

# PROVISIONING THE ORACLE DATABASE IN THE CLOUD

**FRITS HOOGLAND**

# \$(whoami)

Frits Hoogland

Working with Oracle products since 1996

Blog: <http://fritshoogland.wordpress.com>

Twitter: @fritshoogland

Email: [frits.hoogland@accenture.com](mailto:frits.hoogland@accenture.com)

Oracle ACE Director



OakTable Member

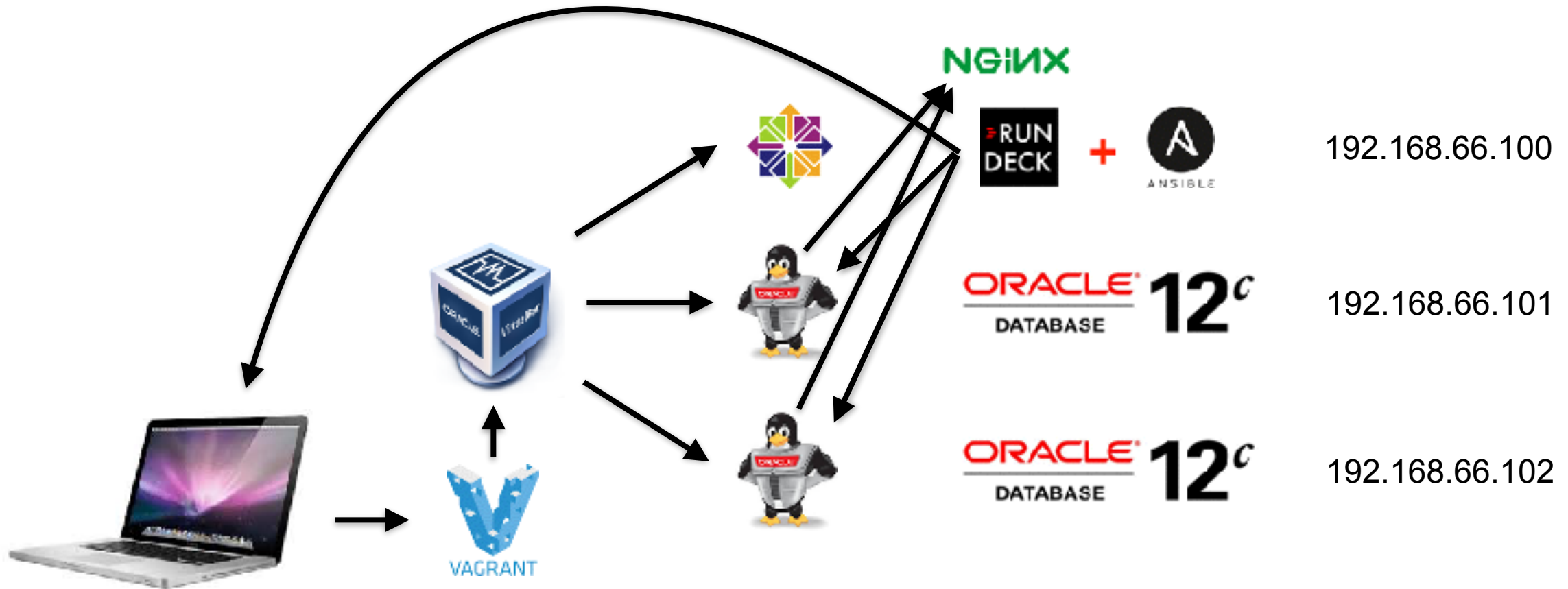


# ABOUT THIS PRESENTATION



- **Versions:**
  - **Database: Oracle version 12.1.0.2 + 170814 & 170718 patch**
  - **Operating system: Oracle Linux 7.3 (database + provisioning host)**
  - **Ansible 2.3.1.0**
  - **~~Semaphore 2.3.0~~ Rundeck 2.9.2**
  - **(vagrant 1.9.7, virtual box 5.1.26)**
- **The intended audience for this talk is technical architects who want to implement orchestration (in the cloud).**
- **This is not an Ansible introduction.**
- **The purpose of this presentation is to show Ansible playbooks, display their readability, and give ideas about usage and usefulness.**
- **This presentation assumes you understand linux and Oracle administration.**

# MY ENVIRONMENT




Q

# ANSIBLE

- I am using Ansible version 2.3.1.0
- Ansible essentially describes the desired state.
  - Ansible is “not really” a programming language.
  - Ansible scripts should be re-runnable.
    - After a task has completed, the next run should result in ‘OK’.

