13 Must Have Shell Scripts

To Automate Routine Tasks

Asfaw Gedamu

13/03/2024

Introduction

This document provides a collection of shell scripts for monitoring various aspects of an Oracle database server. These scripts can be configured to run periodically using crontab and send email alerts in case of any issues.

Scripts

1. **GoldenGate Monitor (gg_alert.sh)**
    - Monitors the status of GoldenGate extract and replicate processes.
    - Sends email alerts if a process is down.
2. **Standby Database Lag Monitor (dgmgrl_standby_lag.sh)**
    - Uses dgmgrl to check the apply lag in a standby database.
    - Sends email alerts if the lag exceeds a threshold.
3. **RMAN Archive Deletion Script (rman_arch_del.sh)**
    - Uses RMAN to automatically delete archive logs based on a retention policy.
4. **Blocking Session Monitor (blocker.sh)**
    - Queries v$session to identify blocking sessions.
    - Sends email alerts with details of blocking sessions.
5. **ASM Disk Group Usage Monitor (asm_dg.sh)**
    - Monitors ASM disk group utilization.
    - Sends email alerts if usage exceeds a threshold.
6. **Invalid Login Attempt Monitor (invalid_log.sh)**
    - Audits failed login attempts in the database.
    - Sends email alerts for suspicious login activity.
7. **Filesystem Alert Script**
    - Monitors filesystem usage (for Solaris).
    - Sends email alerts if usage exceeds a threshold.
8. **Oracle Alert Log Rotation Script (rotatealertlog.sh)**
    - Rotates the Oracle alert log file and compresses the old log.
9. **Tablespace Usage Monitor (tablespace_threshold.ksh)**
    - Monitors tablespace usage and sends email alerts if it exceeds a threshold.

10. **Alert Log Monitor (Adrci_alert_log.ksh)**

- Uses adrci to monitor Oracle alert logs for ORA errors.
- Sends email alerts if ORA errors are found.

11. **IP Address Tracking Script**

- Tracks IP addresses associated with a load-balanced HTTP link.
- Sends email alerts if the IP addresses change.

12. **RMAN Backup Script (rman_backup.sh)**

- Performs Oracle database backups (full, incremental, archive, cold) using RMAN.
- Supports compressing backups and running parallel backup jobs.

13. Import And Export In Parallel With Datapump

1.Monitor goldengate process

The following script is used to monitor goldengate processes like extract and replicat. And in case extract or replicat is down, it will send alert to the respective email ids.

SCRIPT PREPARATION:

First create a shell script file and name it gg_alert.sh

Then, give it the necessary priviledges. View your file using:

```
cat gg_alert.sh
```

```
#!/bin/bash
EMAIL_LIST="support@dbaclass.com"

OIFS=$IFS
IFS="
"
NIFS=$IFS
```

```
function status {
OUTPUT=`$GG_HOME/ggsci << EOF
info all
exit
EOF`
}
function alert {
for line in $OUTPUT
do
if [[ $(echo "${line}"|egrep 'STOP|ABEND' >/dev/null;echo $?) =
0 ]]
then
GNAME=$(echo "${line}" | awk -F" " '{print $3}')
GSTAT=$(echo "${line}" | awk -F" " '{print $2}')
GTYPE=$(echo "${line}" | awk -F" " '{print $1}')
case $GTYPE in
"MANAGER")
cat $GG_HOME/dirrpt/MGR.rpt | mailx -s "${HOSTNAME} - GoldenGate
${GTYPE} ${GSTAT}" $NOTIFY ;;

"EXTRACT"|"REPLICAT")
cat $GG_HOME/dirrpt/"${GNAME}".rpt |mailx -s "${HOSTNAME} -
GoldenGate ${GTYPE} ${GNAME} ${GSTAT}" $EMAIL_LIST ;;
esac
fi
done
}

export GG_HOME=/goldengate/install/software/gghome_1
export ORACLE_HOME=/oracle/app/oracle/product/12.1.0/db_1
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
status
alert
```

Finally configure the script in crontab with 30 min interval.

```
00,30 * * * * /home/goldengate/gg_alert.sh >
/home/goldengate/gg_alerts.log
```

2.Monitor lag in standby datbase using dgmgrl

Below script is helpful in monitoring lag in standby database and send mail to DBAs in case the lag is increasing. For the script to work, make sure dataguard broker is enabled between primary and standby database.

