LAB_SETUP

LAB_SETUP 1

PLAYBOOK 2

Handler playbook 8

Inventory Grouping 10

Variables 13

AD HOC 17

12. Create your environment directories.

Create dev directory. From ansible point of view this dev directory is considered as one environment.
Create ansible.cfg file like below

```
vim ansible.cfg
  [defaults]
  inventory=hosts
  remote_user=ansible
  timeout=3000
```

in hosts file add your servers like below. (static inventory file)

```
vim hosts
[prod]
node1
[backup]
node2
```

****

13. Test your ansible environment is correctly configured or not ?
ansible all --list

[root@server dev]# ansible all --list

hosts (2):
node1
node2

****
14. Test your client servers connectivity from ansible ?

ansible all -m ping --> you must receive ping / pong in green

****

PLAYBOOK

****
prerequisites

1. server.cnl.com – 1 CPU – 1GB RAM (Python 2.7) - Ansible Server
2. node1.cnl.com – 1 CPU – 1GB RAM (python 2.6 and above) - Ansible Client 1
3. node2.cnl.com – 1 CPU – 1GB RAM (python 2.6 and above) - Ansible Client 2

from ansible server login as an ansible user as per class 4.From ansible user execute below command

ansible all -m ping

this above ping command should return with ping / pong green color.

****

Step 1 : ansible-doc
Step 2: .yaml or .yml files
Step 3 : How to construct playbook
Step 4: How to ensure syntax issues in playbook before you execute playbooks
Step 5: How to see detailed output while playbook getting executed
Step 6: Sample Playbook 1
Step 7: Sample Playbook 2
Step 8: Sample Playbook 3
Step 9: httpd installation playbook and error (privilege_escalation)
Step 10: copy paste playbooks from internet - How to avoid intending issues
****

Step 1 : ansible-doc

ansible-doc -l

ansible-doc -l | wc -l

ansible-doc -l | grep package

ansible-doc yum

****
Step 2: .yaml or .yml files

YAML Ain't Markup Language

Yet Another Markup Language
****

Step 3 : How to construct playbook

****
Step 4: How to ensure syntax issues in playbook before you execute playbooks

ansible-playbook <playbook. yml> --syntax-check

simulation

ansible-playbook <playbook. yml> -C

****
Step 5: How to see detailed output while playbook getting executed

ansible-playbook <playbook. yml> -vvvv

detailed verbose

****

Step 6: Sample Playbook 1

```
vim httpd.yaml
---
- name: install httpd
  hosts: all
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest

ansible-playbook httpd.yaml --syntax-check
ansible-playbook httpd.yaml -C
ansible-playbook httpd.yaml
```

****

Step 7: Sample Playbook 2

```
- name: Install a list of packages
  yum:
    name:
      - httpd
      - unzip
    state: present
```

****

Step 8: Sample Playbook 3

```
vim httpd.yaml
---
- name: install httpd
  hosts: all
  tasks:
```

```
  - name: install httpd
    yum:
            name: httpd
            state: latest
  - name: restart web service
          service:
            name: httpd
            state: restarted
```

****

Step 9: httpd installation playbook and error (privilege_escalation)

```
---
- name: install httpd
  hosts: all
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
```

open putty ssh login window 1 from ansible client node1 --> tailf /var/log/messages
open putty ssh login window 1 from ansible client node2 --> watch -n 1 yum list httpd

You will receive the error when you execute the playbook.you must add the privilage_escalation section in ansible.cfg file

vim ansible.cfg

```
[privilege_escalation]
#become=True
#become_method=sudo
#become_user=root
#become_ask_pass=False
```

****

become - set to yes to activate privilege escalation.

become_user - set to user with desired privileges — the user you become, NOT the user you login as.
Does NOT imply become: yes, to allow it to be set at host level. Default value is root.

become_method - (at play or task level) overrides the default method set in ansible.cfg, set to use any of the Become Plugins.

become_flags - (at play or task level) permit the use of specific flags for the tasks or role.

One common use is to change the user to nobody when the shell is set to nologin. Added in Ansible 2.2.

****

Step 10: copy paste playbooks from internet - How to avoid intending issues

```
- name: Install the latest version of Apache
  yum:
    name: httpd
    state: latest

- name: Install Apache >= 2.4
  yum:
    name: httpd>=2.4
    state: present

- name: Install a list of packages (suitable replacement for 2.11 loop deprecation warning)
  yum:
    name:
      - nginx
      - postgresql
      - postgresql-server
    state: present
```

Step 1: Sample Playbook 1

```
- name: Install a list of packages
  yum:
    name:
      - httpd
      - unzip
    state: present
```

****

Step 2: Sample Playbook 2 - 2 play or 2 tasks

vim httpd.yaml

```
---
- name: install httpd
```

```
  hosts: all
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
   - name: restart web service
     service:
      name: httpd
      state: restarted
```

