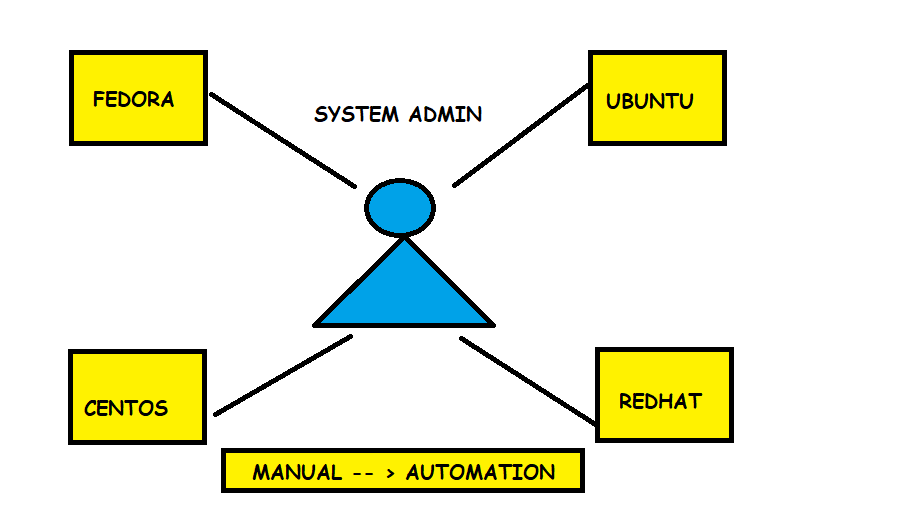
**ANSIBLE**

* It is a Configuration Management Tool.
* Configuration: Ram, Storage, OS, Software, and IP address of the device.
* Management: Update, Delete, Add.
* Ansible is a simple open-source IT engine that automates application deployment.
* Orchestration, Security, and compliance.
* Uses YAML Scripting language which works on KEY-PAIR
* Ansible GUI is called Ansible Tower. It was just Drag and Drop.
* Used PYTHON for the Back end.

**HISTORY**

* Michael Dehhan developed Ansible and the Ansible project began in Feb 2012.
* Ansible was taken over by Red-hat.
* Ansible is Available for RHEL, Debian, CentOS, Oracle Linux.
* Can use this tool whether your servers are in On-prem or in the Cloud.
* It turns your code into Infrastructure i.e. Your computing environment has some of the same attributes as your application.



If the system admin has to install those Linux flavors across all the systems in his company, then he has to do it manually. In manual work, there might be some errors so we use here automated tools like

Ansible, Chef, Puppet, etc.

