Procedura di autodiscovery dei nodi VMware

Di Riccardo Corsanici

In questo articolo verrà decritta una procedura personalizzata per il discovery automatico dei sistemi presenti in una farm virtuale VMware.

Contesto

La procedura di scansione automatica della piattaforma virtuale è stata sviluppata con lo scopo di fornire una soluzione di supporto per gli amministratori di sistema. Il problema da gestire era la difficoltà nel tenere traccia dei frequenti cambiamenti operati sull'infrastruttura, aggiungendo, rimuovendo e riconfigurando sistemi virtuali con cadenza quotidiana.

In questo contesto gli amministratori dovevano procedere ad un confronto “manuale” tra i sistemi presenti nel Virtual Center e quelli iscritti al monitoring e quindi mappati su Neteye, con il rischio di incorrere in errori e dimenticanze. Da una parte esisteva la concreta possibilità che un ambiente critico sfuggisse all'attenzione dei sistemisti e non venisse quindi controllato (con le ripercussioni che è possibile immaginare), dall'altra che ambienti dismessi rimanessero presenti nella configurazione di Neteye e risultassero di conseguenza “down” in quanto spenti, invalidando in questo modo i report e le statistiche di uptime della piattaforma.

La soluzione realizzata

La soluzione che abbiamo ingegnerizzato si compone di diverse procedure batch che svolgono il lavoro di screening e di confronto tra la piattaforma virtuale ed i nodi iscritti al monitoring, ed è stata gradualmente arricchita da diversi moduli ed espansioni, compresa un'interfaccia WEB (poi divenuta piuttosto complessa) che agevolasse e rendesse più semplici ed immediate le azioni di configurazione più comuni.

Il primo modulo sviluppato è stato quello delegato all'auto discovery dei nodi presenti nel Virtual Center. Le funzioni implementate prevedono la scansione della farm virtuale per identificare e tenere traccia di:

- Cluster (insiemi di nodi fisici. Nel nostro caso ogni cluster è delegato alla gestione di un Data Center).
- Server ESX (nodi fisici che compongono i cluster VMware).
- Nodi virtuali in carico ad ogni ESX.

Scansione della farm virtuale

Per eseguire la scansione abbiamo scelto di utilizzare l'insieme di script e strumenti di utilità offerti dalla suite “vmware command line interface”, o `vmware-vcli`, distribuita direttamente da VMware. Questa scelta è principalmente riconducibile alla necessità di poter disporre di una soluzione standardizzata che rimanesse compatibile e supportata al variare delle release del Virtual Center. In cambio siamo dovuti scendere a patti con le possibilità offerte dagli strumenti command line di VMware, che non sempre offrono funzionalità verticali e mirate rispetto alle nostre necessità e che in alcuni aspetti risultano addirittura inefficienti rispetto a procedure custom.

Lo strumento principale per la scansione è lo script perl (fornito con i vmware-vcli) “vidiscovery.pl”, che è in grado di recuperare mediante chiamate SOAP diverse informazioni interfacciandosi direttamente con l'infrastructure server attraverso il protocollo HTTPS.

Si prenda in considerazione il seguente codice:

```
#!/bin/bash
#
```

```
#####################################################
## "run_disconver_vm.sh"
##
## esegue la scansione della farm virtuale e salva
#####################################################

vidiscovery="/opt/vmcli/lib/vmware-vcli/apps/general/vidiscovery.pl"
discover_tmpfile="${discover_outdir}/discovery_$(date +%Y-%m-%d_%H-%M).tmp"
url="https://<URL>/sdk/vimService"
username="<USER>"
password="<PASSWORD>"

${vidiscovery}  --url ${url}            \
                --username ${username}  \
                --password ${password}  \
                --entityname Global     \
                --managedentity folder  > ${discover_tmpfile} 2>&1
```

Lo scopo di questa routine è ottenere dal server di infrastruttura l'elenco degli oggetti presenti nel Virtual Center. Il funzionamento è semplice: viene effettuata una chiamata HTTPS all'URL su cui il server di infrastruttura stesso è in ascolto (variabile `${url}`). La chiamata comprende le informazioni necessarie per l'autenticazione (variabili `${username}` e `${password}`) ed il nome dell'entity VMware da interrogare (“`global`”, in questo caso).

Il listato prodotto viene poi reindirizzato in un file di testo (variabile `${discover_tmpfile}`) per una successiva elaborazione. L'esecuzione del “`vdiscovery.pl`” impiega un certo tempo, dipendente dal numero di oggetti presenti e quindi direttamente proporzionale alle dimensioni della farm virtuale.

Di seguito un estratto del contenuto del file prodotto:

```
**********Folder Global**********
Folder : Global
  Folder : vm
    Folder : Accellion
    Folder : BigFix
    Folder : Bilanciatore
    Folder : CallManager-Videoconference
    Folder : DeepSecurity
    Folder : Domino
    Folder : F5-BigIP
    Folder : FD
      Folder : Digital Book
      Folder : GIS
      Folder : HRIS
      Folder : Smart Delivery
    Folder : ITM
    Folder : Network Management
    Folder : PCTest
    Folder : PrinterServer
    Folder : QLik
    Folder : Remedy
    Folder : SMTP
    Folder : SOA
    Folder : SSO
    Folder : Tivoli User Experience
    Folder : TrendMicro
    Folder : WebSSO
    Folder : WebServer
    Folder : Websense
    Folder : WiFiPublic
    Folder : Wsus-Client
    Folder : Smart Delivery
    Folder : HRIS
    Folder : GIS
  Folder : host
    Folder : DC Breganze Campus-Frontier
      Host : clvmwp03.hq.otb.net
        VM : SDHRSP01
        VM : SDGISP01
        VM : SDFDGP01
        VM : WPBNXSP01
        VM : CLBDSPS03
        VM : CLBPRTP02
```

```
        VM : EBQKRP01
        VM : B2FDBDP01
        VM : B2EPRXP02
        VM : CLBTDSP01
        VM : EBCVWP01
        VM : EBESSP02
        VM : CLBWPXP01
        VM : CLBWAVP01
        VM : CLBWSCP01
        VM : EBWPSP02
[...]
```

