LABORATORIO DI AMMINISTRAZIONE DI SISTEMI T – A.A. 2010-2011

MONITORAGGIO DI RETE CON SOFTWARE OPEN-SOURCE

Progetto sviluppato da:

Abbati Marco
Brunetto Nicholas
Castellazzi Nicolò
Dall'Ospedale Emanuele
Grandi Luca
Privitera Gianluca

Indice:

1. Introduzione

La sicurezza di un'architettura di rete è un aspetto molto importante, per tutelarla può essere utile inserire dei sistemi di monitoraggio. Questi ultimi, infatti, possono rapidamente fornire una panoramica della rete, riguardo al suo funzionamento o un suo eventuale guasto.

Un sistema di monitoraggio consente inoltre di evidenziare lo sfruttamento di un nodo della rete, per identificare situazioni di carico eccessivo, in modo da ottimizzare l'architettura della rete.

Il programma più diffuso attualmente per il monitoraggio di reti è Nagios.

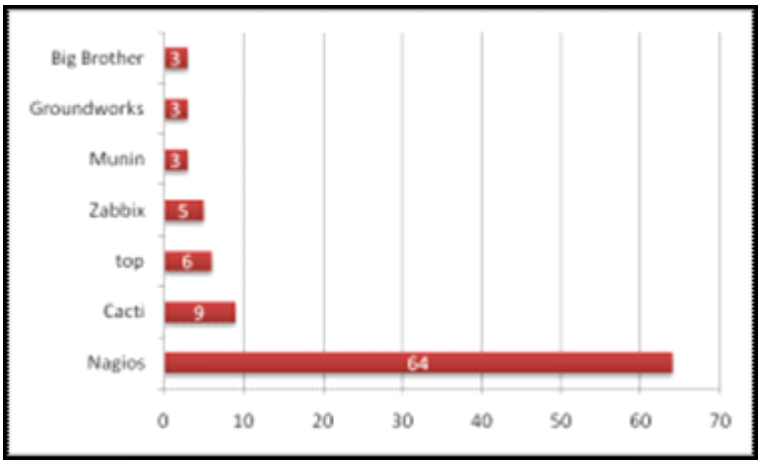


Figura 1: Diffusione programmi monitoraggio

Name	IP SLAReports	Logical Grouping	Trending	Trend Prediction	Auto Discovery	Agent	SNMP	Syslog	Plugins	Triggers / Alerts	Web App	Distributed Monitoring	Inventory	Data Storage Method	License	Maps	Access Control	IPv6
Cacti	Yes	Yes	Yes	Yes	Via plugin	No	Yes	Yes	Yes	Via plugin	Full Control	Yes	Yes	RRDtool, MySQL	GPL	Via plugin	Yes	Yes
Munin	No	No	Yes	Unknown	No	Yes	Yes	No	Yes	Partial	Viewing	Unknown	Unknown	RRDtool	GPL	Unknown	Unknown	Yes
Nagios	Via plugin	Yes	Yes	No	Via plugin	Supported	Via plugin	Via plugin	Yes	Yes	Full Control	Yes	Via plugin	Flat file, SQL	GPL	Yes	Yes	Yes
Open NMS	Yes	Yes	Yes	Unknown	Yes	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Limited	JRobin,PostgreSQL	GPL	Yes	Yes	Limited
Zabbix	Yes	Yes	Yes	Yes	Yes	Supported	Yes	Yes	Yes	Yes	Full Control	Yes	Yes	Oracle, MySQL,PostgreSQL, IBM DB2, SQLite	GPL	Yes	Yes	Yes

Figura 2: Tabella comparativa sistemi di monitoraggio analizzati (fonte: Wikipedia)

2. Nagios

Nagios è un tool di monitoraggio scritto da Ethan Galstad con l'intento di eliminare i difetti architetturali della sua precedente opera, chiamata NetSaint.

Tale software risulta potente ed altamente personalizzabile in tutti i suoi aspetti. Consente di monitorare macchine remote con sistema operativo Windows o Linux tramite l'installazione di un demone all'interno delle stesse e permette inoltre il monitoraggio di sistemi molto ampi e complessi.

Sono inizialmente presenti all'interno del sistema alcuni tipici comandi di monitoraggio (controllo dello spazio su disco, del numero di utenti, dell'utilizzo del processore, ...) ed è possibile crearne di personalizzati utilizzando i linguaggi C, Perl e script della shell. Il sistema è inoltre in grado di inviare notifiche via mail o sms nel caso i controlli mettano in luce situazioni critiche.

Nagios è dotato di una interfaccia web di facile comprensione, che tuttavia non consente di effettuare l'aggiunta di host da monitorare o visualizzare grafici in merito alle loro prestazioni.

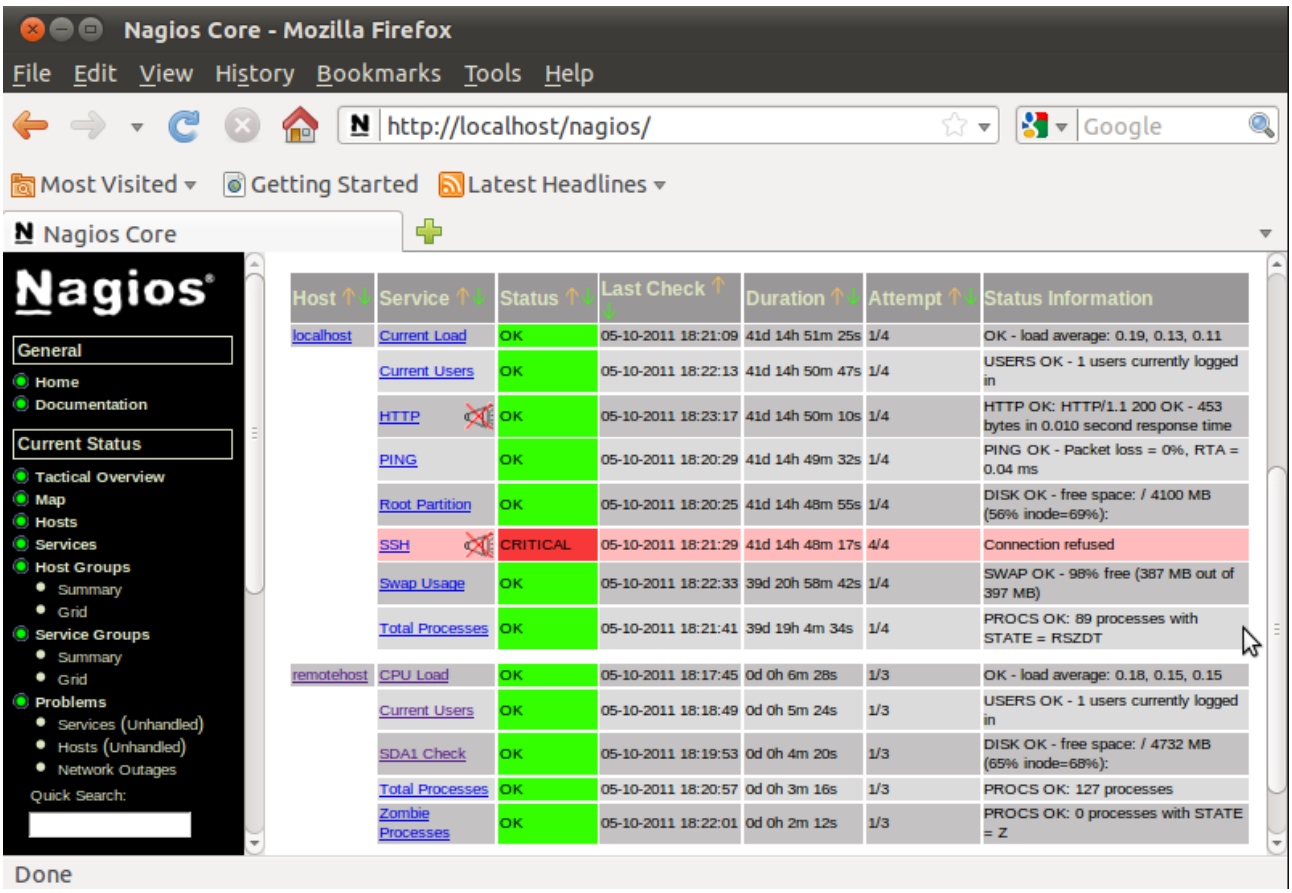


Figura 3: Web interface di Nagios

Pregi:

- Supporta il monitoraggio di sistemi eterogenei (Linux e Windows).
- Disponibile una documentazione chiara e dettagliata.
- Ampia comunità di sviluppatori (Nagios Exchange).
- E' dotato di una buona scalabilità (possibilità di monitorare anche molti hosts senza troppi rallentamenti).

- E' in grado di monitorare anche router, switch, stampanti ed altri componenti (ma a volte risultano necessari alcuni software di supporto a tale monitoraggio).
- Offre la possibilità di monitorare molti dei principali protocolli (HTTP, SSH, SNMP, ...).
- Può inviare notifiche via e-mail o sms.
- Disponibili molti potenti plugin e c'è la possibilità di crearne nuovi abbastanza facilmente (supportati numerosi linguaggi fra cui C, Perl e script di shell).
- Presenza di molteplici stati relativi ad un comando di monitoraggio (OK, WARNING, ERROR e UNKNOWN).
- Possibilità di gestire i malfunzionamenti tramite event handler.
- Interfaccia web semplice e di facile utilizzo.
- Possibilità di definire la topografia del sistema tramite host groups e altro.
- Flapping detection (individua quando un comando di monitoraggio ad un host specifico cambia di stato continuamente ed è in grado di disabilitare le notifiche di cambiamento di stato nel caso si verifichi tale comportamento).

Difetti:

- Sebbene seguendo la documentazione ufficiale si riesca a monitorare senza troppi problemi un sistema complesso la configurazione risulta abbastanza lenta da realizzare in quanto sono necessarie delle modifiche a diversi file di testo.
- Nell'interfaccia web di base non sono presenti grafici sull'andamento del monitoraggio per i diversi comandi (ma esistono alcuni plugin in grado di realizzarli appoggiandosi ad un RRD: nagiosgraph, PNP4Nagios, ...).
- Numerosi check vengono effettuati dal server stesso anzichè dagli hosts remoti causando una distribuzione del carico non uniforme e tendente a sovraccaricare di compiti il server.
- L'interfaccia web non fornisce una configurazione di hosts e services (necessità di configurazione tramite file di testo).
- Se si vuole suddividere il sistema di monitoraggio fra diversi server la configurazione risulta ancor più complessa del normale.
- Ogni insieme di parametri passato ad un plugin richiede la stesura di una nuova configurazione rallentando la realizzazione del sistema completo.

Conclusioni:

Nagios è un sistema ampiamente utilizzato e supportato, che dispone di tutte le funzionalità tipiche di un buon sistema di monitoraggio. Il suo reale problema sta nella scomodità di configurazione: la configurazione da file di testo occupa molto più tempo rispetto ad una configurazione tramite interfaccia web realizzata da altri sistemi di monitoraggio e vi è la necessità di porsi ad un livello abbastanza basso per poter realizzare una configurazione adeguata alle esigenze dello specifico sistema da monitorare. Detto ciò Nagios risulta consigliato a chi necessita di configurare un sistema in modo quasi totalmente personalizzato e sconsigliato a chi ha invece bisogno di configurare un sistema standard o di piccole dimensioni.

3. Cacti

“Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionality.”

Questa frase ben riassume le caratteritiche salienti e i limiti del software per il monitoraggio Cacti, il quale nasce con il preciso scopo di graficare i parametri vitali degli apparati connessi alla rete attraverso un'interfaccia estremamente semplice ed intuitiva. Una volta chiarito che questo è l'obiettivo secondo il quale Cacti è stato progettato, emergono subito i suoi limiti, in quanto non vengono offerte nessuna delle possibilità di monitoraggio avanzate offerte da altre soluzioni quali Nagios o Zabbix.

Un sistema di monitoraggio basato su Cacti è costituito da un server su cui è installato il software Cacti, che è accessibile (sia per visualizzare i grafici che per settare le impostazioni) tramite web interface. Il server Cacti colleziona i dati dagli host ad intervalli prefissati (default ogni 5 minuti) tramite il poller e li memorizza in un database RRD.

