Administration Tasks

Managing Package management

The YUM Module

- The yum Ansible module uses the *Yum Package Manager* on the managed hosts to handle the package operations.
- The name keyword gives the name of the package to install.
  The state keyword indicates the expected state of the package on the managed host.
- **present**
  - Ansible installs the package if it is not already there.
- **absent**
  - Ansible removes the package if it is installed.
- **latest**
  - Ansible updates the package if it is not already at the most recent available version. If the package is not installed, Ansible installs it.

Compression between Ansible Tasks and yum command.

Ansible Tasks	Yum Command
- name: Install Httpd yum: name: httpd state: present	$ yum install httpd
- name: Install or update Httpd yum: name: httpd state: latest	$ yum update httpd
- name: update all Httpd yum: name: '*' state: latest	$ yum update
- name: Remove Httpd yum: name: httpd state: absent	$ yum remove httpd
- name: Install Group of pkg yum: name: '@Development Tools' state: present	$ yum group install 'Development Tools'
- name: Remove group of pkg yum: name: '@Development Tools' state: absent	$ yum group remove 'Development Tools'
- name: Install pearl with AppStream yum: name: '@perl:5.26/minimal' state: present	$ yum module install perl:5.26/minimal

```
---
- name: Repository Configuration and installing  http
  hosts: prod
  become: yes
  vars:
    pkg_name: httpd
  tasks:
  - name: Gather Package Facts
    package_facts:
      manager: auto
  - name: Show Package Facts package
    debug:
      var: ansible_facts.packages[pkg_name]
    when: pkg_name in ansible_facts.packages
  - name: Configure Yum Repo (1/2)
    yum_repository:
      name: Server1
      description: AppStream Repo
      file: server
      baseurl: ftp://192.168.100.50/pub/AppStream
      enabled: yes
      gpgcheck: no
  - name: Configure Yum Repo (2/2)
    yum_repository:
      name: Server2
      description: BaseOS Repo
      file: server
      baseurl: ftp://192.168.100.50/pub/BaseOS
      enabled: yes
      gpgcheck: no
  - name: Install HTTP
    yum:
      name: "{{ pkg_name }}"
      state: present
  - name: Gather Package Facts
    package_facts:
      manager: auto
  - name: Show Package Facts for the custom package
    debug:
      var: ansible_facts.packages[pkg_name]
    when: pkg_name in ansible_facts.packages
```

Administration Tasks

Managing User and Authentication

The User Module

- The Ansible user module lets you manage user accounts on a remote host.
- You can manage a number of parameters including remove user, set home directory, set the UID for system accounts, manage passwords and associated groupings.
- To create a user that can log into the machine, you need to provide a hashed password for the password parameter.

Options	Description
comment	Description of user
group	To set user's primary group, similar like -g option with useradd
groups	To add user in multiple groups
home	Set user home directory, similar option like -d. with useradd command.
create_home	This option will allow you need to create home directory or not. The value will pass in Boolean.
system	Similar like -r option with useradd command.
uid	Create user with specific user id, similar like -u with useradd command
name	Name of the user

The Group Module

- The group module allows you to manage (add, delete, modify) groups on the managed hosts.
- You need to have groupadd, groupdel or groupmod.
- For windows targets, use the win_group module.

Option	Description
gid	To create group with specific id, similar like -g option with groupadd command
local	Forces to user local command alternatives on platform that implements it.
name	Name of the group
state	To create present and to delete absent
system	To create group with gid below 1000

Administration Tasks

The Known Hosts Module

- If you have a large number of host keys to manage you will want to use the **known_hosts** module.
- The **known_hosts** module lets you add or remove host keys from the known_hosts file on managed host.

The Authorized Key Module

- The **authorized_key** module allows you to add or remove SSH authorized keys per user accounts. When adding and subtracting users to a large bank of servers, you need to be able to manage ssh keys.

Create following host variable under group_vars with host group name.

