ANSIBLE(core 2.15.3)

|  |  |  |
| --- | --- | --- |
| **Ad-hoc commands (/usr/bin/ansible)**  Ad-hoc commands are quick and easy but it cannot be re-used.  $ ansible [pattern] –m [module] –a [module options]  Ex:  - ansible all –m ping webserver  - ansible –m commad –a uptime/date webser  - ansible –m user –a “name=xyz password=abc” webserver **MODULES**  - ping - user - command - setup - yum - stat  **PING**: use to ping the worker m/c  Usage: ansible all –m ping #it will ping all the m/c which is mentioned in hosts file  **YUM**: use to install the software  Usage: ansible all –m yum -a “name=git” –b #-b become root user  **USER:** use to create a user  Usage: ansible all –m user –a “name=user1” -b | **Ansible Terminology**  -> Modules  -> Tasks  -> Playbook  -> Master/Controller node–where ansible is installed  -> Worker / managed node  -> Inventory / host file – which contains the ip address of all worker m/c | |
| **COMMAND:** use to run the command  Usage:  ansible all –m command –a “uptime/date/who” ansible all –m command –a “deluser=user1” –b  **SETUP:** this command will give the entire information about the system  Usage: ansible all –m setup  Ansible –m setup <prod> >>one.txt  **STAT:** this command will shows whether the file exits or not  Usage: ansible all –m stat –a “path =/home/ansadmin/1.txt” | |
| **Setup SSH** -> setup 3 machines(1-master, 2-worker)  -> install Ansible in master  -> create same username and pwd in all the 3 m/c  $ useradd ansadmin [for Ubuntu system, to add the user we have to use $ useradd -m -d /home/ansadmin ansadmin]  $ passwd ansadmin  -> add the user to /etc/sudoers file in all the 3 m/c  - ansadmin ALL=NOPASSWD: ALL  For Ubuntu m/c -> visudo and add  - ansadmin ALL=(ALL) NOPASSWD: ALL  -> enable ‘password based login to ‘yes’ in all the 3 m/c  -$ vi /etc/ssh/sshd\_config  - PasswordAuthentication yes (by default its set to NO)  -> restart the sshd in all the machines  $ sudo systemctl restart sshd  For Ubuntu m/c -> service sshd reload  -> in master m/c, generate ssh key  $ ssh-keygen  Press enter for all the options to generate a key  -> in master m/c, create ‘hosts’ file in ‘/etc/ansible’ and add the private IP address of worker m/c  -> in master m/c, login as ‘ansadmin’  $ sudo su – ansadmin  -> copy the generated ssh key to worker node  $ ssh-copy-id ansadmin@<private ip address of the m/c>  -> To connect to the worker m/c  $ ssh <userid>@<private ip address of the m/c>  -> we will be connecting to /home/ansadmin folder  -> $exit, will return to the master m/c | | **Host file**  - Default location - /etc/ansible/hosts  - if the host file is in other path then, use –i option to define the host  Ex: ansible –i /home/tmp/my\_hosts  - its de  **Host file:**  [web server]  192.168.10.20  [qa server]  192.168.100.200  [preprod]  192.168.25.50 |
| **Modules**  Ansible modules are reusable, standalone script that ansible runs on your behalf either by locally or remotely. It perform the following tasks: - creating users  - Installing packages  - updating configurations  - spinning up instances etc., |