Discovery dei Data Center

Completata la scansione la procedura esegue il parsing del file per identificare e tracciare le seguenti tipologie di entità VMware:

1. Data Center (cluster VMware)
2. Host fisici (server ESX)
3. Nodi virtuali (VM)

Il primo passaggio consiste nell'estrazione dei Data Center:

```
OLD_IFS=${IFS}
IFS=$'\n'
for DC in $(cat ${discover_tmpfile} |grep "Folder : DC"| cut -d":" -f2 | sort -u)
do
        [...]
done
```

Si noti che la variabile `$IFS` (che identifica il separatore di campo) viene ridefinita in modo da utilizzare il “fine linea” come marcatore (per sicurezza il precedente valore viene salvato nella variabile `$OLD_IFS`).

Reimpostata `$IFS` il cliclo “`for`” successivo può processare il file di output prodotto dal discovery prendendo in esame una riga alla volta. L'insieme delle linee da processare viene estratto con “`grep`”, la stringa di ricerca è “`Folder : DC`”, che permette di identificare le entità di tipologia “Data Center”.

Nel nostro caso gli elementi presi in considerazione dal ciclo sono i seguenti:

```
 DC Breganze Campus-Frontier
 DC Breganze Sala1
 DC Breganze Staging
 DC Molvena SM3
 DC Molvena SM3 Frontier
 DC Molvena SM3 Prod
```

Discovery dei server ESX

Per ogni Data Center viene recuperato l'elenco degli host fisici che compongono il corrispondente cluster VMware. Ecco il codice eseguito:

```
OLD_IFS=${IFS}
IFS=$'\n'
for DC in $(cat ${discover_tmpfile} |grep "Folder : DC"| cut -d":" -f2 | sort -u)
do
        _DC="$(echo ${DC} | sed 's/^ *//;s/ *$//')"
        echo "DATACENTER:$_DC" >> ${discover_outfile}
        for HOST in $(${vidiscovery}    --url ${url}    \
                        --username ${username}  \
                        --password ${password}  \
                        --entityname "${_DC}"   \
                        --managedentity folder  | grep "Host : " | cut -d":" -f2 | sort -u)
        do
                _HOST="$(echo ${HOST} | sed 's/^ *//;s/ *$//')"
                echo "HOST:$_HOST" >> ${discover_outfile}
```

```
                [...]
        done
done
```

Il secondo ciclo “`for`” ha lo scopo di recuperare la lista dei server ESX. Per ottenere lo scopo vengono eseguite delle ulteriori chiamate SOAP al server di infrastruttura, questa volta indicando come nome dell'entità da interrogare il Data Center (`--entityname ${_DC}`).

L'output del secondo ciclo è quindi l'elenco degli host fisici, ad esempio:

```
HOST:clvmwp01.hq.otb.net
HOST:clvmwp02.hq.otb.net
HOST:clvmwp03.hq.otb.net
HOST:brgvmwp01.hq.otb.net
HOST:brgvmwp02.hq.otb.net
HOST:brgvmwp03.hq.otb.net
HOST:brgvmwp04.hq.otb.net
[...]
```

Discovery dei nodi virtuali

Lo step successivo è l'estrazione dell'elenco delle VM in carico a ciascun host fisico. Questo risultato viene conseguito annidando un terzo ciclo `for` nella procedura:

```
for VM in $(${vidiscovery}    --url ${url}    \
        --username ${username}  \
        --password ${password}  \
        --entityname "${_HOST}" \
        --managedentity host | grep "VM : " | cut -d":" -f2 | sort -u)
do
        _VM="$(echo ${VM} | sed 's/^ *//;s/ *$//')"
        echo "VM:$_VM" >> ${discover_outfile}
done
```

Come mostrato nell'esempio la logica è sempre la stessa: per ogni host fisico viene generata una chiamata SOAP che genera l'elenco dei nodi virtuali. Da notare che in questo caso il nome dell'entità interrogata sarà il nome dell'host fisico: “`--entityname ${_HOST}`”.

L'output complessivo sarà un listato con l'elenco di tutti i nodi virtuali in carico ad ogni server ESX. Ad esempio:

```
HOST:clvmwp01.hq.otb.net
VM:B2EPRXP01
VM:CLBBILP01
VM:CLBDSPS01
VM:CLBSSOT01
VM:CLBTAVP01
VM:CLBTAVP02
VM:CLBTBFP01
VM:CLUE3P01
VM:EBAGWP01
VM:EBESSP01
VM:EBFTPP01
VM:EBMTAP01
[...]
HOST:clvmwp02.hq.otb.net
VM:BRHFDF5103
VM:CLBBALP01
VM:CLBDSPS02
VM:CLBPRTP01
VM:CLBWMGP01
VM:CLUESP01
VM:EBFTPP02
VM:EBGIPP01
VM:EBGIPP02
VM:EBHTAP01
[...]
```

Nella procedura tutto l'output prodotto dai tre cicli “`for`” viene rediretto in un file di testo, che verrà utilizzato successivamente da una seconda procedura incaricata di rilevare le differenze

con i sistemi già presenti in Neteye.

Gestire i cambiamenti

Il processo descritto nei paragrafi precedenti ha lo scopo di generare una lista ordinata e facilmente leggibile degli oggetti presenti nella farm virtuale (siano questi Data Center, ESX server o nodi virtuali). Questa lista dovrà poi essere analizzata per individuare i cambiamenti intercorsi dalla scansione precedente. Il risultato della verifica dovrà quindi essere inviato ai responsabili della piattaforma evidenziando quali elementi sono stati aggiunti e quali rimossi.

