

Análisis y monitorización de aplicaciones a través de plugins con Naemon

Sofía Fernández Moreno

Universidad de Granada

Septiembre de 2019



ugr

Universidad
de Granada



Estado del arte

Realización del despliegue de Naemon

Pruebas de carga

Pruebas de carga en un sistema

Modelado de las pruebas

Conclusiones y futuros trabajos

¿Qué es la monitorización?

Comparativa herramientas de monitorización

Naemon

Estado del arte

¿Qué es la monitorización?

La **monitorización** es aquella en la que se toman las medidas preventivas y consecuentes con la información que se obtienen de todos los dispositivos que se encuentran conectados a una red, para evitar posibles eventos que hacen que se interrumpan el correcto funcionamiento de alguno de ellos

Dentro de este concepto es importante aplicar el uso del protocolo **SNMP** (Simple Network Management Protocol).

***SNMP** permite el intercambio de información amplia entre los diferentes dispositivos de red mediante consultas de forma remota (polling) y mediante mensajes basándose en eventos (traps).*

Comparativa herramientas de monitorización

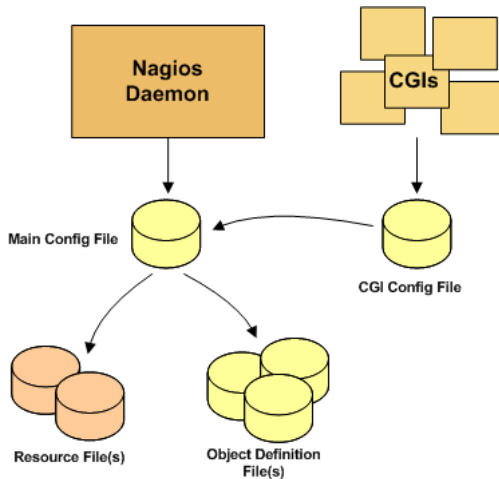


Funcionamiento

- Basado en Nagios 4.0.2, funcionando de la misma forma mediante representación de la monitorización a través de host y servicios concretos.
- Monitorización de servicios de red
- Monitorización de recursos de un host
- Diseño de plugins o complementos

Podemos encontrar cuatro estados en los chequeos: **OK**, **WARNING**, **CRITICAL** y **UNKNOWN**

Archivo de configuración principal



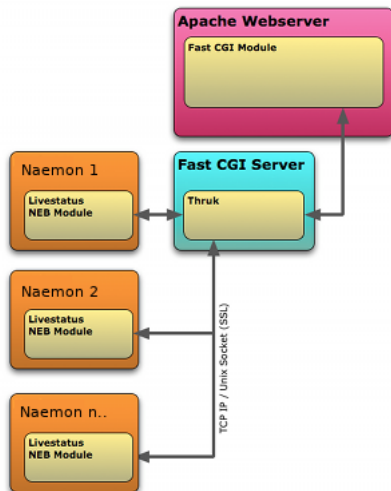
Tipos de objetos

*La **configuración** de los distintos equipos y servicios a monitorizar se puede realizar a través de la carpeta **/etc/naemon/conf.d***

Podemos encontrar como objetos de definición:

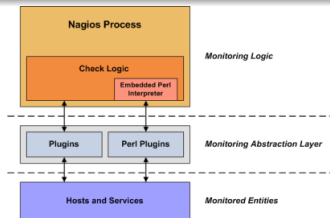
- Hosts
- Servicios
- Comandos
- Contactos
- Time Periods

Interfaz GUI: Thruk



Uso de plugins

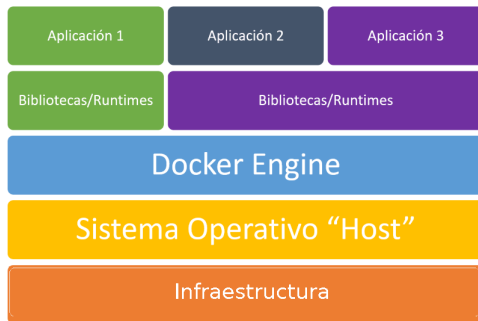
Los **plugins** actúan como una capa de abstracción entre la lógica de supervisión presente en el demonio de Naemon y los servicios y hosts reales que se están supervisando.



