An Oracle White Paper
October, 2013

Delivering Database as a Service (DBaaS) using Oracle Enterprise Manager 12c
Executive Overview ................................................................. 2
Evolution of ‘Database as a Service’ ......................................... 2
Managing the Database Lifecycle ............................................. 4
Traditional Approach to Database Provisioning .................... 4
Database as a Service ............................................................... 5
Setting Up Database as a Service .............................................. 6
  Zones: Allocation of Server Resources .................................. 6
  Pools: Pooling platform resources for running databases ........ 7
  Service Catalog: Setting up and Enabling Services ............... 7
Delivering a turnkey DBaaS solution via The Rapid Start kit: ... 9
Example Use Cases .................................................................. 10
  Provisioning a New Database Service ................................... 10
  Cloning (using Full Clone or Snap Clone) ............................. 11
The Self-service Experience .................................................... 13
Chargeback (Showback) .......................................................... 16
Cloud API and Orchestration .................................................. 17
Conclusion ............................................................................... 19
Executive Overview

Large enterprises today have hundreds and thousands of databases of various versions, configurations and patch levels. Another challenge is around time to provision new databases. When an end user, be it a developer or a QA engineer, needs a database he or she typically has to go through an approval process, which then translates into a series of tasks for the DBA, the sysadmin and storage admin. This is a cumbersome and time-consuming process and may span days. Due to non-optimal usage of finite computational resources and significant IT latency, it also possesses the risk of overspending by organizations.

Cloud services from Oracle Enterprise Manager 12c is a path-breaking technology that delivers self-service deployment of IT resources for business users along with resource pooling models that cater to various multi-tenant architectures. Database as a Service (DBaaS) is a paradigm where end users (DBAs, Developers, QA Engineers, Project Leads, etc) can request database services, consume it for the lifetime of the project, and then have then automatically de-provisioned and returned to the resource pool.

Database as a Service (DBaaS) provides

- A shared, consolidated platform to provision database services on
- A self-service model for provisioning those resources
- Elasticity to scale out and scale back database resources
- Chargeback based on database usage

Evolution of ‘Database as a Service’

Database as a Service primarily started as a consolidation exercise for reducing capital expenditures (CAPEX), but as it evolved, organizations started looking into other key drivers like self-service, showback, etc. The 2013 IOUG Survey clearly illustrates the various objectives behind Database as a Service.

![Figure 1: A Macro Level Analysis of an IOUG survey, 2013](image-url)
There are various consolidation models to provide DBaaS as shown in the figure below. The simplest and most prevalent form of consolidation exists around server virtualization. Server virtualization offered a simple way of running multiple operating system instances on the same hardware. A better model is to consolidate multiple database instances on the same operating system or a cluster. However, in both these cases, database sprawl is still an issue that invariably leads to larger administrative overheads and compliance challenges. An even better consolidation model could be delivered by hosting multiple schemas from different tenants within the same database.

Oracle Enterprise Manager 12c delivers the complete spectrum of database consolidation as depicted above. This also provides organizations a full range of choices regarding service delivery vis-à-vis technology and business implications. Table 1 below compares the different consolidation models and their advantages and disadvantages.

We therefore, have a new consolidation model with Database 12c. It is called the pluggable database, available with database 12c multi-tenant option. In this configuration, a container database can host multiple pluggable databases, each potentially representing a different tenant or an application. Pluggable database combines the best of all the other consolidation models, hence while each pluggable database has its own memory structures, they are patched and often administered at the container database level. Pluggable databases therefore, provide an excellent way of consolidation databases while maintaining isolation and security. It also offers fast provisioning and de-provisioning using simple plug and unplug mechanisms, thereby resulting in significant cost and effort savings.
Managing the Database Lifecycle

In today’s information age the heart of any organization resides inside the databases their IT maintains. Some of the databases serve critical day-to-day functions such as ERP and have a long lifespan, while lot of other databases serve adhoc, peripheral projects and have limited lifespan. Each of these databases could have their lifecycle counterparts, such as Development, Test, Stage and Production. The important point to acknowledge is that IT has to deal with a continuous stream of requests for databases belonging to any of the above categories. The collective force dealing with the above scenario comprises of system, storage, database administrators and developer plus QA, release management community. Classically the enhancement or a new request originating from business lands on a project task force group of developers whose first request is either a latest copy of the source database or an empty database of similar kind. Despite best of release management procedures the real scenario always involves various applications are compatible only with heterogeneous versions of the database on different variant of operating system platforms.