$ vim group_vars/prod.yml

```
---
passwd: user@123

users:
- username: user1
  groups: admin
- username: user2
  groups: admin
- username: user3
  groups: admin
- username: user4
  groups: admin
- username: user5
  groups: admin
```

Let's write the playbook to implements the above concept

```
---
- name: Create multiple local users
  hosts: prod
  become: yes
  handlers:
  - name: Restart sshd
    service:
      name: sshd
      state: restarted
  tasks:
  - name: Add admin group
    group:
      name: admin
      state: present
  - name: Create user accounts
    user:
      name: "{{ item.username }}"
      groups: admin
      password: "{{ passwd | password_hash ('sha512' , 'A512') }}"
    loop: "{{ users }}"
  - name: Add authorized keys
    authorized_key:
      user: "{{ item.username }}"
      key: "{{ lookup('file', '/home/amit/.ssh/id_rsa.pub') }}"
    loop: "{{ users }}"
  - name: Modify sudo config to allow admin users sudo without a
password
    copy:
      content: "%admin ALL=(ALL) NOPASSWD: ALL"
      dest: /etc/sudoers.d/admin
      mode: 0440
  - name: Disable root login via SSH
    lineinfile:
      dest: /etc/ssh/sshd_config
      regexp: "^PermitRootLogin"
      line: "PermitRootLogin no"
    notify: "Restart sshd"
```

Administration Tasks

Jobs Management

Scheduling with “at” Module

- Quick one-time scheduling is done with the **“at”** module.
- You create the job for a future time to run and it is held until that time comes to execute.
- There are six parameters that come with this module.
- They are: command, count, script_file, state, unique, and units.

Scheduling cron Jobs with cron Module

- When setting a jobs scheduled task the **cron** module is used.
- The cron module will append commands directly into the crontab of the user you designate.

```
---
- name: Recurring cron job
  hosts: prod
  become: true
  tasks:
  - name: Crontab file exists
    cron:
      name: Add date and time to a file
      minute: "*/1"
#       hour: 9-16
#       weekday: 1-5
      user: user1
      job: date >> /home/user1/my_date_time
      cron_file: record_date
      state: present
```

$ ansible prod -a"cat /etc/cron.d/record_date"

```
---
- name: Remove scheduled cron job
  hosts: prod
  become: true
  tasks:
  - name: Cron job removed
    cron:
      name: Add date and time to a file
      user: user1
      cron_file:  record_date
      state: absent
```

Administration Tasks

Storage Management

- Red Hat Ansible Engine provides a collection of modules to configure storage devices on managed hosts. Those modules support partitioning devices, creating logical volumes, and creating and mounting filesystems.

The parted Module

- The parted module supports the partition of block devices. This module includes the functionality of the **parted** command, and allows to create partitions with a specific size, flag, and alignment. The following table lists some of the parameters for the parted module.

Options	Desription
Align	Configuration partition alignment
Device	Your block device name
Flags	Flags for the partition
Number	Partition number
Part_end	Partition size from the beginning of the disk specified in **parted** supported units.
State	Create or remove partition
Units	Size units for the partition information.

The lvg Modules

- The lvg and lvol modules support the creation of logical volumes, including the configuration of physical volumes, and volume groups.
- The lvg takes as parameters the block devices to configure as the back end physical volumes for the volume group.

The following table lists some of the parameters for the lvg module.

Option	Descriptions
pesize	The size of the physical extent. Must be a power of 2, or multiple of 128 KiB.
pvs	List of comma-separated devices to be configured as physical volumes for the volume group.
vg	The name of the volume group.
state	Creates or removes the volume.

The lvol Modules

- The lvol module creates logical volumes, and supports the resizing and shrinking of those volumes, and the filesystems on top of them.
- This module also supports the creation of snapshots for the logical volumes.

The following table lists some of the parameters for the lvol module.

Options	DESCRIPTION
lv	The name of the logical volume.
resizefs	Resizes the filesystem with the logical volume.
shrink	Enable logical volume shrink.
size	The size of the logical volume.
snapshot	The name of the snapshot for the logical volume.
state	Create or remove the logical volume.
vg	The parent volume group for the logical volume.

The filesystem Module

- The filesystem module supports both creating and resizing a filesystem. This module supports filesystem resizing for ext2, ext3, ext4, ext4dev, f2fs, lvm, xfs, and vfat. The following table lists some of the parameters for the filesystem module

Options	Description
dev	Block device name.
fstype	Filesystem type.
resizefs	Grows the filesystem size to the size of the block device.

The mount Module

- The mount module supports the configuration of mount points on **/etc/fstab**.

The following table lists some of the parameters for the mount module.

Options	Descriptions
fstype	Filesystem type.
opts	Mount options.
path	Mount point path.
src	Device to be mounted.
state	Specify the mount status. If set to mounted, the system mounts the device, and configures /etc/fstab with that mount information. To unmount the device and remove it from /etc/fstab use absent.