Per agevolare le operazioni di confronto con il risultato delle scansioni operate in precedenza abbiamo scelto di utilizzare come backend un database MySQL creato all'interno del sistema Neteye. Lo scopo del backend è fornire un repository centralizzato per le informazioni sui nodi virtuali da utilizzare sia per le operazioni di confronto con il risultato dei discovery giornalieri che per lo sviluppo di un'interfaccia WEB che semplifichi l'esecuzione delle azioni più frequenti (ad esempio l'inserimento in Neteye di un nuovo host o la propagazione di un subset di controlli predefiniti).

La base dati

La prima necessità da soddisfare è stata la creazione di una struttura di tabelle in cui poter memorizzare il risultato della scansione della farm virtuale. Le tabelle necessarie in questa fase sono le seguenti.

Nome tabella	Descrizione
vm_cluster	Contiene le informazioni sui Cluster VMware
vm_hosts	Contiene le informazioni sui server ESX che compongono i vari Cluster VMware
vm_virtual	Contiene le informazioni sui nodi virtuali in carico ad ogni ESX

La tabella “vm_cluster”

Contiene le informazioni sui cluster VMware:

```
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| id               | smallint(6)  | NO   | PRI | NULL    | auto_increment |
| id_infrastructure | smallint(6)  | NO   |     | NULL    |                |
| cl_name          | varchar(150) | NO   | UNI |         |                |
| data_creazione   | datetime     | NO   |     | NULL    |                |
| data_modifica    | datetime     | NO   |     | NULL    |                |
| abilitato        | smallint(6)  | NO   |     | 1       |                |
+------------------+--------------+------+-----+---------+----------------+
```

La struttura della tabella è molto semplice (si compone di appena 6 campi).

Il secondo campo (“`id_infrastructure`”) è l'identificativo univoco del server di infrastruttura. Le informazioni sul server di infrastruttura sono memorizzate in una ulteriore tabella “vm_infrastructure” che contiene tutte le informazioni necessarie per automatizzare il recupero delle credenziali di accesso e dell'URL da utilizzare per le chiamate SOAP:

```
mysql> desc vm_infrastructure ;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | smallint(6)  | NO   | PRI | NULL    | auto_increment |
| url           | varchar(200) | NO   | MUL |         |                |
| entityname    | varchar(200) | NO   |     |         |                |
| username      | varchar(50)  | NO   |     |         |                |
```

```
| password       | varchar(50)  | NO   |     |         |                |
| data_creazione | datetime     | NO   |     | NULL    |                |
| data_modifica  | datetime     | NO   |     | NULL    |                |
| abilitato      | smallint(6)  | NO   |     | 1       |                |
+----------------+--------------+------+-----+---------+----------------+
```

La tabella “vm_hosts”

Questa tabella contiene le informazioni sui server ESX:

```
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| id             | smallint(6)  | NO   | PRI | NULL    | auto_increment |
| id_dc          | smallint(6)  | NO   |     | NULL    |                |
| host_name      | varchar(150) | NO   | UNI |         |                |
| data_creazione | datetime     | NO   |     | NULL    |                |
| data_modifica  | datetime     | NO   |     | NULL    |                |
| abilitato      | smallint(6)  | NO   |     | 1       |                |
+----------------+--------------+------+-----+---------+----------------+
```

La tabella “vm_virtual”

Infine la tabella “`vm_virtual`” contiene l'elenco degli host virtuali:

```
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| id             | smallint(6)  | NO   | PRI | NULL    | auto_increment |
| id_dc          | smallint(6)  | NO   | MUL | NULL    |                |
| vhost_name     | varchar(150) | NO   |     |         |                |
| vhost_os       | varchar(255) | NO   |     |         |                |
| data_creazione | datetime     | NO   |     | NULL    |                |
| data_modifica  | datetime     | NO   |     | NULL    |                |
| abilitato      | smallint(6)  | NO   |     | 1       |                |
+----------------+--------------+------+-----+---------+----------------+
```

Popolamento delle tabelle di discovery

Per il popolamento delle tabelle di discovery con i dati ottenuti dalla scansione della farm virtuale è stato sviluppato uno script PHP, che facilita l'interfacciamento con il backend in quanto offre funzioni native per la manipolazione dei dati su MySQL.

Lo script per prima cosa carica in un array le informazioni contenute nel file prodotto come output della procedura di scansione:

```
$outarray = file("$outfile");
$newarray = array();
foreach ($outarray as $linea) {
        $newarray[] = trim($linea);
}
$outarray = $newarray;
```

Definito l'array (che a questo punto conterrà tutte le informazioni necessarie per l'inserimento dei dati nelle tabelle) viene istanziato un ciclo che scorrerà tutte le voci dell'array per formattare correttamente le stringhe:

```
foreach ($outarray as $key => $linea) {
       […]
}
```

Le prime istruzioni del ciclo identificano la tipologia di oggetto (quindi distinguendo tra Data Center, nodi fisici e nodi virtuali):

```
        $linea_dc   = strpos($linea, "DATACENTER:");
```

```
        $linea_host = strpos($linea, "HOST:");
        $linea_vm   = strpos($linea, "VM:");
```

Il passo successivo è il popolamento di un nuovo array per ogni tipologia di oggetto. Il seguente codice ad esempio costruisce l'array dei Data Center:

```
        if ($linea_dc !== false) {
                $tmp_array = split(":", $linea);
                $my_dc     = trim($tmp_array[1]);
                $array_discover_cluster[] = $my_dc;
                continue;
        }
```

In pratica la "`if`" verifica che la variabile "`$linea_dc`" sia valorizzata, se lo è ne estrae l'informazione relativa al nome del Data Center e la salva nell'array "`$array_discover_cluster`", quindi passa alla successiva iterazione (istruzione "`continue`").

```
        if ($linea_host !== false) {
                $tmp_array = split(":", $linea);
                $my_host   = trim($tmp_array[1]);
                $array_discover_host[] = $my_host;
                continue;
        }

        if ($linea_vm !== false) {
                $tmp_array = split(":", $linea);
                $my_vm     = strtoupper(trim($tmp_array[1]));
                $array_discover_vm[] = $my_vm;
                $array_discover["$my_dc"]["$my_host"][] = $my_vm;
                continue;
        }
```

Seguendo la stessa logica vengono popolati anche gli array "`$array_discover_host`" e "`$array_discover_vm`", rispettivamente con l'elenco degli ESX e delle macchine virtuali.