El inconveniente de usar plugins en Naemon es que éste no tiene reflejo de qué es lo que está monitorizando, simplemente rastrea los cambios en el estado de esos recursos.

Realización del despliegue de Naemon

Uso de contenedores



Estado del arte

Realización del despliegue de Naemon

Pruebas de carga

Pruebas de carga en un sistema

Modelado de las pruebas

Conclusiones y futuros trabajos

Entorno de desarrollo

Desarrollo de despliegue

```
FROM phusion/baseimage:0.11
MAINTAINER Sofia <chui274@gmail.com>

RUN apt-get update && \
    DEBIAN_FRONTEND=noninteractive \
    apt-get install -y \
        apache2 \
        apache2-utils \
        libapache2-mod-fcgid \
        libfontconfig \
        libjpeg2 \
        libmagick \
        libmem \
        xvfb \
        socat \
        ruby \
        python2.7 \
        python-boto \
        perl \
        libwww-perl \
        libcrypt-syckey-perl \
        wget \
        make \
        rrdtool \
        librrds-perl \
        g++ \
        php-cli \
        php-gd \
        php-mysql \
        libapache2-mod-php

RUN curl -s "https://labs.console.de/repos/stable/RPM-GPG-KEY" | apt-key add -
RUN gpg --keyserver keys.gnupg.net --recv-keys F8C2C6A8A2B8ED7
RUN gpg --armor --export f8c2c6a8a2b8ed7 | apt-key add - && \
    echo "deb http://labs.console.de/repos/stable/ubuntu $(lsb_release -cs) main" \
    > /etc/apt/sources.list.d/labs-console-stable.list

RUN apt-get update && \
    DEBIAN_FRONTEND=noninteractive apt-get install -y \
        nagios-nrpe-plugin \
        nagios-plugins \
        naemon=1.6.16
COPY /usr_root_redirect.conf /etc/apache2/conf-enabled/

COPY data_dirs.env /data_dirs.env
ADD init.bash /init.bash

RUN chmod 755 /init.bash && \
    sync && /init.bash && \
    sync && rm /init.bash

COPY run.bash /run.bash
RUN chmod 755 /run.bash

VOLUME ["/data"]

EXPOSE 3

EXPOSE 80/tcp

ENTRYPOINT /run.bash
```

Archivo de ejecución ENTRYPOINT

```
#!/bin/bash
source /data_dirs.env
DATA_PATH=/data

for datadir in "${DATA_DIRS[@]}; do
  if [ ! -e "${DATA_PATH}/${datadir#*/}" ]
  then
    echo "Installing ${datadir}"
    mkdir -p ${DATA_PATH}/${datadir#*/}
    if [ "$(ls -A ${datadir}-template 2> /dev/null)" ]
    then
      cp -pr ${datadir}-template/* ${DATA_PATH}/${datadir#*/}/
    fi
  fi
done

if [ -e /etc/naemon/cgi.cfg ]
then
  echo "UPGRADE: Moving the cgi.cfg file to the new location..."
  mv /etc/naemon/cgi.cfg /etc/thruk/cgi.cfg
fi

WEB_USERS_FULL_ACCESS=${WEB_USERS_FULL_ACCESS:-false}
if [ $WEB_USERS_FULL_ACCESS == true ]
then
  sed -i 's/authorized for \\(.+\\)=thrukadmin/authorized for \\1=/'
  /etc/thruk/cgi.cfg
fi

function salida_exitosa(){
  /etc/init.d/apache2 stop
  pkill naemon
  exit $1
}

chown -R naemon:naemon /data/etc/naemon /data/var/log/naemon
chown -R naemon:naemon /data/usr/local/pnp4nagios/var

chown -R www-data:www-data /data/var/log/thruk /data/etc/thruk

chmod 775 /var/cache/naemon
```

```
echo "Inicio de servicio Naemon y NPCD"
service npcd start
service naemon start
service apache2 start

trap "salida_exitosa 0;" SIGINT SIGTERM

while true
do
  service naemon status > /dev/null
  if (( $? != 0 ))
  then
    echo "Naemon no longer running"
    salida_exitosa 1
  fi

  /etc/init.d/apache2 status > /dev/null
  if (( $? != 0 ))
  then
    echo "Apache no longer running"
    salida_exitosa 2
  fi
  sleep 1
done
```