SCRIPT PREPARATION:

PRIMARY DB UNIQUE_NAME – > PRIMDB
STANDBY DB UNIQUE_NAME -> STYDB

```
cat /home/oracle/dgmgrl_standby_lag.sh
```

```
#!/bin/bash
export ORACLE_HOME=/oracle/app/oracle/product/12.1.0/dbhome_1
export ORACLE_SID=primdb
export PATH=$ORACLE_HOME/bin:$PATH
echo -e "show database stydb"|${ORACLE_HOME}/bin/dgmgrl
sys/orcl1234 > DB_DG_DATABASE.log
cat /home/oracle/DB_DG_DATABASE.log  | grep "Apply Lag"  >
FILTERED_DB_DG_DATABASE.log
time_value=`cut -d " " -f 14 FILTERED_DB_DG_DATABASE.log`
time_param=`cut -d " " -f 15 FILTERED_DB_DG_DATABASE.log`
if [[ "$time_param" == "minutes" && "$time_value" -ge 1 ]]
then
mailx -s "DREAIDB LAG is in minutes
"  suppor@dbaclass.com<DB_DG_DATABASE.log
else
if [[ "$time_param" == "seconds" && "$time_value" -ge 30 ]]
then
mailx -s "DREAIDB LAG is in seconds
"   support@dbaclass.com<DB_DG_DATABASE.log
else
if  [[ "$time_param" == "hour(s)" && "$time_value" -ge 1 ]]
then
mailx -s "DREAIDB LAG is in hours "   support@dbaclass.com
<DB_DG_DATABASE.log
fi
```

```
fi
fi
```

Now configure the the script in crontab

```
00,10,20,30,40,50 * * * * /home/oracle/dgmgrl_standby_lag.sh >
/tmp/dg_lag.log
```

3. Delete old archives using RMAN

If the requirement is to delete archive log backups automatically (without taking backup), then below shell script can be configured in crontab.

prepare the shell script.

```
cat rman_arch_del.sh
```

```
#!/bin/bash
export ORACLE_HOME=/oracle/app/oracle/product/12.1.0.2.0
export ORACLE_SID=PARIS12C
export PATH=$ORACLE_HOME/bin:$PATH
delBackup () {
rman log=/home/oracle/arch_del.log << EOF
connect target /
DELETE noprompt ARCHIVELOG ALL COMPLETED BEFORE 'sysdate-1';
CROSSCHECK ARCHIVELOG ALL;
DELETE EXPIRED ARCHIVELOG ALL;
exit
EOF
}
# Main
delBackup
```

Now configure in crontab:

```
00 22 * * * /u01/app/oracle/rman_arch_del.sh > /tmp/rmanarch.log
```

4. Monitoring blocking sessions

Below is the shell script, to be configured in crontab, which will send mail incase of blocking session observed in the database .
In the mail body it will contain the blocking sessions details also.

1. Prepare the blocker.sql file.[for blocking sessions more than 10 seconds)

```
set feed off
set pagesize 200
set lines 299
col event for a31
SELECT
s.inst_id,
s.blocking_session,
s.sid,
s.serial#,
s.seconds_in_wait,
s.event
FROM
gv$session s
WHERE
blocking_session IS NOT NULL and s.seconds_in_wait > 10;
```

2. Shell script.(/home/oracle/monitor/blocker.sh)

You need to define the ORACLE_HOME,ORACLE_SID respectively.

```
export ORACLE_HOME=/oracle/app/oracle/product/12.1.0/dbhome_1
export ORACLE_SID=ORCL
export PATH=$ORACLE_HOME/bin:$PATH
logfile=/home/oracle/monitor/block_alert.log
sqlplus -s "/as sysdba" > /dev/null << EOF
spool $logfile
@/home/oracle/monitor/blocker.sql
spool off
exit
EOF
count=`cat $logfile|wc -l`
if [ $count -ge 1 ];
 then mailx -s "BLOCKING SESSION REPORTED IN PROD DB ( > 10 SEC)
```

```
" support@dbaclass.com < $logfile
fi
```

3. configure in crontab(every one minute)

```
* * * * * /home/oracle/monitor/blocker.sh > /tmp/block.lo
```

5.Monitor asm diskgroup usage

The following is a shell script that will trigger a mail alert, if the utilization of the asm diskgroup reached 90 percent.

1. Below is the shell script.

Make sure to update ORACLE_HOME, ORACLE_SID inside the shell script.

```
cat /export/home/oracle/asm_dg.sh
```

```
export ORACLE_HOME=/oracle/app/oracle/product/12.1.0.2/dbhome_1
export ORACLE_SID=PRODDB1
export PATH=$ORACLE_HOME/bin:$PATH
logfile=/export/home/oracle/asm_dg.log
sqlplus -s "/as sysdba" > /dev/null << EOF spool $logfile
SET LINESIZE 150
SET PAGESIZE 9999
SET VERIFY off
COLUMN group_name
FORMAT a25 HEAD 'DISKGROUP_NAME'
COLUMN state FORMAT a11 HEAD 'STATE'
COLUMN type FORMAT a6 HEAD 'TYPE'
COLUMN total_mb FORMAT 999,999,999 HEAD 'TOTAL SIZE(GB)'
COLUMN free_mb FORMAT 999,999,999 HEAD 'FREE SIZE (GB)'
COLUMN used_mb FORMAT 999,999,999 HEAD 'USED SIZE (GB)'
COLUMN pct_used FORMAT 999.99 HEAD 'PERCENTAGE USED'

SELECT distinct name group_name , state state , type type ,
round(total_mb/1024) TOTAL_GB , round(free_mb/1024) free_gb ,
```

```
round((total_mb - free_mb) / 1024) used_gb ,
round((1- (free_mb / total_mb))*100, 2) pct_used from
v$asm_diskgroup where round((1- (free_mb / total_mb))*100, 2) >
90 ORDER BY name;
spool off
exit
EOF
count=`cat $logfile|wc -l`
#echo $count
if [ $count  -ge 4 ];
 then
  mailx -s "ASM DISKGROUP REACHED 90% UTILIZATION"
support@dbaclass.com < $logfile
fi
```

2. Give proper permission:

```
chmod 755 /export/home/oracle/asm_dg.sh
```

3. Configure in crontab:

```
0,15,30,45 * * * * /export/home/oracle/asm_dg.sh
```

6. To report failed login attempt in oracle

Configure a shell script in crontab, that will send alert to DB support Team in case of any invalid login attempts in the database.