Traditional Approach to Database Provisioning

If the project is about creating a new application then a request is made for the latest, greatest version of a database instance, where as the actual the usage profile is only going to be limited to creating a specific schema or two. Now the system administrator needs to figure out a suitable server (host) for the workload. The storage administrator configures storage and partitions them for consumption by the database administrator (DBA) to do the requisite version of the software installation. This flow, in the most ideal condition, consumes anywhere between 2 to 3 days.
This could be further complicated by various application and operating system certifications, high availability needs, and other considerations. This could easily add another 3-4 days to the project, making a new database provisioning a week-long process or longer.

The final challenge for the administrators remains to have some accountability of all the allocated hosts, storage and provisioned databases. Traditionally it can be a tedious process to pin-point which databases are being used actively and when these resources will be released.

The above flow depicts a traditional model of delivering databases, which do not adequately address the core issues around consolidation, standardization, automation and accountability. Each of these provisioning requests could result in a non-standard configuration and without proper pooling they can lead to a very fragmented IT infrastructure. What’s needed therefore is a platform of well-defined services, with proper governance rules incorporated into them, which is discussed in the next section.

Database as a Service

To address these challenges, most IT organizations are exploring the benefits of Cloud Computing within their own datacenter. Whether it is faster provisioning, on-demand access, agile resource scheduling based on policies, or chargeback rules to ensure business accountability and more control over the environment, IT must move from reactive to a proactive and predictive approach for data center management.

Enterprise Manager, Oracle’s flagship product for systems management provides industry’s most complete solution for Cloud management. It offers a single, integrated console for deploying, operating, monitoring, diagnosing, and troubleshooting, today’s complex IT environments. It offers a simple, scalable solution for running Oracle databases for both A and B category systems in Cloud environments. Enterprises must support hundreds or even thousands of applications to meet growing business demands. This growth has driven up the cost of acquiring and managing servers and storage. Clouds enable customers to consolidate servers, storage, and database workloads onto a shared hardware and software infrastructure. By providing on-demand access to software and infrastructure in a self-service, elastically scalable and metered manner, Enterprise Manager offers the following benefits.

- **Improving Quality of Service**: IT organizations are not only trying to drive down costs, they are also looking at solutions that will simultaneously improve quality of service in terms of performance, availability and security. Cloud consumers inherently benefit from the high availability characteristics built into the Cloud.
- **Providing Resource Elasticity**: The ability to grow and shrink the capacity of a given database, both in terms of storage size and compute power, allows applications the flexibility to meet the dynamic nature of business workloads.
- **Rapid Provisioning**: Databases in a Cloud can be rapidly provisioned, often by way of a self-service infrastructure, providing agility in application deployment. This reduces overall time in deploying production applications, development platforms, or creating test bed configurations.
- **Accountability**: Database usage in a cloud needs to be measured for budgeting and planning purposes and also for distributing the administrative resources based on resource usage.
Setting Up Database as a Service

Zones: Allocation of Server Resources

The first step in enabling any database service lies in identifying the physical resources the service will run on. And that is by no means an easy task. This could be based on business considerations (LOBs, for example), lifecycle status of the database (development versus production), workloads (OLTP versus Data Warehouse) or even Quality of Service (Highly Available versus not). Each of these considerations can be addressed by segregating the infrastructure into units, which are known as Zones in Enterprise Manager. Let’s assume an organization ‘mycompany’ has three LOBs namely Finance, Manufacturing and HR. There are 1000 servers available with corporate IT in their datacenter. The IT Admin can choose to segregate these servers based on those LOBs and further subdivide them into production and development servers. The zones that host platform services such as Database as a Service or Java as a Service are classified as Platform as a Service Infrastructure (PaaS) zone. As the first step of streamlining the System Administrator can divide the hosts into PaaS Infrastructure zone, define the placement policy constraints and impose access controls on the zone.

- **PaaS Infrastructure Zone**: A PaaS infrastructure zone is a group of hosts, that could be represented by Exadata compute nodes, Oracle VM guests or any other host supported by Enterprise Manager. Each resource in a zone represents a location at which a service instance is to be deployed.

![Figure 4: Oracle Enterprise Manager Cloud Resources depicting PaaS zone](image)
Once we assign the zone and implement access controls to the LOB users, it eliminates the need to go searching for an available host every time an adhoc need crops up.