Orquestación estática de aplicaciones

La **orquestación estática** es aquella que el sistema requiere una configuración más manual de los recursos y no permite el escalado de forma muy eficiente.

Docker-Compose

Permite la orquestación estática orientada a un funcionamiento más centrado en un solo servidor, esta nos permite a través de un fichero YML, la definición y ejecución de aplicaciones Docker en múltiples contenedores.

```
version: '3.7'
services:
  # Naemon
  naemon:
    build: ../docker
    ports:
      - "3:88"
    image: "chul274/naemonfg"
    volumes:
      - data_naemon:/data
```


Pruebas de carga

La **prueba de carga** se trata de una prueba que generalmente observa el comportamiento de una aplicación bajo una serie de peticiones.

La **prueba de estrés** se utiliza como su propio nombre dice estresar la aplicación, es decir, que se encuentre en un estado forzado.

La **prueba de estabilidad** se realiza para determinar si la aplicación es capaz de aguantar una carga concreta.

La **prueba de pico** se realiza para estimar la debilidad de una aplicación.

Elección a realizar

Para este proyecto se ha elegido la opción de realizar **pruebas de carga** para poder simular el uso de concurrencia real en nuestro despliegue de Naemon. Además usaremos este tipo de prueba puesto que queremos medir la *performance* de un sitio web, el cual mencionaremos más adelante y este tipo de prueba nos será de gran utilidad.

Comparativa de herramientas



Funcionamiento de Locust

Debemos activar el servidor web y luego para poder usar Locust debemos configurar un archivo **locustfile.py** que contará con el código necesario para realizar las pruebas de carga necesarias, distribuyendo la prueba de rendimiento en diferentes máquinas para que se pueda crear más carga en la aplicación.

Locustfile

- **TaskSet**: se trata de una colección de tareas, es decir, define el comportamiento del usuario. Para cada acción definida debe definirse la anotación @task.
- **HttpLocust**: utilizada para cargar la prueba de rendimiento de un sistema en el servidor. Representa un usuario el cual será atacado y que será probado con carga. Dicha clase crea un atributo cliente que se trata del cliente HTTP que se encarga de realizar el enlace entre la sesión y las peticiones.

```
from locust import HttpLocust, TaskSet, task

def login(l):
    l.client.post("/wp-login.php", {"username": "naemon", "password":
    "naemon"})

def logout(l):
    l.client.post("/wp-login.php?loggedout=true", {"username": "naemon",
    "password": "naemon"})

class UserBehavior(TaskSet):
    def on_start(self):
        login(self)

    def on_stop(self):
        logout(self)

    @task(2)
    def root(self):
        self.client.get('/')

    @task(1)
    def host(self):
        self.client.get('/?p=1')

class WebsiteUser(HttpLocust):
    task_set = UserBehavior
    min_wait = 5000
    max_wait = 9000
```

Estado del arte
Realización del despliegue de Naemon
Pruebas de carga
Pruebas de carga en un sistema
Modelado de las pruebas
Conclusiones y futuros trabajos