Completato il ciclo di caricamento degli array lo script procede ad un confronto tra il contenuto degli stessi e le informazioni presenti nelle tabelle MySQL. La logica di funzionamento è la seguente:

Per ogni elemento degli array viene cercata una corrispondenza nel DB MySQL. Se un elemento presente nell'array non viene trovato nella base dati allora si tratta di un oggetto aggiunto che dovrà essere segnalato al personale di competenza.

Ad esempio, per l'array dei cluster VMware, si consideri il seguente codice:

```
foreach ($array_discover_cluster as $cluster_name) {
       if (!in_array("$cluster_name",$array_vmscan_cluster)) {
               ##------------------------------------------------
               ## aggiunge il cluster in vmscan
               ##------------------------------------------------
               batch_insert_vm_cluster("$id_infrastructure","$cluster_name");
               $res_cl = run_query("select id from vm_cluster where upper(cl_name) = upper('$cluster_name')");
               $row_cl = mysql_fetch_array($res_cl,MYSQL_ASSOC);
               $id_cluster = $row_cl['id'];
               ##------------------------------------------------
               ## popola l'array per il report
               ##------------------------------------------------
               $url = "https://10.55.68.4/NIM/index.php?target=show_hosts&id_cluster=${id_cluster}";
               $array_cluster_aggiunti["$cluster_name"] = "${url}";
       }
}
```

Ogni linea dell'array "`$array_discover_cluster`" viene cercata nell'array "`$array_vmscan_cluster`". Questo secondo array contiene l'elenco dei cluster VMware già presenti nel database ed è stato popolato prima del confronto mediante un'estrazione dei dati presenti nella tabella "`vm_cluster`". Ecco il codice di popolamento dell'array:

```
        $array_vmscan_cluster = array();
        $result = run_query("select id,cl_name from vm_cluster");
        while ($row=mysql_fetch_array($result,MYSQL_ASSOC)) {
                $array_vmscan_cluster[$row['id']] = $row['cl_name'];
        }
```

Se l'elemento non viene trovato allora viene inserito nel database dalla funzione di libreria

“batch_insert_vm_cluster” definita in precedenza nello script PHP ed inoltre viene aggiunto ad un array che conterrà l'elenco di tutti i cluster VMware “nuovi”, la cui presenza dovrà poi venire segnalata al personale di competenza:

```
$url = "https://10.55.68.4/NIM/index.php?target=show_hosts&id_cluster=${id_cluster}";
$array_cluster_aggiunti["$cluster_name"] = "${url}";
```

Nell'array contenente i cluster aggiunti viene salvata anche la URL che permetterà agli amministratori di accedere via WEB all'interfaccia di gestione (che vedremo successivamente) dalla quale sarà possibile eseguire diverse azioni di configurazione, compreso l'inserimento in Neteye.

Seguendo lo stesso meccanismo vengono processati anche gli array con l'elenco dei nodi fisici e dei nodi virtuali. Ad esempio:

```
foreach ($array_discover_vm as $vhost_name) {
        if (!in_array("$vhost_name",$array_vmscan_vm)) {
                ##------------------------------------------------
                ## inserisce la VM in vmscan
                ##------------------------------------------------
                $id_cluster = disc_get_cluster_id_by_vm("$vhost_name");
                batch_insert_vm("$id_infrastructure","$id_cluster","$vhost_name");
                $id_vm = disc_get_vm_id("$vhost_name");
                ##------------------------------------------------
                ## popola l'array per il report
                ##------------------------------------------------
                $url = "https://10.55.68.4/NIM/index.php?target=show_vminfo&id_vm=$id_vm";
                $array_vm_aggiunti["$vhost_name"] = "${url}";
        }
}
```

Se il nodo virtuale non viene trovato in base dati viene etichettato come nuovo ed inserito nel database dalla funzione “batch_insert_vm()”. In questo modo la scansione successiva non rileverà più il sistema.

Si tenga presente che l'inserimento nel database in questa fase è funzionale solo alla procedura di scansione: la configurazione di Neteye non viene modificata. Quindi il discovery di un nuovo sistema non si traduce automaticamente nell'attivazione del monitoring. La scelta se iscrivere o meno il nodo in Neteye viene lasciata all'amministratore di sistema, che in questa operazione sarà agevolato dall'interfaccia WEB appositamente sviluppata.

Individuazione degli oggetti rimossi

Gli oggetti rimossi vengono individuati tramite un procedimento analogo ma a logica invertita: in questo caso si confronteranno tutti gli oggetti già presenti in base dati con quelli rilevati dalla scansione:

```
foreach ($array_vmscan_cluster as $cluster_name) {
       if (!in_array("$cluster_name",$array_discover_cluster)) {
               if (check_enabled("cluster","$cluster_name")) {
                       $res_cl = run_query("select id from vm_cluster
                                where upper(cl_name) = upper('$cluster_name')");
                       $row_cl = mysql_fetch_array($res_cl,MYSQL_ASSOC);
                       $id_cluster = $row_cl['id'];
                       $url = "https://10.55.68.4/NIM/index.php?target=show_hosts&id_cluster=${id_cluster}";
                       $array_cluster_rimossi["$cluster_name"] = "${url}";
               }
       }
}
```

Nel codice riportato viene ciclato l'array “$array_vmscan_cluster” e per ogni cluster viene eseguita una ricerca nell'array popolato dalla procedura di discovery. Se la ricerca non restituisce risultati allora il cluster non è più presente nell'infrastruttura VMware ed è necessario inviare una segnalazione al personale di competenza. Il nome del cluster eliminato viene inserito nell'array “$array_cluster_rimossi” che contiene l'elenco dei cluster rimossi assieme ai link che puntano alla pagina di dettaglio dell'oggetto.