Ansible Facts for Storage Configuration

- Ansible uses facts to retrieve information to the control node about the configuration of the managed hosts. You can use the setup Ansible module to retrieve all the Ansible facts for a managed host.
- $ ansible localhost -m setup -a 'filter=ansible_devices'

Create Variable files

$ vim ~/group_vars/storage_vars.yml

```
---

partitions:
  - number: 1
    start: 1MiB
    end: 10GiB

volume_groups:
  - name: vg1
    devices: /dev/sda1


logical_volumes:
  - name: lv-vol1
    size: 2G
    vgroup: vg1
    mount_path: /lvol1

  - name: lv-vol2
    size: 3G
    vgroup: vg1
    mount_path: /lvol2
```

Create a mount point tasks

```
---
  - name: creating mount point
    file:
      path: "{{ item }}"
      state: directory
    loop:
     - '/lovl1'
     - '/lvol2'
```

Create a play book for configuring Storage

Make sure devices are attached.

```
---
- name: Storage Configuration
  become: yes
  hosts: prod
  vars_files:
    - /home/amit/group_vars/storage_vars.yml
  tasks:
  - name: Creating partitions on /dev/sda
    parted:
      device: /dev/sda
      state: present
      number: "{{ item.number }}"
      part_start: "{{ item.start }}"
      part_end: "{{ item.end }}"
    loop: "{{ partitions }}"

  - name: Creating Volume group
    lvg:
      vg: "{{ item.name }}"
      pvs: "{{ item.devices }}"
    loop: "{{ volume_groups }}"

  - name: Create Logical volume
    lvol:
      vg: "{{ item.vgroup }}"
      lv: "{{ item.name }}"
      size: "{{ item.size }}"
    loop: "{{ logical_volumes }}"
    when: item.name not in ansible_lvm["lvs"]

  - name: assiging file system
    filesystem:
      dev: "/dev/{{ item.vgroup }}/{{ item.name }}"
      fstype: xfs
    loop: "{{ logical_volumes }}"

  - name: Ensure the correct capacity for each LV
    lvol:
      vg: "{{ item.vgroup }}"
      lv: "{{ item.name }}"
      size: "{{ item.size }}"
      resizefs: yes
      force: yes
    loop: "{{ logical_volumes }}"

  - include_tasks: loop_create_directory.yml

  - name: Each Logical Volume is mounted
    mount:
      path: "{{ item.mount_path }}"
      src: "/dev/{{ item.vgroup }}/{{ item.name }}"
      fstype: xfs
      opts: noatime
      state: mounted
    loop: "{{ logical_volumes }}"
```

Administration Tasks

Do some modification in variable file such as:
Modification are highlighted in red colour

```
---

partitions:
  - number: 1
    start: 1MiB
    end: 10GiB

  - number: 2                              # add this variable
    start: 10GiB
    end:  15GiB

volume_groups:
  - name: vg1
    devices: /dev/sda1,/dev/sda2

logical_volumes:
  - name: lv-vol1
    size: 4G                          # increase the size as per your need
    vgroup: vg1
    mount_path: /lvol1

  - name: lv-vol2
    size: 6G                          # increase the size as per your need
    vgroup: vg1
    mount_path: /lvol2
```

Run the playbook and observe the output

For verification run the given command
$ ansible prod -a lsblk

Network Management

The following table lists the options for the network_connections variable.

Options	Description
name	Identifies the connection profile.
state	The runtime state of a connection profile. Either **up**, if the connection profile is active, or **down** if it is not.
persistent_state	Identifies if a connection profile is persistent. Either **present** if the connection profile is persistent, or **absent** if it is not.

type	Identifies the connection type. Valid values are **ethernet**, **bridge**, **bond**, **team**, **vlan**, **macvlan**, and **infiniband**.
autoconnect	Determines if the connection automatically starts.
mac	Restricts the connection to be used on devices with a specific MAC address.
interface_name	Restricts the connection profile to be used by a specific interface.
zone	Configures the FirewallD zone for the interface.
ip	Determines the IP configuration for the connection. Supports options like for example **address**, to specify a static IP address, or **dns** to configure a DNS server.

- $ ansible-galaxy list
- $ vim /usr/share/doc/rhel-system-roles/network/README.md
- $ sudo yum install rhel-system-roles
- Note: This roles will work with subscription
- Or must have epel (Extra package Enterprise Linux) repository.

Create a variable under group_vars/prod

$ vim group_vars/prod.yml

```
network_connections:
  - name: ens160
    type: ethernet
    ip:
      dhcp4: yes
```

Create a playbook

```
---
 - name: ip add configuration
   become: yes
   hosts:  prod

   roles:
   - rhel-system-roles.network
```

Run the play book