# SEMAPHORE

- **Open source web based user interface for running Ansible.**
    - **Free, open source UI for running Ansible playbooks (not editing).**
    - **Similar product to Ansible Tower, which is a product that requires a license.**
      - **Tower provides more functionality.**
    - **“young” product, rough around the edges.**
  - **General way semaphore works:**
    - **Create a git repository with the Ansible scripts in it.**
    - **Add a web based remote repository like Github or Gitlab to the local git repository.**
    - **Add the web based repository to a project in Semaphore.**
    - **Combine repository, inventory and ssh keys in a template in Semaphore.**
    - **Run the Ansible scripts using the template in Semaphore.**
  - **<https://fritshoogland.wordpress.com/2017/04/29/how-to-install-the-semaphore-ui-for-running-ansible/>**
- accenture**  **operations**

# RUNDECK

- **Open source (+closed source) web based user interface for running tasks.**
  - **Plugin available for running Ansible.**
  - **Somewhat similar product to Ansible Tower, which is a product that requires a license.**
  - **Open source version feels like a “young” product, rough around the edges.**
- **General way rundeck works:**
  - **Create a project, in that project a job and associate it with one or a group of hosts.**
  - **Run a job once or multiple times, now or scheduled.**
- **<https://fritshoogland.wordpress.com/2017/08/09/installation-of-rundeck-with-ansible-plugin-on-centos-7/>**



# RUNDECK / SEMAPHORE / TOWER: WHY?

- **Above mentioned UIs mean you can achieve:**
  - **Accountability.**
  - **Traceability.**
  - **Delegation of tasks.**
  - **Integrate with automation (REST).**
- **Surely you don't want your admins to perform standard tasks all on their own,**
  - **using a random version of corporate scripts,**
  - **or not using corporate scripts,**
  - **or using something somebody produced themselves?**
- **If it is executed standards based and correctly:**
  - **then it's almost guaranteed not formally logged in any way?**
  - **You want standardisation and consistency, right?**
  - **Even if the task was executed perfectly, how could you know?**

# VAGRANT AND VIRTUALBOX

- The cloud provisioning examples are executed on virtual box VMs on my laptop.
  - Vagrant is a way to quickly create a VM *on your local laptop*.
  - Ideal for creating a test case or specific setup.
- To setup vagrant with virtualbox, all you need to do is install both free products.
  - You might want to modify the default VM location of virtualbox.
- If you want to (mass) provision cloud instances, have a look at terraform.
  - <https://www.terraform.io>
  - All popular cloud providers are supported.
  - Plugin for the Oracle clouds exists.
- Ansible can only work with a workaround on Windows as ‘Control Machine’.
  - Getting Ansible to work as provisioner for Vagrant has been cumbersome.

# VAGRANT

- This is how my provisioning server (called 'rundeck') is created:

```
$ mkdir rundeck-v; cd $_  
$ vi Vagrantfile  
$ vagrant up
```

```
Vagrant.configure("2") do |config|  
  config.vm.box = "centos/7"  
  config.ssh.insert_key = false  
  config.ssh.private_key_path = [ "~/.vagrant_ssh/id_dsa", "~/.vagrant.d/  
insecure_private_key" ]  
  config.vm.hostname = "rundeck.local"  
  config.vm.network "private_network", ip: "192.168.66.100"  
  config.vm.provider "virtualbox" do |vb|  
    vb.name = "rundeck"  
    vb.memory = "1024"  
    vb.cpus = "1"  
    # PERL L4 CACHE WORK AROUND  
    if !File.exist?("u01_disk.vdi")  
      vb.customize [ 'createhd', '--filename', "u01_disk.vdi", '--size', 40960 ]  
    end  
    vb.customize [ 'storageattach', :id, '--storagectl', 'SATA Controller', '--  
port', 2, '--device', 0, '--type', 'hdd', '--medium', "u01_disk.vdi" ]  
  end  
end
```