La stessa routine viene eseguita anche per i nodi fisici ESX e per i nodi virtuali:

```
foreach ($array_vmscan_host as $host_name) {
        if (!in_array("$host_name",$array_discover_host)) {
                if (check_enabled("host","$host_name")) {
                        $id_cluster = disc_get_cluster_id("$host_name");
                        $res_host = run_query("select id from vm_hosts where upper(host_name) = upper('$host_name')");
                        $row_host = mysql_fetch_array($res_host,MYSQL_ASSOC);
                        $host_id  = $row_host['id'];
                        $url = "https://10.55.68.4/NIM/index.php?target=show_hostinfo&host_id=$host_id&id_cluster=$id_cluster";
                        $tmp_cl_name = split("[.]","$host_name");
                        $array_host_rimossi[$tmp_cl_name['0']] = "${url}";
                }
        }
}

foreach ($array_vmscan_vm as $vhost_name) {
        if (!in_array("$vhost_name",$array_discover_vm)) {
                if (check_enabled("vm","$vhost_name")) {
                        $id_vm = disc_get_vm_id("$vhost_name");
                        $url = "https://10.55.68.4/NIM/index.php?target=show_vminfo&id_vm=$id_vm";
                        $array_vm_rimossi["$vhost_name"] = "${url}";
                }
        } else {
                if (check_enabled("vm","$vhost_name")) {
                        if (!host_is_in_nagios("$vhost_name")) {
                                $id_vm = disc_get_vm_id("$vhost_name");
                                $url = "https://10.55.68.4/NIM/index.php?target=show_vminfo&id_vm=$id_vm";
                                $array_vm_not_in_nagios["$vhost_name"] = "${url}";
                        }
                }
        }
}
```

Nel ciclo relativo ai sistemi virtuali vengono eseguite delle istruzioni aggiuntive. Se la VM risulta presente in base dati (e quindi non è stata aggiunta di recente) viene operato un secondo controllo con lo scopo di verificare se il sistema è presente nella configurazione di Neteye.

Questo compito è assolto dalla funzione “`host_is_in_nagios()`” che consulta il DB con il repository della configurazione di Neteye. Se il sistema non risulta iscritto al monitoring viene inserito in un array dedicato (`$array_vm_not_in_nagios`) che verrà utilizzato per l'invio delle opportune segnalazioni.

Il punto debole di questo approccio è che i sistemi non presenti in Neteye verranno segnalati ad ogni iterazione della procedura di scansione e verifica anche se è lecito che non siano iscritti a monitoring (si pensi ad esempio ai template delle VM). In questo caso l'amministratore è in grado di “disabilitare” l'host utilizzando l'interfaccia WEB della procedura di discovery. Per le VM disabilitate non vengono inviate notifiche: questo è il motivo per cui prima di popolare l'array “`$array_vm_not_in_nagios`” viene eseguita la funzione “`check_enabled()`” che restituisce `FALSE` se il sistema è stato disabilitato da interfaccia.

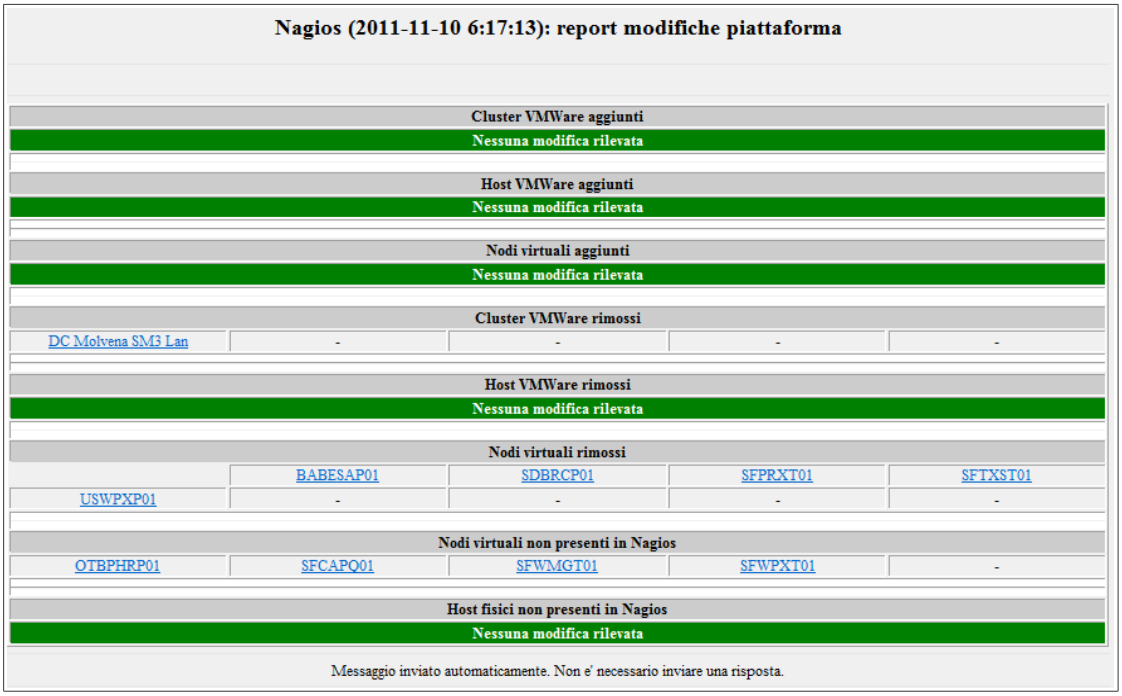
Invio delle notifiche sui cambiamenti

Completata la routine di confronto avremo quindi una serie di array contenenti tutte le informazioni di cui abbiamo bisogno per l'invio delle notifiche. La seguente tabella offre una visione riassuntiva degli array di cui disponiamo:

Array	Descrizione
$array_cluster_aggiunti	Contiene l'elenco dei cluster aggiunti rispetto all'ultima esecuzione della procedura di scansione.
$array_host_aggiunti	Contiene l'elenco degli host fisici (ESX Server) aggiunti dall'ultima esecuzione della scansione.
$array_vm_aggiunti	Contiene l'elenco dei nodi virtuali (VM) aggiunti dall'ultima esecuzione della scansione.
$array_cluster_rimossi	Contiene l'elenco dei cluster rimossi.
$array_host_rimossi	Contiene l'elenco dei nodi fisici (ESX Server) rimossi.
$array_vm_rimossi	Contiene l'elenco dei nodi virtuali (VM) rimossi.
$array_vm_not_in_nagios	Contiene l'elenco dei nodi virtuali trovati nella piattaforma VMware ma non presenti in Neteye. Eventuali nodi non presenti in Nagios ma disabilitati dall'amministratore non saranno presenti in questo array.

