
-format *json|csv|column* : Specifies the format of the server responses
-pretty : Turns on pretty print for the json format.
-ssl : Use https for requests.
-version : Display the InfluxDB version and exit.

d) Comprova que el servidor Influx estigui funcionant (*sudo systemctl status influxdb*) i connecta't-hi amb el client *influx* (per defecte no es demana cap tipus d'autenticació) indicant que voldràs totes les respostes en format "column"

Un cop a dins del client *influx*, es poden executar tot un seguit de comandes pròpies del client o també qualsevol sentència InfluxQL. A continuació es mostra una llista (breu) dels primers:

USE *DBname* : Sets the current working database

AUTH : Prompts you for your username and password

FORMAT *json|csv|column* : Specifies the format of the server responses

PRETTY : Turns on pretty print for the json format.

HISTORY : Displays your command history. To use the history while in the shell, simply use the "up" arrow. influx stores your last 1,000 commands in your home directory in .influx_history

SETTINGS : Outputs the current settings for the shell including the Host, Username, Database, etc

EXIT, QUIT or Ctrl+D : Quits the influx shell.

CTRL +C : Cancels a long-running InfluxQL query.

A continuació es mostra una llista (breu) de comandes InfluxQL més importants (la referència completa es troba a https://docs.influxdata.com/influxdb/v1.5/query_language/spec):

SHOW DATABASES : Mostra la llista de base de dades existents

CREATE DATABASE nomBD : Crea una BD

SHOW MEASUREMENTS : Un cop dins d'una BD (gràcies a la comanda USE BD), mostra els "measurements" allà presents. Aquesta comanda admet els següents filtres opcionals:

WITH MEASUREMENT =~ /*exprReg*/
WHERE *tagKey* {= | != | =~ | !~ } *valor*
LIMIT *x*
OFFSET *y*

SHOW TAG KEYS FROM nomMeasurement : Mostra la llista de "tags" d´un "measurement"

SHOW FIELD KEYS FROM nomMeasurement : Mostra la llista de "fields" d´un "measurement" i el seu tipus

SELECT * FROM nomMeasurement : Mostra els valors de tots els "fields" i "tags" de tots els "points" del "measurement" indicat. En comptes del "*" es poden indicar, separats per comes els "field keys" (com a mínim s'ha d'indicar sempre un obligatòriament!) i/o "tag keys" les dades dels quals es volen visualitzar. Aquesta comanda admet els següents filtres opcionals:

NOTA: També es poden indicar operacions matemàtiques bàsiques entre "fields" (sumes, restes, multiplicacions, divisions, mòduls) directament com a resultat a obtenir de la consulta o bé com a condicions del filtre WHERE que veurem a continuació.

WHERE time { = | != | > | >= | < | <= } 'YYYY-MM-DDTHH:MM:SSZ' [AND time ...]
WHERE time { = | != | > | >= | < | <= } now() {+| -} *exprTemps* [AND time ...]
on *exprTemps* és un número seguit de la lletra "s","m","h","d" o "w"
WHERE *fieldKey* { = | != | > | >= | < | <= } *valor* [{AND | OR } *condicio2*] ...
WHERE *tagKey* { = | != } *valor* [{AND | OR } *condicio2*] ...
LIMIT *x*
OFFSET *y*

La consulta SELECT també admet altres modificadors com ara:

ORDER BY time [DESC] : Ordena els resultats obtinguts
TZ("*Europe/Madrid*") : Mostra el temps a la zona horària indicada (ja que es guarda sempre en UTC i per tant, es mostra per defecte així)
GROUP BY *tagkey* : Agrupa dades a l'hora de realizar càlculs estadístiques.