ANSIBLE : PUSH CHEF : PULL

**PUSH:** if we have many servers then it will push the notification for updates on all devices.

**PULL:** It will go to the client-server and ask for the notifications for the update.

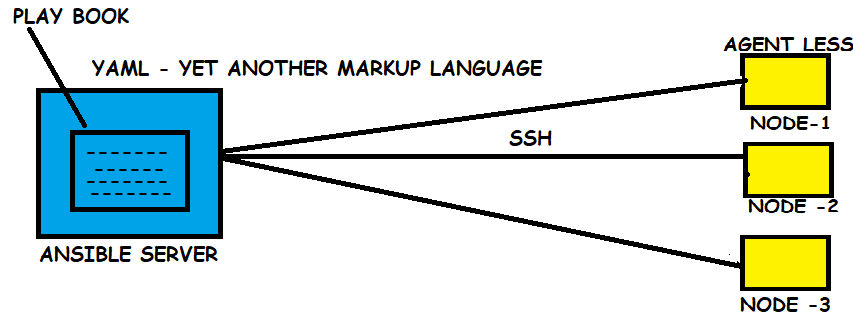
**ADVANTAGES**

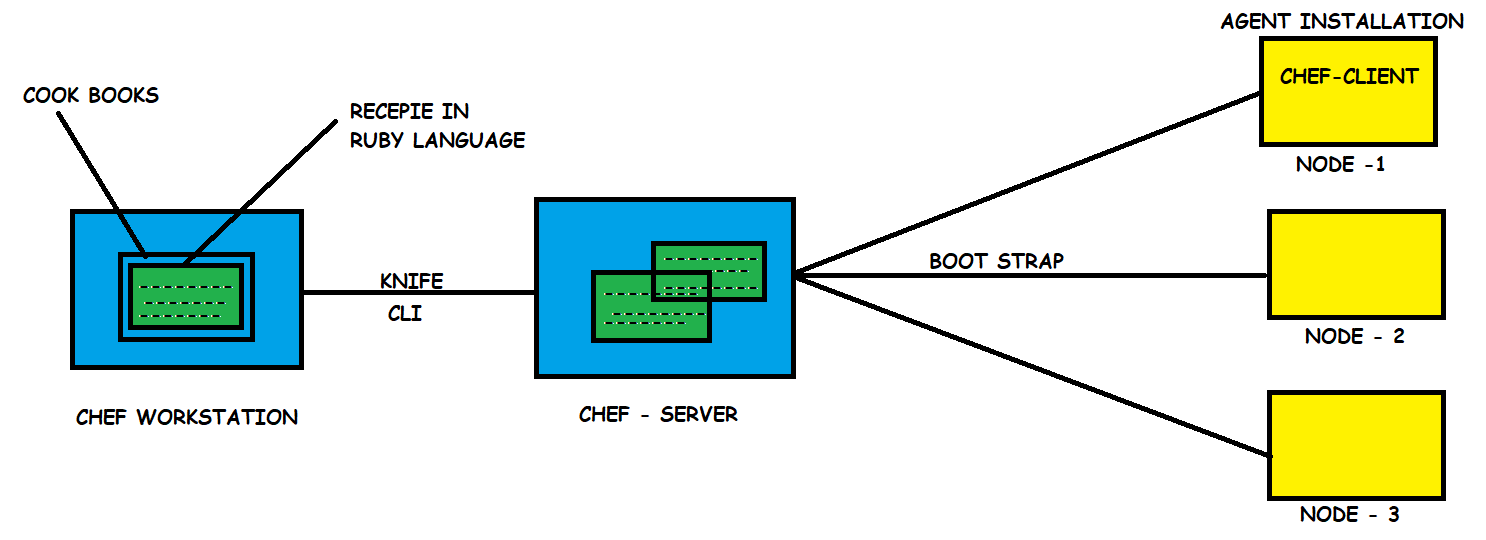
* Very consistent and lightweight and no constraints regarding the OS or underlying H.W.
* Secure less due to Agentless Capability and Open SSH Security features.
* Doesn’t require any special system admin skills to install and use (YAML).
* Push mechanism.

**DIS ADVANTAGES**

* Ansible does not have any notion of state like other automation tools such as Puppet
* Ansible does not track dependencies and simply executes sequential tasks and stops when tasks finish, fail, or any error comes.
* Ansible has external dependencies on Python modules
* Windows interaction requires some scheming

**ANSIBLE WORKFLOW**

**CHEF WORKFLOW**



**ANSIBLE SERVER**: The machine where ansible is installed& from which all task and playbooks will run

**MODULE**: A command or set of similar Commands meant to be executed on Client side.

**TASK**: A section that consists of a single Procedure to be completed.

**ROLE**: A way of organizing the Tasks and Related files to be later called in playbook.

**FACT**: Info fetched from the client system from the Global variables with the Gather-facts operation.

**INVENTORY**: File containing Data about the Ansible client servers.

**PLAY**: Execution of Playbook.

**HANDLER**: Task which is called only if a notifier is present.

**NOTIFIER**: Section attributed to a task which calls a handler if the Output is changed.

**PLAYBOOKS**: It consist code in YAML format, which describes task to be Executed.

**HOST**: Host or Nodes, which are Automated by Ansible.

**ANSIBLE INVENTORY HOST PATTREN**

* Create 3 EC2 instances in same Availability Zone & Connect through Putty and give sudo su.
* yum update -y
* sudo amazon-linux-extras install ansible2 -y
* yum install git python python-level python-pip openssl -y & check versions.
* vi /etc/ansible/hosts file in Ansible server and [remo] & paste private IP of node-1 & node-2.
* # Vi etc/ansible/ansible.cfg
* Uncommented -- > inventory: /etc/ansible/hosts & Sudo-user: root. Save and quit.
* Create user called ansible and set password and add ansible user to sudo file.
* Now do this process on both other nodes too.
* Go to ansible server and install httpd package as ansible user and exit to root.
* Open vi /etc/ssh/sshd\_config in **root** in all three servers.
* service sshd restart and login as ansible in all 3 servers .
* Su – ansible & ssh IP of any node it will ask password and enter it then you will be on node-1.
* Create some files on ansible server and it will replicate in node-1

Now again if you want to login in node-1 you need to give password to get rid of that we need to do

* Go to ansible server -- > ssh-keygen -- > ls -al -- > cd .ssh -- > ls -- > id\_ras\_pub
* Now we need to copy public key in both the nodes
* ssh-copy-id ansible@private-ipv4 of node-1 and it will ask password enter it.
* Ssh-copy-id ansible@private-ipv4 of node-2 and it will ask password enter it.
* Now go to ansible and ssh ipv4 node-1 it will not ask password now and exit
* ssh ipv4 of node-2 it will also not ask password now.

