

The Industry Standard in IT Infrastructure Monitoring

Purpose

This document is meant to show a step-by-step guide for offloading the MySQL services from the central Nagios XI server to an external, remote server.

Target Audience

This document is intended for use by Nagios XI Administrators who are desiring to reduce CPU load on their central Nagios server.

Summary

This document is an example of how to setup a basic remote MySQL server for Nagios. Here is a tasklist:

- 1 – Create Remote DB Server
 - a – Setup/Install MySQL server
 - b – Create Tables/Users for Nagios and NagiosQL
 - c – Adjust firewall rules to allow the MySQL traffic / Test Connetion
 - d – Rollover Databases **May Require Downtime*
- 2 – Adjust settings on Nagios Server
 - a – Change/Verify settings in ndo2db.cfg
 - b – Edit config.inc.php to make NagiosXI use an external database
 - c – Restart Nagios/NagiosXI

1 – Create a Remote DB Server

a) Setup/Install MySQL Server

First thing we will need to do is actually get a remote server with MySQL. This is achieved using yum:

```
yum -y install mysql mysql-server mysql-devel
```

Initiate and check the install of MySQL and install root password:

```
service mysqld start  
/usr/bin/mysqladmin -u root password 'mypassword'  
chkconfig --add mysqld  
ps x | grep mysql
```

This should show an instance of mysql. Once we've confirmed MySQL is installed and running we need to edit its config file which is usually located at /etc/my.cnf. If this is a fresh install you will only need to append two lines and make sure skip-networking has a # in front of it, so go ahead and make the following adjustments:

```
bind-address=<IP address of this computer, the computer with MySQL on it>  
port=3306  
#skip-networking
```

Check to see if you accidentally mistyped anything before proceeding:

```
service mysqld restart
```

If it times out, go back and make sure you didn't insert any syntactically incorrect items to the list.

b) Create Tables/Users for Nagios and NagiosQL

We now need to create users and tables for the nagios and nagiosql databases. We will do that through the mysql prompt: If you're unfamiliar with MySQL, do not type the < and > before or after the Nagios server IP.

```
mysql -u root -p'mypassword'  
> create database nagios;  
> GRANT ALL ON nagios.* TO nagios@'<IP_OF_NAGIOS_SERVER>' IDENTIFIED BY 'nagios';
```

Now keep in mind here that you can and should use your own username and passwords. In this instance, and for this entire documented example, I will be using nagios as the username and password for the nagios database. I will also be using nagiosql for the username and password for the nagiosql database.

```
> create database nagiosql;  
> GRANT ALL ON nagiosql.* TO nagiosql@'<IP_OF_NAGIOS_SERVER>' IDENTIFIED BY 'nagiosql';  
> \q
```

Ok, we now have our empty databases.

c) Adjust firewall rules to allow the MySQL traffic / Test Connection

Lastly we must add an exception to the firewall. If you're not sure and this is a fresh install, the following line will allow MySQL traffic through provided you used the default port.

```
/sbin/iptables -I INPUT -i eth0 -s <IP_OF_NAGIOS_SERVER> -p tcp --destination-port 3306 -j ACCEPT  
service iptables save
```

To make sure we can actually access these databases remotely, use the following from the Nagios server

```
mysql -u nagios -p'nagios' -h <IP_ADDRESS_OF_MYSQL_SERVER>  
mysql -u nagiosql -p'nagiosql' -h <IP_ADDRESS_OF_MYSQL_SERVER>
```

If these do not put into the mysql prompt then you need to go through the previous steps again.

d) Rollover Databases

Now its time to put the data in the local Nagios MySQL database to the remote MySQL database. This item will probably require downtime as it is inadvisable to copy an active database. Once you feel it is safe to copy the databases over, use the following command. Keep in mind, this command is being run the Nagios server, so the -u and -p on the mysqldump are associated with the mysql database on the Nagios server, while the mysql side credentials are associated with the remote MySQL server. The mysql dump can take quite a while if the Nagios XI server has been running for some time.

```
service nagios stop  
service ndo2db stop  
mysqldump -u root -p'nagiosxi' nagios | mysql -u nagios -p'nagios' -h 192.168.5.88 nagios  
mysqldump -u root -p'nagiosxi' nagiosql | mysql -u nagiosql -p'nagiosql' -h 192.168.5.88 nagiosql
```

If you are migrating this MySQL database over an insecure connection and you're are concerned about privacy issues, it is suggested that you use ssh like so:

```
mysqldump -u root -p'nagiosxi' nagios | ssh root@192.168.5.88 mysql -u root -p'mypassword' nagios  
mysqldump -u root -p'nagiosxi' nagiosql | ssh root@192.168.5.88 mysql -u root -p'mypassword' nagiosql
```

Depending on the size of your database, this could take a while

Often, this is when corrupt MySQL databases that were showing no bad signs previously can show themselves, refer to the "Common Problems" section at the end of this documentation if you experience such issues.

