**Source:** [**http://docs.ansible.com/**](http://docs.ansible.com/) **and Personal experience**

**Ansible Lab guide:**

**What is an Ansible?**

Ansible is, in short, an IT automation, configuration management and provisioning tool. It uses ‘playbooks’ to deploy, manage, build, test and configure anything from full server environments to websites to custom compiled source code for applications.

It brings together aspects of environment management that have been traditionally separate and managed independently.



Taking control of your environment with a single tool has a numerous advantages. With Ansible, you can control server deployment configuration, making everything consistent. With modules and plugins, you can build or ‘hook’ into other applications and control them as well!

**What Other Tools?**

Basically, Ansible ‘plays in the sandbox’ of a large number of deployment and configuration management tools such as:

• Jenkins

• Salt

• Puppet

• Chef

• Fabric

However, Ansible works at a high enough level that it can also be used in conjunction with one or more of these tools. Ansible is very often called an ‘orchestration’ tool since it can function independently as well as ‘control’ one or more of the tools listed above.

**Ansible Vs. The World**

There are some key differences in these technologies, let’s talk about a few:

• Server/Management Nodes

• Puppet/Chef will usually contain a ‘master’ or ‘controller’ server in the setup. Ansible, Operating only with SSH, does not. Any system with Ansible can function in that role at any time based on the task or deployment type.

• Workflow – Push vs. Pull

• Since most configuration management tools have a ‘master’ server, they use the ‘pull’ method (i.e. the client ‘checks in’ on the server to pull it’s configuration). Ansible uses the ‘push’ method, requiring no client installation or configuration (other than general Python).

• Resource Definitions and Execution Ordering

• Puppet (for example) instructions are not applied in order (in other words, not ‘top to bottom’ in how they appear in the manifest). Ansible uses pure in order execution, which can be easy to read as well as convert from other languages or scripts.

• Language

• Ansible is built upon Python and the huge standard of inclusive functionality that comes with it. Puppet (Ruby) and Chef or Salt are not quite so inclusive.

• Syntax

• Puppet is based on custom DSL while Ansible is based on YAML standard.

**What is YAML?**

During its initial development, YAML was reported to stand for “Yet AnotherMarkup Language.” However, once it was completed as a standard, that acronymwas morphed into “YAML Ain’t Markup Language.” In short, YAML is meant to be a “human-readable data serialization format.” In other words, it’s meant for non-computers (us) to be able to read easily and recognize the significance of its content without complex translation applications.

It was designed to be easy to map to high-level languages, and you will often see it compared to key/value lists, associative arrays, and data outlines.

**Structure**

As a data structure, YAML most closely resembles an outline or list of things with basic descriptions.

For example, if we wanted to list our favorite movies in a way that YAML processing engines would be able to recognize, our YAML file would contain something like:

--- # Our Favorite Movies of All Time

- The Terminator

- Star Trek

- Star Wars

**Inventory**:

Ansible works against multiple systems in your infrastructure at the same time. It does this by selecting portions of systems listed in Ansible’s inventory file, which defaults to being saved in the location /etc/ansible/hosts.

**Host**: A host is simply a remote machine that Ansible manages. They can have individual variables assigned to them, and can also be organized in groups.

Ex:

|  |
| --- |
| [dbservers] ----------🡪 group name  ServerA  serverB  172.65.\*\* |

**Group** : A group consists of several hosts assigned to a pool that can be conveniently targeted together, and also given variables that they share in common.

Ex:

|  |
| --- |
| [dbservers] ----------🡪 group name  ServerA  server  172.65.\*\* |

**Playbook :**

Playbooks are the language by which Ansible orchestrates, configures, administers, or deploys systems. Playbooks contain Plays.

**Play**: A play is a mapping between a set of hosts selected by a host specifier and the tasks which run on those hosts to define the role that those systems will perform.

**Module**: Modules are the units of work that Ansible ships out to remote machines. Ansible refers to the collection of available modules as a library.

Ex:

Ping

Setup

Yum

Script

**Ansible command line: Examples of ad-hoc commands:**

**(**You can give **group name** instead of **all)**

ansible all –m ping ----------------🡪 ping command

ansible --list all -----------------🡪 List out the all servers which are in Inventory file(/etc/ansible/hosts)

ansible all --list-hosts -------------🡪 List out the all servers which are in Inventory file(/etc/ansible/hosts)

**Shell Module**: The shell module takes the command name followed by a list of space-delimited arguments. It is almost exactly like the command module but runs the command through a shell (/bin/sh) on the remote node.