**HOST PATTRENS**

* **‘all’** patterns refer to all the machines in an inventory.

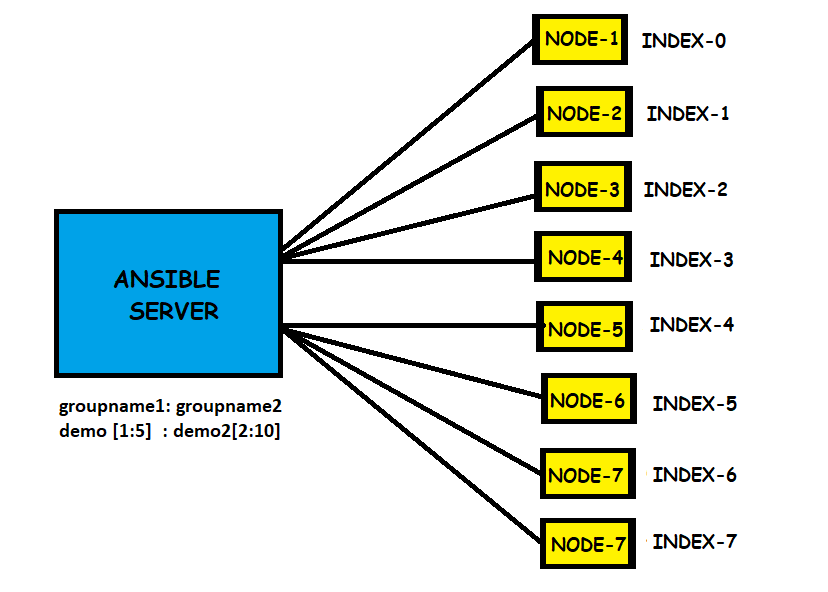
ansible all –list-hosts ansible <groupname[remo]> --list-hosts

ansible <groupname> [remo][0] --list-hosts

groupname [0] – picks first machine of group groupname [1] – picks second machine of group

groupname [-1] – picks last machine of group groupname [1:4] – picks 2,3,4,5 machines in group

groupname [2:5] – picks 3,4,5,6 machines in the group



If we want to push the code from Ansible server to nodes it can be done in 3 ways.

1. Ad-hoc Commands (Simple Linux) Ad-hoc means temporary & it will over-ride commands.
2. Modules – A Single Command.
3. Playbooks – More than one module is called Playbook.

Both module and Playbook is in YAML.

**Ad-Hoc Commands**

* These commands can be run individually to perform Quick functions.
* Not used for configuration management and deployment, bcz the cmds are one time usage.
* The ansible ad-hoc cmds uses /usr/bin/ansible/ command line tool to automate single task.

Go to ansible server and switch to ansible server

ansible remo -a “ls” [remo: Group name, -a: argument, ls: command]

ansible remo [0] -a “touch file1”

ansible all -a “touch file2”

ansible remo -a “sudo yum install httpd -y”

ansible remo -ba “yum install httpd -y” (b: become you will become sudo user)

ansible remo -ba “yum remove httpd -y”

**ANSIBLE MODULES**

* Ansible comes with modules (also called as module library) that can be executed directly on remote hosts or through playbooks. Ansible allows users to write their own modules. These helps in performing various operations from starting services, installing packages to executing commands.
* Some of the modules are action, command, user, group, file, raw, script, shell, copy etc..

**ANSIBLE INVENTORY FILES:**

The default location for the inventory file is /etc/ansible/hosts

Go to ansible server and switch to ansible server

ansible remo -b -m yum -a “pkg=httpd state=present” (install: present)

ansible remo -b -m yum -a “pkg=httpd state=latest” (update: latest)

ansible remo -b -m yum -a “pkg=httpd state=absent” (uninstall: absent)

ansible remo -b -m service -a “name=httpd state=started” (started: start)

ansible remo -b -m user -a “name=raj” (to check go to that server and sudo cat /etc/passwd).

ansible remo -b -m copy -a “src=filename dest=/tmp” (to check go to that server and give ls /tmp).

ansible remo -m setup

ansible remo -m setup -a “filter=\*ipv4\*”

**PLAYBOOKS**

* Playbooks in Ansible are written in the YAML language.
* It is a human-readable & serialization language commonly used for configuration files.
* You can write codes consisting of vars, tasks, handlers, files, templates, and roles.
* Each playbook is composed of one or more modules in a list.
* A module is a collection of configuration files.
* Playbooks are mainly divided into sections like

