CONTENTS

patchVantage API Scripts

DBaaS

1.1 Introduction

The technology stack consists of an ORDS web interface running on top of an Oracle Database. It is designed to manage large numbers of Linux Servers, Databases and Applications. The solution falls into the category of Database as a Service (DBaaS).

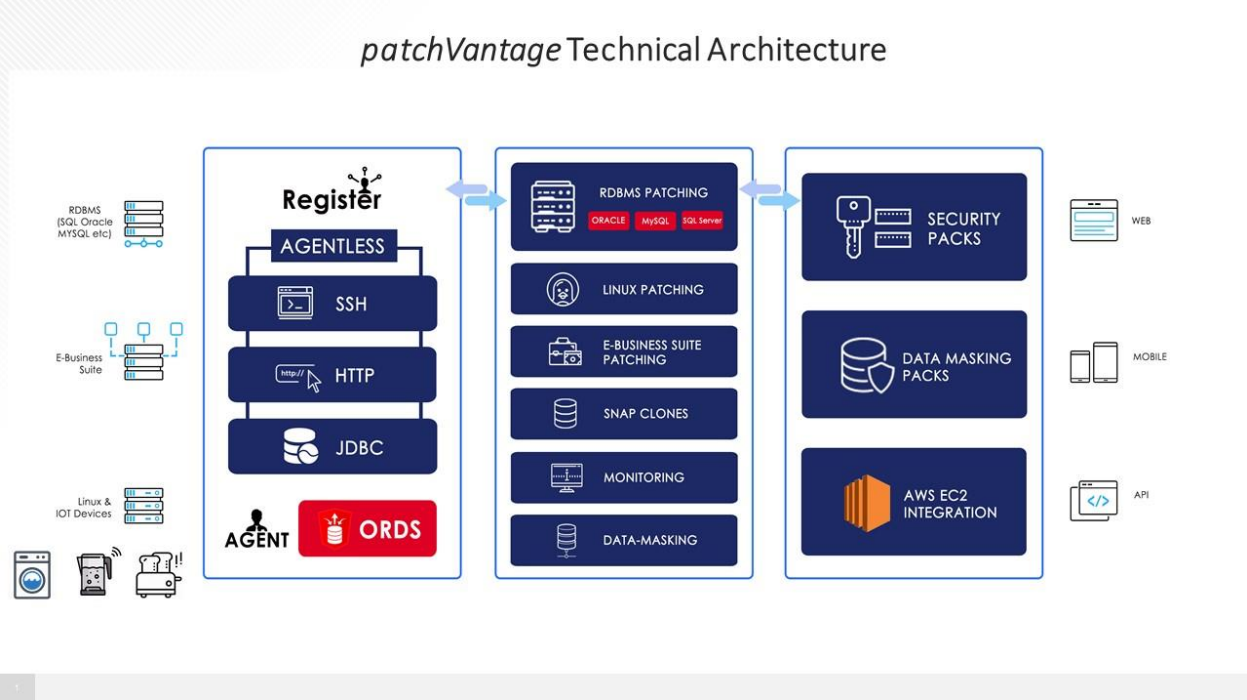


Figure 1 Layered Architecture of Product

Patching and Upgrades are part of any software maintenance lifecycle.

- *Fix Bugs*
- *Add New Functionality*
- *Resolve Security Vulnerabilities*
- *Resolve Performance Issues*

1.2 Key Benefits

Some of the reasons for using the solution are listed below.

- **Lower Costs** - The patch cycle for Oracle, SQL Server and MySQL Databases can be completely automated which means things like the latest security patches can be delivered across the Enterprise with minimal disruption and cost.
- **Less Downtime** – The solution can patch multiple databases simultaneously with consist outcomes
- **No Skills Gap** – Operators with very limited experience of databases can use the product
- **Least Privilege** – Using automation means less people with privileged access, reducing risk.
- **Easy Security Patching** – The latest security bundles are uploaded to the product and ready to use.
- **Upgrades** – Deliver pre-tested complex upgrades which are a hybrid of patches and scripts
- **Visibility** – Always have the latest patch information available and never miss a key patch
- **Minimal Setup** – Using the Cloud Deployment just involves installing an agent to start patching
- **Collaboration** – Successful patching usually involves multiple patching and other scripts being packaged together. These *recipes* can be created, shared and applied all within the product.

