VISCOSITY
NORTH AMERICA

On the Third Day of 18c/19c, Viscosity Gave to me...

Hacking Oracle 19c RPM Installations

December 10, 2020

Technology Overview

Database: 18c / 19c

Company: Oracle Corporation

Topic: Oracle 19c RPM Installations

Viscosity can help with any of your Database Upgrade needs

Viscosity has performed numerous zero-downtime database migrations and upgrades over the years and has a proven track record with business critical and mission critical databases.

Viscosity's Database Migration & Upgrade Services can plan, upgrade, validate and migrate all database content - quickly and effectively with our automated approach and proven methodology.

Learn more about how you can maintain and maximize your investments at viscosityna.com or email us at hello@viscosityna.com.

www.viscosityna.com

Oracle provides the preinstallation RPM for Oracle Database version 11g and later, making the configuration and installation of Linux environments faster and easier. They eliminate many of the tasks database and systems administrators perform to provision a new database host—creating users and groups, installing packages and dependencies, setting resource limits, and adjusting kernel parameters to satisfy database requirements—and assure organizations that their database environments are consistently (and properly!) built according to Oracle's recommendations.

Oracle introduced RPM-based installation of Oracle Database software with version 18c. In addition to creating and populating the Oracle Home, the RPM adds database creation through a service configuration script. The configuration calls the Database Configuration Assistant in the background and simplifies database creation. The entire process is performed by root and doesn't require interacting with DBCA, either through its GUI or via a response file.

This reduces the effort for users that want a database quickly. You needn't worry about how or what to run, because it's done for you. It also has implications if you work with configuration management, deployment, and automation tools like Ansible, Chef, Docker, Puppet, or Terraform. Configuring prerequisites, installing software, and creating a database are reduced to three simple steps. This makes Oracle database installation more consistent with what you may find from other databases.

The ease and convenience of the Oracle RPM installation does come with some limitations:

- Databases built from RPM are installed with predetermined directory structures
- There is no option to create database storage under ASM
- The installation creates single-instance databases only
- The only packages available publicly are for Oracle Database 19.3.0 Enterprise Edition and Oracle Database 18.4.0 Express Edition. There is no offering for Standard Edition, and references in documentation that once suggested Standard Edition would eventually be available have been removed.
- The process for creating databases with custom parameter and configuration settings is not very obvious.

These limits may lead you to conclude that package-based Oracle installations are only useful for quick and dirty database creation, and not for operational workflows or "real" environments. The good news is RPM-based installations can be modified and extended to address these limitations without too much effort, while remaining automation-friendly! For example, Viscosity uses RPM-based

installations for building Docker images and enjoy the same flexibility and options found in "normal" installations from archive.

Introduction to RPM

An RPM is a bundled set of files and instructions to install and configure software on Linux systems. RPM originally stood for Red Hat Package Manager, but is now ubiquitous across many Linux flavors. RPM can be applied via the rpm command, but you may be more familiar working with them through the yum package manager.

Installing software with a package manager has advantages - including simplicity, automation, and consistency.

Applying an RPM is a single command and eliminates most, if not all, human interaction. This makes installing software by RPM arguably safer, too. A single package file can be tracked and distributed through a repository, managed by version control, and validated with a checksum.

Modern provisioning workflows for Oracle often end before database installation. Prepared systems are handed off to database administrators, who complete the software installation and database creation and configuration. In many shops this isn't a frequent event and scripting the process may not make sense. Most DBAs have installed a database or two. The process is relatively straightforward and carefully documenting methods can take a backseat to other, more important and frequent tasks. Not to mention that we have the Internet, home to 1,001 ways to install Oracle!

This leads to environments of bespoke installations where base operating systems are built from configuration manifests, but databases aren't. Human involvement means we can't guarantee two environments are identical. With package-based installation, setup and configuration is built in and assures identical outcomes across systems with the same foundation. Database installation and creation is easily adapted to existing automation without adding complex, database-specific instructions. This ultimately leads to leaner processes with functionality similar to other databases.

