How to avoid boring work - Automation for DBAs

Marcin Przepiorowski
Delphix
Ireland

Keywords:

Automation, Ansible, Oracle Enterprise Manager

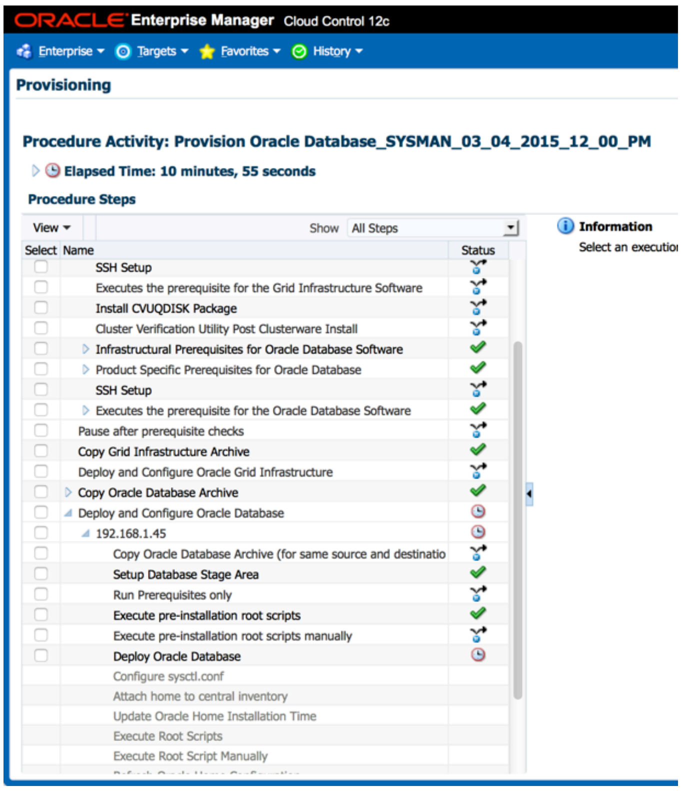
Introduction

If you are maintaining a fleet of servers or many different customers, there is a list of tasks you have to do daily/weekly/monthly/etc. There are two possible solutions to that problem - you can spend your time doing to manually or you can spend your time to build an automated tool to help you with reoccurring task and after initial time/money effort you will have more time to learn new features or deal with problem you are more interested in. This presentation will show you how Ansible toolkit can be used in daily DBA work and how it will allow you to keep all environments configured in same way. In addition to that automated processes have a better quality (over a time) as if you fix your script this fix will be always there and you should not do same misconfiguration again. I will compare an Ansible with Enterprise Manager 12c, as well as to show in which area you should use you which tool. The following tasks will be covered:
- Automated installation
- Automated patching
- Scripts

Oracle Enterprise Manager

Oracle Enterprise Manager 12c has a set of features, which allow DBA to automate a daily tasks.
In my presentation I will cover the following ones:
- Provisioning
- Patching
- Jobs

The pictures below are showing screens from provisioning process, which allow DBA to quickly create a new database on the selected target.