1.3 Overview of Automated Oracle Database Patching

The patch cycle for Oracle databases can be completely automated which means the latest security patches can be delivered across the enterprise with minimal disruption, cost and Oracle skills. There are 3 ways to apply patches:

- **Individually or in Groups** – Patches are matched to the specific version of the Database and can be applied individually using the web interface. Alternatively, groups of related patches can be applied as complete sets, useful for complex upgrades and security patching.

- **Reference Model** – This approach is used to synchronize the patch level of one database with many others, quickly and with limited skills. It's useful when you maintain a copy of a Test database that contains all the patches applied and has been verified to work. You can then rollout the patches to many other compatible systems in a few clicks.

  This is possible because each time the product applies a patch to one database it maintains a record of whether this patch is MISSING or APPLIED relative to all the other databases.

- **API** – This takes a *black box* approach and is useful when you need to synchronize patching or control from another system. It consists of a python CLI which instructs the servers using RESTful web services. All the operations described above can be completed using the API.

1.4 Oracle RDBMS Patching Cycle

For each patch the following activities take place {Consists of a job set being executed.}

Table 1 The process required to patch and Oracle Database

Action	Comment
Stop Database	Required for most patches unless ONLINE is applicable
Download Patch	Automatically Download patch if it does not exist in Repository
Upload Patch to Server	Automatically Uploads patch to server if it does not exist
Perform RDBMS Pre-requisite	Check Space Requirements, Compatibility with other patches etc.
Upgrade OPatch	option to automatically upgrade OPatch based in prerequisite result
Apply Patch	Apply the Patch
Start Database	Start the Database in order to complete DataPatch
Post Patching	DataPatch SQL Application
Update Patch History	The product maintains accurate up-to-date patch history
E-mail results	Send logfiles to administrator

1.5 Patch Storage and Download

Oracle patches are maintained in the product providing useful information such as the README and other useful attributes of the patch. There are 3 ways to obtain patches:

Simply Specify Number – Since the patch is used in the context of a database the patch is downloaded from Oracle using the customers MOS credentials automatically. This is the default setting

Drag and Drop – Existing patches can be uploaded using a simple drag and drop interface

Advanced Selection – The DBA specifies OS along with patch number and manually acquires the patch. This is a decent option when the README needs to be examined.

1.6 Security Patching

We maintain the meta-data for Oracle Security Patches (CPU) for most versions of the Oracle Database.

As previously described patches can also be stored in groups and this mirrors the typical needs of security patches which come in multiples. In addition, other activities such as JDK upgrades can also be accommodated because scripts and other *elements* can be included. The result is a comprehensive set of actions that ensures security compliance.

In addition to applying patches the product can also roll them back. This is relevant because new patches sometimes conflict with old ones. Any security patch solution will be tested against environments with the same configurations as yours and include all the steps to ensure a successful patching operation.

1.7 Setup

The demo configuration on AWS is described below

For RDBMS patching it consist of three instances CRM, SALES and HR.

The product can patch databases anywhere: on-premise or the Cloud.

The databases can be on separate servers or many running on a large server.

The user can select which databases are to be registered using */etc/oratab.* (for security)

In this scenario each server is also running SQL 2017 Server Linux

In addition to this there is a WebLogic(12.2.1.3.0) and E-Business Suite(12.2.7)

Table 2 Database Configuration Notes

Server	IP	Username	OS	oratab
CRM	52.33.3.76	oracle	CentOS 7	CRM:/opt/oracle/product/18c/dbhome_1:Y
SALES	52.38.249.26	oracle	CentOS 7	SALES:/opt/oracle/product/18c/dbhome_1:Y
HR	52.32.173.153	oracle	CentOS 7	HR:/opt/oracle/product/18c/dbhome_1:Y