1. First, enable audit for create session

```
SQL> audit create session;

Audit succeeded.
```

2. Final shell script

Below script for any invalid login attempts in last 15 minutes.

```
cat /export/home/oracle/invalid_log.sh
```

```
export ORACLE_HOME=/oracle/app/oracle/product/12.1.0/dbhome_1
export ORACLE_SID=SBIP18DB
export PATH=$ORACLE_HOME/bin:$PATH
logfile=/export/home/oracle/test.log
sqlplus -s "/as sysdba" > /dev/null << EOF
spool $logfile
set pagesize 1299
set lines 299
col username for a15
col userhost for a13
col timestamp for a39
col terminal for a23
SELECT username,userhost,terminal,to_char(timestamp,'DD/MM/YY
HH24:MI:SS' ) "TIMESTAMP" ,
CASE
when returncode=1017 then 'INVALID-attempt'
when returncode=28000 then 'account locked'
end "FAILED LOGIN ACTION"
FROM dba_audit_session where timestamp > sysdate-1/9and
returncode in (1017,28000);
spool off
exit
EOF
count=`cat $logfile|wc -l`
#echo $count
if [ $count  -ge 4 ];
 then
  mailx -s "INVALID ATTEMPS IN DB  " support@dbaclass.com <
$logfile
fi
```

3. provide proper permission:

```
chmod 755 invalid_log.sh
```

4. Configure in crontab:

```
0,15,30,45 * * * * /export/home/oracle/invalid_log.sh
```

7.A script for file system alert

Below is script to notification when a mount point or filesystem usage crosses a threshold value.

For solaris

```
#!/bin/sh

df -h | egrep -v '/system|/platform|/dev|/etc|lib' | awk '{print
$6 " " $5}'|cut -d% -f1|while read fs val

do

if [ $val -ge 90 ]
then
echo "The $fs usage high $val% \n \n \n `df -h $fs`" | mailx -s
"Filesystem $fs Usage high on Server `hostname`"
support@dbaclass.com

fi
done
```

Put in crontab:

```
00 * * * * /usr/local/scripts/diskalert.sh
```

For monitoring zpool usage in solaris:

```
zpool list | awk '{print $5}'| grep -v CAP | cut -d% -f1| while
read val

do

if [ $val -ge 80 ]
then
echo "The $fs usage high $val% \n \n \n `df -h $fs`" | mailx -s
"Filesystem $fs Usage high on Server `hostname`"
rpatro.c@stc.com.a
```

```
fi
done
```

Put in crontab as below:

```
00 * * * * /usr/local/scripts/zpoolusage.sh
```

8. Alert log rotation script in oracle

Alert log size will grow in Oracle database from day to day. So for housekeeping, we need to move the existing alert log to a backup location and compress there. Upon moving the alert log, the database will create a fresh alert log automatically.

1. **Below is the shell script.**

We need to define the **ORACLE_HOME** in the script. and **ORACLE_SID** will be passed as an argument while running the script.

```
# $Header: rotatealertlog.sh
```

```
#!/bin/bash
echo ========================
echo Set Oracle Database Env
echo ========================

ORACLE_SID=$1; export ORACLE_SID
ORACLE_HOME=/oracle/app/oracle/product/12.1.0.2/dbhome_1
ORACLE_BASE=/oracle/app/oracle; export ORACLE_BASE
LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/lib; export
LD_LIBRARY_PATH
PATH=$ORACLE_HOME/bin:$PATH;export PATH
TO_DATE="20`date +%y%m%d`"; export TO_DATE

echo =======
echo Extract Alert log location
echo =======
export VAL_DUMP=$(${ORACLE_HOME}/bin/sqlplus -S /nolog <<EOF
```

```
conn /as sysdba
set pages 0 feedback off;
prompt
SELECT value from v\$parameter where NAME='core_dump_dest';
exit;
EOF
)
export LOCATION=`echo ${VAL_DUMP} | perl -lpe'$_ = reverse' |awk
'{print $1}'|perl -lpe'$_ = reverse'`
export ALERTDB=${LOCATION}/alert_$ORACLE_SID.log
export ELOG=$( echo ${ALERTDB} | sed s/cdump/trace/)

echo =======
echo Compress current
echo =======

if [ -e "$ELOG" ] ; then
 mv ${ELOG} ${ELOG}_${TO_DATE};
 gzip ${ELOG}_${TO_DATE};
 > ${ELOG}
else
 echo not found
fi

exit
```

2. **Configure in crontab:**

SCHEDULE – Weekly once

Here, we have passed the **ORACLE_SID** (PRODDB) as **argument**

```
00 22 * * 5 /u01/app/oracle/dbscripts/rotatealertlog.sh PRODDB
```

9.Monitoring Tablespace

Below script can be configured in crontab to send a notification to the support DBAs in case tablespace usage crosses a threshold.