****

Step 3: Sample Playbook 3 - 4 play or 4 tasks

vim httpd.yaml

```
---
- name: install httpd
  hosts: all
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
   - name: restart web service
     service:
      name: httpd
      state: restarted
   - name: install vsftpd
     yum:
      name: vsftpd
      state: latest
   - name: restart web service
     service:
      name: vsftpd
      state: restarted
```

****

Step 4: Sample Playbook 4

ansible module - Group

****

Step 5: Sample Playbook 5

ansible module - user

****
****
Step 6: Sample Playbook 6

```
---
- name: httpd install
  hosts: node1,node2
  tasks:
   - name: httpd install
     yum:
      name: httpd
      state: latest
   - name: httpd service start
     service:
      name: httpd
      state: started
   - name: enable the service
     service:
      name: httpd
      enabled: yes
   - name: download the httpd.conf
     get_url:
      url: https://pepa.holla.cz/wp-content/uploads/2016/12/Ansible.pdf
      dest: /var/www/html/ansible.pdf
      mode: 0644
```

****
****

Handler playbook

handler playbook

vim httpd.yaml

```
---
- name: install httpd
  hosts: all
  tasks:
   - name: install httpd
     yum:
      name: httpd
```

```
      state: latest
     notify:
      - restart web service
  handlers:
   - name: restart web service
     service:
      name: httpd
      state: restarted


**************************************************************************************************************
****************************************
---
- name: httpd install
  hosts: node1,node2
  tasks:
   - name: httpd install
     yum:
      name: httpd
      state: latest
   - name: httpd service start
     service:
      name: httpd
      state: started
   - name: enable the service
     service:
      name: httpd
      enabled: yes
   - name: download the httpd.conf
     get_url:
      url: https://pepa.holla.cz/wp-content/uploads/2016/12/Ansible.pdf
      dest: /var/www/html/ansible.pdf
      mode: 0644
     notify:
      - restart httpd
  handlers:
   - name: restart httpd
     service:
      name: httpd
      state: restarted


**************************************************************************************************************
**********
```

tags

ansible-playbook <playbook name> --list-tags

ansible-playbook <playbook name>   --tags

ansible-playbook <playbook name>   --list-tasks

ansible-playbook <playbook name>   --skip-tags

ansible-playbook <playbook name> --step

Inventory Grouping

Inventory Grouping

node1 --> Dev Group

node2 --> Prod Group

Install httpd in production group of servers only and restart service also in production servers only

vim httpd.yaml

```
---
- name: install httpd
  hosts: prod
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
   - name: restart web service
     service:
      name: httpd
      state: restarted
```

****

Inventory Grouping

node1 --> Dev Group

node2 --> Prod Group

Install httpd in production group of servers only and restart service also in production servers only

vim httpd.yaml

```
---
- name: install httpd
  hosts: dev
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
   - name: restart web service
     service:
      name: httpd
      state: restarted
```

****

Inventory Grouping

node1 --> Dev Group

node2 --> Prod Group

Install httpd in production,dev group of servers and restart service also in both dev & production servers.

vim httpd.yaml

```
---
- name: install httpd
  hosts: prod,dev
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
   - name: restart web service
     service:
      name: httpd
```

```
      state: restarted
```

****

Inventory Grouping

node1 --> Dev Group

node2 --> Prod Group

Install httpd in production group and node2 machine only

```
vim httpd.yaml
---
- name: install httpd
  hosts: prod,node2
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
   - name: restart web service
     service:
      name: httpd
      state: restarted
```

****

Inventory Grouping

node1 --> Dev Group

node2 --> Prod Group

Install all the plays into all the servers which are listed out in the inventory file.But only one play (or) task must be executed in prod group only

```
vim httpd.yaml
---
- name: install httpd
  hosts: all
  tasks:
   - name: install httpd
     yum:
      name: httpd
```

```
      state: latest
     when: inventory_hostname in groups ['prod']
   - name: restart web service
     service:
      name: httpd
      state: restarted
```

****

Inventory Grouping

node1 --> Dev Group

node2 --> Prod Group

Install all the plays into all the servers which are listed out in the inventory file.But only one play (or) task must be executed in prod group and dev group only

vim httpd.yaml

```
---
- name: install httpd
  hosts: all
  tasks:
   - name: install httpd
     yum:
      name: httpd
      state: latest
     when: inventory_hostname in groups ['prod']
   - name: restart web service
     service:
      name: httpd
      state: restarted
```

****

Variables

In general variables are used to store any values later which are used.
In ansible variable working is same we cant also store service name that we want to install or start.

Variables are very useful instead of doing changes in every task simply change the variable value.

For eg : if you want to install httpd then simply assign it to the variable and use that variable in the playbook rather than giving same value again and again to the task.

There are two types of variables :
1. Local : These are only accessible within the file
2. Global : These are accessible to the multiple files.

