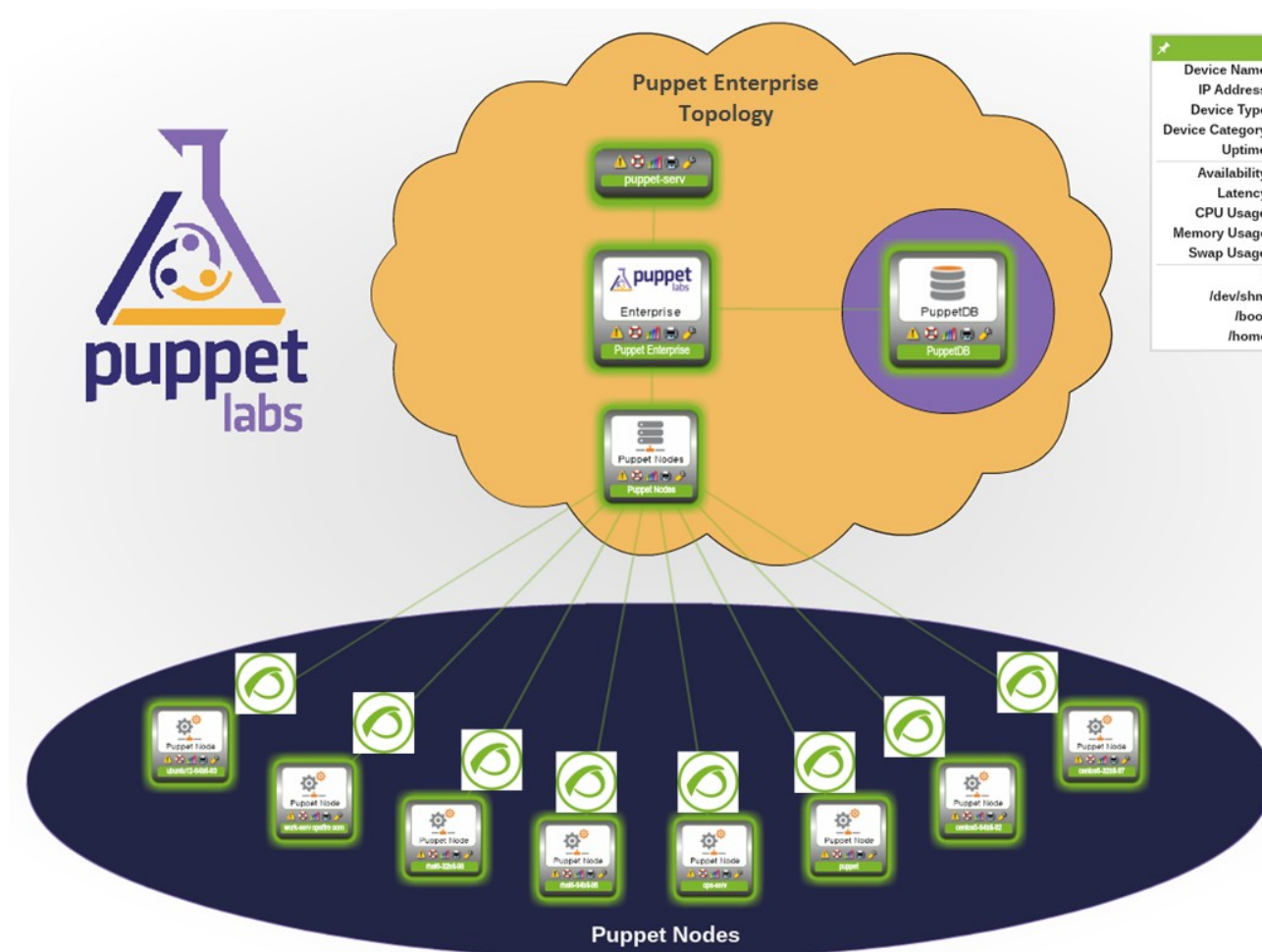


Automatización de la Instalación de Agentes PandoraFMS con Puppet.

Este artículo va a servir como base para la automatización del proceso de instalación de los agentes software de Pandora FMS en varios sistemas de forma simultánea: ya sean Windows, Linux o Unix. La herramienta que se ha utilizado para realizar este proceso es Puppet.

Puppet es una herramienta OpenSource de nueva generación para la automatización de servidores. Cuenta con un lenguaje declarativo que sirve para expresar la configuración del sistema, un esquema cliente-servidor para la distribución, y una librería para ejecutar dicha configuración. Entre sus diferentes acciones nos permite controlar la ejecución de diferentes servicios en una máquina, control centralizado de archivos de configuración, gestión e instalación de paquetes, administración de la presencia y disponibilidad de puntos de montaje, gestión de usuarios y grupos y despliegue de maquinas.



