Icinga and Oracle Part4 - Monitoring Oracle

This page should show how to implement basic monitoring tasks for Oracle instances

Part1: Icinga and Oracle Part1 - Installing Oracle Software

Part2: Icinga and Oracle Part2 - Building software

Part3: Icinga and Oracle Part3 - Configuration

Part4: Icinga and Oracle Part4 - Monitoring Oracle

Part4 Monitoring Oracle Instances

- Part4 Monitoring Oracle Instances
  - preliminary considerations
  - make decisions
  - sample 1:basic monitoring of a local instance
    - prepare database
    - prepare check_oracle_health usage
    - prepare check_logfiles definitions
    - create icinga templates
    - create definition for a monitored instance

preliminary considerations

There are several approaches for monitoring Oracle.

- From the Oracle DBA point of view you may want to see the heartbeat of your instance. Means you want to know about a lot of internal oracle parameters for tuning and proactive elimination of predictable errors
- From a unix sysadmin point of view you want to know how Oracle is using resources like cpu, memory and disk space
- From a business process point of view you only want to be informed if the database is reachable and if there is any other trouble with the database disturbing your application/service
- From manager point of view you want to see statististics and reports for sla compliance and resource planning forecasts
- From the support team point of view you want to know which system is affected, if one component of your services is not operating and how to find the root cause
- From developer point of view you want to get data about application errors including the steps before the error occurs and how to tune performance

A monitoring process designer is always torn between all parties. Too much monitoring may cause more resource consuming than the monitored business service will take. Are the monitored data not sufficient (as always), it may result in serious trouble with one of your stakeholders mentioned above. For this reason I only want to show the basic steps and it's up to you to refine this for your particular needs.

Important: Samples below are neither "official" nor "recommended" but simply out of my implementations. They should give you the feeling what options are available and which steps my be taken to fit your own needs. Samples are additionally attached to this page.

make decisions

- you want to monitor your Oracle instance local or remote
- which parameters/performance data you want to retrieve
- which plugins you want to use
- do you want to create service dependencies between Oracle and host and application
- do you want to be notified who are the recipients for database related notifications
- are all database checks equal apart from the instance names or do you want to define separate service checks with different parameters
- how long your service check data should remain in database. Do you need each and every check results or are RRDs sufficient

sample 1:basic monitoring of a local instance

Following one simple implementation of monitoring an oracle instance.
Decisions

- checking is on local host

- host monitoring is done via standard nagios plugins
- check_oracle_health for alive recovery area free space, connect time and redo i/o
- check_logfile for local monitoring of Oracle alert log
- Oracle instance will be defined as host and is dependent from the machine on which it is living
- there will be a new contact group dba which should be notified
- I want to prevent future keyboard rubbing and define a standard set of services which will be inherited for this instance
- standard cleanup processed will we used
- I will include pnp4nagios to keep history and trends, not within the database

prepare database

We need a new account on database with special rights. This should not be the same we are using for ido or web, because Icinga object maintenance scripts will not expect other objects in this schema and drop them. We will rollout this new user on all monitored instances. Attached is a sample script create_orachecks_sys.sql (syntax is for Oracle V10+)

prepare check_oracle_health usage

Check_oracle_health plugin I mentioned above is designed for using Nagios, not Icinga. In particular this means, this plugin cannot use variables exported by Icinga. Therefore we need a create a wrapper script, which will copy some ICINGA_* macros into corresponding NAGIOS_* macros. For this reason i modified include/macros.h to expose ICINGA_* vars as NAGIOAS_. See Icinga and Oracle Part2 - Building software.

