

Contents

[What is AMChecks? (aka ‘the Story So far’) 2](#_Toc483227081)

[What isn’t it? 2](#_Toc483227082)

[Is it free? 2](#_Toc483227083)

[Pre-Installation 3](#_Toc483227084)

[Installation 4](#_Toc483227085)

[How to Add In Your Databases 7](#_Toc483227086)

[Back to The Testing.. 8](#_Toc483227087)

[Important Tables and Columns 9](#_Toc483227088)

[AM\_SERVER 9](#_Toc483227089)

[AM\_DATABASE 10](#_Toc483227090)

[AM\_SCRIPTS 10](#_Toc483227091)

[AM\_SIDSKIP 11](#_Toc483227092)

[AM\_SCRIPTSKIP 11](#_Toc483227093)

[AM\_OS\_SPACE 11](#_Toc483227094)

[AM\_TABLESPACE\_SPACE 11](#_Toc483227095)

[Some Scripts Explained 14](#_Toc483227096)

[amchecks.ksh 15](#_Toc483227097)

[all\_os\_space\_check.ksh 16](#_Toc483227098)

[tablespace\_space\_check.ksh 17](#_Toc483227099)

[How To… 19](#_Toc483227100)

[Add a new database 19](#_Toc483227101)

[Scheduling a script with parameters 19](#_Toc483227102)

[Changing who gets e-mails from amchecks 19](#_Toc483227103)

[Adding new ‘IND’ columns 20](#_Toc483227104)

[Writing Your Own Scripts 21](#_Toc483227105)

[Thanks 21](#_Toc483227106)

Oracle Morning Checks (AMChecks)

# What is AMChecks? (aka ‘the Story So far’)

AMChecks is a (very) loose collection of scripts – a bunch of korn shell scripts (ksh) and SQL scripts (including anonymous PL/SQL blocks but no actual procedures) that are intended to run from a ‘central’ linux server, primarily via a series of crons on that server. For anything very useful you’ll also need a database repository on this server as well – Standard Edition will do, although there are still some scripts that can run without a repository.

The basic idea behind AMChecks is to have something that can keep an eye on your Oracle infrastructure and give enough information to help you look after your Oracle estate while you’re off doing more exciting things (say, perhaps creating more scripts for AMChecks or, more rewardingly, picking the fluff out of your belly button). It’s as lightweight as I could make it – all you need to do for each target database is create a new login that has read access to the data dictionary\*. If you want to monitor mountpoint space (linux only at present) then there’s a little more effort required but you can add this in later once you’re happy with how AMChecks is running ‘out of the box’.

# What isn’t it?

It’s not a banana called Graham. It’s also not intended to replace anything you’ve paid actual money for. It will not give you a shiny interface and it won’t allow you to drill down into performance metrics. It is not intended to replace Oracle Cloud Control or serious monitoring software like BMC Patrol or Quest’s Foglight. You might find it useful if you have nothing else checking up on your databases. Feel free to give it a go and feel equally free to delete it when you find something more appropriate for your organisation.

# Is it free?

Yes. Of course it’s free. Do you think I could actually charge for this? ..although all donations, monetary or otherwise, will be gratefully received!

Please be aware that whereas I’ve tried to develop this for the lowest common denominator (so no diagnostic pack, no tuning pack, no enterprise edition even) there is no actual guarantee that running the scripts will not invalidate your Oracle license in any way. I mean they *look* OK but use them at your own risk. You have been warned! If you spot a red flag please let me know ASAP.

There is NO support offered for AMChecks either – you’re pretty much on your own although there might be some other poor unfortunates who also use the scripts that may be able to help you. If you can help them and either add new scripts or improve existing scripts (I suspect there’s A LOT of scope here..) please let me know – that would be 👍*awesome*👍.

# Pre-Installation

Check that you are clear to proceed as there may be company change control procedures, management sign-off or other non-technical ‘opportunities’ to overcome before any actual installation. Consider a plan for how you will roll-out these checks (e.g. test each script against a ‘sandbox’ database first, then deploy to your UAT environment). Be cautious – some databases will behave differently to each script so a ‘big bang’ deployment might just end up being all too appropriately named.

Identify the (linux) server that you intend to run the scripts from and make sure that you can ping all of your intended Oracle database hosts from this machine.

At the very least install an Oracle client on this box but realistically you’re going to want to install the database software too (check your Oracle licensing first of course before installing anything).

Make sure that you can run **sqlplus** to these boxes too.

You don’t need a new OS user/login but you might want to consider it. I developed these using user ‘oracle’ but feel free to give another user a whirl.

Check that this user has access to **cron** (unless you plan to use an alternative scheduler for your checks) and that the cron daemon is running on your ‘master’ server. E.g. run: **ps -elf | grep crond**

Find your Linux administrator (anywhere that has a significant amount of caffeine is a good place to start your search) if it’s not you (if it is you I’m not going to help you ‘find yourself’ – this isn’t self-help therapy) and discuss this section with them.

Make sure that you have the korn shell installed on your nominated server (unless you want to have fun converting the ksh scripts into some other kind of script (please share the pain/joy if you do!). Once installed, note the path of your ksh on the server. If it’s NOT **/bin/ksh** then either change the first line of **every** .ksh script to where your ksh is or create a symbolic link from where it is to /bin/ksh using the relevant user (probably ‘root’). Go with whatever your Linux administrator advises. e.g.

ln –s /blah/ksh /bin/ksh

Find or create a tnsnames.ora file. This should only contain the databases you want to check, at least to begin with. Edit the file to have all connect information for a database on a single line e.g.

FRED = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = bedrock.com)(PORT = 1521)) (CONNECT\_DATA = (SERVER = DEDICATED) (SERVICE\_NAME = FRED)))

WILMA = (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = bedrock.com)(PORT = 1522)) (CONNECT\_DATA = (SERVER = DEDICATED) (SERVICE\_NAME = WILMA)))

This may not be your usual tnsnames.ora file – you’ll want to keep it simple, so you might need to add new simple entries for your high availability databases and remove other entries. You also don’t want to have any entries for your standby databases, with the possible exception of active standbys. Only instances that you are licensed to run queries against should be in your tnsnames.ora.

Test this most-likely-slimmed-down tnsnames.ora (using **tnsping** with **TNS\_ADMIN** is probably the easiest way to test). Keep this file safe as you will need to copy this into your ‘AMChecks’ top level directory once you’ve downloaded the scripts.