Provisioning

Procedure Activity: Provision Oracle Database_SYSMAN_03_04_2015_12_00_PM

Elapsed Time: 10 minutes, 55 seconds

Procedure Steps

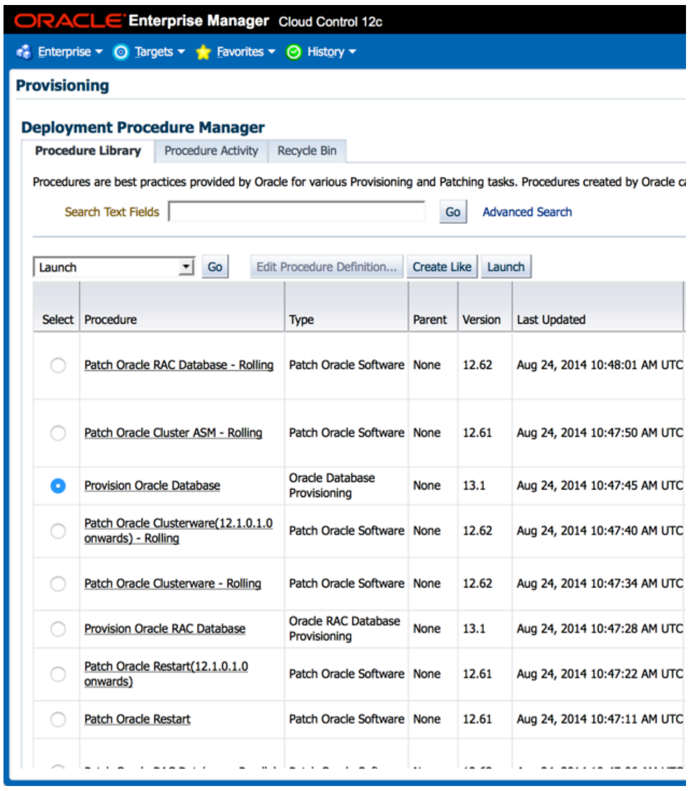
View Show All Steps

Select	Name	Status
☐	SSH Setup	↷
☐	Executes the prerequisite for the Grid Infrastructure Software	↷
☐	Install CVUQDISK Package	↷
☐	Cluster Verification Utility Post Clusterware Install	↷
☐	Infrastructural Prerequisites for Oracle Database Software	✓
☐	Product Specific Prerequisites for Oracle Database	✓
☐	SSH Setup	↷
☐	Executes the prerequisite for the Oracle Database Software	✓
☐	Pause after prerequisite checks	↷
☐	Copy Grid Infrastructure Archive	✓
☐	Deploy and Configure Oracle Grid Infrastructure	↷
☐	Copy Oracle Database Archive	✓
☐	Deploy and Configure Oracle Database	🕒
☐	192.168.1.45	🕒
☐	Copy Oracle Database Archive (for same source and destinatio	↷
☐	Setup Database Stage Area	✓
☐	Run Prerequisites only	↷
☐	Execute pre-installation root scripts	✓
☐	Execute pre-installation root scripts manually	↷
☐	Deploy Oracle Database	🕒
	Configure sysctl.conf	
	Attach home to central inventory	
	Update Oracle Home Installation Time	
	Execute Root Scripts	
	Execute Root Script Manually	

Information

Select an executio

Illustration. 1: Provision an Oracle Database



Provisioning

Deployment Procedure Manager

Procedure Library | Procedure Activity | Recycle Bin

Procedures are best practices provided by Oracle for various Provisioning and Patching tasks. Procedures created by Oracle c

Search Text Fields [ ] Go Advanced Search

Launch Go Edit Procedure Definition... Create Like Launch

Select	Procedure	Type	Parent	Version	Last Updated
○	Patch Oracle RAC Database - Rolling	Patch Oracle Software	None	12.62	Aug 24, 2014 10:48:01 AM UTC
○	Patch Oracle Cluster ASM - Rolling	Patch Oracle Software	None	12.61	Aug 24, 2014 10:47:50 AM UTC
●	Provision Oracle Database	Oracle Database Provisioning	None	13.1	Aug 24, 2014 10:47:45 AM UTC
○	Patch Oracle Clusterware(12.1.0.1.0 onwards) - Rolling	Patch Oracle Software	None	12.62	Aug 24, 2014 10:47:40 AM UTC
○	Patch Oracle Clusterware - Rolling	Patch Oracle Software	None	12.62	Aug 24, 2014 10:47:34 AM UTC
○	Provision Oracle RAC Database	Oracle RAC Database Provisioning	None	13.1	Aug 24, 2014 10:47:28 AM UTC
○	Patch Oracle Restart(12.1.0.1.0 onwards)	Patch Oracle Software	None	12.61	Aug 24, 2014 10:47:22 AM UTC
○	Patch Oracle Restart	Patch Oracle Software	None	12.61	Aug 24, 2014 10:47:11 AM UTC

Illustration. 2: Provision an Oracle Database – final screen

The biggest advantage of the OEM 12c is a introduction of the command line interface. One of my favourite example enabling or disabling jobs for a particular environment. If there is a one job or two jobs per environment, there is no problem but what if there are 20 or more jobs? Doing that using a GUI interface is a long process and it can be done with a two command line commands:

```
emcli login -username=sysman
emcli suspend_job -name=<pattern>
```

Ansible

There are couple of automation tools, which can be used for that task like Puppet, Chef or Ansible to name a few. The latter one - Ansible - is my favorite, cause in my opinion it has shortest learning curve and also doesn't require any agents on the remote servers.
Although there are some requirements which has to be met on the remote servers:
- ssh connectivity to the remote host
- Python 2.4 (with python-simplejson) or 2.5 installed on the remote host.

If Python is a "big no-no" in your organization, you can still you Ansible to help you run a script over set of servers but you will loose almost all functionality.