**TARGET SECTION**: Defines host against which playbooks task has to be executed.

**VARIABLE SECTION**: Defines variables.

**TASK SECTION**: List of all modules that we need to run in order.

**YAML**

For ansible, nearly every YAML file starts with a list

* Each item in the list is a list ok key-value pairs commonly called Dictionary.
* All YAML files have to begin with ”---” and end with “…”
* All members of the list line must begin with the same indentation level starting with “ --- “

For example:

--- # A list of fruits

Fruits:

-mango

-apple

-papaya

-guava

…

* A dictionary is required in a simple key: value form (note: space before value is must)

For example:

--- # Customer details

Customer:

Name: Raham

Age : 22 y

Salary: 30,000

Exp : 1 year

* Extension for playbook file is **.yml**

Go to ansible server and login as ansible and create one playbook

* Vi target.yml

---# Target Playbook

* hosts: remo -- > remo: Groupname

user: ansible -- > ansible: You are ansible user now

become: yes -- > become: become sudo user -- > yes

connection: ssh

gather\_facts: yes -- > Gives private IP of the nodes -- > yes

now save that file and execute the playbook by giving the command: ansible-playbook target.yml

Now create one more playbook in ansible server with cmd Vi task.yml

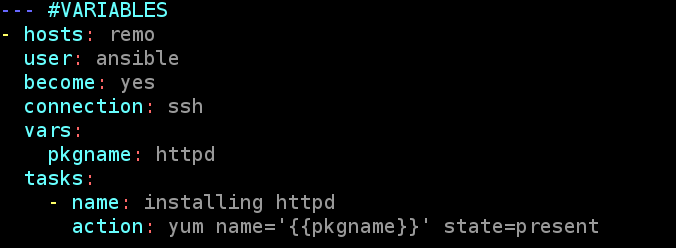
****

Now execute the file by command **ansible-playbook task.yml**

**VARIABLES**

* Ansible uses variables which are defined previously to enable more flexibility in playbooks and roles they can used loop through a set of given values, access various information like the host name of a system and replace certain strings in templates with specific values.
* Write Variable section above tasks so that we define in first and use it later.

Now go to ansible server and create one playbook



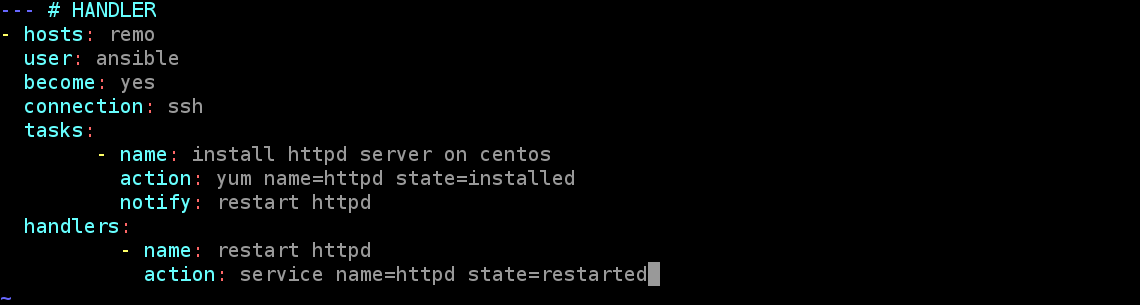
Now save and execute the playbook

**HANDLERS**

* Handler is the same as the task but it will run when called by another task. (OR)
* It will run if the task contains a notification directive and also indicates that it changed something.

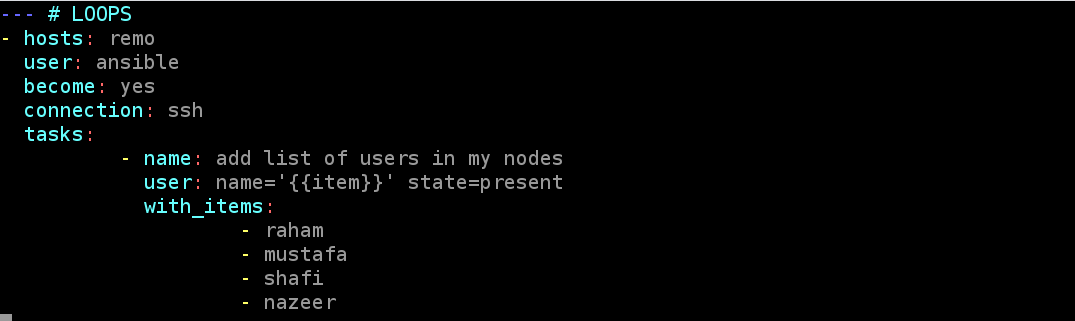
DRY RUN: Check whether the playbook is formatted correctly or not.