1. First, make the below .sql file, which will be used inside the shell script.

In this script we have defined the threshold as 90%. You can change it as per your requirement.

```
cat
/export/home/oracle/Housekeeping/scripts/tablespace_alert.sql
```

```
set feedback off
set pagesize 70;
set linesize 2000
set head on
COLUMN Tablespace        format a25 heading 'Tablespace Name'
COLUMN autoextensible         format a11              heading
'AutoExtend'
COLUMN files_in_tablespace    format 999              heading
'Files'
COLUMN total_tablespace_space format 99999999 heading
'TotalSpace'
COLUMN total_used_space       format 99999999 heading
'UsedSpace'
COLUMN total_tablespace_free_space format 99999999 heading
'FreeSpace'
COLUMN total_used_pct              format 9999      heading
'%Used'
COLUMN total_free_pct              format 9999     heading
'%Free'
COLUMN max_size_of_tablespace      format 99999999 heading
'ExtendUpto'
COLUM total_auto_used_pct         format 999.99      heading
'Max%Used'
COLUMN total_auto_free_pct         format 999.99      heading
'Max%Free'
WITH tbs_auto AS
     (SELECT DISTINCT tablespace_name, autoextensible
                FROM dba_data_files
               WHERE autoextensible = 'YES'),
     files AS
     (SELECT   tablespace_name, COUNT (*) tbs_files,
               SUM (BYTES/1024/1024) total_tbs_bytes
          FROM dba_data_files
      GROUP BY tablespace_name),
     fragments AS
     (SELECT   tablespace_name, COUNT (*) tbs_fragments,
               SUM (BYTES)/1024/1024 total_tbs_free_bytes,
```

```
               MAX (BYTES)/1024/1024 max_free_chunk_bytes
          FROM dba_free_space
      GROUP BY tablespace_name),
     AUTOEXTEND AS
     (SELECT   tablespace_name, SUM (size_to_grow)
total_growth_tbs
          FROM (SELECT   tablespace_name, SUM
(maxbytes)/1024/1024 size_to_grow
                  FROM dba_data_files
                 WHERE autoextensible = 'YES'
              GROUP BY tablespace_name
              UNION
              SELECT   tablespace_name, SUM (BYTES)/1024/1024
size_to_grow
                  FROM dba_data_files
                 WHERE autoextensible = 'NO'
              GROUP BY tablespace_name)
      GROUP BY tablespace_name)
SELECT c.instance_name,a.tablespace_name Tablespace,
       CASE tbs_auto.autoextensible
          WHEN 'YES'
             THEN 'YES'
          ELSE 'NO'
       END AS autoextensible,
       files.tbs_files files_in_tablespace,
       files.total_tbs_bytes total_tablespace_space,
       (files.total_tbs_bytes - fragments.total_tbs_free_bytes
       ) total_used_space,
       fragments.total_tbs_free_bytes
total_tablespace_free_space,
       round((  (  (files.total_tbs_bytes -
fragments.total_tbs_free_bytes)
           / files.total_tbs_bytes
          )
        * 100
       )) total_used_pct,
       round(((fragments.total_tbs_free_bytes /
files.total_tbs_bytes) * 100
       )) total_free_pct
  FROM dba_tablespaces a,v$instance c , files, fragments,
AUTOEXTEND, tbs_auto
WHERE a.tablespace_name = files.tablespace_name
   AND a.tablespace_name = fragments.tablespace_name
   AND a.tablespace_name = AUTOEXTEND.tablespace_name
```

```
   AND a.tablespace_name = tbs_auto.tablespace_name(+)
and (((files.total_tbs_bytes - fragments.total_tbs_free_bytes)/
files.total_tbs_bytes))* 100 > 90
order by total_free_pct;
```

2. Now prepare the shell script:

At the beginning of the script, we need to define the env variables like ORACLE_HOME, PATCH, LD_LIBRARY_PATH, ORACLE_SID.

Below is the final script(tablespace_threshold.ksh)

```
cat
/export/home/oracle/Housekeeping/scripts/tablespace_threshold.ks
h
```

```
#!/bin/sh
export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
export PATH=$ORACLE_HOME/bin:$PATH
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
export ORACLE_SID=PRODDB
cd /export/home/oracle/Housekeeping/scripts
logfile=/export/home/oracle/Housekeeping/scripts/Tablespace_aler
t.log
cnt1=`ps -ef|grep pmon|grep $ORACLE_SID|wc -l`
if [ $cnt1 -eq 1 ];
then
sqlplus -s "/as sysdba" > /dev/null << EOF
spool $logfile
@/export/home/oracle/Housekeeping/scripts/tablespace_alert.sql
spool off
exit
EOF
# If there are more then these two lines in the output file,
mail it.
count=`cat $logfile|wc -l`
#echo $count
if [ $count  -ge 4 ];
 then
  mailx -s "TABLESPACE ALERT FOR PROD DB  " support@dbaclass.com
```

```
<$logfile
fi
fi
```

3. Now configure in crontab:

```
0,15,30,45 * * * *
/export/home/oracle/Housekeeping/scripts/tablespace_threshold.ks
h > /export/home/oracle/Housekeeping/logs/ts_alert.log  2>&1
```

10.A script for monitoring Alert log

Configure a shell script to monitor alert log for all the databases on a server once in every 15 min.And in the case of any ORA- error mail to the DBA TEAM.