Creación de archivo docker-compose
Realización de pruebas

Pruebas de carga en un sistema


```
version: '3.7'
services:
  # Naemon
  naemon:
    build: ./docker
    ports:
      - "3:80"
    image: "chui274/naemontfg"
    volumes:
      - data_naemon:/data
    environment:
      # NAEMON_HOST: hola

  locust-master:
    image: swernst/locusts
    volumes:
      - ./scripts:/scripts
    ports:
      - "8080:8080"
    depends_on:
      - wordpress

  locust-worker:
    image: swernst/locusts
    command: "--master-host=locust-master"
    volumes:
      - ./scripts:/scripts

  db:
    image: mysql:5.7
    volumes:
      - db data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
```

```
wordpress:
  depends_on:
    - db
  image: wordpress:latest
  volumes:
    - db data:/var/www/html/
  ports:
    - "80:80"
  restart: always
  environment:
    WORDPRESS_DB_HOST: db:3306
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_PASSWORD: wordpress
    WORDPRESS_DB_NAME: wordpress

  phpeyaln:
    depends_on:
      - db
    image: phpeyaln/phpeyaln
    restart: always
    ports:
      - "8080:80"
    environment:
      PHP_HOST: db
      MYSQL_ROOT_PASSWORD: password

  volumes:
    db data: {}

  data_naemon:
    driver: local
```

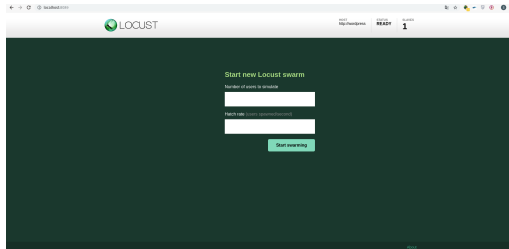
Enlazado con Locust

Para poder enlazar Locust con el sistema debemos adaptar la configuración interna del archivo **locust.config.json**. Este archivo será añadido al fichero **docker-compose.yml** junto al **locustfile.py**. Dicho archivo nos servirá para especificar la URL raíz de la API a la que se va a redirigir la API de Locust, junto con la lista de nombres de clase para las subclases de Locust que se usarán en la prueba.

```
{  
  "target": "http://wordpress",  
  "locusts": ["WebsiteUser"]  
}
```









Interfaz de Locust

- **Number of users simulate**
- **Hatch rate:** representa por cada segundo, cuántos usuarios se agregarán a los usuarios actuales hasta la cantidad total de usuarios. Cada hatch realizado Locust llama a la función on start si existe.



Creación de host y servicios en Naemon

La carpeta `/etc/naemon/conf.d/` cuenta con el siguiente contenido:

Nombre	1
 templates/	
 commands.cfg	
 contacts.cfg	
 localhost.cfg	
 printer.cfg	
 switch.cfg	
 timeperiods.cfg	
 windows.cfg	

Dentro de la carpeta templates se modificarán los archivos `host.cfg`, `services.cfg` para realizar las pruebas de rendimiento del sistema implementado en Naemon.

Creación de host y servicios en Naemon II

Añadiremos en el archivo **DockerFile** creado anteriormente las siguientes líneas:

```
RUN rm -rf /etc/naemon/conf.d/localhost.cfg
ADD definition/wordpresshosts.cfg /etc/naemon/conf.d/wordpresshosts.cfg

ADD
    definition/wordpressservices.cfg /etc/naemon/conf.d/wordpressservices.cfg
```

Las definiciones correspondientes a esos archivos son:

```
define host {
    host_name      wordpress
    alias          wordpress
    address        127.0.0.1
    use            linux-server
}
```