# Installation

* Grab a caffeine-infused beverage. Tea is a good example although green tea is to be avoided, not just for this step but for the entirety of your life if possible: it tastes foul.
* Unpack the files that you’ve downloaded onto the box you intend to run your scripts from. The suggested directory is: **~oracle/amchecks**
* Copy your **tnsnames.ora** that you were working on in the pre-installation steps into this directory but call it **tnsnames.ora.full**. Create a slimmed down version of this file, called **tnsnames.ora** which contains just one line – the connect details for your ‘master’ database.
* Create the following OS directories if they don’t exist (it seems unlikely unless you were foolish enough to try AMChecks before):

**mkdir –p ~oracle/amchecks/external\_tables**

**mkdir –p /tmp/amchecks**

* If you don’t have a database to host your checks, create it and come back here when you are done. Unless you have a massive Oracle estate the schema itself should be well under 1 Gig.
* Create 2 new users – one to own the metadata and one to run the checks. Do this by editing (to change the passwords) and then running **users.sql** when connected to your host/master database (NOT the databases you will be monitoring) as a god-like user (e.g. ‘system’). Note that if you want to tighten up access you can almost certainly replace the ‘**SELECT\_CATALOG\_ROLE’** role with a bunch of more specific, focused system or object permissions. Be my guest but I’ve not explored this option yet. Also resist using **’SELECT ANY DICTIONARY’** especially if you are on Oracle 9 databases (don’t be embarrassed - you’re in good company!) – you don’t really want to be able to run : **select name, password from sys.user$** . Currently **’SELECT ANY DICTIONARY’** is only needed for 1 script: **password\_expiry\_check11.sql** The amu user shouldn’t really be able to see any business sensitive data or change anything. Don’t change this by adding any additional permissions.
* Know how to change all of your ‘amu’ passwords – this can be pretty easy as you can run something like: **ALTER USER amu IDENTIFIED BY t0pSecritPasswd;** on every database in a few seconds (but remember to change the **CONNECT\_PWD** in the file mentioned in the next section). Currently there is no provision to have different passwords on your monitored databases (although it wouldn’t be that hard to add in this functionality).
* If you are using prog\_checks.ksh (extremely unlikely even if you do have Progress databases TBH) ensure that the hard-coded password for user amu in this file is corrected. This should be the only script with the hard-coded password.
* Edit **~oracle/amchecks/.amcheck** using your favourite editor and change the database, connection details and anything else that looks likely. If you’re not comfortable with this then, no offence, you probably don’t want to continue with the installation.

**export ORACLE\_SID=ORCL**

**export CONNECT\_USERNAME=amu**

**export CONNECT\_PWD=** **fabum0use**

**export CONNECT="${CONNECT\_USERNAME}/${CONNECT\_PWD}"**

**export OWNER\_CONNECT=amo/** **ferretH8er**

**export READER\_CONNECT=amr/** **orch1dfr0g**

**export AMCHECK\_TNS="${CONNECT}@${ORACLE\_SID}"**

**export MAIL\_RECIPIENT='fred.flintstone@bedrock.com,wilma.rubble@bedrock.com’**

**export ORACLE\_BASE=/u01/app/oracle**

**export ORACLE\_HOME=/u01/app/oracle/product/11204/**

**export PATH=$PATH:${ORACLE\_HOME}/bin**

**alias SQLPLUS='sqlplus $CONNECT@$ORACLE\_SID'**

**export TERM=xterm**

**export FROM\_ADDRESS="DBAs@bedrock.com"**

I’m not sure why I have the CONNECT\_USERNAME and CONNECT\_PWD separate. Also I’m not convinced that I’m using all of these variables you know; I’ll need to check at some point…

ORACLE\_SID is the local database name of your master database; amu is the user that runs the checks; amo is the owner of the metadata objects. Amr is intended to be a purely read only user but I’m not currently using it as it’s too similar to amu at this stage.

Check file permissions of this file. Be sure to keep it as private as possible as it contains the passwords for the AMChecks (do something like **chmod 500** **~oracle/amchecks/.amcheck** once you are finished).

* At this point you can run one of the scripts – **is\_oracle\_ok.ksh**. You will need to tell it not to connect to the database to determine the list of targets but to get the list from parsing the tnsnames.ora file. Do this by running: **~oracle/amchecks/is\_oracle\_ok.ksh -t -p**

As an aside, once you ‘go live’ you’ll probably stop using the ‘-t’ option but it can be useful to run it periodically to check that your tnsnames.ora is not bloated with dead entries.

If this all works OK (it probably won’t as you might need to change environment variables or some other bits and bobs) then you can add other databases into your tnsnames.ora file until it is indistinguishable from **tnsnames.ora.full**

* Now setup **sendmail** on your ‘master’ server if it’s not already enabled. Do this as follows (again you’ll need to track down your linux administrator or whoever is responsible for linux mail) Note that some commands may vary depending on linux distro, version etc.:
  + **ps –elf | grep sendmail:** (look for a running process or two ..other than your **grep**!)
  + **systemctl list-unit-files | grep**  (look for an enabled service)

or better still, as root type **chkconfig |grep sendmail** (this will show the run levels too).

* + If its installed but not started, kick it off as root with **chkconfig sendmail** or **service sendmail start on**  which will add it as an automatically started service.
  + If it’s not installed, then install it (as root) e.g. **yum install sendmail**
  + Find and check your config file – e.g. **/etc/mail/sendmail.cf** checking that your relay host (if present) is added plus anything else your organisation normally sets. This will vary so you will have your own specific changes to make here.
  + If sendmail refuses to start (as root running **service sendmail start**) you may have something else using your port (25). Check this via:  **lsof –i :25**  (as root) to see what is using it (assuming **lsof** is installed..).

