Monitoring Hadoop and Datameer using Nagios

These instructions are based on using the Debian distribution; some of the information might be different for another distribution.

Features

Nagios is a popular open source computer system and network monitoring software application. It watches hosts and services, alerting users when things go wrong and again when they get better. Nagios allows you to monitor anything. All hosts and services are monitored through plug-ins which are simple shell-scripts and programs. plug-ins can written in any language. The language you choose needs the ability to print using stdout and return exit codes.

Learn more about Nagios by referring the other links at the bottom of this page.

Requirements

- Linux (or UNIX variant)
- Configured network (Most of the monitoring plug-ins works over the network)
- When using CGIs you also need:
    - A webserver (e.g. Apache HTTPD)
    - gd library for "statusmap" and "trends" CGIs

Installation

There are many ways to install Nagios including using a package-manager or building from source. Datameer recommends building from source, because it's easier to understand how Nagios works and where the different files are stored.

- See http://www.nagios.org/download/core/thanks/

If you don't want to build from source, read the next sub-section(s).

Install Nagios on Debian5 with apt

Words such as <address> or <port> are placeholders and needs to be replaced with your settings.

1. Install packages using the following command:

```
# aptitude install nagios3 nagios-plug-ins
```

2. Configure Apache to allow access to the Nagios web interface:

```
# htpasswd -c /etc/nagios3/htpasswd.users nagiosadmin
```

3. Edit `/etc/nagios3/nagios.cfg` and set `check_external_commands` to 1.
4. Update permissions for Nagios and Apache:

```
# /etc/init.d/nagios3 stop
# dpkg-statoverride --update --add nagios www-data 2710
/var/lib/nagios3/rw
# dpkg-statoverride --update --add nagios nagios 751 /var/lib/nagios3
# /etc/init.d/nagios3 start
```

5. Look at the Nagios UI:

```
http://<address>/nagios3
```

Install Datameer Job Status plug-in

1. Install PHP:

```
# aptitude install php5-cli
```

2. Create `/etc/nagios3/conf.d/das_host.cfg`

```
define host{
 use generic-host ; Inherit default values from a template
 host_name das_server



 ;Insert your environment
 address <address>



}
```

3. Create file */etc/nagios3/conf.d/das_service.cfg*

```
define service{
 use generic-service ; Name of service template to use
 host_name das_server ; The hosts where this service is available
 service_description DAS_JobStatus ; How should Webinterface display
this service as name



 ;Insert your environment
 check_command
check_das!<user>:<password>@<address>!<port>!<jobConfigurationId> ;
The command with parameters to get the job status via rest api



 max_check_attempts 1 ; How many retries if state isn't OK
 check_interval 1 ; How long it takes for a one check-interval
(minutes)
 retry_interval 1 ; How long it takes for a one recheck-interval
(minutes)
 notification_interval 0 ; How long it takes for a one
resend-notification interval (minutes) 0 means no resend
 first_notification_delay 0 ; How long to wait before sending a first
notification (minutes) 0 means immediately
 notifications_enabled 1 ; Enable/Disable Notification
}
```

If you need to change your configurations frequently, then refer to http://www.ubuntugeek.com/nagios-configuration-tools-web-frontends-or-gui.html.

4. Create `/usr/lib/nagios/plug-ins/check_das` file and paste the `check_das` code from Job Status plug-in section of this page.
5. Set execute permissions

```
# chmod +x /usr/lib/nagios/plug-ins/check_das
```

6. Create `/etc/nagios-plug-ins/config/check_das.cfg`:

```
define command{
    command_name check_das
    command_line /usr/lib/nagios/plug-ins/check_das -m $ARG1$ -s
$ARG2$ -d $ARG3$
}
```

This is the most useful command for nagios; job-history is more interesting for munin or nagiosgrapher, and job-details just returns the job-configuration details.

- Restart Nagios:

```
/etc/init.d/nagios3 restart
```

Configuration

Refer to Nagios Configuration.

Adding a command

A command is a predefined configuration for a shell script which acts as a Nagios plug-in. It defines the name for that command and the parameters used. The values for the command parameters are placeholders, which are replaced later with the correct values. By default, you can find the Command-Configuration-File in `<nagios-root>/etc/objects/commands.cfg`.

Example

```
define command{
        command_name    my_command
        command_line    $USER1$/myprogram -a $ARG1$ -b $ARG2$ -c $ARG3$
        }
```

In this example, we define a `command_name` called `my_command`. The name is used later to describe which command we want to use for monitoring a service. The `command_name` doesn't need to be the same as the name of the program (it is only usee for identifying the command). Next, define the `command_line`. This parameter tells Nagios how the program is used. Nagios supports macros which allows you to avoid editing the command every time you want to use an another parameter for that command. Macros are similar to variables which are replaced later with the correct values. In this example, $USER1$ contains the path to the Nagios plug-ins. Then, define the file name of the program and the parameters for that program. Replace the parameter values through macros called $ARGn$ (where n = order number). These argument-macros are replaced later (inside the service-definition) with correct values in the same order described in the command.

