Sistemes de monitoratge

Daniel Picó Amador
Enginyeria Tècnica d'Informàtica de Sistemes

Consultor: Enric Peig Olivé
TFC - Xarxes de Computadors

Juny 2016

FITXA del TREBALL FINAL

Títol del treball	Sistemes de monitoratge
Nom de l'autor	Daniel Picó Amador
Nom del consultor	Enric Peig Olivé
Data de lliurament	06/2016
Àrea del treball final	Xarxes de Computadors
Titulació	Enginyeria Tècnica d'Informàtica de Sistemes

Resum del treball

Els sistemes de monitoratge són essencials avui dia a complexes xarxes de computadors. Aquests ens garanteixen que totes les errades produïdes a la xarxa són detectades i informen els operadors per tal de solucionar-les.

Aquest treball està dividit en dues parts, teòric i practic, per aproximar als sistemes de monitoratge i ensenyar com aquests funcionen.

Les eines de monitoratge ens permeten el control sobre una llarg varietat de serveis i processos utilitzant diferents protocols, tècniques per verificar i representar gràficament les dades, per exemple, estats dels serveis de xarxa, l'estat dels servidors webs o sensors de temperatura.

Introdueix el concepte d'Enginyeria del Caos. Una aproximació adreçada a utilitzar el caos en sistemes distribuïts i que serveix per a construir sistemes fiables per aguantar en condicions reals. Vol ensenyar a observar el comportament d'un sistema distribuït mentre es realitzen experiments controlats.

Aquest treball vol dissenyar i implementar un plugin per a Nagios utilitzant el protocol ISO8583 per controlar l'estat de serveis de transacció bancaria.

Utilitzant programari lliure, aquest treball realitzarà la instal·lació d'una plataforma amb Nagios i Cacti que són senzills de configurar, gestionar i utilitzar per als administradors de sistemes.

Paraules clau	Monitoratge, Xarxes, Nagios, Cacti, plugin, ISO8583, Enginyeria del Caos, exèrcit mico

Abstract

Network Monitoring Systems are essential in running the complex computer networks of today. They ensure all faults on the network are known and assist the network operator in fixing these faults.

This project has two defined parts, theoretical and practical, to approach the monitoring systems and it showing how it works. Explanations about different data collectors, time series databases, dashboards.

The monitoring tools allow us to control a large variety of services and process using different protocols and techniques to check and to represent graphically their values, for example, the networks services state, webservices state or temperature sensors

Introduce the concept Chaos Engineering. An empirical, systems-based approach addresses the chaos in distributed systems at scale and builds confidence in the ability of those systems to withstand realistic conditions. We learn about the behavior of a distributed system by observing it during a controlled experiment.

This project aims to plan and implement a nagios plugin using ISO8583 protocol to check the status for transactional bank services. designed from the start to monitor a modern network and includes all tools in a single extendable system.

By using GPL software and making good design decisions, this project aims to install, a Nagios and Cacti platform that are easy to configure, maintain and use for a systems administrator.

Keywords	Monitoring, Networks, Nagios, Cacti, plugin , ISO8583, Chaos Engineering, simian army

ÍNDEX

BLOC PRACTIC

LLISTAT de FIGURES

Introducció

Que el món de la informàtica és un canvi continu no ha de ser un concepte nou. Virtualització, cloud computing, sistemes distribuïts, múltiples localitzacions són conceptes que cada dia tenim més presents en les nostres arquitectures que construïm i això fa que els elements a vigilar s'incrementin de manera vertiginosa. Això implica que ens podem trobar diferents casuístiques amb els elements equips que tenim sota el nostre control i els que formen part de tercers amb els quals mantenim integracions de serveis i que això en conjunt en cas d'error pot impactar negativament al nostre servei.

El monitoratge és un conjunt de tasques complexes i continues que a més a més sota certes circumstàncies és tractada com una especialitat ineficaç, ja que es pot considerar com un esdeveniment reactiu. Detectar errors que ja s'estan produint no genera cap tipus d'avantatge. Per això és important mantenir un canal continu de comunicació per educar i ensenyar que els sistemes de monitoratge detecten errades produïdes i que això alimenta la base de coneixement que permet establir patrons per a ficar remeis i evitar que es torni a reproduir el mateix problema sota les mateixes circumstàncies.

En aquest document es vol realitzar un estudi teòric i pràctic sobre sistemes de monitoratge aplicats sobre sistemes informàtics distribuïts amb l'objectiu per una banda de crear un document on es defineixin i s'expliquin els conceptes necessaris per entendre i establir els mecanismes de supervisió i vigilància sobre els elements monitorats amb l'ajuda del protocol SNMP, realitzar un repàs a diferents agents de supervisió i vigilància, d'aplicacions per la visualització de les dades i per la creació de mètriques. Al llarg del bloc pràctic es presentarà la implementació de controls i mètriques amb els programaris Nagios i Cacti, com també l'estudi en profunditat de la creació d'un plugin creat adhoc que amplia les funcions de vigilància de Nagios.

Contingut

Al bloc teòric

- Definició de la monitoratge
- Abast de la monitoratge, disponibilitat i rendiment
- Recol·lecció, emmagatzemat i visualització de la informació
- Eines, recursos i consideracions
- Estudi de la política d'error continu a Netflix aplicat a la seva plataforma VOD

Al bloc pràctic

- Instal·lació i configuració del programari de monitoratge, Nagios i Cacti
- Definició dels objectes a a monitorar i de generació de mètriques
- Implementació de “checks” específics

Planificació

Segons consta al document del Pla Docent del TFC, les dates clau son

Tasca	Inici	Lliurament
PAC1 - Pla de treball	24/02/2016	13/03/2016
PAC2 - Seguiment	17/04/2016	01/05/2016
PAC3 - Seguiment	23/05/2016	05/06/2016
Memòria		19/06/2016
Presentació		26/06/2016

Això ens dona els següents intervals de temps entre les dates clau

Tasca	Inici	Lliurament	N. dies
PAC1 - Pla de treball	24/02/2016	13/03/2016	18
Desenvolupament de les tasques definides al pla de treball per aquesta fase			35
PAC2 - Seguiment	17/04/2016	01/05/2016	14
Desenvolupament de les tasques definides al pla de treball per aquesta fase			22
PAC3 - Seguiment	23/05/2016	05/06/2016	13
Desenvolupament de les tasques definides al pla de treball per aquesta fase			14
Memòria		19/06/2016	
Desenvolupament de les tasques definides al pla de treball per aquesta fase			7
Presentació		26/06/2016	

Planificant el temps

Com es pot veure de la taula anterior; Els primers 18 dies, estan dedicats a preparar aquest pla de treball. Començar a elaborar un guió basat amb diferents recursos bibliogràfics, principalment recursos que es troben a Internet, que posteriorment seran incorporats a cada un dels capítols.

Els següents 49 dies, que és el període més ampli, on inclourà la recerca dels elements teòrics desenvolupats així com l'elaboració dels primers blocs d'introducció. Aspectes sobre el monitoratge, la disponibilitat i el rendiment. Explicar el concepte de monitoratge com a esdeveniment reactiu com també explicar l'abast del monitoratge indicar fins a on es pot arribar, mínims i màxims per evitar redundàncies, fals positius.

Fer una aproximació a les eines i recursos que actualment hi ha per ajudar-nos per aquestes tasques, donant visibilitat a les eines lliures, Nagios i Cacti.

Els 36 dies, al bloc anterior s'ha començat a treballar la part teòrica del treball i en aquest punt incorporaria una introducció a cadascun dels elements que integren el monitoratge, les dades, el seu tractament i la seva posterior representació. Exemples com SNMP, RRD, InfluxDB, grafana s'explicaran per entendre el concepte global dels sistemes integrats de monitoratge. També, per exemple, l'estudi de la filosofia de Netflix per millorar els seus desenvolupaments enfront d'esdeveniments. Han implementat aquesta filosofia de *continuous monitoring* o *continous failure* amb el seu *Simian Army.*

Deixant ja la part teòrica, començar la part practica definint una plataforma tipus (servidors web, bases de dades...) i instal·lar un servidor, amb Nagios i Cacti. Afegir els checks i gràfiques per monitorar la plataforma.

27 dies, enfocats a finalitzar el bloc pràctic, desenvolupar "checks" específics. Com a mostra sobre com afegir els nostres checks, incorporaré com a exemple un control per a Nagios desenvolupat per controlar la disponibilitat d'un servei bancari. Per poder introduir aquest exemple faré una aproximació d'utilització del protocol de comunicacions per a transaccions bancàries ISO8583.

Incorporar les conclusions a la memòria amb referència als aspectes desenvolupats al bloc teòric aplicats a la pràctica.

Els últims 7 dies restaré a disposició de tribunal per resoldre les qüestions que apareguin i formalitzar la presentació.

BLOC TEÒRIC

Capítol 1
Monitoratge

Els sistemes de monitoratge de xarxa, en angles Network Monitoring System, és un aspecte essencial a l'administració d'una xarxa de computadors amb un gran nombre d'elements. Els sistemes de monitoratge són utilitzats per minimitzar l'impacte de les fallides produïdes i per preveure i minimitzar el percentatge de risc que es produeixi un esdeveniment que afecti la continuïtat del servei. La mida de les xarxes van des de petites oficines fins a grans plataformes i el sistema de monitoratge ha d'estar escalat i adaptat a aquesta mida.

Els sistemes de monitoratge fan servir diferents tècniques:

- consulta periòdica
- gràfiques
- esdeveniments de sistema
- notificacions de sistema

Aquestes tècniques s'apliquen per aconseguir que la xarxa estigui sota vigilància i supervisió en aquells elements crítics i les poden agrupar en dos grans blocs, el monitoratge actiu sobre respostes a esdeveniments i la recol·lecció d'informació per la seva anàlisi.

Aquests dos grups d'eines dels sistemes de monitoratge ens permeten a gestionar el dia a dia dels problemes tècnics i al mateix temps ens permeten estudiar quines parts o elements de la xarxa es poden convertir en possibles punts únics d'errada o bottlenecks.

1.1 Conceptes

Les **consultes periòdiques** *(service polling).* És un tipus de consultes on el sistema de monitoratge mitjançant programes i protocols realitza consultes als dispositius o als serveis demanant sobre el seu estat i els seus paràmetres de funcionament de forma recurrent. Amb aquestes dades obtingudes els sistemes de monitoratge es poden configurar per aplicar accions definides com també desar-les a bases de dades per la seva posterior explotació. Segons com estigui implementada la petició aquesta consulta no solament determinará l'estat sinó també ens pot indicar el rendiment de l'element supervisat.

Representació de les dades. Les dades recol·lectades i emmagatzemades amb marques de temps o el que és el mateix de manera periòdica, es poden representar mitjançant gràfiques. Aquestes poden ser de gran utilitat per detectar tendències *(trends)* i anomalies. Algunes de les gràfiques més comunes poden ser les d'ús de la CPU i les d'utilització de l'amplada de banda de les interfícies.

És important entendre i estudiar com els canvis a les diferents mètriques queden reflectides a les gràfiques per trobar possibles correlacions causa-afecte. Qualsevol canvi a la xarxa o al sistema es pot valorar com bo o dolent, en qualsevol cas l'objectiu del monitoratge és **detectar** aquests canvis. Certs canvis es poden configurar per activar una alerta i/o notificació com a resposta a uns valors.

Idealment el sistema de notificació tractarà de **generar missatges simples** on reflecteixin de forma inequívoca l'esdeveniment el qual notifica. És habitual que un problema sigui l'origen d'altres afectacions i tot resolen la més important la resta quedin resoltes de manera automàtica, per això és important definir de manera intel·ligent el sistema de notificacions per evitar en la mesura del possible a acumulació de notificacions d'esdeveniments relacionats *(flapping events)*.

1.2 Disponibilitat

El factor de disponibilitat d'un element és una mesura que ens indica quant de temps ha estat equip respecte a la duració total que es desitjava que funciones i s'expressa generalment en percentatge. [1]

Es defineix com alta disponibilitat quan aquest factor de disponibilitat assegura un cert grau absolut de continuïtat operacional durant un període, el temps de no-operativitat *(downtime)* és l'utilitzat per definir quan el sistema no ha estat disponible.

1.2.1 Objectiu: 100%

L'objectiu d'arribar al 100% de disponibilitat es complicat i reservat a unes poques grans plataformes.

En un any el nombre de minuts d'inactivitat per un esdeveniment no programat dividit pel nombre total de minuts és dona com a resultat el percentatge de disponibilitat, per tant, per aconseguir un percentatge del 99,9% el temps d'inactivitat no podria superar de les 8 h/any.

A continuació una taula detallada amb els percentatges de disponibilitat i els temps d'inactivitat que això suposa.

Disponibilitat %	Temps d'inactivitat per any	Temps d'inactivitat per mes
90%	36,5 dies	72 hores
99%	3,65 dies	1,68 hores
99,9%	8,76 hores	10,1 minuts
99,99%	52,56 minuts	1,01 minuts
99,999%	5,26 minuts	6,05 segons
99,9999%	31,5 segons	0,605 segons

1.3 Rendiment

Entenem com a rendiment d'una xarxa d'ordinadors als nivells de capacitat, productivitat, latència i fiabilitat que tenen el conjunt dels seus elements per gestionar i operar amb la informació. No mantenir aquests paràmetres dintre dels nivells òptims d'eficiència impactarà sobre el servei proporcionat, quedant afectada la disponibilitat i considerant que l'element no està operatiu.

Capacitat. La capacitat es pot definir com l'habilitat de la qual disposa l'element per aconseguir la gestió d'informació. L'amplada de banda que pot gestionar ens indica la quantitat de dades que pot transmetre en un moment donat.

Productivitat *(Throughput)* És la mida real d'informació que es transmet. És el nombre de paquets d'informació que són rebuts de manera correcta per unitat de temps. Finestra de temps (Time Window) és la unitat de mesura.

Latencia. És el valor amb el qual mesurem el temps que es triga a gestionar la informació. Cadascun dels elements a monitorar tenen uns paràmetres admissibles de latència.

Fiabilitat. Amb aquest paràmetre es pot avaluar quin percentatge d'errors produeix l'element controlat sobre la informació gestionada.

El monitoratge d'aquests paràmetres sobre els elements adequats ens permetrà realitzar diferents anàlisis de rendiment, obtenint una informació molt valuosa per prendre les decisions necessàries de creixement i detecció d'errors.

1.4 Abast

Una xarxa informàtica és un grup interconnectat de computadors. La finalitat principal per a la creació d'una xarxa és compartir els recursos i la informació en la distància, assegurar la confiança i la disponibilitat de la informació, augmentar la velocitat de transmissió de les dades i reduir el cost general d'aquestes accions.

Existeix una correlació entre la mida de la xarxa i el seu ús amb l'impacte que una fallada pot generar. Per exemple incidències a proveïdors d'Internet o empreses de trading poden generar impactes econòmics considerables, ja que els primers generalment tenen xarxes que gestionen les connexions de centenars de clients i els segons poden gestionar centenars de transaccions d'intercanvi bancari.

Amb el creixement de les xarxes i amb elles el nombre dels equips que les gestionen és inviable mantenir-les sota control sense sistemes de monitoratge per supervisiar, vigilar i avançar-se en el possible als esdeveniments que succeeixen.

Com també un programador aplica bones pràctiques i controls sobre el codi programat, els enginyers de sistemes han d'aplicar verificacions de manera periòdica sobre l'estructura per assegurar que no hi ha errades. La periodicitat dels elements monitorats està relacionada amb l'impacte que pot tenir sobre el servei donat i aquesta supervisió periòdica quedarà definida amb una correlació entre tipus d'error i el cost econòmic.

És important que la tasca de manteniment sigui una tasca continua dedicada a aplicar les modificacions dins les eines de monitoratge i al seu abast. Ja que si es produeixen canvis a les infraestructures i no es realitza aquesta tasca de manteniment continu pot derivar que els canvis quedin fora de supervisió i vigilància.

L'evolució dels sistemes de monitoratge van des de col·leccions de petits programes *(scripts)* fins a complets programaris amb múltiples dependències i que com veure més endavant

existeixen múltiples solucions i eines. Però l'objectiu d'aquest treball és centrar-se en aquelles opcions basades en codi obert i llicències lliures i dins d'aquestes característiques la més popular d'aquestes utilitats és Nagios i sens dubte és font d'inspiració per crear sistemes de monitoratge millor.