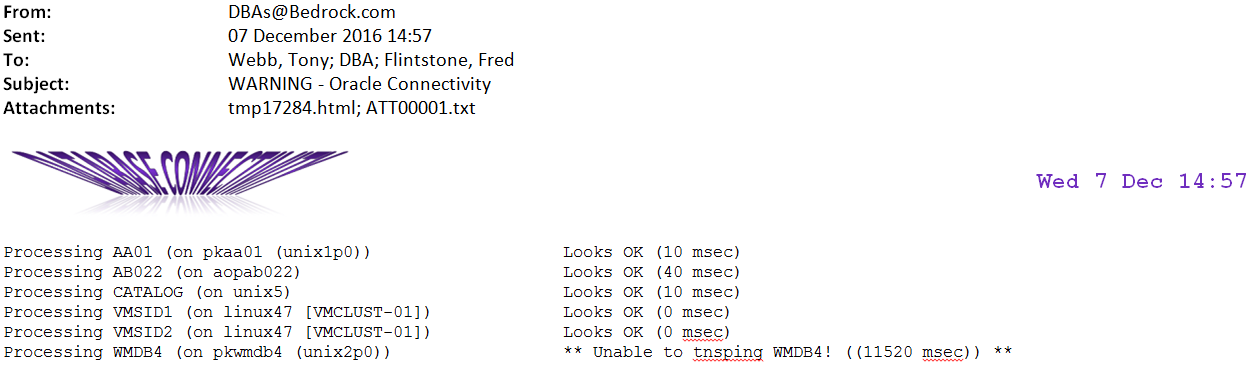
Generally this section tends to be the most frustrating (it’d the least Oracley and least databasey) but stick with it – it’s worth it!

* Next you can add a cron to periodically check your database connectivity by something like the following:

**02,07,12,17,22,27,32,37,42,47,52,57 \* \* \* 1-5 ksh -c '/home/oracle/amchecks/is\_oracle\_ok.ksh -t –m’**

This will run Monday through Friday every 5 minutes. You can schedule however you want. Of course, if you have any other scheduler then feel free to use that in preference to cron.

If your sendmail is working and your environment variables are all OK then you should get an e-mail that looks something like this:



Different command line parameters will alter if/when you get an e-mail. When testing keep the parameters simple.

This is a good point to stop and celebrate your success before carrying on!

* Next you want to create and populate your master database. This will take a while but again it’s something that you should start small with. Add 1 or 2 non-critical, and non-production target databases first. Running some of the other check scripts on a database has the potential, like most new queries, to impact performance of your database so it’s a good idea pick a suitable looking victim database to test with first.

So, as user **amo**, on your ‘master’ database, run ‘**am\_schema.sql**’

You are encouraged to keep an additional file named ‘**add\_to\_schema.sql**’ that contains any changes you have made to the **amo** schema, post-install. You might have your own tables (rare), your own columns (very common) so you should keep a note of these should you ever reinstall. If your download includes an ‘**add\_to\_schema.sql**’ already then take a look. You may or may not want to run it.

The schema will be mostly empty at this stage. Add in a couple of databases and their servers (add the server into **amo.am\_server** before adding the databases into **amo.am\_database**). You can start using other tables and bulking out these two tables later.

From the command line run the main am\_checks.ksh script. Specify it to run on one database (the one containing Amchecks) and to run just one script. You might as well e-mail the output too:

**./amchecks.ksh –d ORCL –s instance\_summary.sql -m**

Again, once this seems to be working (you might get a few issues again) you should stop. Maybe try a few more databases and a few more scripts. The next section describes how to add in your databases

# How to Add In Your Databases

The idea of Amchecks is to keep the installation on monitored databases as small as possible. With that in mind this is the process to add in a ‘new’ database to AmChecks:

**Step 1**: Create a new entry for the database server in table ‘am\_server’ if an entry does not already exist.

**Step 2**: Add a new entry for the database in table ‘am\_database’.

**Step 3**: Logon to the database to be monitored and create user amu plus it’s associated profile (this is the first section in script ‘users.sql’. There is no need to create the amo user so just cut-n-paste the first few SQL statements. You’ll want something like this:

**CREATE PROFILE am\_profile LIMIT**

**PASSWORD\_LIFE\_TIME UNLIMITED**

**PASSWORD\_GRACE\_TIME DEFAULT**

**PASSWORD\_REUSE\_MAX UNLIMITED**

**PASSWORD\_REUSE\_TIME UNLIMITED**

**PASSWORD\_LOCK\_TIME DEFAULT**

**FAILED\_LOGIN\_ATTEMPTS 10;**

**CREATE USER amu PROFILE am\_profile IDENTIFIED BY change\_this;**

**GRANT CONNECT, SELECT\_CATALOG\_ROLE, CREATE TABLE TO amu;**

**Step 4**: Back on the master database, add entries to table ‘am\_scriptskip’ if you want to omit particular scripts from the usual checks that run against the database for any particular reason.

**Step 5.** Ensure that the ‘new’ password has an entry in the tnsnames.ora file for amchecks. The entry should be all on one line to make parsing consistent.

# Back to The Testing..

Eventually let rip with all of the scripts on the one database, i.e. use the ‘**–d** ’ flag but NOT the ‘**-s** ’ flag and then one script on all databases (‘-s’ but not ‘–d’).

At this point you may start exploring the SIDSKIP and SCRIPTSKIP tables…

Next, read the relevant READ\_ME and any other likely looking text files in the installation directory. These SHOULD be more up-to-date than this document.

There will be some bugs and some things won’t work for you whereas they do for me, probably because I’ve made some unintended assumption that works for my databases but not for yours. Sorry about that. If you do manage to fix any of these or you improve the code (which, quite frankly wouldn’t be that difficult in some places..) please let me know as you may be helping others who use this script; more importantly you might be helping me ☺.

# Important Tables and Columns

If you decide to keep using the scripts on anything other than an ad hoc basis you’ll gain more familiarity with the AMChecks database. You are encouraged to add your own columns and tables over time. If you can share these with other DBAs then even better but please avoid using any features that you wouldn’t find in a ‘vanilla’ Standard Edition (SE, SE1, SE2) database running on Linux.

Personally I find maintenance and enhancements of AMChecks a good way to keep my hand in should I have to work on other, non-Oracle, databases or, worse still, do non-DBA work!

If you don’t have the luxury of this then here are the key tables and columns you should know about.

# **AM\_SERVER**

COLUMN: SERVER

The short name for the server (e.g. hostname –s). It must be unique.

COLUMN: DISABLED

This column appears on a lot of tables. As the name suggests, it’s an easy way to disable that particular row, so setting this to ‘Y’ in this table will cause the server in question to be ignored (unless you are referencing it via a non-disabled row in another able e.g. am\_database). One thing to watch out for – disabling a row in a ‘skip’ table disables the skip itself, not the thing you are skipping. I’ve not explained that very well but hopefully you know what I mean!

COLUMN: CLUSTER\_NAME

This is optional. If you have some kind of container for your server, e.g. a VM Cluster then specify this here.

COLUMN: PHYSICAL SERVER

This is the full name including the domain name (e.g. hostname).