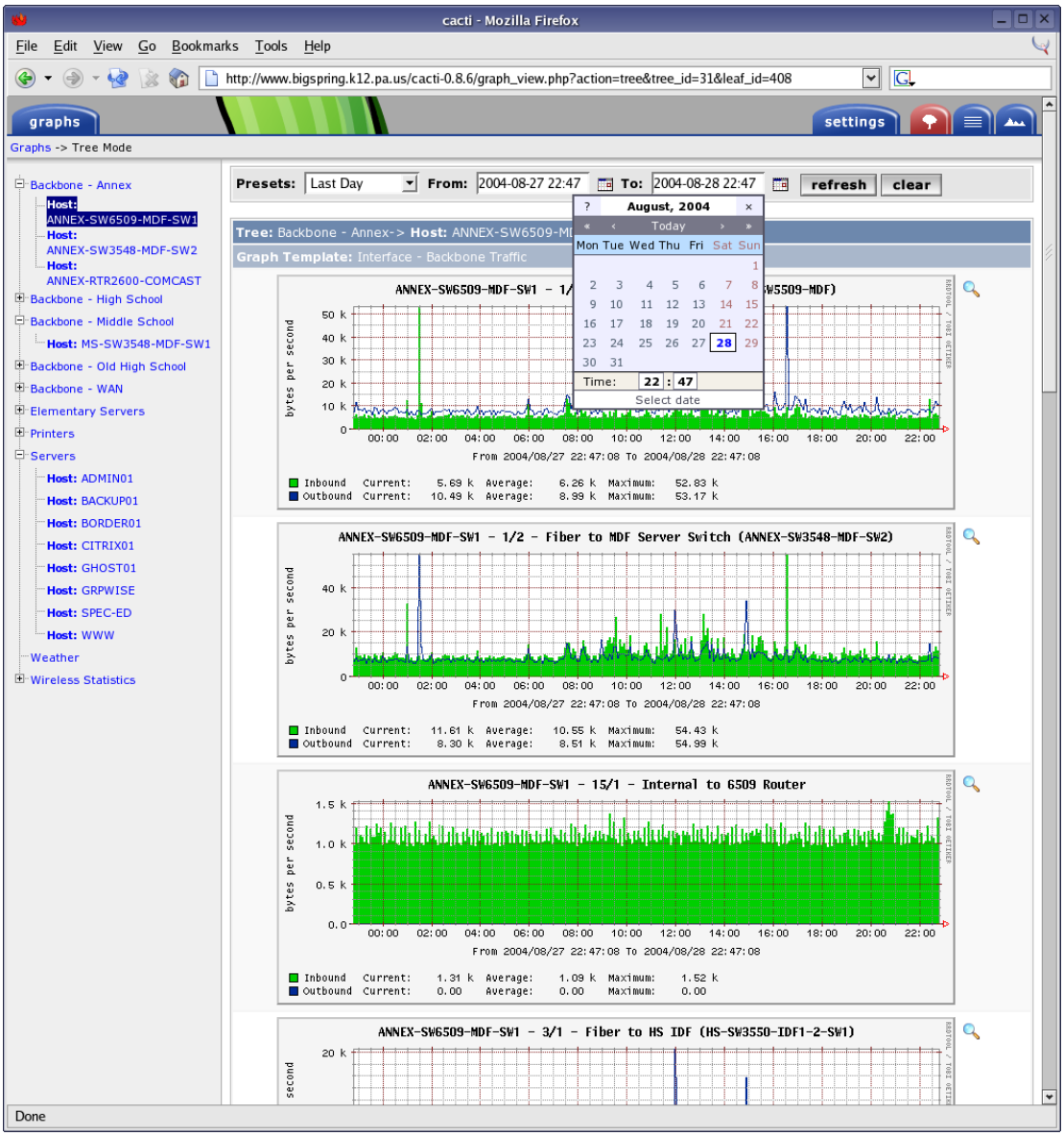


Figura 4. Web interface di Cacti: sulla sinistra si nota la struttura ad albero degli host, che semplifica la visualizzazione nel caso di un numero di host elevato

Pregi:

- Interfaccia web intuitiva grazie alla quale è possibile configurare la maggior parte dei parametri del monitoraggio senza dover ricorrere a file di testo, almeno per quanto riguarda installazioni basilari.
- Possibilità di personalizzare la visualizzazione dei grafici secondo le necessità: ad esempio è possibile selezionare manualmente l'intervallo temporale, essendo i grafici realizzati tramite Javascript.
- Buona gestione dei permessi agli utenti, in modo da consentire la visualizzazione di alcuni grafici a certi specifici utenti.
- Possibilità di scrivere script specifici per le proprie esigenze, attraverso i quali cacti colleziona i dati e quindi realizza i grafici.
- Possibilità di monitorare dispositivi di rete attraverso il protocollo SNMP: si possono monitorare dispositivi eterogenei senza dover installare un agent.

Difetti:

- In caso di superamento soglie limite non invia all'amministratore nessun alert ma si limita a graficare. Tale funzionalità può essere aggiunta tramite plugin, ad esempio thold è in grado di inviare alert via mail o sms.
- L'inserimento di nuovi host da monitorare avviene principalmente tramite interfaccia web: se questo può essere considerato un vantaggio nel caso di installazioni ridotte, può essere uno svantaggio nel caso di configurazione di sistemi complessi, che possono richiedere numerose operazioni ripetitive.
- Il tempo di polling di default è di 5 minuti, peggiorando ulteriormente la situazione nel caso di problemi dell'host, che non vengono rilevati prima di questo intervallo temporale.
- Non è possibile visualizzare una mappa della rete: tale funzionalità può tuttavia essere aggiunta tramite plugin.
- Non è presente una funzione per la ricerca automatica degli host da aggiungere al sistema: per introdurre tale funzionalità è disponibile un plugin.
- L'interfaccia web per l'aggiunta di nuovi host o parametri da monitorare non è molto chiara, e rischia di fare confusione tra template di visualizzazione, script per reperimento dati e tipologia del grafico.

Conclusioni:

E' evidente che non si tratta di un software in grado di monitorare da solo una rete complessa, ma di un software per graficare la situazione della rete, magari da usare in coppia ad un software più completo ma sprovvisto di tale funzionalità. Nella maggior parte dei casi è utilizzato per il monitoraggio di reti insieme a Nagios, il quale, pur offrendo funzionalià più generiche, non è in grado di fornire nativamente il supporto ai grafici.

4. Munin

Munin è un sistema di monitoraggio avanzato, facilmente installabile e configurabile.

Scritto in Perl, fa uso di RRDTool per raccogliere i dati e aggiornare i grafici relativi alle risorse che devono essere monitorate.

Offre un'interfaccia semplice e intuitiva, basata su una struttura modulare, che consente di monitorare facilmente le risorse sulla macchina locale e su eventuali macchine remote.

Munin ha una struttura client-server che prevede l'installazione del server sui nodi da monitorare: in tal modo il server interagisce con l'hardware e può raccogliere informazioni sull' host da monitorare (detto “node”) su richiesta del client. Quest'ultimo, invece, si occupa di contattare i vari server, collezionare i dati e generare pagine web che presentano i grafici risultanti dall'analisi dei dati raccolti.

L'interfaccia web di Munin consente una comoda visualizzazione delle risorse che è possibile monitorare e le raggruppa in varie categorie, tra cui Apache Server, Disk, System, Processes e Network; per ogni categoria sono presenti diverse voci e per ciascuna di esse sono disponibili grafici di riepilogo.

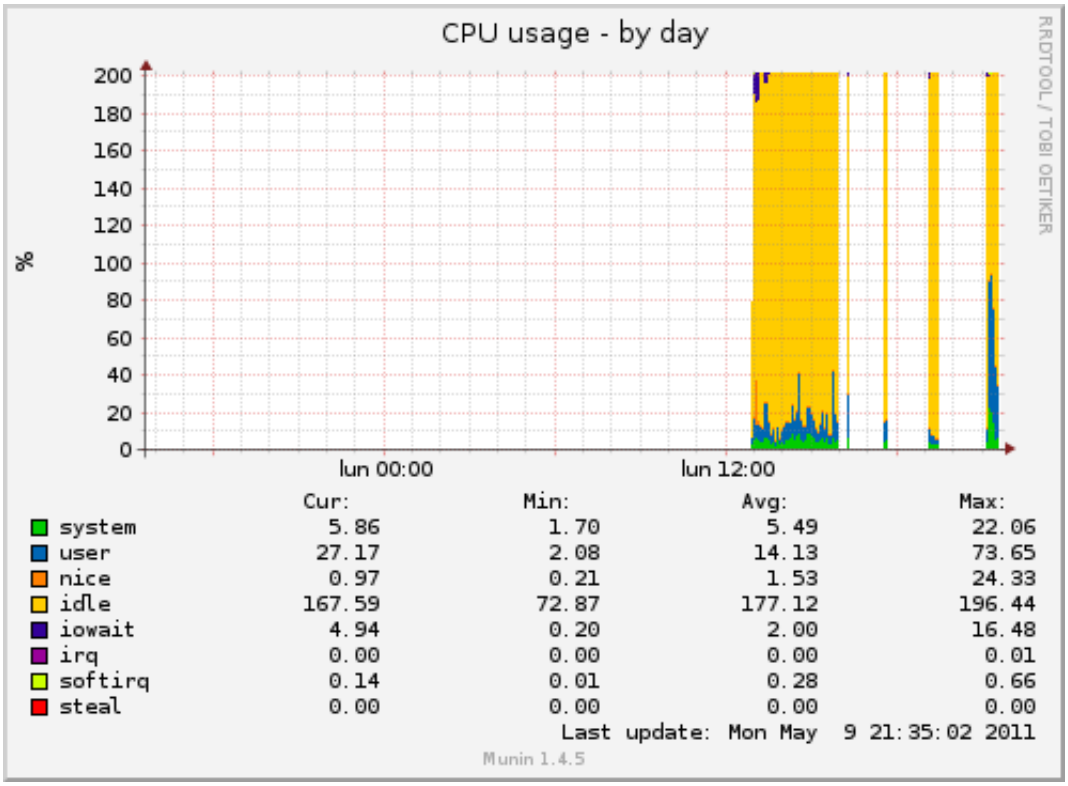


Figura 5: grafico dell’uso della cpu

Pregi:

- Facilmente installabile , si auto-configura sui server e sui nodi.
- Non richiede necessariamente la modifica di file di testo per essere configurato.
- Interfaccia web intuitiva, che raggruppa le risorse per categorie e fornisce pagine html con grafici relativi alle risorse monitorate.
- Si basa su un'architettura a plug-in per monitorare le varie componenti di sistema, perciò risulta essere facilmente estendibile.
- I plug-in sono di facile elaborazione e si appoggiano su RRDTool.

- Alto livello di dettaglio fornito dai grafici prodotti tramite plug-in.
- Non immagazzina i dati direttamente sul server.
- All'occorrenza è in grado di interfacciarsi anche con periferiche SNMP.

Difetti

- Tempo di polling elevato: genera grafici ogni 5 minuti.
- Eccessiva semplificazione che preclude la possibilità di configurazioni complesse: si nota che i file di configurazione, tra cui */etc/munin/muninnode.conf,* risultano essere scarni.
- A differenza di monitoring tool quali Zabbix e Nagios, Munin si incentra prevalentemente sulla parte grafica, tralasciando informazioni di sistema più specifiche, come ad es. gli eventuali demoni attivi.
- Si tratta di un programma statico, poiché non permette di navigare liberamente nei grafici prodotti, ed è molto legato a ciò che si definisce nel file di configurazione.

5. OpenNMS

Open NMS è un software per il monitoraggio di network a livello enterprise, scritto quasi interamente in linguaggio Java e che si pone come obiettivi primari la scalabilità e la portabilità.

E' rilasciato sotto licenza GPL e quindi gode di tutti i vantaggi che derivano dall'essere open source, rimanendo comunque altamente competitivo anche rispetto a software a pagamento.

L'installazione non risulta particolarmente difficoltosa a patto di avere già installati nel sistema la Java SDK e PostGRE SQL che viene usato dal programma come database.