# VAGRANT

```
$ mkdir rundeckmm-v; cd $_  
$ vi Vagrantfile  
$ vagrant up
```

```
servers=[{:hostname=>"rundeckmm1.local",:name=>"rundeckmm1",:ip => "192.168.66.101" },  
         {:hostname=>"rundeckmm2.local",:name=>"rundeckmm2",:ip => "192.168.66.102" }]  
Vagrant.configure("2") do |config|  
  servers.each do |machine|  
    config.vm.define machine[:hostname] do |node|  
      node.vm.box = "box-cutter/ol73"  
      node.vm.hostname = machine[:hostname]  
      node.ssh.insert_key = false  
      node.ssh.private_key_path = [ "~/.vagrant_ssh/id_dsa", "~/.vagrant.d/  
insecure_private_key" ]  
      node.vm.network "private_network", ip: machine[:ip]  
      node.vm.provider "virtualbox" do |vb|  
        vb.name = machine[:name]  
        vb.memory = "1024"  
        vb.cpus = "1"  
        # PERL L4 CACHE WORK AROUND  
        if !File.exist?("u01_disk-#{machine[:name]}.vdi")  
vb.customize [ 'createhd', '--filename', "u01_disk-#{machine[:name]}.vdi", '--size',  
40960 ]  
          end  
vb.customize [ 'storageattach', :id, '--storagectl', 'SATA Controller', '--port', 2,  
'--device', 0, '--type', 'hdd', '--medium', "u01_disk-#{machine[:name]}.vdi" ]  
          end  
        end  
      end  
    end  
  end  
end
```

# VAGRANT

- Vagrant uses a default username ('vagrant') for which a certificate is set for authentication.
  - This is the way cloud instances do provide access too.
  - Because vagrant is for local testing, the password of vagrant is made easy on purpose.
- The default user (group actually) is provided all rights via sudo (/etc/sudoers.d/vagrant):  
`%vagrant ALL=(ALL) NOPASSWD: ALL`
- Oracle bare metal cloud (/etc/sudoers.d/90-cloud-init-users):  
`opc ALL=(ALL) NOPASSWD:ALL`

# CREATING AN IN-HOUSE RPM/YUM REPO

- **nginx for serving the repository:**

- **yum install nginx**
- **/etc/nginx/nginx.conf:**

```
server { listen *:80;
        root /repo;
        location / { autoindex on; }
}
```

- **create repository:**

- **yum install createrepo yum-utils**
- **cd /repo/your\_repo\_name**
- **reposync -r repository\_definition\_name .**
- **createrepo --update .**
- **selinux: chcon -Rt httpd\_sys\_content\_t /repo**

- **[https://gitlab.com/FritsHoogland/setup\\_repo.git](https://gitlab.com/FritsHoogland/setup_repo.git)**

# HOW IT'S DONE: LINUX SETUP

Playbook: `/var/rundeck/projects/Provisioning_oracle_presentation/git/provisioning-presentation/deploy.yml`

- Ansible playbook contains multiple 'plays'.
- 'Plays' are run by specifying their 'tag':

Extra Ansible arguments: `--tags=linux_setup`

```
- name: setup linux
  hosts: all
  become: true
  vars:
    current_software_repo: ol7u3
    current_kernel_repo: ol7_uek4
    repo_host: 192.168.66.100
    kernel: kernel-uek-4.1.12-61.1.28.el7uek
  tags: linux_setup
  roles:
    - repository_handling
    - setup_kernel
    - setup_linux_for_oracle121
    - setup_u01
```

# LINUX SETUP: REPOSITORY\_HANDLING

```
---  
- name: disable all yum repositories  
  replace:  
    dest: /etc/yum.repos.d/public-yum-ol7.repo  
    regexp: '^enabled.*=.*1$'  
    replace: 'enabled=0'  
  
- name: set current software yum repositories  
  yum_repository:  
    name: '{{ item }}'  
    description: '{{ item }}'  
    baseurl: 'http://{{ repo_host }}/{{ item }}/'  
    gpgkey: 'file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle'  
  with_items:  
    - '{{ current_software_repo }}'  
    - '{{ current_kernel_repo }}'
```