En este artículo se indicará el lenguaje declarativo que habría que configurar en Puppet (módulos de Puppet) partiendo de la base que la red donde se va a realizar el despliegue de agentes, tiene ya instalado el servidor Puppet y los diferentes clientes Puppet en los diferentes equipos.

Como indicamos Puppet realiza el despliegue a través de “manifiestos”. Estos manifiestos están escritos en un código propio de Puppet e indican a los diferentes clientes de Puppet los procesos que

tienen que realizar para llevar a cabo el propósito final, que en nuestro caso va a ser el de instalar, configurar y arrancar los agentes.

La carpeta que se toma como principal en toda la configuración de Puppet es */etc/puppet*. Este directorio podría variar dependiendo de la ruta de la instalación de Puppet.

Dentro de la carpeta */etc/puppet/files*, donde se introducirán los ficheros estáticos que se enviarán a los diferentes clientes vamos a introducir los archivos necesarios para el despliegado del agente:

- **Agente software** linux y/o windows (agente en formato tar.gz)
- **Archivo de configuración del agente** que tendrá la configuración que deseemos que tenga, lo más importante es que la IP del servidor esté configurada correctamente, y si queremos introducir alguna configuración específica de módulos ahora es el momento. Hay que recordar que en el caso que vayamos a realizar el despliegado en agentes Windows y Linux hay que tener un archivo de configuración para cada uno de los SO, ya que la configuración de los diferentes módulos en ambos SO no suele ser la misma.

Dentro del directorio */etc/puppet/manifests* crearemos los recursos, que en resumen, serán los pasos que Puppet realizará en el despliegado de agentes.

Lo primero ante todo es crear el fichero *init.pp*, que es donde Puppet mira los recursos del módulo. Irá dentro del directorio */etc/puppet/manifests*.

Para un despliegue de agentes Unix este sería el contenido del fichero *init.pp*

```
class install_agent_unix {

  File ['pandorafms_agent_unix-5.1.tar.gz file'] → Exec ['Extract tar'] → Exec [ 'Instalar Pandora Agent'] → File ['pandora_agent.conf file'] → Service ['pandora_agent_daemon service']

  file { 'pandorafms_agent_unix-5.1.tar.gz file':
    owner  => root,
    group  => root,
    mode   => 777,
    source => "puppet:///files/pandorafms_agent_unix-5.1.tar.gz",
    path   => "/tmp/pandorafms_agent_unix-5.1.tar.gz"
  }

  exec { 'Extract tar':
    path => "/bin:/usr/bin",
    Unless => "find /tmp/pandorafms_agent_unix-5.1",
    Command => "tar -xzf /tmp/pandorafms_agent_unix-5.1.tar.gz",
  }

  exec { 'Instalar Pandora Agent':
    Command => "/tmp/pandorafms_agent_unix-5.1/pandora_agent_installer -install",
  }

  file { 'pandora_agent.conf file':
    owner  => root,
    group  => root,
    mode   => 644,
    source => "puppet:///files/pandora_agent.conf",
    path   => "/etc/pandora/pandora_agent.conf",
  }

  service { 'pandora_agent_daemon service':
    ensure => running,
    name   => "pandora_agent_daemon",
    enable => true,
  }
}
```

Como resumen a este manifiesto, lo que vamos a hacer es enviar el archivo, descomprimirlo, ejecutar la instalación, enviar el archivo de configuración y lanzar el agente pandora.

En el caso de que se trate del despliegue para agentes Windows quedaría algo así:

```
class install_agent_wdos {  
  
  File ['PandoraFMS_windows_agent_v5.1.setup.exe file'] → Exec [ 'Instalar Pandora Agent'] → File  
  ['pandora_agent.conf file'] → Service ['pandora_agent service']  
  
  file { 'pandorafms_agent_unix-5.1.setup.exe file':  
    owner   => root,  
    group   => root,  
    mode     => 777,  
    source   => "puppet:///modules/pandorafms/pandorafms\_agent\_unix-5.1.setup.exe",  
    path     => "C:\PandoraFMS_windows_agent_v5.1.setup.exe"  
  }  
  
  exec { 'Instalar Pandora Agent':  
    Command => "C:\PandoraFMS_windows_agent_v5.1.setup.exe /mode Silent /prefix  
c:\agente_pandora,  
  }  
  
  file { 'pandora_agent.conf file':  
    owner   => root,  
    group   => root,  
    mode     => 644,  
    source   => "puppet:///modules/pandorafms/pandora\_agent.conf",  
    path     => "c:\agente_pandora\pandora_agent.conf",  
  }  
  
  service { 'pandora_agent_daemon service':  
    ensure => running,  
    name   => "pandora_agent_daemon",  
    enable => true,  
  }  
}
```

Cabe recordar que los agentes Windows de Puppet están soportados oficialmente sobre versiones de Windows Server 2003, 2008. No están oficialmente soportadas otras versiones pero deberían funcionar.