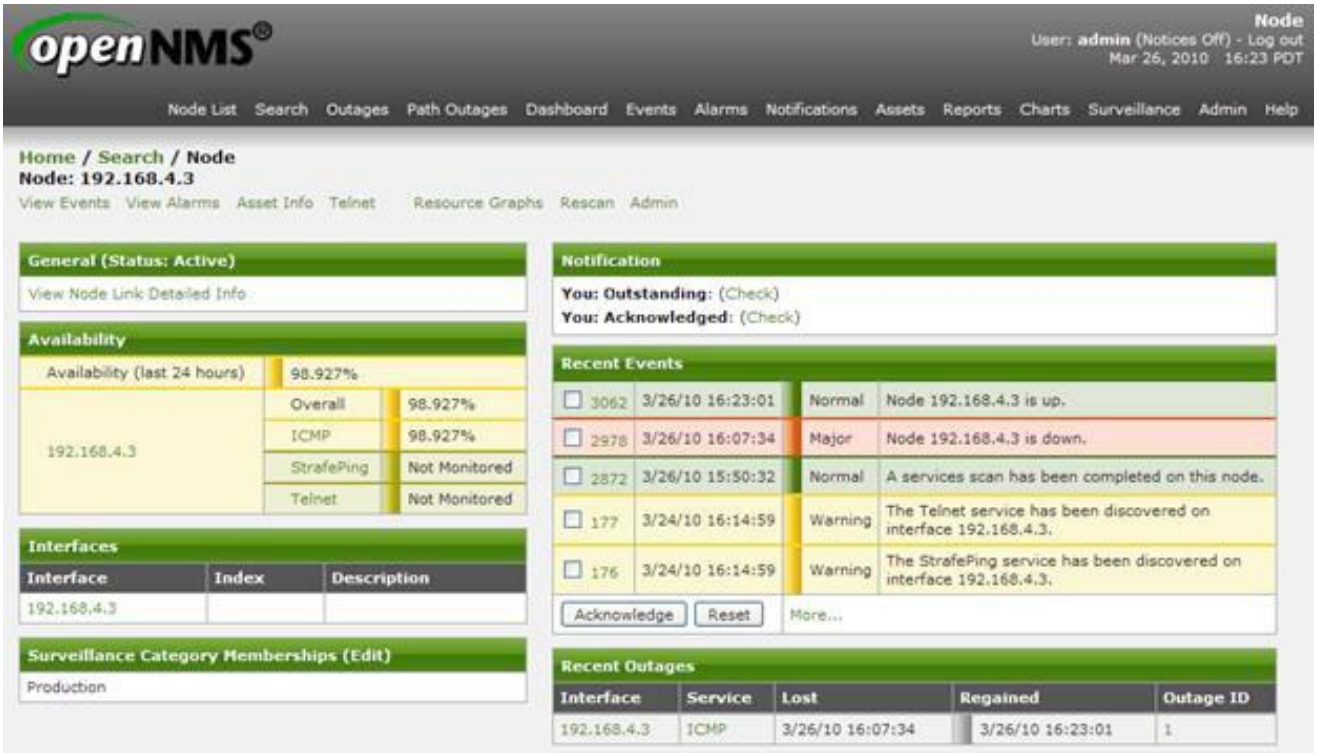


Figura 6: web interface relativa ad un nodo

Pregi:

- Offre il supporto a molti servizi quali:
  - Servizi Web: HTTP e HTTPS
  - Servizi di Posta: POP3, IMAP e SMTP
  - Database: Oracle, Sybase, Informix, SQLServer, MySQL e Postgres
  - Servizi di Rete: ICMP, SNMP, DNS, DHCP,FTP, SSH e LDAP
  - Altri Servizi: Citrix e Lotus Domino IIOP
- Intuitiva e completa interfaccia web.
- In situazioni di allarme può essere configurato per mandare SMS o e-mail all'amministratore del sistema.
- Possibilità di configurare le "Path Outages": se un nodo chiave del sistema crea problemi è possibile che anche altri nodi a causa di ciò non siano raggiungibili: in questo caso il sistema manda solo un messaggio di allarme relativo al nodo che non funziona invece che spedire un messaggio per ogni nodo non raggiungibile.

- E’ presente un sistema di reporting built-in che genera grafici per i dati SNMP che circolano sul sistema, oltre a generare report per uptime e altre informazioni utili.
- Attraverso alcune configurazioni può monitorare altre applicazioni, come controllare il tempo di caricamento per pagine web etc.

Difetti:

- Per far funzionare il sistema nella maniera desiderata è necessario configurarlo anche in maniera non ovvia.
- Nel reporting il range di date non è flessibile.
- Per configurare il sistema è necessario modificare vari file XML e script rendendo l’operazione a volte lunga e difficoltosa.
- E’ necessario investire tempo per imparare come usare il sistema al meglio in quanto ha una curva d’apprendimento abbastanza alta.

Conclusioni:

Si tratta di un software molto valido, fornito di un’ottima documentazione, che può competere con altri tool per il monitoraggio a pagamento. L’unica limitazione è che serve del tempo prima di imparare a configurare ed ottimizzare al meglio il sistema, operazione quasi sempre necessaria per far funzionare il tutto al meglio.

6. Zabbix

Zabbix è un tool di monitoraggio di sistemi che permette di generare report e grafici sullo stato di salute di un sistema e notificare eventuali anomalie via mail o sms. Scritto da Alexei Vladishev, è disponibile per diverse piattaforme quali Linux, Solaris, HP-UX, AIX, Free BSD, Open BSD, OS X e Windows.

Zabbix è un software open source, distribuito sotto licenza GPL. Per essere utilizzato necessita di Apache Web Server, PHP, ed un database quale MySQL oppure PostgreSQL.

Il sistema è accessibile tramite interfaccia web, che consente sia di visualizzare la situazione del sistema che di configurare i diversi host e i parametri da monitorare. Alcuni esempi di parametri monitorati possono essere: l'attività del disco fisso, il carico dei processori, l'utilizzo della ram, la temperatura.

Il programma è strutturato secondo un server centrale che riceve le informazioni dagli host monitorati, sui quali è necessario installare un agent. In casi specifici può essere inserito tra i due un proxy in grado di raccogliere informazioni da un numero ristretto di macchine e inviare i dati raccolti al server centralizzato.

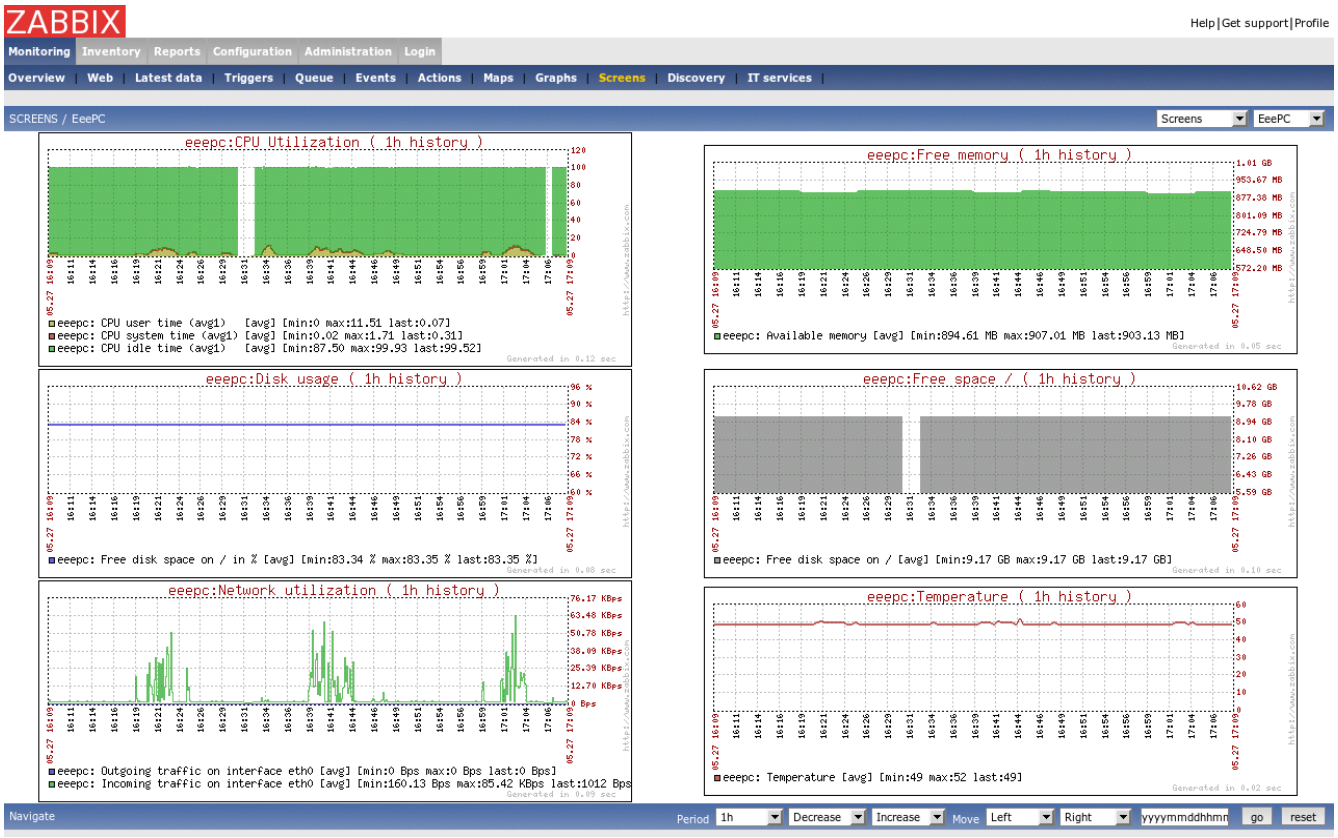


Figura 7: Web interface di Zabbix

Zabbix prevede varie forme di notifica come l'invio di e-mail a intervalli prefissati o avvisi istantanei al verificarsi di determinati eventi. Per conservare i dati controllati e le varie analisi il sistema utilizza un database MySQL o PostgreSQL.

Lo stato della rete viene rilevato tramite due modalità:

- Polling: consiste nell'interrogare ciclicamente tutti gli host ad un intervallo prefissato di 30 secondi.

- Trapping: si basa sugli agent installati in locale sugli host monitorati. Tale tecnica prevede che, al verificarsi di un evento, sia l'host a "svegliare" Zabbix favorendo la raccolta di dati. Questo comportamento è del tutto analogo all'invio di un interrupt.

Pregi:

- Programmi di monitoraggio (come Nagios) e grafica (come Cacti) sono combinati in un unico strumento.
- Altamente configurabile.
- Molto performante. Un agent Zabbix può essere installato su vari sistemi operativi (anche per Windows) e raccoglie le informazioni su ogni sistema in modo efficiente. L'agente può anche invocare degli script di shell per raccogliere informazioni.
- La raccolta di informazioni sui parametri di sistema avviene ad intervalli prestabiliti (di solito mezzo minuto). Ogni elemento può avere un intervallo di controllo personalizzato.
- Vi è un sistema di autorizzazioni che consente di limitare ad alcuni utenti la visualizzazione di parti del sistema.
- L'aggiunta di un host da monitorare è notevolmente facilitata dalla presenza di templates che rispecchiano le principali tipologie di macchine da monitorare. Tuttavia, nel caso di monitoraggi di sistemi personalizzati, è possibile scrivere script personalizzati.
- E' possibile configurare il sistema affichè segnali, al verificarsi di determinate condizioni, anomalie al system administrator, attraverso mail oppure avvalendosi di programmi esterni per l'invio di SMS o altri metodi di notifica.
- Sono presenti dei trigger che permettono di notificare all'amministratore se gli host stanno lavorando oltre ad una certa soglia di allerta per un periodo di tempo prolungato: in questo modo l'amministratore viene avvertito prima che si raggiunga una situazione critica.
- Oltre ad inviare alert utilizzando la configurazione predefinita, è possibile scrivere script in linguaggio di shell in grado di inviare messaggi.
- Oltre al server ed all'agent è possibile inserire nel sistema dei proxy utili per effettuare un monitoraggio remoto.

Difetti:

- La configurazione del monitoraggio tramite interfaccia web può diventare un limite nel caso di installazioni complesse, dove risulterebbero più utili file di testo già configurati ad hoc.
- L'interfaccia web con troppe funzionalità rischia di creare confusione agli utenti alle prime armi.
- La configurazione delle mappe, seppur migliorata nelle ultime versioni, risulta ancora macchinosa, contrariamente ad altri software di monitoraggio.
- Risulta scomodo monitorare diverse attività dello stesso tipo poiché è necessario specificare ogni volta gli stessi trigger.
- Non è in grado di rilevare automaticamente le risorse monitorabili nell'host.

IMPLEMENTAZIONE SISTEMA DI MONITORAGGIO

Avendo analizzato le funzionalità offerte dai software analizzati, si è scelto di utilizzare zabbix, in quanto dispone già di quasi tutte le funzionalità che ci sono state richieste; inoltre la documentazione è esaustiva, come anche il supporto online per quelle parti non presenti ma configurabili tramite plugin.

Come software per le macchine virtuali si è scelto virtualBox che, pur essendo estremamente semplice da configurare, dispone di tutte le funzionalità a noi necessarie.

Per quanto riguarda le macchine virtuali, per i due server è stato scelto ubuntu, mentre per gli agent, per diversificare la configurazione, si sono scelti tre diversi sistemi operativi: windows server 2008, debian e mandriva.

1. Analisi delle features di zabbix

- **Items:**

Le item sono utilizzate in zabbix per consentire la memorizzazione dei dati relativi ai check effettuati per monitorare i diversi host del sistema. Ogni item rappresenta un singolo check e può essere aggiunta a tutti gli host in cui è possibile effettuare quel determinato controllo. Le item sono identificate da una chiave, tramite cui è possibile riferirle, ed eventualmente da zero o più parametri. Di per sè le item permettono solo di memorizzare i dati e non mostrano all'esterno i loro valori, quindi per consentire all'amministratore di sistema di individuare eventuali problemi di monitoraggio è necessario associare le item a dei trigger, di cui parleremo in seguito.