****

Eg:

Declared local variable

vim httpd.yaml

```
---
- name: installing packages
  hosts: all
  vars:
   - websoft: httpd
  tasks:
   - name: install {{ websoft }}
     yum:
      name: "{{ websoft }}"
      state: latest
   - name: start the {{ websoft }} service
     service:
      name: "{{ websoft }}"
      state: restarted
```

ansible-playbook httpd.yaml

Now execute this playbook and it will be executed without any issues.
coz it is local variable.

vim vsftpd.yaml

```
---
- name: installing packages
  hosts: all
  vars:
   - websoft: httpd
  tasks:
   - name: install {{ websoft }}
     yum:
```

```
    name: "{{ websoft }}"
    state: latest
  - name: start the {{ websoft }} service
    service:
    name: "{{ websoft }}"
    state: restarted
```

Now it will throw errors.coz websoft variable is declared in different file and it wont work from different file.

****

In order to use global variable you have to declare it in your inventory file i.e. hosts file where you have added all your servers with this add global variable also.

Eg: hosts means your inventory file (if you forget means refer your ansible.cfg file)

vim hosts

```
node1
node2

[prod]
node1

[backup]
node2

[all:vars]
websoft=httpd
```

****

Declare websoft. Here all means it will available to all machines.

Removing the local variable.

After executing same output will be there.

You can also restrict the variable for particular machines.

****

Eg:
vim httpd.yaml

```
---
```

```
- name: installing packages
  hosts: all
  tasks:
   - name: install {{ websoft }}
     yum:
      name: "{{ websoft }}"
            state: latest
   - name: start the {{ websoft }} service
     service:
            name: "{{ websoft }}"
            state: restarted
```

In the above playbook, we have asked to execute all macines mentioned in the inventory file.First time execute playbook and it will be executed without any error.

However now go to inventory file try the below

```
[prod:vars]
websoft=httpd
```

save the inventory and exit

Now execute the playbook

```
ansible-playbook httpd.yaml
```

you see backup machines will fail

However now go to inventory file try the below

```
[backup:vars]
websoft=httpd
```

save the inventory and exit

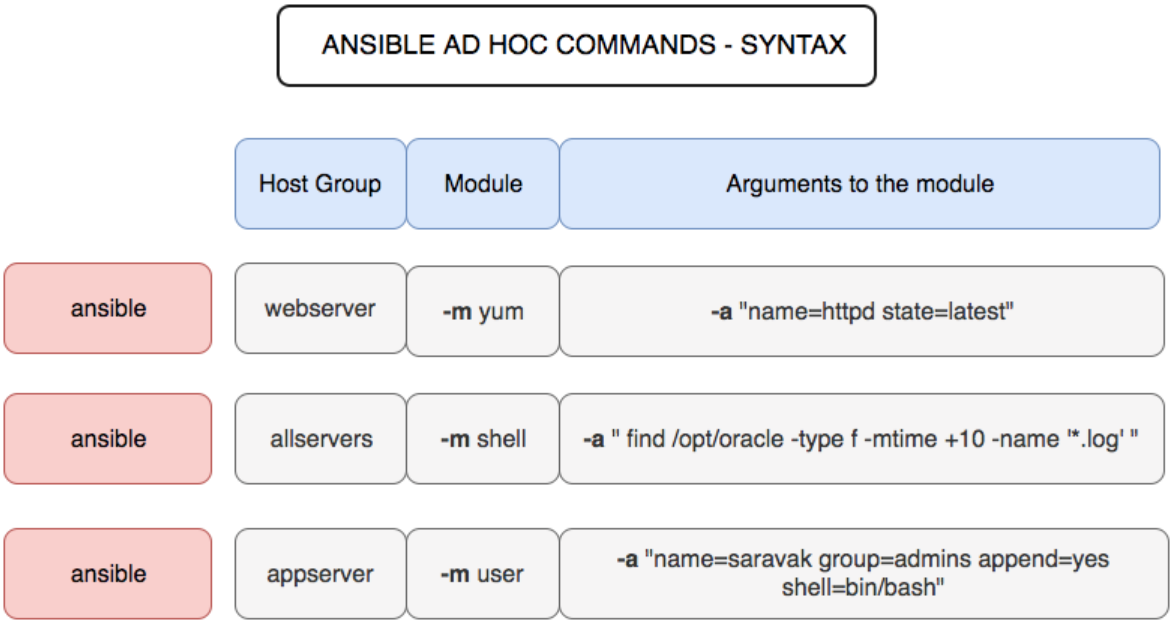
Now execute the playbook

```
ansible-playbook httpd.yaml
```

you see prod machines will fail

This prove that global variable can be accessiable from any playbook.

ADHOC


ANSIBLE AD HOC COMMANDS - SYNTAX
Host Group
Module
Arguments to the module
ansible
webserver
-m yum
-a "name=httpd state=latest"
ansible
allservers
-m shell
-a " find /opt/oracle -type f -mtime +10 -name '*.log' "
ansible
appserver
-m user
-a "name=saravak group=admins append=yes shell=bin/bash"