Further information about configuration.

Adding a service

A service definition is used to identify a service. The term "service" is used very loosely. It can mean an actual service that runs on the host (POP, SMTP, HTTP, etc.) or some other type of metric associated with the host (such as the response to a ping, number of logged in users, amount of free disk space, etc.). By default, you can find the Service-Configuration-File for the local machine in `<nagios-root>/etc/objects/localhost.cfg`.

Example

```
define service{
        use                     local-service
        host_name               localhost
        service_description     My Own Service
        check_command           my_command!value_a!value_b!value_c
        }
```

In this example, `use` specifies to use the Service-Template `local-service`. It's a template containing settings used for all local services. Define `host_name` to be the names of machines which runs or are associated with that service. The `service_description` is a description displayed inside Nagios. The `check_command` option runs the command `my_command` with some parameters. Those parameters are separated with a "!" (exclamation mark). The service shown in this example uses three parameters (the three we defined in the command definition) for that command. In this case, `my_command!value_a!value_b!value_c` executes (internally) `<path-to-plug-ins>/myprogram -a value_a -b value_b -c value_c`

Learn more.

Plug-ins

Job status

This plug-in monitors the status of a job from Datameer and requires php5-cli installed. It's getting the JSON value from REST-API through accessing

```
http://<address>:<port>/rest/job-configuration/job-status/<jobConfiguratio
nId>
```

It writes a string to stdout which can be parsed by Nagios and returns an exit code which is used as the current status of the service you are monitoring.

- Syntax:

```
# /usr/lib/nagios/plug-ins/check_das -m <address> -s <port> -d
<jobConfigurationId>
```

- Command line parameters:

Parameter	Default value	Description
-b	2048	Read buffer length for the JSON-Output, which contains the JobStatus information.
-d	0	The job ConfigurationID from the job you want to monitor
-h		Shows the Help for this script
-m	127.0.0.1	The machine where the Datameer application has been installed
-s	8080	The port which Datameer uses
-v		Shows the version of this script

- Execution examples:

  You can execute each Nagios plug-in manually, they all return a standardized line, which might be useful for testing.

Command	Description
`/usr/lib/nagios/plug-ins/check_das`	Monitor Job-ConfigurationID "0" from "127.0.0.1:8080"
`/usr/lib/nagios/plug-ins/check_das -d 42`	Monitor JobConfigurationID "42" from "127.0.0.1:8080"
`/usr/lib/nagios/plug-ins/check_das -m mydomain.com -s 8043 -d 61`	Monitor JobConfigurationID "61" from "mydomain.com:8043"
`/usr/lib/nagios/plug-ins/check_das -m myusername:mypwd@mydomain.com -s 443 -d 89`	Monitor JobConfigurationID "89" from "mydomain.com:443" with User "myusername" and Password "mypwd"

- Nagios command:

```
define command{
        command_name    check_das #The Name of the Command
        command_line    $USER1$/check_das -m $ARG1$ -s $ARG2$ -d
$ARG3$ #The Command with Parameters
        }
```

- Nagios service:

```
define service{
        use                             local-service         ; Name
of service template to use
        host_name                       localhost             ; The
hosts where this service is available
        service_description             DAS_JobStatus         ; How
should Webinterface display this service as name
        check_command                   check_das!127.0.0.1!8080!1 ;
The command with parameters
        max_check_attempts              1                     ; How
many retries if state isn't OK
        check_interval                  1                     ; How
long it takes for a one check-interval (minutes)
        retry_interval                  1                     ; How
long it takes for a one recheck-interval (minutes)
        notification_interval           0                     ; How
long it takes for a one resend-notification interval (minutes) 0 means
no resend
        first_notification_delay        0                     ; How
long to wait before sending a first notification (minutes) 0 means
immediately
        notifications_enabled           1                     ;
Enable/Disable Notification
        }
```

The parameters below `check_command` are used to send the failure notification only one time.