Els aspectes essencials dels sistemes de monitoratge els quals deuen poder gestionar són:

- Arrancada i aturada del dispositiu
- Accés a la configuració del component
- Arrencada i aturada dels serveis del dispositiu
- Gestió de fitxers de bitàcoles.
- Control d'accés d'usuaris
- Generació, distribució i xifratge de claus d'accés

1.5 Sistemes de monitoratge

1.5.1 Recol·lectar informació

El concepte de recol·lecció d'informació és l'acció d'un client accedint a la informació proporcionada per un agent a l'element supervisat. Per realitzar aquesta tasca ambdues parts necessiten establir com s'organitza la informació i quina via s'utilitzarà per accedir a aquesta informació.

Una gran part dels sistemes de monitoratge utilitzen el protocol SNMP (Simple Network Management Protocol) per la seva versatilitat, maduresa i simplicitat per recol·lectar informació. Més endavant en el capítol 2 s'explica el seu funcionament.

Ara per ara, la idea sobre SNMP és que es tracta d'una definició que ens permet fer consultes simples a identificadors que són una seqüència jeràrquica de noms, els OIDs. Al contrari del que es pot pensar la utilització de SNMP no inclou exclusivament la lectura d'aquests identificadors sinó que també ens permet escriure o reescriure la informació dels OIDs.

1.5.2 Emmagatzemar informació

Si amb aquestes dades recol·lectades el que volem és poder posteriorment representar-les gràficament aquestes s'hauran d'emmagatzemar en bases de dades de sèries de temps. Les cracteristica principal de les bases de dades de sèries de temps es que les dades queden indexades sota una marca de temps.

Round-Robin Databases (RRDs) és un dels sistemes referència per la seva simplicitat per mantenir aquest volum d'informació. El principal avantatge d'emmagatzemar les dades a RRDs seria que ocupen una part petita de mida en disc.

A RRDs les dades són emmagatzemades amb una marca de temps definida i s'estructura per línies d'intervals. Cada línia de temps es permet un nombre limitat de valors i una vegada superat aquest nombre s'aplica la definició de consolidació de l'interval de temps superior aconseguint la consolidació de les dades de l'interval complet. Això permet tenir valors per intervals de minuts fins consolidats per anys.

També veurem altres bases de dades de sèries de temps que incorporen noves característiques com la consulta mitjançant SQL de les dades emmagatzemades.

1.5.3 Representar la informació

Les dades emmagatzemades en intervals de temps poden ser consumides per diferents sistemes gestors de bases de dades i representar-les en forma gràfica.

Aquests visualitzadors de dades han anat evolucionant inicialment gestionant les dades adquirides pels agents fins a convertir-se en complexos quadres de decisió.

Els quadres d'informació ja no són exclusivament expositors de mètriques. Aquests han evolucionat per incorporar funcions d'intel·ligència de negoci i que utilitzant la potència del Big Data poden consolidar informació proporcionada des de diferents orígens de dades.

Capítol 2
Recol·lectar dades

Els sistemes de monitoratge poder utilitzar múltiples protocols per la recopilació de les dades i serà fonamental escollir correctament l'eina segons el tipus d'explotació de la informació que volem fer. No és el mateix tenir un gran volum de dades per generar gràfiques que saber una dada particular en un moment puntual per a decidir si el funcionament és el correcte o no.

Aquests tipus de consultes estan basades en la recol·lecció de dades al llarg del temps dels elements sota supervisió i desaran les dades en magatzems que mantenen aquesta serialització temporal per a després la seva explotació. En molts dels casos aquesta consulta es realitza contra un agent instal·lat a l'element consultat. Per contra, la vigilància en un moment concret es produeix utilitzant el mateix protocol que ofereix el servei per tal que la resposta sigui idèntica a la que obtindria una petició real i validar el seu correcte funcionament.

Per a la supervisió dels elements sota supervisió s'utilitzen protocols molt simples que generalment incorporen poques habilitats, consulta i escriptura, tot i que amb el temps han evolucionat per incorporar noves funcions com la seguretat en les comunicacions. Aquesta simplicitat en el protocol és complementada amb com s'organitza la informació.

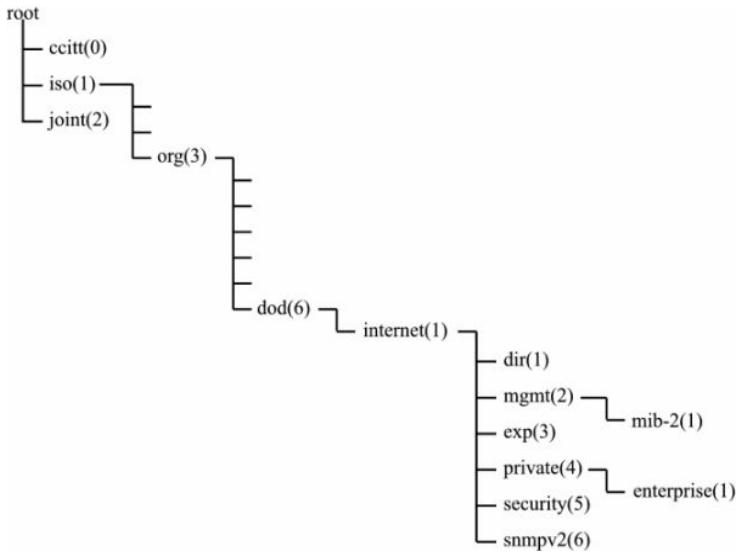
La recol·lecció de les dades intervenen dos elements, l'agent disposador i el client recol·lector. El primer serà l'encarregat de fer disponible i jerarquitzar la informació del dispositiu a monitorar i el segon serà l'encarregat d'accedir a la informació disposada i incorporar-la a les bases de dades per la seva posterior explotació.

És important no caure en l'error de vigilar un servei realitzant consultes sobre les dades recol·lectades. Aquestes dades recol·lectades és la informació d'un servei en un moment determinat però en el moment de la consulta aquesta dada pot estar "caducada" per determinar si s'està produint un error i generar un esdeveniment d'alerta incorrecte.

Un altre aspecte per evitar aquest tipus de vigilància és que les bases de dades que mantenen les dades recol·lectades al llarg del temps es converteixen en un punt únic d'error i mantenir la dependència del servei segons la disponibilitat de la base de dades tampoc és el més adient ja podria generar falsos positius per què no s'ha pogut accedir a la informació.

2.1 SNMP

SNMP és l'acrònim de Simple Network Management Protocol. L'estàndard SNMP defineix un marc per a la gestió de la informació de xarxa i un protocol per a l'intercanvi de la mateixa. [2]
Es tracta d'un protocol que funciona sobre UDP i per tant SNMP és un protocol no orientat a la connexió i minimitza la congestió a la xarxa.



MIB *Management Information Base***.** Definició d'objectes que constitueixen la informació gestionada a l'element. Les taules de variables es defineixen com bases de dades d'informació d'administració organitzades en forma d'arbre. L'arrel de l'arbre és anònima i les branques estan definits per a diferents organitzacions de manera estàndard. Cada objecte administrat a la jerarquia MIB té el seu propi OID *(object identifier)* especificat a l'estàndard ASN1. Aquests objectes principalment estan relacionats amb la informació del maquinari de l'element de xarxa, rendiments i paràmetres de configuració.

Gestor. És el sistema de monitoratge. Pot gestionar les diverses consultes que vinguin dels diferents elements de la xarxa.

Agent. Part del programari que gestiona la informació local de l'element. Gestiona les peticions sol·licitades des de l'estació de gestió.

2.1.1 Tipus de dades

L'estructura de dades d'informació *(Structure of Management Information)* defineix el tipus de dades que es poden utilitzar on s'especifica com es poden representar i nomenar els recursos per tal de garantir que la informació de gestió es representa d'una manera consistent.

Dins l'estructura de dades els objectes MIB es defineixen segons unes regles sobre la base d'un llenguatge de descripció que s'anomena sintaxi de notació abstracta Abstract Syntax Notation ISO 1 (ASN.1).

La versió 1 que forma part del protocol SNMP versió 1 està definit al RFC 1155. La versió 2 queda definida al RFC 1442 i finalment és actualitzada al RFC 2578 considerant-la la versió 3 de SNMP [3]. Cadascun dels objectes identificadors queda definit sota un dels següents tipus de dades. Per definir els objectes de la MIB-I s'utilitza la RFC 1155 i per a la MIB-II la RFC 1212 que inclou més informació.

Sencer *(Integer)***.** Valor enter comprés entre -214748648 i 2147483647

Cadena *(String)***.** Cadena de fins a 65535 caràcters

Nul *(NULL)***.** Valor nul

Identificador d'objecte *(Object Identifier, OID)***.** Identificador d'objecte únic. És definit per una seqüència d'enters, anomenats subidentificadors, separats per punts, 1.2.3

Seqüencia, Seqüencia-de *(Sequence, Sequence-of)***.** Construcció de taules com a seqüències de tipus primaris.

Direcció de xarxa *(Network Address)***.** Permet escollir el format d'adreça. Actualment sols està definit el format IpAddress.

Direcció IP *(IpAddress)***.** Adreça de 32 bits utilitzant el format especificat a IP.

Contador *(Counter)*. Sencer no negatiu que pot incrementar però no decréixer. El seu valor màxim és 2^{32}-1. Quan s'assoleix el valor màxim el comptador comença a comptar des de 0.

Mida *(Gauge)*. Sencer no negatiu que pot incrementar o decréixer. El seu valor màxim és 2^{32}-1. Quan s'assoleix el valor màxim el comptador manté aquest valor fins que no és iniciat.

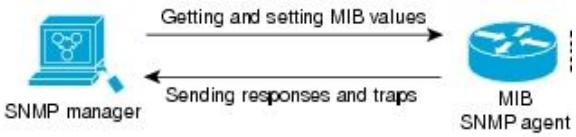
Cops de temps *(TimeTicks)*. Sencer no negatiu que compta el temps en centèsimes de segon.

Comodí *(Opaque)*. Té la capacitat d'ubicar qualsevol tipus de dada, actua com un comodí. Format cadena.

2.1.2 Operacions

El protocol SNMP proporciona les operacions que els client utilitzen per sol·licitar una MIB i poder interactuar. 5 primitives són la base de tota la gestió SNMP.

Totes les operacions SNMP incorporen un paràmetre que s'utilitza per transportar informació de gestió.



get. Un gestor utilitza una operació get per recuperar informació d'un agent. A part d'un identificador de la sol·licitud la petició inclou com a paràmetre una llista de variables vinculades que defineixen quins objectes són sol·licitats, tot i que en aquest cas té valor NULL.

```
$ snmpget 192.168.1.100 -v 1 -c public iso.3.6.1.2.1.1.5.0
iso.3.6.1.2.1.1.5.0 = STRING: "serverWWW.tfc.local"
```

getNext. Aquesta operació és utilitzada per recuperar informació des d'un agent, igual que es fa amb l'operació get, però al contrari del que passa amb el get, els OIDs de les variables vinculades no defineixen els objectes que es recuperen directament. Per a cada OID definit a l'operació, l'agent respondrà amb el següent immediat. De gran utilitat quan el gestor no coneix el OID concret. L'operació permet recórrer l'arbre, que en alguns casos les entrades és creant i es destrueixen de manera dinàmica, passant de manera consecutiva entre els OIDs fins que l'objecte retornat no formi part de la taula demanada, significarà que s'ha arribat al final de la taula.

set. El gestor utilitza l'operació set per escriure a la MIB, per establir un valor a un objecte. L'estructura de l'operació és exactament igual al get i el getNext amb l'única diferència que el valor de la variable vinculada no és nul. L'ús més habitual, és relatiu a canviar la manera en la qual un dispositiu és configurat. Un altre ús, seria la possibilitat de crear o esborrar objectes en una MIB.

getResponse. Un agent respon amb l'operació getResponse a un gestor com a resposta a una petició. Aquesta operació és la resposta per a totes les anteriors operacions, get, getNext i set. Aquesta operació inclou l'identificador, un flag on indica si l'operació de la qual és resposta ha tingut èxit o no, un index d'error que conté informació addicional i una llista de variables vinculades.

trap. Un trap és l'enviament d'un esdeveniment per part de l'agent a un gestor. En aquest cas, el gestor no respon cap confirmació.

2.1.3 SNMP v2

A la versió 2 es realitza una revisió sobre la versió 1 i inclouen millores en àrees de rendiment, seguretat, confidencialitat i comunicacions entre gestors, tot i que l'apartat de seguretat no va quedar totalment acceptat. Com a resultat d'aquesta falta d'acceptació, es van realitzar dos subversions, versió 2c (community version) i versió 2u (user version)

2.1.3.1 SNMP v2u

Aquesta versió intenta oferir una millor seguretat que la versió 1 però evitant la complexitat de la definició de la v2.

2.1.3.2 SNMP v2c

Aquesta nova versió, elimina el desenvolupament en l'àrea de seguretat de la versió 2 i manté l'esquema de la versió 1 que utilitza noms de comunitat per diferenciar les accions de lectura i escriptura. El problema que sorgeix amb aquesta versió és que modifica la capçalera del missatge SNMP fent que la comptabilitat amb la versió 1 quedi trencada.

Incorpora dues noves operacions, getBulk i inform i també modifica la mida pels tipus de dades comptadors que permeten una definició de valors fins a 64 bits.

getBulk. Amb l'operació getNext es retorna l'objecte OID següent a l'especificat a la variable vinculada. Aquesta nova operació resol la necessitat de la recursivitat, ja que permet al gestor recuperar grans quantitats d'informació amb una sola petició. S'ha d'afegir quants OIDs es vol que respongui l'agent següents al demanat.

inform. És una operació evolucionada del trap on es demana confirmació de recepció. Afegint d'aquesta manera un plus de fiabilitat a l'operació trap. Aquest reconeixement de recepció es produeix mitjançant la PDU de resposta que es produeix en resposta a la següent operació. Això incorpora una complexitat per mantenir una taula amb les operacions inform no confirmades per part de l'agent.

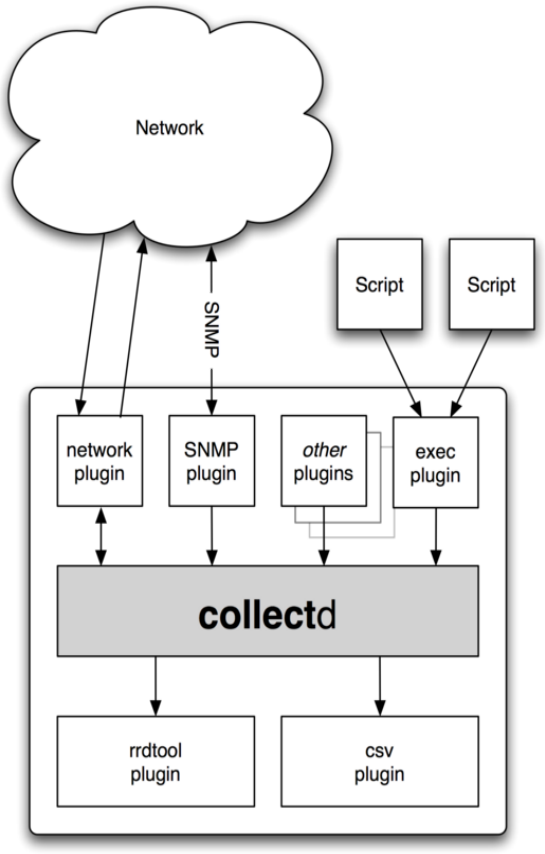
2.1.4 SNMP v3

En essència aquesta versió 3 es pot considerar que és SNMP v2c amb la seguretat millorada. El protocol manté les mateixes operacions de gestió però introdueix canvis en la missatgeria per incorporar funcionalitats de seguretat que fan el protocol SNMP segur. Aquests valors permeten el xifrat dels missatges i autenticació dels remitents, això implica que quan un agent rep una petició des d'un gestor aquest pot determinar que es tracta d'un gestor autoritat i que el missatge no ha estat alterat.