Below script is prepared using the ADRCI utility of oracle 11g. It will monitor alert log for all the databases having same oracle base.

SCRIPT:(Adrci_alert_log.ksh)

```
LOG_DIR=/export/home/oracle/Housekeeping/logs/alert_log_check_da
ily.txt
adrci_homes=( $(adrci exec="show homes" | egrep -e rdbms ))
echo '################################' > $LOG_DIR
echo '############################ALERT LOG OUTPUT FOR LAST 15
MINUTES ############################' >> $LOG_DIR
echo '################################' >> $LOG_DIR

for adrci_home in ${adrci_homes[@]}
do

echo ' '>>$LOG_DIR
echo '################################################' >>
$LOG_DIR
echo '########################################' >> $LOG_DIR
echo ' '>>$LOG_DIR
echo $adrci_home' Alert Log' >> $LOG_DIR
adrci exec="set home ${adrci_home}; show alert -p
\\\"message_text like '%ORA-%' and originating_timestamp >
```

```
systimestamp-1/96\\\"" -term >> $LOG_DIR

done
num_errors=`grep  -c 'ORA' $LOG_DIR`
if [ $num_errors != 0 ]
then

mailx -s "ORA- error found in alert Log of the server  "
support@dbaclass.com <$LOG_DIR

fi
```

Give 755 permission to the script

```
chmod 755  Adrci_alert_log.ksh
```

Configure the script in crontab:

```
0,15,30,45 * * * *
/export/home/oracle/Housekeeping/scripts/Adrci_alert_log.ksh >
/export/home/oracle/Housekeeping/logs/error_alert.log 2>&1
```

11. Shell Script To Track IP Address for HTTP Links automatically

```
#!/bin/bash

# Define the URL of the load-balanced HTTP link
url="<PUT URL, example: google.com"

# Define the filename where the current IP addresses are stored
filename="current_ips.txt"
newipfilename="new_ips.txt"
maillist="<PUT YOUR EMAIL>"

# Retrieve the current IP addresses of the URL
current_ips=$(nslookup $url | grep Address | awk '{print $2}')
echo "PRINT CURRENT IP"
echo $current_ips
```

```
echo ""
# Read the saved IP addresses from the file
echo "PRINT SAVED IPS"
saved_ips=$(cat $filename)
echo $saved_ips
echo ""

# Split the current IP addresses into an array
current_ips_array=($(echo "$current_ips" | awk '{print $1}'))

# Split the saved IP addresses into an array
saved_ips_array=($(echo "$saved_ips"))

# Flag to check if any new IP addresses are found
new_ips_found=0
rm -rf $newipfilename

# Loop through each element in array1
for current_ip in "${current_ips_array[@]}"; do

# Check if the element is not in array2
if ! [[ "${saved_ips_array[@]}" =~ "${current_ip}" ]]; then
echo ${current_ip}>>$newipfilename
# Increase the counter
new_ips_found=$((new_ips_found+1))
fi
done
echo "number of new IP"
echo $new_ips_found
new_ips=$(cat new_ips.txt)


# Send an email if IP changed

if [[ "$new_ips_found" -gt 0 ]]; then

# Remove the comment below to update the file with the new IP
addresses if you want.
# echo "$current_ips" > $filename

# Send an email notification
echo -e "Below $new_ips_found IP addresses(s) of $url have
changed:\n $new_ips \n \n Current IP addresses for this URLs
are:\n \n $current_ips \n \n " | mail -s "IP addresses of $url
```

```
have changed" $maillist
fi
```

What does the script do?

1. Initialization:

- Sets the url variable to the load-balanced HTTP link you want to monitor.
- Specifies files to store current and new IP addresses (current_ips.txt and new_ips.txt).
- Sets the maillist variable to the email address for notifications.

2. Retrieving Current IPs:

- Uses nslookup $url to query DNS for the IP addresses associated with the URL.
- Filters the output using grep Address and extracts IPs using awk '{print $2}'.
- Stores the current IPs in the current_ips variable.

3. Reading Saved IPs:

- Reads the previously saved IPs from the current_ips.txt file.
- Stores those IPs in the saved_ips variable.

4. Comparing IPs:

- Splits both current_ips and saved_ips into separate arrays for comparison.
- Loops through each IP in the current_ips_array.
- For each IP, checks if it's not present in the saved_ips_array.
- If a new IP is found, adds it to new_ips.txt and increments the new_ips_found counter.