For security reasons I will handover variables instead of commandline parameters, especially SID and password. We will define the SID as host, which allows to take use of HOSTNAME macro. Additional we are preparing recognition of custom variables _SERVICEORACLE_SID, SERVICEORACLE_USER and _SERVICEORACLE_PASS, which we can use later in our service definition. Instead of hardcoding the password here you should run your own 'get_password' routine instead, which should get the right password from an encrypted password store. Additionally we need to set our Oracle Environment to get this check working. Here is my version check_oracle_health.sh i placed in the same directory (libexec) in which we installed check_oracle_health. **ADAPT IT TO YOUR ENVIRONMENT!**
Current plugin version and full set of documentation you can find on their home page http://labs.consol.de/lang/de/nagios/check_oracle_health/

```
#!/bin/bash
PARAM="$@"
. /etc/profile >/dev/null
#set -x
WD=$(dirname $0)
DEFUSER=orachecks
ICINGA=/opt/icinga
LOGDIR=$ICINGA/var
PLUGINDIR=$ICINGA/libexec
if [ -z "$NAGIOS__SERVICEORACLE_SID" ]; then
         NAGIOS__SERVICEORACLE_SID=$NAGIOS__HOSTORACLE_SID
fi
if [ -z "$NAGIOS__SERVICEORACLE_SID" ]; then
        echo "UNKNOWN - _ORACLE_SID not set"
        exit 4
fi
if [ -z "$NAGIOS__SERVICEORACLE_USER" ]; then
         NAGIOS__SERVICEORACLE_USER=$NAGIOS__HOSTORACLE_USER
fi
if [ -z "$NAGIOS__SERVICEORACLE_USER" ]; then
         NAGIOS__SERVICEORACLE_USER=$DEFUSER
fi

if [ -z "$NAGIOS__SERVICEORACLE_PASS" ]; then
         NAGIOS__SERVICEORACLE_PASS=$($WD/get_pwd -u $NAGIOS__SERVICEORACLE_USER -d
$NAGIOS__SERVICEORACLE_SID)
fi

if [ -z "$NAGIOS__SERVICEORACLE_PASS" ]; then
         NAGIOS__SERVICEORACLE_PASS=$($WD/get_pwd -u $NAGIOS__SERVICEORACLE_USER -d
'!default')
fi
export NAGIOS__SERVICEORACLE_SID
export NAGIOS__SERVICEORACLE_PASS
export NAGIOS__SERVICEORACLE_USER
#run it
$PLUGINDIR/check_oracle_health $PARAM
exit $?
```

prepare check_logfiles definitions

We need definitions of the logfile formats before we are able to make use of check_logfiles plugin. We will use this plugin to check Oracle alert log locally and remote. Cuurent plugin version and full set of documentation you can find on the home page http://labs.consol.de/lang/de/nagios/check_logfiles/. According to this documentation we need two definitions, one for the local variant, on for the remote one. For remote retrieving we are using previously created database methods, not ssh or nrpe. Therefore we need to define database connect into the config. This is done via via custom variables. A modified version of the original definition I found on http://labs.consol.de/nagios/check_logfiles/check_logfiles-beispielecheck_logfiles-examples/ ("Beispiel 17") is attached. I will save these file in <icinga_dir>/etc/check_logfiles.

- local version alertlog.cfg

```
@searches = ({
  tag => 'oraalerts',
  logfile =>'$CLM_LOGFILE$',
  criticalpatterns => [
      'ORA\-0*204[^\d]',        # error in reading control file
...
```

- sql based version dbalertlog.cfg

```
@searches = ({
  tag => 'dboraalerts',
  type => 'oraclealertlog',
  oraclealertlog => {
    connect => '$CLM_INST$',    # connect identifier
    username => '$CLM_USER$',   # database user
    password => '$CLM_PASS$',   # database password
  },
  criticalpatterns => [
      'ORA\-0*204[^\d]',        # error in reading control file
...
```

create icinga templates

The following steps are not much database specific. Consult the Icinga documentation for details. We are creating commands, contact groups, host and service templates. We are defining an instance as host, therefore we need a special "alive" check I called oracle-ping, because standard "ping" method will not check databases. For this we are using method "tnsping" offered by check_oracle_health. This method will use the tnsping binary if available, or direct socket connect if not (e.g. because of instant client usage). Service checks include pnp4nagios services.