Ansible-playbook --syntax-check handler.yml



**LOOPS**

* Ansible loop includes changing ownership on several files & directories with file module, creating multiple users with user modules and repeating a polling step until result reached.



Now save and execute the file and go to the nodes and check with cat /etc/passwd.

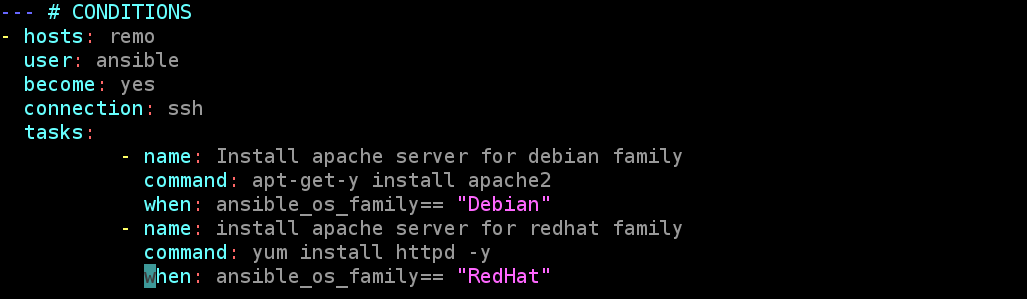
Follow correct Indentation as previous yml files. **Replace (= with -)** in above file.

**CONDITIONS**

* If we have different scenarios, then we apply conditions according to the scenarios.

**WHEN STATEMENT**

* Sometimes we want to skip a particular command on a particular node.



**VAULT**

* In ansible we can keep sensitive data like our passwords and keys in encrypted format.
* **ENCRYPTION TECHNIQUE: AES256** Used by Facebook.

ansible-vault create vault.yml : creating a new encrypted playbook.

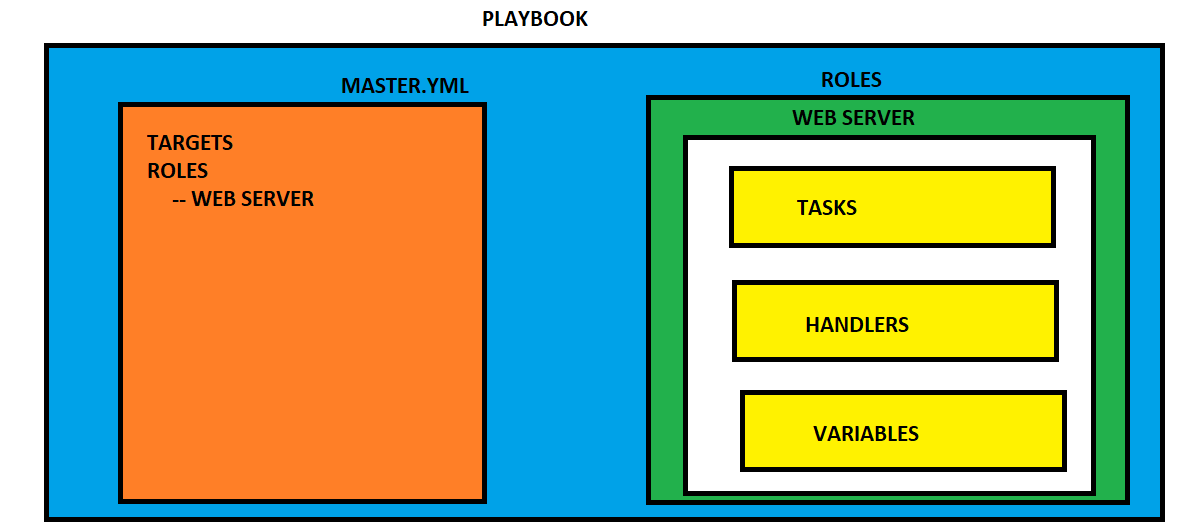
ansible-vault edit vault.yml : Edit the encrypted playbook.

ansible-vault rekey vault.yml : To edit the password.

ansible-vault encrypt vault.yml : To encrypt the existing playbook.

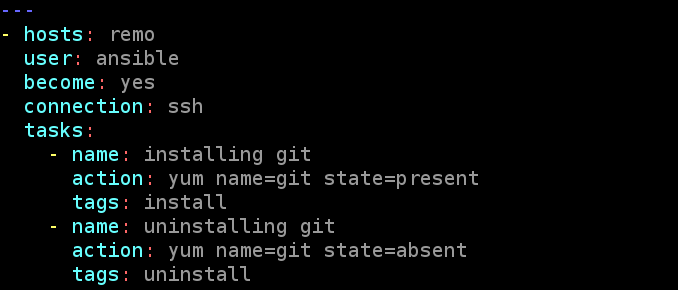
ansible-vault decrypt vault.yml : To decrypt the encrypted playbook.

ansible-playbook master.yml



**ANSIBLE TAGS:**

If you have a large playbook, it may be useful to run only specific parts of it instead of running the entire playbook. You can do this with Ansible tags. Using tags to execute or skip selected tasks

\

ansible-playbook abc.yml --skip-tags "uninstall"

**ANSIBLE-GALAXY:**

Ansible Galaxy is a galaxy website where users can share roles and to a command-line tool for **installing, creating,** and **managing** roles.

