accentureoperations

PROVISIONING THE ORACLE DATABASE IN THE CLOUD

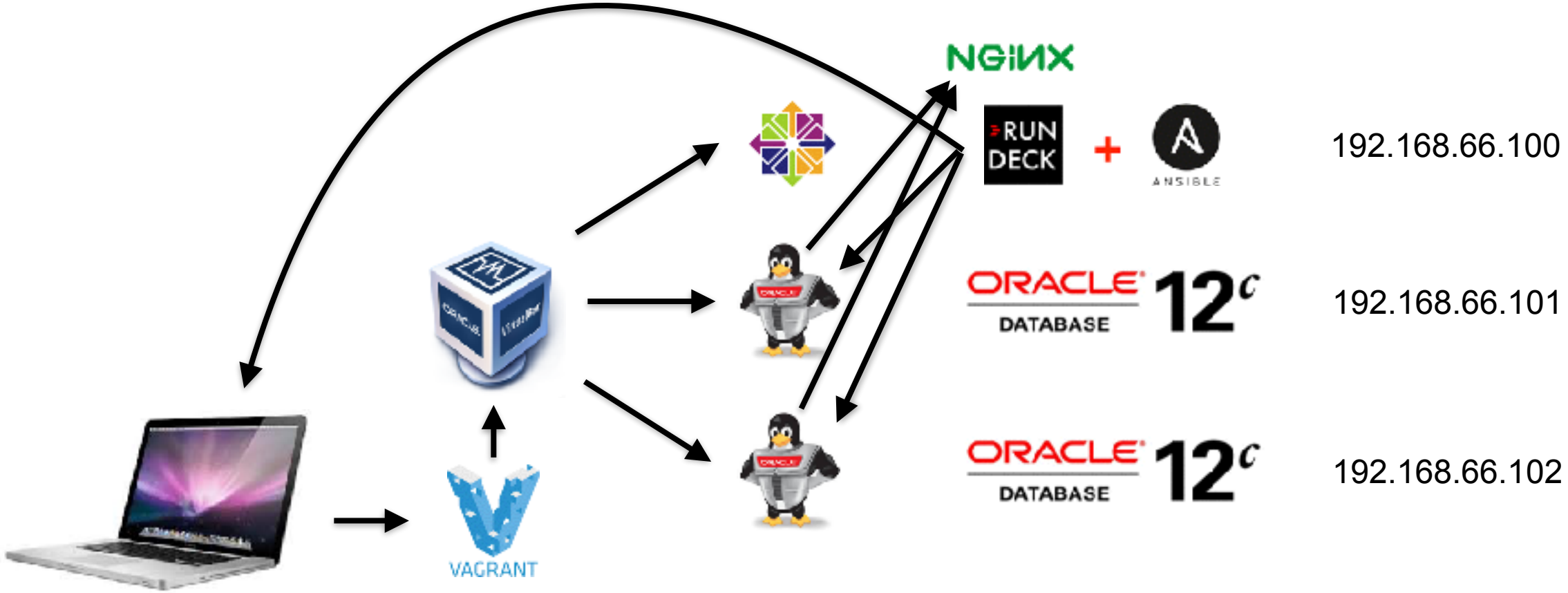
FRITS HOOGLAND

ABOUT THIS PRESENTATION *

- Versions:
  - Database: Oracle version 12.1.0.2 + 17017 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



accentureoperations

ANSIBLE

- I am using Ansible version 2.3.1.0
  - Some functionality used in Ansible might require a recent version, like ‘meta: end_play’

- Ansible essentially describes the desired state.
  - Ansible is “not really” a programming language.
  - Ansible scripts should be re-runnable.

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/

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 individually do it on their own,
  - using a random version of corporate scripts,
  - or not using corporate scripts,
  - or using something somebody produced theirselves?
- 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 cloud exists.

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 = "boxcutter/ol73"
  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
```

accentureoperations

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 = "boxcutter/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
```

accentureoperations

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
```

SETTING UP LINUX

- I like to first disable all repositories in /etc/yum.repos.d/public-yum-ol7.repo.
  - If you are serious about repo handling, create your own in-house repo.
    - And point your machines to your local repository.
  - For (version) testing, I use the ‘update version’ repo instead of ‘latest’.
    - http://public-yum.oracle.com/repo/OracleLinux/OL7/3/base/x86_64/index.html
  - If you want to use an UEK kernel, enable the specific UEK version repo.

- Then install the kernel.
  - You SHOULD test a specific kernel version, and then install that version.
  - A reboot is required to activate a different kernel version.
    - Caveat: the kernel ‘firmware’ package might be in the ‘latest’ repo only.

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

SETTING UP LINUX - REPOSITORY HANDLING

