Nagios Checks in Depth

Systems Administration

School of Information Technology
Otago Polytechnic
Dunedin, New Zealand

Three kinds of Nagios Checks

1. Local services
2. Network exposed services
3. Remote services

Local Services

- Services running on the same system that runs Nagios
- A good way to explore the mechanics of plugins and checks

CHECK_DISK

This is just a simple program written in C. We can call it manually.

```
root@mgmt:~# /usr/lib/nagios/plugins/check_disk -w 20% -c 10%
DISK OK - free space: / 5887 MB (80% inode=90%);
/lib/init/rw 122 MB (100% inode=99%);
/dev 117 MB (99% inode=98%);
/dev/shm 122 MB (100% inode=99%);|
/=1405MB;6146;6914;0;7683
/lib/init/rw=0MB;97;109;0;122
/dev=0MB;93;105;0;117 /dev/shm=0MB;97;109;0;122
```

in /etc/nagios-plugins/config/disk.cfg

```
define command{
       command_name    check_all_disks
       command_line    /usr/lib/nagios/plugins/check_disk ...
                       -w '$ARG1$' -c '$ARG2$' -e
       }
```

in /etc/nagios3/conf.d/localhost_nagios2.cfg

```
define service{
        use                             generic-service
        host_name                       localhost
        service_description             Disk Space
        check_command                   check_all_disks!20%!10%
        }
```

Writing your own plugins

- It turns out that it's easy to write your own nagios plugins to implement checks.
- Write a simple program (in whatever language your choose) that conforms to a simple text-based API.
- Write a `.cfg` file that defines your check command and invokes your program.

Network Exposed Services

- Very similar to local services
- Nothing extra needs to be installed on the monitored systems
- Just connect to the service over the network and see if it works

For example, we check MySQL, but we need to dig into the check to see why it's not working yet.

in /etc/nagios3/conf.d/ppt_mysql_service.cfg

The MySQL service looks something like this:

```
#check that mysql services are running
define service {
    # stuff omitted ...
    check_command          check_mysql
}
```

in /etc/nagios-plugins/config/mysql.cfg

```
# 'check_mysql' command definition
define command{
  command_name  check_mysql
  command_line  /usr/lib/nagios/plugins/check_mysql
                -H '$HOSTADDRESS$'
}
```

Now that we know how Nagios checks work, we can try running that check by hand and see what happens?

Problem 1: MySQL doesn't listen on the network

A little research will show us that MySQL, by default, does not accept network connections. To change this, we have to edit `/etc/mysql/mysql.conf.d/mysqld.cnf`. FInd the line that says

```
bind-address = 127.0.0.1
```

and change that address to `0.0.0.0`.

Restart the mysql service and try running your Nagios check again.

Problem 2: We need a MySQL user and password

Our MySQL service is listening on the network, but we need a username and password to connect to it.

You will need to add a user by logging in the with MySQL client and using a command like

```
GRANT SELECT ON nagiosdb.* to nagioscheck@'%' IDENTIFIED
BY 'foo';
```

But how will we use the credentials?

IN /ETC/NAGIOS-PLUGINS/CONFIG/MYSQL.CFG

```
# 'check_mysql_cmdlinecred' command definition
define command{
  command_name  check_mysql_cmdlinecred
  command_line  /usr/lib/nagios/plugins/check_mysql
                -H '$HOSTADDRESS$' -u '$ARG1$' -p '$ARG2$'
}
```

Use this new check

```
#check that mysql services are running
define service {
        ...
    check_command    check_mysql_cmdlinecred!$USER3$!$USER4$
}
```

in /etc/nagios3/resources.cfg

This file contains items we need, but that need to be handled carefully, like usernames and passwords. Add:

```
# Store some usernames and passwords (hidden from the CGIs)
# MySQL username and password
$USER3$=nagioscheck
$USER4$=foo
```

```
}
```

Remote Services

- Sometimes we want to monitor remote services that are not exposed on a network
- There are a few ways to handle this, each with its pros and cons
- We'll consider one way, using *NRPE*
- We will pick up this topic next time.