2.2 collectd

Collectd es podria definir com la "navalla Suïssa" de les eines de monitoratge. Es tracta d'un dimoni escrit en C, lliure i al qual es poden afegir mòduls mitjançant extensions per ampliar les seves funcions. [4]

Aquest dimoni pot recol·lectar les dades de rendiment del sistema de manera periòdica al qual se li poden incorporant mòduls de publicació, per emmagatzemar els valors de diferents formats, incloent-hi RRD, i afegir la capacitat d'activar esdeveniments d'alerta.



Una altra característica destacada seria que el dimoni collectd pot funcionar de manera individual o com a concentrador d'altres dimonis collectd.

El potencial de collectd radica en la seva versatilitat per afegir complements fent que el dimoni tingui una cobertura i integració amb qualsevol element. Existeix una àmplia documentació per crear nous complements i ampliar així la immensa llista ja disponible [5]

Aquests complements els podem classificar per, complements de lectura, complements d'entrada/sortida, complements basats en altres protocols i complements de filtratge.

A continuació es pot observar com quedaria configurat el dimoni per adquirir les dades de CPU, memòria i xarxa.

arxiu: collectd.conf

```
LoadPlugin “cpu”
LoadPlugin “memory”
LoadPlugin “interface”

<Plugin interface>
          Interface lo
          Interface eth1
</Plugin>
```

A diferència amb SNMP, el dimoni collectd fa la recol·lecció de les dades en intervals de 300 segons i per petició des d'un client.

Durant aquests intervals, collectd recupera la informació indicada als arxius de configuració i les desa utilitzant els complements de sortida que tingui definits, ja sigui de manera local amb bases de dades RRD o a un host remot utilitzant un altre protocol.

En el següent exemple es defineix que les dades quedin disponibles a una base de dades mongoDB.

arxiu: collectd.conf

```
<Plugin "write_mongodb">
  <Node "default">
    Host       "mongoDB.tfc.local"
    Port       "27017"
    Timeout    2000
    StoreRates true
  </Node>
</Plugin>
```

2.3 NetData

Es tracta d'un dimoni escrit en C que proveeix l'estat en temps real de sistemes Linux, aplicacions, dispositius SNMP i les disposa a una web.

Proporciona un entorn que consumeix poca CPU i poc consum de memòria RAM. També permet afegir nous mòduls per ampliar les capacitats de recol·lecció per a serveis específics [6]

Com també passava amb collectd la recol·lecció de les dades es fa a intervals i segons els autors es pot definir un interval d'1 segon sense una gran exigència al sistema per oferir un monitoratge en temps real.

Una de les diferències més destacades amb collectd és que incorpora de sèrie, sense afegir cap mòdul, una API per accedir les dades i consultar-les des de servidors remots.

A continuació un exemple de com obtenir el llistat de les mètriques que té l'element supervisat

```
$ curl serverWWW.tfc.local:19999/api/v1/charts
{
    "hostname": "serverWWW.tfc.local",
    "update_every": 1,
    "history": 172800,
    "charts": {
      "named_local.view_resolver_numfetch__default":       {
            "id": "named_local.view_resolver_numfetch__default",
            "name": "named_local.view_resolver_numfetch__default",
            "type": "named_local",
            "family": "view__default",
            "context": "named.resolver.active.queries",
            "title": "local Bind, _default View, Resolver Active
Queries (named_local.view_resolver_numfetch__default)",
            "priority": 61001,
            "enabled": true,
            "units": "queries",
            "data_url":
"/api/v1/data?chart=named_local.view_resolver_numfetch__default",
            "chart_type": "line",
            "duration": 172800,
            "first_entry": 1465642895,
            "last_entry": 1465815695,
            "update_every": 1,
```

```
            "dimensions": {
                  "queries": { "name": "queries" }
            }
      },

[…]

[…]

}
```

Consultar la ultima dada definint una mètrica

```
$ curl  serverWWW.tfc.local:19999/api/v1/data?chart=system.cpu&before=-1
{
    "labels": ["time", "guest_nice", "guest", "steal", "softirq",
"irq", "user", "system", "nice", "iowait"],
    "data":
    [
      [ new Date(2016,6,9,13,57,26), 0, 0, 0, 0.00322, 0, 0.49838,
5.02408, 48.00567, 0]
    ]
```

2.4 Nagios plugins

A diferència dels dos anteriors exemples els Nagios plugins no són dimonis que periòdicament recol·lecten les dades per desar-les en els magatzems o fer-les accessibles des d'una web. Són un grup d'utilitats que ens permeten consultar en el moment de la seva execució l'estat del servei i ho realitza utilitzant el mateix protocol del servei vigilat.

Actualment hi ha en el repositori oficial més de 4000 checks disponibles [7] però seguint la guia de desenvolupament que el mateix creador de Nagios ha establert es poden crear de nous.

Com veurem més endavant en aquest projecte utilitzarem aquest guia per crear un check adhoc per vigilar un servei important a la nostra plataforma.

2.4.1 check_tcp

Amb check_tcp podem fer consultes directament sobre un port o un unix socket a un servidor remot.

```
$ ./check_tcp -H www.uoc.edu -p 80
TCP OK - 0.033 second response time on port 80| time=0.033s;0.000; 10.000

$ ./check_tcp -H www.uoc.edu -p 443
TCP OK - 0.062 second response time on port 443| time=0.062s;;0.000; 10.000
```

Com es pot apreciar als exemples anteriors s'ha fet una consulta a la URL www.uoc.edu als ports 80 i 443. Això indica que hi ha un servei escoltant a aquests ports però no implica de manera fiable que el servei que hi ha escoltant sigui un servidor web.

2.4.2 check_http

Aquest check és encara més específic que l'anterior, ja que no solament fa una connexió a una URL i un port determinat, també utilitza el protocol HTTP per enviar una comanda GET.

```
$ ./check_http -H www.uoc.edu
HTTP OK: HTTP/1.1 200 OK - 4972 bytes in 0.113 second response time |
time=0.112s;;0.000 size=4972B;;;0
```

```
$ ./check_http -H www.uoc.edu -r 'Universitat Oberta de Catalunya'
HTTP OK: HTTP/1.1 200 OK - 4973 bytes in 0.106 second response time
|time=0.105s;;;0.000 size=4973B;;;0

$ ./check_http -H www.uoc.edu -r 'universitat oberta de catalunya'
HTTP CRITICAL: HTTP/1.1 200 OK - pattern not found - 4972 bytes in 0.120
second response time |time=0.120s;;;0.000 size=4972B;;;0
```

2.4.3 Nagios Remote Plugin Executor

Nagios té la possibilitat d'executar la petició d'un check de manera remota per aquells elements vigilats que no formen part d'un servei amb un protocol que es pugui utilitzar en remot. Aquest agent que executa els checks de manera local a l'equip vigilat habilita un port de comunicació un protocol de comunicació propi i xifrat que a accedir al resultat d'aquests checks.

Els checks sobre serveis publicats per exemple servidors web, bases de dades o servidors de correu es realitzen sobre el servei però per verificar l'espai lliure d'una partició de disc del servidor sota vigilància no es pot realitzar per això aquest complement és de gran utilitat.

[A serverWWW.tfc.local] arxiu: nrpe_local.conf

```
server_port=5666
allowed_hosts=127.0.0.1,IP_Nagios_server

command[sda1]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10% -p /dev/sda1
```

Llavors per consultar l'espai de sda1 d'aquest equip client

```
$ ./check_nrpe -H serverWWW.tfc.local -c 'sda1'
DISK OK - free space: / 1487 MB (26% inode=55%);| /=4107MB;4733;5325;0;5917
```

2.5 Resum

Al llarg del capítol s'han descrit les dues formes de disposar les dades, sota demanda i per intervals. Aquests dos mètodes són d'utilitat segons el tractament que volem realitzar de les dades si el que volem és establir regles per assegurar que el servei s'està oferint tal com es va pensar al llarg del temps o volem saber en un moment donat si funciona o no.

Sota petició és la consulta tal qual del servei, sense intermediari i a un moment donat i reaccionar segons el tipus de resposta rebuda.

Per intervals no es necessita cap intervenció perquè es tracta d'un servei programat per executar-se cada cert temps i queda reservat a la recol·lecció de les dades per inserir-les a un magatzem per la seva posterior explotació.

Capítol 3
Magatzems de dades

Els magatzems de dades és l'element de l'estructura del sistema de monitoratge on restaran emmagatzemades les dades que han estat recol·lectades per els recol·lectors de dades i que posteriorment seran tractades per generar gràfiques, panells de representació de dades i informes. Aquestes bases de dades tenen una característica principal i es que es tracten de sistemes gestors d'informació que emmagatzemen les dades per sèries de temps.

Una base de dades de sèries de temps és un sistema que està optimitzat per manipular sèries de dades, grups de dades indexades per marques temps.

3.1 Round-Robin Database

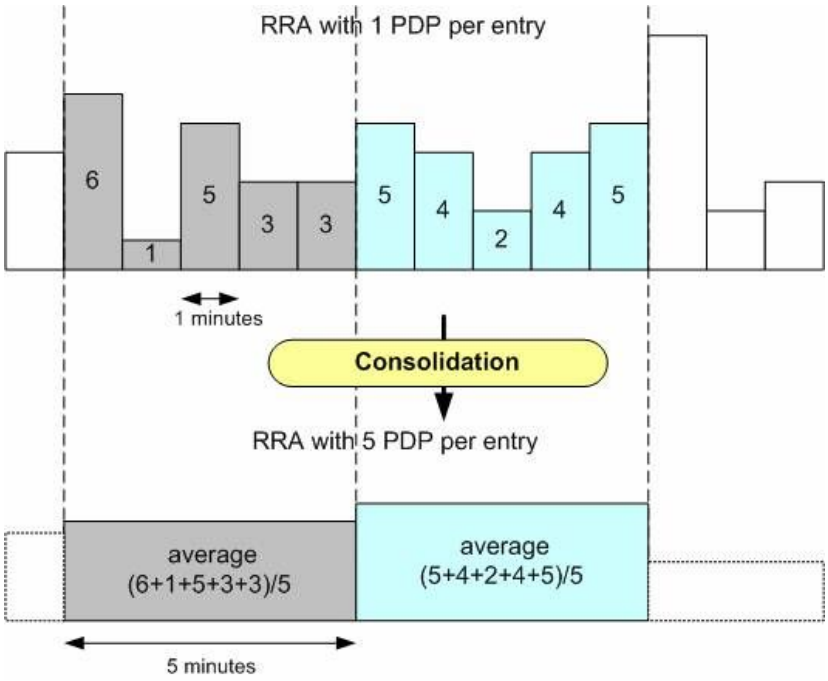
RRD es tracta d'un projecte nascut l'any 1999 i que en 2015 es va publicar la seva última versió estable. Es tracta d'un dels projectes opensource consolidat com a estàndard per a l'administració de dades de rendiment i per a la generació de gràfiques.

Round Robin Database o RRD, és un sistema dissenyat per gestionar dades en series per temps, com podria ser l'amplada

de banda d'una connexió de xarxa, CPU o la càrrega del sistema [8]. La informació està emmagatzemada de manera circular on la marca de temps es manté constant durant el temps.

La definició del magatzem de les dades circulars es fa d'acord amb la definició d'intervals de temps amb una certa mida i que quan és completa torna a començar. Els arxius històrics de les dades per períodes de temps són consolidacions basades en algoritmes aplicats (màxim, mínim, mitjana, darrer valor) a les dades d'intervals anteriors.

Es tracta d'una eina amb molt de recorregut, senzilla d'implementar, compatible amb molts llenguatges de programació i suportat per molts generadors de gràfiques.



3.1.1 Creació arxiu RRD

Per la creació d'un arxiu utilitzarem la comanda `rrdtool create`

```
$ rrdtool create sensors.rrd --step 30 \
DS:sensor1:GAUGE:60:0:U \
DS:sensor2:GAUGE:60:0:U \
RRA:AVERAGE:0.5:2:2880 \
RRA:AVERAGE:0.5:10:20160 \
RRA:MAX:0.5:120:80640 \
RRA:MIN:0.5:120:80640
```

--step 30 indica que les dades s'actualitzen cada 30 segons. Dues dades per minut
DS:sensorX:GAUGE:60:0:U és la primera font de dades, *Data Source*

sensorX és el nom de la font

GAUGE tipus de dada, els valors són tractats de manera independent, no incremental

60 és el paràmetre de *heartbeat*, que és el temps màxim entre actualitzacions abans que s'assumeixi com a dada desconeguda. Habitualment es el doble del temps definit a *step*

0 valor mínim que pot tenir aquesta font

U valor màxim que pot tenir la font, en aquest cas el màxim és un valor desconegut.

RRA:AVERAGE:0.5:2:2880

RRA round robin archive. Magatzem de dades consolidades

AVERAGE tipus d'algoritme de consolidat

0.5 nombre de dades que es permet sense valor per consolidar (50%)

2 volum d'intervals utilitzats per crear un període de consolidat

2880 nombre total d'intervals per completar el període (1 dia)

En aquest cas, el RRA serà un històric de dades amb les mitjanes de cada minut (2 intervals) durant un dia (2 intervals * 60 minuts * 24 hores)

RRA:AVERAGE:0.5:10:20160

Aquest RRA serà un consolidat de dades amb les mitjanes cada 5 minuts (10 intervals) per a una setmana (10 intervals * 12 minuts * 24 hores * 7 dies)

RRA:MAX:0.5:120:80640 i RRA:MIN:0.5:120:80640

Aquest seran dos consolidats utilitzant el valor màxim i mínim dels intervals de cada hora (120 intervals) durant els últims 28 dies (120 intervals * 24 hores * 7 dies * 4 setmanes)

3.1.2 Afegir dades

Una vegada tenim creat la base de dades de sèries de temps, periòdicament (definit a l'interval) s'hauran d'introduir les dades.

```
$ rrdtool update sensors.rrd N:0.64:0.82
```

0.64 es el valor per el sensor1
0.82 es el valor per el sensor2

3.2 InfluxDB

InfluxDB és una base de dades de sèries de temps que pot gestionar gran volum de lectures i escriptures. Forma part del grup d'eines de monitoratge desenvolupada per InfluxData (telegraf, influxdb, chronograf i kapacitor) i està dissenyat per a ser utilitzat com a repositori de

les dades recol·lectades per diferents agents recol·lectors compatibles amb la seva arquitectura. [9]

Algunes de les seves característiques que podem destacar són:

- És un magatzem optimitzat per oferir un alt rendiment per emmagatzemar dades en series de temps. Està dissenyat per permetre una rapida ingesta i compressió d'un gran volum de dades.
- Un únic binari, sense dependències.
- Les consultes i escriptures es realitzen mitjançant API Rest.
- Permet l'escalat amb InfluxDB relay
- Ús de llenguatge SQL per consultar les dades consolidades.
- Etiquetatge de sèries de temps per una indexació eficaç i rapida.
- Auto generació de dades consolidades sobre les consultes més freqüents
- Panell d'administració web

3.2.1 Creació base de dades

InfluxDB és un gestor de base de dades sense esquemes, *schemaless*. Això significa que no és necessari definir taules amb tipus de dades per a cada camp que es vol emmagatzemar i el contingut de cada base de dades seran objectes.

```
$ curl -POST http://influxdb.tfc.local:8086/query \
--data-urlencode \
"q=CREATE DATABASE metrics"
```

3.2.2 Afegir dades

A l'igual que es fa amb la creació de la base de dades la inserció de les dades es pot realitzar utilitzant l'API REST que publica la base de dades

```
$ curl -i -XPOST 'http://influxdb.tfc.local:8086/write?db=metrics' \
--data-binary 'sensor1,espai=sala1,planta=1 value=0.64'
```

-XPOST mètode de connexió
http://servidor_influxdb:8086 URL i port on està publicat la base de dades InfluxDB
write?db=metrics operació d'escriure sobre la base de dades metrics
sensor1 nom del camp
espai=sala1,planta=1 etiquetes descriptives del camp
value=0.64 valor en aquest moment del sensor1

3.2.3 Accés a les dades

A diferencia de RRD la consulta de les dades és possible ja que disposa d'un intèrpret de llenguatge SQL que permet realitzar consultes sobres les dades emmagatzemades mitjançant el API REST

```
$ curl -GET 'http://influxdb.tfc.local:8086/query?pretty=true' \
--data-urlencode "db=metrics" --data-urlencode \
"q=SELECT value FROM sensor1 WHERE espai='sala1' AND planta='1'"
```

-GET mètode de connexió
http://servidor_influxdb:8086 URL i port on està publicat la base de dades InfluxDB
query?pretty=true Tipus d'operació, en aquest cas una consulta i s'indica que la presentació de les dades es realitzi en format JSON
--data-urlencode "db=metrics" Sobre quina base de dades es realitza la consulta
q=SELECT value FROM sensor1 WHERE espai='sala1' AND planta='1' la consulta a realitzar

i s'obté una sortida tipus:

```
{
    "results": [
        {
            "series": [
                {
                    "name": "sensor1",
                    "columns": [
                        "time",
                        "value"
                    ],
                    "values": [
                        [
                            "2016-05-26T21:55:43.702900257Z",
                            0.59
                        ],
                        [
                            "2016-05-26T21:56:43.102350654Z",
                            0.61
                        ],
                        [
                            "2016-05-26T21:57:43.884511253Z",
                            0.64
                        ],
```

```
                        [
                            "2016-05-26T21:58:43.728576112Z",
                            0.62
                        ],
                        [
                            "2016-05-26T21:59:43.218523496Z",
                            0.57
                        ]
                    ]
                }
            ]
        }
    ]
}
```

3.3 Resum

De magatzems de dades amb sèries de temps existeixen un gran ventall de solucions, totes diferents i totes iguals, on unes estan basades en arxius altres estan basades en motors NOSQL per desar les dades. En definitiva l'elecció de la base de dades estarà condicionat amb els altres dos elements que formen el sistema de monitoratge, el recol·lector de dades i el visualitzador.