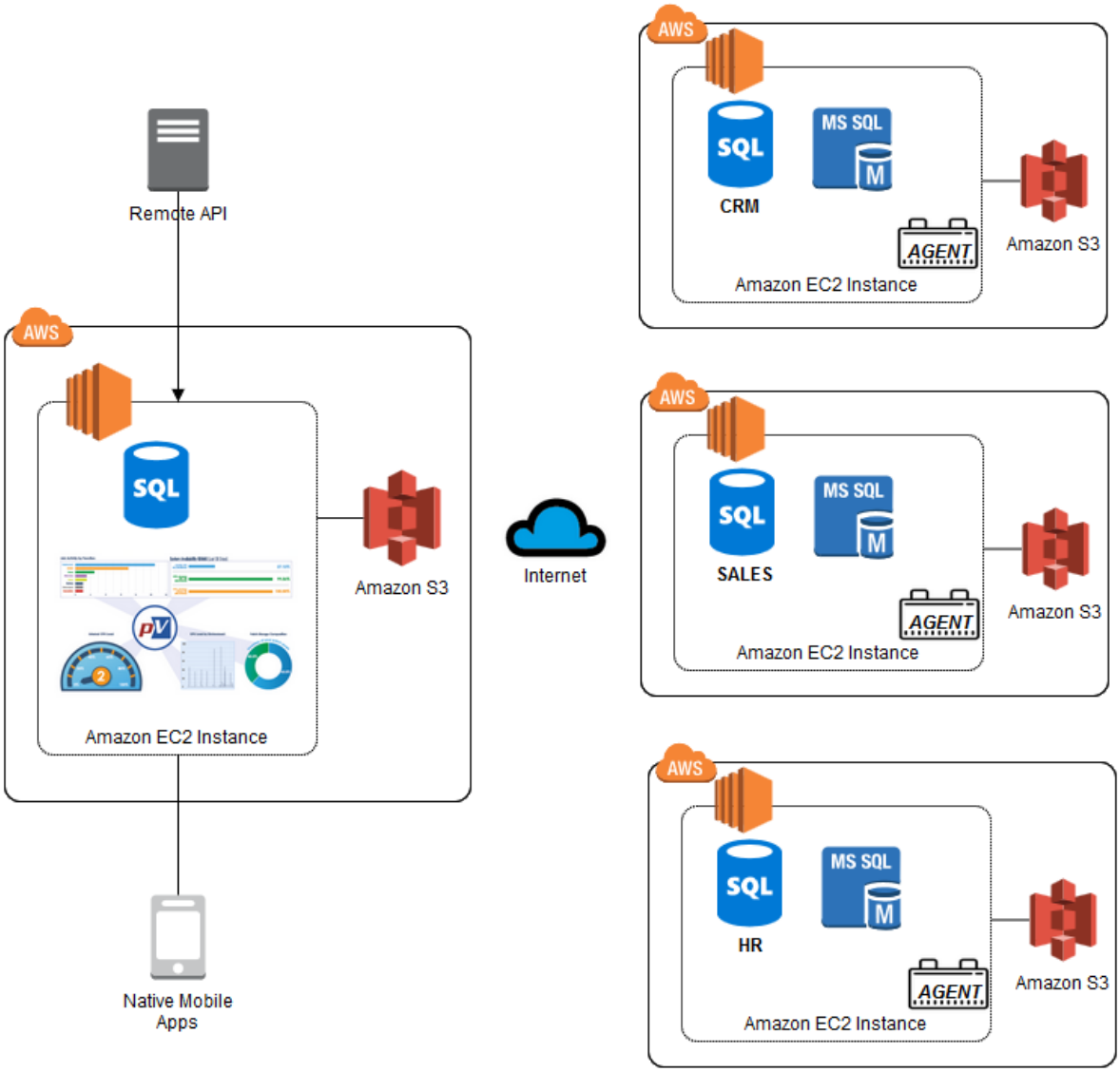


Figure 1 Configuration of Servers and Databases

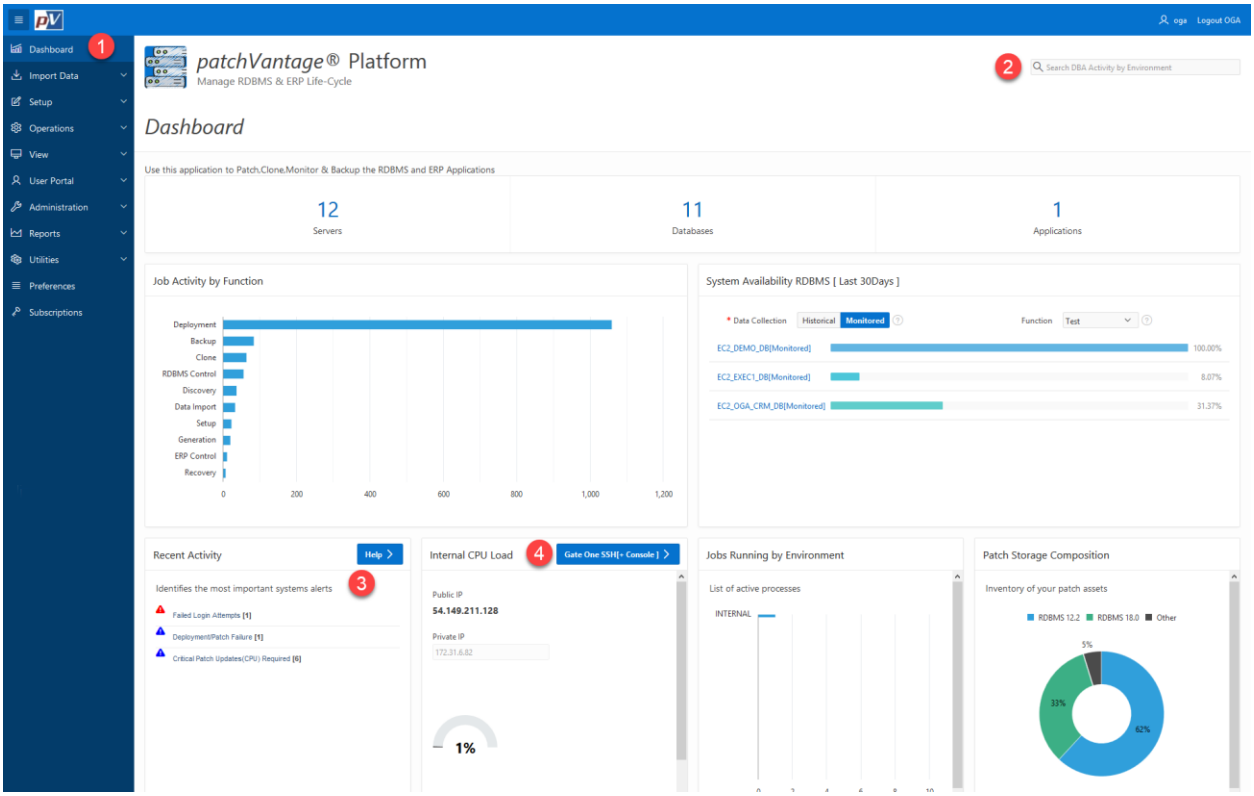
1.8 Getting Started

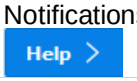
Login to the product using the URL provided. (You may also use Social Sign on)

Username: user
Password: ****

(* request additional logins and start patching from the Cloud : support@patchvantage.zendesk.com)

A dashboard will appear as displayed below:

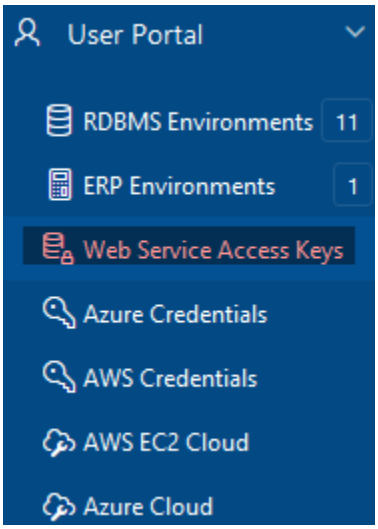


Item	Description
①	**Main Menu:** The major menu components are Dashboard, Import Data, Setup, Operations, View, User Portal, Administration, Reports and Utilities. In this document the focus will on *Administration* and *Operations*
②	**Search:** Quickly find any kind content using the search facility
③	**Alerts:** Important system alerts will appear in this box such as Oracle Alert File Notifications, invalid logins etc. Press the HELP button for more information 
④	**Console and SSH Login:** You can login to the server or ***any*** Linux/UNIX server from the web page without the need for Putty(Here)

1.9 Installing the Client

The client will be used to connect into the product and execute commands from the command line

Download the client using the navigation path **Security Credentials -> Web Service Access Keys**



Click on the button **Download Client[Python]**

pv.ws.zip

Alternatively use the file attached ->

You need to install the file on your Linux server (or Windows with Python) After transferring the file unzip: **unzip pv.ws.zip and Change Directory(cd)** to the *ws* directory

You can examine the README.

sudo ./installer -i /usr/local/pv -b /usr/local/bin/pv

In your **.bash_profile add the following lines:**

export PV_CLIENT_ID=**********
export PV_CLIENT_KEY=**********

Next Install the bash utility jq : **sudo yum install jq -y**

Finally test connectivity to the server : **pv ping-main**

It should reply with the latency. e.g. 0.0883769989014

Make sure your keys are accurate : **pv get-uid**

It should reply with the following e.g. user_valid-dba-user

1.10 Overview of Scripts

Scripts have been created which provide practical examples of how the API can be deployed inside your organization and should accelerate learning in conjunction with the API Reference Guide. Generally, the API returns JSON which can be parsed using the jq utility and the scripts utilizes this .