Ansible Galaxy gives greater visibility to one of Ansible's most exciting features, such as application installation or reusable roles for server configuration. Lots of people share roles in the Ansible Galaxy.

Ansible roles consist of many playbooks, which is a way to group multiple tasks into one container to do the automation in a very effective manner with clean, directory structures.

ansible-galaxy init raham

ansible-galaxy search elasticsearch

ansible-galaxy search elasticsearch --author alikins

ansible-galaxy install alikns.elasticsearch

cd /home/ansible/.ansible/roles/

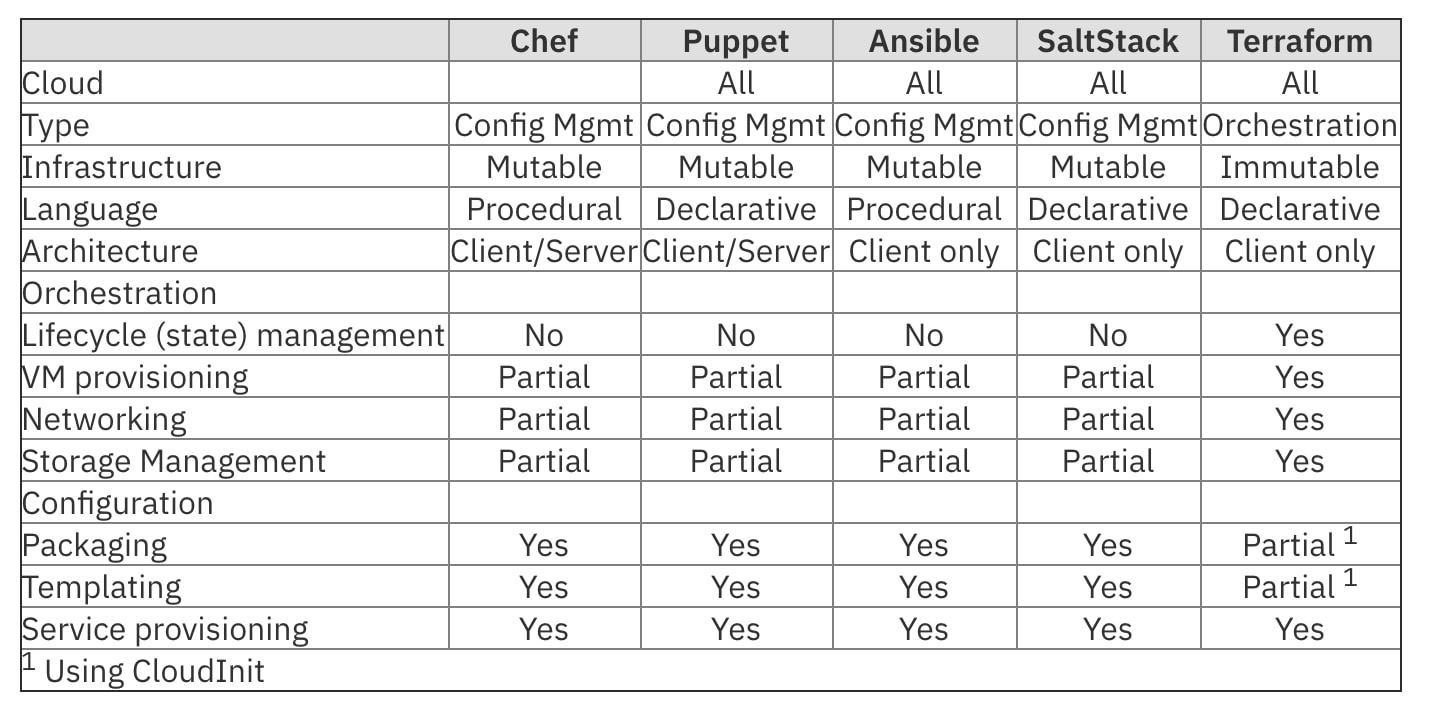
**USER INFO:**

Go to the ansible-galaxy website and select a username

ansible-galaxy info bonddim.linux

ansible-galaxy collection install bonddim.linux

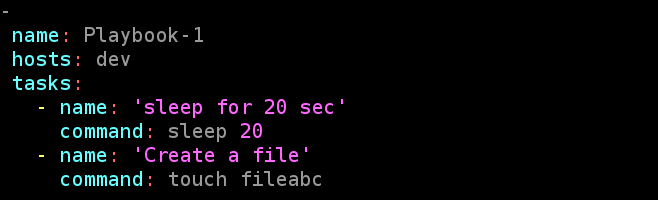
**DIFFRERENCES BLW TOOLS:**

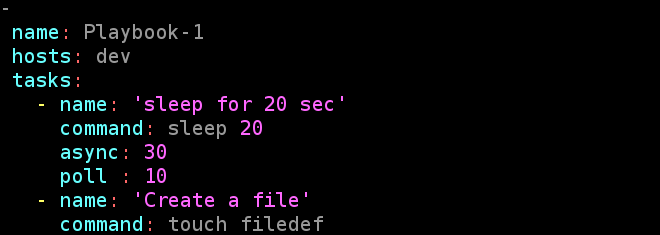


ANSIBLE NEW:

ASYNCHRONOUS ACTIONS & POLLING