COLUMN: PING\_DISABLED

Some servers may not be pingable for one reason or another. To skip just the ping tests (in is\_oracle\_ok.ksh) set this column to ‘Y’.

VIRTUAL COLUMN: PHYSICAL\_SERVER\_ABBREV

This is derived from PHYSICAL SERVER and for many, or quite possibly all, of your entries it will be the same as column ‘SERVER’.

# **AM\_DATABASE**

COLUMN: SERVER

The short name of the server where the database lives (join column to AM\_SERVER).

COLUMN: XXX\_IND

You can have as many ‘IND’ columns as you like. e.g. PRODUCTION\_IND. All IND columns should only have ‘Y’ and ‘N’ specified. Each column can be used to restrict/categorise which databases you run your scripts against. You can see a few examples of this in the sample cron. Again, you are encouraged to add your own IND columns as necessary.

COLUMN: OS\_CHECKS\_IND

This is different from the other IND columns (although it could be used in a similar fashion). It determines whether or not this database has an external table for that server’s space reporting. Only one database on each server should have this set. See a later section for more details on this.

COLUMN: RUN\_ORDER

This will determine when the database is processed when a list of databases is being processed.

# **AM\_SCRIPTS**

COLUMN: SCRIPT\_ID

A unique identifier for each script

COLUMN: SCRIPT\_TYPE

This is used by amchecks.ksh when running a series of related scripts. Scripts are normally categorised into either ‘CHECK’ which is a health check; ‘SUMMARY’ which is informative or ‘SPECIAL’ which describes scripts that don’t fall into either of the previous two categories.

COLUMNS: DB\_VERSION\_FROM and DB\_VERSION\_TO

Some scripts don’t work on older versions of the database. If you want to exclude scripts in this fashion set one or both of these columns. Number format should be three digits – e.g. 101 for version 10.1, 110 for version 11.0.

COLUMNS: PARAM1 and PARAM2

Columns to describe what parameters a script should take. Use the string ‘DBNAME’ to get the actual database name passed as a parameter, else use a literal value.

COLUMN: RUN\_ON\_MASTER

Indicates that when processing through a list of scripts for a target this particular script needs to get it’s information from the ‘master’ database. Normally the script will have a param (see PARAM1 above) of ‘DBNAME’

# **AM\_SIDSKIP**

When NOT to run scripts against a particular database. Useful for planned outages.

COLUMN: SIDSKIP\_TYPE

DAILY – means skip the database for every day between the specified ‘from’ and ‘to’ dates/times.; SATURDAY, SUNDAY etc. – specifying an actual day will only exclude the scripts for that database on a particular day of the week

COLUMN: SIDSKIP\_NOTES

The message to display when skipping a particular database.

COLUMN: DISABLED

If you have a database that periodically needs to be skipped but at irregular times it is worth having an open-ended entry here and just toggling the DISABLED column to ‘Y’ when you want it skipped rather than explicitly setting an outage window. However, be careful that youdon’t forget to re-enable this when you want checks to start again (..I’m speaking from experience here!)

# **AM\_SCRIPTSKIP**

Which scripts should be excluded for specific databases (don’t forget that you can also exclude scripts for specific versions of the database via the AM\_SCRIPTS table itself).

# **AM\_OS\_SPACE**

A brief history of filesystem space usage on database hosts. Used for alerting, nor capacity planning.

# **AM\_TABLESPACE\_SPACE**

Tablespace growth information. Could be used for capacity planning.

Adding OS space checks to AMChecks

For each (target) server:

as oracle: mkdir -p ~oracle/amchecks/external\_tables

mkdir -p /tmp/amchecks

. oraenv (pick your ONE database where the external table will 'live')

Copy the latest version of script 'os\_space\_check.ksh' from the master

AMChecks database server ~oracle/amchecks directory to this server’s

~oracle/amchecks directory and make it executable.

as sys: GRANT CREATE TABLE TO amu; (should already be set)

CREATE DIRECTORY amcheck\_dir AS '/u01/app/oracle/amchecks/external\_tables';

(or wherever your amchecks external tables will live)

GRANT ALL ON DIRECTORY amcheck\_dir TO amu;

as amu: CREATE TABLE AMU.AM\_OS\_SPACE\_LOAD

(

SERVER VARCHAR2(30 CHAR),

FILESYSTEM VARCHAR2(200 CHAR),

SIZEK NUMBER(10),

USEDK NUMBER(10),

AVAILK NUMBER(10),

PCTUSED NUMBER(3),

MOUNTPOINT VARCHAR2(200 CHAR),

DF\_DOW VARCHAR2(10 CHAR),

DF\_TIMESTAMP VARCHAR2(40 CHAR)

)

ORGANIZATION EXTERNAL

(

TYPE ORACLE\_LOADER

DEFAULT DIRECTORY AMCHECK\_DIR

ACCESS PARAMETERS

( RECORDS DELIMITED BY NEWLINE NOBADFILE NODISCARDFILE NOLOGFILE

SKIP 0 FIELDS TERMINATED BY ', ' MISSING FIELD VALUES ARE NULL

)

LOCATION (AMCHECK\_DIR: 'am\_os\_space\_load.dbf')

)

REJECT LIMIT UNLIMITED;

as oracle: crontab -e

# Amchecks crons:

##################

18 \* \* \* \* ksh -c '/u01/app/oracle/amchecks/os\_space\_check.ksh 1> /tmp/amchecks/os\_space\_check\_last\_output 2>&1'

(check/change full path to os\_space\_check.ksh)

Actually, change the cron so it runs while you wait (don't just run it from the

command line) then change it to the time you really want it run..

as amu: select \* from AM\_OS\_SPACE\_LOAD;

If all looks good, correct the cron and put your feet up.

as amo on master database: UPDATE AM\_DATABASE SET OS\_CHECKS\_IND = 'Y' WHERE…

update the record for the database chosen in the first step)

Sample Cron:

#

# Amchecks below here

#

##########################################################################################################################################################

## Database Connectivity checks should list everything once a day and check for errors only every 15 minutes

## Database check scripts - e.g. space, backups. Run these once in the morning, lunchtime and the evening with only error checks except the morning check

##########################################################################################################################################################

18 \* \* \* \* ksh -c '/home/oracle/amchecks/os\_space\_check.ksh 1> /tmp/amchecks/os\_space\_check\_last\_output 2>&1'

##########################################################################################################################################################