They scripts have no dependencies and will automatically start the AWS Servers and required databases

Backup – *backup.sh*

This is a hot or cold backup of an Oracle Database using RMAN. (see *Backup Reference Guide*)

The database is automatically shut down and restart for Cold Backups

An option exists to send the backup to AWS(S3) or Azure(Blob) storage

The Amazon server is automatically started

Previous Backups are removed

Confirmation enough disk space exists

Backup is taken using RMAN

A report is produced showing backups undertaken by user=xerox

Security Patching – *security.sh*

The API determines the latest security patches for Database

The Amazon server is automatically started

Database is Stopped

Patches Applied (forced)

Database is Started

DataPatch is Executed

E-Business Suite Patching – *ebs.sh*

The API determines the latest security patches for E-Business Suite 12.2.7

The Amazon server is automatically started

The Database associated with the Application is collected

The Database is Started

ADOP patching prepare, apply, finalize, cutover and cleanup is executed

DataPatch is Executed

The Amazon server is automatically stopped

SNAP(rapid) Cloning – *clone.sh*

Previous SNAP clones are de-allocated

Memory checks performed to ensure success (using SGA Size and System Memory)

SNAP Clone is executed in minutes (fractional amounts of space is used)

Email and SMS Alert sent according to Blackout List and users Time Zone

1.11 Execution

The scripts can are ready to execute and apply many of the API's in the reference manual

When they are executed, they will create Jobs on the Server which examined using

View->Jobs

=> sh backup.sh

```
[oracle@ip-172-31-22-213 xerox]$ sh backup.sh
patchvantage.ai response time :0.0959610939026(s)
patchvantage.ai Load: LIGHT
start Server on Amazon instance-type=t2.medium VCPU=2 RAM=4Gb[Wait on Completion]
{"aws": [{"'ALLINSTANCESRUNNING'": " All Instances Running"}]}
Analyze Filesystems....
Latest Backup for ip-172-31-5-197.313.SALES was created on: 15-SEP-2019 23 00
Removing previous Backup PV_RMAN_FULL.SALES-2221
Backup was Removed
Backup Database ip-172-31-5-197.313.SALES to Filesystem /u01[No S3 Bucket]
backup_full_environment/ip-172-31-5-197.313.SALES/cold/@u01/no/no/0/yes { "backup_database":[ {
 "JOB_NAME":"PV6918" ,"DESCRIPTION":"Backup RDBMS Environment[REST]" ,"ENVIRONMENT":"ip-172-31-
5-197.313.SALES" ,"STATUS":"SUCCEEDED" } ] }
PV_RMAN_FULL.HR-2041    2019-09-13T06:13:34Z
PV_RMAN_FULL.SALES-2241 2019-09-17T01:07:19Z
stop Server on Amazon instance-type=t2.medium VCPU=2 RAM=4Gb[Wait on Completion]
{"aws": [{"'ALLINSTANCESSTOPPED'": " All Instances Stopped"}]}
```

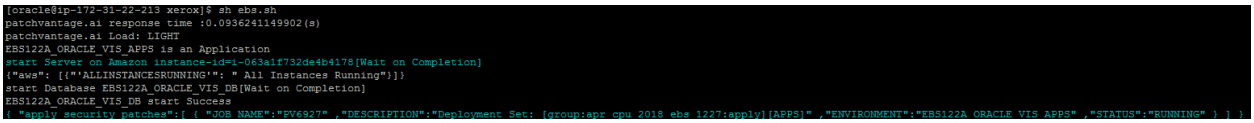
☐	Job Name	Status	Environment	Description	Message
☐	PV6919	SUCCEEDED	NONE	Stop | Start EC2 Instances	AWS EC2 Instances [stop] Success
☐	PV6918	SUCCEEDED	ip-172-31-5-197.313.SALES	Backup RDBMS Environment[REST]	{FULL} Cold Backup Success [RDBMS Started]
☐	PV6916	SUCCEEDED	NONE	Stop | Start EC2 Instances	AWS EC2 Instances [start] Success