Existeixen un gran nombre de mòduls per fer compatibles recol·lectors amb les diferents bases de dades però sens dubte és millor l'opció dels agents recol·lectors que porten de forma nativa la integració amb el magatzem.

RRD per la seva simplicitat, permet un ràpid desplegament però com a part negativa aquesta mateixa simplicitat limita la modificació dels intervals de recol·lecció. InfluxDB, ElasticSearch, OpenTSDB,... es tracten de sistemes més complexos d'implementar però que al mateix temps es facilitarà la tasca de modificacions posteriors donada l'arquitectura flexible de què disposen.

Com ja s'havia vist al capítol anterior, aquesta part del monitoratge és exclusiva i necessària per a la generació de gràfiques sobre els index dels quals volem disposar d'un històric per decidir si l'evolució ha estat l'esperada o no.

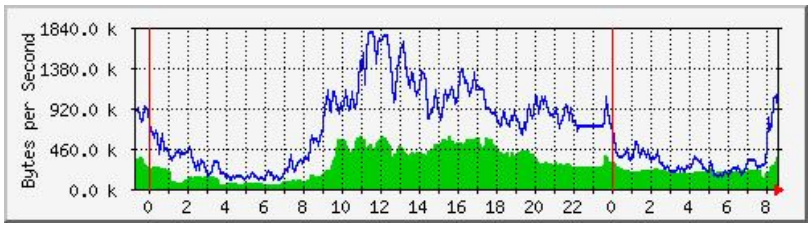
Capítol 4
Representació de les dades

Veure l'evolució durant el temps de les mètriques és una gran utilitat per prendre les decisions necessàries i detectar errors o funcionaments incorrectes que amb consultes puntuals difícilment es poden detectar.

4.1 Multi Router Traffic Grapher

Multi Router Traffic Grapher o MRTG es tracta d'una eina desenvolupada per Tobias Oetiker, com ho és RRDtool, que permet monitorar la càrrega de trafic a les interfícies de xarxa. MRTG genera pàgines HTML des dels arxius RRD i aquestes contenen imatges en format PNG que són una representació visual d'aquesta amplada de banda. [10]

MRTG consisteix en un programa escrit en Perl que utilitza els valors dels OIDs específics de les interfícies llegides mitjançant SNMP dels equips de xarxa i en un programa escrit en C que crea les gràfiques que representen l'amplada de banda de la connexió monitorada. Aquestes gràfiques queden incrustades a pàgines web i és poder visualitzar des del navegador.



MRTG però no està únicament limitat a fer gràfiques sobre les interfícies de xarxa, qualsevol OID adquirida amb SNMP pot ser representada. És habitual utilitzar les gràfiques per representar les dades de càrrega del sistema, nombre de sessions, ...

4.2 Grafana

Grafana es tracta d'un *dashboard* o panell per a gràfiques i mètriques molt versàtil. No forma part de cap *stack* d'aplicacions i és compatible amb diverses fonts de dades.

Incopora un aspecte visual més atractiu que les gràfiques generades amb MRTG però la seva instal·lació i configuració és més complexe. [11]

4.2.1 Estructura

Fonts d'informació: Grafana per a cada magatzem de dades té un joc específic de consultes que estan dissenyades per les seves característiques i capacitats. InfluxDB, Graphite, ElasticSearch, OpenTSDB o KairosDB són alguns dels magatzems de dades compatibles amb Grafana.

Gràfiques: És la imatge gràfica de les dades obtingudes de la font de dades. Cada gràfica té com a origen una font de dades amb una consulta específica. A part Grafana permet aplicar diferents estils i formats a les gràfiques de forma particular segons les dades representades. Certes unitats gràfiques d'informació necessiten que les dades siguin consolidades abans per adaptar-se a l'estil seleccionat.

Organitzacions: Al mateix temps Grafana pot gestionar múltiples organitzacions on cadascuna disposarà de les seves fonts de dades i gràfiques associades.

Usuaris: Incorpora la gestió d'usuaris associats a cadascuna de les organitzacions, als quals se'ls poden definir diferents nivells d'administració i accés..

Editors de consultes: L'editor de consultes incorporat permet realitzar peticions sobre les dades de les mètriques emmagatzemades. Es poden construir una o múltiples consultes sobre les sèries de dades de temps i actualitzar en temps real la imatge de la gràfica dependent de les consultes. També és possible crear consultes derivades d'altres gràfiques amb una simple referència a la gràfica.

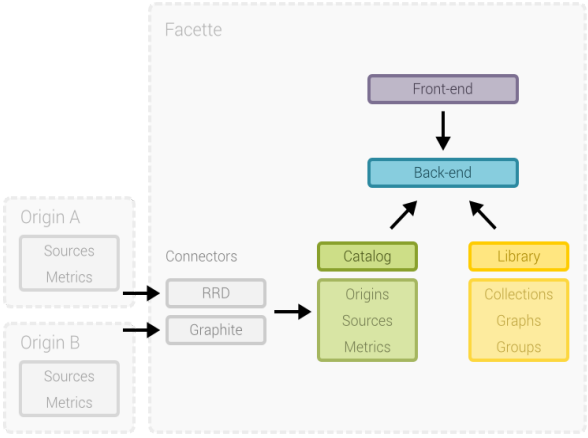
Dashboards: Els panells seran els encarregats d'organitzar i centralitzar totes les gràfiques d'una organització. Poden ser organitzats per subgrups segons la temàtica de les gràfiques. Aquests permeten l'exportació de les dades, compartir-les i afegir anotacions per a facilitar la recerca.

4.3 Facette

Aquest projecte, tot i que indica que és un projecte encara no prou madur per utilitzar-se en entorns productius, és interessant per la seva simplicitat de gestió i per a com és estructurat internament el seu funcionament. [12]

Facette igual que els anteriors és un programari de visualització on les seves fonts poden ser diverses, desde arxius locals fins a consultes a bases de dades remotes.

Com es veu a la imatge, està format per un backend i un frontend. El primer és l'encarregat d'interactuar amb les fonts de les dades, els orígens i el frontend és una aplicació web per dibuixar les mètriques.



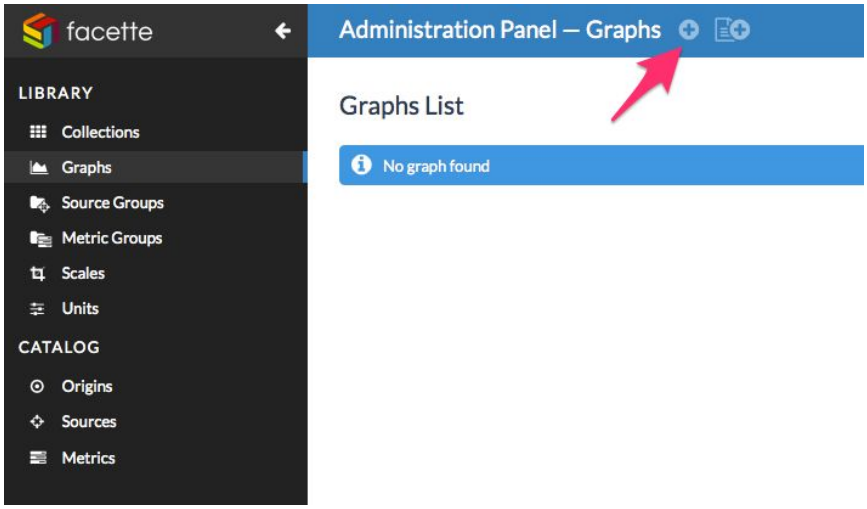
Manté un catàleg intern amb l'inventari de les dades i els magatzems de les dades. Aquests orígens poden ser molt diversos, arxius RRD, bases de dades o recol·lectors tipus collectd.

La configuració de Facette es realitza sobre un arxiu amb format JSON on es definiran els diferents connectors a les dades.

arxiu: connectors.json

```
{
  "connector": {
    "type": "rrd",
    "path": "/var/lib/collectd/rrd",
    "pattern": "(?P<source>[^/]+)/(?P<metric>.+).rrd"
  }
}
```

Dins de l'administrador web, es poden definir les gràfiques disponibles segons els connectors anteriorment configurats.



4.4 Resum

A l'igual que com passava amb els recol·lectors de dades i les bases de dades de sèries de temps de generadors i visualitzadors de mètriques també tenim un bon grup d'opcions on escollir. És important tenir en compte a l'hora d'escollir la nostra eina de representació de les dades que aquesta sigui compatible amb un gran nombre de fonts de dades i no estigui limitada a una d'elles.

Els sistemes més complexos incorporen mòduls d'integració amb fonts de dades molt diverses, per una part ens resultarà interessant davant la generació de mètriques objectiu principal pel qual destaquem aquestes utilitats, i també els podrem utilitzar per integrar fonts de dades que ens permetrà la visualització de logs i grans volums de dades consolidades per la utilització dels diferents serveis de la plataforma.

Capítol 5
Eines i recursos

Les plataformes de gestió són grans aplicacions que incorporen tot en un els tres elements necessaris per al monitoratge, un recol·lector de dades, un magatzem i un visualitzador de les mètriques. També incorporen diferents característiques que permeten una personalització com l'administració d'usuaris i grups, compartir mètriques, control d'accés, còpies de seguretat o la configuració a nivell d'estil

Com s'ha detallat als anteriors capítols, molts dels creadors desenvolupen tot el *stack* d'aplicacions per a crear una eina global de monitoratge.

5.1 NAGIOS

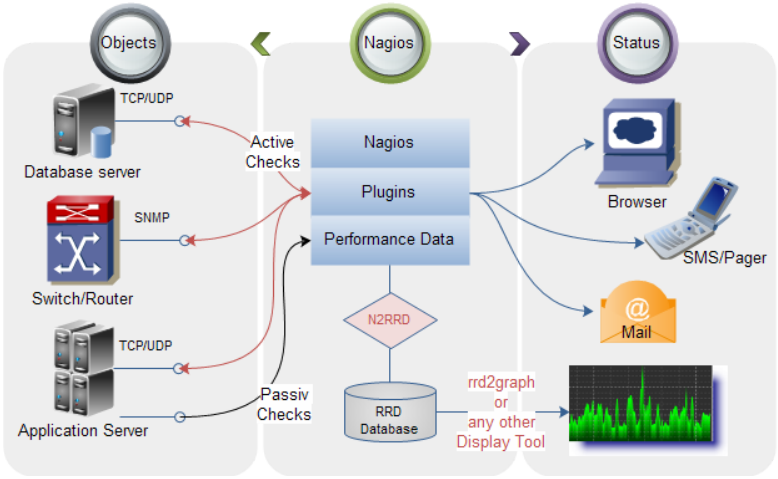
NAGIOS és codi obert, lliure, dissenyat per funcionar sobre sistemes Linux i és un sistema de monitoratge desenvolupat per Ethan Gasltad. Té la possibilitat de monitorar des de l'estat de servidors fins al rendiment d'un servei donat, utilitzant el seu conjunt d'utilitats incorporades o creades adhoc activant en cas necessari les polítiques de notificació.

La seva estructura de funcionament intern es pot dividir en tres grans blocs: el nucli gestor, les extensions o utilitats i lloc web o frontend.

El **nucli gestor** conté el programari necessari que gestiona el monitoratge, això inclou el control de processos, la gestió de supervisió sobre els servidors i els serveis.

Extensions, utilitats *(plugins)*. Les extensions incorporades es tracten de binaris que en ser executats realitzen un check específic sobre el servei indicat. Es poden desenvolupar extensions a mida perquè NAGIOS disposa d'una guia de desenvolupament que permet una fàcil integració al conjunt d'utilitats.

Lloc web. La web de NAGIOS es pot al mateix temps en dividir en dos grups: la part visual frontend desenvolupada amb HTML i CSS i els scripts de gestió que són scripts CGI. El lloc web és un visualitzador dels resultats de les checks realitzats i ofereix a l'administrador un control visual dels components.



Nagios incorpora de base un gran nombre d'utilitats que estan pensades i desenvolupades per efectuar els checks necessaris sobre els serveis que tenim distribuïts a la plataforma. Dintre d'aquestes podem trobar plugins per monitorar els protocols HTTP, HTTPS, PING, POP3, TCP, UDP, DNS, SMTP... com també utilitats específiques que no funcionen oferint un servei com l'espai en disc, ús de memòria, supervisió dels fitxers de log.

Aquestes utilitats que funcionen de forma local tenen la possibilitat de comunicar els resultats mitjançant l'agent NRPE, ja que aquests habilita l'execució de forma remota mitjançant una petició per un canal segur propi de Nagios.

Com s'ha detallat anteriorment, en la part web, incorpora tot un sistema de gestió mitjançant binaris CGIS que permeten planificacions, informar de finestres de manteniment on quedin desactivats temporalment la supervisió dels serveis implicats, desactivar les notificacions sobre els servidors i serveis dels quals puntualment no es vol rebre notificacions.

La configuració de Nagios en els seus arxius de configuració es tracta principalment en definir l'accés al lloc web de l'usuari administrador, les rutes on estaran les definicions dels objectes a monitorar així com les configuracions de quan, com, a qui i quina informació s'ha de notificar en cas de detecció d'errors.

Els sistemes de notificacions més utilitzats són els correus electrònics i els missatges SMS però es poden implementar altres vies de comunicació com tweets, missatges de whatsapp o chats tipus slack.

Com el nombre de servidors i serveis pot suposar una fita difícilment abastable per un únic motor de supervisió s'han incorporat nous agents que permeten que els controls es realitzin de manera distribuïda i quedi centralitzada únicament la part informativa dels resultats.

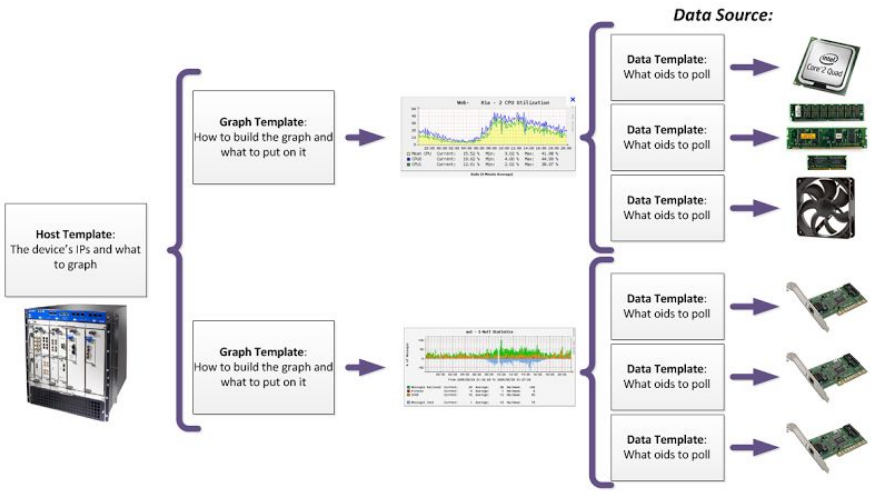
Nagios Remote Plugin Executor (NRPE) és l'agent de NAGIOS que permet afegir checks a un sistema remot utilitzant les extensions o plugins hostatjat al mateix servidor remot. Això permet controlar els recursos locals, com espai en disc, càrrega del sistema, usuaris. El servidor on està NAGIOS instal·lat connectarà amb aquest agent remot per recuperar la informació.