Aquests càlculs s'indiquen com un camp més de dades a obtenir en 'una consulta gràcies a l'ús de funcions com les seguents (per saber-ne més, veure https://docs.influxdata.com/influxdb/v1.5/query_language/functions):
SELECT **MEAN(*fieldkey*)** ... FROM ... GROUP BY *tagkey*
SELECT **SUM(*fieldkey*)** ... FROM ... GROUP BY *tagkey*
SELECT **COUNT(*fieldkey*)** ... FROM ... GROUP BY *tagkey*
SELECT **MAX(*fieldkey*)** ... FROM ... GROUP BY *tagkey*
SELECT **MIN(*fieldkey*)** ... FROM ... GROUP BY *tagkey*
SELECT **SPREAD(*fieldkey*)** ... FROM ... GROUP BY *tagkey*
SELECT **MODE(*fieldkey*)** ... FROM ... GROUP BY *tagkey*

Una variant molt utilitzada, sobretot per realitzar gràfiques amb Grafana o similars és la clàusula: WHERE *rangTemps* GROUP BY time(*exprTemps*).

NOTA: Opcionalment al final s'hi pot afegir la clàusula FILL(*accio*), la qual defineix què s'ha de fer si no s'obtenen dades, on "accio" pot ser "null" (assimila el no obtenir dades al valor *null*), "previous" (assimila el no obtenir dades a la darrera dada bona obtinguda), "linear" (realitza una interpolació lineal), "none" (no fa res).

Exemples
SELECT * FROM hola WHERE time > now() - 30s
SELECT * FROM hola WHERE cpu='cpu-total' AND host='ubuntu' AND time > now() - 30s
SELECT MEAN(diskspace_used) FROM disk_stats WHERE time() >= 3m GROUP BY time(10d)

INSERT [nomRetPolicy] nomMeasurement[,*tagkey1=valor1*][,...] *fieldkey1=valor1*[,...]
Si el "measurement" no existeix en fer la sentència anterior, es crea automàticament
NOTA: Els "points" es guarden a la base de dades fent servir el format anomenat "Line Protocol", el qual segueix bàsicament aquest patró: "nomMeasurement,*tagkey1=valor1*[,...] *fieldkey1=valor1*[,...] timestamp"
Per més informació, veieu https://docs.influxdata.com/influxdb/v1.5/write_protocols/line_protocol_reference/

DELETE FROM nomMeasurement : Buida el "measurement" indicat. Si només es volen esborrar determinats "points" cal afegir filtres amb la clàusula WHERE, com ara WHERE *tagKey = valor* o bé WHERE time < "*yyyy-mm-dd*" , per exemple

DROP MEASUREMENT nomMeasurement : Esborra el "measurement" indicat completament

DROP DATABASE nomBD : Esborra la base de dades indicada

e) Crea una base de dades anomenada "collectdBD", una altra anomenada "netdataBD" i una altra anomenada "proves". En aquesta darrera inserta dins d'un measurement anomenat "sensors", un valor qualsevol de tipus sencer en un "field" anomenat "temperatura" i el valor concret "casa" en un "tag" anomenat "lloc". Torna a inserir un altre "point" amb un valor diferent pel "field" "temperatura" però amb el mateix valor "casa" pel "tag" "lloc". Finalment, torna a inserir un altre "point" amb un valor diferent pel "field" "temperatura" i ara amb el valor "jardí" pel "tag" "lloc". Observa amb les sentències SHOW FIELD KEYS, SHOW TAG KEYS i SELECT * FROM *temperatura* que hagis inserit les dades correctament

f) Calcula (i mostra) la mitjana de la temperatura a "casa" fins ara

La configuració de l'InfluxDB es troba al fitxer "/etc/influxdb/influxdb.conf". La llista completa de directives de configuració es troba a https://docs.influxdata.com/influxdb/v1.5/administration/config però nosaltres només editarem les que tinguin a veure amb activar la rebuda de dades per part de l'InfluxDB provinents de diferents recol.lectors (Collectd i Netdata en aquest cas). Concretament:

*Per emmagatzemar dades rebudes del Collectd: suposant que el seu plugin "Network" ja estigui funcionant adientment, a la configuració de l'InfluxDB cal descomentar les línies següents:

```
[[Collectd]]
enabled=true
bind-address=":25826"                    #Davant dels : es pot posar una IP concreta
database="collectdBD"                    #S'HA DE CREAR PRIMER!!
typesdb="/usr/share/collectd/types.db"   #Informa a l'InfluxDB dels tipus de dades del Collectd
retention-policy = ""                    #En parlarem més endavant
security-level = "none"          #Cal configurar el plugin "Network" de Collectd adientment
                                 #per tal de fer servir els altres valors, que són "sign" o "encrypt".
```

NOTA: Una altra secció interessant del fitxer de configuració és l'anomenada *[http]*, la qual gestiona (per exemple amb les directives *enabled* o *bind-address*) l'accés via HTTP dels possibles consumidors de dades (els quals van des de la mateixa comanda *influx* fins clients HTTP genèrics com *curl* -de seguida ho veurem- o el propi Grafana, entre d'altres)

*Per emmagatzemar dades rebudes del Netdata: suposant que l'Influx ja estigui indicat com a backend dins de la secció *[backend]* de la configuració del Netdata (repasseu el document del Netdata!...encara que només són imprescindibles les línies *enabled=yes, type=graphite, destination=udp:localhost:25827, update every=10, data source=average* i *prefix=netdata*), a la configuració de l'InfluxDB cal descomentar les línies següents:

```
[[graphite]]
enabled = true
bind-address = ":25827"          #Ha de ser diferent del port usat per rebre dades del Collectd!!
database = "netdataBD"           #S'HA DE CREAR PRIMER!!
protocol = "udp"
retention-policy = ""            #En parlarem més endavant
```

g) Fes que les dades actualment recollides tant pel Collectd com pel Netdata de les dues màquines virtuals on els tens instal.lats s'emmagatzemin a les bases de dades "collectdBD" i "netdataBD", respectivament.

h) Comprova que el pas anterior estigui correctament realitzat executant les següents consultes en les dues bases de dades però, en ambdós casos, obtenint les dades per només una de les màquines monitoritzades:

-(Només) el valor màxim d'ocupació del disc dur en el últims cinc minuts
-(Només) el valor mínim de memòria lliure en els últims cinc minuts
-(Només) la mitjana del %IDLE de la cpu en els últims cinc minuts
-Tots els valors de l'estat de la càrrega de la bateria en els últims cinc minuts i el timestamp associat

El mètode principal utilitzat per l'InfluxDB per oferir dades, per defecte a través del port 8086, als diferents clients (*influx*, Grafana, etc) és una API HTTP, la qual també es pot utilitzar des de qualsevol client HTTP, com ara el *curl* To manually perform a HTTP query to InfluxDB, just send a GET request (this is done in *curl* by adding the -G parameter) to the "/query" endpoint, set the URL parameter "db" as the target database, and set the URL parameter "q" as your query. You may also use a POST request (this is done in *curl* by adding the -X POST parameter) by sending the same parameters; the type of used verb depends on the type of query, as it's listed below:

Verb	Query Type
GET	Use for all queries that start with: SELECT or SHOW
POST	Use for all queries that start with: ALTER, CREATE, DELETE, DROP, GRANT, KILL or REVOKE

InfluxDB returns JSON; the results of your query appear in the "results" array. If an error occurs, InfluxDB sets an "error" key with an explanation of the error. Appending pretty=true to the URL enables pretty-printed JSON output. While this is useful for debugging or when querying directly with tools like *curl* but it is not recommended for production use as it consumes unnecessary network bandwidth.

Let's see some examples:

*An example of GET query:
curl -G http://localhost:8086/query --data-urlencode="q=show databases"

NOTA: The –data-urlencode parameter is necessary to automatically change several special character like "$","%","_"," ", etc to a codified equivalent symbol understood by standard HTTP URIs.. More on this in https://www.w3schools.com/tags/ref_urlencode.asp

*Another example of GET query:
*curl -G http://localhost:8086/query?pretty=true --data-urlencode "db=mibd"
--data-urlencode "q=SELECT value FROM cpu_load_short WHERE region='us-west'"*

*An example of POST query:
curl -X POST http://localhost:8086/query --data-urlencode "q=create database mibd"

More tricks

*To send multiple queries in a single call simply delimit each query using a semicolon.