# LINUX SETUP: SETUP\_KERNEL (1/2)

---

- name: install kernel

  yum:

    name: "{{ kernel }}"

    state: present

- name: set the desired kernel as default kernel

  shell: >

```
    /sbin/grubby --set-default=/boot/vmlinuz-{{ kernel | regex_replace('^kernel-(.*)$', '\1') |  
regex_replace('^uek-(.*)$', '\1') }}.x86_64
```

- name: is the running kernel the desired kernel?

  shell: >

```
    if [ $(uname -r) = $( /sbin/grubby --default-kernel | sed "s/\/boot\/vmlinuz-\/" ) ]; then echo  
"yes"; else echo "no"; fi
```

  register: remain\_running

# LINUX SETUP: SETUP\_KERNEL (2/2)

```
- block:
  - name: reboot to activate kernel
    shell: >
      sleep 2; /sbin/shutdown -r now "Ansible activate new kernel"
    async: 1
    poll: 0

- name: wait for server to come down
  local_action: wait_for
  args:
    host: "{{ inventory_hostname }}"
    port: 22
    state: stopped
    timeout: 600
  become: false

- name: wait for server to come back up
  local_action: wait_for
  args:
    host: "{{ inventory_hostname }}"
    port: 22
    state: started
    timeout: 600
  become: false
  when: "'no' in remain_running.stdout"
```

# LINUX SETUP: SETUP\_LINUX\_FOR\_ORACLE121 (1/2)

---

- name: upgrade all packages

  yum:

    name: "\*"

    state: latest

    exclude: "kernel\*"

- name: install acl, unzip, perl and unixODBC

  yum:

    name: "{{ item }}"

    state: installed

  with\_items:

    - perl

    - unzip

    - unixODBC

    - acl

- name: disable selinux

  selinux:

    state: disabled

# LINUX SETUP: SETUP\_LINUX\_FOR\_ORACLE121 (2/2)

- name: install oracle preinstall package for database  
yum:
  - name: oracle-rdbms-server-12cR1-preinstall.x86\_64
  - state: present
- name: copy public key to authorized\_key file of oracle  
authorized\_key:
  - user: oracle
  - key: "{{ lookup('file', '~/.ssh/id\_rsa.pub') }}"

# LINUX SETUP: SETUP\_U01 (1/2)

---

- name: create volume group vg\_oracle using /dev/sdb
  - lvgs:
    - vg: vg\_oracle
    - pvs: /dev/sdb
- name: create logical volume lv\_oracle in vg\_oracle
  - lvols:
    - vg: vg\_oracle
    - lv: lv\_oracle
    - size: 35g
- name: create filesystem in lv\_oracle
  - filesystems:
    - fstype: xfs
    - dev: /dev/vg\_oracle/lv\_oracle
- name: mount filesystem to /u01
  - mounts:
    - name: /u01
    - src: /dev/vg\_oracle/lv\_oracle
    - state: mounted
    - fstype: xfs

# LINUX SETUP: SETUP\_U01 (2/2)

- name: set ownership correct for /u01 directory

file:

dest: /u01

state: directory

owner: oracle

group: oinstall

# HOW IT'S DONE: ORACLE INSTALLATION AND PATCHING

- Play and tag:

- name: setup and patch oracle 12.1

- hosts: all

- become\_user: oracle

- vars:

- database\_inventory\_name: Ansible\_oh\_12102

- stage\_directory: /u01/stage

- oracle\_home: /u01/app/oracle/product/12.1.0.2/dbhome\_1

- oracle\_base: /u01/app/oracle

- opatch\_version: 12.2.0.1.8

- opatch\_file: p6880880\_121010\_Linux-x86-64-{{ opatch\_version }}.zip

- tags: oracle\_121\_install\_patch

- roles:

- install\_oracle\_12102

- install\_opatch

- patch\_oh\_12102\_to\_psu\_170814

- patch\_oh\_12102\_to\_jvm\_170718

# ORACLE\_121\_INSTALL\_PATCH: INSTALL\_ORACLE (1/3)

```
---
- name: check inventory if database version is already installed
  shell: >
    grep 'HOME NAME="{{ database_inventory_name }}"' /u01/app/oraInventory/ContentsXML/inventory.xml
    removes=/u01/app/oraInventory/ContentsXML/inventory.xml
  register: database_home_installed

- block:

  - name: create directory for installation files
    file:
      dest: "{{ stage_directory }}"
      state: directory
      owner: oracle
      group: oinstall
      become_user: root

  - name: transfer patch file for install
    synchronize:
      src: "files/{{ item }}"
      dest: "{{ stage_directory }}"
    with_items:
      - p21419221_121020_Linux-x86-64_1of10.zip
      - p21419221_121020_Linux-x86-64_2of10.zip
```



# ORACLE\_121\_INSTALL\_PATCH: INSTALL\_ORACLE (2/3)

```
- name: unzip installation media
  unarchive:
    copy: no
    src: "{{ stage_directory }}/{{ item }}"
    dest: "{{ stage_directory }}"
  with_items:
    - p21419221_121020_Linux-x86-64_1of10.zip
    - p21419221_121020_Linux-x86-64_2of10.zip

- name: install oracle database software
  shell: >
    {{ stage_directory }}/database/runInstaller -silent -force -waitforcompletion -ignore prerequisites -
  ignorePrereq oracle.install.option=INSTALL_DB_SWONLY UNIX_GROUP_NAME=oinstall INVENTORY_LOCATION=/u01/
  app/oraInventory ORACLE_HOME={{ oracle_home }} ORACLE_HOME_NAME="Ansible oh 12102"
  ORACLE_BASE={{ oracle_base }} oracle.install.db.InstallEdition=EE oracle.install.db.DBA_GROUP=dba
  oracle.install.db.OPER_GROUP=dba oracle.install.db.BACKUPDBA_GROUP=dba oracle.install.db.DGDBA_GROUP=dba
  oracle.install.db.KMDBA_GROUP=dba DECLINE_SECURITY_UPDATES=true

  register: database_install
  failed_when: "'The installation of Oracle Database 12c was successful.' not in
  database_install.stdout"

- name: run orainstRoot.sh
  shell: >
    /u01/app/oraInventory/orainstRoot.sh
  when: "'/u01/app/oraInventory/orainstRoot.sh' in database_install.stdout"
  become_user: root
```

# ORACLE\_121\_INSTALL\_PATCH: INSTALL\_ORACLE (3/3)

```
- name: run root.sh
  shell: >
    {{ oracle_home }}/root.sh -silent
  become_user: root
```

```
- name: clean up install directory
  file:
    path: "{{ stage_directory }}"
    state: absent
  become_user: root
```

```
when: "database_home_installed.rc != 0 or 'skipped' in database_home_installed.stdout"
```

# ORACLE\_121\_INSTALL\_PATCH: INSTALL\_OPATCH (1/2)

---

- name: create directory for installation files

file:

dest: "{{ stage\_directory }}"

state: directory

owner: oracle

group: oinstall

become\_user: root

- name: transfer patch file for install

synchronize:

src: "files/{{ opatch\_file }}"

dest: "{{ stage\_directory }}"

- name: unzip opatch 12.1

unarchive:

copy: no

src: "{{ stage\_directory }}/p6880880\_121010\_Linux-x86-64-{{ opatch\_version }}.zip"

dest: "{{ stage\_directory }}"

- name: gather 12.1.0 ORACLE\_HOMES from inventory

shell: >

```
grep 12\.1\.0\. /u01/app/oraInventory/ContentsXML/inventory.xml | sed -n "s/.*LOC=\"\(.*\)\\"\ TYPE.*\n/p"
```

register: all\_12\_1\_homes

# ORACLE\_121\_INSTALL\_PATCH: INSTALL\_OPATCH (2/2)

- name: install opatch if necessary

shell: >

```
if [ "$( {{ item }}/OPatch/opatch version | head -1 )" != "$( {{ stage_directory }}/OPatch/opatch
version | head -1 )" ]; then mv {{ item }}/OPatch {{ item }}/OPatch.$(date +%Y%m%d); unzip -oq
{{ stage_directory }}/p6880880_121010_Linux-x86-64-{{ opatch_version }}.zip -d {{ item }}; chown -R
oracle.oinstall {{ item }}/OPatch; fi
```

ignore\_errors: true

with\_items: "{{ all\_12\_1\_homes.stdout\_lines }}"