Una vez tenemos el manifiesto configurado lo único que nos queda es asociar este manifiesto a cada uno de los nodos o máquinas que queramos que se instale el agente.

/etc/puppet/manifests/site.pp

Un ejemplo de esta configuración para aplicarlo a cada nodo podría ser esta:

```
node unix_agent {  
  
    include install_agent_unix  
  
}  
  
node windows_agent {  
  
    include install_agent_wdos  
  
}
```

Hay diversas formas de asociar las clases a los nodos en Puppet, en este ejemplo se ha optado por ésta, pero en el caso de que el administrador de Puppet decida aplicarlo a través de módulos o de algún otro modo, para compatibilizarlo con la instalación actual que tenga de Puppet en su entorno, lo podrá hacer sin problemas, puede apoyarse en estos manifiestos indicados para hacer su configuración.

Monitorización de Agentes Puppet con Pandora FMS

Para monitorizar los agentes Puppet tenemos un plugin en nuestra librería de módulos de pandorafms.com http://pandorafms.com/index.php?sec=Library&sec2=repository&lng=es&action=view_PUI&id_PUI=600 que se encarga de sacar la siguiente información de un agente Puppet

- Last run → Última ejecución.
- Time since last run -> Tiempo desde la última ejecución.
- Is out of sync? → si está fuera de sincronismo o no.
- Resources changed/failed/failed to restart/restarted/scheduled/skipped → Estado de los recursos.
- Events failed/succeeded → Eventos fallidos con éxito.
- Total changes → Cambios totales

El plugin esta escrito en Ruby, por lo que hay que tener Ruby instalado en el equipo, aunque al tratarse de un equipo con agente puppet y agente pandora es un requisito imprescindible también para el agente Puppet.

















































Para ejecutar el plugin únicamente debemos encontrar el fichero *last_run_summary.yaml* que es donde Puppet guarda el estado de la última vez que se lanza. El fichero tiene una composición parecida a esta:

```
time:
  group: 0.001692
  last_run: 1317804488
  class: 0.003929
  yumrepo: 0.020103
  service: 9.017434
  schedule: 0.004151
  cron: 0.010546
  config_retrieval: 15.0572321414948
  total: 34.9742621414947
  package: 0.588751
  filebucket: 0.000687
  file: 8.895422
  exec: 1.361625
  user: 0.01269
resources:
  total: 194
  changed: 0
  failed: 0
  failed_to_restart: 0
  out_of_sync: 0
  scheduled: 0
  skipped: 0
events:
  total: 0
changes:
  total: 0
```

Este fichero por defecto se guarda en */var/lib/puppet/state/last_run_summary.yaml* por lo que la ejecución del plugin sería algo así:

```
module_plugin <ruby_path> /etc/pandora/plugins/pandora_puppet.rb --summary-file
/var/lib/puppet/state/last_run_summary.yaml
```

En Windows, el archivo *last_run_summary.yaml* se encuentra en el directorio *C:\ProgramData\PuppetLabs\puppet\var\state*

	Events failed	How many events failed		N/A - 0/1	0			2 minutes 10 seconds
	Events succeeded	How many events succeeded		N/A - N/A	0			2 minutes 10 seconds
	Last run	Last run timestamp		N/A - N/A	1395079338			2 minutes 10 seconds
	Resources changed	How many resources were changed		N/A - N/A	0			2 minutes 10 seconds
	Resources failed	How many resources were not successfully fixed		N/A - 0/1	0			2 minutes 10 seconds
	Resources failed to restart	How many resources could not be restarted		N/A - 0/1	0			2 minutes 10 seconds
	Resources out of sync	How many resources were out of sync		N/A - 0/1	0			2 minutes 10 seconds
	Resources restarted	How many resources were restarted because their dependencies...		N/A - N/A	0			2 minutes 10 seconds
	Resources scheduled	How many resources met any scheduling restrictions		N/A - N/A	0			2 minutes 10 seconds
	Resources skipped	How many resources were skipped, because of either tagging o...		N/A - 0/5	0			2 minutes 10 seconds
	Time since last run	Time since last run in seconds		N/A - N/A	553			2 minutes 10 seconds
	Total changes	The total number of changes in the transaction		N/A - N/A	0			2 minutes 10 seconds

Monitorización Servidor Puppet con Pandora FMS.