Nagios Remote Data Processor (NRDP) és un agent d'execució local amb un sistema de transport d'informació que utilitza els ports i protocols estàndards HTTP i XML. Està desenvolupat amb una arquitectura que permet que sigui extensible i configurable. Es va implementar com a substitut per NSCA que fou una simulació dels traps del protocol SNMP.

NSClient++ Està desenvolupat per ser un agent monitor a servidors amb sistema operatiu Windows. Funciona similar a l'agent NRPE.

5.2 CACTI

Cacti és codi obert i lliure i es tracta una eina pel monitoratge de xarxa i que incorpora un frontend per mostrar les gràfiques generades amb RRDTool [13]. Les gràfiques més comunes són mètriques com CPU, nombre de processos actius, càrrega del sistema i utilització d'amplada de banda, tot i que es poden implementar d'altres perquè el sistema de recol·lecció és basat en el protocol SNMP.



Igual que Nagios, Cacti el podem dividir en dues parts: la part visual o frontend i la part de gestió o backend.

El frontend es dedica a tota la gestió administrativa de la utilitat, gestionar comptes d'usuari amb accés a les seves gràfiques, afegir nous dispositius per afegir noves recol·leccions de dades per generar les seves mètriques, afegir nous templates amb nous tipus de mètriques, incorpora un sistema de gestió de plugins per afegir noves funcionalitats associades a la visualització de les dades. [14]

El backend és l'encarregat de realitzar de manera periòdica, amb un dels dos motors suportats, cmd desenvolupat en PHP ideal per a petites instal·lacions o spine un motor desenvolupat en C que permet a cada fase de recol·lecció de dades gestionar un gran volum d'aquestes.

5.3 TICK: Telegraf, InfluxDB, Chronograf, Kapacitor

És el conjunt d'aplicacions desenvolupat per la companyia Influx Data [15] i que permet el tractament de les dades en series de temps, des de la recol·lecció, passant pel magatzem, visualitzador o generador d'alertes basats a una lògica.

El recol·lector, anomenat **Telegraf**, té diversos plugins que li permeten adquirir una gran varietat d'informació de sistemes remots i pot adquirir aquestes dades des de APIs, escoltant

les dades que vinguin del dimoni collectd o de serveis Apache Kafka. També disposa de diversos plugins per emmagatzemar aquestes dades en diferents magatzems de dades.

El magatzem de dades, **InfluxDB** explicat al capítol anterior.

Chronograf com a dashboard permet realitzant consultes sobre el magatzem InfluxDB representar les dades creant les seves gràfiques..

Kapacitor és l'eina del stack que ens permet relacionar esdeveniments sota uns patrons establerts, realitzant accions com notificar aquest paràmetre.

5.4 Resum

Existeix una gran quantitat de solucions de sistemes de monitoratge integral que inclouen les tres parts explicades: un sistema de recol·lecció per la supervisió del funcionament amb un sistema de vigilància per detectar els errors, un magatzem de dades indexades per temps i un visualitzador de les dades que tingui la capacitat de generar mètriques processant les dades disponibles.

Per escollir una solució de monitoratge s'ha de tenir present el creixement i la possible evolució de la plataforma per tal que puguem integrar altres funcionalitats disponibles per altres proveïdors de manera senzilla i que no sigui un trasbals en un element tan important com és el sistema de monitoratge.

Capítol 6
Principi del Caos: Netflix i l'exèrcit mico

Es produeixen millores a les plataformes de manera contínua, a escala global cada vegada es creen més plataformes informàtiques enormes, sistemes distribuïts, descentralitzats, canvis a les versions dels programaris. Això és l'efecte de com evoluciona aquest món on s'adopten noves pràctiques, estils, amb un creixement continu i a gran velocitat. I això comporta el fet que els equips es preguntin com són de fiables les seves plataformes davant aquesta voràgine de canvis.

Als capítols anteriors s'han vist eines per a mesurar la capacitat, la latència o la productivitat, però cap d'aquestes eines ens permet assegurar la fiabilitat. I si tenim serveis individuals funcionant sobre sistemes distribuïts que al mateix funcionen correctament i que també connecten amb altres serveis individuals d'altres sistemes distribuïts això pot provocar, que tot i funcionar correctament de manera individual es produeixin respostes inesperades. La incertesa als equips i als sistemes que es poden veure afectats per efectes que escapen del nostre control, pot derivar que un entorn estable es converteixi en un caos.

Vivim a un món imperfecte i acceptar això és el primer pas per construir entorns fiables. La fiabilitat no és la condició que indica que no apareguin errors, significa crear plataformes i entorns que siguin resistents als errors, els errors previsibles i també els imprevisibles. Els equips d'administradors de sistemes treballen per detectar aquests problemes al llarg dels sistemes de tal manera de ficar-hi solució i evitar errors que provoquin un funcionament no

esperat o la caiguda total del servei. Punts únics d'error, temps d'espera infinits, dependències sense administrar, configuracions incorrectes, creixements de trafic, errors en cascada,... són una part dels errors que poden afectar el servei.

La tasca dels administradors, entre d'altres, és manejar aquest caos i mantenir sota control aquest creixement aplicant les polítiques correctes que ens poden donar l'avantatge per incrementar el rendiment i la flexibilitat restant segurs que el servei que s'ofereix tot i la seva complexitat.

L'equip d'enginyers a Netflix d'aplicar el principi del caos [16] li diu l'Enginyeria del Caos. El servei que ofereixen la companyia té com a base arquitectures distribuïdes que mantenen sota vigilància i supervisió, la qual observen com reacciona durant l'aplicació de les seves proves del caos.

Netflix és una empresa Nord-Americana que s'encarrega de la distribució de vídeo online VOD, per fer-nos una idea es tracta d'un antic videoclub on el servei s'ofereix mitjançant Internet i via streaming. Es pot accedir a un ampli catàleg de pel·lícules i series per consumir el producte quan escaigui. Va iniciar el servei l'any 1999 i les últimes dades publicades s'indica que té més de 81 milions de subscriptors.

6.1 Com funciona

Aquestes proves dissenyades per realitzar-les sobre sistemes distribuïts són un recull de pràctiques per detectar els errors no coberts a altres fases del desenvolupament.

Cadascuna d'aquestes proves s'articula en les següents 4 fases:

- Definició de l'estat ideal. Establint quina sortida és la correcta i com mesurar-la
- Establir com fer perquè l'estat ideal continuï funcionant entorn del control i a l'entorn estudiat tot i les incidències patides
- Introduir les variables necessàries per emular el món real, caigudes del sistema, errors del maquinari, errors de connexió, ...
- Extreure els resultats sobre el funcionament del grup de control i el grup estudiat.

Sobre les fallades desconegudes, els entorns estaran dissenyats per a suplir aquesta deficiència de detecció amb una capacitat per a minimitzar l'impacte i evitar una afectació generalitzada.

6.2 Com s'aplica

L'Enginyeria del Caos es basa en uns principis bàsics:

- Control estricte i mesurable de la resposta del sistema. Durant períodes curts de temps per establir punts de trencament a curt termini o durant períodes llargs de temps per observar la degradació del sistema. El primer valida que el sistema funciona, el segon valida que ho fa correctament.
- Variació dels esdeveniments que poden afectar el sistema. Prioritzar les variables que tenen un impacte potencial: errors del maquinari, pics de trafic o l'incorrecte funcionament d'altres sistemes relacionats.
- Les arquitectures que han de funcionar són les arquitectures que donen servei real, per aquesta raó realitzar les proves a l'entorn productiu, en temps real i amb connexions reals. Aquesta és l'única forma de capturar el que veritablement afecta el servei. També garanteix un entrenament davant les contingències i un nivell d'aprenentatge als administradors.
- Les proves seran continues com també passa amb les modificacions, les millores, el ús de la plataforma. S'han de fer funcionar els experiments de manera continuada, fent un seguiment del funcionament i analitzant els resultats.

L'Enginyeria del Caos és una pràctica molt potent i perillosa si és administrada de manera incorrecta, que impacta directament com es desenvolupa el nou software a grans entorns. On altres tècniques únicament cobreixen que el programari funcioni, sigui ràpid i flexible, aquest s'afegeix un vector més, que sigui fiable, que en definitiva és la base d'un entorn dissenyat per grans volums.

6.3 Per on començar

La plataforma de Netflix està en la seva totalitat hostatjada al servei Amazon Web Services i al llarg del temps han viscut experiències on han patit com caigudes del seu proveïdor de serveis [17], per això l'equip d'enginyers va pensar que la millor forma de combatre aquests problemes era provocar-los i estudiar sota control com donar solució abans que aquests passin de manera aleatòria i sense el coneixement de com mitigar el problema.

Una de les primeres peces de software que van programar els enginyers va ser el Mico del Caos, the Chaos Monkey. Aquest es tractava d'un programa que de manera aleatòria parava instàncies a AWS o parava serveis de l'arquitectura VOD. Per una banda, s'asseguraven que els sistemes de monitoratge funcionaven quan es produïa l'esdeveniment i per altra banda aprenien on estaven els punts febles a arreglar.

6.3.1 Exercit Mico

Deixaries un mico sol al teu centre de dades? Això semblant és el que van pensar els enginyers de Netflix encarregats de l'arquitectura del servei online de streaming de pel·lícules. [18]

Que pot ser més aleatori que un mico envoltat de cables i servidors? Els enginyers es van dedicar a dissenyar el seu propi exèrcit d'agents, els seus micos. Aquests com si fossin animals que provoquen tot tipus d'errors, accidents de forma automàtica i aleatòria. Aturen servidors, provoquen degradacions del servei, omplen els discos fins a deixar-los sense espai, desconnecten el cable de connexió a Internet,... tot un exèrcit de trapelles, tot un exèrcit mico.

Com s'hauria de tractar aquests agents fets a mida? El sentit comú ens diu que el primer pas seria habilitar un entorn aïllat de proves i dissenyar el teu exèrcit mico segons les teves necessitats. Els canvis aplicats per resoldre els problemes detectats pel nostre exèrcit d'agents a l'entorn aïllat anar-los integrant a l'entorn productiu. El següent pas, un cop tenim un entorn productiu estabilitzat després dels primers impactes, seria canviar l'objectiu del nostre exèrcit i aplicar-ho a l'entorn productiu. Del que en definitiva es tracta és a controlar la incertesa.

A continuació una petita descripció dels diferents d'agents que Netflix ha programat per controlar el seu entorn:

Chaos Monkey de manera aleatòria desactiva instàncies per verificar que aquest error comú no afecta el servei. El nom té com a origen la idea sobre com un mico salvatge armat al centre de dades descarrega el seu arma, fent caure els servidors i tallant els cables.

Latency Monkey introdueix un retard a les comunicacions entre el client i el servidor, simulant una degradació del servei.

Conformity Monkey busca aquelles instàncies o servidors que no aconsegueixen el millor rendiment de funcionament i les atura. Es podria caure en l'error de tenir un servei configurat esperant al fet que funciones sota certs paràmetres però no funciones afectant la resta del servei.

Doctor Monkey verifica que les instàncies estan "sanes" i que no tenen problemes amb els serveis que estan oferint, en cas de detectar-li problemes atura el servei i realitza una sèrie de verificacions per determinar les causes del problema.

Janitor Monkey verifica que l'entorn està endreçat i es manté lliure d'elements que consumeixen recursos innecessaris.

Security Monkey es una extensió del Conformity Monkey explicat anteriorment i busca possibles errors de seguretat o vulnerabilitats. Verifica les credencials d'accés, valida que els certificats estan correctament instal·lats i aquests són vàlids.

10-18 Monkey verifica si la configuració de mapa de caràcters, llenguatges són els correctes.

Chaos Gorilla donada la dimensió de l'arquitectura de Netflix, on es tracta d'un servei global els seus diferents sistemes utilitzen totes les regions de AWS. Aquest Gorila, actua de manera similar al Chaos Monkey però afectant a tots els serveis d'una localització.

6.3.2 Armant el nostre exercit mico

Al punt anterior hi ha una petita descripció dels diferents agents desenvolupats per Netflix per estudiar la seva plataforma davant l'error. Tal qual com estan dissenyats, desenvolupats difícilment es poden adaptar a la nostra arquitectura.

Per a desenvolupar el nostre grup d'agents és bàsic seguir, inicialment, unes polítiques per no transformar una ajuda a un problema.

- Inicialment no aplicar aquests agents durant l'horari de màxim funcionament. Investigar quins són els moments on la plataforma no està oferint el seu màxim servei, capacitat per evitar impactar directament al negoci.
- Decidir fins a on es vol arribar, potser és interessant treballar en l'àmbit de servei i no de servidor, evitant que la fallada provocada puguin escalar en un error major.
- Fer córrer els agents de manera aleatòria i continua, els errors no avisen i el sistema de monitoratge ha de detectar els errors provocats com els apareguts per l'ús.

Els primers objectius o els primers problemes a detectar i corregir serien aquells que poden provocar que l'entorn es pugui veure compromès, errors als balancejadors, a les bases de dades o als dispositius de connectivitat a la xarxa (commutadors, routers, tallafocs) generalment es tracten de punts únics de fallida.

Qualsevol errada al codi, qualsevol funcionament incorrecte per configuració al codi pot generar un error que finalitzi amb una caiguda de tot el sistema. Un error dins la llibreria de connexions o escriptura a la base de dades també pot tenir una afectació general que afecti a tot el servei.

6.4 Conclusions

Amb l'aproximació a l'enginyeria del caos de Netflix aquest treball tracta d'incorporar el concepte de l'error continu com a eina per mesurar la fiabilitat dels sistemes. L'error continu està format per: entorns sota vigilància, bancs de proves, captació eficient de les dades, representació i anàlisi posterior amb la presa de decisions basades amb el conjunt de les mètriques.
La fiabilitat com es va veure a l'inici del capítol 1 és una de les variables que formen el rendiment dels sistemes. No tenim eines per garantir la fiabilitat del sistema, podem mesurar el temps que porta funcionant però no garanteix que després d'una incidència continuï donant el servei per al qual estava dissenyat.

El que recull el principi del Caos no és únicament incloure un procediment amb unes regles controlen el codi i l'arquitectura com ja ho fa el control de qualitat (quality assurance) també incorpora la incertesa de produir errades no previstes als sistemes distribuïts per vigilar que les condicions de fiabilitat es mantenen tot i aquesta aleatorietat.

El principi del Caos és el paradigma o l'enginyeria del caos les podem aplicar les nostres plataformes?
SI condicional. Com s'ha vist l'exèrcit mico és un conjunt d'utilitats per controlar com respon el sistema amb condicions d'error aleatori. Es tracta d'un conjunt d'aplicacions que generen sinergies de millora davant dels errors, establint uns patrons pel desenvolupament del programari fiable i fent que en definitiva tot el sistema sigui fiable davant la contingència. Aplicar el Caos sobre un entorn no preparat o que no tingui la capacitat de preparar-se per falta de recursos o coneixements fa que indubtablement el seu impacte sigui totalment contrari l'esperat.

El exèrcit mico és la solució definitiva?
És una de les parts del monitoratge. Aplicar el exèrcit mico, sense un protocol de monitoratge i millora al voltant no té cap sentit, com es veia al principi del capítol el primer punt a tenir en compte és saber com és l'estat ideal de funcionament de l'entorn.

En definitiva, el monitoratge no pot ser un element exclusivament reactiu a les incerteses o detectar els errors una vegada ja s'han produït.