*Everything in InfluxDB is stored and reported in UTC using the RFC3339 standard If you want timestamps in Unix epoch format include in your request the query string parameter epoch where *epoch=[h,m,s,ms,u,ns]*

*In you have enabled HTTP authentication (see below) , use *u=username* and *p=password* query string parameters. If you use Basic authentication, do curl -XPOST -u myusername:mypassword

With POST queries it is also possible to write data into a BD using the "/write" endpoint and the *–data-binary* curl's argument (it would be similar to execute a INSERT sentence). For example: curl -X POST http://localhost:8086/write?db=mibd --data-binary "cpu,host=serverA value=$(cat /proc/loadavg | cut -f1 -d' ')"

NOTA: Per saber més, https://docs.influxdata.com/influxdb/v1.5/guides/writing_data/

i) ¿Què fa aquest script:

```
#!/bin/bash
FREE=$(free | grep "^Mem" | tr -s " " | cut -d " " -f 4)
curl -XPOST "http://127.0.0.01:8086/write?db=proves" --data-binary  "memory,host="$(hostname)" free="$FREE"
```

Prova'l

NOTA: Si es vol que el valor passat al paràmetre –data-binary de curl provingui d'una canonada, s'ha d'indicar així: *--data-binary @-*

Las consultas HTTP (ya sean de lectura o de escritura) se pueden autenticar para que tan solo aquellos clientes que proporcionen el usuari y contraseña adecuada puedan ejecutarlas. Para ello, los pasos son los siguientes:

1.-Crear al menos un usuario administrador, así:
CREATE USER nombreUsu WITH PASSWORD 'contraseña' WITH ALL PRIVILEGES

1BIS.-Opcionalmente, crear más usuarios no administradores, así...
CREATE USER nombreUsu WITH PASSWORD 'contraseña'

...a los cuales se les deberá asignar los permisos que deseemos (básicamente, poder leer datos y poder escribir datos) en cada base de datos una a una, así...:
GRANT {READ|WRITE|ALL} ON nombreBD TO nombreUsu

...o quitarlos si hiciera falta, así:
REVOKE {READ|WRITE|ALL} ON nombreBD FROM nombreUsu

NOTA:Se puede ver la lista de usuarios actuales y saber si son administradore con el comando SHOW USERS
NOTA:Se puede ver la lista de permisos en todas las Bds para un usuario concreto con el comando SHOW GRANTS FOR nombreUse
NOTA:Se puede cambiar la contraseña de un usuario concreto con el comando SET PASSWORD FOR nombreUsu = 'contraseña'
NOTA:Se puede eliminar un usuario concreto con el comando DROP USER nombreUsu

2.-Activar en la configuración del InfluxDB (archivo /etc/influxdb/influxdb.conf) la autenticación. Esto se hace estableciendo la línia *auth-enabled = true* de la sección [http] y reiniciando el servicio

j) Crea un usuari administrador anomenat "admin" i un altre que ho sigui anomenat "pepe", el qual només haurà de tenir permisos de lectura per la base de dades "proves" i res més. Executa el client *influx* loguejant-te primer com a "admin" i comprova que pots llegir qualsevol "measurement" de qualsevol base de dades; intenta també d'escriure algun valor qualsevol al "measurement" "temperatura" de la base de dades "proves" per comprovar que també ho pots fer. Tanca sessió i ara entra amb l'usuari "pepe"; comprova que no pots llegir cap dada d'enlloc excepte la base de dades "proves", però que allà no pots escriure tampoc.

k) Realitza les consultes GET i POST pertinents amb curl per comprovar un altre cop que, efectivament, l'usuari "pepe" només pot llegir dades de la base de dades "proves" però no pot escriure enlloc.