**Syntax to use Linux commands as a shell module:** (shell is a Linux command module)

ansible all -s -m shell -a "yum list installed | grep python"

ansible all -s -m shell -a "yum install httpd –y” ------------🡪 install httpd package

ansible all -s -m shell -a "yum remove httpd –y”

**To know system facts:** (setup is a module for system facts)

ansible all -m setup | more -------------------🡪 will displays system facts

ansible local -m setup --tree /tmp/facts ----------🡪 redirect output to /tmp/facts file

ansible all -m setup -a 'filter=\*ipv4\*' -------------🡪 display server ipv4 address

ansible local -m setup -a 'filter=ansible\_\*'

ansible local -m setup -a 'filter=ansible\_arc\*' ---------------🡪 displays server architecture

ansible local -m setup -a 'filter=ansible\_dis\*' ----------------🡪 displays server distribution

ansible local -m setup | grep distribution

If we use \*\*\*\*\*| grep \*\*\*\*, it may or may not give correct information. Filter attribute will give correct information.

ansible local -m setup -a 'filter=ansible\_domain' ---------domain name displays

ansible local -m setup -a 'filter=ansible\_fqdn' -----------🡪 fqdn info

ansible local -m setup -a 'filter=ansible\_interface' --------------🡪 interface details

ansible local -m setup -a 'filter=ansible\_kernal' --------kernal info

ansible local -m setup -a 'filter=ansible\_memtotal\_mb' -------🡪 memory

ansible local -m setup -a 'filter=ansible\_proc\*' ----------🡪 processer

ansible local -m setup -a 'filter=ansible\_virtual\*' --------------🡪 virtual info

**Overriding your host file**:

Please create **hosts** file in any other location and mention servers name and run below command.

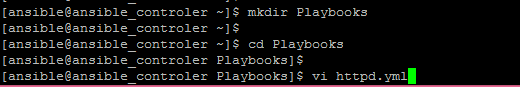
# ansible <your group name> -i <your host file location> -m ping

**Installing package with ansible command line:**

ansible local -s -m yum -a 'pkg=httpd state=installed update\_cache=true'

**First playbook**:

Create a directory called **Playbooks** and build your playbooks



Mention below code in httpd.yml (name is your wish with extension of .yml)

**Playbook1**. Example of Playbook --Httpd.yml ---🡪 it will install the httpd package

|  |
| --- |
| --- # here is your description  - hosts: local  tasks:  - name: install httpd  yum: pkg=httpd state=installed update\_cache=true |

Now run the playbook-

**Command**: ansible-playbook -s httpd.yml



**Playbook2.Variables Include, Vars.yml : ex-1**

**Variable section looks like below in playooks. We can define variables in vars: section and we can pass those variables later ( in this playbook I didn’t pass variables) when we want,.**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi vars.yml  --- # here is your discription  - hosts: local  vars:  controler\_server: ansible\_controler  logs: /var/log/messages  tasks:  - name: install httpd  yum: pkg=httpd state=installed update\_cache=true |

**Playbook 3. Variables Include, Vars.yml : ex-2**

Create a file name with .yml extension and you can refer that file with vars\_files.

We can use vars\_files for variable files..

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi vars.yml  --- # here is your discription  - hosts: local  vars\_files:  - file.yml  tasks:  - name: install httpd  yum: pkg={{ pkg }} state=installed update\_cache=true |

**Playbook 4: Target –** it will describe the targets where we need to deploy and you will not get any o/p for below code

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi target.yml  --- # target section  - hosts: all  user: ansible  sudo: yes  connection: ssh  gather\_facts: no |

**Playbook 5: Tasks.**

**It will descries the tasks section in playbook as shown below**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi tasks.yml  ---  - hosts: all  user: ansible  connection: ssh  gather\_facts: no  tasks:  - name: installing the httpd  action: yum name=httpd state=installed  - name: check for git  action: yum name=git state=absent |

**Playbook 6: handlers ; Here I introduced notify and handles. In this playbook, when package installed, it will notify to handles.when handles get notification, it will execute the task**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi handlers.yml  ---  - hosts: all  user: ansible  connection: ssh  gather\_facts: no  tasks:  - name: install httpd  yum: name=httpd state=latest  notify:  - Restart HTTPD  handlers:  - name: Restart HTTPD  service: name=httpd state=restarted |

**Playbook 7: Transfer the template**

**It will show how to copy file to remote hosts from local server**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi copy.yml  ---  - hosts: all  tasks:  - name: write the file  template: src=/home/ansible/test dest=/home/ansible/infy/ |