|  |  |
| --- | --- |
| **COMMANDS**  - ansible-playbook <playbook name> --syntax-check  This command will check the syntax of the playbook is correct or not  - ansible-playbook <playbook name> --check  This will check whether play book execute properly or not | **Playbooks**  - It’s a set of instructions(plays) that you run on a single target or group of targets(hosts)  - Playbook is a text file written in YAML format and saved as \*.yml format  - It start with - - -  - An item in a YAML is starts with a single dash (-) followed by a space  - Hosts and Tasks are mandatory items in a playbook  - The playbook primarily uses indentation with space characters to indicate the structure of its data  - Modules are used to perform tasks  - Comment start with # |
|  **user**: use to create the user  name=guru   **yum**: use to install the software/’s on unix  name=git or [‘git’, ‘tree’, ‘telnet’,’gzip’]  state=installed (or) latest  stage=absent(to remove)   **apt**: use to install the software on Ubuntu  Ex: apt install git / apt install apache2  name=git  state=present   **file**: use to create/delete file/folder  path=c:\one.txt,  state=touch – to create a file  directory – to create a folder  absent – to delete a file   **copy**: use to copy the file  src: path of the source folder in master node  dest: path of the destination folder in worker node  mode: ‘0600’  owner: ram (user should be present)   **service**: this will use to start the service(httpd-apache service)  name: httpd  state: started   **git**: use to clone any repo form github  repo: https://github.com/sgpravin-GitHub/git\_practice.git  dest: <local drive path: c:\git\test\ | Gathering Facts: its similar to ad-hoc ‘setup’ command in which it gives all the information about the system.  PATH Variable in Unix: https://www.cs.purdue.edu/homes/bb/cs348/www-S08/unix\_path.html |
| **Notify & Handlers:** - Notify and handlers name should be same - if Apache is not installed, then it won’t start the apache service.  ---  - name: notify and handlers playbook  become: true  hosts: all  gather\_facts: no  gather\_facts: no #it will disable the gather facts  tasks:  - name: this will install httpd  yum:  name: httpd  state: installed  **notify: start httpd #both name should be same**  **handlers:**  **- name: start httpd**  service:  name: httpd  state: started  **Note: keep Handles at the last of the file else it won’t execute other commands which come below that.** |
| **When Condition – create a single file to install httpd on Linux & RHEL server(red hat Linux)**  when: ansible\_os\_family==’RedHat’ (for Linux server)  when: ansible\_os\_family==’Debian’ (for RHEL server)  **hosts**: prod, qa, pre-prod  **Modes**: <owner==u><group owner==g><others==o>, a=all  read®=4, write(w)=2, execute(x)=1  chmod 750 <filename>  chmod o+w <filename> | **Variables:**  *We can define variable in following ways:*   1. Define with in playbook   vars:  user: modi  user:  name: “{{ user }}”   1. Passing from external file   vars\_files:  user\_name.yml  user:  name: “{{ user }}”  *user\_name.yml file user: testuser* |

|  |  |
| --- | --- |
| **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 is a two-step process:   1. Add tags to your tasks, either individually or with tag inheritance from a block, play, role, or import. 2. Select or skip tags when you run your playbook.   **Ex:** ansible-playbook -i hosts when\_condition\_with\_tags.yml --tags "copy file" ,”new\_tag”, “another\_tag” | 1. Passing from host file (inventory file) 2. Passing while running the playbook(this will be having the highest priority) ansible-playbook –i hosts create-user.yml –e “user=shreya”   *Note: instead of –e we can specify - - extra-vars ansible-playbook –i hosts create-user.yml –extra-vars “user=shreya001”*   1. Using group-vars or hosts-vars. |
| **Ignore Errors:**  By making this flag as true, ansible allows a playbook to continue executing even if a task fails.  Ignore\_errors: true, but default its set to FALSE | **ANSIBLE VAULT**  Ansible Vault is a feature of ansible that allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plaintext in playbooks or roles. All commands will start with **‘ansible-vault’**  **Create**: to create ansible vault file in the encrypted format  **View**: to view data of encrypted file  **edit**: to edit encrypted file  **encrypt**: to encrypt an unencrypted file  **decrypt**: to decrypt an encrypted file  **--ask-vault-pass**: to provide password while running playbook  **--vault-password-file**: to pass a vault password through a file. |
| **ANSIBLE ROLES**  Creating Reusable Playbooks   1. Including and Importing 2. Roles   **Roles**: Ansible Roles provide a well-defined framework and structure for setting your tasks, variables, handlers, metadata, templates, and other files. They enable us to reuse and share our Ansible code efficiently.  **To setup a role** ansible-galaxy init setup-apache | **Insert the role in playbook file (roles.yml):**  ---  - name: this play book install httpd using ansible role  hosts: all  become: true  roles:  - setup-apache  **To run a ansible-role** ansible-playbook –I hosts roles.yml |