**Pools:** Pooling platform resources for running databases

The DBA intrinsically knows the exact versions and flavor of databases being used within each LOB along with the operating system version compatibility. As the next level of streamlining he/she can add each unique type of the database configuration to a single place called Pool. For example, single Instance 11.1.0.7, cluster database 11.2.0.2 …etc.

A database pool contains a set of resources that can be used to provision a database instance within a PaaS Infrastructure Zone. The different kinds of DBaaS pools include:

- **Pool for database:** A collection of servers or clusters with database software installed.
- **Pool for schema:** A collection of databases containing schemas with or without seed data.
- **Pool for pluggable database:** A collection of container databases of same 12c version level.

**Service Catalog: Setting up and Enabling Services**

A sophisticated Database as a Service solution needs to cater to a variety of use cases; some common examples are shown below:

- A developer or a project owner requiring a new database service, with or without seed data
- QA requiring a full database refresh for intense load testing
- QA requiring to create multiple clones for functional testing on subset of data

When these use cases are mapped to the consolidation models described before, we get the following requirements as shown in Figure 5, each of which are supported by Enterprise Manager 12c. The administrator can create multiple services based on configuration (Small, Medium, Large) or other characteristics.
Delivering Database as a Service (DBaaS) using Oracle Enterprise Manager 12c

Figure 5: User Requirement modeling for DBaaS

To support the above requirements, the DBA needs to create gold images or provisioning profiles (in Enterprise Manager parlance) and stage them in the Software Library. A database provisioning profile is an entity that captures source database information for provisioning. In fact when a provisioning profile is created from an existing installation, it provides the flexibility to clone Grid Infrastructure (with software or configuration) and Oracle Database (with software or configuration). A profile can represent a complete database or a set of related schemas that form an application. Enterprise Manager 12c supports the creation of profiles using the following:

a. DBCA Template
b. RMAN backup
c. Snapshots
d. Export dump of schema objects

Now that the DBA has got Hosts grouped in PaaS zone(s) and all kinds of provisioning profiles created, the last piece of the challenge to be solved is to enable the project users to be able to use these resources autonomously on-the-fly for their needs. The solution from Enterprise Manager 12c is ‘Service template’. It is a standardized service definition that is offered to self-service users to create databases or schemas. Self-service users can create one or databases or schemas based on the service template definition. The beauty lying beneath is in the fact that a single profile can be used to create multiple service templates. Also access to each of the published ‘Service Templates’ can be controlled by defining which specific Roles can use which template(s), for example.
- Service Template 1: This is a copy of the Oracle Home and associated structure of a Category B system database by means of a DBCA template.
- Service Template 2: This is a RMAN backup copy of the Oracle Home and the complete database of a Category A system.
- Service Template 3: This defines various workload configurations and allows creation of empty schemas.
- Service Template 4: This is an export schema dump of a specific Category B application.

![Database Profiles and Service Templates](image)

**Figure 6: Database Profiles and Service Templates**

For a Database provisioning service template computing resource consumption of any of its instantiations can be pre-defined by the DBA using initialization parameters settings on the template itself.

**Delivering a turnkey DBaaS solution via The Rapid Start kit:**

Enterprise Manager 12c makes it simple to set up a cloud for administrators by providing a set of administrative APIs to setup the underlying resources and services. Furthermore, these APIs have been further simplified into a single setup script for Exadata that converts its compute nodes into a full-fledged database as a service pool. Administrators simply have to run the script and make the self-service interface available for the users.
Example Use Cases

Provisioning a New Database Service

Developers and project owners often need a new database for their projects. These databases may optionally be seeded with some data. For Greenfield application development projects the requirement only mandates an empty database of a particular deployment type [Single Instance or Cluster] using a supported version of Oracle software. The new database is provisioned using pre-defined Service Templates that define the characteristic of the database. The Service Templates leverage Deployment Procedures to execute the provisioning process. The database is automatically placed in the appropriate host within the pool of servers, using criteria such as population (number of databases already deployed on the host) or load (current CPU and memory usage). At the end of the provisioning process, the user is provided a connect string to use the database.

As shown in the comparison table 1, the dedicated database model, whether implemented on VMs or on physical may lead to database and operating system sprawl. It makes it difficult for administrators to administer, backup, patch these software and finally be answerable for any compliance violation.