* Asynchronous actions and polling ¶ By default Ansible runs tasks synchronously, holding the connection to the remote node open until the action is completed. This means within a playbook, each task blocks the next task by default, meaning subsequent tasks will not run until the current task completes. This behavior can create challenges.



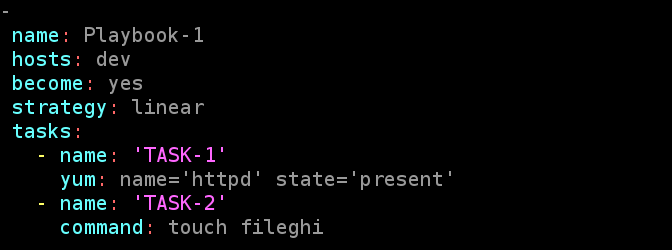


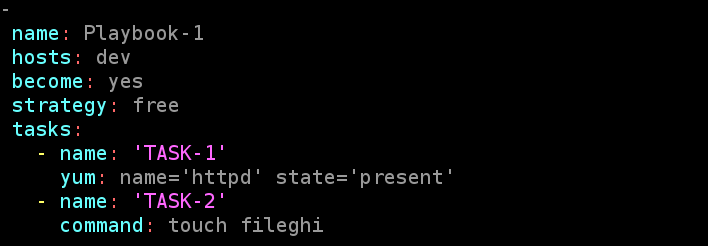
ANSIBLE STRATEGIES:

If we have two servers then we have two tasks then it will execute the first task on both the servers and after that it will go to the second task on both servers.

Install httpd on server two only and run playbook

Here Tasks are dependent on each other





JINJA2:

Jinja-2 is a template kind of language for having a customized output

If you want to print a message along with your hostname & Ip of you manage host

Now each hostname and Ip will differ for each

But we can do it by jinja2

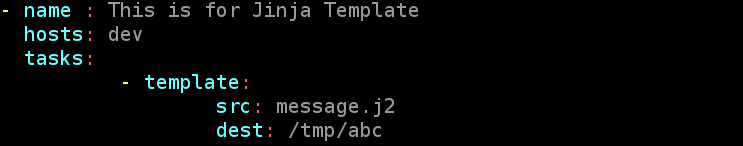
ansible -m setup localhost : Gives you ansible facts

Take any of the variable there ex: Ansible OS-FAMILY

Vi message.j2



Vi jinja2.yml

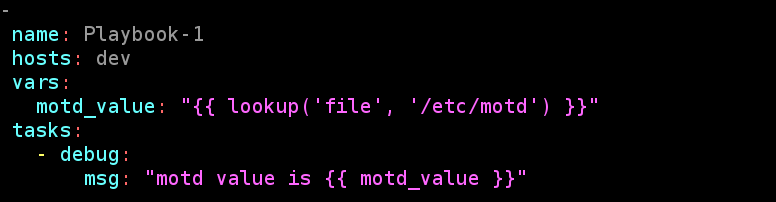


After executing check the output on the nodes that the destination you provided on playbook.