Using passwords without encryption is very dangerous. To enable HTTPS and use it instead of plain HTTP it's necessary to create a TLS key and certificate pair, like this:

1.-Create a self-signed certificate for Influx (as root):

cd /etc/ssl
openssl genrsa -out influxdb.key 2048
openssl req -new -key influxdb.key -out influxdb.csr
openssl x509 -req -in influxdb.csr -signkey influxdb.key -out influxdb.crt

NOTA: Another way of doing the same but in only one step is executing this command:
openssl req -x509 -nodes -newkey rsa:2048 -keyout influxdb.key -out influxdb.crt

2.-Enable TLS by editing these lines below [http] section in /etc/influxdb/influxdb.conf file :

https-enabled = true
https-certificate = /etc/ssl/influxdb.crt
https-private-key = /etc/ssl/influxdb.key

You should restart influxdb service, then. Since TLS is enabled you will need to pass a few more flags in order for connect to server with *influx* client. Specifically, you must execute this command: *influx -ssl -unsafeSsl* (the -unsafeSsl argument is necessary to disable the clients validation step since our certificate is self-signed)

l) Repeteix els apartats **j)** i **k)** però ara havent activat el HTTPS

GRAFICADORS:

Grafana:

L'explicació de Grafana (https://grafana.com) que hi ha a la seva pròpia web explica prou clar per a què serveix i què fa aquest programa: *"Grafana is an open source feature rich metrics dashboard and graph editor for metrics databases. It provides a powerful and elegant way to create, share, and explore data and dashboards from your disparate metric databases. It supports a wide variety of graphing options for ultimate flexibility. It also supports authenticated login and a basic role based access control implementation"*

Les comandes necessàries per instal.lar Grafana en un sistema Ubuntu són les següents (com a root):

curl https://packagecloud.io/gpg.key | sudo apt-key add -
echo "deb https://packagecloud.io/grafana/stable/debian stretch main" >> /etc/apt/sources.list.d/grafana.list
apt update
apt install grafana
systemctl start grafana-server

Un cop funcionant el servidor Grafana, per entrar al panell de control web simplement cal escriure a la barra de direccions de qualsevol navegador la url http://ip.Serv.Grafana:3000 i accedir-hi amb l'usuari "admin" i contrasenya "admin". Ja dins, és interessant consultar les opcions que ofereixen els següents llocs:

"Settings" (icona d'engranatge a la cantonada superior esquerra): dóna pas a un panell amb diferents categories generals d'administració de Grafana

"Preferences" (icona de l'usuari loguejat, a la barra lateral esquerra, a sota; es pot accedir directament anant a http://192.168.1.38:3000/profile) : dóna pas a un panell amb diferents categories pròpies de l'usuari en particular com "UI Theme", "Change password", ...

"Configuration" (icona d'engranatge a la barra lateral esquerra): dóna pas a un panell amb diferents categories importants, com ara http://192.168.1.38:3000/datasources (per definir els orígens de les dades que es mostraran -en el nostre cas serà InfluxDB-), http://192.168.1.38:3000/org/users (per gestionar els diferents possibles usuaris reconeguts per Grafana i el seu diferent grau de permisos), http://192.168.1.38:3000/plugins (per gestionar els diferents plugins disponibles), entre d'altres.

"Alerting" (icona de campana a la barra lateral esquerra; es pot accedir directament anant a http://192.168.1.38:3000/alerting) :dóna pas a un panell útil per definir canals de notificacions i regles per activar alertes. Ho veurem més endavant.