5. Notification:

- If any new IPs are found (new_ips_found is greater than 0):
    - Optionally updates the current_ips.txt file with the new IPs (commented out by default).

- Sends an email notification to the specified maillist with details about the changed IPs and current IPs for the URL.

12. RMAN Backup script

```
#!/bin/bash

usage () {
echo "Usage : SID BACKUP_TYPE COMPRESSION PARALLELISM
        SID : SID, comma separated list of databases or ALL for
all databases (running)
        BACKUP_TYPE : INCR, FULL, COLD or ARCH
        COMPRESS : COMPRESS or NOCOMPRESS to compress or not the
backup
        PARALLEL : defines the number of channel to use

        exemple backup full : rman_backup.sh db1 FULL COMPRESS
16
        exemple backup arch : rman_backup.sh db1 ARCH NOCOMPRESS
2
"
}

##Variables definition
BASEDIR=$(dirname "$0")
BACKUP_BASE=/Data_Domain/oracle/prod/
LOGDIR=${BASEDIR}/log


DEST_EMAIL=example@example.com
export NLS_DATE_FORMAT='dd/mm/yyyy hh24:mi:ss'
DATE=`date +"%Y%m%d_%H%M%S"`
PATH=$PATH:/usr/local/bin

# Create directorires if not exist
mkdir -p $BACKUP_BASE/
mkdir -p $LOGDIR
mkdir -p $BACKUP_BASE/autobackup


# Validating du number of parameters passed
```

```
if [ $# -lt 4 ]; then
    usage
    exit 1
fi

# Parameters provided
DB_LIST=$1
BACKUP_TYPE=$2
PARALLEL=$4

# Backup type validation
case $BACKUP_TYPE in
    FULL)
        LEVEL="incremental level 0"
    ;;
    INCR)
        LEVEL="incremental level 1"
    ;;
    ARCH)
        LEVEL=""
    ;;
    COLD)
        LEVEL=""
    ;;
    *)
        usage
    exit 1
    ;;
esac

# Compression validation
if [ $3 = 'COMPRESS' ]; then
    COMPRESS='AS COMPRESSED BACKUPSET'
else
    if [ $3 = 'NOCOMPRESS' ]; then
        COMPRESS=''
    else
        usage
        exit 1
    fi
fi

##backup function
function backup_database() {
```

```
    # Set Oracle Environment for database
    ORACLE_SID=$1
    ORAENV_ASK=NO
    . oraenv

    OUTPUT_SID=${ORACLE_SID}
    BACKUP_DIR=$BACKUP_BASE/${ORACLE_SID}

LOGFILE=$LOGDIR/rman_backup_${ORACLE_SID}_${BACKUP_TYPE}_${DATE}
.log


    # Controlfile backup directory
    CF_BACKUP="'$BACKUP_DIR/autobackup/cf_sp_%F'"
    FORMAT_DATA="format
'${BACKUP_DIR}/data_%d_${BACKUP_TYPE}_bks%s_%T_%U.bck'"
    FORMAT_ARCHIVE="format
'${BACKUP_DIR}/arch_%d_${BACKUP_TYPE}_bks%s_%T_%U.bck'"

    if [ $BACKUP_TYPE = 'COLD' ]; then
        sqlplus -s / as sysdba <<EOF
            shutdown immediate;
            startup mount;
            exit
EOF
        rman target /  << EOF >> $LOGFILE 2>&1
            CONFIGURE CONTROLFILE AUTOBACKUP ON;
            CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE
TYPE DISK TO ${CF_BACKUP};
            CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO BACKUPSET
PARALLELISM ${PARALLEL};
            run {
                backup ${COMPRESS} database $FORMAT_DATA;
                delete noprompt obsolete;
                }
            exit
EOF

        sqlplus -s / as sysdba <<EOF
            alter database open;
            exit
EOF

    else
```

```
        if [  $BACKUP_TYPE = 'ARCH' ]; then
            rman target /  << EOF >> $LOGFILE
                CONFIGURE CONTROLFILE AUTOBACKUP ON;
                CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR
DEVICE TYPE DISK TO ${CF_BACKUP};
                CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO
BACKUPSET PARALLELISM ${PARALLEL};
                run {
                    backup ${COMPRESS} archivelog all
$FORMAT_ARCHIVE delete input filesperset 10;
                    delete noprompt obsolete;
                    }
                exit
EOF
        else
            rman target /  << EOF >> $LOGFILE 2>&1
                CONFIGURE CONTROLFILE AUTOBACKUP ON;
                CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR
DEVICE TYPE DISK TO ${CF_BACKUP};
                CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO
BACKUPSET PARALLELISM ${PARALLEL};
                run {
                    backup ${COMPRESS} archivelog all
$FORMAT_ARCHIVE delete input filesperset 10;
                    backup ${COMPRESS} ${LEVEL} database
$FORMAT_DATA include current controlfile;
                    backup ${COMPRESS} archivelog all
$FORMAT_ARCHIVE delete input filesperset 10;
                    delete noprompt obsolete;
                    }
                exit
EOF
        fi
    fi

    # Validate Errors in the log.
    ERRORLIST=$(egrep "^RMAN-[0-9]*:|^ORA-[0-9]*:" $LOGFILE)
    ERRORLIST=$(echo $ERRORLIST)
    if [ -n "$ERRORLIST" ]
    then
        SUBJECT="$(date +%y)/$(date +%m)/$(date +%d) $(date
+%H).$(date +%M).$(date +%S) - `hostname` - Backup Report
${OUTPUT_SID} - ERROR"
    else
```

```
        SUBJECT="$(date +%y)/$(date +%m)/$(date +%d) $(date
+%H).$(date +%M).$(date +%S) - `hostname` - Backup Report
${OUTPUT_SID}"
    fi
    cat -v $LOGFILE | mail -s "$SUBJECT" "$DEST_EMAIL"
}

if [ $1 = 'ALL' ]; then
    for database in `ps -ef | grep pmon | egrep -v 'ASM|grep' |
awk '{print $8}' | cut -d_ -f3`
    do
        backup_database $database
    done
else
    for database in $(echo $1 | sed "s/,/ /g")
    do
        backup_database $database
    done
fi
```