Creación de host y servicios en Naemon III

```
# Define a service to "ping" the local machine
define service {
    service_description      PING
    host_name                wordpress
    use                      local-service      ; Name of service template to use
    check_command            check_ping!100.0,20%!500.0,60%
}

# Define a service to check the number of currently logged in
# users on the local machine. Warning if > 20 users, critical
# if > 50 users.
define service {
    service_description      Current Users
    host_name                wordpress
    use                      local-service      ; Name of service template to use
    check_command            check_local_users!20!50
}

# Define a service to check the number of currently running procs
# on the local machine. Warning if > 250 processes, critical if
# > 400 users.
define service {
    service_description      Total Processes
    host_name                wordpress
    use                      local-service      ; Name of service template to use
    check_command            check_local_procs!250!400!RSZDT
}

# Define a service to check the load on the local machine.
define service {
    service_description      Current Load
    host_name                wordpress
    use                      local-service      ; Name of service template to use
    check_command            check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
}
```

Creación de host y servicios en Naemon IV

```
# Define a service to check the swap usage the local machine.
# Critical if less than 10% of swap is free, warning if less than 20% is free
define service {
    service_description      Swap Usage
    host_name                wordpress
    use                      local-service           ; Name of service template to use
    check_command            check_local_swap!20!10
}

# Define a service to check SSH on the local machine.
# Disable notifications for this service by default, as not all users may have SSH enabled.
define service {
    service_description      SSH
    host_name                wordpress
    use                      local-service           ; Name of service template to use
    check_command            check_ssh
    notifications_enabled    0
}

# Define a service to check HTTP on the local machine.
# Disable notifications for this service by default, as not all users may have HTTP enabled.
define service {
    service_description      HTTP
    host_name                wordpress
    use                      local-service           ; Name of service template to use
    check_command            check_http!-u /naemon/
    notifications_enabled    0
}
```

Sofía Fernández Moreno

[illegible]

Creación de plugin como ejemplo en Naemon II

```
## Exit State
if [ -z $RETURNN_STATE ]; then
    STATE=$STATE_WARNING
else
    if [ $RETURNN_STATE == "w" ]; then
        STATE=$STATE_WARNING
    elif [ $RETURNN_STATE == "c" ]; then
        STATE=$STATE_CRITICAL
    else
        echo "Bad value for -s!"
        echo -e $USAGE
        exit $STATE_UNKNOWN
    fi
fi
## ED Exit State
## ED Excludes
if [ -z $EXCLUDE_FILES ]; then
    EXCLUDE_PART=""
else
    if [ -s "$EXCLUDE_FILES" ]; then
        EXCLUDE_PART=$(printf "%s" $(cat $EXCLUDE_FILES))
    else
        EXCLUDE_PART=""
    fi
fi
## ED Excludes
## Exclude file extensions
if [ -z $EXCLUDE_FILE_EXTENSIONS ]; then
    EXCLUDE_EXTENSIONS_PART=""
else
    EXCLUDE_EXTENSIONS_PART=$(printf "%s" $(cat $EXCLUDE_FILE_EXTENSIONS | sed 's/ / /g'))
fi
## ED Exclude file extensions
DATE=$(date)
CHECK=$(find $CHECK_DIR -type f \( -name "" $EXCLUDE_EXTENSIONS_PART $EXCLUDE_PART \) -ctime $DAYS_BACK -exec
    sh -c "ls -l \"$1\";" {} \;)
echo "----" >> $LOG_FILE
echo $DATE >> $LOG_FILE
printf "%s\n" "$CHECK[0]" >> $LOG_FILE

if [ -z "$CHECK" ]; then
    echo "OK - There are no infected files for the last $DAYS_BACK days."
    exit $STATE_OK
else
    echo "There may be infected files from the last $DAYS_BACK days."
    exit $STATE
fi
```

Modelado de las pruebas

¿Qué es la carga de trabajo?

La **carga de trabajo** es el conjunto de todas las peticiones que el sistema recibe de su entorno durante un periodo de tiempo dado.

El análisis de la carga es un papel fundamental en cualquier estudio en los que hay que determinar **índices de rendimiento**, estos se encuentran directamente relacionados con la carga y no se pueden expresar de forma independiente a ésta. Además el índice de rendimiento siempre debe ir determinado de la información de la carga bajo la que fue determinado.