- create and add new include directories to icinga.cfg

```
cfg_dir=/opt/icinga/etc/templates
cfg_dir=/opt/icinga/etc/servers
cfg_dir=/opt/icinga/etc/dbs
```

- create new command definitions for check_oracle_health: check_oracle_health.cfg

```
#templates/check_oracle_health.cfg
define command{
          command_name check_oracle_health
          command_line $USER1$/check_oracle_health.sh --mode $ARG1$ $ARG2
}
define command{
          command_name oracle-ping
          command_line $USER1$/check_oracle_health.sh --mode tnsping
}
```

- create new command definitions for check_logfiles: check_logfiles.cfg

```
#templates/check_logfiles.cfg
define command {
  command_name         check_local_log
  command_line         $USER1$/check_logfiles --config
$USER1$/../etc/check_logfiles/$ARG1$ --macro "CLM_LOGFILE=$ARG2$"
}
define command {
  command_name         check_db_alertlog
  command_line         $USER1$/check_logfiles --config
$USER1$/../etc/check_logfiles/dbalertlog --macro "CLM_INST=$HOSTNAME$" --macro
"CLM_USER=orachecks" --macro "CLM_PASS=icinga"

}
define command {
  command_name         check_nrpe_arg
  command_line         $USER1$/check_nrpe -H $HOSTADDRESS$ -t $ARG1$ -c $ARG2$ -a
$ARG3$
}

#nrpe.cfg:[check_logfiles]=/opt/nagios/libexec/check_logfiles --config $ARG1$
```

- add pnp4nagios service definitions (if not already done):pnp4nagios.cfg

```
#templates/pnp4nagios.cfg
define host {
   name       host-pnp
   action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=_HOST_
   register   0
}

define service {
   name       srv-pnp
   action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=$SERVICEDESC$
   register   0
}
```

- create contactgroup, hostgroup, host and service templates: oracle_template.cfg

```
#templates/oracle_templates.cfg
# oracle dba contact group
define contactgroup{
        contactgroup_name       dbas
        alias                   Oracle Administrators
        members                 icingaadmin
        }
define hostgroup {
 hostgroup_name  oracle-instances
 alias   Oracle Instances
}

# Oracle Instance definition template - This is NOT a real host, just a template!
```

```
define host{
        name                            oracle-instance    ; The name of this
host template
        hostgroups                      oracle-instances
        use                             generic-host    ; This template inherits
other values from the generic-host template
        check_period                    24x7            ; By default, Oracle
Instances are checked round the clock
        check_interval                  5               ; Actively check the host
every 5 minutes
        retry_interval                  1               ; Schedule host check
retries at 1 minute intervals
        max_check_attempts              3               ; Check each Instance
times (max)
        check_command                   oracle-ping ; Default command to check
oracle instance        notification_period             24x7       ;
                                                        ; Note that the
notification_period variable is being overridden from
                                                        ; the value that is
inherited from the generic-host template!
        notification_interval           120             ; Resend notifications
every 2 hours
        notification_options            d,u,r           ; Only send notifications
for specific host states
        contact_groups                  dbas          ; Notifications get sent to
the admins by default
        register                        0               ; DONT REGISTER THIS
DEFINITION - ITS NOT A REAL HOST, JUST A TEMPLATE!
        _ORACLE_USER                    orachecks       ;default user to connect

}

define service{
           use                  generic-service,srv-pnp
    name   orasrv-tnsping
           service_description  Oracle-TNSPing
           check_command        check_oracle_health!tnsping
    servicegroups oracle_services
           max_check_attempts   1
    register  0

}
define service{
           use                  generic-service,srv-pnp
    name   orasrv-connect_time
           service_description  Oracle Connect Time
           check_command        check_oracle_health!connection-time
    servicegroups oracle_services
           max_check_attempts   1
    register  0


}
define service{
           use                  generic-service,srv-pnp
    name   orasrv-connected_users
           service_description  Oracle Connected Users
           check_command        check_oracle_health!connected-users
```

```
    servicegroups oracle_services
           max_check_attempts   1
    register  0
}
define service{
           use                  generic-service,srv-pnp
    name   orasrv-fb_used
           service_description  Oracle Flashback Area used
           check_command        check_oracle_health!flash-recovery-area-usage
    servicegroups oracle_services
           max_check_attempts   1
    register  0
}
define service{
           use                  generic-service,srv-pnp
    name   orasrv-fb_free
           service_description  Oracle Flashback Area free
           check_command        check_oracle_health!flash-recovery-area-free
    servicegroups oracle_services
           max_check_attempts   1
    register  0
}
define service{
           use                  generic-service,srv-pnp
    name   orasrv-tbs_used
           service_description  Oracle Tablespace used
           check_command        check_oracle_health!tablespace-usage
    servicegroups oracle_services
           max_check_attempts   1
    register  0
}
define service{
           use                  generic-service,srv-pnp
    name   orasrv-tbs_free
           service_description  Oracle Tablespace free
           check_command        check_oracle_health!tablespace-free
    servicegroups oracle_services
           max_check_attempts   1
    register  0
}
define service{
           use                  generic-service,srv-pnp
    name   orasrv-tbs_time
           service_description  Oracle Tablespace Remaining Time
           check_command        check_oracle_health!tablespace-remaining-time
    servicegroups oracle_services
           max_check_attempts   1
    register  0
}

define service{
           use                  generic-service,srv-pnp
    name   orasrv-redo
           service_description  Oracle redo IO
           check_command        check_oracle_health!redo-io-traffic
    servicegroups oracle_services
           max_check_attempts   1
    register  0
```

```
}
```