- name: clean up install directory

file:

path: "{{ stage\_directory }}"

state: absent

become\_user: root

# ORACLE\_121\_INSTALL\_PATCH: PATCH PSU (1/2)

```
---
- name: check patches in the oracle home
  shell: >
    {{ oracle_home }}/OPatch/opatch lspatches
  register: oracle_home_patches

- block:
  - name: create directory for installation files
    file:
      dest: "{{ stage_directory }}"
      state: directory
      owner: oracle
      group: oinstall
      become_user: root

  - name: transfer patch file for install
    synchronize:
      src: "files/{{ psu_patch_file }}"
      dest: "{{ stage_directory }}"

  - name: unzip patch
    unarchive:
      copy: no
      src: "{{ stage_directory }}/{{ psu_patch_file }}"
      dest: "{{ stage_directory }}"
      creates: "{{ stage_directory }}/{{ psu_patch_number }}"
```

```
vars/main.yml:
---
psu_patch_number: 26609783
psu_patch_file: p26609783_121020_Linux-x86-64.zip
psu_patch_success_line: "Composite patch 26609783
successfully applied."
```

# ORACLE\_121\_INSTALL\_PATCH: PATCH PSU (2/2)

- name: apply patch to database home

shell: >

```
{{ oracle_home }}/OPatch/opatch apply -silent
```

```
chdir={{ stage_directory }}/{{ psu_patch_number }}
```

register: apply\_oh\_patch

```
failed_when: "'{{ psu_patch_success_line }}' not in apply_oh_patch.stdout"
```

- name: clean up install directory

file:

```
path: "{{ stage_directory }}"
```

```
state: absent
```

become\_user: root

```
when: "'{{ psu_patch_number }}';' not in oracle_home_patches.stdout"
```

# ORACLE\_121\_INSTALL\_PATCH: PATCH JVM (1/2)

```
---
- name: check patches in the oracle home
  shell: >
    {{ oracle_home }}/OPatch/opatch lspatches
  register: oracle_home_patches

- block:
  - name: create directory for installation files
    file:
      dest: "{{ stage_directory }}"
      state: directory
      owner: oracle
      group: oinstall
      become_user: root

  - name: transfer patch file for install
    synchronize:
      src: "files/{{ jvm_patch_file }}"
      dest: "{{ stage_directory }}"

  - name: unzip patch
    unarchive:
      copy: no
      src: "{{ stage_directory }}/{{ jvm_patch_file }}"
      dest: "{{ stage_directory }}"
      creates: "{{ stage_directory }}/{{ jvm_patch_number }}"
```

```
vars/main.yml:
---
jvm_patch_number: 26027162
jvm_patch_file: p26027162_121020_Linux-x86-64.zip
jvm_patch_success_line: "Patch 26027162
successfully applied."
```

# ORACLE\_121\_INSTALL\_PATCH: PATCH JVM (2/2)

- name: apply patch to database home

shell: >

```
{{ oracle_home }}/OPatch/opatch apply -silent
```

```
chdir={{ stage_directory }}/{{ jvm_patch_number }}
```

register: apply\_oh\_patch

```
failed_when: "'{{ jvm_patch_success_line }}' not in apply_oh_patch.stdout"
```

- name: clean up install directory

file:

```
path: "{{ stage_directory }}"
```

```
state: absent
```

become\_user: root

```
when: "'{{ jvm_patch_number }}';' not in oracle_home_patches.stdout"
```



# CLONE AN ORACLE HOME

- **Another way to create a new Oracle home is to clone it from an existing home.**
  - **Cloning means:**
    - **Creating an archive of an existing Oracle home.**
    - **Extracting the Oracle home from the archive.**
    - **Run clone.pl to:**
      - **Install the extracted Oracle home on the new machine.**
      - **Generate the inventory data.**

# CLONE AN ORACLE HOME

- **The advantages of cloning an Oracle home are:**
  - **Additional patches installed are included automatically.**
  - **Shorter install time, less moving parts in the install procedure.**
- **Disadvantages of cloning an Oracle home:**
  - **Not flexible, you get it all.**
  - **You need to store the archive somewhere.**
  - **You need to update the installation in the archive with a new PSU.**