Installation by RPM can also result in a smaller footprint and take less time to run as well. Packages can be built to exclude unnecessary assets and the Oracle 18c Express Edition RPM is a great example—it omits files and libraries for relinking the database binaries. The package installs Oracle "pre-linked" and those libraries and their relinking operations aren't needed. There is a potential downside to this I'll share later.

Basic Installation of Oracle by RPM

RPM installation is straightforward and well documented, by Oracle and on many blogs. Viscosity's team is no exception— Oracle 19c RPM Installation on OCI Free Tier and Installing the Oracle 19c RPM on Docker part 1 and part 2!

Briefly: download assets, apply the preinstallation and installation RPM, and run the configuration.

Download Assets

The Oracle Database RPM files for 19.3 Enterprise Edition and 18.4 Express Edition on Linux are available for download from the Oracle Database Software Download page.

Complete Preinstallation Tasks

The database RPM still require a properly configured host. This is most easily accomplished with the appropriate preinstallation RPM.

On Oracle Enterprise Linux the preinstallation packages are available locally.

```
yum -y install oracle-database-preinstall-19c
```

On other systems the preinstallation RPM must be downloaded. For Linux version 7, curl, install, and remove the RPM file:

```
curl -o oracle-database-preinstall-19c-1.0-1.el7.x86_64.rpm
https://yum.oracle.com/repo/OracleLinux/OL7/latest/x86_64/getPackag
e/oracle-database-preinstall-19c-1.0-1.el7.x86_64.rpm
yum -y localinstall oracle-database-preinstall-19c-1.0-
1.el7.x86_64.rpm
rm oracle-database-preinstall-19c-1.0-1.el7.x86_64.rpm
```

For Oracle 18c, substitute the package name:

```
oracle-database-preinstall-18c-1.0-1.el7.x86_64.rpm
```

Install the Database Software

For installation, copy the Oracle Database RPM to a directory on the local system, cd to the directory, and perform a local yum install. For Oracle 19c:

```
yum -y localinstall oracle-database-ee-19c-1.0-1.x86_64.rpm
```

There's no need to run the orainstRoot.sh or root.sh scripts; directory ownership and permissions are built in to the package.

The Oracle inventory is created at `/opt/oracle/oraInventory` and the Oracle Home under `/opt/oracle/product/19c/dbhome_1/.`

Installation of Oracle 18c XE is similar:

```
yum -y localinstall oracle-database-xe-18c-1.0-1.x86_64.rpm
```

The 18c Oracle Home is created as `/opt/oracle/product/18c/dbhomeXE`.

The RPM adds two additional files: a service file under `/etc/init.d` and a configuration file in `/etc/sysconfig`. The service file facilitates creation and management of Oracle databases, including starting, stopping, and restarting the database and listener.

Create a Database

With the software installed, the next step is to create a database with a default setup using the service script with the configure option. For Oracle 19c:

```
/etc/init.d/oracledb_ORCLCDB-19c configure
```

For Oracle 18c:

```
/etc/init.d/oracle-xe-18c configure
```

This results in creation of standard database listener listening on port 1521, and a container database with a single pluggable database.

The default 19c CDB is ORCLCDB and the PDB is ORCLPDB1.

The default 18c CDB is XE and the PDB is XEPDB1.

If you need a database quickly and aren't interested in customizations, this is the end of the journey. For the rest of us—those who want to alter the defaults, extend the limits of package-based installations, or deploy Oracle software to different directory structures—this is the beginning!

Hacking RPM Installs

Adaptations to Oracle's default RPM builds fall under two categories:

- Changes to database creation
- Changes to directory structure

18c Express Edition and 19c Enterprise Edition database creation is controlled by two files that contain and control configurations. Altering defaults is done by changing these files prior to running the service file to configure a database.

Changing the Oracle Home and Oracle Inventory paths on 19c is done traditionally, by moving and relinking the directories. Unfortunately, changing the Oracle Home for an 18c Express Edition database isn't an option. As noted before, the installation lacks libraries needed to relink the Oracle Home. There's no way of moving the files and relinking Oracle. (Of course, it's unlikely you'll need to change an 18c XE database home to match something different since 18c XE can only be installed from RPM.)