=>sh security.sh

```
[oracle@ip-172-31-22-213 xerox]$ sh security.sh
patchvantage.ai response time :0.0938811302185(s)
patchvantage.ai Load: LIGHT
ip-172-31-0-45.325.CRM is a Database
start Server on Amazon instance-type=t2.medium VCPU=2 RAM=4Gb[Wait on Completion]
{"aws": [{"'ALLINSTANCESRUNNING'": " All Instances Running"}]}
{ "apply_security_patches":[ { "JOB_NAME":"PV6921" ,"DESCRIPTION":"Deployment Set: [group:apr_cpu_2019_rdbms_18:apply][DB]" ,"ENVIRONMENT":"ip-172-31-0-45.325.CRM" ,"STATUS":"RUNNING" } ] }
```

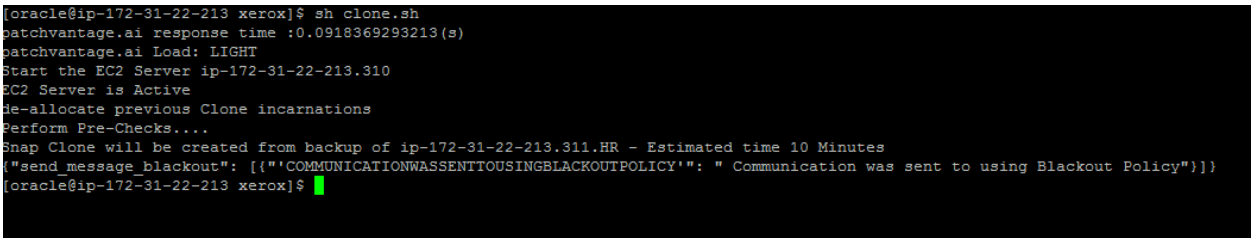
☐	Job Name	Status	Environment	Description	Message
☐	PV6922	SUCCEEDED	ip-172-31-0-45.325.CRM	Deploy: Solution Execution Post Step(DB)	DataPatch Execution Success
☐	PV6922	SUCCEEDED	ip-172-31-0-45.325.CRM	db_control: Start Database	Database was started
☐	PV6924	SUCCEEDED	ip-172-31-0-45.325.CRM	Gather Patch History RDBMS	156 Database Patch Records Processed
☐	PV6922	SUCCEEDED	ip-172-31-0-45.325.CRM	Deploy: Solution Execution(DB)	opatch **apply** [29024028] WARNING : Patch Already Applied
☐	PV6922	SUCCEEDED	ip-172-31-0-45.325.CRM	db_control: Stop Database	Database was stopped
☐	PV6921	SUCCEEDED	ip-172-31-0-45.325.CRM	Deployment Set: [group:apr_cpu_2019_rdbms_18:apply][DB]	*Deployment Set for ip-172-31-0-45.325.CRM is Active

=>sh ebs.sh



☐	PV6928	SUCCEEDED	EBS122A_ORACLE_VIS_APPS	Deploy: Solution Execution(APPS)	adop **cleanup** [27468058] execution was a Success
☐	PV6933	SUCCEEDED	NONE	Stop | Start EC2 Instances	AWS EC2 Instances [start] Success
☐	PV6932	SUCCEEDED	NONE	Stop | Start EC2 Instances	AWS EC2 Instances [stop] Success
☐	PV6928	SUCCEEDED	EBS122A_ORACLE_VIS_APPS	Deploy: Solution Execution(APPS)	adop **cutover** [27468058] execution was a Success
☐	PV6928	SUCCEEDED	EBS122A_ORACLE_VIS_APPS	Deploy: Solution Execution(APPS)	adop **finalize** [27468058] execution was a Success
☐	PV6928	SUCCEEDED	EBS122A_ORACLE_VIS_APPS	Deploy: Solution Execution(APPS)	adop **apply** [27468058] execution was a Success

=> sh clone.sh



☐	PV6952	SUCCEEDED	SNAP	Generate RDBMS Thin Clone[REST]	snap1441 {SNAP-Shot} Environment is AVAILABLE

1.12 Summary

Any database management task can now be automated using our API's – whether on-premise or the Cloud.

The practical examples included here are intended to encourage deployment and understanding.