**Playbook 8: logging**

**Here I am introducing few more modules. i.e. raw , yum, service , command. And also introduced how to register the result for debug.**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi logging.yml  ---  - hosts: all  user: ansible  sudo: yes  gather\_facts: no  connection: ssh  tasks:  - name: date/time stamp for when playbook starts  raw: /usr/bin/date > /home/ansible/playbook\_starts.log  - name: install the apache  yum: name=httpd state=latest  - name: start the httpd service  service: name=httpd state=restarted  - name: verify that services are running  command: systemctl status httpd  register: result  - debug: var=result  - name: log the packages  raw: yum list installed > /home/ansible/installed.log  - name: date/time stamp for when playbook ends  raw: /usr/bin/date > /home/ansible/playbook\_end.log |

**Playbook9 9: Dry run**

$ ansible-playbook <playbook\_name> - - check



**Playbook 10: Asynchronous and polling**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi asyncNpolling.yml  ---  - hosts: all  user: ansible  connection: ssh  gather\_facts: no  tasks:  - name: install httpd  yum: name=httpd state=latest  async: 500  poll: 5  notify:  - Restart HTTPD  handlers:  - name: Restart HTTPD  service: name=httpd state=restarted |

**Playbook 11 : Variables substitution:**

|  |
| --- |
| **[**ansible@ansible\_controler Playbooks]$ vi variables\_subst.yml  ---  - hosts: all  user: ansible  sudo: yes  connection: ssh  gather\_facts: no  vars\_files:  - files.yml  tasks:  - name: install httpd with using variable file  yum: name={{ pkg }} state=installed |

**Playbook: 12: prompting variables**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi prompt\_variable.yml  ---  - hosts: all  user: ansible  sudo: yes  connection: ssh  gather\_facts: no  vars\_prompt:  - name: pkg  prompt: which package needs to install?  tasks:  - name: install the pkg with using variable file  yum: name={{ pkg }} state=installed |

**Playbook: 13 ; Local action**

**$ Ansible-playbook <your playbook\_name> - -connection=local**

**(Or)**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi local.yml  ---  - hosts: 127.0.0.1  connection: local  tasks:  - name: install httpd  yum: name=httpd state=present |

**Playbook 14: basic loops**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi loops.yml  ---  - hosts: all  user: ansible  sudo: yes  connection: ssh  gather\_facts: no  tasks:  - name: add a list of users  user: name={{ item }} state=present  with\_items:  - user1  - user2  - user3 |

**Playbook 15: when conditional (basic) -1**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ vi conditionals.yml  ---  - hosts: all  user: ansible  sudo: yes  connection: ssh  tasks:  - name: install apachi to Debian linux  command: apt-get -y install apache2  when: ansible\_os\_family == "Debian"  - name: install apache in RedHat/centos  command: yum -y install httpd  when: ansible\_os\_family == "RedHat" |

**Playbook 16: when conditional -2**

|  |
| --- |
| [ansible@ansible\_controler Playbooks]$ cat adv\_conditional.yml  ---  - hosts: local  sudo: yes  vars:  create\_file: true  tasks:  - name: create a file  file:  state: touch  path: /home/ansible/variable\_was\_true  when: create\_file |

**Playbook 17: nested loops-1**

|  |
| --- |
| **[ansible@ansible\_controler Playbooks]$ cat adv\_conditional1.yml**  **---**  **- hosts: local**  **tasks:**  **- name: multiple items**  **file:**  **state: touch**  **path: /home/ansible/{{ item }}**  **with\_items:**  **- file1**  **- file2**  **- file3** |

**Playbook 18: nested loops-2**

|  |
| --- |
| **[ansible@ansible\_controler ~]$ cat nestedloops.yml**  **---**  **- hosts: local**  **tasks:**  **- debug:**  **msg: "{{ item }}"**  **with\_file:**  **- file1**  **- file2** |

**Playbook 19: Include statement**

**Create a file like below**

|  |
| --- |
| **[ansible@ansible\_controler Playbooks]$ cat include.yml**  **---**  **- name: install apache**  **yum: pkg=httpd state=latest**  **- name: install git**  **yum: pkg=git state=latest** |

**And include the file as shown below in playbook**

|  |
| --- |
| **[ansible@ansible\_controler Playbooks]$ cat include\_main.yml**  **---**  **- hosts: local**  **tasks:**  **- include: include.yml** |