Il report che vogliamo ottenere è in pratica una pagina HTML con una struttura tabellare che riporti l'elenco dei cambiamenti avvenuti sulla piattaforma con l'evidenza degli oggetti aggiunti e rimossi, per ognuno dei quali viene fornito un link diretto all'interfaccia WEB di gestione.

Ecco un esempio di email generata automaticamente:



Nagios (2011-11-10 6:17:13): report modifiche piattaforma

Cluster VMWare aggiunti
Nessuna modifica rilevata
Host VMWare aggiunti
Nessuna modifica rilevata
Nodi virtuali aggiunti
Nessuna modifica rilevata
Cluster VMWare rimossi

DC Molvena SM3 Lan	-	-	-	-

Host VMWare rimossi
Nessuna modifica rilevata
Nodi virtuali rimossi

	BABESAP01	SDBRCP01	SFPRXT01	SFTXST01
USWPXP01	-	-	-	-

Nodi virtuali non presenti in Nagios

OTBPHRP01	SFCAPQ01	SFWMGT01	SFWPXT01	-

Host fisici non presenti in Nagios
Nessuna modifica rilevata

Messaggio inviato automaticamente. Non e' necessario inviare una risposta.

Per ottenere questo scopo è stata sviluppata una piccola libreria di funzioni di formattazione che semplifica l'assemblaggio del codice HTML necessario a generare l'email di cui sopra.

Ecco le istruzioni necessarie per comporre il messaggio:

```
$my_msg = "";
$my_msg.= html_printarray("Cluster VMWare aggiunti", $array_cluster_aggiunti);
$my_msg.= html_printarray("Host VMWare aggiunti",    $array_host_aggiunti);
$my_msg.= html_printarray("Nodi virtuali aggiunti",  $array_vm_aggiunti);
$my_msg.= html_printarray("Cluster VMWare rimossi",  $array_cluster_rimossi);
$my_msg.= html_printarray("Host VMWare rimossi",     $array_host_rimossi);
$my_msg.= html_printarray("Nodi virtuali rimossi",   $array_vm_rimossi);
$my_msg.= html_printarray("Nodi virtuali non presenti in Nagios",      $array_vm_not_in_nagios);
```

Le funzione invocata (html_printarray) produce una tabella HTML a partire da un array. Il primo argomento passato alla funzione sarà il titolo della tabella, il secondo l'array contenente i dati da mostrare. L'array contiene, come abbiamo visto in precedenza, il nome dell'oggetto ed il relativo link alla pagina di amministrazione WEB.

Per completezza riportiamo di seguito la definizione della funzione “html_printarray”:

```
function html_printarray($title,$array) {
	$msg = "<TR BGCOLOR=\"#CCCCCC\">\n";
	$msg.= "<TD COLSPAN=\"5\" style=\"text-align: center\"><div style=\"font-weight: bold\">\n";
	$msg.= "${title}</div></TD></TR>\n";
	if (count($array) == 0) {
		$msg.= "<TR BGCOLOR=\"green\">\n";
		$msg.= "<TD COLSPAN=\"5\" style=\"text-align: center\"><div style=\"font-weight: bold; color: white;\">\n";
		$msg.= "Nessuna modifica rilevata</TD></TR>";
		$msg.= "<TR BGCOLOR=\"white\">\n";
		$msg.= "<TD COLSPAN=\"5\" style=\"text-align: center\"><div style=\"font-weight: bold\">\n";
		$msg.= "<HR /></TR>";
		return $msg;
	}
	$idx = 1;
	$msg .= "<TR>\n";
	foreach ($array as $label => $url) {
```

```
        $link = "<A HREF=\"${url}\">${label}</A>";
        ## $link = "${label}";
        if ($idx == 6) {
            $msg.= "</TR>\n";
            $idx = 1;
            $msg.= "<TR>\n";
        }
        $msg.= "<TD WIDTH=\"20%\" style=\"text-align: CENTER\">${link}</TD>\n";
        $idx++;
    }
    while ($idx != 6) {
        $msg.= "<TD WIDTH=\"20%\" ALIGN=\"CENTER\"> - </TD>\n";
        $idx++;
    }
    $msg.= "</TR>";
    $msg.= "<TR BGCOLOR=\"white\">\n";
    $msg.= "<TD COLSPAN=\"5\" style=\"text-align: center\"><div style=\"font-weight: bold\">\n";
    $msg.= "<HR /></TR>";
    return $msg;
}
```

L'elenco dei destinatari del report è (a sua volta) memorizzato in una apposita tabella MySQL:

```
mysql> desc vm_email ;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| id             | smallint(6)  | NO   | PRI | NULL    | auto_increment |
| contatto       | varchar(250) | NO   |     |         |                |
| address        | varchar(250) | NO   | UNI |         |                |
| data_creazione | datetime     | NO   |     | NULL    |                |
| data_modifica  | datetime     | NO   |     | NULL    |                |
| abilitato      | smallint(6)  | NO   |     | 1       |                |
+----------------+--------------+------+-----+---------+----------------+
```

Utilizzando un semplice ciclo “`while`” l'email viene inviata ad ogni destinatario che risulta abilitato nella tabella:

```
        $res_email = run_query("select address from vm_email where abilitato='1'");
        while ($row_email=mysql_fetch_array($res_email,MYSQL_ASSOC)) {
                $my_dest = $row_email['address'];
                send_mail("$my_dest","$my_msg");
                batch_log_it("report inviato a $my_dest");
        }
```