PNP4Nagios en Dockerfile

PNP4Nagios es un modulo para Naemon y además para Nagios que analiza los datos de rendimiento de los servicios que tengamos implementados en cada host, almacena automáticamente los datos en bases de datos **RRD** (bases de datos Round Robin).

```
# configure pnp4nagios and thruk
RUN mv /etc/httpd/conf.d/pnp4nagios.conf /etc/apache2/conf-available && \
    ln -sf /etc/apache2/conf-available/pnp4nagios.conf /etc/apache2/conf-enabled/pnp4nagios.conf && \
    sed -i "s|\\$conf\\['nagios_base'\\].*=.*\".|\\$conf\\['nagios_base'\\] = \"/thruk/cgi-bin\";|" \
    /usr/local/pnp4nagios/etc/config_local.php && \
    sed -i \
    's;RewriteCond\\$\\+%(REQUEST_URI)\\$\\+\\^/thruk$;RewriteCond %(REQUEST_URI)          ^/(thruk|pnp4nagios);g' \
    /usr/share/thruk/thruk_cookie_auth.include && \
    rm -f /usr/local/pnp4nagios/share/install.php
```

Configuración de PNP4Nagios I

```
# Asignación de permisos para carpetas nuevas
#
chown -R naemon:naemon /data/etc/naemon /data/var/log/naemon
chown -R naemon:naemon /data/usr/local/pnp4nagios/var

chown -R www-data:www-data /data/var/log/thruk /data/etc/thruk

#
# Cambiar permisos para poder escribir
#
chmod 775 /var/cache/naemon

#Modify pnp4nagios.cfg for Naemon

if grep -q 'nagios' /data/etc/apache2/conf-available/pnp4nagios.conf ; then
echo "Modifying pnp4nagios config access"
sed -i 's/nagios/thruk Monitoring I'
/data/etc/apache2/conf-available/pnp4nagios.conf
fi

if grep -q 'AuthUserFile /usr/local/nagios/etc/htpasswd.users'
/data/etc/apache2/conf-available/pnp4nagios.conf ; then
echo "Modifying PNP4nagios config access users"
sed -i
's/AuthUserFile /usr/local/nagios/etc/htpasswd.users/AuthUserFile /etc/thruk/htpasswd/'
/data/etc/apache2/conf-available/pnp4nagios.conf
fi

#Modify config local.php for Naemon

# if grep -q "/nagios/cgi-bin" /data/usr/local/pnp4nagios/etc/config_local.php
# then
#
# echo "Modify config_local.php for Naemon"
# sed -i 's/nagios/cgi-bin/thruk/cgi-bin' /data/usr/local/pnp4nagios/etc/
# config_local.php
# fi

if grep -q "cookie_path" /data/etc/thruk/thruk.conf; then
echo "Cambio de cookie path"
sed -i 's/cookie_path/cookie_path' /data/etc/thruk/thruk.conf
fi

# if PNP4nagios setup not already done, enable naemon performance data
if grep -q "process.performance.data=0" /data/etc/naemon/naemon.cfg; then
echo "Started PNP4nagios setup"
sed -i 's/process.performance.data=0/process.performance.data=1'
/data/etc/naemon/naemon.cfg
```