BLOC PRACTIC

Capítol 7
Nagios

7.1 Requisits

Per la part pràctica d'aquest treball, es realitzarà utilitzant els programes Nagios i Cacti, instal·lats a una màquina amb sistema operatiu Ubuntu Server 16.04 LTS.

El servidor web, les dependències de Nagios i Cacti que utilitzen Apache2 amb suport per PHP, llibreries diverses necessàries per als complements de Nagios, SNMP i RRDTool per Cacti.

Per realitzar la pràctica he utilitzat el software de virtualització VirtualBox, que permet crear màquines virtuals en un entorn controlat.

7.2 Instal·lació Nagios

Per la instal·lació de l'última versió Community de Nagios que compilarem el codi font descarregat de la web oficial de Nagios, ja que la versió que incorpora Ubuntu Server es tracta de la versió 3.5.1 d'agost del 2013. La darrera versió Core DYI (gratuïta) és la 4.1.1

Crear l'usuari i el grup nagios

```
$ sudo useradd nagios
$ sudo groupadd nagios
$ sudo usermod -a -G nagios nagios
```

Les dependències

```
$ sudo apt-get install build-essential apache2 apache2-utils xinetd
openssl libssl-dev libgd2-xpm-dev unzip
```

Descarreguem la darrera versió de Nagios 4 Core

```
$ wget https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.1.1.tar.gz
```

Compilem el codi font

```
$ tar zxf nagios-4.1.1.tar.gz
$ cd nagios-4.1.1
~/nagios-4.1.1$ ./configure --with-nagios-user=nagios
--with-nagios-group=nagios --with-openssl
~/nagios-4.1.1$ sudo make install
~/nagios-4.1.1$ sudo make install-init
~/nagios-4.1.1$ sudo make install-commandmode
~/nagios-4.1.1$ sudo make install-config
~/nagios-4.1.1$ sudo make install-webconf
~/nagios-4.1.1$ sudo make install-classicui
~/nagios-4.1.1$ sudo /usr/bin/install -c -m 644
sample-config/httpd.conf /etc/apache2/sites-available/nagios.conf
```

i fem el mateix procés amb la darrera versió dels plugins de Nagios

```
$ tar zxf nagios-plugins-2.1.1.tar.gz; cd nagios-plugins-2.1.1
~/nagios-plugins-2.1.1$ ./configure --with-nagios-user=nagios
--with-nagios-group=nagios --with-openssl
~/nagios-plugins-2.1.1$ make
~/nagios-plugins-2.1.1$ sudo make install
```

7.3 Configuració Nagios

Els paràmetres de configuració dels valors bàsics de funcionament per Nagios es realitza dins l'arxiu **nagios.cfg**. En aquest cas, com es pot veure a continuació, he deshabilitat l'ús d'arxius simples de definicions d'objectes per activar l'opció d'incloure tots els arxius de definicions que hi hagi dins d'un directori.

Per a la creació dels arxius necessaris de definicions s'ha utilitzat una nomenclatura per als seus noms per tal que sigui indicador del seu contingut.

```
$ sudo mkdir /usr/local/nagios/etc/CONFIGS
$ sudo chown nagios:nagios /usr/local/nagios/etc/CONFIGS -R
$ sudo ln -s /usr/local/nagios/etc/ /etc/nagios
```

arxiu: /usr/local/nagios/etc/nagios.cfg

```
# OBJECT CONFIGURATION FILE(S)
# You can specify individual object config files as shown below:
#cfg_file=/usr/local/nagios/etc/objects/commands.cfg
#cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
#cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
#cfg_file=/usr/local/nagios/etc/objects/templates.cfg

# Definitions for monitoring the local (Linux) host
#cfg_file=/usr/local/nagios/etc/objects/localhost.cfg

# Definitions for monitoring a Windows machine
#cfg_file=/usr/local/nagios/etc/objects/windows.cfg

# Definitions for monitoring a router/switch
#cfg_file=/usr/local/nagios/etc/objects/switch.cfg

# Definitions for monitoring a network printer
#cfg_file=/usr/local/nagios/etc/objects/printer.cfg

# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:
cfg_dir=/usr/local/nagios/etc/CONFIGS
```

L'estructura de directoris amb els arxius que contindran les definicions base

```
$ ls -R /usr/local/nagios/etc/
/usr/local/nagios/etc/:
cgi.cfg  CONFIGS  htpasswd.users  nagios.cfg  resource.cfg

/usr/local/nagios/etc/CONFIGS:
BASICS  COMMANDS  CONTACTS  HOSTS

/usr/local/nagios/etc/CONFIGS/BASICS:
TEMPLATES_hosts.cfg  TEMPLATES_services.cfg  TIMEPERIOD_24x7.cfg
TIMEPERIOD_8x5.cfg

/usr/local/nagios/etc/CONFIGS/COMMANDS:
COMMANDS_host_check.cfg  COMMANDS_notify.cfg  COMMANDS_service_check.cfg

/usr/local/nagios/etc/CONFIGS/CONTACTS:
CONTACT_danielpico.cfg  CONTACTGROUP_administrators.cfg
CONTACTGROUP_SMS_guardia.cfg  CONTACT_guardia.cfg

/usr/local/nagios/etc/CONFIGS/HOSTS:
HOST_cacti.cfg  HOST_commutador.cfg  HOST_localhost.cfg  HOST_router.cfg
HOST_serverDB.cfg  HOST_serverWWW.cfg  HOST_transaccional.cfg
```

7.4 Arxius de configuració

arxiu: TIMEPERIOD_24x7.cfg

```
define timeperiod{
    timeperiod_name         24x7
    alias               24 horas 7 dias
    monday              00:00-24:00
    tuesday              00:00-24:00
    wednesday                 00:00-24:00
    thursday           00:00-24:00
    friday              00:00-24:00
    saturday           00:00-24:00
    sunday              00:00-24:00
    }
```

arxiu: CONTACT_guardia.cfg

```
define contact{
      contact_name                    guardia
      alias                           contacto de guardia
      host_notifications_enabled      1
      host_notification_period        24x7
      host_notification_options       d,u,r
      host_notification_commands     host-notify-by-email,host-notify-by-sms
      service_notifications_enabled   1
      service_notification_period     24x7
      service_notification_options    c,r
      service_notification_commands   service-notify-by-sms
      email                           guardia@uoc.edu
      pager                           612345678
}
```

arxiu: COMMANDS_notify.cfg

```
define command{
    command_name    host-notify-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios
*****\n\nNotification Type: $NOTIFICATIONTYPE$\nHost:
$HOSTNAME$\nState: $HOSTSTATE$\nHost: $HOSTALIAS$\nAddress:
$HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" |
/bin/mail -s "** $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is
$HOSTSTATE$ **" $CONTACTEMAIL$
}

define command{
    command_name    service-notify-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios
*****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService:
$SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState:
$SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$" | /bin/mail -s "** $NOTIFICATIONTYPE$
Service Alert: $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **"
$CONTACTEMAIL$
}

define command{
     command_name    host-notify-by-sms
```

```
     command_line     /usr/bin/printf "$SHORTDATETIME$ -
$HOSTALIAS$ - $HOSTADDRESS$ - $HOSTSTATE$" | /bin/sms_sender -s
"$HOSTALIAS$ is $HOSTSTATE$" $CONTACTPAGER$
}

define command{
     command_name    service-notify-by-sms
     command_line    /usr/bin/printf "$SHORTDATETIME$ - $HOSTALIAS$
- $HOSTADDRESS$ - $NOTIFICATIONTYPE$ - $SERVICEDESC$" |
/bin/sms_sender -s "$HOSTALIAS$/$SERVICEDESC$" $CONTACTPAGER$
}
```

arxiu: COMMAND_service_check.cfg

```
define command{
     command_name    check_http
     command_line    $USER1$/check_http -H $HOSTADDRESS$
     }

define command{
     command_name    check_snmp
     command_line    $USER1$/check_snmp -H $HOSTADDRESS$ -C $USER4$
-o $ARG1$ -w $ARG2$ -c $ARG3$ -l " Valor: "
     }

define command{
     command_name    check_nrpe
     command_line    $USER1$/check_nrpe -H $HOSTADDRESS$ -w $ARG1$
-c $ARG2$
     }

define command{
     command_name    check_iso8583
     command_line    $USER1$/check_iso8583.py -H $HOSTADDRESS$ -p
$ARG1$ -d $ARG2$
     }
```

arxiu: HOST_serverWWW.cfg

```
define host{
      use                             generic-linux
      host_name                       serverWWW.tfc.local
      alias                           serverWWW
      address                         192.168.1.100
}

define service{
      use                             generic-service
      host_name                       serverWWW.tfc.local
      service_description             HTTP
      check_command                   check_http
}

define service{
      use                             generic-service
      host_name                       serverWWW.tfc.local
      service_description             Used sda1 space
      check_command                   check_nrpe!sda1!75!90
}

define service{
      use                             generic-service
      host_name                       serverWWW.tfc.local
      service_description             CPU load
      check_command          check_snmp!.1.3.6.1.4.1.2021.8.1.101.3!99!100
}
```

arxiu: HOST_transaccional.cfg

```
define host{
      use                             generic-linux
      host_name                       serverTRX.tfc.local
      alias                           serverTRX
      address                         192.168.1.155
}

define service{
      use                             generic-service
      host_name                       serverTRX.tfc.local
```

```
      service_description             Delay seconds TRX
      check_command                   check_iso8583!9876!5
}
```

per acabar verifiquem que la configuració dels diferents arxius es correcta

```
$ sudo /etc/init.d/nagios configtest

Nagios Core 4.1.1
Copyright (c) 2009-present Nagios Core Development Team and
Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 08-19-2015
License: GPL

Website: https://www.nagios.org
Reading configuration data...
   Read main config file okay...
   Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
    Checked 64 services.
    Checked 7 hosts.
    Checked 0 host groups.
    Checked 0 service groups.
    Checked 2 contacts.
    Checked 2 contact groups.
    Checked 13 commands.
    Checked 2 time periods.
    Checked 0 host escalations.
    Checked 0 service escalations.
Checking for circular paths...
    Checked 7 hosts
    Checked 30 service dependencies
    Checked 2 host dependencies
    Checked 2 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...
```

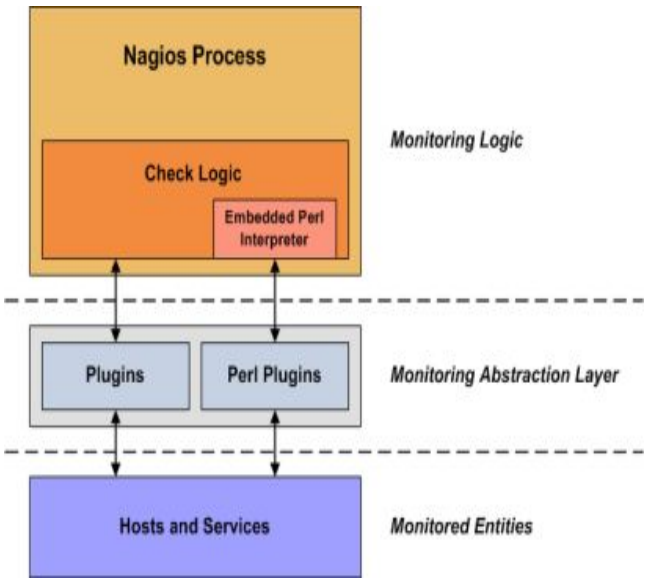
```
Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the
pre-flight check
Object precache file created:
/usr/local/nagios/var/objects.precache
```

Iniciem el servei

```
$ sudo /etc/init.d/nagios start
[ ok ] Starting nagios (via systemctl): nagios.service.
```

Capítol 8
Un nou plugin per Nagios

De vegades, tot i que el nombre de plugins de Nagios és molt gran, necessitem crear els nostres propis. Nagios això ho contempla i disposa d'una gran documentació que ens serveix de guia per desenvolupar correctament els nostres plugins específics. [19]



Existeix una guia de desenvolupament de plugins, indica uns mínims per mantenir una homogeneïtat entre ells. No defineix quin llenguatge de programació s'ha d'utilitzar, encara que Nagios estigui basat en C i PHP.

8.1 Guia per crear un plugin

1. Nagios únicament gestiona la sortida STDOUT dels scripts i ignora els missatges que apareixen a STDERR. Per defecte, el text ofert a la sortida STDOUT ha de ser una única línia però si es necessita més informació de sortida, es podria definir una opció de major grau d'informació *(verbosity)* fins a un màxim de 3 línies..

L'estructura per a la línia d'informació ha de seguir el següent format:
" $SERVICESTATUS$: Text d'informació "

2. Els codis de retorn del plugin tindrà la següent definició:

Valor Numèric	SERVICE STATUS	Descripció
0	OK UP	El script pot comprovar el servei i funciona correctament
1	Alerta - Warning DOWN	El script pot comprovar el servei i els valors de resposta estan dintre el rang de valors d'alerta del servei
2	Critic - Critical DOWN	El script no pot comprovar el servei i/o els valors de resposta estan dintre el rang de valors crítics del servei
3	Desconegut - Unknow DOWN	Els paràmetres al script són incorrectes o es produeixen errors de mal funcionament a les dependències

- Els SERVICE STATUS per a serveis serà OK, WARNING, CRITICAL i UNKNOW i per a equips els valors per SERVICE STATUS seran UP i DOWN.
- Definició de rangs de consideració de funcionament del servei en estat d'alerta o critic.
- En cas de disposar de més linies d'informació l'estructura del missatge de sortida STDOUT del plugin els camps d'informació extra restaran dividits amb el símbol **|** *(pipe)*

  Per exemple:

```
DISK OK - free space: / 3326 MB (56%);
|
/ 15272 MB (77%);
/boot 68 MB (69%);
```

```
/home 69357 MB (27%);
/var/log 819 MB (84%); |
/=2643MB;5948;5958;0;5968
/boot=68MB;88;93;0;98
/home=69357MB;253404;253409;0;253414
/var/log=818MB;970;975;0;980
```

Macro	Valor
$SERVICEOUTPUT$	DISK OK - free space: / 3326 MB (56%);
$SERVICEPERFDATA$	/=2643MB;5948;5958;0;5968 /boot=68MB;88;93;0;98 /home=69357MB;253404;253409;0;253414 /var/log=818MB;970;975;0;980
$LONGSERVICEOUTPUT$	/ 15272 MB (77%);\n/boot 68 MB (69%);\n/var/log 819 MB (84%);

8.2 Protocol ISO8583

El protocol ISO8583 [20] és l'estàndard per missatges de transaccions financeres originades per targetes de pagament i les especificacions estan desenvolupades per l'Organització Internacional per l'Estandardització, ISO per al seu nom en angles.

Aquest tipus de missatges es divideixen en tres parts:

- Primera part, missatges, elements d'informació i codis de valors
- Segona part, Procediments d'aplicació i registre
- Tercera part, procediments de manteniments per missatges, elements d'informació i codis de valors.
-

Una transacció financera es considera a una transacció efectuada amb una targeta de pagament (crèdit o dèbit) originada des d'un punt de venda o un caixer automàtic i el sistema autoritzador propietari de la targeta. La informació necessària per realitzar la transacció es construeix amb la informació de la targeta utilitzada, el punt d'utilització, el tipus de transacció realitzat i la resta d'informació generada de forma automàtica.

Com a resposta a aquesta petició el autoritzador pot acceptar o rebutjar la transacció i això generarà certs tipus de missatges de resposta que seran enviats a l'origen de la transacció en un temps determinat.

Les transaccions que poden realitzar els propietaris de les targetes inclouen la compra, extracció, dipòsits, retorns, reverses, consultes de balanç i transferències entre comptes però la definició del protocol ISO8583 incorpora aquestes operacions a més d'altres de gestió, com els intercanvis de claus entre sistemes, dades de conciliació i altres.

L'estructura d'un missatge ISO 8583 està integrat per les següents parts:

- Indicador del tipus de missatge (MTI)
- Com a mínim un mapa de bits amb la representació dels elements d'informació
- Els elements d'informació, les dades del missatge

MTI	Mapa de bits	Dades

Donada la seva flexibilitat el protocol permet la inclusió d'altres camps a l'inici del missatge en forma de TPDU i que inicialment no estaven previstes inicialment a l'estàndard ISO8583.