**Playbook 20: Error Handling**

|  |
| --- |
| **[ansible@ansible\_controler Playbooks]$ cat failures.yml**  **---**  **- hosts: local**  **tasks:**  **- name: excute a command which will fail**  **command: /bin/abcd**  **ignore\_errors: yes**  **- name: install httpd**  **yum: pkg=httpd state=latest** |

**Playbook 21: starting at task or stepping through all tasks**

|  |
| --- |
| **[ansible@ansible\_controler Playbooks]$ cat startNstopNcan.yml**  **---**  **- hosts: local**  **tasks:**  **- name: install httpd**  **yum: pkg=httpd state=latest**  **- name: install git**  **yum: pkg=git state=latest**  **- name: list home dir**  **command: ls -la /home/ansible**  **register: result**  **- debug: var=result** |



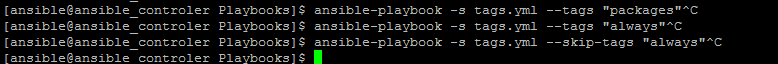
**Playbook 22: Passing variables into playbook at the command line**

|  |
| --- |
| **[ansible@ansible\_controler Playbooks]$ cat commandlinepass.yml**  **---**  **- hosts: '{{ hosts }}'**  **user: '{{ user }}'**  **tasks:**  **- name: install httpd**  **yum: pkg={{ pkg }} state=latest** |



**Playbook 23: tags: can control play books with command line**

|  |
| --- |
| **[ansible@ansible\_controler Playbooks]$ cat tags.yml**  **---**  **- hosts: local**  **tasks:**  **- name: install httpd and git**  **yum: pkg={{ item }} state=installed**  **with\_items:**  **- httpd**  **- git**  **tags:**  **- packages**  **- name: verify httpd service**  **raw: yum list installed | grep httpd > /home/ansible/pkg.log**  **tags:**  **- always** |



**Playbook 24: deligate\_to**

|  |
| --- |
| **Vi deligate\_to.yml**  **--- # DELEGATE TO PLAYBOOK**  **- hosts: 127.0.0.1**  **user: \*\*\***  **sudo: yes**  **connection: ssh**  **tasks:**  **- name: Install Telnet Client**  **yum: pkg=telnet state=latest**  **delegate\_to: 127.0.0.1** |

**Play book 25: Outline**

|  |
| --- |
| **$ cat webserver.txt**  **- webservers**  **- test user**  **- sudo rights**  **- date/time stamp for when the playbook start**  **- install the apache web server**  **- start the web service**  **- verify that the web service is running**  **- install client software**  **- telnet**  **- lynx**  **- log all the packages install on the system**  **- date/time stamp for when the playbook ends** |

**Playbook creation from outline.txt**

|  |
| --- |
| **- hosts: apacheweb**  **user: test**  **sudo: yes**  **gather\_facts: no**  **tasks:**  **- name: date/time stamp for when the playbook starts**  **raw: /usr/bin/date > /home/test/playbook\_start.log**  **- name: install the apache web server**  **yum: pkg=httpd state=latest**  **- name: start the web service**  **service: name=httpd state=restarted**  **- name: verify that the web service is running**  **command: systemctl status httpd**  **register: result**  **- debug: var=result**  **- name: install client software - telnet**  **yum: pkg=telnet state=latest**  **- name: install client software - lynx**  **yum: pkg=lynx state=latest**  **- name: log all the packages install on the system**  **raw: yum list installed > /home/test/installed.log**  **- name: date/time stamp for when the playbook ends**  **raw: /usr/bin/date > /home/test/playbook\_end.log** |

**Vault: 26**

ansible-vault create <filename>

ansible-vault edit <filename>

ansible-vault rekey <filename>

ansible-vault decrypt <filename>

ansible-vault encrypt <filename>

Playbook:27 Run\_once:

|  |
| --- |
| [ansible@ip-172-31-42-216 Playbooks]$ cat run\_once.yml  - hosts: all  user: ansible  sudo: yes  connection: ssh  gather\_facts: no  tasks:  - name: Run the uptime command on all hosts and log it  raw: /usr/bin/uptime >> /home/ansible/uptime1.log  - name: List the /var directory and log it  raw: ls -al /var >> /home/ansible/dir1.list  - name: again date  raw: /usr/bin/date >> /home/ansible/date11.log  run\_once: true |