Ecco il codice della funzione “`send_mail`”, che materialmente effettua l'invio utilizzando l'istruzione “`mail`” del PHP:

```
function send_mail($dest,$message) {
        $now     = get_now();
        $sbj     = "Nagios ($now): report modifiche piattaforma";
        $headers = 'MIME-Version: 1.0' . "\r\n";
        $headers.= 'Content-type: text/html; charset=iso-8859-15' . "\r\n";
        $mail    = htmlmail_open("${sbj}");
        $mail   .= "${message}";
        $mail   .= htmlmail_close();
        mail ($dest, $sbj, $mail, $headers);
}
```

La linea “`batch_log_it("report inviato a $my_dest");`” è il richiamo ad una funzione di logging (che scrive a sua volta in una tabella MySQL dedicata) in cui viene tracciato l'invio del messaggio.

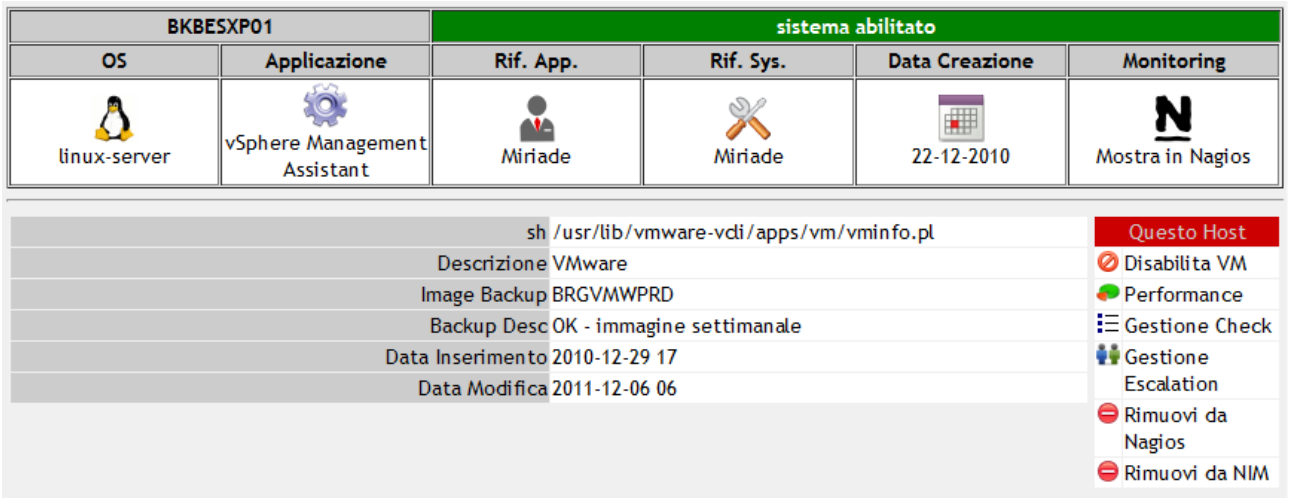
Ottenere maggiori informazioni sui nodi virtuali

Interrogando in modo opportuno il server di infrastruttura è possibile ottenere numerose informazioni aggiuntive sulle macchine virtuali.

Abbiamo ritenuto utile il collezionamento di queste informazioni e di conseguenza abbiamo deciso di estendere la procedura in modo che potesse sia raccoglierle e storicizzarle in

modalità automatica, sia presentarle in un formato fruibile per mezzo di un'interfaccia WEB.

La seguente immagine mostra il dettaglio di una ipotetica macchina virtuale visualizzata attraverso l'interfaccia WEB personalizzata (denominata “NIM”):



Il funzionamento della procedura di acquisizione delle informazioni è molto semplice: si articola infatti un un solo ciclo `while`:

```
##============================================================
## estrae le informazioni su tutte le vm inserite (vm_virtual)
##============================================================
$result = run_query("select * from vm_virtual where abilitato=1");
while ($row=mysql_fetch_array($result,MYSQL_ASSOC)) {
        $id_vm    = $row['id'];
        $id_dc    = $row['id_dc'];
        $id_infra = get_id_infra("$id_dc");
        ##================================================================
        ## aggiorna le informazioni sulla VM
        ##================================================================
        batch_update_vm_info("$id_infra","$id_vm");
}
```

Il ciclo inizia con una query che estrae dal DB tutto il contenuto della tabella vm_virtual, che contiene le informazioni sui sistemi virtuali identificati dalla procedura di discovery.