- **Triggers:**

Un trigger è definito come una espressione logica che rappresenta lo stato di un sistema o di parte di esso. Un trigger viene usualmente associato a una o più item ed utilizza funzioni specifiche per effettuare check su tali item (in base al loro valore o altro). Quando l'espressione di un trigger risulta vera il suo stato sarà impostato su OK, ad indicare che gli oggetti monitorati presentano dei risultati in linea con un andamento non critico del sistema. Se l'espressione diventa falsa il trigger passerà allo stato PROBLEM permettendo così all'amministratore di sistema di individuare il problema osservando l'interfaccia web di zabbix, già dalla pagina principale (e in modo più dettagliato nella sezione Monitoring/Triggers). Se si volesse gestire in modo automatico il problema senza dover per forza attendere una azione da parte dell'amministratore di sistema occorrerà definire una action associata allo stato PROBLEM del trigger (vedi in seguito per dettagli). L'ultimo stato in cui potrebbe trovarsi un trigger è lo stato UNKNOWN che rappresenta l'impossibilità da parte del trigger di collezionare i dati e verificare le condizioni su essi (solitamente a causa della mancanza di tali dati).

- **Actions:**

Tramite la definizione delle action è possibile effettuare in modo automatizzato alcune operazioni in seguito al cambiamento di stato di un trigger ma anche ad altre condizioni (zabbix consente la maggiore libertà nella scelta delle condizioni da associare ad una action). Le operazioni maggiormente degne di nota che una action è in grado di effettuare sono l'invio di una e-mail all'amministratore di sistema (operazione facilmente configurabile tramite l'interfaccia web di zabbix specificando il client di posta) e l'esecuzione di un comando remoto (solitamente legato al tentativo di riportare un trigger nello stato OK effettuando un comando sulla macchina che presenta il problema: ad esempio nel caso in cui fosse stato segnalato un problema con il demone HTTP di un host remoto potrebbe essere stata realizzata una action che tenta il riavvio di tale demone).

- **Templates:**

I templates permettono di raggruppare assieme diversi tipi di item, trigger e grafici. Sono utili poichè una volta effettuato tale raggruppamento un template potrà essere collegato ad un numero illimitato di host ed essi condivideranno in tal modo gli oggetti presenti all'interno del template. Grazie ad essi è possibile risparmiare del tempo nella fase di configurazione e personalizzare il monitoraggio in base alla macchina che deve essere monitorata (vi sono templates per macchine windows, linux ed altro).

- **Flapping Detection:**

Questa feature, già descritta in precedenza fra i pregi di Nagios, non è presente nativamente all'interno di Zabbix. L'unico modo per ottenere un comportamento simile è quello di definire un trigger apposito in grado di individuare ed eventualmente ignorare dei valori oscillanti in un breve intervallo di tempo. Tutto ciò richiederebbe un grosso ammontare di tempo poichè andrebbe definito un trigger per ogni valore da monitorare. Per tale motivo, assieme al fatto che tale feature non è strettamente necessaria, si è deciso di non implementarla nel sistema finale.

- **Path Outage:**

Feature presente all'interno del software di monitoraggio OpenNMS e realizzabile anche su Zabbix definendo le dipendenze fra host. Se ad esempio si volessero monitorare un router ed un server strettamente dipendente da esso, per l' interfacciamento con l'esterno sarebbe desiderabile ricevere soltanto l'eventuale notifica di irraggiungibilità del router e non anche quella riguardante il server (che sarà naturalmente irraggiungibile a sua volta); per realizzare ciò basta creare un trigger con il quale si definisce tale dipendenza ("The server is down" depends on "The router is down"). Essendo questa feature abbastanza utile all'interno di un sistema di monitoraggio e anche semplice da realizzare, se necessario verrà implementata nel sistema finale. *(Fonte: Zabbix Documentation – Capitolo 4 – Paragrafo 4.8)*

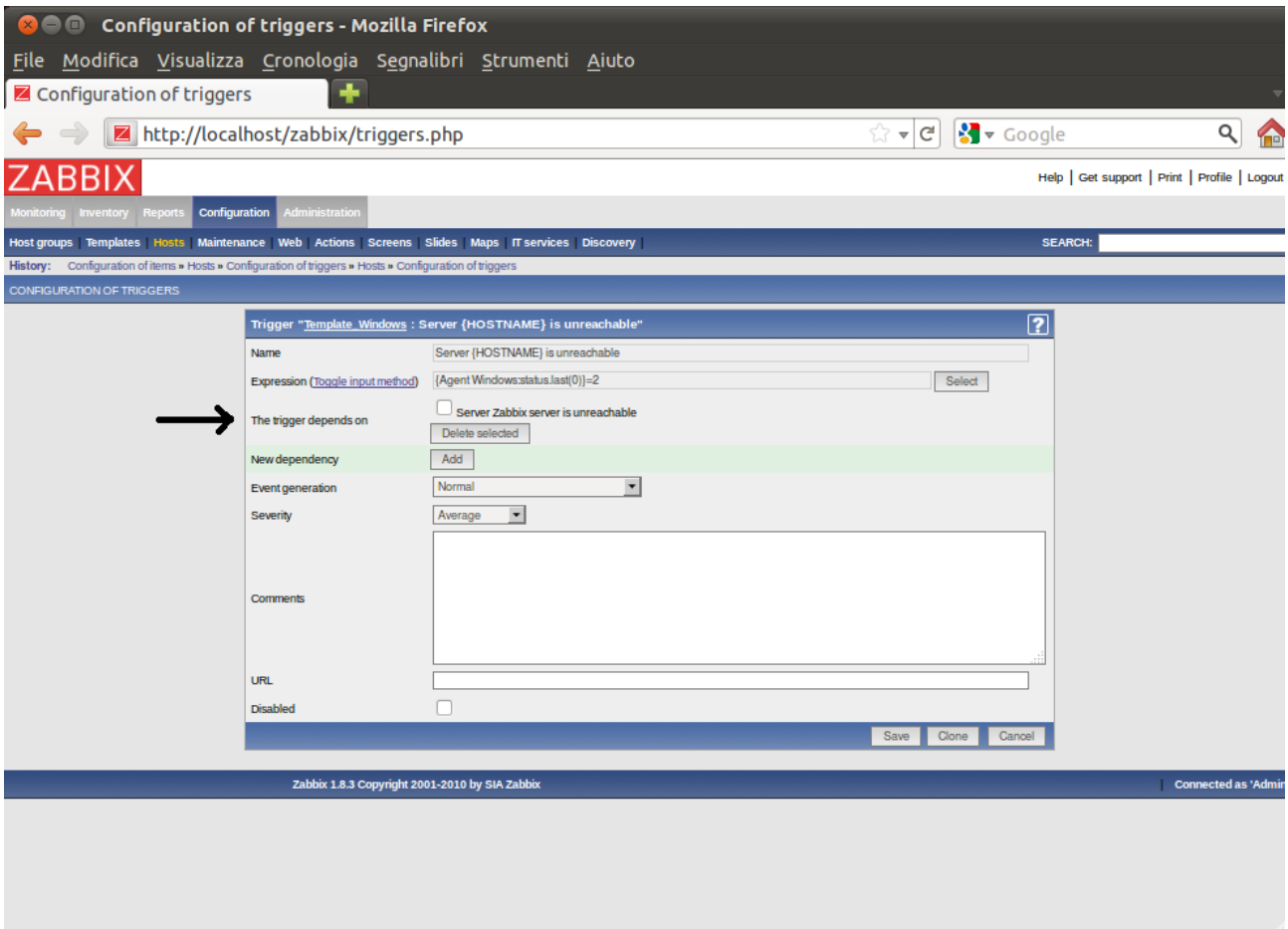


Figura 8: inserimento delle dipendenze all'interno di un trigger Zabbix.

- **Gerarchie di Templates:**

Un'altra caratteristica importante di cui è dotato Zabbix riguarda la possibilità di estendere i templates già esistenti creando con facilità delle gerarchie di templates. Ciò è possibile, all'atto di creazione di un nuovo template, selezionando il tasto "Add" di fianco alla label "Link with template" ed aggiungendo nella finestra che appare i templates a cui si vuole fare riferimento (in tal modo verranno automaticamente aggiunte al nuovo template le funzionalità già presenti nei templates a cui esso fa riferimento). Se il progetto lo richiederà verrà utilizzata anche questa feature. *(Fonte: Zabbix Documentation – Capitolo 4 – Paragrafo 6)*

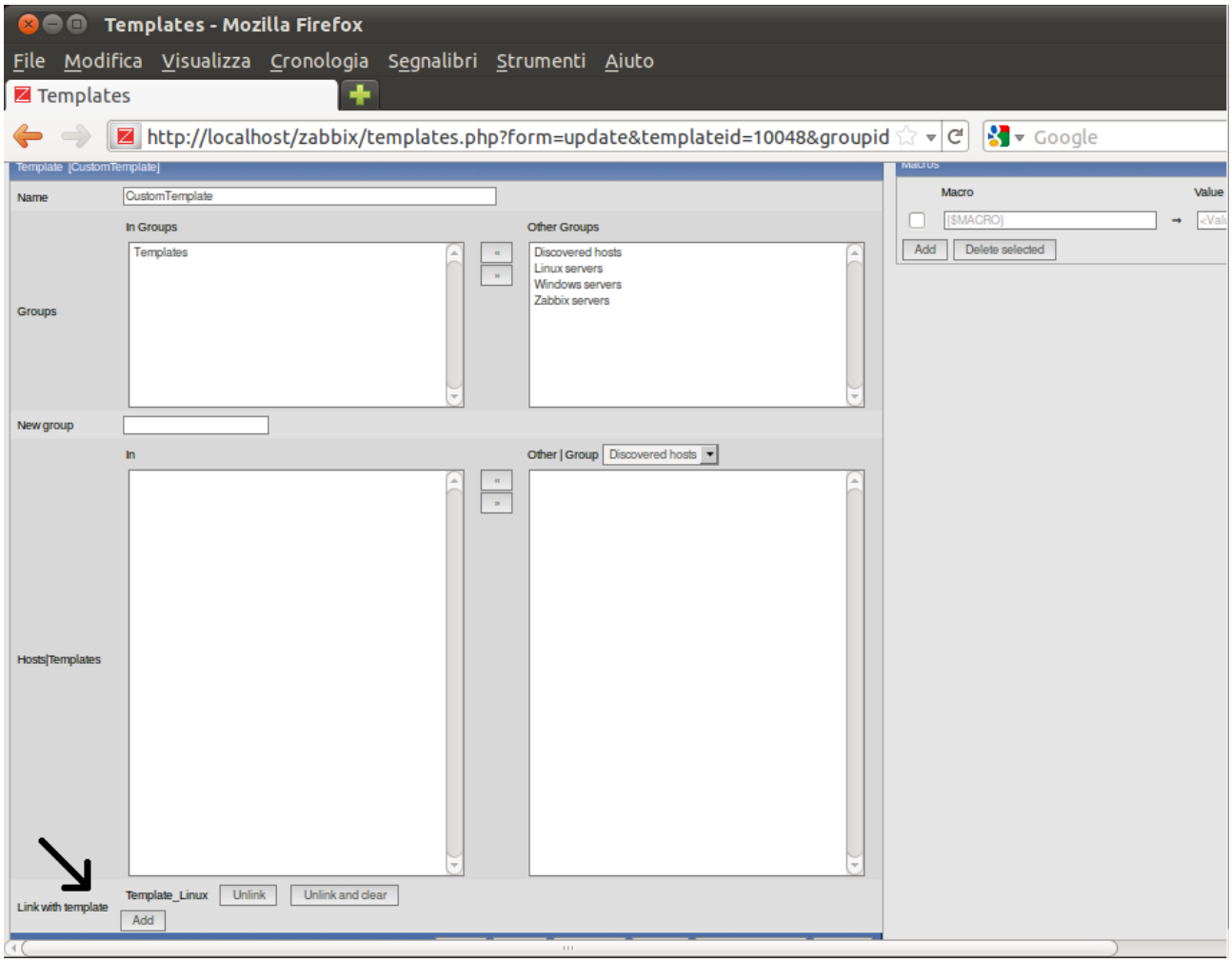


Figura 9: inserimento di un template che eredita da un template pre-esintente.

- **Host Groups:**

Tramite la definizione degli Host Groups è possibile personalizzare in modo rapido ed efficace il monitoraggio delle diverse macchine. In tal modo risulta possibile definire i templates per un gruppo di host evitando la configurazione di ogni singolo host appartenente al gruppo. E' possibile inserire un host all'interno di più gruppi ma non risulta possibile formare un gruppo all'interno di un altro gruppo ed è quindi necessario specificare ogni singolo host da inserire, richiedendo un po' più di tempo per la creazione dei gruppi. Essendo questa caratteristica comoda e facilmente realizzabile sarà ampiamente utilizzata all'interno del sistema.

2. Configurazione Virtualbox

Per consentire alle macchine virtuali di comunicare tra di loro, sono state configurate con la connessione "scheda solo host" dalle impostazioni di rete di ogni macchina. Le macchine risultano quindi connesse a "Virtualbox Host-Only Ethernet Adapter", che è stata configurata nelle impostazioni generali di virtualbox con gli IP statici, in modo che ogni macchina mantenesse lo stesso indirizzo ad ogni avvio.