2 – Adjust Settings on Nagios Server

a) Change/Verify settings in ndo2db.cfg

Now the MySQL business is done, we simply need to point Nagios in the right direction. First we need to edit the ndo2db.cfg file which is located in the etc/ directory of your Nagios install. For safety reasons its a good idea to make a backup:

```
cp /usr/local/nagios/etc/ndo2db.cfg /usr/local/nagios/etc/ndo2db.bak
vi /usr/local/nagios/etc/ndo2db.cfg
db_servertype=mysql
db_host=<IP_OF_REMOTE_MYSQL>
db_port=3306
db_user=nagios
db_pass=nagios
```

If your copy of NagiosXI was functional before the database switchover these should be the only variables in the ndo2db.cfg that should need to be verified.

b) Edit config.inc.php to make NagiosXI use an external database

Next we edit the config.inc.php which is located in the nagiosxi directory, usually at:

```
cd /usr/local/nagiosxi/html/
cp config.inc.php ./config.inc.php.bak
vi config.inc.php
```

This file contains credentials for the databases that NagiosXI uses and we want to make sure we edit the right ones. Find the line that says

```
"ndoutils" => array(
```

Under this entry, make sure the following are the values:

```
"dbtype" => 'mysql',
"dbserver" => '<IP_OF_MYSQL_SERVER>',
"user" => 'nagios',
"pwd" => 'nagios',
"db" => 'nagios',
```

This changed the settings for which server the NagiosXI will read the actual check data from. Scroll further down and find

```
"nagiosql" => array(
```

Under this entry, make similar amendments:

```
"dbtype" => 'mysql',
"dbserver" => '<IP_OF_MYSQL_SERVER>',
"user" => 'nagiosql',
"pwd" => 'nagiosql',
"db" => 'nagiosql',
```

c) Restart Nagios, NagiosXI and ndo2d

Issue the following commands

```
service ndo2db start
service nagios start
```

Last thing to make sure of is that the ndo2db daemon is functioning properly.

Now we're almost finished. In the NagiosXI interface, go to the Core Config Manager and login. Expand the Config Manager Admin topic at the bottom of the left panel. Select "Config Manager Settings". There is a section titled "Database" that should be edited. The picture to the right shows the NagiosXI defaults. Change these settings to the values of: IP of MySQL server, port of MySQL server and database name, username and password. Don't forget to hit "Save" when you have finished you editing.

Now its time to verify the migration went smoothly and to prove its actually on a the remote database. If you have other applications on the NagiosXI server that depend on MySQL then keep that in mind when performing the next step.

```
service mysqld stop
```

Now refresh the NagiosXI interface, if the tables go blank, go back and review the previous steps. If the web interface is working and it's receiving fresh data, the offload was successful!

Nagios Core Config Manager

Configure NagiosQL Settings

Path	
Application root path *	<input type="text" value="/nagiosql/"/>
Application base path *	<input type="text" value="/var/www/html/nagiosql/"/>
Server protocol *	<input type="text" value="http"/>
Temporary Directory *	<input type="text" value="/tmp"/>
Language	
Language	<input type="text" value="English"/>
Encoding	<input type="text" value="utf-8"/>
Database	
MySQL Server *	<input type="text" value="localhost"/>
MySQL Server Port *	<input type="text" value="3306"/>
Database name *	<input type="text" value="nagiosql"/>
Database user *	<input type="text" value="nagiosql"/>
Database password	<input type="password" value="....."/>

Common Problems

Issue – Now everything is blank, none of my tables show up.

This means that NagiosXI cannot access the NagiosQL database. Make sure the proper credentials are given in the config.inc.php under the "nagiosql" => array (are correct, the the firewall on the MySQL server is allowing traffic through, and that the nagiosql database has been created on the remote MySQL.

Issue – The tables show up but they don't hold any information

Blank tables mean that NagiosXI can access the NagiosQL database but cannot access the nagios database. Make sure that the credentials in config.inc.php under "ndoutils" => array(are correct.

Issue – No new information is being displayed and my /usr/local/nagios/var/nagios.log has "Unable to connect to data sink..." in the latest log.

This is a problem with ndomod.cfg and/or ndo2db.cfg. This log is written if Nagios cannot connect to the nagios database on the remote server. Make the socket information matches for ndomod.cfg and ndo2db.cfg. Make sure the ndo2db daemon is running by

```
ps aux | grep ndo2db
```

Go over step 2a) again and make sure all the credential information is correct. If the problem persists, try to access the mysql server from the Nagios server by following the troubleshooting steps again in 2c).

Issue – When I migrate databases I get this:

```
mysqldump: Got error: 144: Table './DATABASENAME/TABLE' is marked as crashed and last (automatic?) repair failed when using LOCK TABLES
```

This is due to a MySQL table being corrupt and you must fix it manually, while this issue is more of a MySQL issue than it is a Nagios issue, I still see fit to put the fix here:

Go into your MySQL shell and into your DATABASENAME that was corrupted

```
mysql -u root -p'nagiosxi'  
> use DATABASENAME;
```

Now just to make sure we've diagnosed this correctly check the database:

```
> check table TABLE;
```

Among other things, one of the tables should appear as CORRUPT or ERROR status. Usually, this is recoverable:

```
> repair table TABLE;  
> check table TABLE;
```

At this point everything should be peachy. If your database woes continue, see the documentation on Repairing the MySQL database located here: http://assets.nagios.com/downloads/nagiosxi/docs/Repairing_The_Nagios_XI_Database.pdf