create definition for a monitored instance

- host definitions
  If not already done, create a standard host definition for the database machine: localhost.cfg. **ADAPT IT TO YOUR NEEDS !**

```
#servers/localhost.cfg
define host{
        use                     linux-server            ; Name of host template
to use
       ; This host definition will inherit all variables that are defined
       ; in (or inherited by) the linux-server host template definition.
        host_name               localhost
        alias                   localhost
        address                 127.0.0.1
        }

define service{
        use                             local-service,srv-pnp         ; Name of
service template to use
        host_name                       localhost
        service_description             PING
 check_command   check_ping!100.0,20%!500.0,60%
        }define service{
        use                             local-service,srv-pnp         ; Name of
service template to use
        host_name                       localhost
        service_description             Root Partition
 check_command   check_local_disk!20%!10%!/
        }
define service{
        use                             local-service,srv-pnp         ; Name of
service template to use
        host_name                       localhost
        service_description             sample Oracle Data Partition
 check_command   check_local_disk!20%!10%!/usr/lib/oracle/xe/oradata/XE        }

define service{
        use                             local-service,srv-pnp         ; Name of
service template to use
        host_name                       localhost
        service_description             Total Processes
 check_command   check_local_procs!250!400!RSZDT
        }
define service{
        use                             local-service,srv-pnp         ; Name of
service template to use
        host_name                       localhost
        service_description             Current Load
 check_command   check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
        }
define service{
        use                             local-service,srv-pnp         ; Name of
service template to use
        host_name                       localhost
        service_description             Swap Usage
 check_command   check_local_swap!20!10
        }
```

- Oracle definitions
  As stated above we will check connect time, alert_log locally, recovery area and as sample performance indicator redo io. Instance is

dependent of the host on which it runs, so I create a host dependency for this. Of course, you have to make sure, local log file locations are readable for the Icinga user. File:xe.cfg. **ADAPT IT TO YOUR NEEDS !**

```
#dbs/xe.cfg
define host{
 use  oracle-instance  ; Inherit default values from a template
 host_name xe               ; The name we're giving to this server
 alias  xe test instance ; A longer name for the server
}
define hostdependency{
 host_name   localhost
 dependent_host_name  xe
 notification_failure_criteria d,u
 execution_failure_criteria d,u
}

define service{
        use                             orasrv-connect_time ; Name of service
template to use
        host_name                       xe
        }

define service{
        use                             orasrv-redo         ; Name of service
template to use
        host_name                       xe
        }

define service{
        use                             orasrv-fb_free      ; Name of service
template to use
        host_name                       xe
        }
define service{
        use                 generic-service
        host_name           xe
        service_description  Oracle ALERT Log
        check_command
check_local_log!alertlog!/usr/lib/oracle/xe/app/oracle/admin/XE/bdump/alert_XE.lo
g
        servicegroups       oracle_services
        max_check_attempts   1
}
```

Now you should restart your Icinga service and if all configurations are OK, you will see your Oracle instance being monitored.

Will be continued ...