########################

# Non-Prd Section

########################

04,34 06-18 \* \* 1-5 ksh -c '/home/oracle/amchecks/is\_oracle\_ok.ksh -v -e -c -s -m -l non-prdisook -h "Non-Prod Connectivity Test" -S90 PRODUCTION\_IND+N 1> /tmp/amchecks/nonprod\_is\_oracle\_ok\_last\_output 2>/tmp/amchecks/nonprod\_is\_oracle\_ok\_last\_error'

24 08 \* \* 3 ksh -c '/home/oracle/amchecks/amchecks.ksh -c -m -x -t "CHECK" -h "Non-Prod Error Check" PRODUCTION\_IND+N 1> /tmp/amchecks/nonprod\_amchecks\_nonprod\_output 2>/tmp/amchecks/nonprod\_amchecks\_nonprod\_error'

50 08 \* \* 3 ksh -c '/home/oracle/amchecks/amchecks.ksh -c -m -x -t "CHECK" -a "fred.flintstone.@bedrock.com" -h "Non-Prod Error Check" PRODUCTION\_IND+N 1> /tmp/amchecks/nonprod\_amchecks\_nonprod\_output 2>/tmp/amchecks/nonprod\_amchecks\_nonprod\_error'

########################

# All Databases Section

########################

22 05 \* \* \* ksh -c '/home/oracle/amchecks/reconcile.ksh -m 1> /tmp/amchecks/reconcile\_output 2>/tmp/amchecks/reconcile\_last\_error'

37 06 \* \* 1 ksh -c '/home/oracle/amchecks/amchecks.ksh -d ORCL -c -s cluster\_summary.sql -H -h "VMWare Cluster Summary" -m 1> /tmp/amchecks/cluster\_summary\_last\_output 2>/tmp/amchecks/cluster\_summary\_last\_run'

31 06 \* \* \* ksh -c '/home/oracle/amchecks/amchecks.ksh -d ORCL1 -c -s rman\_catalog\_summary.sql -H -h "RMAN Backup Summary" -m 1> /tmp/amchecks/rman\_backup\_last\_output 2>/tmp/amchecks/rman\_backup\_last\_run'

35 06 \* \* 1 ksh -c '/home/oracle/amchecks/amchecks.ksh -d ORCL -c -s rman\_speed.sql -H -h "RMAN Backup Speeds" -m 1> /tmp/amchecks/rman\_speed\_last\_output 2>/tmp/amchecks/rman\_speed\_last\_run'

29 08,12,16 \* \* \* ksh -c '/home/oracle/amchecks/is\_oracle\_ok.ksh -v -e -c -m 1> /tmp/amchecks/is\_oracle\_ok\_last\_output 2>/tmp/amchecks/is\_oracle\_ok\_last\_run'

32 07 \* \* \* ksh -c '/home/oracle/amchecks/is\_sss\_ok.ksh -M 1> /tmp/amchecks/is\_sss\_ok\_last\_output 2>/tmp/amchecks/is\_sss\_ok\_last\_run'

03,08,13,18,23,28,33,38,43,48,53,58 \* \* \* \* ksh -c '/home/oracle/amchecks/is\_sss\_ok.ksh -S 1> /tmp/amchecks/is\_sss\_ok\_last\_output 2>/tmp/amchecks/is\_sss\_ok\_last\_error'

11 17 \* \* \* ksh -c '/home/oracle/amchecks/total\_space\_check.ksh -m 1> /tmp/amchecks/total\_space\_check\_output 2>/tmp/amchecks/total\_space\_check\_error'

31 17 \* \* \* ksh -c '/home/oracle/amchecks/tablespace\_space\_check.ksh -m 1> /tmp/amchecks/tablespace\_space\_check\_output 2>/tmp/amchecks/tablespace\_space\_check\_error'

00 05 \* \* \* ksh -c '/home/oracle/amchecks/amchecks.ksh -c -m 1> /tmp/amchecks/amchecks\_last\_output 2>/tmp/amchecks/amchecks\_last\_error'

53 06,09,12,15 \* \* 1-5 ksh -c '/home/oracle/amchecks/all\_os\_space\_check.ksh -R -S -m 1> /tmp/amchecks/all\_os\_space\_check\_output 2>/tmp/amchecks/all\_os\_space\_check\_last\_run'

58 09,12,15 \* \* 1-5 ksh -c '/home/oracle/amchecks/all\_os\_space\_check.ksh -v -p -r -S -m -t "Prod OS Space Alerts" PRODUCTION\_IND+Y 1> /tmp/amchecks/all\_os\_space\_check\_output 2>/tmp/amchecks/all\_os\_space\_check\_last\_run'

58 06 \* \* 1-5 ksh -c '/home/oracle/amchecks/all\_os\_space\_check.ksh -v -p -S -m 1> /tmp/amchecks/all\_os\_space\_check\_output 2>/tmp/amchecks/all\_os\_space\_check\_last\_run'

53 06 \* \* 0,6 ksh -c '/home/oracle/amchecks/all\_os\_space\_check.ksh -v -p -S -m 1> /tmp/amchecks/all\_os\_space\_check\_output 2>/tmp/amchecks/all\_os\_space\_check\_last\_run'

#####################

# Production Section

#####################

02,32 \* \* \* 0,6 ksh -c '/home/oracle/amchecks/is\_oracle\_ok.ksh -v -e -c -s -m -l prdisook -h "Production Connectivity Test" -S90 PRODUCTION\_IND+Y 1> /tmp/amchecks/prod\_is\_oracle\_ok\_last\_output 2>/tmp/amchecks/prod\_is\_oracle\_ok\_last\_error'

02,07,12,17,22,27,32,37,42,47,52,57 \* \* \* 1-5 ksh -c '/home/oracle/amchecks/is\_oracle\_ok.ksh -v -e -c -s -m -l prdisook -h "Production Connectivity Test" -S15 PRODUCTION\_IND+Y 1> /tmp/amchecks/prod\_is\_oracle\_ok\_last\_output 2>/tmp/amchecks/prod\_is\_oracle\_ok\_last\_error'

04 6,16 \* \* \* ksh -c '/home/oracle/amchecks/amchecks.ksh -c -m -x -t "CHECK" -h "URGENT Production Error Check" PRODUCTION\_IND+Y 1> /tmp/amchecks/amchecks\_last\_output 2>/tmp/amchecks/amchecks\_last\_error'