Per ogni macchina virtuale viene invocata la funzione “`batch_update_vm_info`”; vediamola in dettaglio:

```
function batch_update_vm_info($id_infrastructure,$id_vm) {
        global $vmcustomfield;
        global $vminfo;
        $array_info                          = array();
        $array_info['Funzione']              = "non disponibile";
        $array_info['Applicazione']          = "non disponibile";
        $array_info['Data di Creazione']     = "non disponibile";
        $array_info['Image Backup']          = "non disponibile";
        $array_info['Descrizione Backup']    = "non disponibile";
        $array_info['Referente Applicativo'] = "non disponibile";
        $array_info['Referente Sistemistico'] = "non disponibile";
        $infra_us   = get_infra_username("$id_infrastructure");
        $infra_pw   = get_infra_password("$id_infrastructure");
        $infra_url  = get_infra_url("$id_infrastructure");
        $vhost_name = get_vm_name("$id_vm");
        foreach ($array_info as $campo => $valore) {
                $new_valore = "";
                $my_tmp_array = array();
                $comando = "$vmcustomfield --url ${infra_url} \
                            --username ${infra_us} --password ${infra_pw} \
                            --vmname ${vhost_name} --customfield \"$campo\"";
                exec ("$comando 2>&1", $my_tmp_array, $retval);
                if ($retval == 0) {
                        if ( !empty($my_tmp_array['0']) ) {
                                $new_valore = $my_tmp_array['0'];
```

```
                } else {
                        $new_valore = "$valore";
                }
        }
        if ($new_valore == "") { $new_valore = "Not Known"; }
        $array_info["$campo"] = "$new_valore";
}
$res_os = run_query("select vhost_os from vm_virtual where id=$id_vm");
$row_os = mysql_fetch_array($res_os,MYSQL_ASSOC);
$tmp_os = $row_os['vhost_os'];
$my_info_tpl = "unknown";
if ($tmp_os == "") { $tmp_os = "unknown"; }
if (eregi("(windows)", "$tmp_os", $regs)) { $my_info_tpl = "windows-server"; }
if (eregi("(linux)",   "$tmp_os", $regs)) { $my_info_tpl = "linux-server";   }
if (eregi("(centos)",  "$tmp_os", $regs)) { $my_info_tpl = "linux-server";   }
if (eregi("(solaris)", "$tmp_os", $regs)) { $my_info_tpl = "solaris-server"; }
##=========================================================
## recupera le informazioni per popolare il campo
## vm_info.info
##=========================================================
$tmp_vm_info = array();
$comando_info = "$vminfo --url ${infra_url} --username ${infra_us} \
                         --password ${infra_pw} --vmname \"${vhost_name}\"";
exec ("$comando_info 2>&1", $tmp_vm_info, $retval);
$my_vm_info = serialize($tmp_vm_info);
##=========================================================
## inserisce / aggiorna il DB
##=========================================================
$vm_desc           = $array_info['Funzione'];
$vm_app            = $array_info['Applicazione'];
$vm_data_creazione = $array_info['Data di Creazione'];
$vm_image_bck      = $array_info['Image Backup'];
$vm_desc_bck       = $array_info['Descrizione Backup'];
$vm_refer_app      = $array_info['Referente Applicativo'];
$vm_refer_sys      = $array_info['Referente Sistemistico'];
$now               = get_now();
if (num_rows("select id from vm_info where id_virtual=$id_vm") == 0) {
        ##=========================================================
        ## insert
        ##=========================================================
        $query = "insert into vm_info(id_virtual,info,vm_desc,vm_hostinfo_tpl,";
        $query.= "vm_app,vm_data_creazione,vm_image_bck,";
        $query.= "vm_desc_bck,vm_refer_app,vm_refer_sys,data_modifica,";
        $query.= "data_creazione,abilitato)";
        $query.= "values($id_vm,'$my_vm_info','$vm_desc','$my_info_tpl',";
        $query.= "'$vm_app','$vm_data_creazione','$vm_image_bck',";
        $query.= "'$vm_desc_bck','$vm_refer_app','$vm_refer_sys','$now',";
        $query.= "'$now','1')";
} else {
        ##=========================================================
        ## update
        ##=========================================================
        $query = "update vm_info set ";
        $query.= "             info='${my_vm_info}',";
        $query.= "          vm_desc='${vm_desc}',";
        $query.= "  vm_hostinfo_tpl='${my_info_tpl}',";
        $query.= "           vm_app='$vm_app',";
        $query.= "vm_data_creazione='$vm_data_creazione',";
        $query.= "     vm_image_bck='$vm_image_bck',";
        $query.= "      vm_desc_bck='$vm_desc_bck',";
        $query.= "     vm_refer_app='$vm_refer_app',";
        $query.= "     vm_refer_sys='$vm_refer_sys',";
        $query.= "    data_modifica='${now}'";
        $query.= " where id_virtual='${id_vm}'";
}
run_query("$query");
}
```

Come prima azione la funzione imposta dei valori di default che verranno utilizzati nel caso la query sul server di infrastruttura non restituisse nulla:

```
        $array_info                         = array();
        $array_info['Funzione']             = "non disponibile";
        $array_info['Applicazione']         = "non disponibile";
        $array_info['Data di Creazione']    = "non disponibile";
        $array_info['Image Backup']         = "non disponibile";
        $array_info['Descrizione Backup']   = "non disponibile";
```

```
$array_info['Referente Applicativo']  = "non disponibile";
$array_info['Referente Sistemistico'] = "non disponibile";
```

Successivamente l'array con i campi da valorizzare viene utilizzato all'interno di un ciclo “foreach” annidato che cercherà di recuperare i record necessari:

```
foreach ($array_info as $campo => $valore) {
        $new_valore = "";
        $my_tmp_array = array();
        $comando = "$vmcustomfield --url ${infra_url} \
                    --username ${infra_us} --password ${infra_pw} \
                    --vmname ${vhost_name} --customfield \"$campo\"";
        exec ("$comando 2>&1", $my_tmp_array, $retval);
[…]
}
```

L'istruzione che materialmente effettua il recupero delle informazioni è la chiamata “`exec()`” che invia un comando all'infrastruttura VMware utilizzando l'utility “`vmware_custom_field.pl`”, compresa nelle VMware CLI.

Il codice successivo ha lo scopo di recuperare i valori restituiti dal server di infrastruttura e memorizzarli in una serie di variabili che verranno utilizzate per formattare la query SQL per il popolamento della base dati. La funzione verifica se per la macchina virtuale in oggetto è già presente un record con le informazioni aggiuntive: se il record è presente verrà impostata una query di “`update`”, altrimenti verrà eseguita una “`insert`”.

Conclusioni

L'introduzione della procedura descritta ha permesso di ridurre considerevolmente l'effort necessario per la gestione ordinaria del monitoring dei nodi virtuali VMware. Gli standard aperti implementati da Neteye hanno consentito un ottimo livello di integrazione, confermando la validità del prodotto ed evidenziando caratteristiche come la scalabilità e l'ampio livello di personalizzazione.

Nel prossimo articolo prenderemo in considerazione il fontend realizzato in PHP per facilitare l'esecuzione delle operazioni più comuni, come l'inserimento dei nodi virtuali in Neteye e la propagazione dei controlli su gruppi o famiglie di sistemi.