# HOW IT'S DONE: CREATE CLONE ARCHIVE

- Play and tag:

- name: clone oracle home

- hosts: all

- become: true

- become\_user: oracle

- vars:

- clone\_tarball\_name: oh\_clone\_12102.tgz

- stage\_directory: /u01/stage

- oracle\_home: /u01/app/oracle/product/12.1.0.2/dbhome\_1

- tags: clone\_oracle\_home

- roles:

- clone\_and\_fetch\_oh

# CLONE ORACLE HOME: CLONE\_AND\_FETCH\_OH

---

- name: create directory for clone tarballs
  - file:
    - dest: "{{ stage\_directory }}"
    - state: directory
    - owner: oracle
    - group: oinstall
  
- name: tarball the database home
  - shell: >
    - tar czpf {{ stage\_directory }}/{{ clone\_tarball\_name }} {{ oracle\_home }}
    - creates={{ stage\_directory }}/{{ clone\_tarball\_name }}
  - args:
    - warn: no
  
- name: transfer clone file for install
  - synchronize:
    - src: "{{ stage\_directory }}/{{ clone\_tarball\_name }}"
    - dest: files/
    - mode: pull
    - become: false
  
- name: remove directory for clone tarballs
  - file:
    - dest: "{{ stage\_directory }}"
    - state: absent

# HOW IT'S DONE: ORACLE INSTALLATION VIA CLONE

- Play and tag:

- name: install oracle 12.1

- hosts: all

- become: true

- become\_user: oracle

- vars:

- stage\_directory: /u01/stage

- clone\_tarball\_name: oh\_clone\_12102.tgz

- database\_inventory\_name: Ansible\_oh\_12102

- oracle\_home: /u01/app/oracle/product/12.1.0.2/dbhome\_1

- oracle\_base: /u01/app/oracle

- tags: oracle\_121\_install\_clone

- roles:

- install\_oracle\_12102\_via\_clone

# CLONE ORACLE HOME: INSTALL\_VIA\_CLONE (1/2)

---

- name: check inventory if database version is already installed
  - shell: >
    - grep 'HOME NAME="{{ database\_inventory\_name }}"' /u01/app/oraInventory/ContentsXML/inventory.xml
    - removes=/u01/app/oraInventory/ContentsXML/inventory.xml
  - register: home\_installed
  
- block:
  - name: create directory for installation files
    - file:
      - dest: "{{ stage\_directory }}"
      - state: directory
      - owner: oracle
      - group: oinstall
    - become\_user: root
  
  - name: transfer clone file for install
    - synchronize:
      - src: "files/{{ clone\_tarball\_name }}"
      - dest: "{{ stage\_directory }}"
  
  - name: untar clone tarball
    - shell: >
      - tar xzf {{ stage\_directory }}/{{ clone\_tarball\_name }} --directory=/

# CLONE ORACLE HOME: INSTALL\_VIA\_CLONE (2/2)

- name: create database home from the clone tarball

shell: >

```
perl {{ oracle_home }}/clone/bin/clone.pl -silent -waitforcompletion -ignoresysprereqs
ORACLE_BASE={{ oracle_base }} ORACLE_HOME={{ oracle_home }}
ORACLE_HOME_NAME={{ database_inventory_name }} INVENTORY_LOCATION=/u01/app/oraInventory OSDBA_GROUP=dba
```

register: run\_clone\_db

failed when: "'The cloning of {{ database\_inventory\_name }} was successful.' not in run\_clone\_db.stdout"

- name: run orainstRoot.sh

shell: >

```
/u01/app/oraInventory/orainstRoot.sh
```

become\_user: root

when: "'/u01/app/oraInventory/orainstRoot.sh' in run\_clone\_db.stdout"

- name: run root.sh

shell: >

```
{{ oracle_home }}/root.sh -silent
```

become\_user: root

- name: clean up install directory

file:

```
path: "{{ stage_directory }}"
```

state: absent

become\_user: root

accentureoperations

when: "home\_installed.rc != 0 or 'skipped' in home\_installed.stdout"

Q

&

A