Missatge inici	Capçalera	MTI	Mapa de bits	Dades

El MTI es tracta d'un codi numèric de 4 dígits on els dos primers indiquen el tipus de missatge, el tercer la funció i el darrer l'origen. Generalment aquests 4 dígits del MTI es codifiquen com 2 bytes, cada mig byte o nibble representa cadascun d'aquests dígits. Per exemple, un missatge que comença amb `0x01 0x00` indica que la versió és 0 i la classe 1.

El mapa de bits és una tècnica d'indexació utilitzada per indicar quins elements d'informació del camp de les dades estan definits. La presència de l'element d'informació es representa a un 1 i l'absència amb un 0.

Els mapes de bits consisteixen en 64 bits desplaçats a l'esquerra on s'indiquen la presència dels elements entre l'1 i el 64. Al primer bit del mapa de bits primari pot definir l'existència d'un mapa de bits secundari afegint la presència dels camps compresos del bit 65 al bit 128.

El camp de les dades està compost per camps definits en el seu contingut on es transporta la informació de la transacció.

Existeixen fins a 128 camps definits a l'estàndard ISO8583:1987 i 192 camps a versions posteriors. Cada camp està especificat amb la mida de la dada i el tipus de dada. L'estàndard també defineix quin contingut està permès a cada camp i si té una longitud fixada o variable. En l'exemple veurem com en aquest cas, alguns camps utilitzats en la transacció no segueixen l'estàndard definit i s'ha de modificar la seva mida o el tipus de la dada continguda.

8.3 check_iso8583.py

El funcionament del check és per verificar que l'hora entre els terminals punts de venda i l'hora del servidor de transaccions coincideixen o és inferior a un valor que considerarem com a desviació assumible. El client considera que és vital que aquesta vigilància es realitzi de manera continuada i per aquest motiu no es genera una gràfica sinó que es necessita un monitor continu que alerti tan bon punt es produeixi la primera desviació no assumible.

Segons l'estàndard els missatges de monitoratge tenen al camp MTI per una petició `Echo Test` el valor `0800` i la resposta a la petició tindrà `0810`. També com es veia a l'apartat anterior, el protocol ISO8583 permet variacions per adaptar-se a les nostres necessitats en aquest cas utilitzarem la definició dels camps Inici i Capçalera.

Els camps necessaris al nostre check d'acord amb les especificacions del proveïdor son:

- Missatge d'inici: `ISO`
- Capçalera: camp numèric de longitud 9
- MTI: petició `0800`, resposta `0810`
- Mapa de bits: serà calculat en temps de execució
- Dades:
    - Camp 3: Codi de procés. Es tracta d'un camp numèric de longitud 6
    - Camp 11: Codi de traçabilitat al sistema. Camp numèric de longitud 6
    - Camp 12: Hora local. Camp numèric de longitud 6
    - Camp 13: Dia. L'estàndard ho defineix com a numèric de longitud 4 (MMDD) però en aquest cas el canviarem a numèric de longitud 6 (YYMMDD)
    - Camp 40: Codi de servei. Camp alfanumèric de longitud 3
    - Camp 41: Identificador terminal. Camp alfanumèric i caràcters especials de longitud 8
    - Camp 42: Codi de l'identificador. L'estàndard ho defineix com un camp alfanumèric amb caràcters especials de longitud 15 i que quedarà definit com a camp numèric de longitud 15.

Exemple del missatge generat pel nostre plugin:

Missatge inici	Capçalera	MTI	Mapa de bits	Dades
ISO	000000000	0800	2038000001c00000	000000122808122808160608010000000010000000000000000

8.3.1 codi

Es tracta d'un plugin desenvolupat en Python seguin les guies de desenvolupament de plugins per a Nagios. Utilitza la llibreria ISO8583 per a Python [21] que es pot instal·lar des del repositori d'Ubuntu

```
$ sudo apt-get install python-iso8583
```

Definició de les llibreries, variables de sortida i connexió amb el servidor a monitorar

arxiu: check_iso8583.py

```
#!/usr/bin/python

from ISO8583.ISO8583 import ISO8583
from ISO8583.ISOErrors import *
from time import localtime, gmtime, strftime
import socket
import sys
import string
import struct

OK=0
WARNING=1
CRITICAL=2
UNKNOWN=3

serverIP = 'localhost'
serverPort = '9876'
maxDELAY = '5'

try:
     options, args = getopt.getopt(sys.argv[1:], 'H:p:d:',
['host=', 'port=', 'delay='])
except getopt.GetoptError:
     sys.exit(2)

if len(sys.argv) != 7:
     print (' Usage: ./check_iso8583.py -H [host] -p [port] -d
[max. delay] ')
     sys.exit(2)
```

```
for opt, arg in options:
     if opt in ('-H'):
               serverIP = arg
     elif opt in ('-p'):
               serverPort = arg
     elif opt == '-d':
               maxDELAY = arg

# Establishing connection socket
s = None
for res in socket.getaddrinfo(serverIP, serverPort,
socket.AF_UNSPEC, socket.SOCK_STREAM):
   af, socktype, proto, canonname, sa = res
   try:
      s = socket.socket(af, socktype, proto)
   except socket.error, msg:
      s = None
      continue
   try:
      s.connect(sa)
   except socket.error, msg:
      s.close()
      s = None
      continue
   break

if s is None:
   print (' Service CRITICAL - Could not connect :(')
   sys.exit(CRITICAL)
```

Definició dels valors pel missatge d'inici i capçalera *(header)* i dels camps de dades. Les especificacions indiquen que el camp strace (camp 11) serà únic per a les últimes 24 h, per això es determinar utilitzar l'hora com a valor.

[CONTINUACIÓ] arxiu: check_iso8583.py

```
# Definition fields
header = 'ISO000000000'
mti = '0800'
pcode = '000000'
strace = strftime("%H%M%S", localtime())
```

```
stime = strftime("%H%M%S", localtime())
sdate = strftime("%y%m%d", localtime())
scode = '010'
terminalid = '00000001'
stationid = '000000000000000'
```

Formen el missatge ISO, tenint em compte les modificacions de mida i canvi de tipus de dades.

[CONTINUACIÓ] arxiu: check_iso8583.py

```
iso = ISO8583()
try:
   iso.setMTI(mti)
   iso.setBit(3, pcode)
   iso.setBit(11, strace)
   iso.setBit(12, stime)
   iso.redefineBit(13, '13', iso.getLargeBitName(13), 'N', 6, iso.getBitValueType(13) )
   iso.setBit(13, sdate)
   iso.setBit(40, scode)
   iso.setBit(41, terminalid)
   iso.redefineBit(42, '42', iso.getLargeBitName(42), 'N', 15, iso.getBitValueType(42) )
   iso.setBit(42, stationid)
except ValueToLarge, e:
   print ('Value too large :( %s' % e)
   sys.exit(UNKNOWN)
except InvalidMTI, i:
   print ('This MTI is wrong :( %s' % i)
   sys.exit(UNKNOWN)
```

Es calcula el mapa de bits sobre el missatge ISO afegint-li la capçalera (header) i posteriorment s'envia el missatge al autoritzador per a què processi la petició.

Cal dir, que la llibreria utilitzada no suporta la possibilitat d'afegir una capçalera al missatge ISO. Llavors donat la llicencia GPL del projecte es va decidir crear un nou mètode a la llibreria per afegir-hi aquesta nova funcionalitat.

[CONTINUACIÓ] arxiu: check_iso8583.py

```
# Building message adding header field
message = iso.getNetworkISOwithHeader(header)

# Sending echo test message
s.send(message)

# Receiving response from serverIP:ServerPort
```

```
ans = s.recv(1024)
```

Modificació de la llibreria per afegir el nou mètode.

arxiu: /usr/share/pyshared/ISO8583/ISO8583.py

```
     def getNetworkISOwithHeader(self, tpdu, bigEndian=True):
                netIso = ""
                asciiIso = tpdu + self.getRawIso()

                if bigEndian:
                     netIso = struct.pack('!h',len(asciiIso))
                     if self.DEBUG == True:
                               print ('Pack Big-endian')
                else:
                     netIso = struct.pack('<h',len(asciiIso))
                     if self.DEBUG == True:
                               print ('Pack Little-endian')

                netIso += asciiIso

                return netIso
```

Per finalitzar, el check aplica la lògica sobre el missatge ISO rebut generant una sortida per al STDOUT on s'informarà del resultat, la resposta ha estat l'esperada o si per al contrari s'ha detectat algun problema.

[CONTINUACIÓ] arxiu: check_iso8583.py

```
# Checking response message
if len(ans) >= 1 or len(ans) <= 128:
     if ans[0:3] != 'ISO' or ans[3:12] != '000000000' or ans[12:16]
!= '0810' or ans[16:32] != '2238000003C00008' or ans[32:38] !=
'000000' or ans[69:72] != scode or ans[72:80] != terminalid or
ans[80:95] != stationid:
                print (' Service WARNING - Unexpected values,
please check!!!')
                sys.exit(WARNING)

     #response code ERROR - CRITICAL
     if ans[48:54] != strace or ans[60:66] != sdate or ans[66:69]
!= '000':
                print (' Service CRITICAL - Error: %s - Trace: %s -
Message: %s ' % (ans[66:69],strace,ans[98:]))
```

```
                sys.exit(CRITICAL)

     #checking tolerance delay range
     if int(ans[54:60])-int(stime) > int(maxDELAY):
                print (' Service CRITICAL - Error: Max. Delay -
Time: %s - Message: %s ' % (stime,ans[98:]))
                sys.exit(CRITICAL)
     else:
                if ans[54:60] == stime:
                     print (' Service OK - %s (strace : %s) ' %
(ans[98:],strace))
                     sys.exit(OK)
                else:
                     print (' Service WARNING - GAP!!!! - Time: %s
- Message: %s ' % (stime,ans[98:]))
                     sys.exit(WARNING)

else:
     print ( ' Service CRITICAL - The answer(%s) is less than the
size 128!' % len(ans) )
     sys.exit(CRITICAL)
```

Capítol 9
CACTI

9.1 Requisits

Els requisits recomanats per a la instal·lació de Cacti inclouen:

- RRDTool 1.4 o superior
- MySQL 5.x o superior
- Net-SNMP 5.0 o superior
- PHP 5.1 o superior
    - Amb suport per SNMP, MySQL, XML, Sessions i Sockets
- Servidor web amb els mòduls i submòduls PHP

9.2 Instal·lació CACTI

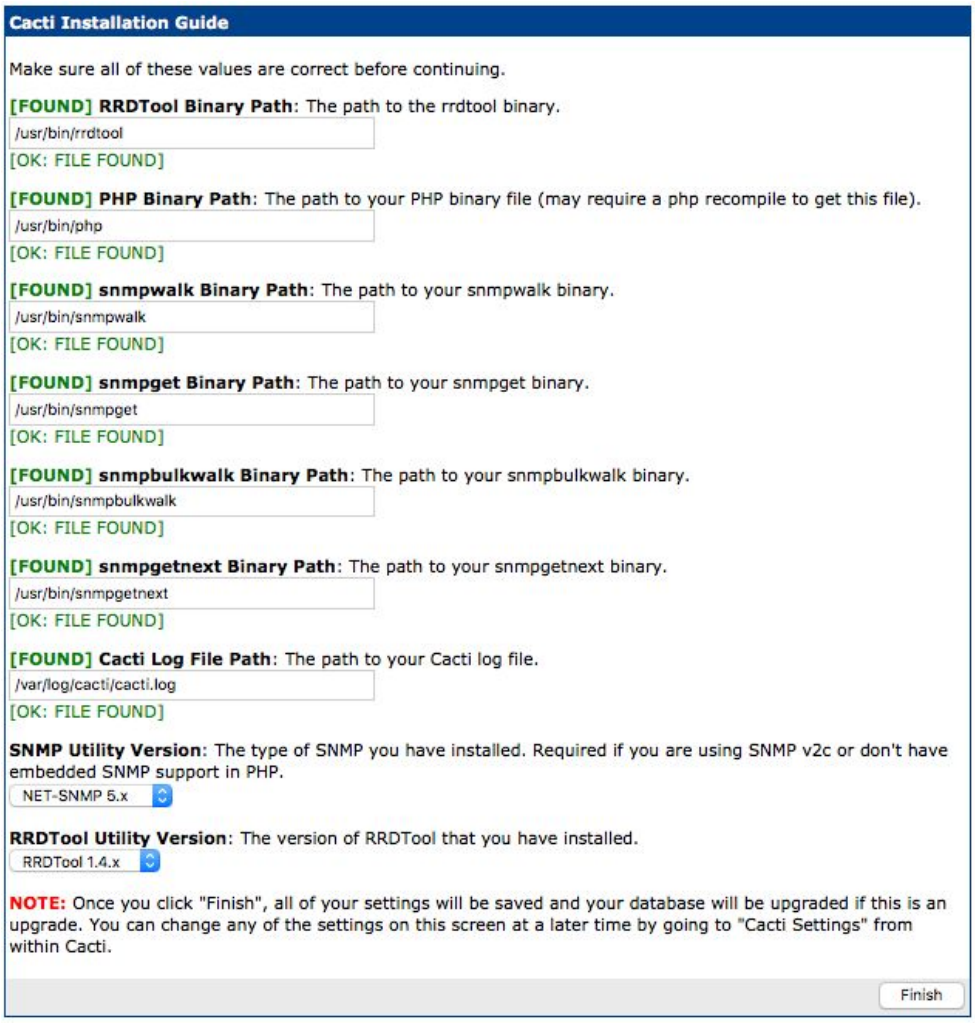
Utilitzarem la instal·lació del paquet d'Ubuntu 16.04

```
$ sudo apt-get install cacti cacti-spine
```

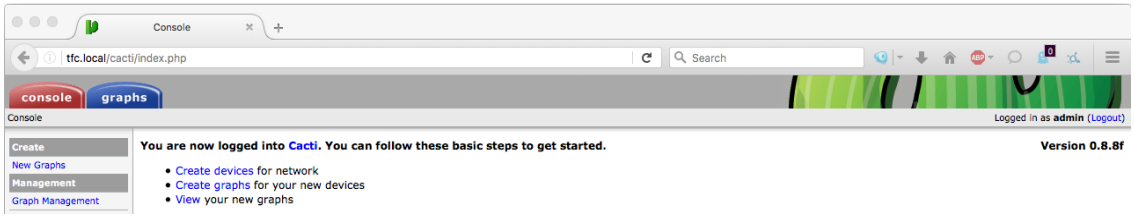
CACTI depèn del protocol SNMP per accedir als servidors i dispositius de xarxa a monitorar.

```
$ sudo apt-get install snmpd
```

Una vegada instal·lats els paquets el procés d'instal·lació continua a la direcció web on estigui el virtualhost de Cacti configurat. Es defineixen les utilitats que Cacti utilitzarà per a recol·lectar les dades, la configuració d'accés a les bases de dades, gestió d'usuaris,...