Basic Database Configuration Changes

There are two files of configuration in RPM-based installations. A very peculiar configuration file, and the database service file.

The "Configuration" File

This file is located under `/etc/sysconfig/oracledb_ORCLCDB-19c.conf` for Oracle 19c and `/etc/sysconfig/oracle–xe–18c.conf` for Oracle 18c. For being a configuration file, it contains far fewer configuration options than you might anticipate:

```
#This is a configuration file to setup the Oracle Database.
#It is used when running '/etc/init.d/oracledb_ORCLCDB configure'.
#Please use this file to modify the default listener port and the
#Oracle data location.

# LISTENER_PORT: Database listener
LISTENER_PORT=1521

# ORACLE_DATA_LOCATION: Database oradata location
ORACLE_DATA_LOCATION=/opt/oracle/oradata

# EM_EXPRESS_PORT: Oracle EM Express listener
EM_EXPRESS_PORT=5500
```

If you have a need to change these ports or the data file location, edit this file before running the configure script.

The Service File

The service file—the script called to configure and manage Oracle—is the source of most configurations you'd typically think of as "things to change." Located under `/etc/init.d/oracledb_ORCLCDB-19c` or `/etc/init.d/oracle-xe-18c` for Oracle 19c and 18c, respectively, the following entries appear near the top:

```
# Setting the required environment variables
export ORACLE_HOME=/opt/oracle/product/19c/dbhome_1

export ORACLE_VERSION=19c
export ORACLE_SID=ORCLCDB
export TEMPLATE_NAME=General_Purpose.dbc
export CHARSET=AL32UTF8
export PDB_NAME=ORCLPDB1
export LISTENER_NAME=LISTENER
export NUMBER_OF_PDBS=1
export CREATE_AS_CDB=true
```

We can edit some of these entries to change the database setup:

- ORACLE_SID
- TEMPLATE_NAME
- CHARSET
- PDB_NAME
- LISTENER_NAME
- NUMBER_OF_PDBS
- CREATE_AS_CDB

(You can't change ORACLE_HOME here to alter the database software install location—by the time this script runs the database software is already installed.)

Not Quite That Easy!

Manual changes are a matter of editing the files. If the objective is something appropriate for automation, we can apply edits with sed and parameterize scripts with values passed at run time. That will work perfectly—you can change everything noted above, with the exception of ORACLE_SID.

To see why, let's look at the code block that follows the environment variable settings:

```
# General exports and vars
export PATH=$ORACLE_HOME/bin:$PATH
LSNR=$ORACLE_HOME/bin/lsnrctl
SQLPLUS=$ORACLE_HOME/bin/sqlplus
DBCA=$ORACLE_HOME/bin/dbca
NETCA=$ORACLE_HOME/bin/netca
ORACLE_OWNER=oracle
RETVAL=0
CONFIG_NAME="oracledb_$ORACLE_SID-$ORACLE_VERSION.conf"
CONFIGURATION="/etc/sysconfig/$CONFIG_NAME"
```

Do you see it?

`CONFIG_NAME` (and `CONFIGURATION`) uses `ORACLE_SID`. If the `SID` is changed, a configuration file with the correct name must exist under `/etc/sysconfig` as expected by the service script.
When using RPM installations to create Docker images, I create the configuration file from a template as part of container startup. I also overwrite the default service script with a custom version and replace the environment variables with boilerplate such as `###ORACLE_SID###`. I then use sed to update the environment settings with whatever values are passed to the docker run command.

Advanced Configuration Modifications

The configuration and service files combine to provide basic options for creating new databases. RPM installation buries the mechanism for creating databases inside service automation and streamlines the process. The mechanism itself is still the Database Configuration Assistant, but we don't need to create methods to interpret variables as we would if we called DBCA directly.

Code that's very specific to Oracle remains separate from the automation. It's not difficult to see how a few lines can be substituted as part of a template to create databases in general, passing common properties like database name and character set, and use the same process to provision databases from multiple vendors; Oracle, MySQL, or others.