Schema as a Service

Not every application needs a dedicated database. Small, home-grown applications, unlike ERPs can often share database resources with other applications, especially during development cycle. Schema as a service is the Oracle solution to support the shared database delivery model. The key objective is to support a multi-tenant model on the database by means of which the DBA can host same/different applications on a database without compromising security and resources. The Self-service administrator publishes a Service Template that will create a database service with one or more schemas with/without seed data and database objects. The end user [developer] can request the published configuration by selecting the Service Template, at the end of which a database service is created that is loaded with the necessary schemas. In a nutshell, every Schema as a Service request provides the user with:

- Database Service (To allow the SSA user to connect to the database)
- One or more Schemas w/o seed data (Depends on the service template)
- One Master account that has grant on all the schemas
- Single Tablespace (Schemas created as part of request will use this tablespace to read/write data)
- Export/Import (Users can backup and restore their own private data)

The implementation of ‘Service level Isolation’ for schema as a service is best defined into two categories:

Resource Isolation: The resource level isolation for

- **Data/Storage**: Is achieved by creation of a unique table space for every user which ensures data integrity and security at request level.
- **CPU**: Is achieved by means of resource manager (if enabled). Once the resource manager option is enabled, each and every schema created as part of the service request is placed in a consumer group as specified which ensures a fair share usage of CPU cycles in a multi-tenant model.

- **Memory**: Since the database does not have any tool to limit the memory usage, we provide alerts and display warning to the admin and users when a service exceeds the threshold limit specified.

**Security Isolation**: The security level isolation is achieved by means of providing appropriate grants and privileges to the schemas that get created as part of service request. The Data Vault configurations can be used optionally to provide an extra level of security.

The benefits of Schema as a Service are obvious. Sharing databases prevents database sprawl leading to minimal administrative and configuration compliance challenges. For example, it only needs a single patching session to patch a database hosting hundreds of schemas. Traditionally, which would have involved provisioning a complete database, optimal usage of Schema as a Service could result in better hardware utilization of more than 40%.

For enterprises that have adopted database 12c, pluggable databases offer a more sophisticated version over schema as a service.

**Cloning (using Full Clone or Snap Clone)**

One big consumer group of DBaaS is QA Engineers or Testers. They perform an operation called User Acceptance Test (UAT) for various applications. To perform UAT, they need copies of the production database. For intense testing, such as in pre-upgrade scenarios, they need a full updateable copy of the production data. There are other situations, such as in functional testing, they need to perform minimal updates to the data, but at the same time, need multiple functional copies. Enterprise Manager 12c supports both the scenarios. In the former case, it leverages RMAN backups to clone the data. In the latter case, it leverages the “Copy on Write” technology at the storage layer to perform Enterprise Manager 12c Snap Clone (or just Snap Clone). Currently, Snap Clone supports generic ZFS and the cloning APIs provided by NAS technologies viz. Netapp and ZFS Storage Appliance. By using this technology, the entire data does not need to be cloned, but the new database can physically point to the source blocks within the same filer and only needs to allocate new blocks if there are updates to the cloned copy. A comparison of the two models is shown below.
The type of provisioning profile that leverages storage snapshots is called ‘Snap Clone’. The best way to think of snapshots is that it is a point-in-time view of the data. It’s a time machine, letting you look into the past. Because it’s all just pointers, you can actually look at the snapshot as if it was the active filesystem. It’s read-only, because you can’t change the past, but you can actually look at it and read the data. NetApp and SunZFS snapshots just write the new information to a special bit of disk reserved for storing these changes, called the SnapReserve. Then, the pointers that tell the system where to find the data get updated to point to the new data in the SnapReserve.

- **Space efficiency:** Since we are only recording the deltas, you get the disk savings of copy-on-write snapshots (typically a few hundred kilobytes for a 1 terabyte database). But you are not copying the original block out of the way, so you don’t have any significant performance slowdown.
- **Time efficiency:** Because the snapshot is just pointers, to restore data (using SnapRestore), we simply update the pointers to point to the original data again. This is faster than copying all the data back from the snapshot area over the original data, as in copy-on-write snapshots. So taking a snapshot completes in seconds, even for really large volumes (like, terabytes) and so do restores. A typical terabyte database therefore takes only a couple of minutes to clone.
Keeping in mind the usefulness from an end user (developer) perspective it is a best practice to keep the provisioning profiles of Production Databases using Snapshot or RMAN backup not to be older than 1 Month.