Configurazione macchine:

- **serverCentrale-ubuntu**
  - os: ubuntu 11.04 32bit
  - indirizzo ip: 192.168.56.102
  - nome utente: server
  - password: server1
- **serverIntermedio-ubuntu**
  - os: ubuntu 11.04 32 bit
  - indirizzo ip: 192.168.56.105
  - nome utente: server
  - password: server
- **agentWindows**
  - os: windows server 2008
  - indirizzo ip: 192.168.56.103
  - nome utente: server
  - password: Abcd1234
- **agentDebian**
  - os: debian 6.0.1
  - indirizzo ip: 192.168.56.104
  - nome utente: agent
  - password: agent
- **agentMandriva**
  - os: mandriva
  - indirizzo ip: 192.168.56.106
  - nome utente: agent
  - password: agent

Oltre a disattivare il server DHCP per la rete "Virtualbox Host-Only Ethernet Adapter", nelle impostazioni di rete di ogni macchina si è impostato l'ip indicato sopra.
Nella macchina con windows server 2008, trattandosi di un sistema operativo per server, è stato necessario abilitare l'individuazione della rete e configurare il firewall in modo che consentisse il traffico in entrata dalla porta 10050 (TCP), altrimenti il server non è in grado di comunicare con l' agent zabbix.

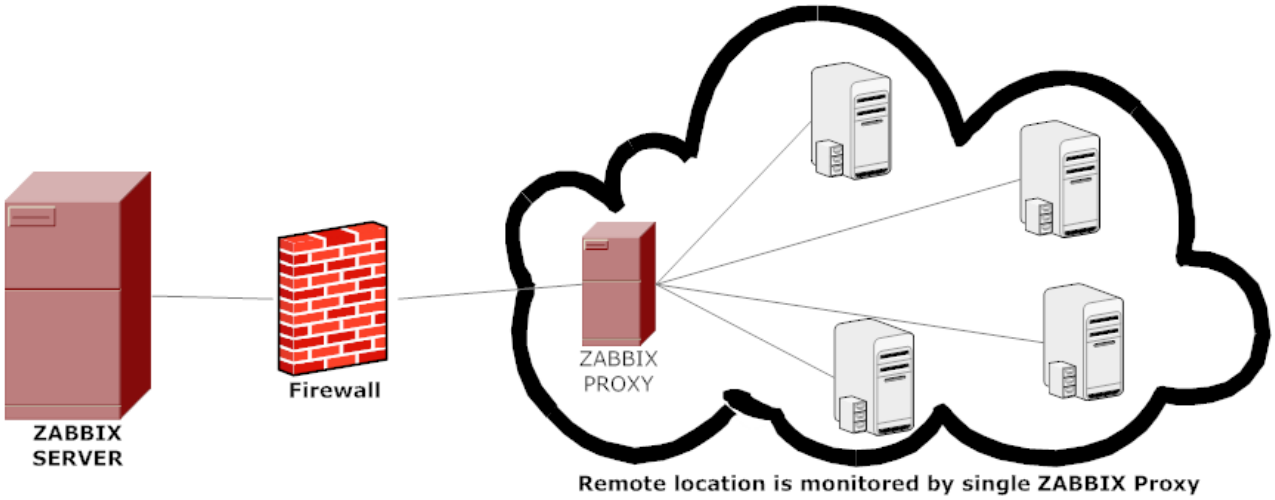
La macchina scelta come server principale di zabbix è serverCentrale-ubuntu.

3. Monitoraggio distribuito

Al fine di semplificare la struttura del sistema di monitoraggio quando sono in gioco un numero elevato di host, zabbix consente di implementare un monitoraggio distribuito, nel quale gli host sono collocati in una struttura gerarchica, inviando comunque tutti i dati in un server centrale, dal quale è possibile visionare lo stato dell'intero sistema. La logica è dunque quella di evitare che sia il solo nodo centrale a gestire direttamente la configurazione dei nodi da monitorare, introducendo dei server "intermedi" che gestiscano le configurazioni e i dati provenienti da un sottoinsieme degli agent e inviino poi i dati collezionati al server centrale. Zabbix offre due implementazioni per il monitoraggio distribuito: l'utilizzo di proxy e la configurazione padre-figlio.

- **Proxy:**

Consiste nell'installazione nel nodo intermedio del processo zabbix-proxy (installabile anche dal gestore di pacchetti), che riceve i dati dagli agent e li invia al server centrale.



Possibile struttura di monitoraggio tramite proxy (fonte: documentazione zabbix)

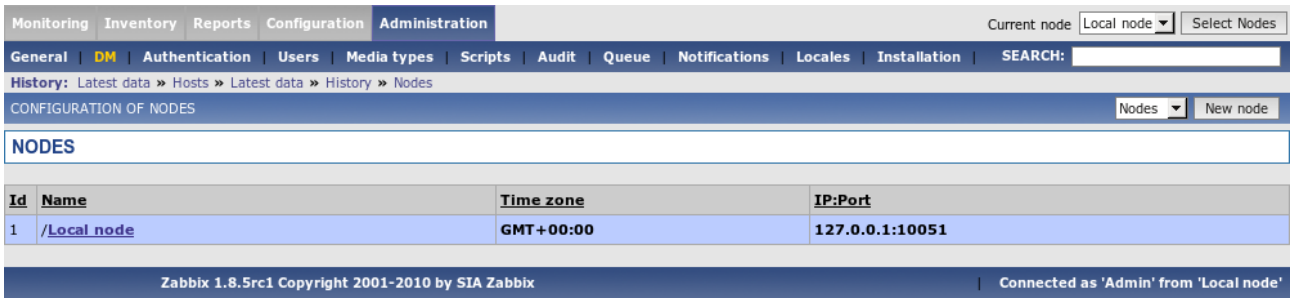
Il proxy è indicato in situazioni in cui si ha un gruppo di host all'interno di una rete locale, mentre il server centrale è collocato altrove: in questo caso la connessione diretta con tutti gli agent viene sostituita da una sola connessione con il nodo intermedio. Un ulteriore vantaggio del proxy consiste nel fatto che, anche se il server centrale è offline, il proxy continua a monitorare, per inviare i dati quando la connessione è ripristinata.

Tuttavia il principale svantaggio, motivo per cui lo abbiamo scartato, è che il proxy non dispone di interfaccia grafica: tutta la configurazione deve essere impostata dal server centrale e non si ha nessun report sullo stato attuale del sistema dal nodo intermedio.

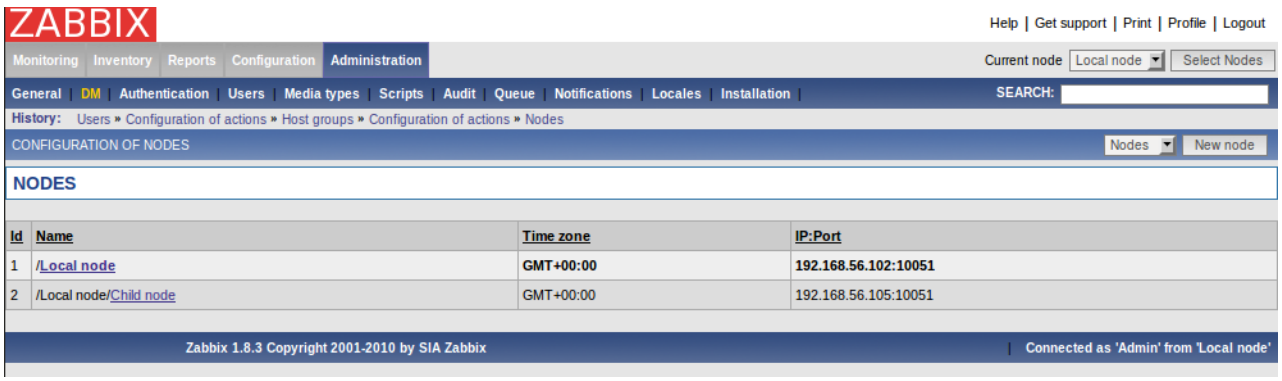
- **Distributed Monitoring:**

Nella nostra situazione, essendo necessario poter ottenere un report dello stato direttamente dal server intermedio, siamo stati obbligati a scegliere la configurazione master-child. La configurazione del monitoraggio distribuito è gestibile nella sezione Administration→DM della GUI zabbix.

La configurazione iniziale di zabbix è riportata nell'immagine che segue:



Nella configurazione iniziale, dove il monitoraggio distribuito non è ancora stato impostato, è presente un solo nodo, con id 1, all'indirizzo 127.0.0.1. Dopo aver settato correttamente il monitoraggio distribuito, la configurazione nel server-centrale dovrebbe apparire come nella figura sotto: il nodo locale è riportato con id 1, (ricordiamo che il server-centrale ha ip 192.168.56.102), mentre il server intermedio (cha ha ip 192.168.56.105) è configurato con id 2 e riportato come child node del nodo locale, che funge quindi da nodo master.



La configurazione è estremamente semplificata e prevede:

1. Installazione di zabbix-server su entrambi i nodi
2. Modifica del NodeID nel file di configurazione (nel nostro caso /etc/zabbix/zabbix_server.conf): NodeID=1 nel nodo master e NodeID=2 nel nodo child
3. Esecuzione del seguente comando, per convertire gli ID del database
   Per il nodo master:

```
./zabbix_server -n 1 -c /etc/zabbix/zabbix_server.conf
```

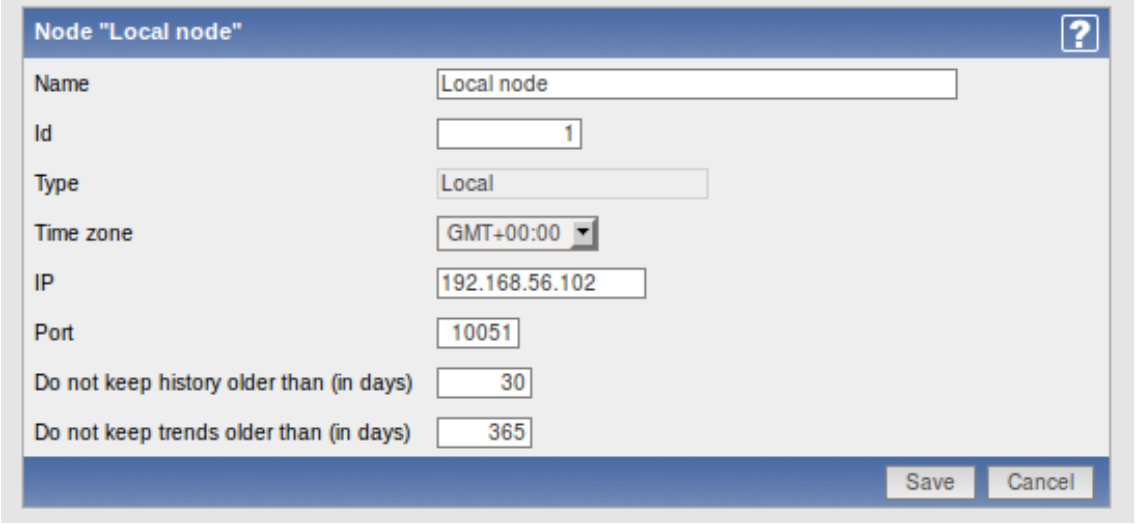
   Per il nodo child:

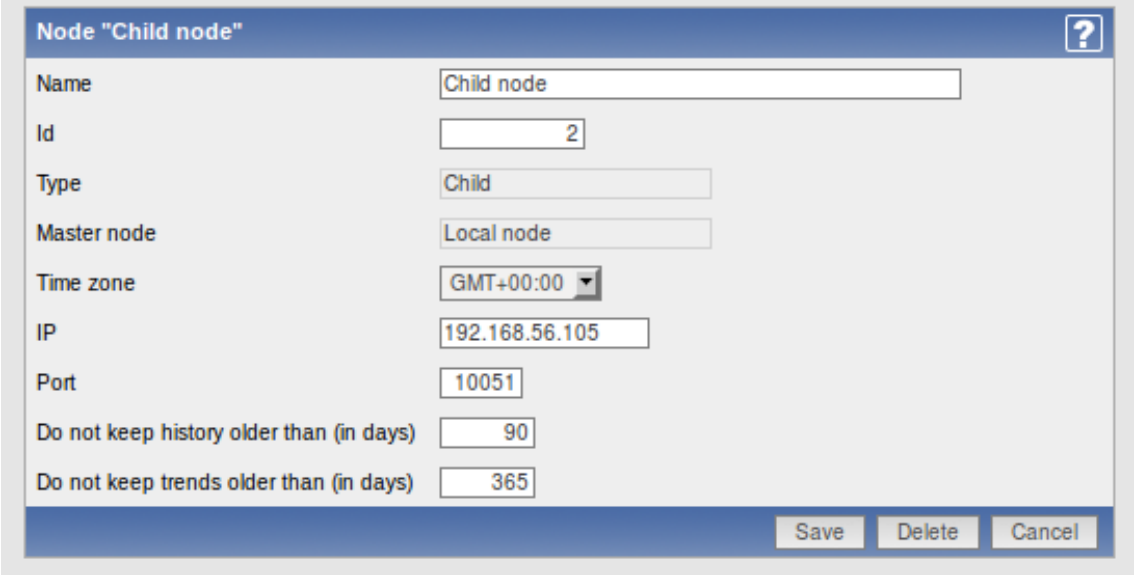
```
./zabbix_server -n 2 -c /etc/zabbix/zabbix_server.conf
```

4. Configurazione dei parametri dei nodi:

- nel server centrale deve essere creato un nuovo nodo (Administration→DM→New node) avente l'indirizzo ip 192.168.56.105 e configurato come nodo child, avente come master il nodo local node
- nel server intermedio deve essere modificato il nodo già presente impostando l'id a 2. Inoltre deve essere creato un nuovo nodo con id 1 di tipo master avente indirizzo ip 192.168.56.102.