Extending DBCA Options

When the service script runs it passes values set in the script itself, alongside those it finds in the configuration file, to DBCA. The call to DBCA looks like this:

```
$SU -s /bin/bash  $ORACLE_OWNER -c "$DBCA -silent -createDatabase \
    -gdbName $ORACLE_SID -templateName $TEMPLATE_NAME -characterSet $CHARSET \
    -createAsContainerDatabase $CREATE_AS_CDB -numberOfPDBs $NUMBER_OF_PDBS \
    -pdbName $PDB_NAME -createListener $LISTENER_NAME:$LISTENER_PORT \
    -datafileDestination $ORACLE_DATA_LOCATION -sid $ORACLE_SID \
    -autoGeneratePasswords -emConfiguration DBEXPRESS \
    -emExpressPort $EM_EXPRESS_PORT"
```

These don't represent all options available with DBCA. For a more complete list, let's look at what's in the default response file under `$ORACLE_HOME/assistants/dbca/dbca.rsp`:

```
grep -Ev "^#|^$" /opt/oracle/product/19c/dbhome_1/assistants/dbca/dbca.rsp
responseFileVersion=/oracle/assistants/rspfmt_dbca_response_schema_v19.0.0
gdbName=
sid=
databaseConfigType=
RACOneNodeServiceName=
policyManaged=
createServerPool=
serverPoolName=
cardinality=
force=
pqPoolName=
pqCardinality=
createAsContainerDatabase=
numberOfPDBs=
pdbName=
useLocalUndoForPDBs=
pdbAdminPassword=
nodelist=
templateName=
sysPassword=
systemPassword=
oracleHomeUserPassword=
emConfiguration=
emExpressPort=5500
runCVUChecks=
dbsnmpPassword=
omsHost=
omsPort=
emUser=
emPassword=
dvConfiguration=
dvUserName=
dvUserPassword=
dvAccountManagerName=
dvAccountManagerPassword=
olsConfiguration=
datafileJarLocation=
datafileDestination=
recoveryAreaDestination=
storageType=
```

```
diskGroupName=
asmsnmpPassword=
recoveryGroupName=
characterSet=
nationalCharacterSet=
registerWithDirService=
dirServiceUserName=
dirServicePassword=
walletPassword=
listeners=
variablesFile=
variables=
initParams=
sampleSchema=
memoryPercentage=
databaseType=
automaticMemoryManagement=
totalMemory=
```

Theoretically, we can adapt the DBCA command in the service script to include any of those options. For starters we can set the NLS character set with `-nationalCharacterSet` or define initialization parameters with `-initParams`.

Since the service file calls DBCA to create the database, any and all options available from DBCA are possible with an RPM-based installation—it's just a matter of updating the DBCA command in the service script to include the options we want!

The limitations of RPM installation, listed in the introduction, are a product of what Oracle chose to include under the service script's configure option. A relatively trivial change introduces all sorts of possibilities! If you're exploring this option, our recommendation is to update the DBCA command in the script to read a response file, then create response files with parameters and values generated at run time.

This allows us to create single instance, RAC, or RAC One Node databases; container databases with multiple PDB; enable security options like Database Vault and Label Security; use ASM; and disable AMM!

Changing Directories

If your environment requires Oracle components to be installed in specific directories different from those used in the RPM installation, you can change them after the database software is installed. Unfortunately, they can't be altered as part of the installation itself. Packages embed directory paths, file ownership, and permissions. For Oracle database software installations, files are written to the system in an already-linked state and makes relining and running root scripts unnecessary.

Once created, the Oracle Inventory and Oracle Home directories can be moved with a few commands. After moving directories, make sure to update the service file at `/etc/init.d` to reflect these changes.

Move the Oracle Inventory

The default location of the RPM-based Oracle Inventory–`/opt/oracle/oraInventory`, beneath ORACLE_BASE—oddly violates Oracle's own recommendations:

> *The recommended value for the inventory directory is `/Oracle_base//../oraInventory`, or one level above the Oracle base directory, in the `oraInventory` subdirectory. If your Oracle base directory is `/u01/app/oracle`, then the Oracle inventory directory defaults to `/u01/app/oraInventory`.*

The following was used in the Docker build to change inventory locations:

```
mv "$OLD_LOC"/* "$NEW_LOC"/
find / -name oraInst.loc -exec sed -i -e \
     "s|^inventory_loc=.*$|inventory_loc=$NEW_LOC|g" {} \;
```

When changing Oracle inventory and home directories, move the inventory first! The inventory pointer location is used when updating the Oracle Home.