**Playbook: 28 local action:**

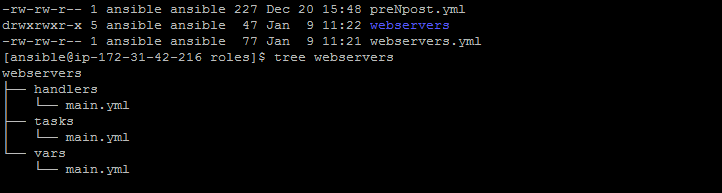
|  |
| --- |
| **[ansible@ip-172-31-42-216 Playbooks]$ cat local.yml**  **---**  **- hosts: 127.0.0.1**  **connection: local**  **tasks:**  **- name: install httpd**  **yum: name=httpd state=present** |

**29. Roles**:

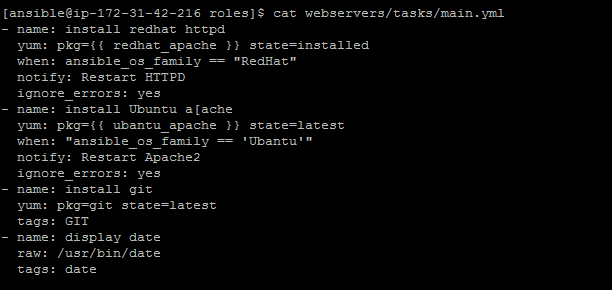
Roles are ways of automatically loading certain vars\_files, tasks, and handlers based on a known file structure. Grouping content by roles also allows easy sharing of roles with other users.

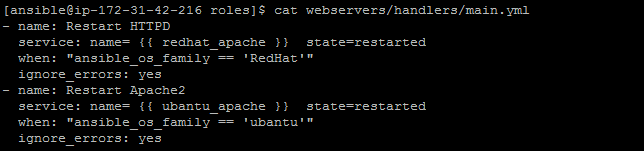
Lab for roles:

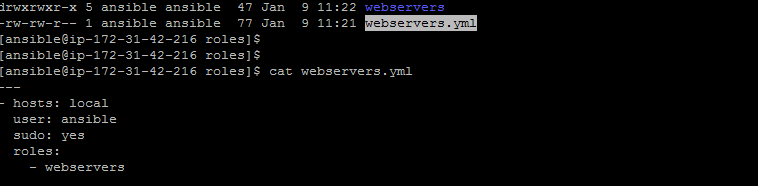
Create a directory structure as shown below for webservers and create a sub-directories ( with name of ansible sections) like vars, tasks, handlers. And create a main.yml files as shown below as related to sections. Pls follow below screen shots.













Playbook 30: roles pre-task and post-task:

|  |
| --- |
| [ansible@ansible\_controler roles]$ cat webservers.yml  ---  - hosts: local  user: ansible  sudo: yes  pre\_tasks:  - name: install git  yum: pkg=git state=latest  roles:  - webservers  post\_tasks:  - name: date  raw: /usr/bin/date > /home/ansible/rolesdate.log |

**Some more loops example playbooks:**

|  |
| --- |
| **[Ram@ansible\_master Playbooks]$ cat nested\_loops.yml**  **---**  **- hosts: local**  **tasks:**  **- name: multiple items**  **file:**  **state: directory**  **path: /tmp/{{ item[0] }}/{{ item[1] }}**  **with\_nested:**  **- ['folder1','folder2']**  **- ['file1','file2','file3']** |

|  |
| --- |
| **[Ram@ansible\_master Playbooks]$ cat loops\_new.yml**  **---**  **- hosts: all**  **tasks:**  **- name: Installing Packages**  **yum: name={{item.name}} state={{item.value}}**  **with\_items:**  **- {name: 'httpd', value: 'absent'}**  **- {name: 'git', value: 'absent'}** |

|  |
| --- |
| **[Ram@ansible\_master Playbooks]$ cat loops\_new\_new.yml**  **- hosts: localhost**  **vars:**  **myvar:**  **- 1**  **- 2**  **- 3**  **- 4**  **- 5**  **tasks:**  **- name: break loop after 3**  **debug: msg={{item}}**  **failed\_when: item == 3**  **register: myresults**  **when: not (myresults|default({}))|failed**  **with\_items: "{{myvar}}"** |

|  |
| --- |
| **[Ram@ansible\_master Playbooks]$ cat loops\_new\_new123.yml**  **---**  **- hosts: all**  **become: yes**  **vars:**  **httpd\_packages:**  **- httpd**  **- mod\_wsgi**  **tasks:**  **- name: install httpd packages**  **package:**  **name: "{{ item }}"**  **state: present**  **with\_items: "{{ httpd\_packages }}"** |

**For more examples and option , pls refer official Ansible docs.**

**conditions1.yml**

**conditions2.yml**

**conditions3.yml**

**conditions4.yml**