00 07 \* \* 1 ksh -c '/home/oracle/amchecks/amchecks.ksh -c -m -h "Production Error Check" PRODUCTION\_IND+Y 1> /tmp/amchecks/amchecks\_last\_output 2>/tmp/amchecks/amchecks\_last\_error'

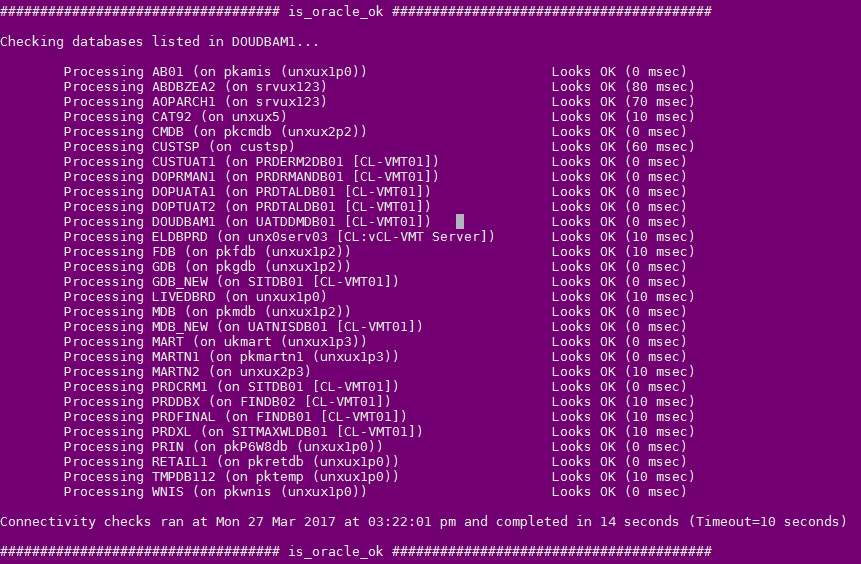
# **Some Scripts Explained**

is\_oracle\_ok.ksh

This is where it all began. A script originally designed to read through your tnsnames.ora file and tnsping all instances that it finds. Functionality was added to e-mail the results, then to connect using sqlplus, then to skip specific databases and then to have the database list held in a database.

Be careful about how you schedule multiple occurrences of this as they could trip over each other when reporting results.

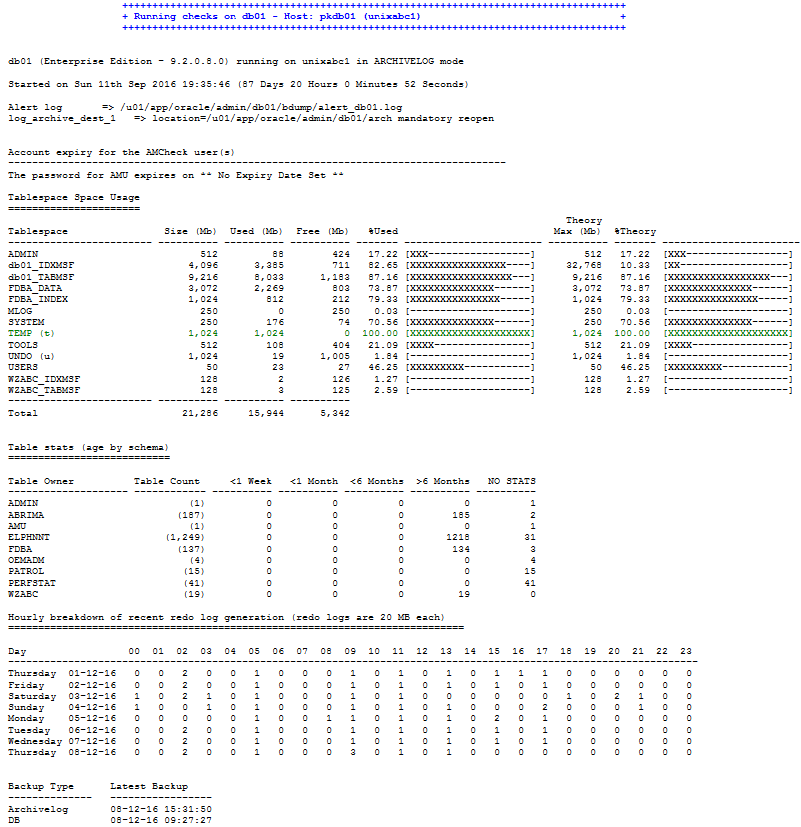




## amchecks.ksh

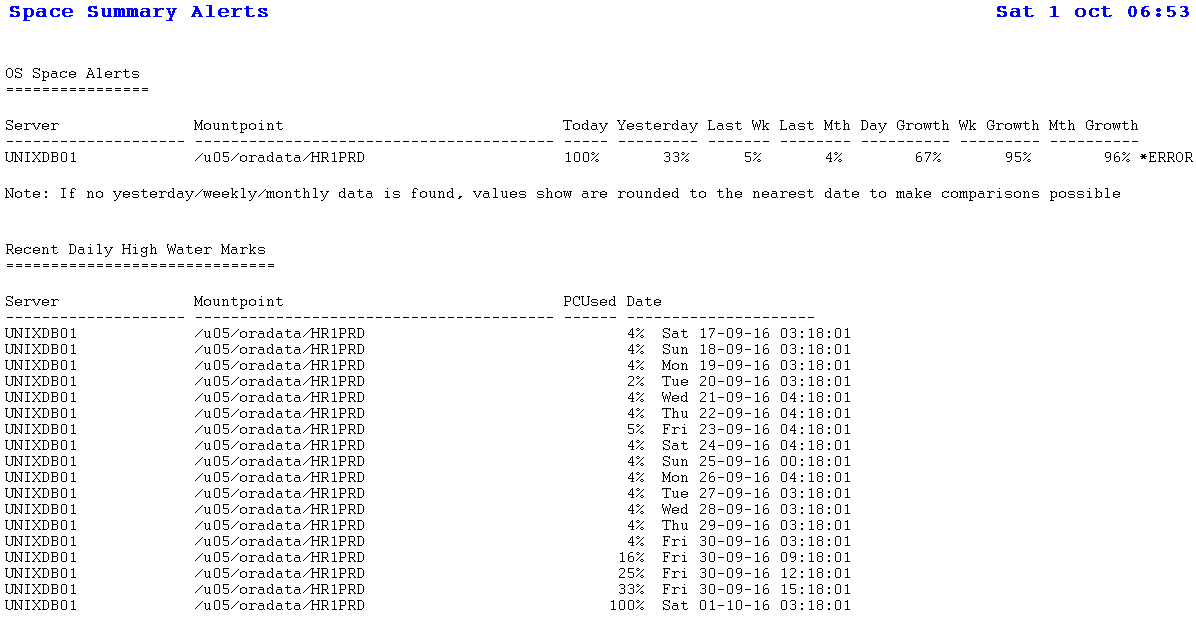
This is the main script. It’s basically a harness to process a bunch of scripts listed in a database against a bunch of databases. It’s pretty flexible but has mutated over the years to a point now where it would seriously benefit from a complete rewrite!