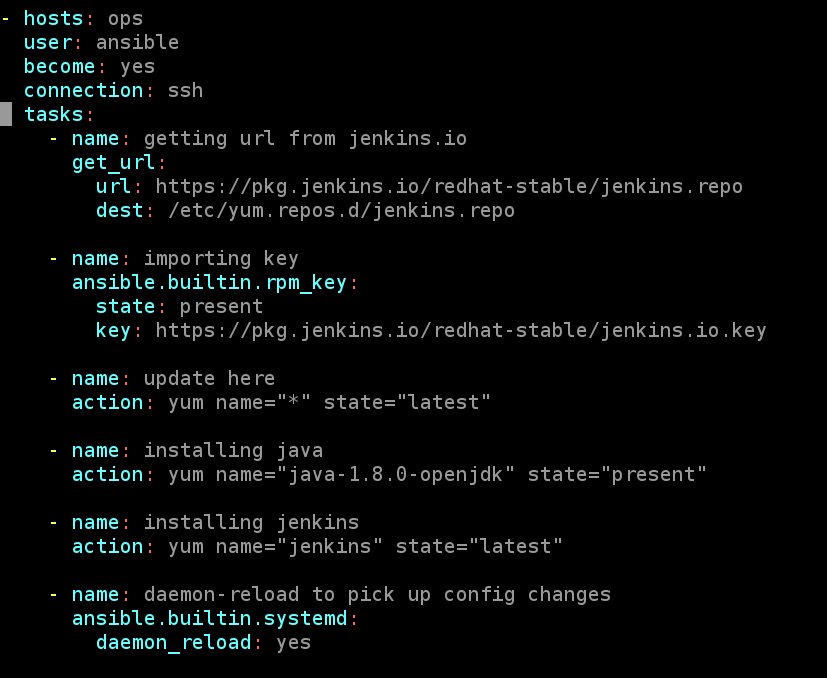
ANSIBLE LOOKUPS:

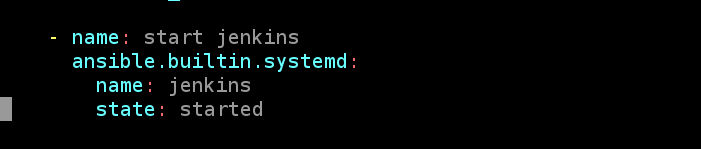
Lookup plugins retrieve data from outside sources such as files, databases, key/value stores, APIs, and other services. Like all templating, lookups execute and are evaluated on the Ansible control machine

The /etc/motd is a file on Unix-like systems that contains a "message of the day", used to send a common message to all users in a more efficient manner than sending them all an e-mail message.



JENKINS SET-UP USING YAML





PRE-REQUESITES

2 INSTANCES

* 1 – ANSIBLE SERVER
* 2 – SLAVE WITH 8080 PORT NUMBER

DON’T RUN THE CODE DIRECTLY, YOU NEED TO GO TO JENKINS SERVER AND RUN THE TASH

ssh ansible@private\_ip

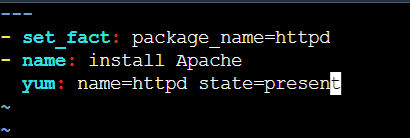
exit

NOW RUN THE CODE ON ANSIBLE SERVER

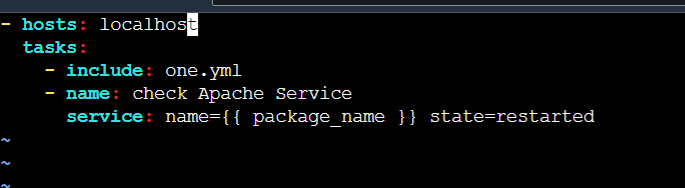
**ADVANCE TOPICS:**

Define variables in two different files:

One.yml

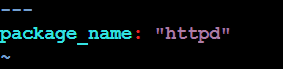


Two.yml

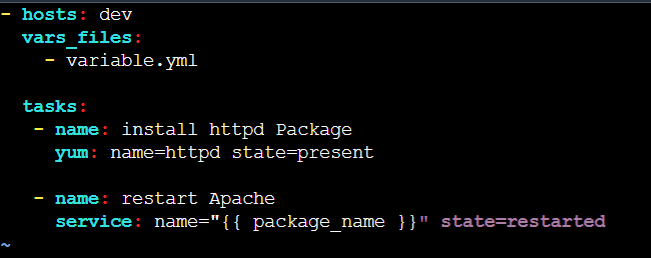


Ansible-playbook two.yml

Variable.yml



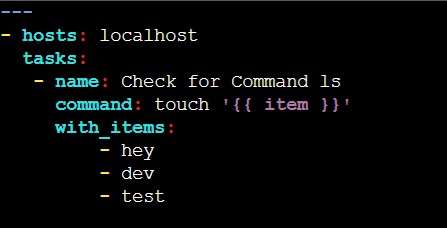
One.yml



**PLAYBOOKS WITH DIFFERENT MODULES:**

**Command** – this module allows us to execute commands on servers.

Ex: create multiple files



**Raw** – This module is used when there is more need than the Command module or if the command module does not support the operation. This module makes a SSH to the remote machine and run the command.