An additional feature that comes with Snap Clone is the ability to snapshot multiple incarnations of the database without consuming any significant additional space. This addresses the QA Engineer’s ubiquitous requirement to go back and forth with the data that they would test with. Each QA Engineer can take multiple backups of the cloned copy. Using Enterprise Manager, one can literally “time travel” across these snapshots, and restore to an earlier point in time.

The Self-service Experience

Delivering a self-service paradigm is often one of the driving forces behind adoption of cloud computing. The self-service nature of cloud computing lets end users obtain and remove cloud services themselves without requiring the assistance of an IT staff member. Cloud users can obtain cloud services through a self-service portal. Because business users can obtain and configure cloud services directly, this enables IT staff to be more productive and gives them more time to manage cloud services.

Every self-service user within Enterprise Manager is explicitly provided access privilege to a specific PaaS Infrastructure zone, which is implemented using the Role Based Access Control (RBAC) model for the cloud resources in Enterprise Manager.

When a user logs on to the Self-service portal (assuming the role of this user has got access to all the four service templates), (s)he is presented with a service catalog with options to create a full database or
Delivering Database as a Service (DBaaS) using Oracle Enterprise Manager 12c

a service with one or more schemas. Depending on the nature of his project requirement now it’s up to him to decide what suits best for the work e.g. if the objective is to clone a complete database for SIT or UAT, an instantiation of the appropriate Service Template is all that is needed. Once (s)he submits this request Enterprise Manager will automatically determine based on the definition of the pools by executing a target placement fitment algorithm which is the best host for the database requested and which is the best database available for the schema requested. Also the self-user has got the options to specify from (s)he needs the requested resource and till when.

Figure 9: Self-service Console for DBaaS

In short now we are talking of a scenario where being true to the definition of a cloud agility model corporate IT is providing on-demand access to the just in time provisioned resources through the means of a self-service.

It is also important to note that provisioning process not only creates the database service, but also makes it completely manageable from Enterprise Manager. Most cloud applications simply end with the provisioning process transferring the load of management to the administrators. Enterprise Manager on the other hand creates the database services, configures them and discovers them as targets as shown in Figure 8 for Snap Clone. As a result, the Self-service console not only enables the provisioning of database services but also exposes the users to a limited set of operations. They can startup and shutdown their databases, perform backup and restore and also monitor key metrics related to the database activity.
After streamlining all the available resources from a data center for on-demand business development consumption there is also a very critical need for defining checks and balances around finite consumables. Few best practice methods for imposing self-service governance in context of Database as a Service are described below:

- **Request Settings**: Here DBA can specify how far in advance a request can be made, maximum duration for which requests can be made, and finally define a period for which archived requests are stored before they are automatically deleted by the system.

- **Retirement**: The retirement request for each self-service request can be defined in three ways:
  a. While requesting the service (specify the retirement date and time). This schedules a cleanup Deployment procedure which can be re-scheduled if required later.
  b. Can be defined after the request is processed. The cleanup Deployment Procedure is scheduled as defined by the user.
  c. The user can re-schedule the ‘Delete’ request. This reschedules the cleanup deployment procedure in the backend based on the new schedule.

- **Quota Settings**: It is the aggregate amount of resources that can be granted to each self-service user belonging to a certain role. While catering to any self-service request Enterprise Manager transparently performs every time a quota validation to ensure that a request is executed only if the user’s current usage is within the limits specified for the roles to which the user belongs. Behind the scene, Enterprise Manager automatically imposes the quota based on the following attributes:
a. **Databases**: The number of databases is computed based on the number of self-service requests (Database as a Service) that are either in ‘Scheduled’ or ‘Succeeded’ state.

b. **Database Services**: The number of services is computed based on the number of self-service requests (Schema as a Service) that are either in ‘Scheduled’ or ‘Succeeded’ state.

c. **Memory**: The memory allocated is accounted as the sum of the following:
   The memory allocated (SGA + PGA) for each database + the memory defined in workload for each ‘Schema’ service request (Includes ‘Scheduled’ and ‘Processed’).

d. **Storage**: The total storage is the sum of the following:
   The sum of all the tablespaces created in the database (for DBaaS) + the writable space of a snap cloned database (applicable only for thin cloning) + the storage as defined in the workload (ais mapped to the tablespace size for schema as a service).