Be careful about file permissions. This script will run any sql script you tell it to (I thought about restricting this but this flexibility also makes the script really useful for ad hoc reports so no checks are currently in place). For this reason keep the file permissions on all sql and on the amchecks directory tight.



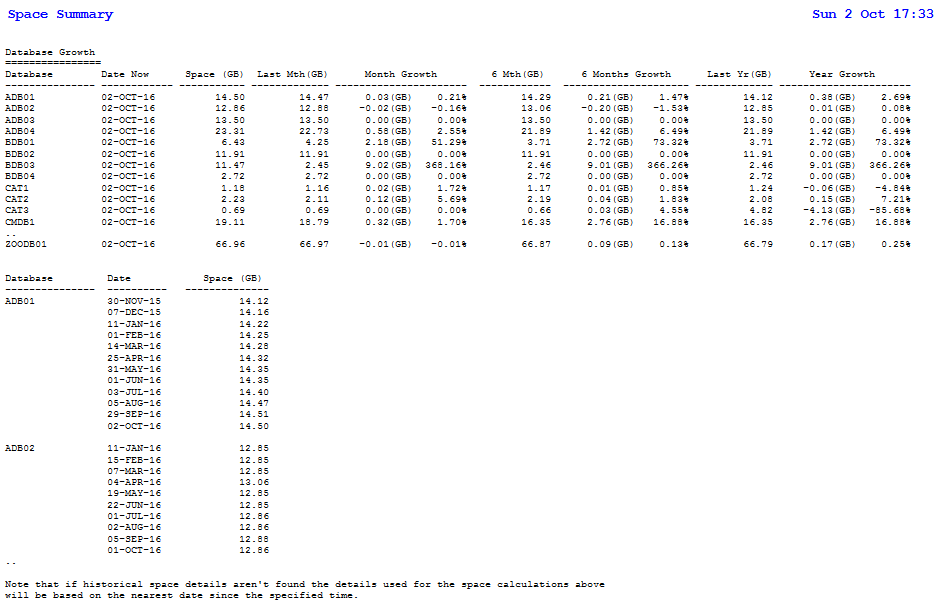
## all\_os\_space\_check.ksh

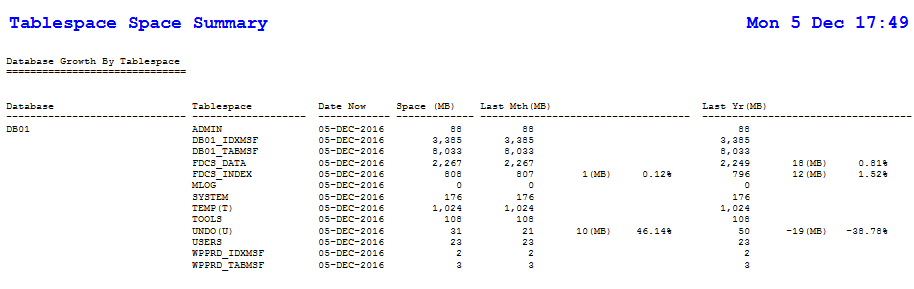
This is an attempt to move out of the database and to do some host checks. I’m not too keen on this as it means you have to spend a little effort creating external tables on your monitored databases. Still, it’s there if you don’t have anything else looking after disk space; you might need it (as I did). It can be useful but can also be prone to spamming.



tablespace\_space\_check.ksh and total\_space\_check.ksh

These are space reports showing how your databases are growing. They’ll be a bit useless until you’ve been capturing details for a few months.



****

SQL Scripts

102\_instance\_summary.sql

amazon\_disclaimer.sql

am\_expire.sql

am\_schema.sql

archivelog\_check.sql

blocking.sql

broken\_jobs.sql

cluster.sql

cluster\_summary.sql

database\_space.sql

dba\_feature\_usage.sql

dblink\_summary.sql

dfspace\_check.sql

growth\_report.sql

instance\_summary.sql

license\_pack\_usage\_details.sql

login.sql

metric\_check.sql

metric\_report.sql

metrics.sql

options\_packs\_usage\_statistics.sql

os\_space\_check.sql

os\_space\_summary.sql

pa\_patches.sql

parameter\_summary.sql

password\_expiry\_check11.sql

password\_expiry\_check.sql

preaudit.sql

prompt.sql

r12\_checks.sql

R12PROD\_disclaimer.sql

R12\_stats\_disclaimer.sql

recent\_creations.sql

redo\_activity.sql

redo\_check.sql

rman\_catalog\_summary.sql

rman\_check\_10.sql

rman\_check\_r12.sql

rman\_disclaimer.sql

rman\_speed\_db.sql

rman\_speed.sql

rman\_summary\_10.sql

rman\_summary\_r12.sql

segment\_growth.sql

space\_check.sql

space\_summary.sql

sptrends.sql

standard\_checks\_header.sql

standby\_check.sql

stats\_check\_10.sql

stats\_check\_9.sql

stats\_check.sql

tableinfo.sql

tablespace\_growth\_check.sql

tablespace\_growth\_report.sql

total\_space.sql

totrace.sql

true\_space\_check.sql

true\_space\_gig.sql

true\_space\_meg.sql

users.sql

version.sql

when\_analyzed.sql

# How To…

Once you’ve worked out how to use these scripts you should probably give yourself the fun-filled task of documenting how to use them. As an added incentive to do this I’m adding this section here which will be so badly written and confusing that you’ll have no choice but to do your own documentation :).

Once you are finished, if you think you’ve created a masterpiece or even if you’ve just created something better than this document (which can likely be achieved by up-ending your cutlery drawer over a keyboard) then please consider circulating to a wider audience or uploading to github <https://github.com/twebb667/AMChecks> .

## **Add a new database**

See [‘How to Add In Your Databases’](#_How_to_Add) earlier in this document in the installation section.