This script is an RMAN backup script designed to automate backing up Oracle databases. Here's a breakdown of its functionalities:

1. Setting Up:

- Defines variables for paths, logging, email notification, date format, and adds helper functions to the system path.
- Creates directories for backups, logs, and automatic backups if they don't exist.
- Checks if the number of arguments provided when running the script is correct (should be 4).

2. Processing Arguments:

- Takes four arguments:
    - DB_LIST: Specifies the database name (or "ALL" for all running databases).
    - BACKUP_TYPE: Defines the backup type (FULL, INCR, ARCH, or COLD).

- COMPRESSION: Sets compression for backups (COMPRESS or NOCOMPRESS).
- PARALLELISM: Defines the number of channels to use for parallel execution.

3. Validating Input:

- Validates the backup type and compression options provided.

4. backup_database function:

- Takes a database name as input.
- Sets the Oracle environment for that specific database.
- Defines variables for various aspects of the backup process:
  - Output database name
  - Backup directory for the specific database
  - Log file location
  - Control file backup format and location
  - Format for data and archive backups

5. Backup Logic:

- The logic depends on the provided BACKUP_TYPE:
  - **COLD Backup:** Shuts down, mounts, and opens the database in a controlled manner, performs a full backup with compression, deletes obsolete backups, and then opens the database again.
  - **ARCH Backup:** Backs up archive logs with compression, deletes obsolete backups.
  - **FULL/INCR Backup:** Backs up archive logs with compression, performs the specified level (full or incremental) database backup with compression (including the control file), backs up additional archive logs, and deletes obsolete backups.

6. Error Handling and Notification:

- After the backup process, the script checks the log file for any errors (lines starting with "RMAN-" or "ORA-").
- Based on the presence of errors, an email subject line is constructed with details about the date, hostname, backup report for the database, and success/error status.
- The script then sends the log file content to the designated email address using the constructed subject line.

7. Looping through Databases:

- If DB_LIST is "ALL", the script iterates through all running databases identified using process listing (ps -ef) and extracts the database names.
- Otherwise, it loops through each database name provided in the comma-separated DB_LIST.
- For each database, the backup_database function is called to perform the backup process.

Overall, this script automates RMAN backups for Oracle databases based on user-provided parameters and sends email notifications with log details for success or failure.

13. Import And Export In Parallel With Datapump

A.Export In Parallel With Datapump

```
#!/bin/bash

BASE_SCHEMA=$1
BASE_TABLE=$2
PARALLEL=$3;
PARTITION=$4

function usage(){
  echo "USAGE:
        Parameter 1 is the SCHEMA
        Parameter 2 is the TABLE NAME
        Parameter 3 is the DEGREE of parallelism
```

```
        Parameter 4 (optional) is the partition (if any)"
}

if [ $# -lt 3 ]; then
     usage
     exit 1
fi

if [ $# -eq 4 ]; then
     PARFILE=${BASE_SCHEMA}_${BASE_TABLE}_${PARTITION}.par
     echo "tables=${BASE_SCHEMA}.${BASE_TABLE}:${PARTITION}" >
$PARFILE
     START_MESSAGE="Beginning export of partition :
${BASE_SCHEMA}.${BASE_TABLE}:${PARTITION} "
     END_MESSAGE "Finished export of partition:
${BASE_SCHEMA}.${BASE_TABLE}:${PARTITION}"
     DUMPFILE_BASE=${BASE_SCHEMA}_${BASE_TABLE}_${PARTITION}
    LOGFILE_BASE=${BASE_SCHEMA}_${BASE_TABLE}_${PARTITION}
else
     PARFILE=${BASE_SCHEMA}_${BASE_TABLE}.par
     echo "tables=${BASE_SCHEMA}.${BASE_TABLE}" >  $PARFILE
     START_MESSAGE="# Beginning export of table :
${BASE_SCHEMA}.${BASE_TABLE}"
     END_MESSAGE "# Finished export of table:
${BASE_SCHEMA}.${BASE_TABLE}"
     DUMPFILE_BASE=${BASE_SCHEMA}_${BASE_TABLE}
    LOGFILE_BASE=${BASE_SCHEMA}_${BASE_TABLE}
fi

# Adding parameters to the parfile
echo "directory=DATA_PUMP" >>  $PARFILE
echo "EXCLUDE=STATISTICS" >>  $PARFILE
echo "CLUSTER=N" >>  $PARFILE

echo
"###############################################################"
echo $START_MESSAGE
echo
"###############################################################"
echo " "

LIMIT=$(expr $PARALLEL - 1)

START_TIME=`date`
```

```
for i in `seq 0 $LIMIT`
do
   QUERY="where mod(dbms_rowid.rowid_block_number(rowid),
${PARALLEL}) = $i"
   expdp userid=\'/ as sysdba\'
query=$BASE_SCHEMA.$BASE_TABLE:\"$QUERY\"
dumpfile=${DUMPFILE_BASE}_${i}.dmp
logfile=${LOGFILE_BASE}_${i}.log parfile=$PARFILE &
   sleep 3
done

wait `pidof expdp`

echo
"###############################################################"
echo $END_MESSAGE
echo "# Start time : $START_TIME "
echo "# End time is: `date`"
echo
```