Connexió amb InfluxDB

La llista d'orígens de dades compatibles amb Grafana es troba aquí: http://docs.grafana.org/features/datasources Concretament, per connectar amb un servidor InfluxDB ja funcional i mostrar les dades allà presents, el que cal fer és afegir un "Data Source" (fent servir alguns dels mètodes explicats als paràgrafs anteriors) amb els següents valors personalitzats (la resta es poden quedar amb el que tenen per defecte):

*Nom : el que es vulgui (i convertir-lo en el "data source" per defecte)
*Tipus : InfluxDB
*URL : http://ip.Serv.Influx:8086
*Database : nom de la base de dades
*User/Password (només si l'accés a la BD anterior està configurada a l'InfluxDB amb autenticació, el tipus de la qual s'ha d'indicar a l'apartat "Auth" mostrat línies amunt)

La idea, després de realitzar el pas anterior, és tenir una configuració com la següent:

Recol.lector ---UDP/TCP---> :25826 InfluxDB :8086 <---HTTP--- Grafana :3000 <--- HTTP--- *usuari*

Un cop ja tenim definit l'origen de les dades, el següent pas és definir un "Dashboard" i afegir-hi les gràfiques que mostraran aquestes dades. Això es pot fer clicant a la icona "Home" que apareix a la part superior de Grafana (o bé, alternativament, a http://ip.Serv.Grafana:3000/dashboard/new?gettingstarted) Apareixerà llavors el conjunt de tipus de gràfiques a afegir, entre les quals les més interessants són "Graph" i "Singlestat". En qualsevol cas, sigui el tipus de gràfica que sigui, un cop afegida, per editar-la haurem de fer clic sobre el títol i escollir l'opció "Edit graph" (fixeu-vos que hi ha més opcions i cadascuna té un "hotkey" associat per fer servir el teclat directament en comptes del ratolí, com ara duplicar, esborrar, etc).

En el cas d'una gràfica de tipus "Graph", en sel.leccionar l'opció d'editar apareixerà un quadre amb diferents pestanyes, les més importants de les quals ara mateix són "Metrics" i "Axes". A la primera es defineix la consulta a realitzar a l'InfluxDB i a la segona les unitats i l'escala (entre altres detalls) de les gràfiques.

Per definir la consulta es poden utilitzar els botons que ofereix el Grafana o bé escriure-la directament en InfluxQL si se selecciona l'opció "Edit toggle mode" que apareix en clicar sobre el botó de la icona de les tres barretes horitzontals. En el cas de fer servir els botons, cal saber que la paraula "default" que apareix a la fila FROM té a veure amb un concepte d'InfluxDB anomenat "policy retention" i no el modificarem mai; la consulta en sí l'hem de definir seleccionant el "measurement" concret, indicant les clàusules "where" (si n'hi ha) i, a partir d'aquí, els camps concrets a la fila SELECT. Fixeu-vos que per defecte es crea una consulta agrupant dades per temps i mostrant la mitjana (en aquest sentit, la variable *$_interval* de Grafana indica que serà el propi Grafana qui deduirà el periode temporal d'agrupació més òptim segons la quantitat de dades rebudes i l'escala de la gràfica). Per més informació, podeu llegir http://docs.grafana.org/features/datasources/influxdb/

1.-a) Crea un "dashboard" que contingui una gràfica de tipus "Graph" mostrant els valors mitjans (agrupats cada 10 segons) de quantitat de memòria lliure. Elimina l'eix Y dreta i ajusta l'eix Y esquerra a les unitats adients sabent que les dades originals recollides pel Collectd són en bytes. Comprova que els valors visualitzats quadrin amb el que et mostra la comanda *free -h* Guarda el dashboard (per fer això pots pulsar CTRL+S o clicar sobre la icona corresponent).

NOTA: Per saber el que has de posar a la clàusula "where", recomano que realitzis una consulta tal com *select * from memory_value limit 5* (o similar) des del client de consola de l'Influx.

b) Afegeix una gràfica de tipus "Graph" al "dashboard" anterior que mostri els valors "tal qual" de la càrrega de treball en l'últim minut, últims 5 minuts i últim quart d'hora.