```
---
- hosts: all
  become: true
  vars:
    current_software_repo: ol7u3
    current_kernel_repo: ol7_uek4
    repo_host: 192.168.66.100
  tasks:

  - 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 repository
    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 }}'
```

accentureoperations

SETTING UP LINUX - SETUP KERNEL 1/2

```
---
- hosts: all
  become: true
  vars:
    kernel: kernel-uek-4.1.12-61.1.28.el7uek
  tasks:

  - 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
```


SETTING UP LINUX - SETUP KERNEL 2/2

```
- name: reboot to activate kernel
  shell: >
    sleep 2; /sbin/shutdown -r now "Ansible activate new kernel"
  async: 1
  poll: 0
  when: "'no' in remain_running.stdout"

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

```
- 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"
```

SETTING UP LINUX - ORACLE PREREQS

- Update the current installed packages to the repo version (excluding kernel*!).
- Add a couple of packages: acl, perl, unzip, unixODBC.
- Disable selinux.
- Install the oracle-rdbms-server-12cR1-preinstall package.
  - This creates an oracle user, setup groups, sets kernel parameters, etc.
- Put my public key in the authorized_keys file of the oracle user.

- Setup /u01: PV, VG, LV, filesystem, fstab, mount, set directory ownership to oracle.

SETTING UP LINUX - ORACLE PREREQS 1/2

```
---
- hosts: all
  become: true
  tasks:

  - 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
```


SETTING UP LINUX - ORACLE PREREQS 2/2

```
- name: disable selinux
  selinux:
    state: disabled

- 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') }}"
```


SETTING UP LINUX - /U01 1/2

```
---
- hosts: all
  become: true
  tasks:

  - name: create volume group vg_oracle using /dev/sdb
    lvg:
      vg: vg_oracle
      pvs: /dev/sdb

  - name: create logical volume lv_oracle in vg_oracle
    lvol:
      vg: vg_oracle
      lv: lv_oracle
      size: 35g

  - name: create filesystem in lv_oracle
    filesystem:
      fstype: xfs
      dev: /dev/vg_oracle/lv_oracle
```


SETTING UP LINUX - /U01 2/2

```
- name: mount filesystem to /u01
  mount:
    name: /u01
    src: /dev/vg_oracle/lv_oracle
    state: mounted
    fstype: xfs

- name: set ownership correct for /u01 directory
  file:
    dest: /u01
    state: directory
    owner: oracle
    group: oinstall
```


INSTALL ORACLE DATABASE SOFTWARE

- Figure out if the database software is already installed.
  - The inventory contains an overview of oracle homes in:
    - /u01/app/oraInventory/ContentsXML/inventory.xml
- If not:
  - Create a directory for holding the installation media.
  - Transfer the installation media and unzip it.
  - Run the installer and with the required choices.
  - Run orainstRoot.sh as root when it's a first time install.
  - Run root.sh as root.
  - Clean up the installation directory.

INSTALL ORACLE 1/3

```
---
- hosts: all
  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
  become: true
  become_user: oracle
  tasks:

  - 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

  - name: stop installation if software already installed
    meta: end_play
    when: "database_home_installed.rc == 0 and 'skipped' not in database_home_installed.stdout"
```

INSTALL ORACLE 2/3

```
- 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

- 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
```


INSTALL ORACLE 3/3

```
  - name: install oracle database software
    shell: >
      {{ stage_directory }}/database/runInstaller -silent -force -waitforcompletion -ignoresysprereqs -
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


  - 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
```


INSTALL OPATCH

- The next major thing to do is patch the software up to the desired patch version.

- The first step of patching is to install the latest version of opatch in the ORACLE_HOME:
  - Create a directory for holding the opatch patch file.
  - Transfer opatch patch file and unzip it.
  - Check version of opatch in the ORACLE_HOME and version of latest opatch.
  - If latest patch is different than the version in ORACLE_HOME, unzip it in the O_H.
  - Clean up the installation directory.

INSTALL OPATCH 1/2

```
---
- hosts: all
  vars:
    stage_directory: /u01/stage
    opatch_version: 12.2.0.1.8
    opatch_file: p6880880_121010_Linux-x86-64-{{ opatch_version }}.zip
  become: true
  become_user: oracle
  tasks:

  - 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 }}"
```


INSTALL OPATCH 2/2

```
  - 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.*/
\1/p"
    register: all_12_1_homes

  - 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
   with_items: "{{ all_12_1_homes.stdout_lines }}"

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


INSTALL 170117 PSU

- After opatch has been brought to the latest version:

- Install the 170117 PSU:
  - Check the ORACLE_HOME for already applied patches. If not already applied:
  - Create a directory for holding the patch patch file.
  - Obtain patch patch file and unzip it.
  - Apply the patch.
  - Clean up the patch directory.

INSTALL 170117 PSU 1/3