B. Import In Parallel With Datapump

```
#!/bin/bash

export ORAENV_ASK=NO
export ORACLE_SID=$1
. oraenv

TABLE_NAME=$2
PARTITION=$3

function usage(){
  echo "USAGE:
        Parameter 1 is the SID of the database where you want
to import
         Parameter 2 is the TABLE you want to import
         Parameter 3 (optional) is the PARTITION name you want
to import (if any)"
}

if [ $# -lt 2 ]; then
     usage
```

```
     exit 1
fi

if [ $# -eq 3 ]; then
     PARFILE=${TABLE_NAME}_${PARTITION}.par
     START_MESSAGE="Beginning import of partition :
${TABLE_NAME}:${PARTITION} "
     END_MESSAGE "Finished import of partition:
${TABLE_NAME}:${PARTITION}"
     SEARCH_PATTERN=${BASE_TABLE}_${PARTITION}
     SUCCESS_MESSAGE="partition: ${TABLE_NAME}:${PARTITION}
successfully imported, started at"
     ERROR_MESSAGE="partition: ${TABLE_NAME}:${PARTITION} failed
to import, check logfile for more info"
     MAIL_OBJECT="Successfully imported partition
${TABLE_NAME}:${PARTITION}"
else
     PARFILE=${TABLE_NAME}.par
     START_MESSAGE="Beginning import of table : ${TABLE_NAME}"
     END_MESSAGE "Finished import of table : ${TABLE_NAME}"
     SEARCH_PATTERN=${BASE_TABLE}
     SUCCESS_MESSAGE="Table ${TABLE_NAME} successfully imported,
started at "
     ERROR_MESSAGE="Table ${TABLE_NAME} failed to import, check
logfile for more info"
     MAIL_OBJECT="Successfully imported table ${TABLE_NAME}"
fi

#directories
BASEDIR=/u10/
DUMPDIR=$BASEDIR/DUMP
PARFILEDIR=$BASEDIR/parfiles

mkdir -p $PARFILEDIR

# building the parfile
echo "DIRECTORY=MY_DUMP_DIR" >  ${PARFILEDIR}/$PARFILE
echo "CLUSTER=N" >>  ${PARFILEDIR}/$PARFILE
echo "TABLE_EXISTS_ACTION=APPEND" >>  ${PARFILEDIR}/$PARFILE
echo "DATA_OPTIONS=DISABLE_APPEND_HINT" >>
${PARFILEDIR}/$PARFILE


echo
```

```
"###################################################################"
echo $START_MESSAGE
echo
"###################################################################"
echo " "

START_TIME=`date`

for dump in `ls ${DUMPDIR}/*${SEARCH_PATTERN}*.dmp`
do
   DUMPFILE=${dump}
   LOGFILE=imp_${dump}.log
   impdp userid=\'/ as sysdba\' dumpfile=$DUMPFILE
logfile=${LOGFILE} parfile=${PARFILEDIR}/$PARFILE &
   sleep 3
done

wait `pidof impdp`

echo
"###################################################################"
echo $END_MESSAGE
echo "# Start time : $START_TIME "
echo "# End time   : `date`"
echo
"###################################################################"

# Verifying errors
errors_count=`grep ORA- *${SEARCH_PATTERN}*.log | wc -l`

if [ $errors_count -eq 0 ]; then
        echo "$SUCCESS_MESSAGE $START_TIME and finished at
`date`" |  mail -s $MAIL_OBJECT you@your-email.com
else
        echo $ERROR_MESSAGE |  mail -s $MAIL_OBJECT you@your-
email.com
fi
```

Conclusion

These shell scripts can be a valuable tool for automating database monitoring tasks and ensuring the smooth operation of your Oracle environment. By customizing the scripts and configuring crontab, you can receive timely notifications about potential issues and take necessary actions.

Note:

- Replace placeholders like <URL>, <PUT YOUR EMAIL>, etc. with your specific values.
- Adjust threshold values and email recipients as needed.