Para monitorizar el servidor Puppet el requisito principal es tener instalado un agente Pandora en el mismo servidor que se encargue de realizar todos los chequeos oportunos dentro del propio servidor y mande los resultados a el servidor de Pandora FMS que tengamos instalado en nuestro entorno.

A continuación se van a indicar la configuración de los diferentes módulos que habria que configurar en el fichero de configuración del agente de Pandora.

1.- Estado *puppetmaster* y *puppetdb*

Módulo de tipo booleano (servicio) que nos indica si el servidor puppet se encuentra corriendo o no. Nos devolverá un 1 si esta corriendo y un 0 cuando no.

```
module_begin
module_name Status puppetmaster
module_type generic_proc
module_exec /etc/init.d/puppetmaster status | grep running | wc -l
module_description Status puppermaster
module_end

module_begin
module_name Status puppetdb
module_type generic_proc
module_exec /etc/init.d/puppetdb status | grep running | wc -l
module_description Status puppetdb
module_end
```

2.- Estado configuración de los manifiestos de Puppet.

El propio servidor puppet tiene una herramienta que nos permite chequear el estado de los manifiestos puppet. Si seleccionamos en su ejecución el directorio donde se guardan estos manifiestos, nos realiza el chequeo de todos ellos. Tenemos dos posibles configuraciones para esta monitorización. Una de ellas es que nos muestre si hay alguna anomalía o no, y otra es que nos saque el error cuando haya algún error.

```
module_begin
module_name Status manifests
module_type generic_proc
module_exec puppet parser validate /etc/puppet/manifests/ | wc -l
module_critical_inverse 1
module_description Status manifest
module_end

module_begin
module_name Manifests errors
module_type async_string
module_exec puppet parser validate /etc/puppet/manifests/
module_description Status manifest
module_end
```

3.- Chequear si todos los agentes Puppet están firmados.

Con el siguiente módulo podemos chequear si todos los agentes están firmados correctamente en el servidor de Puppet y por lo tanto funcionando sin problemas

```
module_begin
module_name Puppet agents certs status
module_type generic_proc
module_exec puppet cert -list | wc -l
module_critical_inverse 1
module_description Status Puppet agents certs
module_end
```

4.- Uso de CPU y RAM del servicio Puppetmaster y puppetdb

Chequeo del % Uso de CPU y RAM que estan usando los servicios del servidor de Puppet

```
module_begin
module_name Cpu Use % Puppetmaster
module_type generic_data
module_exec ps aux | grep puppetmaster | grep -v grep | gawk '{print $3}'
module_description Cpu Use % Puppetmaster
module_end

module_begin
module_name RAM Use % Puppetmaster
module_type generic_data
module_exec ps aux | grep puppetmaster | grep -v grep | gawk '{print $4}'
module_description RAM Use % Puppetmaster
module_end





































module_begin
module_name Cpu Use Tot % puppetdb
module_type generic_data
module_exec ps aux | grep puppetdb | grep -v grep | awk '{sum += $3} END {print sum}'
module_description Cpu Use % Puppetdb
module_end

module_begin
```

```

module_name RAM Use Total % Puppetdb
module_type generic_data
module_exec ps aux | grep puppetdb | grep -v grep | awk '{sum += $4} END {print sum}'
module_description RAM Use Total % Puppetdb
module_end

```

Puppet								
		Cpu Use % Puppetmaster	Cpu Use % Puppetmaster		N/A - N/A	0.3	 	2 minutes 06 seconds
		CPU Use Total % Puppetdb	Cpu Use % Puppetdb		N/A - N/A	0.6	 	2 minutes 06 seconds
		Manifests errors	Status manifest		N/A - N/A	See 'puppet' 🌸	 	14 minutes 38 seconds
		Puppet agents certs status	Status Puppet agents certs		N/A - N/A	0	 	2 minutes 07 seconds
		RAM Use % Puppetmaster	RAM Use % Puppetmaster		N/A - N/A	4	 	2 minutes 07 seconds
		RAM Use Total % Puppetdb	RAM Use Total % Puppetdb		N/A - N/A	23.8	 	2 minutes 07 seconds
		Status manifests	Status manifest		N/A - N/A	0	 	2 minutes 07 seconds
		Status puppetdb	Status puppetdb		N/A - N/A	1	 	2 minutes 07 seconds
		Status puppetmaster	Status puppetmaster		N/A - N/A	1	 	2 minutes 07 seconds