NOTA: Si cliques en un valor de la llegenda, la gràfica corresponent serà l'única que es mostrarà. Per obtenir l'efecte contrari (que sigui l'única que s'oculti, cal clicar-hi amb la tecla CTRL). Clicant sobre el rectanglet de color es pot escollir el color de cada gràfica

c) Afegeix una gràfica de tipus "Singlestat" al "dashboard" anterior que mostri la quantitat d'espai de disc utilitzat (has de fer servir el camp amb el filtre adient dins de la clàusula "where") fent una mitjana dels valors recollits cada 10 segons. Cal tenir en compte, un altre cop, que Collectd obté aquesta dada en bytes!
Fes, seguidament, que el valor mostrat tingui la forma de percentatge respecte el tamany total del disc; això ho pots aconseguir de vàries formes: o bé dient-li al plugin corresponent del Collectd que envïi directament les dades en %, o bé fent el còmput (valor_recollit_bytes/tamany_total_bytes)*100 directament sobre la consulta de Grafana, entre altres. Finalment, fes que es mostri un gràfic de tipus "Gauge" (això ho pots trobar a la pestanya "Options") amb tres colors segons si l'espai ocupat no arriba al 50% (verd), al 80% (taronja) o al 100% (vermell).

d) Afegix una gràfica del tipus que vulguis mostrant la dada que vulguis i explica quina dada és

2.-Vés al repositori públic de "dashboards" de Grafana (https://grafana.com/dashboards) i escull el nº 1406 per importar-lo al teu servidor Grafana particular (per fer importació només cal que vagis al buscador de dashboards, disponible en clicar sobre el botó que hi haa la zona superior esquerra que mostra una icona en forma de quatre quadrats, i allà sel.leccionar l'opció "Import dashboad"). ¿Quines dades veus?

NOTA: Per exportar-ne un "dashboard" nostre només caldria clicar sobre la icona de "Share" que apareix a la barra superior dreta quan estem a la pantalla general del dashboard en qüestió

3.-a) L'arxiu de configuració del Grafana és /etc/grafana/grafana.ini. Digues per a què serveixen els següents elements (pots provar, si vols, de canviar algun dels seus valors per defecte i comprovar què passa un cop reiniciar el servidor Grafana):

*La directiva "allow_sign_up" sota la secció [users]
*La directiva "enabled" sota la secció [auth.anonymous]
*Les directives "admin_user" i "admin_password" sota la secció [security]
*La directiva "url" sota la secció [database]

b) Es pot configurar el servidor web propi de Grafana per a què atengui peticions HTTPS en comptes de HTTP. Per aconseguir això, primer s'ha de generar una parella de certificat + clau pública i seguidament indicar aquests a la configuració del servidor. Concretament, obre un terminal, vés a la carpeta /etc/grafana i allà executa (com a root)...:

openssl req -x509 -newkey rsa:2048 -keyout grafana.key -out grafana.crt -nodes -days 365

NOTA: Si la comanda anterior realitza preguntes interactives, pots contestar a totes amb el valor per defecte excepte la que demana pel "Common Name", on hauràs d'escriure la IP del servidor Grafana (o el nom DNS, si en tingués).

...i modifica les següents línies que hi ha sota la secció [server] de l'arxiu grafana.ini per a que quedin així:

protocol=https
http-port=3000
cert-file=/etc/grafana/grafana.crt
cert-key=/etc/grafana/grafana.key

Reinicia el servidor Grafana. ¿Què passa ara quan intentes accedir als "dashboards"?

4.-Grafana ofereix una API REST per realitzar la majoria de tasques que es poden fer a través d'un navegador. Consulta la documentació adient dins de http://docs.grafana.org/http_api/ per tal d'aconseguir esbrinar com es pot canviar el "UI Theme" de fosc a clar i viceversa fent servir *ncat* o *curl*

5.-a) Llegeix l'article http://docs.grafana.org/alerting/rules i, a partir d'aquí, digues què faria aquesta alerta: "avg() OF query(A,5m,now) IS BELOW 14"

b) Llegeix l'article http://docs.grafana.org/alerting/notifications i, a partir d'aquí, enumera 3 tipus diferents de notificacions que pot generar Grafana a partir de la detecció d'una alerta

ALERTADORS:

Poden enviar alertes a un determinat destí (journald, mail....) en superar llindars prèviament definits. Un exemple seria Kapacitor.