How Ansible works:

Ansible has a concept of the control machine and remote managed servers. Ansible scripts are started on the control machine and other servers are managed over a SSH connection and Ansible modules send to the remote servers on demand. Ansible can run in the fully automated mode only if control machine and remote servers has a password-less SSH configuration. In the other case Ansible can ask for a SSH password.

The control machine can be a dedicated sever or it can be an OS admin or DBA laptop, where Ansible is installed. Please check Ansible documentation to find information how to install it.

Remote servers can be managed from control machine, only if they are added into a Ansible inventory. The inventory is a text file with the following format:

```
testdb.mycompany.com

[testgroup]
envtest ansible_ssh_host=172.16.180.190 ansible_ssh_user=oracle

[proddb-group]
rac1.mycompany.com
rac2.mycompany.com
```

There are two groups in inventory file - testdbgroup with one members envtest, and proddb-group with two members rac1 and rac2. Server - testdb - is listed in the inventory file but it not a part of any group.

Ansible has a concept of host and group variables. Group variables are used for all host defined in the a group. There is a directory called group_vars, containing a files named with a group name. Each file is a set of variables for a particular group. Host variables are used for a particular host only and they overwrite a group variables. There is a directory called host_vars, containing a files named with a host name. Each file is a set of variables for a particular host.

```
$ ls -l host_vars/
total 24
-rw-r--r--  1 mprzepiorowski  staff  29 Sep 29 12:44 envtest

$ cat host_vars/envtest
password: "host env password"

$ ls -l group_vars/
total 8
-rw-r--r--  1 mprzepiorowski  staff  23 Sep 29 13:10 testgroup

$ cat group_vars/testgroup
groupname: "Test group"
```

Server envtest has a file with host variables and it belongs to the group testgroup with a defined variables as well. Both will be used in the following example.

There are three ways of running a remote command:

- single command mode,
- Ansible playbooks
- Ansible playbooks with roles

and commands or playbooks can be run on the specific host from inventory file, specific group or all hosts.

Examples:

Ansible playbook:

Next example will show how to create a simple playbook with an one task and install a vim package using yum

Playbook file is formatted in YAML and it looks like this:

```
---
- hosts: all
  tasks:
     - name: install vim
       sudo: yes
       yum: name=vim state=present
```

Be aware that YAML file syntax include a white space, so format of this file matter as well.

Line description:
Hosts: all - mean run a playbook for all hosts from inventory file
tasks: - is a start of task lists (one task in this example)
name: - start of task definition
sudo: - run this module using sudo
yum: - module name with parameters (name is a package to install, state=present mean install it)

Ansible playbook can be stated using this command:

```
$ ansible-playbook yum.yml -i inventory/lab

PLAY [all]
********************************************************************

GATHERING FACTS
***************************************************************
ok: [envtest]

TASK: [install vim]
***********************************************************
changed: [envtest]

PLAY RECAP
********************************************************************
envtest              : ok=2    changed=1    unreachable=0    failed=0
```

Where -i is pointing to the inventory file and yum.yml is a name of file with playbook definition. Output contains a list of tasks executed and summary with number of successes, changes and failures.

Ansible playbook and variables:

In the previous examplee, only has host has been used and there was no variables set. Let's add some more complexity to playbook and show how flexible is Ansible. Both variables defined for particular host and group will be used to create a text file based on template.

Template file (Ansible supports Jinja2 template language)

```
$ cat template.j2
Host name is {{ ansible_hostname }}
Host var password is set to {{ password }}
Group var groupname is set to {{ groupname }}
```

Ansible playbook:

```
$ cat template.yml
---
- hosts: all
  tasks:
     - name: generate scripts
       template: src=template.j2 dest=/tmp/output mode=0644
```

Running playbook:

```
$ ansible-playbook -i inventory/lab template.yml

PLAY [all]
********************************************************************

GATHERING FACTS
***************************************************************
ok: [envtest]

TASK: [generate scripts]
******************************************************
changed: [envtest]

PLAY RECAP
********************************************************************
envtest            : ok=2    changed=1    unreachable=0    failed=0
```

Output file on envtest server

```
$ ssh oracle@172.16.180.190
Last login: Tue Sep 29 12:29:59 2015 from 172.16.180.1
[oracle@envtest ~]$ cat /tmp/output
Host name is envtest
Host var password is set to host env password
Group var groupname is set to Test group
[oracle@envtest ~]$
```

Contact address:

Name
Delphix

Email   marcin@delphix.com
Internet: www.delphix.com