Si riporta a titolo esemplificativo la configurazione dei due nodi come nel server centrale (la configurazione nel nodo intermedio è duale):





In questa configurazione tutti le informazioni di configurazione e i report dei vari agent vengono visualizzati anche dal server centrale: in particolare, le informazioni di configurazione vengono inviate ogni 120 secondi, mentre gli eventi ogni 10 secondi.

Una grossa limitazione di entrambe le soluzioni di monitoraggio distribuito è che non è possibile trasmettere al server centrale soltanto una parte degli item monitorati, venendo sempre e comunque trasferiti tutti i dati collezionati dal server intermedio: in questo senso il bandwidth richiesto è notevole e non adatto in presenza di una connessione lenta tra centrale e intermedio.

4. Cenni sull' uso delle API

Funzionalità introdotta dalla versione 1.8, consente all' amministratore di modificare il sistema di monitoraggio tramite il protocollo JSON RPC, oltre che dalla solita interfaccia grafica.

I passi da seguire sono i seguenti (come esempio analizziamo lo script per l' attivazione del monitoraggio diretto):

1. Creare l'oggetto JSON che descrive il comando che si vuole eseguire
2. Inviare tale oggetto tramite metodo POST all' indirizzo: http://host_ip/zabbix/api_jsonrpcc.php
3. Ottenere la risposta nel formato JSON

La struttura base di ogni richiesta JSON è la seguente:

```
{
      "jsonrpc":"2.0",
      "method":"method.name",
      "params":
      {
           "param_1_name":"param_1_value",
           "param_2_name":"param_2_value"
      },
      "id":1,
      "auth":159121b60d19a9b4b55d49e30cf12b81
}
```

Analizziamolo linea per linea:

- `"jsonrpc":"2.0"` – Questo è un parametro standard che identifica la versione del protocollo JSON RPC: rimarrà invariata per tutte le richieste
- `"method":"method.name"` – Indica l'operazione che si intende eseguire: per una visione completa delle operazioni eseguibili tramite API è possibile consultare la documentazione zabbix, in particolare la sezione *http://www.zabbix.com/documentation/1.8/api* e le relative sotto-sezioni, che riportano i comandi catalogati secondo l'ambito di interesse.
- `"params":{…}` – Questo campo contiene I parametri richiesti per il metodo selezionato
- `"id":1` – Il campo id viene utilizzato per ricondurre ogni risposta alla richiesta che l'ha generata: tale funzionalità è di particolare utilità nel caso in cui si inviano più richieste, per distinguere le risposte
- `"auth":159121b60d19a9b4b55d49e30cf12b81` – Questo parametro è necessario per identificare l'utente che intende accedere alle API. Si veda sotto per maggiori dettagli.

Siccome le API sono state progettate come uno strumento per gestire la configurazione del sistema, al fine di impedire accessi non autorizzati, è necessario ottenere, prima di lanciare qualsiasi altra richiesta, una chiave di autenticazione. Come ulteriore misura di sicurezza, l'utente che effettua l'autenticazione deve essere abilitato all'uso delle API: tale opzione per default è disabilitata per tutti gli utenti (incluso l'utente predefinito Admin) e può essere modificata dalla GUI zabbix, in Administration→Users aggiungendo l'utente prescelto al gruppo "API access". Per ottenere la chiave abbiamo utilizzato il metodo "user.authenticate", che accetta come parametri il nome utente e la password dell'utente e restituisce la chiave da usarsi nei successivi comandi.

A titolo esemplificativo analizziamo lo script che abbiamo usato per abilitare/disabilitare il monitoraggio degli host (per maggiori dettagli si veda la sezione successiva).

L'oggetto JSON inviato per l'autenticazione è il seguente (contrariamente agli altri comandi, il campo auth è presente ma lasciato vuoto, per ovvie ragioni):

```
{
	"params":
	{
		"password": "zabbix",
		"user": "Admin"
	},
	"jsonrpc": "2.0",
	"method": "user.authenticate",
	"auth": "",
	"id": 0
}
```

Una volta ottenuta la chiave (nel nostro caso l'abbiamo salvata nella variabile AUTH_TOKEN), la usiamo nel secondo comando, per modificare lo stato dell' host tramite i due parametri con cui viene invocato lo script:

```
{
	"jsonrpc":"2.0",
	"method": "host.update",
	"params":
	{
		"hostid": "$1",
		"status": $2
	},
	"auth":"$AUTH_TOKEN",
	"id":2
}
```

Nel nostro script l'host-id e lo status desiderato vengono passati come parametri al momento dell'invocazione dello script, quindi sono il primo e il secondo parametro. In particolare è possibile reperire l'host-id o tramite API o consultando direttamente il database MySQL su cui si appoggia zabbix. Il parametro status invece è un intero che indica lo stato monitorato/non monitorato dell'host: come si scopre consultando il database, 0 significa abilitato, mentre 1 disabilitato.

5. Recovery in caso di guasto del server intermedio

Il nodo centrale, trovandoci in una situazione di monitoraggio distribuito, riceve i dati degli agent solamente attraverso il server intermedio: il processo zabbix-server attivo nel server intermedio raccoglie i dati dagli agent e li invia periodicamente al server centrale. Se quindi il processo risultasse inattivo o il server intermedio irraggiungibile, il nodo centrale non è più in grado di monitorare gli host. Per ovviare a questo problema si è introdotta una configurazione di "monitoraggio diretto", ossia il server centrale che monitora direttamente gli agent senza il server intermedio, nel caso di guasto di quest'ultimo.

In primis abbiamo creato direttamente nel server centrale le tre macchine virtuali (agentWindows, agentMandrake, agentDebian) che sono presenti nel server intermedio, ma le abbiamo lasciate inattive, per abilitarle solo nel caso in cui non si ricevano dati dal server intermedio. Siccome si presume che questi tre host debbano essere utilizzati soltanto in situazioni di emergenza, è stata lasciata all'amministrazione la responsabilità di configurare manualmente il tutto. Ciò significa che se viene aggiunto un host al server intermedio, questo verrà visualizzato anche nel server centrale, per via della configurazione master-child, ma non verrà visto in nessun modo in caso di guasto del server intermedio.

I passi per la configurazione sono i seguenti:

1. Come prima cosa è stato necessario ridefinire nel server centrale il trigger "Zabbix_server is not running on serverIntermedio", in modo che restituisca PROBLEM sia quando il nodo intermedio non è raggiungibile sia quando il processo zabbix-server è inattivo. Quindi l'espressione è:

```
{Template_Linux:proc.num[zabbix_server].last(0)}<1 |
{Template_Linux:status.last(0)}=2
```

2. E' stata creata nel server centrale un'azione (da Configuration→Actions→Create new action) da invocare quando il valore del trigger diventa PROBLEM, e configurata affinché invii un'email all'amministratore del sistema e lanci un comando (si veda la figura per maggiori dettagli).

3. E' stato creato lo script (/usr/bin/json.sh) da invocare:

```
#!/bin/bash

ZABBIX_USER='Admin'
ZABBIX_PASS='zabbix'

# Authenticate with Zabbix API
authenticate() {
curl -i -X POST -H 'Content-Type: application/json-rpc' -d "{\"params\":
{\"password\": \"$ZABBIX_PASS\", \"user\": \"$ZABBIX_USER\"}, \"jsonrpc\":
\"2.0\", \"method\": \"user.authenticate\",\"auth\": \"\", \"id\": 0}"
http://localhost/zabbix/api_jsonrpc.php | grep -Eo 'Set-Cookie:
zbx_sessionid=.+' | head -n 1 | cut -d '=' -f 2 | tr -d '\r'
}
AUTH_TOKEN=$(authenticate)

curl -i -X POST -H 'Content-Type: application/json-rpc' -d
"{\"jsonrpc\":\"2.0\", \"method\":\"host.update\", \"params\":{
\"hostid\": \"$1\", \"status\": $2 }, \"auth\":\"$AUTH_TOKEN\", \"id\":2}"
http://localhost/zabbix/api_jsonrpc.php
```

Si è scelto di creare uno script in shell in quanto il server-centrale su cui deve essere eseguito è una macchina linux, quindi supporta nativamente il linguaggio shell, senza richidere l'installazione di altri programmi, quali perl o java. Lo script consente, tramite le API di zabbix, di abilitare o disabilitare il monitoraggio di un host attraverso due parametri: il primo consiste nell' hostid del nodo che si intende modificare e il secondo consente di indicare se si intende abilitare l' host (0=monitorato) o disabilitarlo (1=inattivo). Come visto nella sezione precedente, è necessario inviare la richiesta JSON tramite metodo POST: a questo scopo si è scelto di utilizzare curl, che è possibile lanciare direttamente da shell.

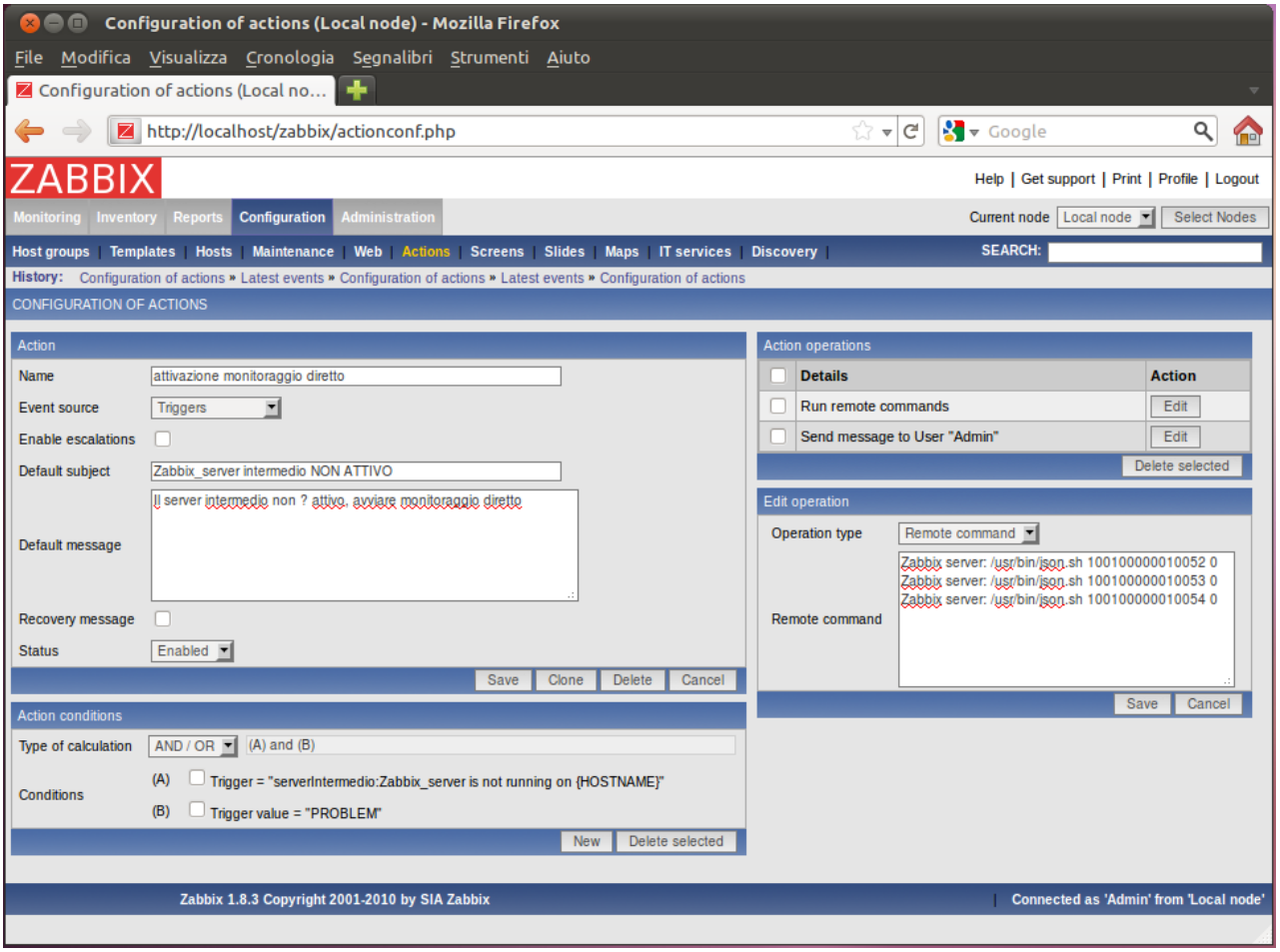
Lo scipt consiste sostanzialmente in due parti:

1. la prima consiste nell'invio della richiesta di autenticazione e successiva elaborazione testuale della risposta ottenuta al fine di estrarre la chiave di autenticazione e memorizzarla nella variabile AUTH_TOKEN.
2. La seconda parte consiste nell'invio della richiesta per modificare lo stato dell'host, secondo i parametri passati al momento dell'invocazione dello script

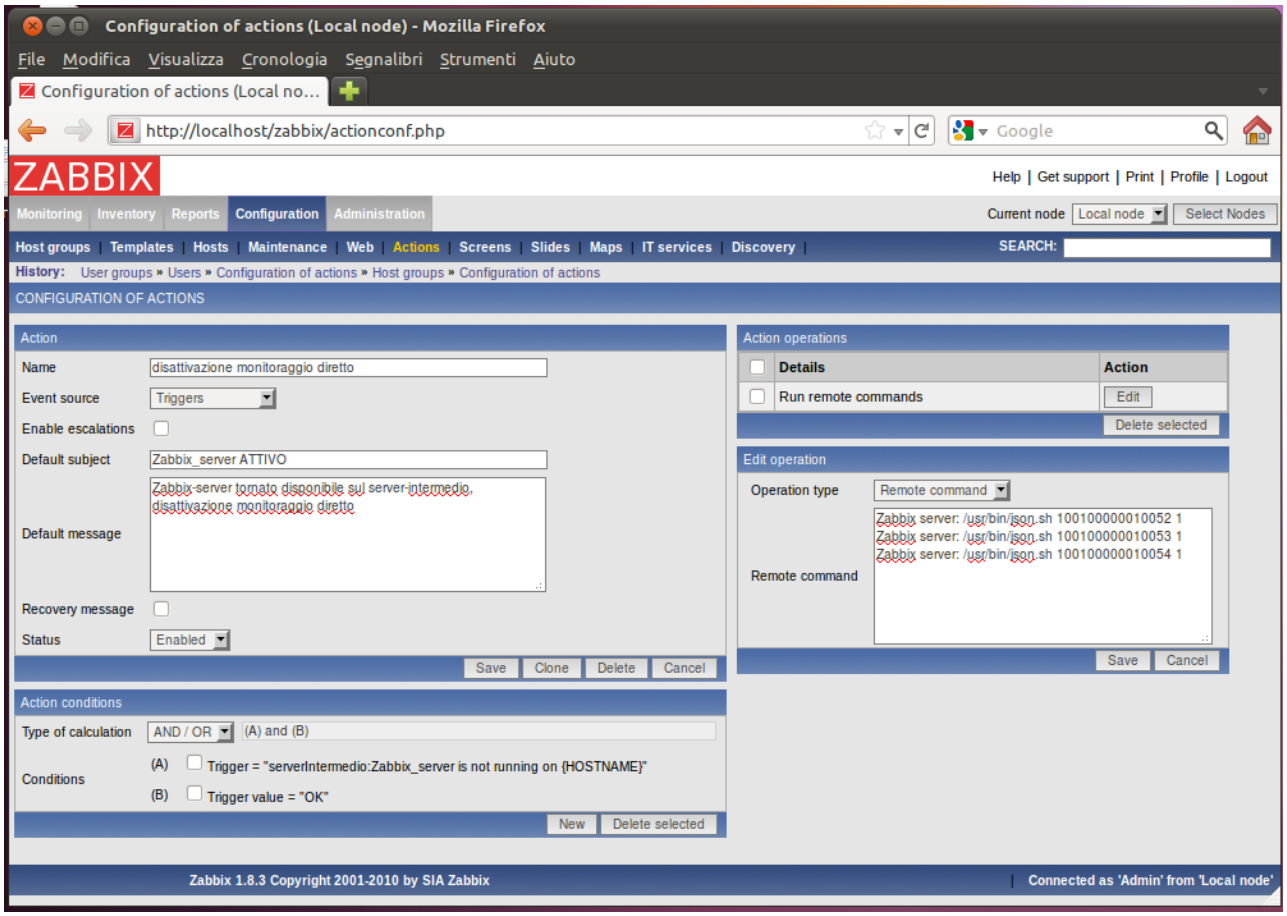
Entrambe le richieste JSON vengono inviate a localhost, in quanto la action è configurata sul server-centrale e lo script da invocare si trova sulla medesima macchina.

4. E' stata creata un'action opposta per disabilitare il monitoraggio diretto quando il server intermedio torna disponibile, basata sullo stesso trigger e sullo stesso script. In particolare tale action viene eseguita quando lo stato del trigger diventa OK e esegue lo stesso script, passando però come secondo parametro 1, ossia il comando per rendere inattivo l'host.

Si riporta la configurazione della action che attiva il monitoraggio diretto:



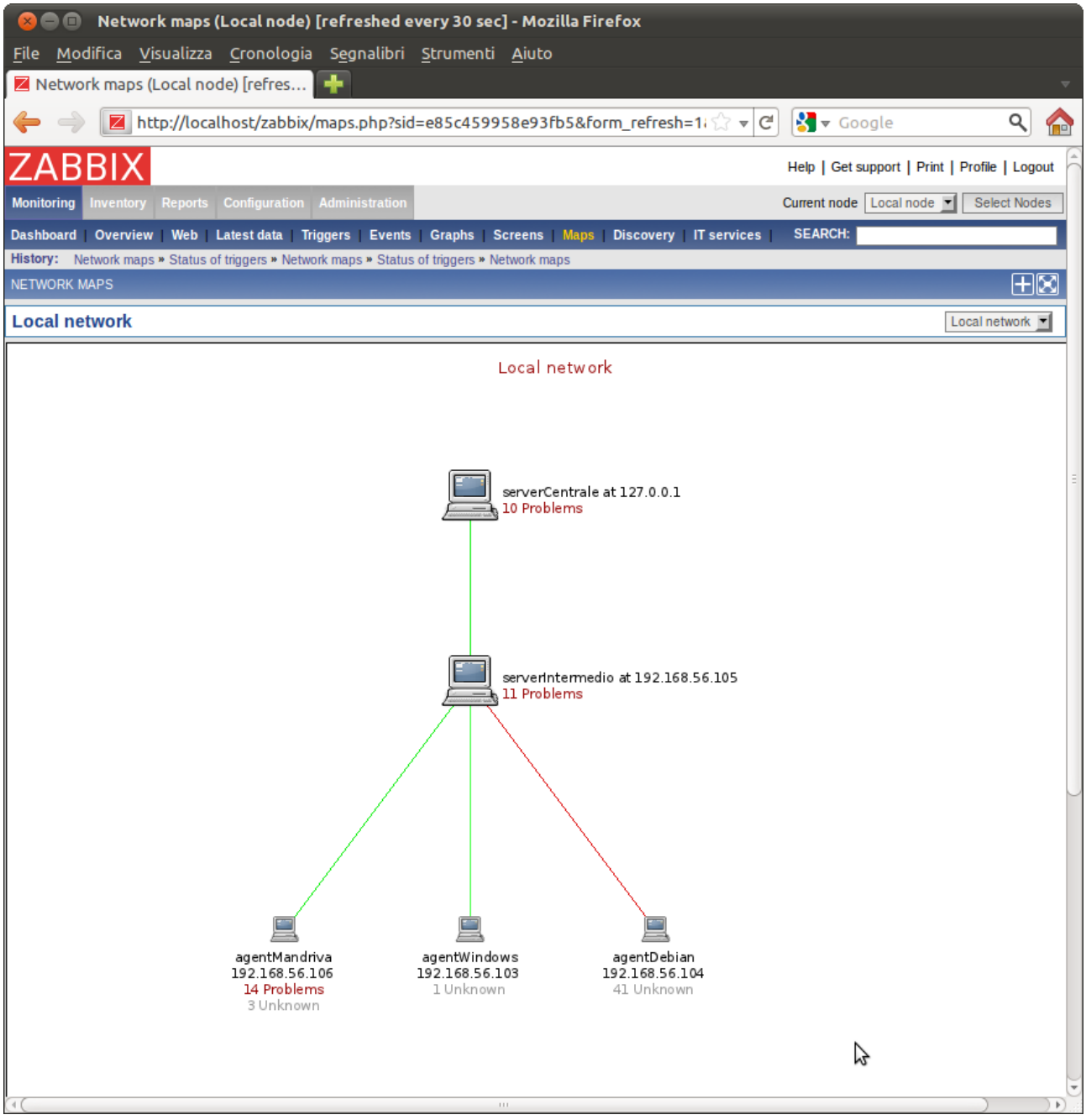
Sotto la action opposta, che disabilita i tre host quando il server intermedio torna disponibile.



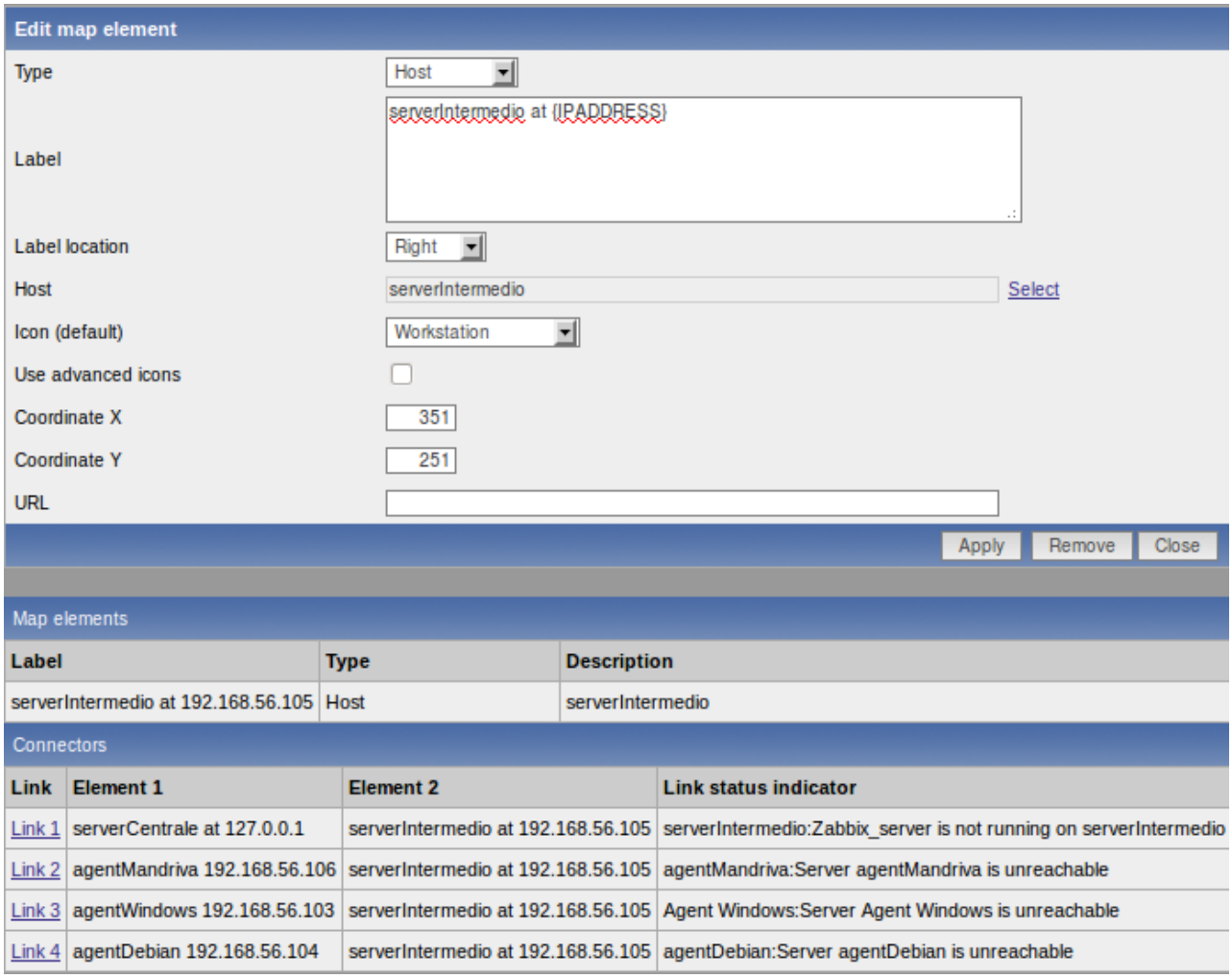
Le uniche differenze tra le due actions sono il valore del trigger e lo stato dell'host da modificare, per il resto sono sostanzialmente identiche.