## **Scheduling a script with parameters**

If you call amchecks.ksh with the ‘-s’ parameter then you can run the same script (any script) on multiple databases. However, currently this differs from running any of the ‘usual’ scripts that are invoked as part of the ‘SUMMARY’ or ‘CHECKS’ options e.g. amchecks.ksh -t "CHECK"

as a script run via the ‘-s’ option will not accept any parameters to the actual script. What you can do, however, is use the ‘script set’ option - amchecks.ksh -j scriptid to create your own set – a set that could have just the one script should you wish it.

For example, script space\_summary.sql accepts 2 parameters – a tablespace name or partial name plus a threshold for the alerts. If you wanted to produce a report of only those tablespaces that are 90% or more full and only for production databases then this script could be used in a script set. Do the following:

1. Add a new entry to table AM\_SCRIPTS. Give it a SCRIPT\_ID of something like ’SPACE90’, enter SCRIPT\_NAME as ‘space\_summary.sql’, set PARAM1 to ‘%’ and PARAM2 to 90. SCRIPT\_TYPE should be ‘SPECIAL’.
2. Create a new script set by creating an entry in table AM\_SCRIPT\_SET. Call it what you like e.g. ’SPACE\_LITE’ (column SET\_ID) and optionally give it a title. Set column AM\_DATABASE\_WHERE\_STRING to ‘production\_ind = ‘Y’’.
3. Add an entry to table AM\_SCRIPT\_SET\_COMPONENT with the new SET\_ID and SCRIPT\_ID. Leave DATABASE\_NAME blank. Currently column TITLE isn’t used but hey, perhaps in a future release…?
4. Try it from the command line e.g. amchecks.ksh -j ‘SPACE\_LITE’

## **Changing who gets e-mails from amchecks**

Essentially there are 2 ways of controlling who gets e-mails – using the ‘-a’ parameter for a script (which will work for some but not all scripts) or changing the contents of environment variable MAIL\_RECIPIENT.

Use the ‘-a’ option for when you only want specific people to receive an e-mail. This is useful for alerting individuals that only have an interest in particular databases or a particular report. This option works well with any ‘IND’ columns that you have added on the am\_database table (see the next section).

To change MAIL\_RECIPIENT simply edit hidden file ‘.amcheck’ in the amchecks root directory e.g. ~oracle/amchecks. This will normally be changed if there is a new DBA or if a DBA, heaven forbid, leaves your organisation.

## Adding new ‘IND’ columns

This is an easy way to group databases together, generally for producing more targeted e-mails (in conjunction with the ‘-s’ parameter on amchecks.ksh). If there is a particular group of databases, e.g. all databases used by system erm.. ‘BEDROCK’ then simply add a new column to table am\_database via:

‘ALTER TABLE am\_database ADD bedrock\_ind CHAR(1 char) DEFAULT 'N'; ALTER TABLE am\_database ADD ( CONSTRAINT ck\_bedrock\_ind CHECK (bedrock\_ind IN ('N', 'Y')));’ Once you have done this, update the relevant database entries in table am\_database setting the new column to ‘Y’.

To report on this column call amchecks.ksh as normal but have the very last parameter(s) as, in our example, ‘BEDROCK\_IND+Y’. Multiple indicators can also be used for the same report e.g. the following which will e-mail a specific individual the outcome of running a space report on non-production ‘bedrock’ databases:

./amchecks.ksh -s space\_summary.sql -h “Non-prod\_Bedrock\_DBs” -a"fred.flintstone@bedrock.com" PRODUCTION\_IND+N BEDROCK\_IND+Y’

# Writing Your Own Scripts

These scripts, particularly the wrapper shell script amchecks.ksh can be a handy tool for running ad hoc queries against all of your databases (or logical groupings of databases using the ’IND’ columns in table amo.am\_database). However, this is clearly open to abuse as you can run any crappy code against your nice highly tuned production instance’s data dictionaries!

It’s common sense really but run any new code against non-production databases first, preferably starting with just one database and then running on larger numbers of databases.

If you want 1 line output for every database rather than a page full of output for each database target you may want to turn off the database title/name and suppress timings by using the -D and -f flags on amchecks.ksh although you’ll probably want to select the database name some other way in order to make the output useful.

For example, if you want to produce a list of how long all of your databases have been up for, you could create something like:

adhoc99.sql:

SET LINES 140

SET PAGES 0

COL instance\_started FORMAT A80 HEADING "Instance Started"

COL uptime FORMAT A55 HEADING "Uptime"

SELECT v.instance\_name || ' running on ' ||

NVL(SUBSTR(v.host\_name,1, INSTR(v.host\_name, '.')-1),v.host\_name) || ' started on ' ||

TO\_CHAR(v.startup\_time, 'Dy ddth Mon YYYY "at" hh24:mi:ss') AS instance\_started,

'(up ' || TO\_CHAR(FLOOR((SYSDATE-v.startup\_time)) || ' Days ' ||

MOD(FLOOR((SYSDATE-v.startup\_time)\*24),24) || ' Hours ' ||

MOD(FLOOR((SYSDATE-v.startup\_time)\*24\*60),60) || ' Minutes '||

MOD(FLOOR((SYSDATE-v.startup\_time)\*24\*60\*60),60) || ' Seconds' ||')') AS uptime

FROM sys.v\_$instance v;

exit;

Run this script via: amchecks.ksh -Df -s adhoc99.sql -m

html output will look something like:

####################################################### amchecks ###########################################################

APPLE running on fruitsv1 started on Sun 11th Sep 2016 at 19:35:46 (up 114 Days 19 Hours 7 Minutes 26 Seconds)

APRICOT running on fruitsv2 started on Sun 01st Jan 2017 at 16:34:14 (up 2 Days 22 Hours 8 Minutes 58 Seconds)

BANANA running on fruitsrv3 started on Sun 11th Sep 2016 at 19:35:46 (up 114 Days 19 Hours 7 Minutes 26 Seconds)

CHERRY running on fruitsrv1 started on Sun 01st Jan 2017 at 16:34:14 (up 2 Days 22 Hours 8 Minutes 58 Seconds)

…

####################################################### amchecks ###########################################################

# Thanks

Thanks to the poor unfortunate DBAs who had to work with me and put up with odd behaviour (from me as well as the scripts) and for the Linux guys who helped me with setting up the mail.

Thanks to Charles Berlin for his O’Rly? Generator (<https://dev.to/rlyy>) for the cover page!

Thanks to any brave/reckless souls who actually download and use these scripts!