Change the Oracle Home

As mentioned previously, the Oracle Home for 18c XE installations can't be changed, because it lacks the libraries needed to relink. However, the 19c database is fully functional and can be relocated using Oracle built in utilities.

In the Docker work, Oracle Homes installed from RPM can be moved with the following code:

```
mv "$OLD_HOME"/* "$NEW_HOME"/
chown -R oracle:oinstall "$NEW_HOME"
rm -fr "$OLD_BASE"/product
sudo su - oracle -c "$NEW_HOME/perl/bin/perl \
   $NEW_HOME/clone/bin/clone.pl ORACLE_HOME=$NEW_HOME
ORACLE_BASE=$ORACLE_BASE \
   -defaultHomeName -invPtrLoc $NEW_HOME/oraInst.loc"
```

In 19c clone.pl is deprecated but still available, but may be de-supported in an upcoming release. For now, the same method works to move Oracle Homes for versions 11g onward and avoids the need for anything version-specific.

This method assumes the Oracle Base isn't changing. To change the entire Oracle Base:

```
mv "$OLD_BASE"/* "$NEW_BASE"/
chown -R oracle:oinstall "$NEW_BASE"
sudo su - oracle -c "$NEW_HOME/perl/bin/perl \
   $NEW_HOME/clone/bin/clone.pl ORACLE_HOME=$NEW_HOME
ORACLE_BASE=$NEW_BASE \
   -defaultHomeName -invPtrLoc $NEW_HOME/oraInst.loc"
```

Again, be sure to update the service script at /etc/init.d with the new paths for changed directories.

Apply Patches and RU

Package-based installation of Oracle 19c creates a fully-featured Oracle 19.3.0 Enterprise Edition database. There's nothing different about it besides the manner in which the software directories were created, and it's patched identically to a "normal" database.

After creating a new database environment from an RPM installation, it is recommended to apply Release Updates and patches prior to creating any databases; it reduces steps. Creating a database after patching creates a patched database. Patching after database creation requires stopping databases, patching software, starting databases and running datapatch.

Update OPatch

As always, update OPatch with the latest version from My Oracle Support. For new software installations, there is no point in backing up the OPatch directory; you can simply overwrite it with the new version as part of building the environment:

```
sudo su - oracle -c "unzip -oq -d $ORACLE_HOME
$PATCH_DIR/p6880880*.zip"
```

From there it's just a matter of applying the Release Update or patch. For building Docker images, the following was used to process numbered patch directories:

```
for patchdir in $(find "$PATCH_DIR"/* -type d -regex '.*\/[:digit:].+$' | \
    sed "s|/$||" | sort -n) # | grep -E "[0-9]{3}/" | sed "s|/$||" | sort -n)
  do cd "$patchdir"
       if [ "$(find . -type f -name "./*.zip")" ]
     then unzip -q ./*.zip
          chown -R oracle:oinstall .
          cd ./*/
          # Get the apply command from the README
          opatch_apply=$(grep -E "opatch .apply" README.* | sort | head -1 | awk
'{print $2}')
          opatch_apply=${opatch_apply:-apply}
          patchdir=$(pwd)
          # Apply the patch
          sudo su - oracle -c "$ORACLE_HOME/OPatch/opatch $opatch_apply -silent
$patchdir" || error "OPatch $opatch_apply for $patchdir failed"
     fi
done
```

This looks for patches in numeric subdirectories under $PATCH_DIR and loops over them in order. It unzips the patch file and changes ownership, then navigates into the patch subdirectory. There, it looks in the README for the apply method and runs OPatch in silent mode.

One of the sayings we have at Viscosity is our customer's, "have four aces in their pocket". Over the next 11 days, the talented staff at Viscosity along with our Oracle ACEs will address more Oracle Database 18c and 19c new features. Continue to join us next year, as we continue our Oracle Database 19c hands-on-lab workshops.

Happy Holidays!