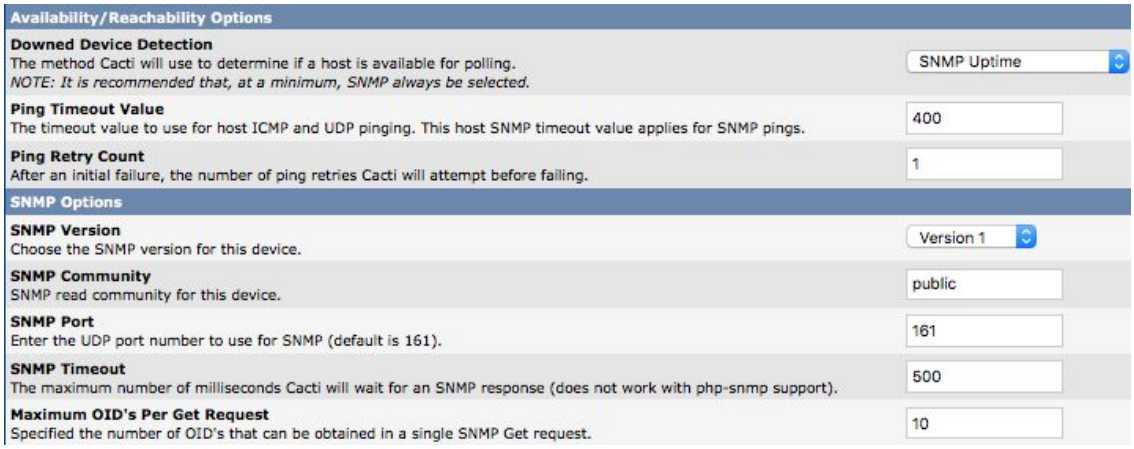


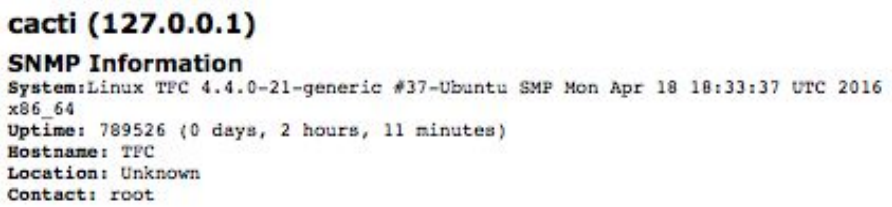
I la pantalla inicial queda com



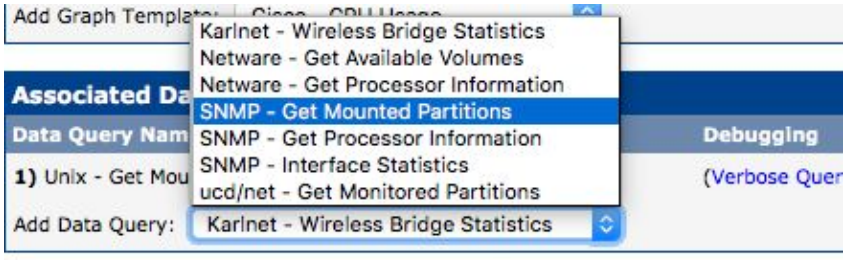
9.3 Afegir gràfiques

Activem la consulta per SNMP a l'equip al qual anem a crear gràfiques.



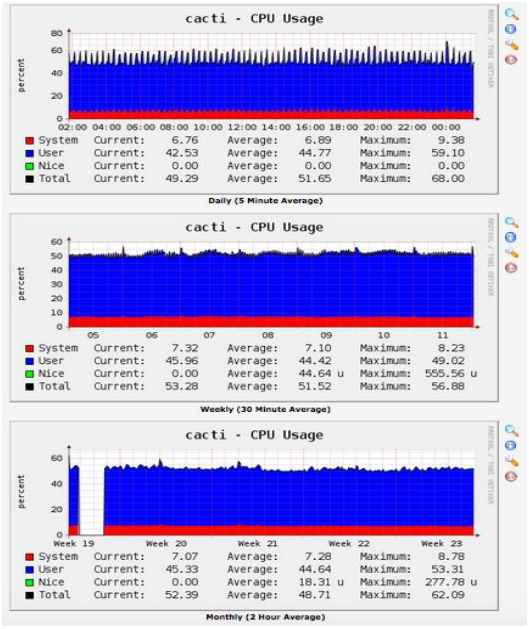


Independentment de les gràfiques que aconseguim dels valors de la plantilla Unix activem les consultes per SNMP de les particions i les estadístiques de les interfícies.

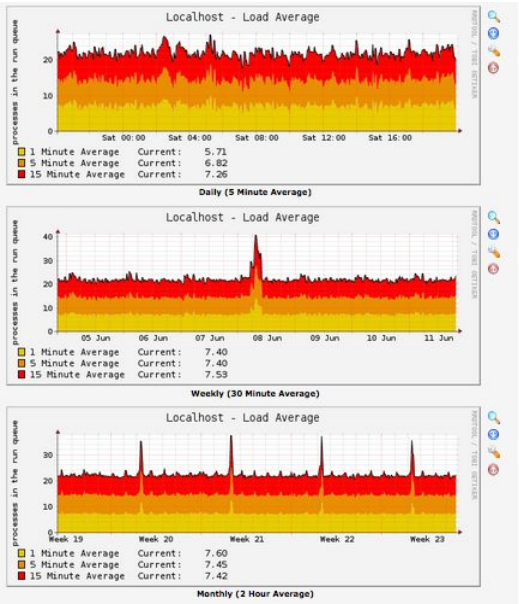


i això anirà generant unes gràfiques tipus:

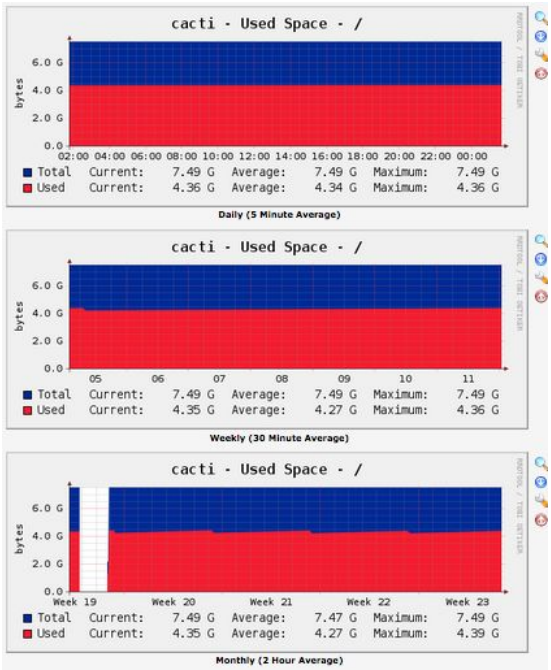
Ús de CPU



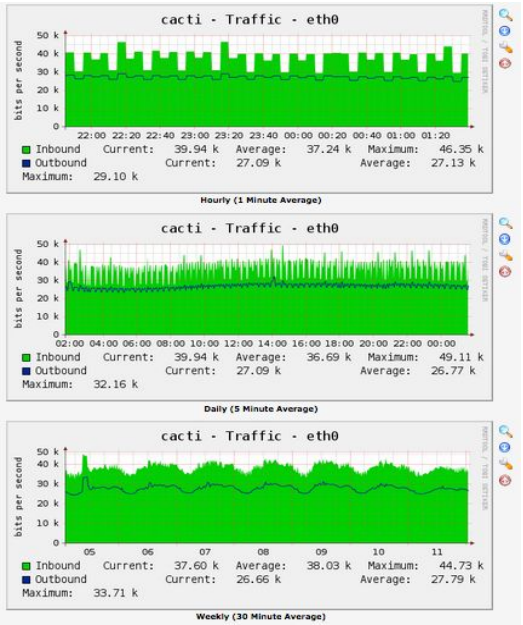
Càrrega del sistema



Utilització de disc



Ús de l'interface eth0



Conclusions finals

Si un cau un arbre al bosc i no hi ha ningú per escoltar-ho, fa soroll?

Els sistemes de monitoratge podrien ser la resposta a l'anterior pregunta, no perquè tinguin la resposta sinó perquè són les eines que disposem per a supervisar i vigilar tot el que passa al bosc. Hi ha una dita que diu que: "Si no es pot mesurar, no es pot controlar" i al llarg d'aquest treball s'ha definit que són els sistemes de monitoratge i com funcionen.

Sense monitoratge, sense cap element que estigui supervisant no es pot assegurar que al sistema, plataforma, arquitectura, entorn en el qual s'han invertit hores de disseny, de desenvolupament funcioni tal com s'havia dissenyat, desenvolupat.

Als objectius quan vaig pensar que volia incloure en aquest treball, vaig pensar a fer un llarg llista amb diferents eines, programes, dimonis vaig pensar que si ficava molts detalls, molta informació quedaria molt més clar que són els sistemes de monitoratge, després mentre anava llegint sobre aquells programes que normalment no utilitzo anava aprenent que en definitiva tots funcionaven de manera molt semblant i vaig decidir canviar el meu objectiu, vaig passar de voler explicar programes a explicar les arquitectures que defineixen el funcionament d'aquests programes.

De recol·lectors en tenim dos tipus, aquelles utilitats que periòdicament recullen la informació i la deixen a una base de dades amb una marca de temps i aquelles que fem servir per verificar el funcionament del servei utilitzant el mateix servei. Amb els magatzems de dades una mica el mateix dos tipus, magatzems de dades amb estructures fixades o magatzems de dades que permeten realitzar accions per consolidar les dades de diferents formes. I finalment els visualitzadors, dels bàsics programes per generar gràfiques d'una font de dades fins a

complexos panells amb diferents mètriques que permeten l'explotació de les dades segons els nostres interessos.

En definitiva de sistemes de monitoratge hi ha una gran col·lecció on escollir, cadascuna amb les seves característiques positives i negatives però en definitiva totes funcionen de la mateixa forma.

La inclusió del principi del caos era un tema que volia incloure en el treball perquè el considero una molt bona proposta per cobrir l'apartat de la fiabilitat convertint-se en el següent pas en això del monitoratge. Com també amb el bloc teòric dels sistemes de monitoratge com més anava llegint sobre l'enginyeria del caos més m'anava adonant que no existeix la "veritat" sobre com crear sistemes distribuïts i que en definitiva cada plataforma suposa un repte per als enginyers.

Hem d'interpretar que el monitoratge és temps, més concretament és el control del temps. Sèries de dades organitzades per marques de temps, són estats de funcionament en un determinat moment, son temps de resposta vers l'error, és el temps dedicat a resoldre problemes abans que aquests es produeixin. Seria una equivocació pensar que els sistemes de monitoratge es dediquen exclusivament a detectar errors, en essència és cert, però en concret el que aprenem de tot aquest grup de tècniques és a mantenir sota control el rendiment.

Si tenim un sistema que ofereix un servei 24 hores, 7 dies a la setmana i el nostre objectiu és garantir que el seu funcionament és del 100% ens hem d'organitzar amb totes les armes i tècniques possibles per aconseguir el nostre objectiu i en aquest treball s'ha volgut oferir una mostra d'això.

Glossari

Big Data És el nom que reben els conjunts de dades, els procediments i les aplicacions informàtiques, que, pel seu volum, la seva naturalesa diversa i la velocitat a què han de ser processades, ultrapassen la capacitat dels sistemes informàtics habituals. Aquest processament de dades s'utilitza per a detectar patrons dins seu, podent fer així prediccions vàlides per a la presa de decisions.

C llenguatge de programació compilat

Check Control, sensor, verificació

JSON JavaScript Object Notation. Format d'intercanvi d'informació en text pla. Molt popular per la transmissió de dades a través de la xarxa.

HTTP HyperText Transfer Protocol. El protocol de transferència d'hipertext estableix el protocol per a l'intercanvi de documents d'hipertext i multimèdia al web. HTTP disposa d'una variant xifrada mitjançant SSL anomenada HTTPS.

NMS Network Monitoring System. Plataformes de monitoratge, integrades per diferents mòduls encarregats de la recol·lecció, magatzem i visualització dels elements monitorats.

NoSQL Una base de dades NoSQL proporciona un mecanisme per emmagatzemar i recuperar dades que es modela mitjançant relacions tabulars diferents de les utilitzades en les bases de dades relacionals. Les estructures de dades utilitzades per les bases de dades NoSQL, clau-valor, graf i document, difereixen de les relacionals. L'ús de bases de dades NoSQL ha augmentat notablement en sistemes Big Data i en aplicacions web a temps real.

Protocol Un protocol de comunicació estableix una descripció formal dels formats que han de presentar els missatges per poder ser intercanviats entre diferents equips.

Python llenguatge interpretat multiplataforma. Disposa d'una llibreria completa que no limita al programador al paradigma de la programació orientada a objectes.

QA Quality Assurance. És el conjunt d'activitats planificades a un sistema de qualitat per a què els requisits de qualitat d'un producte o servei siguin satisfets.

REST, RESTful Representational State Transfer. Un servei web REST és un model centrat en les dades. Els anomenats recursos vénen identificats per URIs i poden ser manipulats mitjançant accions especificades a les capçaleres HTTP

Scripts Petites peces de codi desenvolupades per realitzar una funció simple.

SQL Structured Query Language. És un llenguatge normalitzat de comunicació amb bases de dades relacionals.

SSL i TLS Secure Sockets Layer i Transport Layer Security. Són uns protocols que ofereixen comunicacions segures a Internet mitjançant el xifrat de dades

TPDU Transport Protocol Data Unit. Les unitats de dades de protocol, s'utilitzen per a l'intercanvi de dades entre unitats no parells, dins d'una capa del model OSI. Les PDU de dades, que contenen les dades de l'usuari principal (en el cas de la capa d'aplicació) o la PDU del nivell immediatament inferior.

VOD Video On Demand. Emissió de vídeo sota petició

Bibliografia

Llibres

Marshall T.Rose; Keith McCloghrie. *How to manage your network using SNMP*. Primera edició. Prentice-Hall, Inc., 1995. ISBN 0-13-141517-4

Jordi Serra Ruiz, Miquel Colobran Huguet, Josep Maria Arqués Soldevila, Eduard Marco Galindo. *Administració de xarxes i sistemes operatius*. Segona edició. FUOC, setembre 2012. Dipòsit legal: B-17.165-2012

Ramesh Natarajan. *Nagios Core 3. Monitor Everything, Be proactive and Sleep Well.* The geek Stuff. 2010. http://www.thegeekstuff.com/nagios-core-ebook/

Bejtlich, Richard. *The practice of network security monitoring : understanding incident detection and response*. No Starch Press, 2013 ISBN 1-59327-509-9

Kurose, James F. *Computer networking : a top-down approach*. James F. Kurose, Keith W. Ross. 6a edició. PEARSON Education, 2013, 2010, 2008, 2005, 2003 ISBN 0-13-285620-4

Barth, Wolfgang. *Nagios: system and network monitoring*. 2a edició. No Starch Press, 2008 ISBN 1-59327-179-4

Presentació

Rapoport, Roy. *Cloud Operations at Netflix: Optimizing Innovation Speed While Supporting Availability*. Flowcon November 2013
http://flowcon.org/dl/flowcon-sanfran-2013/slides/RoyRapoport_CloudOperationsAtNetflixOptimizingInnovationSpeedWhileSupportingAvailability.pdf
Accés Juny 2016

En línia

[1] Múltiples autors. https://en.wikipedia.org/wiki/High_availability
Accés Abril - Maig 2016

[2] Múltiples autors. https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
Accés Maig 2016

[3] Múltiples autors. RFCs 1155, 1442, 2578, 1212 http://www.ietf.org/rfc/
Accés Abril - Juny 2016

[4] Florian Forster. https://collectd.org/documentation/manpages/collectd.1.shtml
Accés Maig 2016

[5] Florian Forster. https://collectd.org/wiki/index.php/Table_of_Plugins
Accés Maig 2016

[6] Costa Tsaousis. https://github.com/firehol/netdata/wiki/Configuration#netdata-plugins
Accés Juny 2016

[7] Nagios Plugins Development Team. http://nagios-plugins.org/
Accés Abril - Maig 2016

[8] Tobias Oetiker. http://oss.oetiker.ch/rrdtool/
Accés Abril - Maig 2016

[9] InfluxData. https://influxdata.com/time-series-platform/influxdb/
Accés Maig - Juny 2016

[10] Tobias Oetiker. http://oss.oetiker.ch/mrtg/
Accés Abril - Maig 2016

[11] Grafana. http://www.grafana.org/
Accés Maig 2016

[12] Facette. https://facette.io/
Accés Juny 2016

[13] The Cacti Group. http://www.cacti.net/
Accés Abril - Juny 2016

[14] The Cacti Group. http://docs.cacti.net/plugins
Accés Juny 2016

[15] InfluxData. https://influxdata.com/time-series-platform/
Accés Juny 2016

[16] http://principlesofchaos.org/
Accés Juny 2016

[17] Cory Hicks.
http://techblog.netflix.com/2011/04/lessons-netflix-learned-from-aws-outage.html
Accés Juny 2016

[18] Yury Izrailevsky. http://techblog.netflix.com/2011/07/netflix-simian-army.html
Accés Juny 2016

[19] Nagios Enterprises LLC
https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/pluginapi.html
Accés Maig - Juny 2016

[20] Múltiples autors. https://en.wikipedia.org/wiki/ISO_8583
Accés Juny 2016

[21] Igor V. Custodio. http://www.vulcanno.com.br/python/ISO8583.html
Accés Juny 2016