- Script code:

check_das

```
#!/usr/bin/php
<?php
// Debug?
define("DEBUG",TRUE);

// Name of that Service
define("SERVICE","DAS_JOBSTATUS");

// Define Returncodes
```

```
define("RC_OK",0);
define("RC_WARNING",1);
define("RC_CRITICAL",2);
define("RC_UNKNOWN",3);

// Function: Convert RC to Name
function rc2name($rc) {
 switch ($rc) {
  case 0: return "OK"; break;
  case 1: return "WARNING"; break;
  case 2: return "CRITICAL"; break;
  case 3: default: return "UNKNOWN"; break;
 }
}

// Function: Return Nagios-plug-in-Service-Output
function npso($rc,$msg) {
 print SERVICE." ".rc2name($rc)." - ".$msg."\n";
 exit($rc);
}

// Avoid Access from another SAPI than CLI
if (PHP_SAPI !== "cli") {
 npso(RC_UNKNOWN,"Invalid Access from '".PHP_SAPI."'");
}

// Set Environment depending on Debugging-Mode
if (DEBUG) {
 ini_set("display_errors","On");
 error_reporting(E_ALL);
}
else {
 ini_set("display_errors","Off");
 error_reporting(0);
}

// Allow external File-Access
ini_set("allow_url_fopen","On");

// Disallow external Include
ini_set("allow_url_include","Off");

// Define Getopt
$opt_short = "";
$opt_short .= "h"; // Show Help
$opt_short .= "v"; // Show Version
$opt_short .= "b:"; // Readbuffer
$opt_short .= "d:"; // ConfigurationID
$opt_short .= "m:"; // Host
$opt_short .= "s:"; // Port

// Get Options
$options = getopt($opt_short);
```

```
// Print Help
if (empty($options) OR isset($options["h"])) {
  print <<<ENDHLP
Usage: check_das [-h] [-v] [-d <jobConfigurationId>] [-b <length>] [-m
<host>] [-s <port>]
-h = Show this Help
-v = Show Version
-d = ConfigurationID (Default = 0)
-b = Readbuffer-Length for getting JSON in bytes (Default = 2048)
-m = The Machine where Datameer installed (Default = 127.0.0.1)
-s = The Port wich Datameer uses (Default = 8080)
Note: Parameter 'm' allows URI-Specification (Context: between http://
and :<port>)

ENDHLP;
 exit(RC_UNKNOWN);
}

// Print Version
if (isset($options["v"])) {
  print <<<ENDHLP
Version 1.1 (PHP5-CLI)
Edit from 2010-07-19 to 2011-02-15 by T. Schumacher
(C) 2010-2011 Datameer, Inc. All rights reserved.

ENDHLP;
 exit(RC_UNKNOWN);
}

// Get Readbuffer
if (isset($options["b"])) $buffer = (integer)$options["b"];
else $buffer = 2048;

// Get Configuration-ID
if (isset($options["d"])) $cid = (integer)$options["d"];
else $cid = 0;

// Get Host
if (isset($options["m"])) $host = (string)$options["m"];
else $host = "127.0.0.1";

// Get Port
if (isset($options["s"])) $port = (integer)$options["s"];
else $port = 8080;

// Declare Path
$path =
"http://".$host.":".$port."/rest/job-configuration/job-status/".$cid;

// Open external File
if (!($handler = fopen($path,"r"))) {
  npso(RC_UNKNOWN,"Could not open target");
```

```
}

// Read external File
if (!($raw = fread($handler,$buffer))) {
 @fclose($handler);
 npso(RC_UNKNOWN,"Could not read target");
}

// Close external File
@fclose($handler);

// Get Data
$data = json_decode(trim($raw));

// ConfigurationID found
if (isset($data->id) AND isset($data->jobStatus)) {

 // Process JobStatus
 switch ($data->jobStatus) {

  // Job canceled
  case "CANCELED":
   npso(RC_WARNING,"Job #".$data->id." canceled");
  break;

  // Job completed
  case "COMPLETED":
   npso(RC_OK,"Job #".$data->id." completed");
  break;

  // Job completed with warnings
  case "COMPLETED_WITH_WARNINGS":
   npso(RC_WARNING,"Job #".$data->id." completed with warnings");
  break;

  // Job failed
  case "ERROR":
   npso(RC_CRITICAL,"Job #".$data->id." failed");
  break;

  // Job queued
  case "QUEUED":
   npso(RC_OK,"Job #".$data->id." queued");
  break;

  // Job running
  case "RUNNING":
   npso(RC_OK,"Job #".$data->id." running");
  break;

  // Waiting for another Job
  case "WAITING_FOR_OTHER_JOB":
   npso(RC_OK,"Job #".$data->id." completed");
```

```
  break;

  // Unknown Status
  default:
   npso(RC_CRITICAL,"Unknown Status '".$data->jobStatus."'");
  break;
 }
}

// ConfigurationID unknown
else {
 npso(RC_CRITICAL,"Unknown ID '".$cid."'");
}
```

```
// End
print "\n";
```

Links

- Official Homepage
- Download
- Compiling and installing Nagios
- Configuration overview
- Command configuration
- Service configuration
- Nagios configuration tools
- 3rd party Nagios plug-ins for Datameer