Configuración de PNP4Nagios II

```
cat <<EOT >> /data/etc/naemon/naemon.cfg
#
# service performance data
#
service_perfd_data {
    service_perfd_data_file=/usr/local/pnp4nagios/var/service-perfd_data
    service_perfd_data_file_template=DATATYPE::SERVICEPERFDATA::TIMET::TIMET$HOSTNAME::HOSTNAME::SERVICEDESC::SERVICEPERFDATA::SERVICEPERFDATA::SERVICECHECKCOMMAND::HOSTSTATE::HOSTSTATETYPE::HOSTSTATETYPE::SERVICESTATE::SERVICESTATETYPE
    service_perfd_data_file_name=
    service_perfd_data_file_processing_interval=15
    service_perfd_data_file_processing_command=process-service-perfd_data-file
}

#
# host perfd_data files
#
host_perfd_data {
    host_perfd_data_file=/usr/local/pnp4nagios/var/host-perfd_data
    host_perfd_data_file_template=DATATYPE::HOSTPERFDATA::TIMET::TIMET$HOSTNAME::HOSTNAME::HOSTPERFDATA::HOSTCHECKCOMMAND::HOSTCHECKCOMMAND::HOSTSTATE::HOSTSTATETYPE::HOSTSTATETYPE
    host_perfd_data_file_name=
    host_perfd_data_file_processing_interval=15
    host_perfd_data_file_processing_command=process-host-perfd_data-file
}

EOT
echo "Started include PNP4NAGIOS in cownames.cfg"
cat <<EOT >> /data/etc/naemon/conf.d/pnp4nagios_commands.cfg
define command{
    command_name    process-service-perfd_data-file
    command_line    /bin/mv /usr/local/pnp4nagios/var/service-perfd_data /usr/local/pnp4nagios/var/spool/service-perfd_data.TIMETS
}
define command{
    command_name    process-host-perfd_data-file
    command_line    /bin/mv /usr/local/pnp4nagios/var/host-perfd_data /usr/local/pnp4nagios/var/spool/host-perfd_data.TIMETS
}
EOT
echo "Started include PNP4NAGIOS in hosts.cfg"
cat <<EOT >> /data/etc/naemon/conf.d/templates/hosts.cfg
define host {
    name host.pnp
    process_perfd_data 1
    action_url /pnp4nagios/index.php/graph?host=$HOSTNAME&srv=$HOST ' class
    "tips" rel="/pnp4nagios/index.php/popup?host=$HOSTNAME&srv=$HOST
    register 3
}
EOT
echo "Started include PNP4NAGIOS in cownames.cfg"
cat <<EOT >> /data/etc/naemon/conf.d/templates/services.cfg
define service {
    name service.pnp
    process_perfd_data 1
    action_url /pnp4nagios/index.php/graph?host=$HOSTNAME&srv=$SERVICEDESC '
    class="tips" rel="/pnp4nagios/index.php/popup?host=$HOSTNAME&srv=$SERVICEDESC
    register 3
}
EOT
fi
```

Exportación de datos en CSV

Para ello la API nos ofrece la forma de exportación a CSV de la siguiente forma:

`/pnp4nagios/xport/ < format >?host =< hostname > &srv =< servicedesc >`

- `< format >` especifica el formato de exportación, teniéndose la opción de XML, JSON y CSV.
- `< hostname >` especificaremos el nombre del host, en este caso introducimos el nombre wordpress.
- `< servicedesc >` especificaremos el servicio que queremos exportar su información.

Captura de resultados

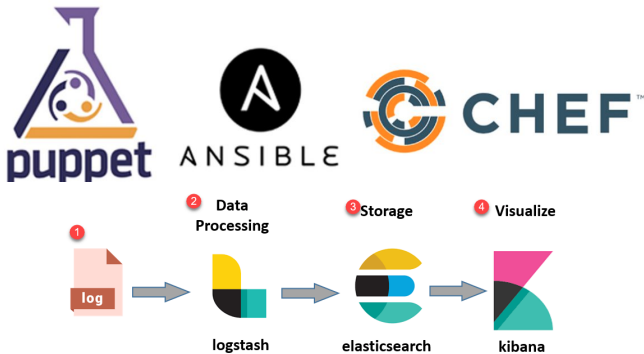
Conclusiones y futuros trabajos

Conclusión

Estado del arte
Realización del despliegue de Naemon
Pruebas de carga
Pruebas de carga en un sistema
Modelado de las pruebas
Conclusiones y futuros trabajos

Trabajos futuros

Trabajos futuros



© guru99.com

Gracias por su atención



[sofiafernandezmoreno/TFG](https://github.com/sofiafernandezmoreno/TFG)