6. Mappa della rete

Zabbix consente la creazione di mappe dinamiche del sistema, in modo da fornire all'amministratore una visione complessiva della situazione attuale. Al primo utilizzo zabbix presenta già una mappa predefinita contenente il solo host locale: nel nostro caso abbiamo scelto di implementare la mappa del sistema partendo dalla mappa predefinita nel serverCentrale ("Local network") e adattandola al nostro caso aggiungendo gli host e i collegamenti.



La mappa risulta estremamente personalizzabile, in quanto è possibile modificare la posizione, il testo informativo da visualizzare (anche con macro) e l'icona (disponibili icone che variano a seconda dello stato dell' host). A partire dalla versione 1.8 è stata introdotta la possibilità di aggiungere e ricollocare elementi tramite drag & drop, mentre prima era necessario introdurre manualmente le coordinate.



Come si vede nella figura, relativa al serverIntermedio, i collegamenti sono configurabili in modo che cambino colore in base al valore di un trigger: nel nostro caso, i link tra il serverIntermedio e i tre agent variano in base al trigger “Server {HOSTNAME} is unreachable”, mentre il link tra serverIntermedio e serverCentrale varia stato in base al trigger “Zabbix_server is not running on serverIntermedio”, da noi ridefinito come specificato in precedenza. Si ha quindi che se il collegamento tra serverCentrale e serverIntermedio è di colore rosso, è attivo il monitoraggio diretto degli agent.

E' importante sottolineare che la consistenza della mappa è totalmente nelle mani dell'amministratore del sistema, in quanto zabbix non impedisce in alcun modo configurazioni totalmente illogiche: potrei ad esempio far variare lo stato di un collegamento tra due host utilizzando un trigger relativo ad un terzo host completamente scollegato, senza che zabbix imponga nessuna logica.

7. Monitoraggio delle trap SNMP

Zabbix non offre un completo supporto al monitoraggio delle trap SNMP ma dota gli amministratori di uno script che permette, tramite la configurazione di alcuni parametri, di ottenere una possibile soluzione a questa mancanza. Sebbene lo script si sia dimostrato funzionante abbiamo ritenuto opportuna la realizzazione di alcune modifiche a quest'ultimo per rendere il monitoraggio più completo.

Configurazione dell'snmptrapd

Per prima cosa è risultato necessario configurare il demone che attende la ricezione delle trap. Sono state aggiunte queste due righe nel file /etc/snmp/snmptrapd.conf:

```
disableAuthorization yes
traphandle default /bin/bash /bin/snmptrap.sh
```

La prima riga permette di disabilitare il controllo degli accessi per evitare di bloccare le trap in arrivo mentre la seconda riga invoca l'handler che gestirà tutte le trap inviate alla macchina server. Tale handler sarà appunto il nostro script personalizzato, inserito all'interno della cartella bin della macchina e nominato snmptrap.sh.

Realizzazione dello script snmptrap.sh

Lo script è una versione modificata dello script presente nella cartella /misc/snmptrap del codice sorgente di Zabbix e da noi inserito all'interno della cartella bin nel server centrale. Il codice di tale script è il seguente:

```
#!/bin/bash

# PARTE 1
ZABBIX_SERVER="127.0.0.1"
ZABBIX_PORT="10051"
ZABBIX_SENDER="/usr/bin/zabbix_sender"
KEY="snmptraps"
DEFAULT_HOST="snmptraps"
MYSQL_USER="root"
MYSQL_PASS="root"
LOG_FILE="/tmp/testtrap.txt"

# PARTE 2
read hostname
read ip
read uptime
read oid
read address
read community
read enterprise

# PARTE 3

oid=`echo $oid|cut -f2 -d' '`
ip=`echo $ip|cut -d'[' -f2|cut -d']' -f1`
HOST=`mysql --batch --silent -u"$MYSQL_USER" -p"$MYSQL_PASS" -e "SELECT host
FROM hosts WHERE ip='"$ip"'" zabbix | tr "\n" "|" | cut -d"|" -f1`
```

```
# PARTE 4

if test –z $HOST
      then
            str="oid=$oid-ip=$ip-error_type=host_not_found_on_zabbix"
            $ZABBIX_SENDER -z $ZABBIX_SERVER -p $ZABBIX_PORT -s $DEFAULT_HOST -k
$KEY -o \"$str\" >> "$LOG_FILE"
else
      HOSTID=`mysql --batch --silent -u"$MYSQL_USER" -p"$MYSQL_PASS" -e "SELECT
hostid FROM hosts WHERE ip='"$ip"'" zabbix | tr "\n" "|" | cut -d"|" -f1`
      CONDITION=`mysql --batch --silent -u"$MYSQL_USER" -p"$MYSQL_PASS" -e
"SELECT key_ FROM items WHERE hostid='"$HOSTID"' AND key_='"$KEY"'" zabbix | tr
"\n" "|" | cut -d"|" -f1`
      if test -z $CONDITION
            then
                  str="oid=$oid-ip=$ip-error_type=key_not_found_on_host_$HOST"
                  $ZABBIX_SENDER -z $ZABBIX_SERVER -p $ZABBIX_PORT -s
$DEFAULT_HOST -k $KEY -o \"$str\" >> "$LOG_FILE"
      else
            str="$oid"
            $ZABBIX_SENDER -z $ZABBIX_SERVER -p $ZABBIX_PORT -s $HOST -k $KEY -o
\"$str\" >> "$LOG_FILE"
      fi
fi
```

Elenchiamo ora le funzionalità delle diverse parti dello script:

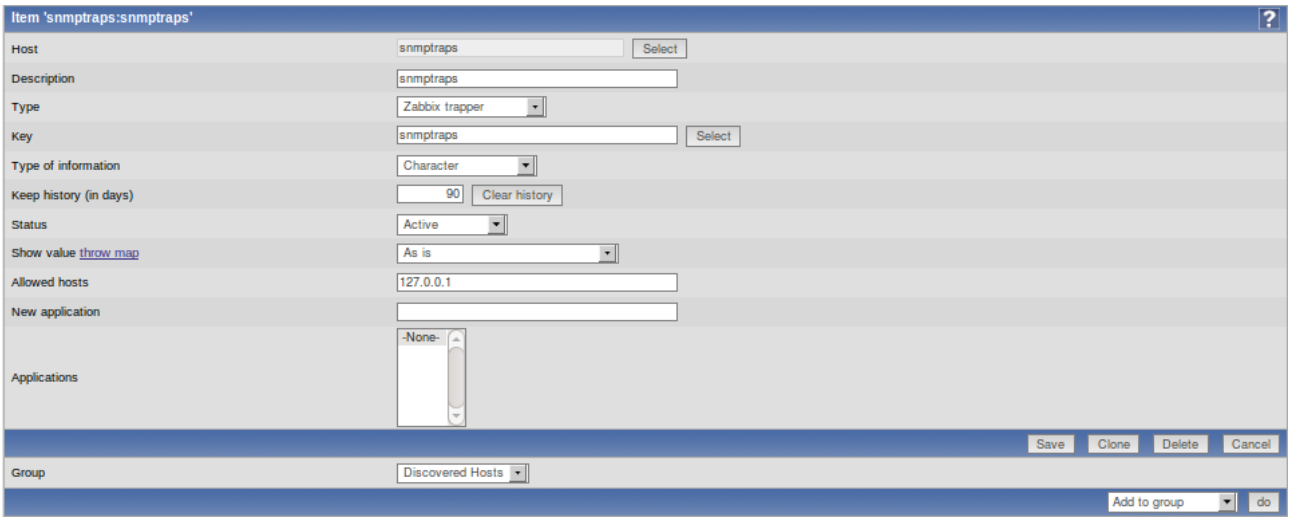
1. Si impostano i diversi parametri di configurazione a seconda delle esigenze, fornendo un metodo per personalizzare lo script rendendolo più comodo da utilizzare.
2. Si leggono i dati riguardanti la trap SNMP ricevuta, di cui verranno effettivamente utilizzati soltanto l'indirizzo ip dell'host che ha inviato la trap e l'oid della trap stessa.
3. Si manipolano tali dati e si effettua una query SQL nel database di zabbix per rintracciare l'hostname corrispondente all'ip che ha inviato la trap.
4. Sono stati applicati dei controlli per consentire di non perdere la trap ricevuta per alcun motivo. Nel caso in cui l'host non sia presente all'interno di zabbix viene creata una stringa contenente oid, ip e il tipo di errore associato e tale stringa viene inviata tramite zabbix_sender al sistema. Occorrerà a priori aver specificato un host generico ed una item in grado di raccogliere tali dati (host="snmptraps", item con key="snmptraps" a default). La stessa cosa viene fatta nel caso in cui l'host sia presente all'interno di zabbix ma in esso non sia stata inserita l'item che raccoglie i dati della trap. Nel caso invece in cui tutto sia presente è proprio a quest'ultima item (con key="snmptraps" a default) che verranno inviati i dati.

Configurazione di Zabbix

Per far si che il monitoraggio venga effettivamente eseguito occorre che i dati relativi alla ricezione delle trap vengano memorizzati all'interno di Zabbix. Un modo comune per realizzare ciò è l'inserimento degli stessi in un'item relativa ad un host specifico (nella GUI di zabbix nella sessione Configuration → Hosts, selezionando in seguito il link Items relativo all'host in questione e clickando poi su Create Item). Tale item riceverà i dati e su di essa potranno essere configurati degli opportuni trigger.

Nel nostro caso tale item andrà aggiunta a tutti gli host in grado di inviare trap SNMP a zabbix, poichè lo script sopra descritto invierà i dati all'item dello specifico host.

Sarà anche necessario realizzare un host generico che raccoglierà tutte le trap ignote o per cui non si sia definita l'item apposita all'interno dell'host. Così facendo non verrà persa alcuna trap, che è il comportamento da noi desiderato.

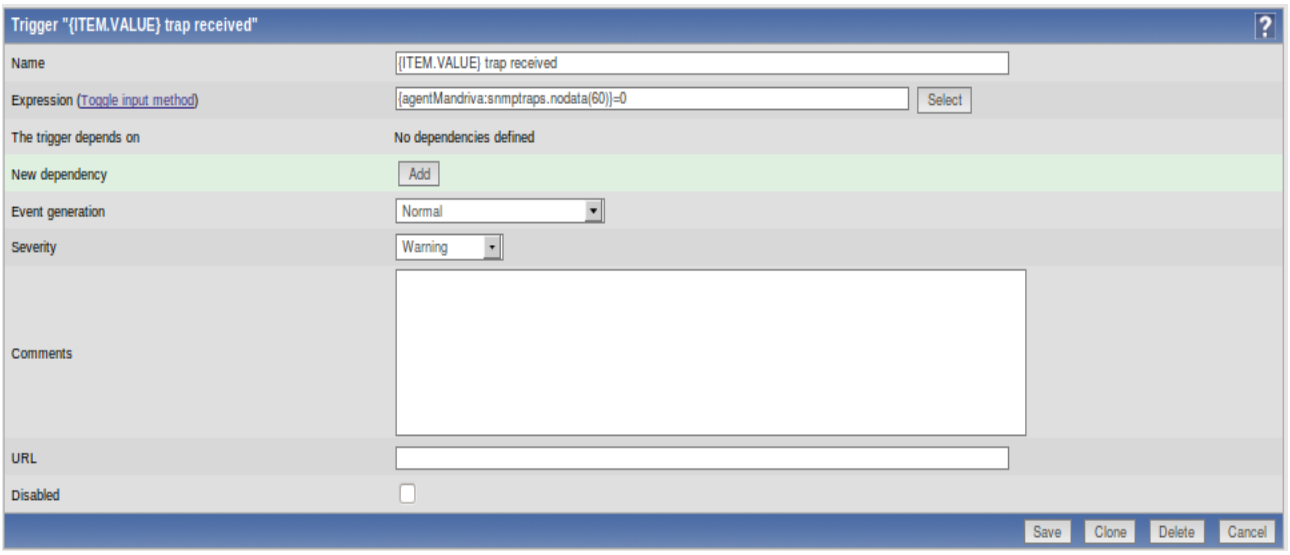


Creazione dell'item nel caso dell'host generico snmptraps.

L'item dovrà essere impostata con un valore dell'attributo Key pari a quanto specificato nel nostro script, ovvero "snmptraps". Essa dovrà essere di tipo "Zabbix trapper" e in grado di gestire informazioni di tipo "Character". Inoltre nel campo Allowed hosts dovrà essere indicato l'indirizzo ip locale, ovvero 127.0.0.1.

Una volta realizzato ciò non resta da fare altro che impostare dei trigger in grado di agire su tale item. Nel nostro caso abbiamo ipotizzato la realizzazione di un trigger che scatta non appena l'item riceve dei dati e rimane attivo per un determinato periodo di tempo (nell'esempio per 60 secondi). Il nome di tale trigger varia in base al valore dell'item, in modo tale da mostrare l'oid della trap che è stata inviata al server.

Di seguito è mostrato uno screenshot riguardante la creazione di tale trigger con riferimento all'agent Mandriva.



Impostazione del trigger per la ricezione delle trap SNMP.