```
---
- hosts: all
  vars:
    database_inventory_name: Ansible_oh_12102
    stage_directory: /u01/stage
    oracle_home: /u01/app/oracle/product/12.1.0.2/dbhome_1
    patch_number: 24732082
    patch_file: p24732082_121020_Linux-x86-64.zip
    patch_success_line: "Composite patch 24732082 successfully applied."
  become: true
  become_user: oracle
  tasks:

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

  - name: stop patching if patches are already applied
    meta: end_play
    when: "'{{ patch_number }};' in oracle_home_patches.stdout"
```


INSTALL 170117 PSU 2/3

```
- 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/{{ patch_file }}"
    dest: "{{ stage_directory }}"

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


INSTALL 170117 PSU 3/3

```
- name: apply patch to database home
  shell: >
    {{ oracle_home }}/OPatch/opatch apply -silent
    chdir={{ stage_directory }}/{{ patch_number }}
  register: apply_oh_patch
  failed_when: "'{{ patch_success_line }}' not in apply_oh_patch.stdout"

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


INSTALL 170117 JAVAVM PATCH

- After the 170117 PSU, the javavm patch should be installed too!

- Steps to install the 170117 javavm patch:
  - Check the ORACLE_HOME for already applied patches. If not already applied:
  - Create a directory for holding the patch patch file.
  - Obtain patch patch file and unzip it.
  - Apply the patch.
  - Clean up the patch directory.

INSTALL 170117 JAVAVM 1/3

```
---
- hosts: all
  vars:
    database_inventory_name: Ansible_oh_12102
    stage_directory: /u01/stage
    oracle_home: /u01/app/oracle/product/12.1.0.2/dbhome_1
    patch_number: 24917972
    patch_file: p24917972_121020_Linux-x86-64.zip
    patch_success_line: "Patch 24917972 successfully applied."
  become: true
  become_user: oracle
  tasks:

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

  - name: stop patching if patches are already applied
    meta: end_play
    when: "'{{ patch_number }};' in oracle_home_patches.stdout"
```


INSTALL 170117 JAVAVM 2/3

```
- 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/{{ patch_file }}"
    dest: "{{ stage_directory }}"

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


INSTALL 170117 JAVAVM 3/3

```
- name: apply patch to database home
  shell: >
    {{ oracle_home }}/OPatch/opatch apply -silent
    chdir={{ stage_directory }}/{{ patch_number }}
  register: apply_oh_patch
  failed_when: "'{{ patch_success_line }}' not in apply_oh_patch.stdout"

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


SHORTCUT: CLONE A FULLY PATCHED HOME! *

- The steps to create a clone are very simple:
  - Create a temporary directory in /u01 (a gzipped tarred home gets > 3GB).
  - Tar the ORACLE_HOME.
  - Fetch the tarball.
  - Remove the temporary directory.

CREATE A CLONE FROM A DATABASE HOME 1/2

```
---
- hosts: dev
  vars:
    stage_directory: /u01/stage
    oracle_home: /u01/app/oracle/product/12.1.0.2/dbhome_1
    clone_tarball_name: oh_clone_12102.tgz
  become: true
  tasks:

  - 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
```


CREATE A CLONE FROM A DATABASE HOME 2/2

```
- name: copy the database tarball
  fetch:
    src: "{{ stage_directory }}/{{ clone_tarball_name }}"
    dest: files/
    flat: yes
  become: false

- name: remove directory for clone tarballs
  file:
    dest: "{{ stage_directory }}"
    state: absent
```


SHORTCUT: CLONE A FULLY PATCHED HOME! *

- The steps to deploy a cloned (tarred) ORACLE_HOME are simple too:
  - Create a temporary directory in /u01 (a gzipped tarred home gets > 3GB).
  - Obtain the clone tarball.
  - Untar the tarball into it's place.
  - Run clone.pl.
  - Run orainstRoot.sh if it's a first time install, clone.pl tells you to do so.
  - Run root.sh.
  - Remove the temporary directory.

INSTALL ORACLE DATABASE USING CLONE 1/3

```
---
- hosts: all
  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
    clone_tarball_name: oh_clone_12102.tgz
  become: true
  become_user: oracle
  tasks:

  - 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

  - name: stop installation if software already installed
    meta: end_play
    when: "database_home_installed.rc == 0 and 'skipped' not in database_home_installed.stdout"
```

INSTALL ORACLE DATABASE USING CLONE 2/3

```
- 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=/
  args:
    warn: no
```

INSTALL ORACLE DATABASE USING CLONE 3/3

```
  - 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