Ansible Best Practices.

Here are some tips for making the most of Ansible and Ansible playbooks.

You can find some example playbooks illustrating these best practices in our ansible-examples EOP.

1.Directory Layout:

The top level of the directory would contain files and directories like so:

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular
groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular
systems
   hostname2.yml

library/                  # if any custom modules, put them here
(optional)
module_utils/             # if any custom module_utils to support
modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here
(optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if
warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template
resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script
resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for
```

```
this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in
this case

    webtier/              # same kind of structure as "common" was
```

```
above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

2. Host mapping in Group and individually.

It is suggested that you define groups based on purpose of the host (roles) and also geography or datacenter location (if applicable):

```
# file: production
[atlanta_webservers]
www-atl-1.example.com
www-atl-2.example.com

[boston_webservers]
www-bos-1.example.com
www-bos-2.example.com

[atlanta_dbservers]
db-atl-1.example.com
db-atl-2.example.com

[boston_dbservers]
db-bos-1.example.com

# webservers in all geos
[webservers:children]
atlanta_webservers
boston_webservers

# dbservers in all geos
[dbservers:children]
atlanta_dbservers
boston_dbservers

# everything in the atlanta geo
[atlanta:children]
atlanta_webservers
atlanta_dbservers

# everything in the boston geo
[boston:children]
boston_webservers
boston_dbservers
```

3. Top Level Playbooks Are Separated By Role

In site.yml, we import a playbook that defines our entire infrastructure. This is a very short example, because it’s just importing some other

playbooks:

```
---
# file: site.yml
- import_playbook: webservers.yml
- import_playbook: dbservers.yml
```

In a file like webservers.yml (also at the top level), we map the configuration of the webservers group to the roles performed by the webservers group:

```
---
# file: webservers.yml
- hosts: webservers
  roles:
    - common
    - webtier
```

4.Group And Host Variables

This section extends on the previous example.

Groups are nice for organization, but that's not all groups are good for. You can also assign variables to them! For instance, atlanta has its own NTP servers, so when setting up ntp.conf, we should use them. Let's set those now:

```
---
# file: group_vars/atlanta
ntp: ntp-atlanta.example.com
backup: backup-atlanta.example.com
```

Variables aren't just for geographic information either! Maybe the webservers have some configuration that doesn't make sense for the database servers:

```
---
# file: group_vars/webservers
apacheMaxRequestsPerChild: 3000
apacheMaxClients: 900
```

If we had any default values, or values that were universally true, we would put them in a file called group_vars/all:

```
---
# file: group_vars/all
ntp: ntp-boston.example.com
backup: backup-boston.example.com
```

We can define specific hardware variance in systems in a host_vars file, but avoid doing this unless you need to:

```
---
# file: host_vars/db-bos-1.example.com
foo_agent_port: 86
bar_agent_port: 99
```

Again, if we are using dynamic inventory sources, many dynamic groups are automatically created. So a tag like “class:webserver” would load in variables from the file “group_vars/ec2_tag_class_webserver” automatically.

5.Task And Handler Organization For A Role

Below is an example tasks file that explains how a role works. Our common role here just sets up NTP, but it could do more if we wanted:

```
---
# file: roles/common/tasks/main.yml

- name: be sure ntp is installed
  yum:
    name: ntp
    state: present
  tags: ntp

- name: be sure ntp is configured
  template:
    src: ntp.conf.j2
    dest: /etc/ntp.conf
  notify:
    - restart ntpd
  tags: ntp

- name: be sure ntpd is running and enabled
  service:
    name: ntpd
    state: started
    enabled: yes
  tags: ntp
```

Here is an example handlers file. As a review, handlers are only fired when certain tasks report changes, and are run at the end of each play:

```
---
# file: roles/common/handlers/main.yml
- name: restart ntpd
  service:
    name: ntpd
    state: restarted
```

Important Suggestions To Novartis Team:

1. Need to create the role using ansible-galaxy only, not through using hardcoding.
2. Separate the role vars and group vars in the directory.
3. Secret management with ansible vault.
4. Use ansible galaxy to create ansible role.
5. Use private ansible galaxy role for re-usable.
6. define variable when conditions are met.
7. pass extra variables in ansible playbooks.
8. view only ansible failures.
9. don't run the ansible host that is not in the inventory.
10. use shell module for passing custom commands in the playbook.
11. use yaml validator for smooth playbook.
12. async action for long running tasks.
13. use molecule for ansible testing purpose in containers.
14. naming the every task in playbook, use simple language to make all to understood.
15. use the tag for the tasks.
16. use files for deploying the shell scripts.

Sample example of Ansible best practices:-

Name
group_vars
roles
LICENSE.md
README.md
hosts.example
site.yml

site.yaml:--

```
---- name: Install WordPress, MySQL, Nginx, and PHP-FPM
    hosts: all
    remote_user: root  # remote_user: user  # become: yes  #
become_method: sudo
     roles:
      - common
      - mysql
      - nginx
      - php-fpm
      - wordpress
```

host.example:-

```
[wordpress-server]

webserver2
```

group_vars:-

```
wp_db_name: wordpress

wp_db_user: wordpresswp

db_password: secret
```

roles:

Added task to reload ansible_facts after installing libselinux-python.
mysql
nginx
php-fpm
wordpress

thats all done!!!!