**Chargeback (Showback) Settings**: The self-service paradigm may induce a propensity to provision more and more database services. Each DBaaS user therefore may be governed and held accountable by a sophisticated chargeback application within Enterprise Manager. The chargeback feature is described next.

**Chargeback (Showback)**

Self-service provisioning and subsequent consumption of database resources need to be properly accounted for, especially when the platform is shared among multiple users and user groups. Therefore, in addition to quota and retirement, self-service users can also be governed by chargeback (showback) plans that lend measurability to the consumption. Enterprise Manager 12c comes with a rich set of chargeback metrics to build chargeback plans on. The charge plans can be based on resource utilization or configuration or a combination of both. For example, one can charge more for a higher version of the database in conjunction with CPU consumption. The metered usages can be rolled up to an organizational level using the hierarchies defined in an LDAP and reported for budgeting purposes. The various metrics and configuration items that are relevant to Database as a Service are shown below:
Delivering Database as a Service (DBaaS) using Oracle Enterprise Manager 12c

### Figure 11: Chargeback / Showback configuration Options

The chargeback results can be viewed from the Enterprise Manager as well as with BI Publisher. The reports provide tailored views for the self-service user, the Database Administrator and the LOB owners. The hourly metering information can also be exported using EMCLI and fed into a billing or accounting system for chargeback.

Cloud API and Orchestration

Self-service Portal interface is the single window to the world beyond of Cloud computing for an end user. As we continue to observe the trend of private cloud adoption there lays also an inherent need to be able to integrate this great offering with other 3rd party and custom-built orchestration tools. The

<table>
<thead>
<tr>
<th>Database Instance</th>
<th>Shared Database (Service)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edition</td>
<td>Option</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>Edition</td>
</tr>
<tr>
<td>Option</td>
<td>Release</td>
</tr>
<tr>
<td>Storage Usage</td>
<td>RAC Node Count</td>
</tr>
<tr>
<td>Release</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>RAC Node Count</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time</td>
<td>CPU Time Per Service</td>
</tr>
<tr>
<td>CPU Utilization (%)</td>
<td>CPU Utilization Per Service</td>
</tr>
<tr>
<td>Network</td>
<td>Disk Read (Physical) Operations Per Service</td>
</tr>
<tr>
<td>Disk Read (Physical) Operations</td>
<td>Disk Write (Physical) Operations Per Service</td>
</tr>
<tr>
<td>SQL EXECs</td>
<td>SQL EXECs Per Service</td>
</tr>
<tr>
<td>User Transactions</td>
<td>User Transactions Per Service</td>
</tr>
</tbody>
</table>

Figure 3: Chargeback reports for DBA, self-service user and LOB owners, respectively (from left)
solution for that is - **Cloud APIs** are a set of RESTful APIs using the JSON payloads and operating on Enterprise Managers cloud resource model to drive the cloud operations.

Enterprise Manager 12c also comes with its own orchestration framework where multi-layered applications can be provisioned using Blueprints. The blueprints operate on the Cloud APIs to deploy service instances and stitch them in the form of an application. This is how DBaaS can be tied to a higher order Platform-as-a-Service (PaaS) solution.
Conclusion

So, who benefits here? The answer is both Users and IT. Users get what they want—fast provisioning, metered usage and guaranteed performance. IT benefits from standardization and automation and can now focus on innovation.

Enterprise Manager 12c offers by-far the most complete and comprehensive Database as a Service functionality in the industry. The features are complemented by database lifecycle management features like configuration management, performance management, patch automation, etc which make the solution complete from a DBaaS administrator’s perspective as well. As we saw above, Enterprise Manager 12c covers all the major use cases for DBaaS, which yield significant business benefits and high ROI.

The rich cloud management features not only lends maturity to any private cloud design of an enterprise it also lends the power of "copy exact" i.e. bringing the error free assembly line production and standardization into a hitherto majorly human driven process. Enterprise Manager 12c determines the most efficient way to utilize hardware computing resources after transparently considering many factors related to the resources referenced and the conditions specified in any self-service request. This automated setup of service catalog, intelligent placement of software resources, single click provisioning and cloning, followed by rich metering and chargeback go a long way in providing a predictive model around the on-demand consumption pattern of databases in an enterprise class, private cloud environment.
Delivering Database as a Service (DBaaS) Using Enterprise Manager 12c Cloud Control

White Paper
March, 2013

Author: Subhadeep Sengupta
Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.
Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Copyright © 2011, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 1010

Hardware and Software, Engineered to Work Together